Eurographics Conference on Visualization (EuroVis) 2016
K.-L. Ma, G. Santucci, and J. van Wijk
(Guest Editors)

Volume 35 (2016), Number 3

MCFTLE: Monte Carlo Rendering of Finite-Time Lyapunov Exponent Fields

Tobias Günther[1], Alexander Kuhn[2] and Holger Theisel[1]

[1]Visual Computing Group, University of Magdeburg
[2]Zuse Institute Berlin

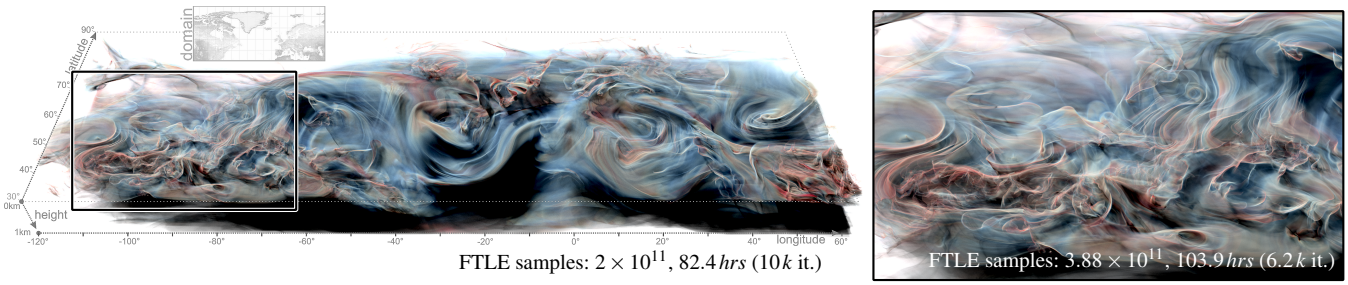


Figure 1: High-quality FTLE visualization of a European Centre for Medium-Range Weather Forecasts (ECMWF) reanalysis simulation of the wind velocity field of the northern hemisphere at April, 10th in 2010. The three-dimensional FTLE field emphasizes the spatial turbulence structure in the wind field and illustrates differences above land masses of North America (left and detail region), the North Atlantic region (center) and Europe (right). Integration duration (GPU): $\tau = 10$, majorant extinction $\bar{\sigma}_t = 1$, FTLE range $R = [0.35, 0.61]$, $\bar{\sigma}_t$ grid: $100 \times 50 \times 10$, left: 1600×600, right: 1400×800 pixels.

Abstract

Traditionally, Lagrangian fields such as finite-time Lyapunov exponents (FTLE) are precomputed on a discrete grid and are ray casted afterwards. This, however, introduces both grid discretization errors and sampling errors during ray marching. In this work, we apply a progressive, view-dependent Monte Carlo-based approach for the visualization of such Lagrangian fields in time-dependent flows. Our approach avoids grid discretization and ray marching errors completely, is consistent, and has a low memory consumption. The system provides noisy previews that converge over time to an accurate high-quality visualization. Compared to traditional approaches, the proposed system avoids explicitly predefined fieldline seeding structures, and uses a Monte Carlo sampling strategy named Woodcock tracking to distribute samples along the view ray. An acceleration of this sampling strategy requires local upper bounds for the FTLE values, which we progressively acquire during the rendering. Our approach is tailored for high-quality visualizations of complex FTLE fields and is guaranteed to faithfully represent detailed ridge surface structures as indicators for Lagrangian coherent structures (LCS). We demonstrate the effectiveness of our approach by using a set of analytic test cases and real-world numerical simulations.

This is the authors preprint. The definitive version is available at http://diglib.eg.org/ and http://onlinelibrary.wiley.com/.

1. Introduction

In flow visualization, finite-time Lyapunov exponent (FTLE) fields became one of the most popular tools to analyze coherent structures in unsteady vector fields [PPF*11, SLM05]. They describe how nearby-released particles separate over time, which allows to approximate regions of coherent behavior and barriers, i.e., Lagrangian coherent structures (LCS) or material structures, that particles will never cross [HY00, Hal01]. LCS have been shown to be valuable in a wide range of application scenarios such as predicting ocean pollutant transport due to oil mishaps, studying the spread of algae in water currents, or to observe the feeding habits of a jellyfish (see [Hal15] for an introduction). The FTLE field is a *Lagrangian*

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd. Published by John Wiley & Sons Ltd.

measure, i.e., a scalar field that arises from integration in a vector field and poses specific challenges with respect to the required computational resources and analysis complexity. All integration-based scalar fields and Lagrangian features have a key property in common: due to the integration, features are non-local and are defined at sub-grid resolution compared to the resolution of the underlying vector field. Since observable features (e.g., ridges) can become arbitrarily thin, the discretization and sampling of these fields is a fundamental challenge. In 3D, these fields are typically precomputed on grids and are afterwards view-dependently ray casted, which introduces both grid discretization errors and sampling errors during ray marching. Further, even adaptive discretization [SP07, BGT12] can still be quite memory intensive.

In this paper, we aim for high accuracy renderings of integration-based scalar fields, with focus on FTLE, by the use of Monte Carlo rendering techniques. Similar to [KPB12], we adopt progressive volumetric radiance transfer simulations to obtain high quality, aliasing-free renderings. In this paper, however, we concentrate on renderings of *Lagrangian measures*. A scalable implementation therefore requires an acceleration data structure that allows to quickly determine an upper bound for the integration-based scalar value in a certain region. Since this cannot be decided a-priori due to sampling problems, the technical novelty of this paper is that we adapt the underlying rendering method to progressively converge to the correct upper bounds. The remaining technical aspects (free path sampling, transmittance estimator and acceleration data structure) are applications of recent research from the offline rendering community.

In the rendering community, a Monte Carlo method is called *consistent* if it converges to the exact solution, i.e., is without any discretization errors. Consistent Monte Carlo techniques start noisy and progressively converge, which typically requires time. Our method is therefore not meant to replace recent interactive systems [BGT12], but is rather aiming at HQ renderings for marketing, and the generation of exact ground truth data. To the best of our knowledge, we present the first consistent renderings of Lagrangian measures. In summary, our method avoids the intermediate grid discretization, is without ray marching artifacts, has a low and bounded memory consumption, is easily parallelized on GPUs and is easy to implement.

2. Related Work

In the following, we formally introduce FTLE and review the recent work in consistent volume rendering.

2.1. Finite-Time Lyapunov Exponents

Given is an unsteady vector field $\mathbf{v}(\mathbf{x},t)$ that describes a time-dependent fluid flow. The movement of mass-less tracer particles is governed by $\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}(t) = \mathbf{v}(\mathbf{x}(t),t)$, i.e., the particle trajectory is always tangential to the flow. Such particle trajectory is commonly referred to as *pathline*. The *flow map* $\phi_t^\tau(\mathbf{x}) = \phi(\mathbf{x},t,\tau)$ is a shortened notation, which maps a particle seeded at $(\mathbf{x},t)$ to its destination after pathline integration for duration τ. The (spatial) gradient of the flow map $\nabla\phi(\mathbf{x},t,\tau) = \frac{\partial}{\partial\mathbf{x}}\phi(\mathbf{x},t,\tau)$ describes the behavior of particles released close to each other. We are interested in their separation behavior, which is characterized by the right Cauchy-Green deformation tensor $\nabla^\mathrm{T}\nabla$. Its largest real, positive eigenvalue λ_max denotes the (squared) largest magnitude of separation. Accounting for the exponential growth and normalizing by the integration duration τ yields the finite-time Lyapunov exponent (FTLE) [HY00, Hal01]:

$$\mathrm{FTLE}(\mathbf{x},t,\tau) = \frac{1}{|\tau|}\ln\sqrt{\lambda_\mathrm{max}(\nabla^\mathrm{T}\nabla)}, \tag{1}$$

Commonly, ridges of FTLE fields are often used as indicators for Lagrangian coherent structures (LCS) [SLM05]. In the literature, a variety of alternative LCS extraction techniques can be found (see [OHH15] for an overview) that typically exploit differential properties of the flow map and/or Lagrangian properties of the flow around tracer trajectories. For FTLE, the probably most widespread implementation (and the one we used) is to seed particles close to one another and approximate the flow map gradient by taking finite differences of their reached destinations, as in Haller and Yuan [HY00, Hal01]. The flow visualization community proposed alternative methods, such as localized FTLE [KPH*09], streak surface-based extraction [ÜSE13] and timeline tracking [KER*14]. A benchmark comparison of further computation methods was compiled by Kuhn et al. [KRWT12].

FTLE computations can be accelerated, e.g., by adaptive refinement of the flow map by Catmull-Rom interpolation [GGTH07], by the observation of filtered height ridges [SP07] or around automatically detected geometric structures [BT13]. Further, higher order flow map approximation [ÜSK*12] and timeline refinement [KER*14] schemes have been proposed. There exist several integral curve approximation schemes, such as hierarchical lines [HSW11], interpolation [COJ15, AOGJ15] and edge maps [BJB*12], which can be used to speed up any Lagrangian analysis technique. Generally, this requires a compromise between memory consumption, interactive response (speed) and accuracy (quality). An interactive 3D FTLE visualization was developed by Barakat et al. [BGT12], who interleave a computation and rendering phase to view-dependently build an adaptively sampled hierarchical representation of the FTLE field. Similar to our method, it produces more accurate FTLE approximations within multiple rendering passes. The accuracy of their method, however, is bound by memory and as it is based on ray marching, it cannot deliver a consistent solution. In this paper, we strive for a consistent method that operates on a small and fixed memory bound, and avoids discretization onto (adaptive) grids and ray marching errors.

2.2. Consistent Volume Rendering

In rendering, Monte Carlo sampling methods are deeply established as a practical and general approach to calculate light transport [Vea98]. Generally, a Monte Carlo estimator computes the average of a sequence of n measurements M_i with $i \in \{1, ..., n\}$ that is supposed to match some unknown quantity Q. Thereby, each measurement has an error $M_i - Q$. If the expected value of this error is zero, the method is called *unbiased*. A famous example in light transport is bidirectional path tracing [LW93]. However, even *biased* methods can be made *consistent* if the bias converges to zero as n increases, such as in progressive photon mapping [HOJ08]. Note that being biased does not necessarily mean that convergence is reached slower. In this paper, we seize an *unbiased* approach (Section 3). Our subsequent acceleration, however,

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd.

is *biased* (Section 4). Nevertheless, the method remains *consistent*. Consistent light transport in participating media has also been extensively researched [CPP*05], including improved importance sampling [KF12], efficient beam estimates [JNT*11] and their union with general light transport [KGH*14], free path sampling [RSK08] and its acceleration [YIC*10, SKTM11], as well as efficient transmittance estimators [NSJ14].

In visualization, direct volume rendering has found many applications [MHB*00, HLSR08]. Photorealistic light transport on regular scalar fields was considered in the Exposure renderer by Kroes et al. [KPB12]. Note that integration-based scalar fields (such as FTLE) are much more expensive to evaluate. A scalable implementation requires additional changes to the necessary acceleration data structures [YIC*10, SKTM11] that could be neglected in [KPB12]. We describe how we tailored the acceleration to FTLE fields in Section 4. A general sampling and reconstruction framework for progressive rendering was described by Frey et al. [FSME14]. Recent surveys on GPU-based volume rendering have been compiled by Beyer et al. [BHP14] and Balsa Rodríguez [BRGIG*14]. Typically, volume rendering approaches employ ray marching. Ray marching, however, results in unpredictable bias [NSJ14], and becomes much slower in high resolution volume data (in integration-based volumes the features can become arbitrarily thin) than Monte Carlo methods, as demonstrated by Yue et al. [YIC*10].

3. Monte Carlo FTLE Rendering

Our method is generally applicable to a progressive and consistent rendering of integration-based scalar fields. As such, we keep the theory and problem formulation general. Given is a scalar field $F(\mathbf{x}) : \mathcal{D} \to \mathbb{R}$ that is defined over the spatial domain $\mathcal{D} \subseteq \mathbb{R}^3$. In this paper, we regard $F(\mathbf{x})$ as the FTLE value at $\mathbf{x}$ at a given time t and integration duration τ, see Eq. (1).

$$F(\mathbf{x}) = \mathrm{FTLE}(\mathbf{x}, t, \tau) \tag{2}$$

We drop the temporal arguments for brevity, as they remain fixed. Next, we explain the light transport model that we use and describe its progressive Monte Carlo-based computation.

3.1. Volume Rendering Equation

We employ a single-scattering model [Max95], as illustrated in Fig. 2. For this, we need to set an extinction coefficient $\sigma_t(\mathbf{x})$ and a scattering albedo color $\mathbf{c}(\mathbf{x})$, which are both derived via transfer functions from the scalar field $F(\mathbf{x})$. Fig. 3 shows the transfer function that we used throughout the paper for color $\mathbf{c}(\mathbf{x})$. Extinction $\sigma_t(\mathbf{x})$ was mapped linearly on the same value range. The scattering albedo $\mathbf{c}$ is the ratio between scattering and extinction coefficients $\mathbf{c} = \sigma_s/\sigma_t$, and denotes the probability of a scattering event at a certain location. Intuitively speaking, the color red means that all red photons scatter, whereas blue and green photons are absorbed.

The transmittance T_r is the fraction of light that reaches point $\mathbf{x}'$ if emitted from $\mathbf{x}$ (or vice versa), when traveling on a straight line. In inhomogeneous media, it is defined as

$$T_r(\mathbf{x} \leftrightarrow \mathbf{x}') = e^{-\int_0^d \sigma_t(\mathbf{x}_s)\, \mathrm{d}s} \tag{3}$$

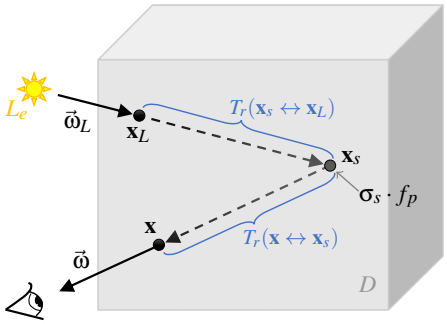


Figure 2: Illustration of single-scattering light transport. A directional light emits radiance L_e in direction $\vec{\omega}_L$, which enters the domain D at $\mathbf{x}_L$. The transmittance $T_r(\mathbf{x}_s \leftrightarrow \mathbf{x}_L)$ accounts for the attenuation on the way toward $\mathbf{x}_s$, where it scatters with coefficient σ_s. The amount of light that is scattered toward the viewer in direction $\vec{\omega}$ is determined by $f_p(\mathbf{x}_s, \vec{\omega}_L \to \vec{\omega})$. Finally, the transmittance $T_r(\mathbf{x} \leftrightarrow \mathbf{x}_s)$ attenuates toward the entry point $\mathbf{x}$ of the view ray.

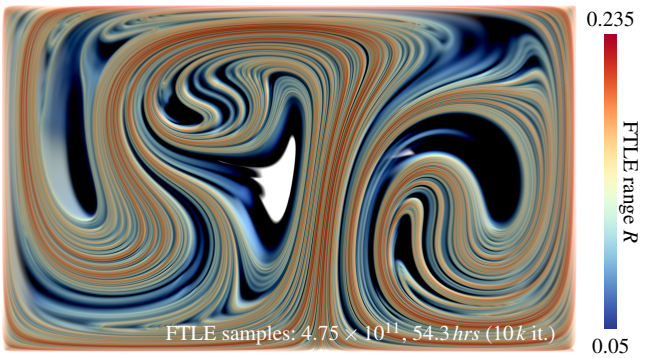


Figure 3: DOUBLE GYRE flow on the GPU without $\bar{\sigma}_t$ grid (Section 4) at 4000×2000 pixels. Integration time: $T = [0, 40]$, $\bar{\sigma}_t = 100$, FTLE range $R = [0.05, 0.235]$.

with $\mathbf{x}_s = \mathbf{x} + s\,\vec{\omega}$ and $s \in [0, d]$ parameterizes the ray in direction $\vec{\omega}$ from $\mathbf{x}$ to $\mathbf{x}'$. Note that by definition $0 \leq T_r \leq 1$.

The incoming radiance L at $\mathbf{x}$ with incoming direction $\vec{\omega}$ is governed by the *volume rendering equation* [Jar08, CPP*05]:

$$L(\mathbf{x} \leftarrow \vec{\omega}) = \int_0^d T_r(\mathbf{x} \leftrightarrow \mathbf{x}_s)\, \sigma_s(\mathbf{x}_s)\, L_i(\mathbf{x}_s \leftarrow \vec{\omega})\, \mathrm{d}s \tag{4}$$

$$\approx \frac{1}{n} \sum_{i=1}^{n} \frac{T_r(\mathbf{x} \leftrightarrow \mathbf{x}_s)\, \sigma_s(\mathbf{x}_s)\, L_i(\mathbf{x}_s \leftarrow \vec{\omega})}{p(\mathbf{x}_s)}\,. \tag{5}$$

Here, without emission and surface interaction, thus the integral only contains the in-scattered radiance L_i. Distributing samples $\mathbf{x}_s$ along the ray according to the probability

$$p(\mathbf{x}_s) = \sigma_t(\mathbf{x}_s)\, T_r(\mathbf{x} \leftrightarrow \mathbf{x}_s) \tag{6}$$

importance samples the integrand with respect to high FTLE values (high extinction σ_t) and high transmittance T_r, which further reduces Eq. (5) to

$$L(\mathbf{x} \leftarrow \vec{\omega}) \approx \frac{1}{n} \sum_{i=1}^{n} \mathbf{c}(\mathbf{x}_s)\, L_i(\mathbf{x}_s \leftarrow \vec{\omega}) \tag{7}$$

Later in Section 3.2, we explain a method that determines the scat-

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd.

tering location $\mathbf{x}_s$ of the photon according to probability $p(\mathbf{x}_s)$. The traveled distance from $\mathbf{x}$ to the scattering location $\mathbf{x}_s$ is called free flight distance $d_i(\mathbf{x} \to \mathbf{x}')$.

Assuming single-scattering from a directional light source with radiance L_e, incident direction $\vec{\omega}_L$, and $\mathbf{x}_L$ being the intersection of the light ray with the domain boundary, the in-scattered radiance L_i at a point $\mathbf{x}_s$ toward the viewer in direction ω is:

$$L_i(\mathbf{x}_s \leftarrow \vec{\omega}) = f_p(\mathbf{x}_s, \vec{\omega}_L \to \vec{\omega})\, T_r(\mathbf{x}_s \leftrightarrow \mathbf{x}_L)\, L_e \tag{8}$$

with $f_p(\mathbf{x}_s, \vec{\omega}_L \to \vec{\omega}) = 1/(4\pi)$ being an isotropic phase function (other choices are possible [AD16]), which describes how much light coming from $\vec{\omega}_L$ is scattered at $\mathbf{x}_s$ toward $\vec{\omega}$.

Given the scattering event at $\mathbf{x}_s$, we need to determine the fraction of light that arrives from the light source, i.e., $T_r(\mathbf{x}_s \leftrightarrow \mathbf{x}_L)$. Raab et al. [RSK08] and Szirmay-Kalos et al. [SKTM11] computed this for inhomogeneous participating media consistently as the expected value of a visibility experiment, in which they randomly emit photons at $\mathbf{x}_s$ and test if they reach $\mathbf{x}_L$:

$$\nu(\mathbf{x}_s) = \begin{cases} 1 & \text{if } d_i(\mathbf{x}_s \to \mathbf{x}_L) \geq \|\mathbf{x}_s - \mathbf{x}_L\| \\ 0 & \text{else} \end{cases} \tag{9}$$

Then, the final estimate of the incoming radiance becomes:

$$L(\mathbf{x} \leftarrow \vec{\omega}) \approx \frac{1}{4\pi n} \sum_{i=1}^{n} \mathbf{c}(\mathbf{x}_s)\, \nu(\mathbf{x}_s)\, L_e \tag{10}$$

Several improvements to the transmittance calculation have been proposed, including the separation of T_r into multiple parts and solving them separately [SKTM11, NSJ14], and estimating the visibility by multiple free path runs to reduce variance [JNT*11]. The more recent ratio tracking and residual tracking [NSJ14] lower the variance by weighting full paths, rather than taking them as binary estimates.

3.2. Free Path Sampling

Evaluating Eq. (10) requires a method to generate samples $\mathbf{x}_s$ along a ray according to probability $p(\mathbf{x}_s)$ in Eq. (6). The most commonly used method is the so-called *Woodcock tracking*, which is also known as *free path sampling*. It probabilistically determines the free flight distance of a photon until it reaches a scattering event in inhomogeneous media. The method was invented in the 60s in the neutron transport community [WMHL65] and was picked up by [RSK08] in the rendering community, where it had great impact in practice [JNT*11, NSJ14].

The idea of the method is to fill up inhomogeneous media with virtual particles to obtain a homogeneous medium with a joint, spatially constant so-called *majorant extinction* $\bar{\sigma}_t$. It then randomly samples the free flight distance in the joint homogeneous medium along the ray and thereby generates tentative particle interactions, which are probabilistically selected as either being virtual or real. In case of a virtual interaction ($\sigma_t/\bar{\sigma}_t < rand()$), the random walk continues. As proven by Coleman [Col68], the algorithm is *unbiased* and generates samples according to probability $p(\mathbf{x}_s)$. The algorithm is listed in Alg. 1. Carter et al. [CCT72] and Galtier et al. [GBC*13] showed variants that handle non-bounding majorants, but at the cost of increased variance [NSJ14].

Input: Ray origin $\mathbf{x}_0$ and normalized direction $\vec{\omega}$, majorant extinction $\bar{\sigma}_t$ and a ray interval $(d_{min}, d_{max}]$ to evaluate.
Output: Free flight distance d.
$d \leftarrow d_{min} - \ln(1 - rand())/\bar{\sigma}_t$
while $d \leq d_{max} \wedge \sigma_t(\mathbf{x}_0 + d\,\vec{\omega})/\bar{\sigma}_t < rand()$ **do**
  $d \leftarrow d - \ln(1 - rand())/\bar{\sigma}_t$
end

Algorithm 1: This algorithm determines the free flight distance of a photon along a ray $(\mathbf{x}_0, \vec{\omega})$. It implements $d_i(\mathbf{x} \to \mathbf{x}')$ with $\vec{\omega}$ being the normalized direction vector from $\mathbf{x}$ to $\mathbf{x}'$, $d_{min} = \|\mathbf{x} - \mathbf{x}_0\|$, and $d_{max} = \|\mathbf{x}' - \mathbf{x}_0\|$. (Pseudo code adapted from [YIC*10].)

4. Acceleration by Spatially-Varying Extinction Bounds

The expected worst case number of samples a free path run performs on a ray interval $(d_{min}, d_{max}]$ is $\bar{\sigma}_t \cdot (d_{max} - d_{min})$, since the expected step size in the joint homogeneous medium is $1/\bar{\sigma}_t$. Thus, the runtime strongly depends on how tightly $\bar{\sigma}_t$ bounds the true extinction σ_t. This is especially crucial in "empty" areas, where a globally defined upper bound strongly over-estimates the extinction. Therefore, Yue et al. [YIC*10] proposed a kd-tree-based space partition of the domain, in which they stored a bounding majorant extinction $\bar{\sigma}_t$ per volume segment. In a related approach, Szirmay-Kalos et al. [SKTM11] defined several segmentations over regular grids.

In this paper, we follow one approach in [SKTM11] and use piecewise constant bounds stored on a coarse voxel grid, i.e., a constant value per voxel. They have shown that for a correct transition between volume segments, a free path run always has to start at the entry point in the next voxel. If a scattering event is found inside a voxel it is reported, otherwise the run proceeds in a DDA traversal with the next voxel until the ray leaves the domain, see Alg. 2.

Input: Ray origin $\mathbf{x}_0$ and normalized direction $\vec{\omega}$, and a ray interval $(d_{min}, d_{max}]$ to evaluate.
Output: Free flight distance d.
$\mathcal{V} \leftarrow$ locate voxel containing $\mathbf{x}_0 + d_{min}\,\vec{\omega}$
do
  $\hat{d}_{min} \leftarrow \mathcal{V}.entry()$
  $\hat{d}_{max} \leftarrow \mathcal{V}.exit()$
  $\bar{\sigma}_t \leftarrow \mathcal{V}.majorant()$
  $d \leftarrow freePathRun(\mathbf{x}_0, \vec{\omega}, \bar{\sigma}_t, \hat{d}_{min}, \hat{d}_{max})$
  if $d < \hat{d}_{max} \vee d \geq d_{max}$ **then break**;
while $\mathcal{V} \leftarrow \mathcal{V}.nextVoxel()$;

Algorithm 2: Determines free flight distance along a ray $(\mathbf{x}_0, \vec{\omega})$ by DDA traversal in a voxel grid that contains the majorant extinction $\bar{\sigma}_t$. Alg. 1 implements *freePathRun*.

Finding Local Upper Bounds

In integration-based scalar fields a tight upper bound cannot be evaluated locally. Ridge features might become arbitrarily thin and trying to sample them is a costly endeavor. For this reason, we start with an (under-estimating) approximation of the true upper bound by taking a small number of k samples per voxel of the acceleration grid (typically, $k \approx 4$). This initial approximation causes the method to be *biased*. Over time, the free path runs take additional samples, at which we update the upper bounds. Since the free path sampling

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd.

produces a continuous and unbiased sampling of the domain, the true upper bound is found in the limit. Thus, the bias converges to zero, and hence the method remains *consistent*.

The initial upper extinction bound estimate of a voxel might be zero, even though a sharp ridge might exist in the voxel. In this case, all future free path runs would skip the voxel, and no further samples would be taken. The method would become inconsistent. To guarantee a certain minimum sampling rate, we clamp the extinction of each voxel to a lower bound, which adds additional tracing cost. The lower extinction bound is chosen such that a voxel with diagonal extent d has a $p_{\mathcal{V}}$ probability of being sampled by a free path run along the diagonal, i.e., the lower extinction bound is $p_{\mathcal{V}}/d$. Thereby, $p_{\mathcal{V}}$ is a user parameter that balances performance versus quality (convergence to true upper bound), and we empirically set $p_{\mathcal{V}} = 0.1$: In practice, we found that setting $p_{\mathcal{V}} = 0$ gives visually similar results. In the ABC flow, the root mean squared error (RMSE) for $p_{\mathcal{V}} = 0$ after 1,000 iterations was reached with $p_{\mathcal{V}} = 0.1$ after 750 iterations already. For $p_{\mathcal{V}} = 0.1$ the runtime increases by about factor 1.22. Thus, the additional cost for assuring consistency amortizes.

5. Implementation

We implemented our method on the CPU (multi-threaded) and on the GPU using DirectCompute. The accompanying material contains C++ demo code that implements the Monte Carlo FTLE rendering in the DOUBLE GYRE flow on the CPU. The CPU code is easy to integrate into existing visualization frameworks, and may serve the purpose of generating ground truth data for other rendering methods.

We computed the flow map gradients by taking finite differences [HY00, Hal01]. Thereby, the separation distance was set to 10^{-6} to capture thin ridge structures. Generally, separation distances cannot be arbitrarily small, since for even smaller values numerical problems arise in the computation of the eigenvalues of the Cauchy-Green tensor in Eq. (1), see Kuhn et al. [KRWT12] for a study of this effect. The numerical issues in the underlying scalar field are independent of the rendering method, and thus do not make the Monte Carlo approach unbiased or inconsistent.

Alg. 3 computes the incoming radiance L of a pixel according to Eq. (10). Thereby, *generateViewRay* uniformly samples the pixel for anti-aliasing purposes and the method *intersectRayWithDomain* calculates the entry and exit distance of the view ray into and out of the domain. Note that d_{min} is set to 0 if the ray origin is inside the domain. Finally, Alg. 2 implements the method *freePathRunDDA*.

6. Results

We applied our method to a number of analytic and real-world flows. In all figures, we list the rendering time, number of iterations (rays per pixel), and the total number of FTLE samples taken to compute the image. Note that we deliberately used a high number of iterations to produce high quality images. A convergence series, showing that early results are useful, too, is shown in the next section.

6.1. Double Gyre

The DOUBLE GYRE [SLM05] is a periodic 2D unsteady vector field that is commonly used as a benchmark for FTLE computations. In this paper, we define it in the temporal-periodic domain $D \times T = [0, 2] \times [0, 1] \times [0, 10]$ and use the parameterization

$$\mathbf{v}(x, y, t) = \begin{pmatrix} -0.1\pi \sin(f(x,t)\,\pi) \cos(y\pi) \\ 0.1\pi \cos(f(x,t)\,\pi) \sin(y\pi)\, \frac{\mathrm{d}}{\mathrm{d}x} f(x,t) \end{pmatrix} \tag{11}$$

with $f(x,t) = a(t)\,x^2 + b(t)\,x$ and $a(t) = 0.25\sin(t\,\pi/5)$ and $b(t) = 1 - 0.5\sin(t\,\pi/5)$. In order to create a 3D unsteady sequence, we set it constant along the z dimension. An example is shown in Fig. 3. Fig. 4 shows a convergence series, conveying an impression of what early results look like and at which rate the noise reduces.

Input: Pixel coordinate (p_x, p_y), number of iterations n.
Output: Incoming radiance L of pixel.
$L \leftarrow 0$;
for $i = 1$ **to** n **do**
  $(\mathbf{x}_0, \vec{\omega}) \leftarrow generateViewRay(p_x, p_y)$
  $(d_{min}, d_{max}) \leftarrow intersectRayWithDomain(\mathbf{x}_0, \vec{\omega})$
  $d \leftarrow freePathRunDDA(\mathbf{x}_0, \vec{\omega}, d_{min}, d_{max})$
  if $d < d_{max}$ **then**
    $\mathbf{x}_s \leftarrow \mathbf{x}_0 + d\,\vec{\omega}$
    $L \leftarrow L + \mathbf{c}(\mathbf{x}_s)\,\nu(\mathbf{x}_s)\,L_e$
  end
end
$L \leftarrow L/(4\pi n)$

Algorithm 3: Computes radiance according to Eq. (10).

6.2. ABC flow

The ABC (Arnold-Beltrami-Childress) flows are a class of parameterizable 3D unsteady flows that are often studied to assess turbulence. We used the following parameterization:

$$\mathbf{v}(x, y, z, t) = \begin{pmatrix} c(t)\sin(z) + \cos(y) \\ \sqrt{2}\sin(x) + c(t)\cos(z) \\ \sin(y) + \sqrt{2}\cos(x) \end{pmatrix} \tag{12}$$

with $c(t) = \sqrt{3} + (1 - e^{-0.1\,t})\sin(2\pi t)$. The flow is defined in the domain $D \times T = [0, 2\pi]^3 \times [0, 40]$. Fig. 5 gives an example of the particularly sharp ridges obtained in this flow.

6.3. Rabinovich-Fabrikant Equations

The RABINOVICH-FABRIKANT equations describe a parameter-dependent 3D steady dynamical system that exhibits chaotic behavior for certain parameter configurations. We consider it in the domain $D = [-15, 15]^3$:

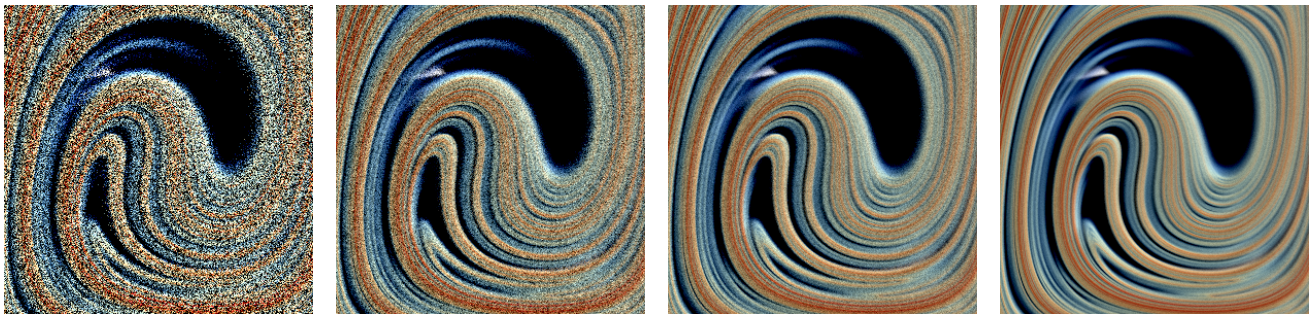
$$\mathbf{v}(x, y, z) = \begin{pmatrix} y(z - 1 + x^2) + \gamma x \\ x(3z + 1 - x^2) + \gamma y \\ -2z(\alpha + xy) \end{pmatrix} \tag{13}$$

here, with $\alpha = 0.98$ and $\gamma = 0.1$. This flow is difficult to analyze numerically, as traditional integrators reach different attractors depending on the step size [DC04]. In Fig. 6, we used a fourth-order Runge-Kutta integrator with step size $\Delta h = 0.1$.

6.4. Boussinesq

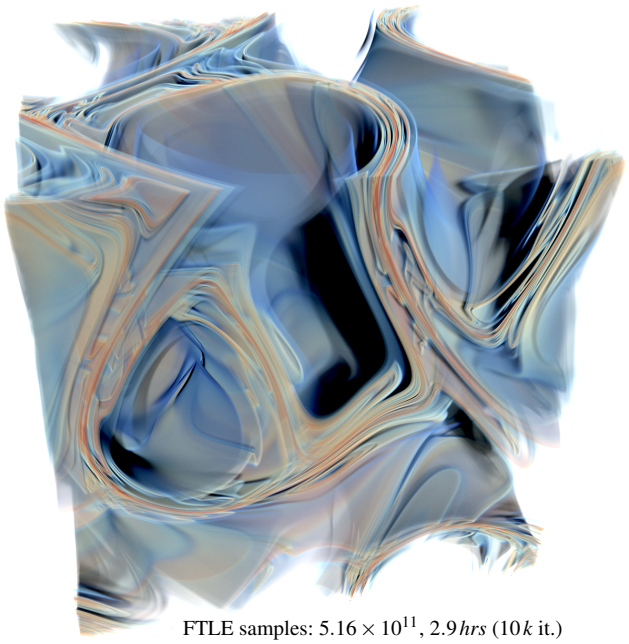
The BOUSSINESQ flow was provided by Tino Weinkauf, was simulated using Gerris Flow solver [Pop04] and uses the Boussinesq-approximation to generate the turbulent vortex behavior. It contains

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd.



(a) 1.1 secs, 3 iter. (b) 10 secs, 27 iter. (c) 1 min, 160 iter. (d) 10 mins, 1600 iter.

Figure 4: Convergence series of a close-up in the DOUBLE GYRE at 300×300 pixels, using the GPU (without $\bar{\sigma}_t$ grid). The majority of the noise vanishes in a few minutes. Integration time: $T = [0, 40]$, $\bar{\sigma}_t = 100$, FTLE range $R = [0.05, 0.235]$.



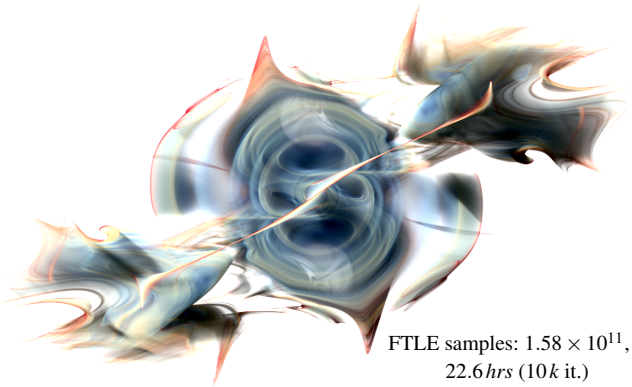
FTLE samples: 5.16×10^{11}, $2.9\,hrs$ ($10\,k$ it.)

Figure 5: ABC flow on the GPU without $\bar{\sigma}_t$ grid at 1500×1500 pixels, showing several sharp ridges. Integration time: $T = [5, 15]$, $\bar{\sigma}_t = 35$, FTLE range $R = [0.38, 0.92]$.

a 2D unsteady convection simulation that develops around a heated cylinder. In Fig. 7, the start time of the integration is mapped to the third spatial dimension.

6.5. Simulated Cloud-topped Boundary Layer (CTBL)

The CTBL data set contains a cloud resolving boundary layer simulation (UCLA-LES, details in [Ste13]). A large eddy simulation (LES) was used on a finite size domain (longitudinal, latitudinal extend: $10\,km$ with 384 cells, height extend $3.2\,km$ with 130 cells) under idealized conditions: It uses double-periodic boundary conditions and homogeneous surface forcing, while large-scale information are taken from the COSMO-DE simulation model. The purpose of this model is to study the detailed cloud dynamics on high spatial and temporal resolutions for sub-scale parametrization of synoptic



FTLE samples: 1.58×10^{11}, $22.6\,hrs$ ($10\,k$ it.)

Figure 6: RABINOVICH-FABRIKANT flow on the GPU with $\bar{\sigma}_t$ grid of $50 \times 20 \times 5$ at 1200×750 pixels. Integration duration $\tau = 20$, $\bar{\sigma}_t = 5$, FTLE range $R = [0, 0.47]$.

climate simulations. For this task, convective flow patterns are of central interest since they are tightly coupled with cloud production and interaction processes. FTLE fields are specifically useful to visualize the complex spatial structure and distribution of turbulent plumes produced by the UCLA-LES model. The results for our method are shown in Fig. 8.

6.6. ECMWF Reanalysis

The ECMWF data set shows a large scale ECMWF reanalysis simulation of the weather in the northern hemisphere (long. range: -120° to 60° with 1200 cells, lat. range: 30° to 90° with 40 cells, height from $0\,m$ to $1050\,m$ with 90 cells) from April $10-19$, 2010. Similarly to the CTBL example, convective flow features in the wind velocity field are of central interest, since they are strongly related to the exchange of energy and transport of trace gases in the atmosphere. LCS features have been shown to characterize transport patterns and mixing behavior of atmospheric flows [Hal15]. In Fig. 1, FTLE highlights the characteristic structure of atmospheric features above the North American land surface (bottom left, detailed zoom), the North Atlantic ocean (center) and the European land mass (bottom right corner). Specifically the spatial turbulent structure of vortices (cyclones) and stream-like features are empha-

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd.

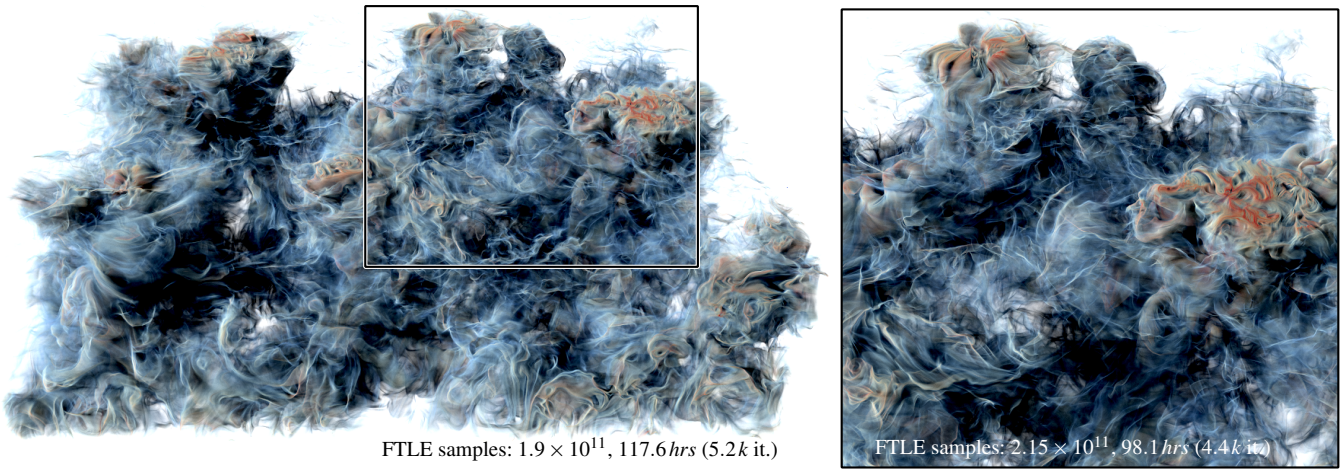


Figure 8: CLOUD-TOPPED BOUNDARY LAYER flow on the GPU with $40 \times 20 \times 20$ $\bar{\sigma}_t$ grid. Integration duration: $\tau = 30$, majorant extinction $\bar{\sigma}_t = 2$, FTLE range $R = [0.09, 0.21]$, left: 1200×640, right: 800×800 pixels.

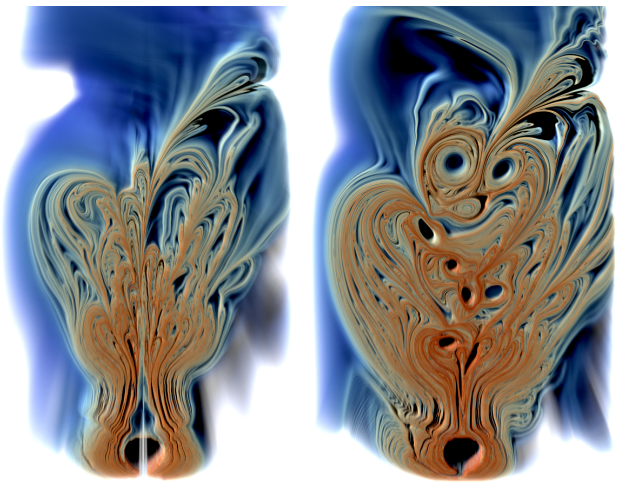


Figure 7: BOUSSINESQ flow on the GPU with $\bar{\sigma}_t$ grid of $5 \times 10 \times 5$ at 600×1600 pixels. We set $\bar{\sigma}_t = 60$ and FTLE range $R = [0.02, 1]$. The flow is shown at different start times (front slice), left: $t_0 = 1$, right: $t_0 = 5$, both with $\tau = 10$.

sized. Note that we used time-averaged velocity fields to compute FTLE fields.

7. Discussion

In the following, we compare our method with traditional ray casting and report timings, number of FTLE samples and memory consumption for the figures shown throughout the paper. Afterwards, we observe the convergence behavior of the Monte Carlo integration and discuss the limitations.

7.1. Comparison with Ray Casting

First, we show the artifacts that grid discretization and ray marching entail. The results are obtained in the ABC flow with our CPU code using a resolution of 100×100 pixels. For comparison, a GPU-based high resolution image of the same flow is shown in Fig. 5.

In Figs. 9a and 9b, the scalar field was discretized onto a 200^3 or 500^3 regular grid, respectively, and was afterwards ray casted. Thereby, the separation distance of particles in the FTLE computation was set to the voxel size. The FTLE value range depends on the separation distance, same as the sharpness of the ridges that can be recovered. The tracing time is reported per iteration.

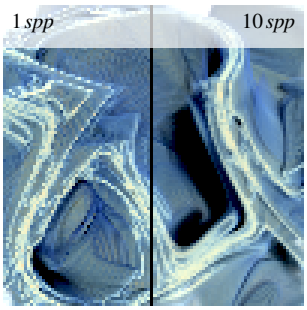
Figs. 9c and 9d show the same setting as in (a)–(b), but with smaller separation distance. Here, the discretization onto the grid causes an under-sampling of the ridges. Since the discrete sampling does not preserve maxima, the transfer functions return more transparent and blueish values (color shift). We display the result with 1 sample per pixel (*spp*) to show the systematic bias and the average of 10 *spp*, i.e., 10 tracing iterations. Fig. 9e and Fig. 9f apply the ray casting directly to the FTLE scalar field without discretization. The step size discretization during ray marching results in Fig. 9e in heavy aliasing. We used the same step sizes as in (a)–(d) and show 1 *spp*. Although early ray termination was used, the computation time of ray casting is high.

Fig. 9g and Fig. 9h show results of our consistent Monte Carlo-based approach. The discretization artifacts (aliasing and color shifts) are avoided; instead the result contains the typical Monte Carlo noise. In a time comparable to Fig. 9c, 70 iterations could be computed per pixel in Fig. 9g. With Fig. 9d setting the reference time, 1055 iterations were computed in Fig. 9h. The precomputation of the initial $\bar{\sigma}_t$ grid (60^3, 4 samples per voxel) took 17 *s*. Our method takes multiple *spp*, while the initial $\bar{\sigma}_t$ grid is precomputed once per FTLE field.

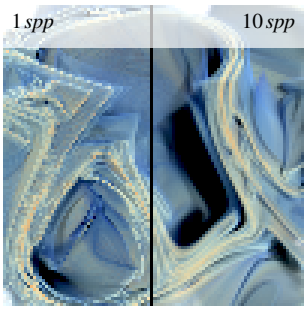
7.2. Timings, FTLE Samples and Memory Consumption

We used an AMD Phenom II X6 1055T CPU, and an Nvidia GeForce GTX 970 GPU with 4 GB VRAM. The results in Fig. 9 were obtained on an Intel Core i7-2600K CPU. Table 1 lists for all datasets the viewport resolution, and the average time and average number of

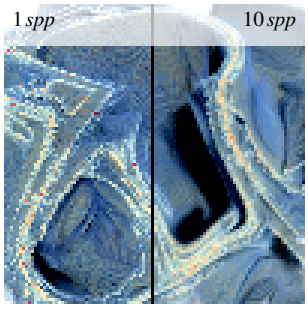
© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd.



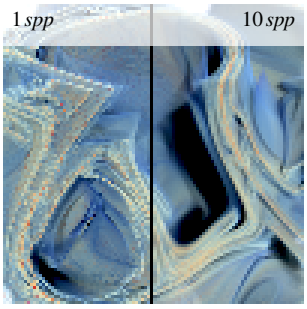
(a) Ray casting of discrete grid with 200^3 voxels (30.5 *MB*), grid precomp.: 22 *s*, tracing: 1.3 *s*.



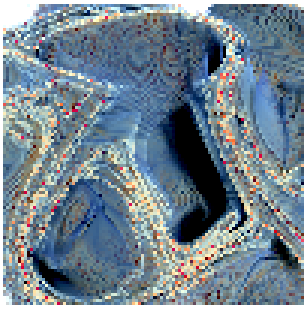
(b) Ray casting of discrete grid with 500^3 voxels (476 *MB*), grid precomp.: 5 *min*, tracing: 7.2 *s*.



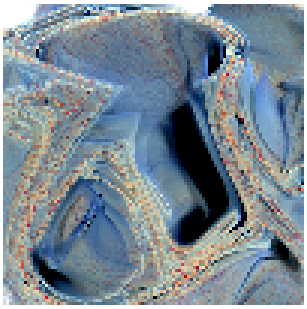
(c) Ray casting of discrete grid with 200^3 voxels (30.5 *MB*), grid precomp.: 129 *s*, tracing: 1.3 *s*.



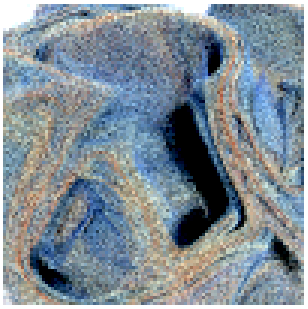
(d) Ray casting of discrete grid with 500^3 voxels (476 *MB*), grid precomp.: 30 *min*, tracing: 7.2 *s*.



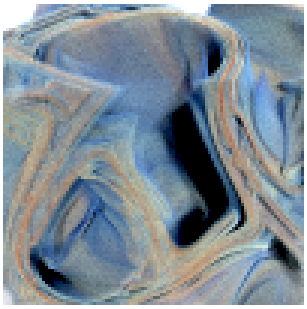
(e) Ray casting without grid (evaluate FTLE at each sample), 11.3 *min*, insufficient sampling rate (aliasing artifacts).



(f) Ray casting without grid (evaluate FTLE at each sample), 69.9 *min*, higher sampling rate than (e), still some artifacts.



(g) Monte Carlo (ours), 70 iterations, 111.3 *s*, no discretization artifacts, contains noise.



(h) Monte Carlo (ours), 1055 iterations, 29.7 *min*, less noise.

Figure 9: Comparison of Monte Carlo and ray casting.

FTLE samples (cost) per iteration per pixel. The latter is hardware-independent and only depends on the dataset and view. The CPU timings are in milliseconds (*ms*), whereas GPU timings are in microseconds (μs). We conducted experiments with and without $\bar{\sigma}_t$ grids, and list the required main memory. Our method stores four floats per pixel for the accumulated radiance (RGBA) and an optional (scalar) acceleration grid (both single precision). Thus, 16 *MB* are required per megapixel plus $< 1\,MB$ for a $\bar{\sigma}_t$ grid, as we have used on the GPU. The DOUBLE GYRE was rendered at highest resolution, and has thus the largest memory consumption.

The GPU code is orders of magnitude faster than the CPU code, and especially on the CPU the acceleration greatly increases the performance. In the RABINOVICH case (Fig. 6), for instance, large portions of the domain are empty. Thus, here, the accelerated CPU version is about $15\times$ faster. The speedup obtained by the $\bar{\sigma}_t$ grid depends on the tightness of the upper extinction bounds. If large areas in the domain have low extinction, much can be gained compared to a global constant upper bound. This is visible in the BOUSSINESQ case in Fig. 7. The left image has more empty areas and here the time and cost savings are greater than in the right image.

On the GPU, arithmetic (ALU) calculations are much faster than memory accesses (e.g., lookups in the $\bar{\sigma}_t$ grid) which is why we generally recommend lower $\bar{\sigma}_t$ grid resolutions than for the CPU in order to reduce memory I/O pressure. Our manually-tuned grid resolutions are listed in Table 2. Tracing in analytic fields is entirely ALU bound and requires no memory access at all. For the two analytic examples (DOUBLE GYRE in Fig. 3, ABC in Fig. 5), the $\bar{\sigma}_t$ lookups are more expensive than the brute force evaluation of more FTLE samples, while the balance tips in favor of the $\bar{\sigma}_t$ grid for the RABINOVICH case. In general, the $\bar{\sigma}_t$ grid can be recommended for sampled real-world flows, as here the $\bar{\sigma}_t$ lookup cost amortizes much faster.

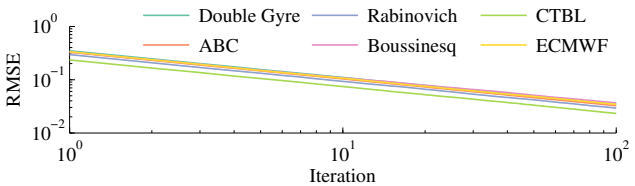


Figure 10: Plotting the RMSE over time in log-log scale shows the linear convergence expected from MC methods.

7.3. Convergence

In (bounded) Monte Carlo integration, the variance decreases asymptotically to zero. Plotting the root-mean-square error (RMSE) over time, i.e., the difference between the progressively estimated radiance from Eq. (10) and the ground truth radiance, in log-log scale thereby shows a linear convergence behavior in Fig. 10. The plotted RMSE for our data sets confirms that the method converges as expected. In this experiment, we used the GPU for the DOUBLE GYRE and ABC flow without $\bar{\sigma}_t$ grid and for the others with $\bar{\sigma}_t$ grid.

7.4. Limitations

Our bottleneck is the pathline integration. Recently, interpolation of pathlines from a set of candidates rather than full numerical

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd.

Data set	Viewport resolution	CPU time (*ms*) w/o acc.	CPU time (*ms*) w/ acc.	GPU time (μs) w/o acc.	GPU time (μs) w/ acc.	cost w/o acc.	CPU cost w/ acc.	GPU cost w/ acc.	memory (in *MB*)
DOUBLE GYRE, Fig. 3	4000×2000	2.41	**1.14**	**2.44**	8.25	5.94	2.79	5.20	122.07
ABC, Fig. 5	1500×1500	0.52	**0.13**	**0.46**	1.00	22.93	5.90	10.25	34.33
RABINOVICH, Fig. 6	1200×750	1.82	**0.12**	11.40	**9.04**	91.31	5.73	17.56	13.75
BOUSSINESQ, Fig. 7 (left)	600×1600	4.08	**0.13**	100.33	**12.83**	47.12	1.49	8.93	14.65
BOUSSINESQ, Fig. 7 (right)	600×1600	1.73	**0.17**	36.58	**9.08**	24.74	2.09	7.96	14.65
CTBL, Fig. 8	1200×640	14.10	**3.65**	250.83	**106.01**	62.99	16.09	47.55	11.78
CTBL, Fig. 8 (close-up)	800×800	20.94	**5.88**	220.00	**125.41**	93.38	25.77	76.49	9.83
ECMWF, Fig. 1	1600×600	3.29	**0.54**	79.50	**30.90**	54.17	7.56	20.83	14.84
ECMWF, Fig. 1 (close-up)	1400×800	6.82	**1.28**	121.03	**33.40**	111.56	17.62	34.64	17.28

Table 1: Viewport resolution, average time and average number of FTLE samples (average cost) per iteration per pixel, as well as the memory consumption for the respective figures throughout the paper (single precision). W/o acc. the CPU and GPU have identical cost.

Data set	$\bar{\sigma}_t$ grid (CPU)	$\bar{\sigma}_t$ grid (GPU)
DOUBLE GYRE	$100 \times 50 \times 1$	$20 \times 10 \times 1$
ABC	$60 \times 60 \times 60$	$20 \times 20 \times 20$
RABINOVICH	$300 \times 300 \times 30$	$50 \times 20 \times 5$
BOUSSINESQ	$200 \times 200 \times 50$	$5 \times 10 \times 5$
CTBL	$200 \times 100 \times 100$	$40 \times 20 \times 20$
ECMWF	$200 \times 100 \times 100$	$100 \times 50 \times 10$

Table 2: Manually-tuned resolutions of the $\bar{\sigma}_t$ grids.

integration has been investigated [COJ15, AOGJ15]. Clearly, any interpolation-based approach introduces an error that makes the rendering *inconsistent*. Nevertheless, if faster previews are desired in an entirely Monte Carlo-based framework, this is a promising approach to take. Every Monte Carlo-based approach essentially reduces variance on some quantity, which shows up as slowly reducing noise. In the rendering community, reconstruction filters and noise removal are applied to remove the residual noise in order to satisfy the often narrow time budgets. Out of that need, a large number of filters have been developed [ZJL*15] that are applicable here as well.

8. Conclusions

In this paper, we applied a consistent light transport simulation method to the rendering of Lagrangian scalar fields, focusing on FTLE. While previous approaches discretize the FTLE field onto a (possibly adaptive) grid [BGT12], our method avoids discretization errors completely and operates on a fixed and small memory bound. We used a progressive Monte Carlo sampling technique named Woodcock tracking for distributing the samples along the view ray, which obtains consistent solutions with iterative previews and is free of ray marching artifacts. With this, we obtained high quality visualizations of FTLE fields, which may serve as ground truth or are available for marketing applications. A technical novelty of this paper is that we tailored the acceleration data structure to FTLE, by progressively converging to correct local upper extinction bounds, which are not available a-priori in Lagrangian scalar fields. In the future, we would like to incorporate approximating techniques to provide faster previews, such as pathline interpolation methods [COJ15, AOGJ15]. Further, we would like to experiment with adaptive sampling and reconstruction filters [ZJL*15]. The recent residual tracking and ratio tracking [NSJ14] are alternative approaches to estimate transmittance, which could be evaluated in the context of Lagrangian scalar fields in future work. Using regular grids to find local bounds of the majorant extinction as in [SKTM11] requires to set the right voxel resolution manually to get the best performance. Yue et al. [YIC*10] avoided this by construction of a kd-tree with an adequate stopping criterion, which we would like to test. Our method is purely image-based. For further processing of the FTLE ridge surfaces, geometry extraction techniques are still a challenging topic.

Acknowledgements

This work was supported by DFG grant number TH 692/8-1 and was partially funded by the German Federal Ministry of Education and Research under grant number 01LK1213A.

References

[AD16] AMENT M., DACHSBACHER C.: Anisotropic ambient volume shading. *IEEE Transactions on Visualization and Computer Graphics 22*, 1 (2016), 1015–1024. 4

[AOGJ15] AGRANOVSKY A., OBERMAIER H., GARTH C., JOY K. I.: A multi-resolution interpolation scheme for pathline based Lagrangian flow representations. In *Proc. SPIE, Visual Data Analysis Conference* (2015), vol. 9397, p. 93970K. 2, 9

[BGT12] BARAKAT S. S., GARTH C., TRICOCHE X.: Interactive computation and rendering of finite-time Lyapunov exponent fields. *IEEE Transactions on Visualization and Computer Graphics 18*, 8 (2012), 1368–1380. 2, 9

[BHP14] BEYER J., HADWIGER M., PFISTER H.: A survey of GPU-based large-scale volume visualization. In *Proc. EuroVis State of the Art Reports* (2014), The Eurographics Association. 3

[BJB*12] BHATIA H., JADHAV S., BREMER P., CHEN G., LEVINE J. A., NONATO L. G., PASCUCCI V.: Flow visualization with quantified spatial and temporal errors using edge maps. *IEEE Transactions on Visualization and Computer Graphics 18*, 9 (2012), 1383–1396. 2

[BRGIG*14] BALSA RODRÍGUEZ M., GOBBETTI E., IGLESIAS GUITIÁN J., MAKHINYA M., MARTON F., PAJAROLA R., SUTER S.: State-of-the-art in compressed GPU-based direct volume rendering. *Computer Graphics Forum 33*, 6 (2014), 77–100. 3

[BT13] BARAKAT S. S., TRICOCHE X.: Adaptive refinement of the flow map using sparse samples. *IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE SciVis) 19*, 12 (2013), 2753–2762. 2

[CCT72] CARTER L. L., CASHWELL E. D., TAYLOR W. M.: Monte Carlo sampling with continuously varying cross sections along flight paths. *Nucl. Sci. and Eng. 48*, 4 (1972), 403–411. 4

[COJ15] CHANDLER J., OBERMAIER H., JOY K. I.: Interpolation-based pathline tracing in particle-based flow visualization. *IEEE Transactions on Visualization and Computer Graphics 21*, 1 (2015), 68–80. 2, 9

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd.

[Col68] COLEMAN W. A.: Mathematical verification of a certain Monte Carlo sampling technique and applications of the technique to radiation transport problems. *Nucl. Sci. and Eng. 32*, 1 (1968), 76–81. 4

[CPP*05] CEREZO E., PÉREZ F., PUEYO X., SERON F. J., SILLION F. X.: A survey on participating media rendering techniques. *The Visual Computer 21*, 5 (2005), 303–328. 3

[DC04] DANCA M.-F., CHEN G.: Bifurcation and chaos in a complex model of dissipative medium. *International Journal of Bifurcation and Chaos 14*, 10 (2004), 3409–3447. 5

[FSME14] FREY S., SADLO F., MA K.-L., ERTL T.: Interactive progressive visualization with space-time error control. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (2014), 2397–2406. 3

[GBC*13] GALTIER M., BLANCO S., CALIOT C., COUSTET C., DAUCHET J., HAFI M. E., EYMET V., FOURNIER R., GAUTRAIS J., KHUONG A., PIAUD B., TERRÉE G.: Integral formulation of null-collision Monte Carlo algorithms. *Journal of Quant. Spectroscopy and Rad. Transfer 125* (2013), 57–68. 4

[GGTH07] GARTH C., GERHARDT F., TRICOCHE X., HAGEN H.: Efficient computation and visualization of coherent structures in fluid flow applications. *IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE Visualization) 13*, 6 (2007), 1464–1471. 2

[Hal01] HALLER G.: Distinguished material surfaces and coherent structures in three-dimensional fluid flows. *Phys. D 149*, 4 (2001), 248–277. 1, 2, 5

[Hal15] HALLER G.: Lagrangian coherent structures. *Annual Review of Fluid Mechanics 47* (2015), 137–162. 1, 6

[HLSR08] HADWIGER M., LJUNG P., SALAMA C. R., ROPINSKI T.: Advanced illumination techniques for GPU volume raycasting. In *ACM SIGGRAPH ASIA Courses* (2008), pp. 1:1–1:166. 3

[HOJ08] HACHISUKA T., OGAKI S., JENSEN H. W.: Progressive photon mapping. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 27*, 5 (2008), 130:1–130:8. 2

[HSW11] HLAWATSCH M., SADLO F., WEISKOPF D.: Hierarchical line integration. *IEEE Transactions on Visualization and Computer Graphics 17*, 8 (2011), 1148–1163. 2

[HY00] HALLER G., YUAN G.: Lagrangian coherent structures and mixing in two-dimensional turbulence. *Phys. D 147*, 3–4 (2000), 352–370. 1, 2, 5

[Jar08] JAROSZ W.: *Efficient Monte Carlo Methods for Light Transport in Scattering Media*. PhD thesis, UC San Diego, La Jolla, CA, USA, 2008. 3

[JNT*11] JAROSZ W., NOWROUZEZAHRAI D., THOMAS R., SLOAN P.-P., ZWICKER M.: Progressive photon beams. *ACM Trans. Graph. (SIGGRAPH Asia) 30*, 6 (2011), No. 181. 3, 4

[KER*14] KUHN A., ENGELKE W., RÖSSL C., HADWIGER M., THEISEL H.: Time line cell tracking for the approximation of Lagrangian coherent structures with subgrid accuracy. *Computer Graphics Forum 33*, 1 (2014), 222–234. 2

[KF12] KULLA C., FAJARDO M.: Importance sampling techniques for path tracing in participating media. *Computer Graphics Forum (Proc. EGSR) 31*, 4 (2012), 1519–1528. 3

[KGH*14] KŘIVÁNEK J., GEORGIEV I., HACHISUKA T., VÉVODA P., ŠIK M., NOWROUZEZAHRAI D., JAROSZ W.: Unifying points, beams, and paths in volumetric light transport simulation. *ACM Trans. Graph. (Proc. SIGGRAPH) 33*, 4 (2014), 103:1–103:13. 3

[KPB12] KROES T., POST F. H., BOTHA C. P.: Exposure render: An interactive photo-realistic volume rendering framework. *PLoS ONE 7*, 7 (07 2012), e38586. 2, 3

[KPH*09] KASTEN J., PETZ C., HOTZ I., NOACK B., HEGE. H.-C.: Localized finite-time Lyapunov exponent for unsteady flow analysis. In *Proc. Vision, Modeling and Visualization* (2009), pp. 265–274. 2

[KRWT12] KUHN A., RÖSSL C., WEINKAUF T., THEISEL H.: A benchmark for evaluating FTLE computations. In *Proc. IEEE PacificVis* (2012), pp. 121–128. 2, 5

[LW93] LAFORTUNE E. P., WILLEMS Y. D.: Bi-directional path tracing. *Proc. Compugraphics* (1993), 145–153. 2

[Max95] MAX N.: Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics 1*, 2 (1995), 99–108. 3

[MHB*00] MEISSNER M., HUANG J., BARTZ D., MUELLER K., CRAWFIS R.: A practical evaluation of popular volume rendering algorithms. In *IEEE Symp. VolVis* (2000), pp. 81–90. 3

[NSJ14] NOVÁK J., SELLE A., JAROSZ W.: Residual ratio tracking for estimating attenuation in participating media. *ACM Trans. Graph. (SIGGRAPH Asia) 33*, 6 (2014), 179:1–179:11. 3, 4, 9

[OHH15] ONU K., HUHN F., HALLER G.: LCS tool: a computational platform for lagrangian coherent structures. *Journal of Computational Science 7* (2015), 26–36. 2

[Pop04] POPINET S.: Free computational fluid dynamics. *Cluster World 2*, 6 (2004). gfs.sourceforge.net [Online; accessed 25-February-2016]. 5

[PPF*11] POBITZER A., PEIKERT R., FUCHS R., SCHINDLER B., KUHN A., THEISEL H., MATKOVIĆ K., HAUSER H.: The state of the art in topology-based visualization of unsteady flow. *Computer Graphics Forum 30*, 6 (2011), 1789–1811. 1

[RSK08] RAAB M., SEIBERT D., KELLER A.: Unbiased global illumination with participating media. In *Monte Carlo and Quasi-MC Methods 2006*. Springer, 2008, pp. 591–605. 3, 4

[SKTM11] SZIRMAY-KALOS L., TÓTH B., MAGDICS M.: Free path sampling in high resolution inhomogeneous participating media. *Computer Graphics Forum 30*, 1 (2011), 85–97. 3, 4, 9

[SLM05] SHADDEN S. C., LEKIEN F., MARSDEN J. E.: Definition and properties of Lagrangian coherent structures from finite-time Lyapunov exponents in two-dimensional aperiodic flows. *Phys. D 212*, 3–4 (2005), 271–304. 1, 2, 5

[SP07] SADLO F., PEIKERT R.: Efficient visualization of Lagrangian coherent structures by filtered AMR ridge extraction. *IEEE Transactions on Visualization and Computer Graphics (IEEE Visualization) 13*, 6 (2007), 1456–1463. 2

[Ste13] STEVENS B.: Introduction to UCLA-LES, 2013. http://www.mpimet.mpg.de/fileadmin/atmosphaere/herz/les_doc.pdf [Online; accessed 25-February-2016]. 6

[ÜSE13] ÜFFINGER M., SADLO F., ERTL T.: A time-dependent vector field topology based on streak surfaces. *IEEE Transactions on Visualization and Computer Graphics 19*, 3 (2013), 379–392. 2

[ÜSK*12] ÜFFINGER M., SADLO F., KIRBY M., HANSEN C., ERTL T.: FTLE computation beyond first-order approximation. In *Eurographics (Short Papers)* (2012), pp. 61–64. 2

[Vea98] VEACH E.: *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University, Stanford, CA, USA, 1998. 2

[WMHL65] WOODCOCK E., MURPHY T., HEMMINGS P., LONGWORTH S.: Techniques used in the GEM code for Monte Carlo neutronics calculations in reactors and other systems of complex geometry. In *Applications of Comp. Meth. to Reactor Problems* (1965), vol. 557, Argonne National Laboratory, pp. ANL–7050. 4

[YIC*10] YUE Y., IWASAKI K., CHEN B.-Y., DOBASHI Y., NISHITA T.: Unbiased, adaptive stochastic sampling for rendering inhomogeneous participating media. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 29*, 6 (2010), 177:1–177:8. 3, 4, 9

[ZJL*15] ZWICKER M., JAROSZ W., LEHTINEN J., MOON B., RAMAMOORTHI R., ROUSSELLE F., SEN P., SOLER C., YOON S.-E.: Recent advances in adaptive sampling and reconstruction for Monte Carlo rendering. *Computer Graphics Forum (Proc. Eurographics) 34*, 2 (2015), 667–681. 9

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd.