

Dye Advection Without the Blur: A Level-Set Approach for Texture-Based Visualization of Unsteady Flow

D. Weiskopf

Institute of Visualization and Interactive Systems
University of Stuttgart

Abstract

Dye advection is an intuitive and versatile technique to visualize both steady and unsteady flow. Dye can be easily combined with noise-based dense vector field representations and is an important element in user-centric visual exploration processes. However, fast texture-based implementations of dye advection rely on linear interpolation operations that lead to severe diffusion artifacts. In this paper, a novel approach for dye advection is proposed to avoid this blurring and to achieve long and clearly defined streaklines or extended streak-like patterns. The interface between dye and background is modeled as a level-set within a signed distance field. The level-set evolution is governed by the underlying flow field and is computed by a semi-Lagrangian method. A reinitialization technique is used to counteract the distortions introduced by the level-set evolution and to maintain a level-set function that represents a local distance field. This approach works for 2D and 3D flow fields alike. It is demonstrated how the texture-based level-set representation lends itself to an efficient GPU implementation and therefore facilitates interactive visualization.

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

1. Introduction

Vector field visualization plays an important role in various scientific and engineering disciplines. Typical applications stem from simulations in computational fluid dynamics, calculation of physical vector fields, such as electromagnetic fields or heat flow, or from measurements of actual wind or fluid flows. Dye advection is an intuitive and versatile technique to visualize both steady and unsteady vector fields and it can readily be combined with noise-based dense representations. Dye is released at user-specified positions and therefore supports an interactive and user-centric visual exploration process. At the same time, a dense representation can provide additional information on the overall flow behavior and serve as a background context in which the user can focus on details of the dye-based visualization.

The goal of this paper is to provide a fast and high-quality dye advection scheme that can be used in conjunction with texture-based dense flow visualization techniques. Real-time visualization is particularly important in this context because dye advection makes sense only in an interactive environ-

ment in which seed points can be freely chosen. Unfortunately, previous implementations of texture-based dye advection rely on linear interpolation operations and are therefore subject to diffusion artifacts. A novel approach for dye advection is proposed to avoid this blurring and to achieve long and clearly defined streaklines. The main idea is to model the interface between dye and background as a level-set within a signed distance field. The level-set evolution is governed by the flow field that has to be visualized. A reinitialization technique is used to counteract the negative effects of this evolution and to maintain a level-set function that represents a distance field. It is demonstrated that this approach is suitable for 2D and 3D flow fields. The level-set and the corresponding numerical operations can be directly mapped to a GPU implementation. An efficient, purely GPU-based approach is described for the 2D case.

2. Previous Work

Dye advection is typically applied in combination with texture-based flow visualization techniques. The same

mechanisms can often be used to realize both dye and noise transport. An important example for this approach is texture advection [MB95], which can be used for a dense, noise-based and a sparse, dye-based representation at the same time. For example, Image Based Flow Visualization [vW02] is a 2D texture advection technique that supports dye advection. Similarly, the 2D Lagrangian-Eulerian Advection (LEA) scheme [JEH02] allows for simultaneous noise and dye transport. LEA addresses the issue of numerical diffusion in dye advection by constantly restoring the contrast of the transported dye. Within the context of Line Integral Convolution (LIC) [CL93], dye visualization can be used to highlight features [SJM96]. This idea can be extended to transport several colors simultaneously [UIM*03].

One reason for recent advances in texture-based flow visualization is the increasing performance and functionality of GPUs. Since dense texture representations need a large number of computations, graphics hardware can be exploited to improve the performance of 2D texture-based flow visualization [HWSE99, JEH00, WHE01, WEE03, vW02]. Texture advection can be extended to vector fields on curved surfaces [LJH03, vW03] and in 3D [TvW03, WE04]. The Chameleon system [LBS03] allows the user to interactively change the rendering style for 3D flow visualization by using graphics hardware. An overview on the state of the art in dense and texture-based vector field visualization techniques can be found in [LHD*04, SMM00].

Another related line of research is focused on level-set techniques for visualization, image processing, or simulation. An overview on level-sets and their applications can be found in [OF03]. For example, level-sets are used to model the interface between fluid and air in the simulation of the dynamics of water [FF01, EMF02]. Applications of level-sets in visualization include surface processing [TWB03], the segmentation of volumetric data [LKH03, MBZW02], and an implicit surface representation in a flow [WJE00].

Tracking of interfaces plays an important role in computational fluid dynamics and computational physics. In this paper, only a brief list of related publications in this field can be given. One class of numerical methods makes use of a Lagrangian tracking of particles, as, e.g., described in [UT92]. Volume-of-fluid (VOF) techniques [HN81] model the fraction of a material in each computational cell to reconstruct the material interface; examples are the SLIC method (Simple Line Interface Calculation) [NW76], the PLIC technique (Piecewise Linear Interface Construction) [You82], or a cubic-polynomial interpolation to represent the VOF fraction [YIW*91]. Another approach is the ghost fluid method [FAMO99], in which a level-set function is used to track the motion of a material interface.

3. Dye Advection by Texture-Based Material Transport

The traditional method for texture-based dye advection is briefly reviewed. Properties of the dye, such as its color, are

stored on a regularly sampled grid or texture. This property field is denoted by $p(\mathbf{x})$. The points \mathbf{x} are from the domain of the n D vector field, \mathbb{R}^n . In this Eulerian approach, the position of a dye element is implicitly given by the location of the corresponding texel. Dye is transported along the time-dependent vector field $\mathbf{v}(\mathbf{x}, t)$. The Lagrangian formulation of the underlying equation of motion,

$$\mathbf{v}(\mathbf{x}(t), t) = \frac{d\mathbf{x}(t)}{dt} \quad ,$$

can be integrated to compute the pathline of an advected massless particle,

$$\mathbf{x}(t_1) = \mathbf{x}(t_0) + \int_{t_0}^{t_1} \mathbf{v}(\mathbf{x}(t), t) dt \quad . \quad (1)$$

In the above equations, t denotes time. Equation (1) can be used to compute the transport of dye in a semi-Lagrangian manner (see, e.g., [Sta99, JEH00]). One possible alternative for dye advection is realized by a backward texture-lookup in the property field,

$$p(\mathbf{x}(t_0), t_0) = p(\mathbf{x}(t_0 - \Delta t), t_0 - \Delta t) \quad . \quad (2)$$

Starting from the current time step t_0 , an integration backwards in time provides the position at the previous time step, $\mathbf{x}(t_0 - \Delta t)$. The property field is evaluated at this previous position to access the value that is transported to the current position. Bilinear (in 2D) or trilinear (in 3D) interpolation is applied to reconstruct the property field at locations different from grid points. This method can be considered as a semi-Lagrangian approach because property is represented in a Eulerian way on a texture, while the temporal evolution relies on a Lagrangian integration of particle paths.

Similarly to Equation (2), an integration forward in time leads to a forward mapping in which property values are actively scattered from their current location to a new position. In an analytic description, backward and forward mapping are equivalent. Slight differences may appear when a numerical approximation for path integration is employed. Both alternatives are used in previous work, e.g., forward mapping occurs in [vW02] and backward mapping is employed in [JEH00, WHE01, vW03].

Semi-Lagrangian texture-based dye advection is widely used because it has a number of important advantages. It is easy to implement and very efficient, and it can be directly mapped to GPUs and combined with other interactive texture-based GPU techniques. In particular, texture advection can be simultaneously applied to dye and noise textures, saving computational costs for separate transport mechanisms. Another advantage is the flexibility of a texture representation. For example, dye can be virtually painted into the flow by the user—without any restriction with respect to orientation, size, or topology. Finally, the texture approach does not need mechanisms for particle removal (in convergent flow regions) or particle insertion (in divergent flow regions), as it is required in purely particle-based Lagrangian methods.

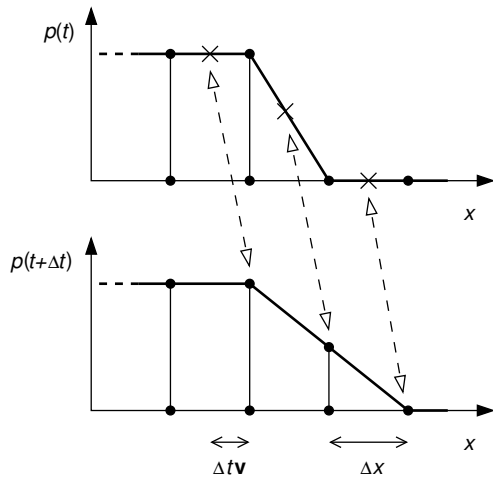


Figure 1: Numerical diffusion caused by linear interpolation.

Unfortunately, texture-based dye advection is subject to numerical diffusion that leads to an inappropriate blurring of the visualization. The underlying problem is the continuous resampling with a tensor product of linear filters (i.e., bilinear interpolation in 2D and trilinear interpolation in 3D). Figure 1 illustrates how linear interpolation leads to artificial diffusion in a simple 1D example. The top image shows the property values p for the time step t . The dye is represented by large values p , the background by vanishing p . The bottom image shows the property values for the following time step $t + \Delta t$. A uniform vector field is applied to transport the dye to the right. In this example, the step size Δt and the velocity \mathbf{v} are chosen in a way that the dye is shifted by one half of the grid spacing Δx , i.e., $\Delta t \mathbf{v} = \Delta x / 2$. Ideally, the profile for the values p should not be changed by such a translation. However, the access to positions in between grid points leads to a dilution of dye by linear interpolation. As a result, the width of the interface between dye and background is increased from one grid spacing to two grid spacings. Subsequent advection steps will continuously increase the size of this interface layer. Figure 2 shows an example of numerical diffusion within a 2D circular flow.

This blurring effect is not related to physical diffusion that may occur in experimental flow visualization. In fact, the smearing is only caused by the advection algorithm and depends on many different parameters, such as step size, velocity, grid resolution, and the orientation with respect to the main axes of the texture. Furthermore, the blurring prevents users from clearly identifying streaklines or other isolated visual patterns. In summary, this artificial diffusion should be avoided for an effective vector field visualization.

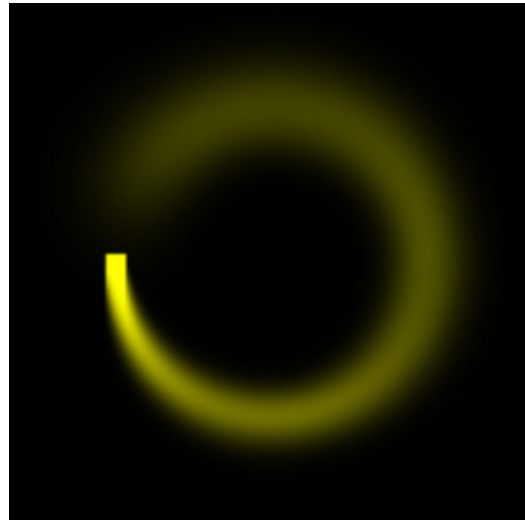


Figure 2: Numerical diffusion in a circular flow.

4. Level-Set Techniques for Tracking Dye Interfaces

A level-set approach can be used to overcome the above mentioned problems in dye advection. Level-set techniques are employed to track interfaces in many other applications, e.g., the interface between fluid and air can be modeled by level-sets in a water simulation [FF01, EMF02]. The goal of this paper is to provide a dye advection method that is able to accurately track the boundary between dye and background, while it still retains the aforementioned advantages of semi-Lagrangian property advection.

Distance Field as Level-Set

In the level-set approach of this paper, the interface between dye and background is represented as an implicit surface within a signed distance function. A signed distance function $\phi(\mathbf{x})$ describes the distance of the point \mathbf{x} to the surface. The interface is the implicit surface for $\phi(\mathbf{x}) = 0$. It is assumed that $\phi(\mathbf{x})$ is negative inside the dye region and positive on the outside. Dye is modeled as a massless marker material that is perfectly advected along the input vector field. For such a convection scenario, the evolution of the level-set is governed by

$$\frac{\partial \phi(\mathbf{x}, t)}{\partial t} + \mathbf{v}(\mathbf{x}, t) \cdot \nabla \phi(\mathbf{x}, t) = 0 \quad .$$

This partial differential equation could be discretized and solved by different Eulerian techniques [OF03]. Instead, a semi-Lagrangian approach is used because it is well-suited for such a convection process and leads to a stable evolution even for large step sizes [Sta99]. Equation (2) is now applied to $\phi(\mathbf{x}, t)$:

$$\phi(\mathbf{x}(t_0), t_0) = \phi(\mathbf{x}(t_0 - \Delta t), t_0 - \Delta t) \quad . \quad (3)$$

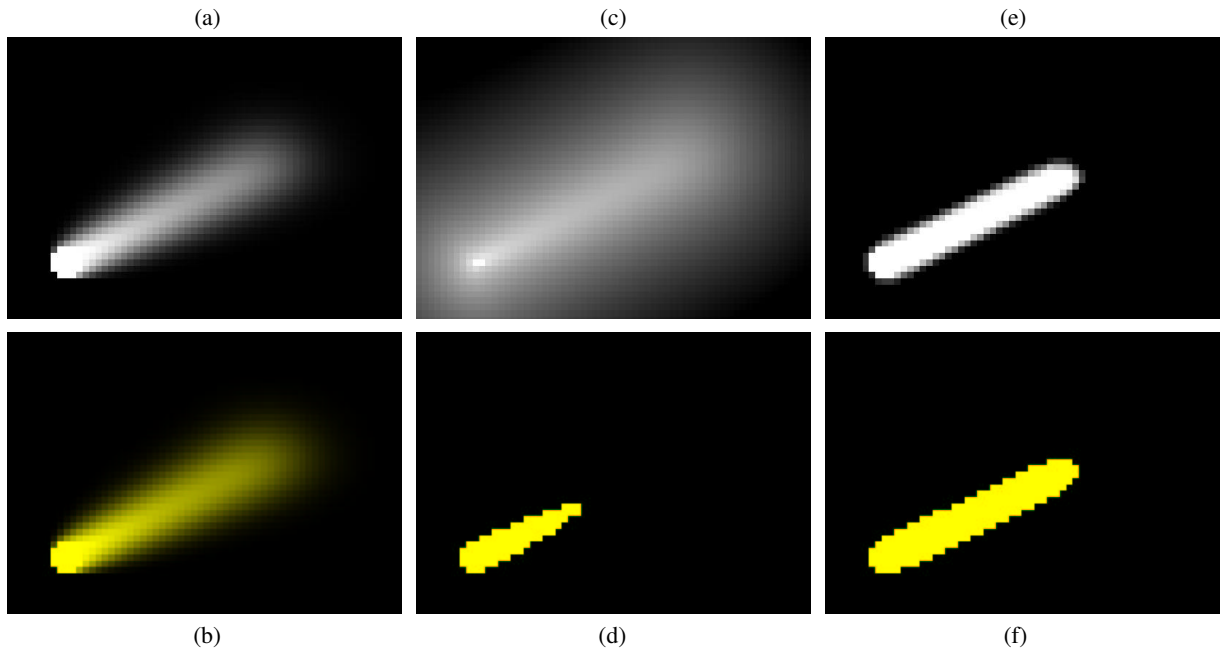


Figure 3: Comparison between traditional dye advection (a,b), the level-set approach without reinitialization (c,d), and the level-set approach with reinitialization (e,f).

According to Equation (1), the Lagrangian integration of $\mathbf{x}(t_0 - \Delta t)$ takes into account a time-dependent vector field. Therefore, the level-set advection scheme can be used for steady and unsteady flow alike.

Besides the transport of dye, the injection of additional dye has to be modeled in this level-set approach as well. The newly injected dye is represented by a signed distance field, ϕ_{inject} . After each integration step (3), the new dye and the “old”, transported dye, ϕ_{transp} , are combined according to

$$\phi(\mathbf{x}) = \begin{cases} \phi_{\text{inject}}(\mathbf{x}) & \text{if } \phi_{\text{inject}}(\mathbf{x}) < \phi_{\text{transp}}(\mathbf{x}) \\ \phi_{\text{transp}}(\mathbf{x}) & \text{otherwise} \end{cases}.$$

In this way, the signed distance is modified by the injection field at positions that are closer to the new dye region, compared to the original distance from the interface.

The distance field itself cannot be directly used for the final visualization. In fact, dye is drawn where the signed distance is negative and the background remains visible where the distance is positive. Figures 3 (c) and (d) demonstrate level-set dye advection. Dye is injected in the lower left part of the domain and transported by a uniform vector field that points towards the upper right of the domain. Image (c) visualizes the distance field: Bright regions correspond to negative distances, while dark regions correspond to positive distances. This illustration uses a nonlinear gray-scale mapping to cover all relevant distance values. Figure 3 (d) shows the

final visualization of the dye, which is displayed in yellow color.

For comparison, the traditional advection method is included in Figures 3 (a) and (b). Image (a) shows a gray-scale table applied to the property values, image (b) displays the same values by a color table that resembles the color coding from Figure 3 (d). As discussed previously, traditional dye advection leads to a blurring and widening of the streak-line. In contrast, the level-set approach in Figure 3 (d) exhibits a sharp interface boundary. Unfortunately, the streak-line shrinks and eventually disappears completely. The reason for this behavior can be seen in the underlying distance field (Figure 3 (c)). The semi-Lagrangian level-set evolution builds upon a bilinear resampling in the distance field and therefore is affected by interpolation artifacts that are similar to those in traditional dye advection. Here, positive distance values surround the rather small dye region and dilute the negative distances inside the dye. Therefore, this simple distance-based scheme is not appropriate for dye advection.

Reinitialization of the Level-Set

The underlying problem is that, as the level-set evolves, ϕ drifts away from its initialized value as signed distance. This issue is apparent in other level-set applications as well [OF03]. Therefore, in general, the level-set needs to be reinitialized periodically to a signed distance. Fortunately, a complete reconstruction of the signed distance field is not

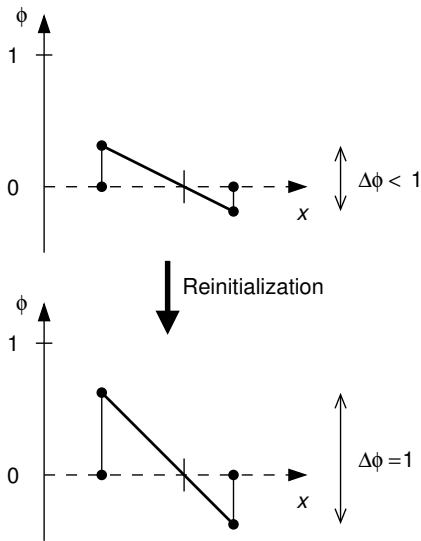


Figure 4: Reinitialization of $\Delta\phi$ back to unit distance along the intersection edge.

required for dye advection. In fact, only a local level-set around the interface has to be modeled as signed distance. Unlike many other applications of level-set techniques, we do not need a higher-order representation of the level-set function and can therefore restrict ourselves to reinitialize only those grid points that are adjacent to the interface. In what follows, only a local “distance field” is used, where distance values are clamped to the range $[-1, 1]$. The spacing of the uniform grid is assumed to have unit length.

Figure 4 illustrates the initialization method in 1D. First, the edges between grid points are classified as either intersection or non-intersection edges. Due to the linear interpolation between grid points, an intersection with the interface is uniquely identified by a sign change of ϕ between two adjacent grid points. Grid points that are not adjacent to the interface are reset to $+1$ or -1 , depending on the sign of ϕ . Intersection edges, however, trigger a reinitialization to a signed distance. Here, the values at the two adjacent grid points are modified under the constraint that the position of the interface is not changed. Using indices $i = 1, 2$ for these two grid points, the reinitialized values are

$$\phi_i^r = \frac{\phi_i}{\|\phi_1\| + \|\phi_2\|},$$

where ϕ_i are the values before reinitialization. For very small dye regions, the interface can cross two adjacent edges. Therefore, the reinitialization method is extended to take into account the case that a grid point is attached to more than a single intersection edge. Here, the final distance is set to the average of the values from the reinitialization along the two edges.

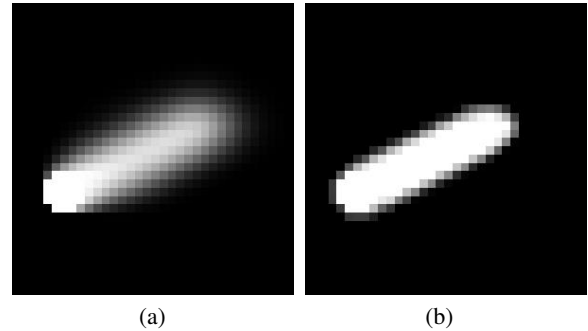


Figure 5: Local distance field reinitialization in 2D. Image (a) shows the local distance field before reinitialization, (b) directly after reinitialization.

This averaging method directly leads to the extension of the reinitialization technique to more than one dimension. In nD , a grid point is attached to $2n$ edges. Once again, the edges are classified as intersection or non-intersection edges. For each intersection edge, the adjacent grid points receive a preliminary modified value ϕ_i^r that is calculated along this edge. The final distance value is computed by taking the average of all these contributions from adjacent edges. If a grid point is not located next to the interface, its value is reset to $+1$ or -1 . In any dimension n , reinitialization only needs access to grid elements in the direct neighborhood, which leads to a fast computation of the updated level-set function.

Figures 3 (e) and (f) show the result of this reinitialization process for the same test scenario as in Figures 3 (a)–(d). Here, the reinitialization is applied after every tenth step of the level-set evolution (Equation (3)). The reinitialization guarantees that the level-set is not diluted by surrounding positive distance values. Therefore, a constant width of the streakline is maintained. At the same time, a clearly defined interface between dye and background is achieved. Figure 5 demonstrates the effect of reinitialization in more detail. The left image represents the local distance field before reinitialization. The right picture shows the local distance field directly after the reinitialization, which clearly retains the position of the interface. The blurring has been removed and a thin boundary layer between dye and background has been restored.

Figure 6 shows the visualization of a 3D vector field with a swirling convergent feature. The two images depict the same time step from different camera positions. Level-set dye advection was used with a reinitialization after each fifth integration step. The resolution of the domain is 128^3 .

Discussion

All texture-based advection techniques have the advantage that the regions covered by dye can change their topology while the dye evolves. A typical example is a streakline that

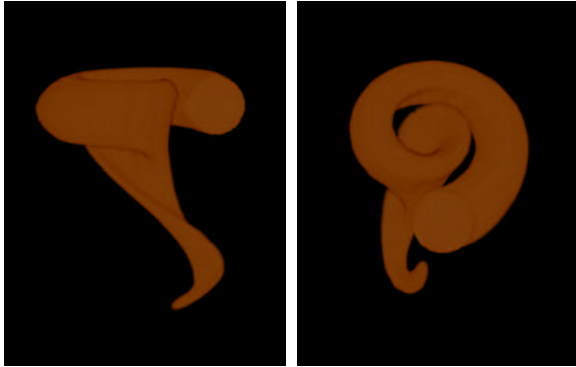


Figure 6: Level-set dye advection for a 3D vector field with a swirling feature.

is transported across the seed point of another streakline. Figure 7 illustrates this behavior as generated by the level-set approach with reinitialization. In Figure 7 (a), the two streaklines are still separated. Figure 7 (b) shows the visualization just after the two streaklines have attached, and Figure 7 (c) at a later time step. Such a merger of different dye structures would require complex algorithms and data structures if a mesh-based representation with purely Lagrangian transport was used.

Another advantage of texture-based advection is its built-in support for transporting extended regions of dye. The visualization performance is independent of the number of cells covered by dye. Therefore, a user interface is possible that allows the user to control dye injection by virtually painting into the flow [WEHE02]. This interaction approach is more difficult to implement for particle-based Lagrangian tracking. To maintain a dense and uniform coverage of the domain in regions of divergent or convergent flow, data structures have to be implemented that allow for dynamic insertion and deletion of particles in the Lagrangian approach [SK98]. Alternatively, a large number of particles can be traced to achieve a minimum density of particles in divergent flow areas, which decreases the efficiency of the implementation [dLvL00].

The issue of numerical diffusion is also addressed by Jobard et al. [JEH02] in the context of traditional texture-based dye advection. They use a method that re-adjusts the contrast of the dye density values after each advection step. To achieve an appropriate sharpening of dye streaks, the parameters of the contrast-enhancing mapping have to be chosen correctly. If the mapping function is too steep, contrast is increased too much and the visualization may appear “frozen”, similarly to the effect of nearest-neighbor sampling in LEA [JEH02]. On the other hand, dye will smear out if the mapping function is too flat. In contrast, level-set reinitialization is independent of the rate of artificial diffusion and does not require a subtle choice of parameters.

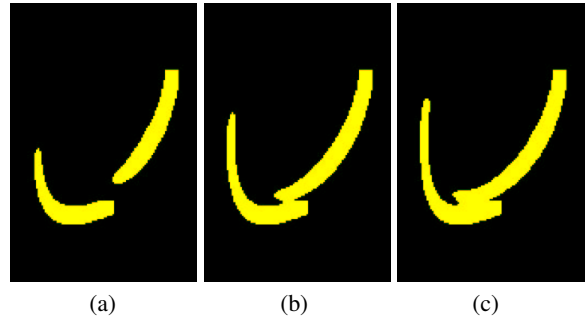


Figure 7: Change of topology of the dye-covered area. The images show snapshots from an animated visualization, chronologically ordered from left to right.

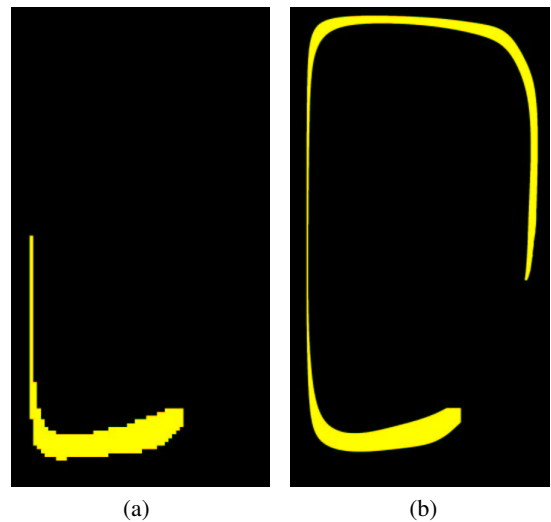


Figure 8: Truncation of streaklines in low-resolution level-set advection (a). Image (b) represents the correct streakline generated at high resolution.

Reinitialization results in a boundary layer that has a width of one or two texels everywhere, regardless of the size of the smeared-out zone before reinitialization (see Figure 5).

A disadvantage of level-set advection with reinitialization is a failure of transporting small structures that are in the range of only one cell. Figure 8 (a) shows a streakline that incorrectly ends after being “compressed” in a convergent region. For comparison, Figure 8 (b) represents the correct streakline as generated at high resolution. In general, the level-set approach with reinitialization has a slight tendency towards shrinking the advected dye structures.

5. Extensions

The visual representation of the distance fields from level-set dye advection can be improved by a number of extensions.

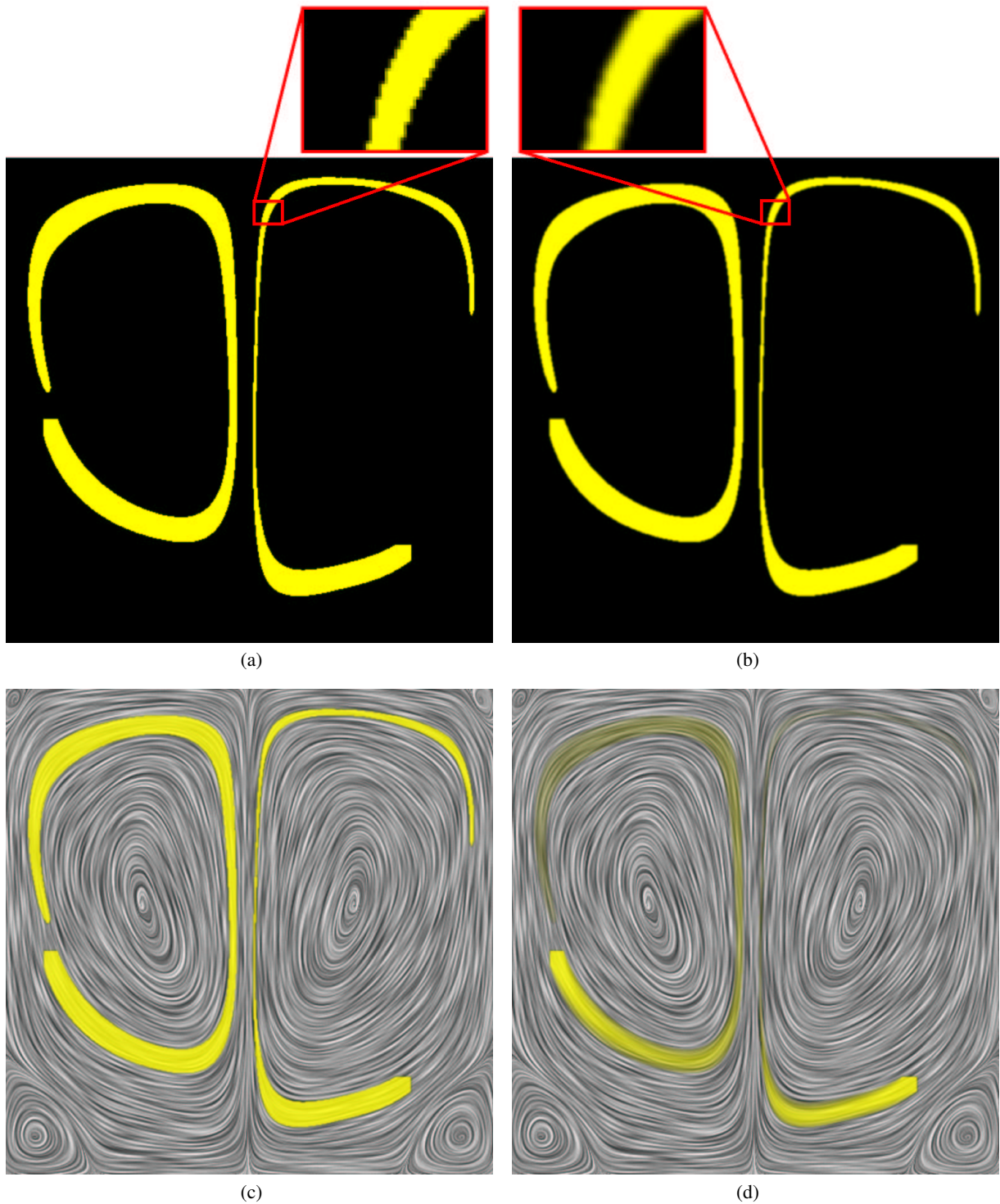


Figure 9: Extensions for level-set dye advection. Image (a) shows a jaggy interface caused by a binary decision with respect to the isovalue $\phi = 0$. Jaggy artifacts are smeared out by image-space smoothing in (b). A background LIC image is included in (c). Dye is faded out in (d).

The first extension addresses the final display of distance-based dye. Since dye regions are extracted with a sharp threshold at isovalue 0, the underlying grid structure will become apparent in the form of jaggies, as shown in Figure 9 (a). (Note that the final display has a higher resolution than the computational grid; bilinear interpolation for texture fetches leads to a slight blurring of the dye boundary.) In Figure 9 (b), jaggy artifacts are reduced by image-space smoothing. For example, a Gaussian filter or a box filter can be employed for smoothing. This filtering by convolution can be efficiently handled by GPU operations working in image-space [Mit02].

Like in previous approaches with traditional dye advection, the dye visualization can be combined with a dense, noise-based vector field visualization. As an example, dye is modulated by a background LIC image in Figure 9 (c).

As another extension, the “age” of dye can be transported along the flow to gradually fade out dye structures. Age is stored in another property texture and advected in a semi-Lagrangian manner. The measure for age has values between 1 (“just born”) and 0 (“infinitely old”). Newly injected dye carries an age measure of 1. In each advection step, the age of the existing dye elements is decreased according to a user-specified factor. The original color from the dye extraction process is modulated by this age measure to provide the final image, as shown in Figure 9 (d).

6. Implementation and Results

Both the 2D and 3D level-set dye advection techniques have been implemented in a C++ program. The grids for the distance field are represented by arrays onto which the methods from Sections 4 and 5 can be directly mapped. The images for Figures 3, 5, and 6 were generated by this CPU-based implementation.

To improve rendering performance and achieve a real-time visualization, 2D dye advection has been additionally realized on a GPU. This GPU version is based on DirectX 9.0 and was developed and tested on a Windows XP PC with ATI Radeon 9800 XT (256 MB) GPU and Pentium 4 (3.2 GHz) CPU. 2D advection has been implemented first because an efficient render-to-texture functionality is only supported by 2D textures so far. However, there should not be any problems in realizing a corresponding 3D version, as will be discussed shortly. For example, the ARB Superbuffer extension [Per03] could provide a mechanism to directly render into 3D textures, which would lead to a comparable implementation in 3D. The images for Figures 2 and 7–9 were generated by the GPU-based 2D advection tool. In the following, details of this implementation are presented.

The local distance field is represented by a 2D texture with a single 16 bit fixed-point channel. The vector field is also stored in a 2D texture. The current implementation represents the vector field in two 16 bit fixed-point channels. A

32 bit floating-point texture could be used if a higher accuracy is required. The semi-Lagrangian evolution of the distance field is realized by a pixel shader program that computes the backward mapping according to Equation (3). The corresponding fragments are generated by rendering a single viewport-filling quadrilateral. First, the pixel shader accesses the vector field to compute a short particle path according to Equation (1). An explicit second-order midpoint scheme is used because the accuracy of first-order Euler integration is inappropriate for tracing long streaklines. Afterwards, the computed particle position is used to access the distance field from the previous time step. The required bilinear interpolation is directly supported by 16 bit fixed-point textures. Actually, two copies of the distance texture are held on the GPU—one for the previous time step and the other one for the field that has to be updated in the current time step. After each iteration, both textures are swapped in a ping-pong scheme.

Periodically, the level-set texture has to be reinitialized by another pixel shader program. This pixel shader needs the values from the center point and from the adjacent $2n$ grid points along the main axes in nD , i.e., 5 texels have to be loaded in the 2D case. The grid points are accessed by nearest-neighbor sampling in the level-set texture. The corresponding texture coordinates are generated by a vertex shader program. In this way, dependent texture lookups (texture indirections) are avoided. The method from Section 4 computes preliminary reinitialization values for each edge. Since this computation has the same structure for all edges, the four-vector capabilities of the GPU can be exploited to process all four edges simultaneously and thus greatly reduce the number of instructions. In total, the implementation needs 23 numerical instruction slots and 5 slots for texture loading. In the 3D case, the number of adjacent edges increases from four to six. Therefore, 7 texture accesses would be needed. As a rough estimate, the number of numerical instructions would be increased by a factor of two. Both numbers are still well within the limits of current GPUs like the ATI Radeon 9800 or NVidia GeForce FX.

Table 1 documents the rendering performance for 2D GPU-based dye advection on the aforementioned hardware configuration. The viewport sizes were 800×600 and 1200×900 pixels, respectively. The first line shows the performance of the semi-Lagrangian advection scheme. These numbers apply to the traditional property-based dye advec-

Table 1: Rendering performance in fps.

Viewport	800×600	1200×900
No reinitialization	326.8	151.1
Permanent reinitialization	152.2	68.8
Reinitialization after 10 steps	267.5	121.5

tion and the distance-based representation alike. The numbers in the second and third rows reflect the rendering speed for a combination of semi-Lagrangian advection and reinitialization. If reinitialization is applied after each single iteration step, the performance is roughly fifty percent of the semi-Lagrangian advection scheme. The third line represents a more realistic example in which the level-set is periodically reinitialized after every tenth iteration. Here, the performance is only 20 percent below pure semi-Lagrangian advection.

7. Conclusion and Future Work

A level-set approach for texture-based dye advection has been introduced to avoid the numerical diffusion that appears in traditional dye advection techniques. The method consists of two major elements: evolution of the level-set by semi-Lagrangian advection and periodic reinitialization to a signed distance field. Reinitialization only needs access to grid cells in a local neighborhood and therefore is very fast. Level-set dye advection works in 2D and 3D, can be combined with other GPU-based flow visualization techniques, and is only slightly slower than traditional dye advection. Therefore, this approach could serve as a versatile element in interactive flow visualization and be used to replace traditional dye advection modules in such visualization systems.

For future work, it would be interesting to investigate if interactive flow visualization benefits from more sophisticated and more accurate techniques for interface tracking, as known from computational fluid dynamics and computational physics.

Acknowledgments

The author would like to thank the anonymous reviewers for helpful remarks to improve the paper. Special thanks to Bettina Salzer for proof-reading. This work was supported by the Landesstiftung Baden-Württemberg.

References

- [CL93] CABRAL B., LEEDOM L. C.: Imaging vector fields using line integral convolution. In *Proceedings of ACM SIGGRAPH* (1993), pp. 263–272.
- [dLvL00] DE LEEUW W., VAN LIERE R.: Spotting structure in complex time dependent flow. In *Proceedings Dagstuhl '97 Scientific Visualization* (2000), Hagen H., Nielson G. M., Post F. (Eds.), IEEE Computer Society, pp. 47–53.
- [EMF02] ENRIGHT D. P., MARSCHNER S. R., FEDKIW R. P.: Animation and rendering of complex water surfaces. *ACM Transactions on Graphics* 21, 3 (2002), 736–744.
- [FAMO99] FEDKIW R., ASLAM T., MERRIMAN B., OSHER S.: A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method). *Journal of Computational Physics* 152, 2 (1999), 457–492.
- [FF01] FOSTER N., FEDKIW R.: Practical animation of liquids. In *Proceedings of SIGGRAPH* (2001), pp. 23–30.
- [HN81] HIRT C. W., NICHOLS B. D.: Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of Computational Physics* 39, 1 (1981), 201–225.
- [HWSE99] HEIDRICH W., WESTERMANN R., SEIDEL H.-P., ERTL T.: Applications of pixel textures in visualization and realistic image synthesis. In *ACM Symposium on Interactive 3D Graphics* (1999), pp. 127–134.
- [JEH00] JOBARD B., ERLEBACHER G., HUSSAINI M. Y.: Hardware-accelerated texture advection for unsteady flow visualization. In *IEEE Visualization* (2000), pp. 155–162.
- [JEH02] JOBARD B., ERLEBACHER G., HUSSAINI M. Y.: Lagrangian-Eulerian advection of noise and dye textures for unsteady flow visualization. *IEEE Transactions on Visualization and Computer Graphics* 8, 3 (2002), 211–222.
- [LBS03] LI G. S., BORDOLOI U., SHEN H. W.: Chameleon: An interactive texture-based framework for visualizing three-dimensional vector fields. In *IEEE Visualization* (2003), pp. 241–248.
- [LHD*04] LARAMEE R. S., HAUSER H., DOLEISCH H., VROLIJK B., POST F. H., WEISKOPF D.: The state of the art in flow visualization: Dense and texture-based techniques. *Computer Graphics Forum* 23 (2004). In print.
- [LJH03] LARAMEE R. S., JOBARD B., HAUSER H.: Image space based visualization of unsteady flow on surfaces. In *IEEE Visualization* (2003), pp. 131–138.
- [LKH03] LEFOHN A. E., KNISS J. M., HANSEN C. D., WHITAKER R. T.: Interactive deformation and visualization of level set surfaces using graphics hardware. In *IEEE Visualization* (2003), pp. 75–82.
- [MB95] MAX N., BECKER B.: Flow visualization using moving textures. In *Proceedings of the ICASW/LaRC Symposium on Visualizing Time-Varying Data* (1995), pp. 77–87.
- [MBZW02] MUSETH K., BREEN D. E., ZHUKOV L.,

- WHITAKER R. T.: Level-set segmentation from multiple non-uniform volume datasets. In *IEEE Visualization* (2002), pp. 179–194.
- [Mit02] MITCHELL J. L.: Image processing with 1.4 pixel shaders in Direct3D. In *Direct3D ShaderX: Vertex and Pixel Shader Tips and Tricks* (2002), Engel W. F. (Ed.), Wordware, pp. 258–269.
- [NW76] NOH W. F., WOODWARD P. R.: SLIC (Simple line interface calculation). In *Lecture Notes in Physics 59: Proceedings of the Fifth International Conference on Numerical Methods in Fluid Dynamics* (1976), Springer, pp. 330–340.
- [OF03] OSHER S., FEDKIW R.: *Level Set Methods and Dynamic Implicit Surfaces*. Springer, New York, 2003.
- [Per03] PERCY J.: OpenGL extensions. ATI Presentation at SIGGRAPH 2003, access via <http://www.ati.com/developer>, 2003.
- [SJM96] SHEN H.-W., JOHNSON C. R., MA K.-L.: Visualizing vector fields using line integral convolution and dye advection. In *1996 Volume Visualization Symposium* (1996), pp. 63–70.
- [SK98] SHEN H.-W., KAO D. L.: A new line integral convolution algorithm for visualizing time-varying flow fields. *IEEE Transactions on Visualization and Computer Graphics* 4, 2 (1998), 98–108.
- [SMM00] SANNA A., MONTRUCCHIO B., MONTUSCHI P.: A survey on visualization of vector fields by texture-based methods. *Recent Res. Devel. Pattern Rec. 1* (2000), 13–27.
- [Sta99] STAM J.: Stable fluids. In *Proceedings of SIGGRAPH* (1999), pp. 121–128.
- [TvW03] TELEA A., VAN WIJK J. J.: 3D IBFV: Hardware-accelerated 3D flow visualization. In *IEEE Visualization* (2003), pp. 233–240.
- [TWBO03] TASDIZEN T., WHITAKER R., BURCHARD P., OSHER S.: Geometric surface processing via normal maps. *ACM Transactions on Graphics* 22, 4 (2003), 1012–1033.
- [UIM*03] URNESS T., INTERRANTE V., MARUSIC I., LONGMIRE E., GANAPATHISUBRAMANI B.: Effectively visualizing multi-valued flow data using color and texture. In *IEEE Visualization* (2003), pp. 115–122.
- [UT92] UNVERDI S. O., TRYGGVASON G.: A front-tracking method for viscous incompressible multi-fluid flows. *Journal of Computational Physics* 100, 1 (1992), 25–37.
- [vW02] VAN WIJK J. J.: Image based flow visualization. *ACM Transactions on Graphics* 21, 3 (2002), 745–754.
- [vW03] VAN WIJK J. J.: Image based flow visualization for curved surfaces. In *IEEE Visualization* (2003), pp. 123–130.
- [WE04] WEISKOPF D., ERTL T.: GPU-based 3D texture advection for the visualization of unsteady flow fields. In *WSCG 2004 Short Communication Papers Proceedings* (2004), pp. 181–188.
- [WEE03] WEISKOPF D., ERLEBACHER G., ERTL T.: A texture-based framework for spacetime-coherent visualization of time-dependent vector fields. In *IEEE Visualization* (2003), pp. 107–114.
- [WEHE02] WEISKOPF D., ERLEBACHER G., HOPF M., ERTL T.: Hardware-accelerated Lagrangian-Eulerian texture advection for 2D flow visualization. In *Vision, Modeling, and Visualization VMV '02 Conference* (2002), pp. 77–84.
- [WHE01] WEISKOPF D., HOPF M., ERTL T.: Hardware-accelerated visualization of time-varying 2D and 3D vector fields by texture advection via programmable per-pixel operations. In *Vision, Modeling, and Visualization VMV '01 Conference* (2001), pp. 439–446.
- [WJE00] WESTERMANN R., JOHNSON C., ERTL T.: A level-set method for flow visualization. In *IEEE Visualization* (2000), pp. 147–154.
- [YIW*91] YABE T., ISHIKAWA T., WANG P. Y., AOKI T., KADOTA Y., IKEDA F.: A universal solver for hyperbolic equations by cubic-polynomial interpolation II. Two- and three-dimensional solvers. *Computer Physics Communications* 66, 2–3 (1991), 233–242.
- [You82] YOUNGS D. L.: Time-dependent multi-material flow with large fluid distortion. In *Numerical Methods for Fluid Dynamics* (1982), Morton K. W., Baines M. J. (Eds.), Academic Press, pp. 273–285.