

Article

Physics-Based Differentiable Rendering for Efficient and Plausible Fluid Modeling from Monocular Video

Yunchi Cen , Qifan Zhang  and Xiaohui Liang * 

School of Computer Science and Engineering, Beihang University, Beijing 100191, China; cencyc@buaa.edu.cn (Y.C.); zhangqf_m@buaa.edu.cn (Q.Z.)

* Correspondence: liang_xiaohui@buaa.edu.cn

Abstract: Realistic fluid models play an important role in computer graphics applications. However, efficiently reconstructing volumetric fluid flows from monocular videos remains challenging. In this work, we present a novel approach for reconstructing 3D flows from monocular inputs through a physics-based differentiable renderer coupled with joint density and velocity estimation. Our primary contributions include the proposed efficient differentiable rendering framework and improved coupled density and velocity estimation strategy. Rather than relying on automatic differentiation, we derive the differential form of the radiance transfer equation under single scattering. This allows the direct computation of the radiance gradient with respect to density, yielding higher efficiency compared to prior works. To improve temporal coherence in the reconstructed flows, subsequent fluid densities are estimated via a coupled strategy that enables smooth and realistic fluid motions suitable for applications that require high efficiency. Experiments on synthetic and real-world data demonstrated our method's capacity to reconstruct plausible volumetric flows with smooth dynamics efficiently. Comparisons to prior work on fluid motion reconstruction from monocular video revealed over 50–170x speedups across multiple resolutions.

Keywords: monocular video; fluid reconstruction; differentiable renderer



Citation: Cen, Y.; Zhang, Q.; Liang, X. Physics-Based Differentiable Rendering for Efficient and Plausible Fluid Modeling from Monocular Video. *Entropy* **2023**, *25*, 1348. <https://doi.org/10.3390/e25091348>

Academic Editors: Nikolay Kolev Vitinov and Zlatinka I. Dimitrova

Received: 16 August 2023

Revised: 9 September 2023

Accepted: 14 September 2023

Published: 17 September 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Realistic fluid flow reconstruction is of paramount importance for a wide range of applications, including those related to special effects in film and video, artistic design, and digital media generation. However, this task poses significant challenges. On the one hand, fluid phenomena are ubiquitous in daily life, and people have high expectations for the quality of their rendering in artistic and media contexts. However, existing methods that attempt to reconstruct fluid motion through a combination of physics-based constraints and density estimation require tens of minutes per frame for reconstruction, rendering them unsuitable for interactive rendering applications.

To reconstruct fluid motion, existing computed tomography (CT) methods (e.g., [1,2]) rely on sparse-view images and solve a least-squares problem that relates pixels and voxels using a visual hull. However, their reconstruction quality suffers as the number of views decreases, resulting in an under-determined inverse problem. Additionally, the sophisticated setups required by these methods make calibration difficult and limit their adaptability. Our method focuses on monocular video reconstruction, which enables us to use a wide variety of existing videos and capture new fluid motions with ease. Rather than solely simulating fluids from images, recent works (e.g., [3,4]) have employed a physics-based prediction scheme to add transport constraints to optimization and couple density estimation to velocity estimation. Although these physics-based priors have improved the temporal coherence of flow motion reconstruction, the time-consuming fluid density reconstruction and complex physics-based prediction scheme remain bottlenecks to reconstruction efficiency.

Differentiable rendering has been a major focus in computer graphics recently. It comprises techniques that integrate rendering into end-to-end optimization by obtaining

useful derivatives of the rendering process. However, integrating differentiable rendering into fluid reconstruction frameworks poses a huge challenge due to the complexity and nonlinearity of the relation between pixel intensities and fluid density. Franz et al. [5] proposed a differentiable-rendering-based fluid reconstruction framework, where automatic differentiation (AD) is used to compute the derivatives in a ray marching process. However, the current automatic differentiation framework is typically inefficient, especially when optimizing numerous gradient parameters.

To improve the efficiency of fluid reconstruction, we propose a physics-based differentiable rendering framework that analytically computes radiance derivatives with respect to density. Unlike prior automatic differentiation techniques, which have inherent computational overhead, our method directly calculates derivatives by deriving a differential form of the radiance transfer equation under single-scattering assumptions. This analytical approach avoids redundant operations and gradient tracking, significantly accelerating fluid density optimizations compared to previous differentiable rendering techniques.

Furthermore, we couple our optimized renderer with a constrained density and velocity estimation strategy. By propagating velocities and densities across time steps while enforcing physical constraints, we achieve temporally coherent fluid reconstructions far more efficiently than the costly tomographic methods used previously. Our major contributions are summarized as follows:

- An efficient physics-based differentiable rendering framework is proposed for fluid density reconstruction, which is more efficient than previous models.
- The differential form of the Radiative Transfer Equation (RTE) based on single scattering is derived.
- The transport constraints and the differentiable rendering step are coupled for fluid density updating, achieving smooth fluid motion reconstruction.

2. Related Work

Density reconstruction: Reconstructing fluid density from 2D observations has been a popular research topic in computer graphics. Computed tomography (CT) methods, originally developed for medical applications [6], are widely used for fluid density reconstruction. Two major categories of tomographic reconstruction algorithms are available: direct and iterative methods. Direct methods rely on the analytical inverse conversion from input images to a volume, which is derived using the Fourier transform and calculus. They require dense multi-view inputs that are difficult to obtain for fluids. Therefore, iterative methods, which formulate the relationship between input images and a volume as an energy function that is minimized using iterative algorithms, are more commonly used for fluid density reconstruction.

Ihrke and Magnor [1] proposed a tomography method to reconstruct a volumetric model from multiple images of fire by minimizing the least-squares energy using the conjugate gradient method. To enhance the efficiency of reconstruction, Ihrke and Magnor [7] used an adaptive grid to reconstruct a 3D density function from its projections, which was suitable for thin smoke and flames. Atcheson et al. [8] captured 3D gas flows using the time-resolved Schlieren tomography system. Some attempts to improve sparse tomographic reconstruction have been proposed. Okabe et al. [2] used one to two input videos and augmented their tomographic approach with appearance transfer. They reconstructed an initial density volume with regular tomography and iteratively improved the density until the reconstruction satisfied additional view constraints. Eckert et al. [4] pushed the limit of the sparse tomography problem by introducing a single-view reconstruction approach for plumes. They compensated for the lack of information by using physics-based and geometric priors.

Most of these methods often required a difficult calibration step, and the reconstruction quality dropped with the decrease in the number of views. Our method focuses on reconstructing fluid density from monocular video, offering distinct advantages, including

reduced costs, a drastically simplified setup, and eliminating tedious procedures like camera calibration and synchronization.

Velocity reconstruction: Reconstructing fluid motion from captured videos is more challenging than reconstructing volumetric density, as motion is indirectly observed by tracing temporal changes in density. Two families of techniques exist for estimating the velocity field of a fluid: tracer-based and tracer-free approaches. Tracer-based approaches introduce tracers, such as particles or dye, into the fluid, and then the fluid velocity can be retrieved by tracing these tracers. Various PIV methods, including tomographic PIV [9], synthetic aperture PIV [10], structured-light PIV [11–13], and plenoptic PIV [14,15], are widely used in different fields to characterize fluid flows. However, some methods require specialized hardware and complex camera setups, as seen in [9,11]. Though PIV methods are widely used, methods like [9,11] require carefully chosen particles, specialized hardware, and complex camera setups. Tracer-free approaches, like background-oriented Schlieren tomography (BOS) [16,17], use the phase change due to refractive index differences in the fluid to track fluid flow. These approaches retrieve only either the density field or the velocity field of the fluid.

Visible-light-capturing is an alternative approach for fluid motion reconstruction. Corpetti et al. [18] used optical flow with an additional divergence-free and curl smoothness prior to regularizing 2D cloud motions. Similarly, some previous works have taken divergence-free constraints into account [19–21], yet most of them suffer from the complexity of solving higher-order regularization terms. Recent works [4,5] have coupled fluid density updating and velocity calculation to improve the temporal coherence of flow reconstruction. Our work adopts a similar coupled computation strategy to guarantee the time evolution of the fluid flow.

Differentiable rendering: Differentiable rendering is a technique that allows for the computation and propagation of derivatives of scene parameters through images. This technique has a wide range of applications, including solving analysis-by-synthesis problems and training machine learning pipelines incorporating forward-rendering processes. Loper and Black [22] introduced *OpenDR*, which approximates the backward pass of the traditional mesh-based graphics pipeline and has inspired several follow-up works [23–27]. Kato et al. [26] proposed *Neural Renderer*, an approximate gradient for rendering a mesh that enables the integration of rendering into neural networks. Instead of approximating the backward pass, some methods approximate the rasterization of the rendering process, enabling the computation of useful gradients. Liu et al. [27] introduced *Soft Rasterizer*, which provides an accurate soft approximation of the standard rasterizer. Although these approximation methods reduce rendering time, generating photorealistic images containing complex interactions of light, geometry, and materials becomes difficult. Li et al. [28] proposed the PBDR framework, which supports global illumination. However, all the methods mentioned so far are limited to surface-based light transport differentiation.

Recently, there has been a focus on improving the efficiency of volumetric differentiable rendering. Nimier-David et al. [29] introduced the radiative backpropagation approach to differentiable rendering. They observed that differentiable rendering is equivalent to the solution of a reversed light transport problem. Although this approach does not memorize intermediate states, it also has an expensive computation time, which becomes quadratic in the number of scattering events along a light path. To address this, Vicini et al. [30] proposed the path replay backpropagation approach, which achieves a computation time that is linear in the number of scattering events and further improves the performance. Although these approaches significantly accelerate performance compared to naive automatic differentiation implementations, prior works have shown that AD methods can be efficient if carefully designed. Weiss et al. [31] proposed an efficient strategy for differentiable direct volume rendering using automatic differentiation. They demonstrated that by carefully designing the backward pass and leveraging intermediate outputs from the forward rendering, AD can achieve excellent performance for differentiating rendered images with respect to volume densities. While representing an important advancement in

efficient automatic differentiation for volumetric data, calculating derivatives via AD still fundamentally relies on propagating gradients through long computational graphs. This carries inherent overhead compared to directly computing derivatives analytically. Our work builds upon these insights by deriving an analytical form of the radiance transfer equation, enabling the direct and efficient computation of radiance derivatives with respect to density. By avoiding gradient tracking and redundant operations, our proposed approach aims to further improve performance for optimizing dynamic volumetric densities, such as in fluid reconstruction tasks.

3. Methods

3.1. Overview

As illustrated in Figure 1, our computational framework is composed of two main algorithmic components: the **differentiable rendering component** (Section 3.2) and the **coupled density and velocity estimation component** (Section 3.3). The primary aim of this framework is to efficiently reconstruct a temporally coherent sequence of fluid density fields, denoted as $\{\rho\}$. This reconstruction is designed to align the density fields with the input monocular fluid images $\{I\}$ while conforming to essential physical priors.

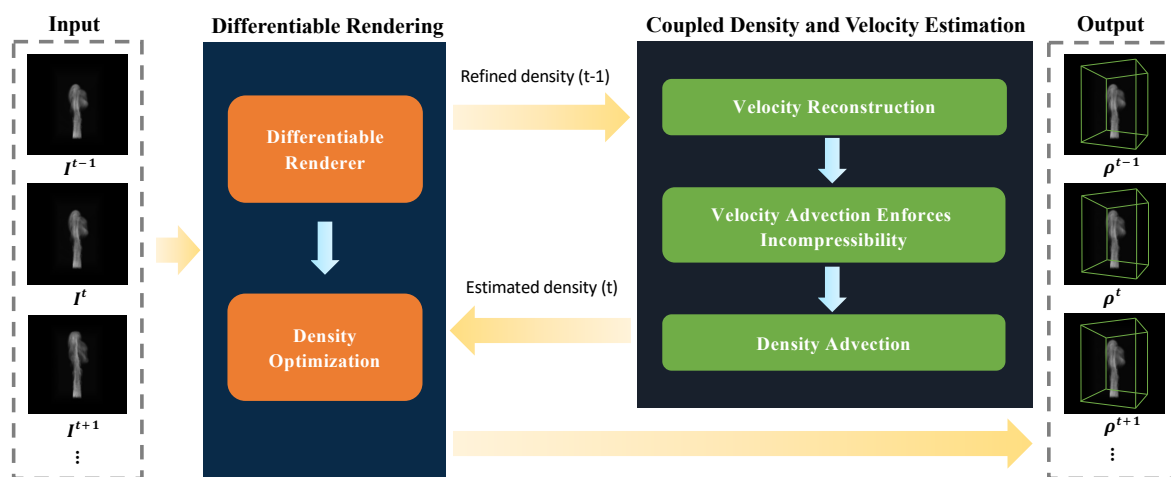


Figure 1. Our framework consists of two primary algorithm components: the **differentiable rendering component** and the **coupled density and velocity estimation component**. In this diagram, yellow and blue arrows delineate the directional flow of data between external and internal modules, respectively. The **differentiable rendering component** is utilized to refine volumetric representations extracted from a temporal sequence of fluidic images. We note that the current density fields are associated with the temporal epoch $t - 1$. These outputs subsequently serve as inputs for the **coupled density and velocity estimation component**. This component initially estimates the velocity field based on the density at temporal epoch $t - 1$. It then advects this velocity field to the subsequent temporal epoch t while enforcing incompressibility constraints. Finally, the density field is advected in accordance with the updated velocity field. The advected density volumes, along with their corresponding input images, are inputted into the differentiable renderer for final corrections.

Drawing from Algorithm 1, we will now expound on how our framework operates. Initially, we take a sequence of monocular fluid images $\{I\}$ as input and reconstruct the first two frames of fluid density ρ^1 and ρ^2 through the **differentiable rendering component** (Lines 2–3). Here, I^t refers to the image at a time t within the input video sequences, where $\{t | t \in \mathbb{N}_0, t \leq F\}$ and F denotes the total number of frames in the sequence. With the obtained initial fluid density, the subsequent frames of estimated fluid density $\tilde{\rho}^t$ are then computed using the **coupled density and velocity estimation component** (Line 5), which ensures that the fluid flows adhere to the underlying physical priors. Next, given the estimated density $\tilde{\rho}^t$ as input, the final fluid density ρ^t is corrected via our differentiable renderer using the target image I^t (Line 6).

Algorithm 1: Fluid motion reconstruction

```

1 Function FluidMotionRestruction( $\{I\}$ ):
    /* initial density estimation */
2    $\rho^1 \leftarrow \text{Reconstruction}(I^1, \rho^1)$ 
3    $\rho^2 \leftarrow \text{Reconstruction}(I^2, \rho^2)$ 
    /* coupled estimation */
4   for  $t = 3$  to  $F$  do
5      $\tilde{\rho}^t \leftarrow \text{CoupledEstimation}(\rho^{t-2}, \rho^{t-1})$ 
6      $\rho^t \leftarrow \text{Reconstruction}(I^t, \tilde{\rho}^t)$ 
7   end
8   return  $\{\rho\}$ 

```

To provide an overview of our algorithms and their relations to the formulas presented in each subsection, we include a schematic diagram in Figure 2. As depicted, our proposed method comprises five key algorithms. The core algorithm is Algorithm 1 (fluid motion reconstruction), which invokes the *Reconstruction* and *CoupleEstimation* algorithms. The *Reconstruction* algorithm reconstructs density fields from input images by leveraging our proposed differentiable renderer, to render images and compute derivatives. Equations (1)–(5), introduced in Section 3.2.1, govern the radiance calculation. Equations (6)–(10), presented in Section 3.2.2, enable radiance derivative computations.

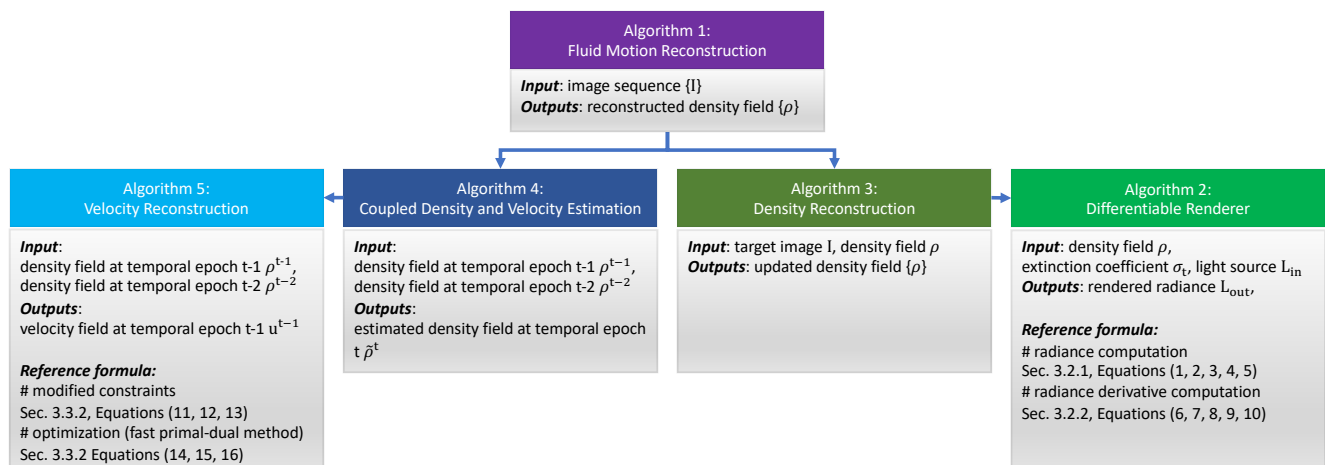


Figure 2. This figure provides a schematic diagram overviewing the relations between our proposed algorithms and the formulas presented in each subsection (Algorithm 2: Differentiable Renderer. Algorithm 3: Density Reconstruction. Algorithm 4: Coupled Density and Velocity Estimation. Algorithm 5: Velocity Reconstruction).

Algorithm 4 (coupled density and velocity estimation), introduced in Section 3.3, advects the reconstructed density fields from time step $t - 1$ and $t - 2$ to t while enforcing key physical constraints. Since it takes density fields as input, Algorithm 4 calls Algorithm 5 (velocity reconstruction) to estimate velocity fields from the previous two time steps' density fields to perform advection. Velocity reconstruction is formulated as an optimization problem with Equations (11)–(13) (Section 3.3.2) as constraints. Algorithm 5 efficiently solves this using a fast primal-dual approach governed by Equations (14)–(16). Finally, the reconstructed velocity fields are advected to density fields from time step $t - 1$ to t . This procedure couples the reconstruction of densities and velocities in a physics-constrained manner for robust fluid motion estimation.

Further technical details concerning our proposed differentiable rendering framework and the coupled density and velocity estimation will be discussed below.

Algorithm 2: Differentiable renderer

```

1 Function DiffRenderer( $\rho, \sigma_t, L_{in}$ ):
2   foreach pixel position  $(i, j) \in (ImgW, ImgH)$  do
3      $x, \omega \leftarrow GetRay(i, j)$ 
4      $x_a, x_b \leftarrow Hit(x, \omega)$ 
5      $x_u \leftarrow x_a$ 
6      $D_b = \|x_b - x\|$ 
7      $L_{ss}, \dot{L}_{ss} \leftarrow 0$ 
8      $\text{/* ray marching from } x_a \text{ to } x_b \text{ */}$ 
9     repeat
10       $\text{/* radiance computation */}$ 
11       $J_{ss}(v_{x_u}, \omega) \leftarrow \text{Equation (5)}$ 
12       $L_{ss}(x, \omega) \leftarrow L_{ss}(x, \omega) + \tau(v_{x_u}, v_{x_b})\kappa_t(v_{x_u})J_{ss}(v_{x_u}, \omega)$ 
13       $\text{/* derivative computation */}$ 
14       $\dot{J}_{ss}(v_{x_u}, \omega) \leftarrow \text{Equation (9)}$ 
15       $\dot{L}_{ri}(v_{x_u}, \omega) \leftarrow -L_{in}(\omega)\tau(v_{x_a}, v_{x_u})\dot{\kappa}_t(v_{x_u})$ 
16       $\dot{L}_{ss}(v_{x_u}, \omega) \leftarrow \tau(v_{x_u}, v_{x_b})\kappa_t(v_{x_u})\dot{J}_{ss}(v_{x_u}, \omega)\Delta d$ 
17       $+ (\dot{\tau}(v_{x_u}, v_{x_b})\kappa_t(v_{x_u}) + \tau(v_{x_u}, v_{x_b})\dot{\kappa}_t(v_{x_u}))$ 
18       $\quad \cdot J_{ss}(v_{x_u}, \omega)\Delta d$ 
19       $\dot{L}_{out} \leftarrow \dot{L}_{ri}(v_{x_u}, \omega) + \dot{L}_{ss}(v_{x_u}, \omega)$ 
20       $\text{/* recording the gradient along the path integral */}$ 
21       $gradRec(x).position.push\_back(v_{x_u})$ 
22       $gradRec(x).grad.push\_back(\dot{L}_{out})$ 
23       $x_u \leftarrow x_u + \omega \cdot \Delta d$ 
24       $D_u = \|x_u - x\|$ 
25    until  $D_u < D_b$ 
26     $L_{ri}(x, \omega) \leftarrow \text{Equation (2)}$ 
27     $L_{out}(x, \omega) \leftarrow L_{ri}(x, \omega) + L_{ss}(x, \omega)$ 
28  end
29  return  $L_{out}, gradRec$ 

```

Algorithm 3: Density reconstruction

```

1 Function Reconstruction( $I, \rho$ ):
2   repeat
3      $L_{out}, gradRec \leftarrow DiffRenderer(\rho, \sigma_t, L_{in})$ 
4      $loss \leftarrow MSE(I, L_{out})$ 
5     foreach pixel position  $(i, j) \in (ImgW, ImgH)$  do
6        $x, \omega \leftarrow GetViewpoint(i, j)$ 
7        $diff \leftarrow I[i, j] - L_{out}(x, \omega)$ 
8       for  $k = 0$  to  $gradRec(x).size()$  do
9          $v_p \leftarrow gradRec(x).position[k]$ 
10         $grad \leftarrow gradRec(x).grad[k]$ 
11         $densityGrad[v_p] \leftarrow 2 * diff * grad + densityGrad[v_p]$ 
12      end
13    end
14     $\text{/* update density field */}$ 
15    parfor each position  $v_p \in \text{voxels}$  do
16       $\rho[v_p] \leftarrow \rho[v_p] - LR * densityGrad[v_p]$ 
17    end
18  until  $loss < minLoss$ 
19  return  $\{\rho\}$ 

```

Algorithm 4: Coupled density and velocity estimation

```

1 Function CoupledEstimation( $\rho^{t-1}, \rho^{t-2}$ ):
2    $u^{t-1} \leftarrow \text{VelRecon}(\rho^{t-2}, \rho^{t-1})$ 
3    $\tilde{u}^t \leftarrow \Pi_{\text{DIV}}(\text{advect}(u^{t-1}, u^{t-1}))$ 
4    $\tilde{\rho}^t \leftarrow \text{advect}(\rho^{t-1}, \tilde{u}^t)$ 
5   return  $\tilde{\rho}^t$ 

```

Algorithm 5: Velocity reconstruction

```

1 Function VelRecon( $\rho^{t-1}, \rho^{t-2}$ ):
2   for  $k=0$  to  $\text{maxIter}$  do
3      $X_{k+1} \leftarrow X_k + \sigma Y_k - \sigma \text{prox}_{f, \sigma}(\frac{1}{\sigma} X_k + Y_k)$ 
4      $u_{k+1}^{t-1} \leftarrow \text{prox}_{g, \frac{1}{\gamma}}(u_{k+1}^{t-1} - \gamma X_{k+1})$ 
5      $Y_{k+1} = u_{k+1}^{t-1} + \theta(u_{k+1}^{t-1} - u_k^{t-1})$ 
6      $r_{k+1} = u_{k+1}^{t-1} - u_k^{t-1}$ 
7      $\epsilon = \sqrt{3}\epsilon^{\text{abs}} + \epsilon^{\text{rel}} \|u_{k+1}^{t-1}\|$ 
8     if  $\|r_{k+1}\| \leq \epsilon$  then
9       break;
10    end
11  end
12  return  $u_{k+1}^{t-1}$ 

```

3.2. Differentiable Rendering

A fluid is a participating medium that can scatter or absorb light, altering its direction and intensity as it passes through. Modeling the behavior of light in a fluid can be difficult and computationally expensive, especially when multiple scattering events are involved. Many studies have addressed these challenges by assuming simplified lighting conditions or uncluttered backgrounds in the fluid reconstruction process. This approach helps minimize interference with accuracy. Our work follows this trend and posits the following hypotheses as a result:

- We assume that the single-scattering model is sufficient for rendering the rough shape of the fluid volume. Therefore, we ignore the radiance contribution of multiple-scattering radiance and emitted radiance.
- Our method is limited to pure volume scenes, which reduces the complexity of derivative computation caused by visibility.

We now depict the technical details of our differentiable rendering framework. First, we introduce the participating media rendering based on single scattering (Section 3.2.1). We then derive the differentiation form of the RTE (Section 3.2.2), which can be directly used to compute the radiance derivative with respect to density. We finally present the algorithm of fluid density reconstruction via our proposed differentiable renderer (Sections 3.2.3 and 3.2.4).

3.2.1. Participating Media Rendering

In order to lay the foundation for differentiating the radiative transfer equation (RTE), we will briefly review light transport in participating media and introduce the RTE based on the single-scattering model. When light passes through a participating media, it is attenuated along the direction ω and ultimately reaches the viewpoint x with the total radiance $L_{\text{out}}(x, \omega)$. As noted earlier, we assume that the light transport within the volume

is governed solely by single scattering. This process can be mathematically modeled by the RTE, whose integral form is expressed as follows:

$$L_{out}(x, \omega) = L_{ri}(x, \omega) + L_{ss}(x, \omega). \quad (1)$$

where L_{out} comprises the reduced incident radiance L_{ri} and single-scattering radiance L_{ss} . The reduced incident radiance $L_{ri}(x, \omega)$ describes the ambient light $L_{in}(\omega)$ that arrives directly at viewpoint x along direction ω with attenuation by the participating medium. The single-scattering radiance $L_{ss}(x, \omega)$ represents that the radiance of light has scattered only once before arriving at viewpoint x along the direction ω , which is computed by integrating radiance contributions along the corresponding view ray. Before detailing the expressions of the radiance, the following notations need to be predefined.

Let us denote the volume density of the participating medium at position x as $\rho(x)$, and let the optical coefficient be defined as $\kappa_i(x)$, where $\kappa_i \in \{\kappa_a, \kappa_s, \kappa_t\}$ indicate the absorption, scattering, and extinction coefficient, respectively. The extinction coefficient is $\kappa_t(x) = \kappa_a(x) + \kappa_s(x)$, and the scattering albedo is $\Omega = \kappa_s(x)/\kappa_t(x)$. The expressions of L_{ri} and L_{ss} are as follows:

$$L_{ri}(x, \omega) = L_{in}(\omega)\tau(x_a, x), \quad (2)$$

$$L_{ss}(x, \omega) = \int_{D_b}^{D_a} \tau(x_u, x_b)\kappa_t(x_u)J_{ss}(x_u, \omega)du, \quad (3)$$

where D_a and D_b are the distances from the viewpoint x to the volume boundary points x_a and x_b , respectively. x_u is an arbitrary position inside the participating medium volume. $\tau(x_0, x_1)$ is the transmittance from x_0 to x_1 , which is computed as

$$\tau(x_0, x_1) = e^{-\int_{D_1}^{D_0} \kappa_t(x_u)du}, \quad (4)$$

where x_0 and x_1 are arbitrary positions inside the participating medium volume, and D_0 and D_1 are the distances from viewpoint x to x_0 and x_1 , respectively. As both τ and L_{ss} are expressed in continuous forms, they cannot be directly calculated in computers. We derive their discrete forms to facilitate computer program implementation in Section 3.2.2.

To compute L_{ss} , J_{ss} is required. The single-scattering term J_{ss} represents the reduced incident radiance, with its first scattering interaction occurring at the position v_p of the voxel grid:

$$J_{ss}(v_p, \omega) = \frac{\Omega}{4\pi} \int_{\mathbb{S}^2} L_{ri}(v_p, \omega')p(\omega, \omega')d\omega', \quad (5)$$

where $\mathbb{S}^2$ indicates a solid angle over the unit sphere. Due to Equation (5), we need to calculate an integration over the unit sphere $\mathbb{S}^2$, while Monte Carlo (MC) integration provides a means for estimating J_{ss} . However, for more efficiency, we use distinct tricks for different illuminations. For environment map lighting, we utilize importance sampling to estimate J_{ss} efficiently. For simple point or parallel lights, J_{ss} can be directly calculated without MC as the lighting directions and radiance are known. Rather than relying on MC estimation for the complete differentiable rendering, we use it only to efficiently estimate J_{ss} where needed. The phase function $p(\omega, \omega')$ describes the directional scattering distribution of the participating media.

The necessary formulas for forward rendering have been introduced. In the following section, we will derive the differential form of the RTE. This form can be integrated into the differentiable rendering process to calculate the derivative of the RTE directly.

3.2.2. Differential Form of RTE

Here, we introduce the differentiation of the RTE based on the single-scattering model for computing the density derivative. The differentiation of Equation (1) can be expanded into two terms, namely the differentiation of the reduced incident radiance $\dot{L}_{ri}$ and the

differentiation of the single-scattering radiance $\dot{L}_{ss}$. The differential form of the RTE is therefore

$$\dot{L}_{out}(x, \omega) = \dot{L}_{ri}(x, \omega) + \dot{L}_{ss}(x, \omega). \quad (6)$$

Since our work only focuses on the fluid reconstruction task, the derivative $\dot{L}_{out}$ can be confined as $\dot{L}_{out} = \frac{dL_{out}}{d\rho}$ (w.r.t. density ρ only).

Derivation of $\dot{L}_{ri}$. As shown in Equation (6), the first component of the differential form of the RTE is $\dot{L}_{ri}$, which can be derived by differentiating L_{ri} of Equation (2):

$$\dot{L}_{ri}(x, \omega) = L_{in}(\omega) \dot{\tau}(x_a, x). \quad (7)$$

In order to compute $\dot{L}_{ri}$, $\dot{\tau}$ is required. Differentiating the transmittance τ of Equation (4) yields

$$\dot{\tau}(v_{x_0}, v_{x_1}) = -\tau(v_{x_0}, v_{x_1}) \sum_{p=x_0}^{x_1} \dot{\kappa}_t(v_p) \Delta d, \quad (8)$$

where $\dot{\kappa}_t = \sigma_t, \sigma_t$ is the extinction cross-section, v_p is a voxel position of a uniform 3D voxel grid, and Δd is the length of the sampling step.

Derivation of $\dot{L}_{ss}$. The second component of the differential form of the RTE is $\dot{L}_{ss}$. Given the single-scattering radiance assumption, where light scattering occurs only once before arriving at viewpoint x , differentiating L_{ss} of Equation (3) yields

$$\begin{aligned} \dot{L}_{ss}(x, \omega) &= \sum_{p=x_a}^{x_b} (\dot{\tau}(v_p, v_{x_b}) \kappa_t(v_p) + \tau(v_p, v_{x_b}) \dot{\kappa}_t(v_p)) J_{ss}(v_p, \omega) \Delta d \\ &+ \sum_{p=x_a}^{x_b} \tau(v_p, v_{x_b}) \kappa_t(v_p) \dot{J}_{ss}(v_p, \omega) \Delta d. \end{aligned} \quad (9)$$

To compute $\dot{L}_{ss}$, $\dot{J}_{ss}$ is required. Differentiating J_{ss} of Equation (5) yields

$$\dot{J}_{ss}(v_p, \omega) = \frac{\Omega}{4\pi} \int_{\mathbb{S}^2} \dot{L}_{ri}(v_p, \omega') p(\omega, \omega') d\omega'. \quad (10)$$

The derivation of Equation (10) requires taking the derivative of an integral. To accomplish this, we can utilize the Reynolds transport theorem [32,33], which is a widely used method for computing derivatives of hydrodynamic integral equations. However, since our work is concerned with purely participating media volumes, we can assume that L_{ri} exhibits continuity in $\mathbb{S}^2$, and thus the derivative of the interface can be disregarded.

3.2.3. Differentiable Renderer

The cornerstone of our differentiable renderer lies in the radiance transport equation and its differential form. Unlike previous works, we derive an analytical differential expression for the radiance transport equation, thereby enabling the direct computation of density derivatives without the exclusive reliance on automatic differentiation. Our algorithmic approach is delineated in Algorithm 2, where we employ ray marching to both render images and compute the associated derivatives. Specifically, the algorithm calculates the final radiance L_{out} while simultaneously storing the requisite gradients for subsequent density field updates. Lines 9–13 of the algorithm illustrate that we execute both forward and backward passes concurrently. This dual-pass approach facilitates the reuse of J_{ss} during the backpropagation phase.

It is worth noting that Equations (5) and (9), appearing in lines 9 and 11 of Algorithm 2, are essential for performing integration over the unit sphere. While a straightforward Monte Carlo method could be employed for this estimation, it would entail computationally expensive calculations. To mitigate this, we adopt specialized techniques tailored to different types of illuminations. As elaborated in Section 3.2.1, we utilize importance

sampling for environment map illuminations and direct computation for point/parallel light illuminations when calculating J_{ss} .

Furthermore, as depicted in lines 14–15, *gradRec* is used for recording the gradients associated with the sampling positions. Rather than saving gradients for all voxels, we implement a sparse structure *gradRec* that only stores gradients along sampled ray paths. When marching rays through the volume, we push gradients onto the stack as intersections occur along the ray (Algorithm 2 lines 14–15). When a density field update finishes (Algorithm 3), the stack is popped, releasing memory. Our optimized approach reduces memory usage from $O(N^3)$ to $O(RN)$ for R rays of length N . We only maintain gradients along active ray paths rather than the entire volume grid. These computational strategies, in conjunction with our analytical radiance derivatives, culminate in a highly efficient differentiable renderer, thereby facilitating fluid density optimization with enhanced computational efficacy.

3.2.4. Density Reconstruction

Building upon our proposed differentiable rendering framework, we present the reconstruction of the density field using our framework and provide details in Algorithm 3. We begin by computing the loss function of the target and primal images, followed by reading the density gradient from the recorded sampling positions of the generated ray. The gradients of density with respect to the loss value are computed and saved in the volume data *densityGrad*. Finally, we iterate through and update each voxel value of the density field, and the memory of the structure *gradRec* and *densityGrad* are released.

3.3. Coupled Density and Velocity Estimation

Most previous works for fluid reconstruction directly updated the fluid density in each time step using tomographic methods from input video sequences. While this works well for systems with a large number of cameras, the reconstruction quality drops significantly when the number of cameras (i.e., projections) reduces, making the tomography problem formulation ill-posed. To solve this ill-posed problem and improve the reconstruction quality, we apply a coupled estimated strategy, which makes the efficient reconstruction of realistic and smooth fluid flow possible. A possible solution is coupled density and velocity updating, which was developed by Eckert et al. [4], yet their method was extremely time-consuming and impractical, with an average reconstruction time of up to 50–60 min per frame. We therefore propose a simplified strategy for coupled density and velocity estimation that is highly efficient, making interactive fluid motion reconstruction possible.

3.3.1. Coupled Estimation

Our approach is to predict the density field $\tilde{\rho}^t$ for the current time step t based on the given states of time $t - 1$ and time $t - 2$. The details of our coupled density and velocity estimation are depicted in Algorithm 4. To proceed, we first estimate velocity u^{t-1} based on the densities of time $t - 1$ and time $t - 2$ according to Algorithm 5. We then predict the velocity and density for the current time step t based on physical priors, i.e., the density and velocity transport over the time steps and the incompressibility of the velocity. Note that incompressibility is ensured by projecting the velocities onto the space of divergence-free velocities, called Π_{DIV} . We advect velocity u^{t-1} with itself and ensure its incompressibility to create a velocity guess $\tilde{u}^t$. The density guess $\tilde{\rho}^t$ is created by advecting the density ρ^{t-1} forward with the velocity guess $\tilde{u}^{t-1}$ (lines 3–4 of Algorithm 4).

3.3.2. Velocity Reconstruction

While the last two steps shown in lines 3–4 of Algorithm 4 are similar to those in [4], our velocity reconstruction process is different. Notably, we do not apply *depth regularization* to gain control over the motion in depth. Because this regularized term breaks the sparse property of the matrix, the optimization problem cannot be solved efficiently.

To improve the efficiency of velocity reconstruction, we propose a modified constraint as follows:

$$\min_{u^{t-1}} f(u^{t-1}) = \left\| \frac{\rho^{t-1} - \rho^{t-2}}{\Delta t} + u^{t-1} \cdot \nabla \rho^{t-1} \right\|^2 + E_{smooth}(u^{t-1}) + E_{kinetic}(u^{t-1}) \quad (11)$$

$$\text{subject to } \nabla \cdot u^{t-1} = 0. \quad (12)$$

In Equation (11), the first component is the constraint from an optical flow problem $\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho u) = 0$. In order to further confine our solution, we add both smoothness and kinetic energy (Tikhonov) regularized terms to the velocity [34]. These regularized terms are expressed as

$$\begin{aligned} E_{smooth}(u) &= \alpha \|\nabla u\|^2, \\ E_{kinetic}(u) &= \beta \|u_{t-1}\|^2, \end{aligned} \quad (13)$$

where α, β are the weights of these regularized terms, respectively. $E_{smooth}(u)$ is a smoothness regularizer controlled by weight α . It penalizes large changes/gradients in the velocity field by minimizing the $L2$ norm of ∇u . This smoothness constraint helps reduce noise and discontinuities in the estimated velocities. $E_{kinetic}(u)$ is a kinetic energy or Tikhonov regularizer weighted by β . It minimizes the $L2$ norm of the velocity values themselves. This constraint discourages large velocity magnitudes and excessive motion. Together, these regularization terms help confine the velocity solution space to plausible flows that are smooth and controlled in magnitude. The relative weights α and β allow the balancing of the smoothing and kinetic energy constraints as needed for the data.

In order to compute the velocity, as described in Equations (11) and (12), we make use of the fast primal-dual method (PD) for convex optimization as introduced by Chambolle and Pock [35]. Instead of solving for the whole complex optimization problem at once, we split the complex optimization problem into two manageable components, where proximal operators act as efficient solvers for each subproblem. Iterative variable updates ensure that the solution converges to the optimal value of the problem in Equations (11) and (12). The simplified PD updates are given by

$$\begin{aligned} X_{k+1} &= X_k + \sigma Y_k - \sigma \mathbf{prox}_{f, \sigma} \left(\frac{1}{\sigma} X_k + Y_k \right), \\ u_{k+1}^{t-1} &= \mathbf{prox}_{g, \frac{1}{\gamma}} (u_{k+1}^{t-1} - \gamma X_{k+1}), \\ Y_{k+1} &= u_{k+1}^{t-1} + \theta (u_{k+1}^{t-1} - u_k^{t-1}), \end{aligned} \quad (14)$$

where $\{\sigma, \gamma, \theta\}$ are parameters that affect convergence; k is the iteration number; X, Y are helper variables; and $\mathbf{prox}$ is the proximal operator for each subproblem. f of $\mathbf{prox}_{f, \sigma}$ targets Equation (11), where velocity u^{t-1} is unknown, and the object function f (Equation (11)) is solved by least squares. The optimization equation of $\mathbf{prox}_{f, \sigma}$ is given by

$$\mathbf{prox}_{f, \sigma}(\xi) = \underset{u^{t-1}}{\operatorname{argmin}} f(u^{t-1}) + \frac{\sigma}{2} \|u^{t-1} - \xi\|^2, \quad (15)$$

where ξ is a generic variable. The proximal operator $\mathbf{prox}_{g, \frac{1}{\gamma}}$ targets Equation (12), and this constraint assumes that the velocity field is incompressible:

$$\mathbf{prox}_{g, \frac{1}{\gamma}}(\xi) = \Pi_{\text{DIV}}(\xi). \quad (16)$$

The process of velocity reconstruction is shown in Algorithm 5. Lines 3–5 are the PD updates, and lines 6–10 check the convergence of the iterative computation.

3.4. Implementation Details

We implemented our proposed approach using a combination of Python and C++. The differentiable renderer was developed in Python, leveraging the Taichi programming language [36,37] for parallelization and high-performance computing on GPUs. Taichi was chosen due to its excellent compatibility with existing deep learning frameworks, allowing integration with neural networks or other models. Additionally, Taichi provided high-efficiency parallel primitives and differentiable programming constructs that significantly accelerated the core computations in our renderer compared to standard Python.

The coupled density and velocity estimation component was implemented in C++ for computational efficiency and numeric precision. We interfaced with this C++ module from the Python renderer using the pybind11 library. Pybind11 enabled us to seamlessly pass data between the Python and C++ code.

We evaluated our method on a workstation with an Intel Xeon E5-2620 v4 CPU and an Nvidia GeForce RTX 2080 Ti GPU. All results and timings provided were obtained by executing the code on the GPU hardware, leveraging parallelized implementations for performance.

4. Result

4.1. Evaluation

We conducted a rigorous validation of our method through both qualitative visualizations and quantitative analyses. Our validations focused primarily on the claimed contributions of the proposed differentiable renderer and reconstruction framework. As the accuracy of the estimated gradients and computational performance are critical metrics for evaluating a differentiable renderer, we first validated these two aspects. Subsequently, we conducted ablation studies to assess the reconstruction capabilities of our differentiable renderer and the efficacy of the transport constraints. Furthermore, given the significant impact of the estimated velocity on reconstructed fluid motions, we analyzed comparisons between our velocity estimates and the ground-truth data. Finally, we compared our approach against previous methods through both qualitative visualizations and quantitative efficiency metrics.

Validation of gradient computation. To validate the effectiveness of our proposed differentiable renderer, we optimized a density field and analyzed the evolution of the calculated derivatives. The initial density configuration was in the shape of a rabbit, which was optimized by updating the density field based on a target smoke image. The image sequence in Figure 3 shows the images derived throughout the optimization iterations.

Initially, the density inside the rabbit shape needed to decrease, while the density in the smoke region had to increase to match the target. Since the smoke density was thinner than the rabbit density, the derivatives for the smoke density diminished faster, disappearing after around 30 iterations. In contrast, significant derivatives for the rabbit density persisted from iterations 0 to 50. By iteration 99, the optimization had converged to the target image. Overall, Figure 3 demonstrates that the density field derivatives followed the expected variations throughout the optimization process.

Performance validation of our differentiable renderer. Since our differentiable renderer significantly differed from previous methods based on differential volumetric path tracing in implementation, different factors impacted its performance. Previous physics-based differentiable renderers heavily relied on the Monte Carlo method to estimate the radiance and its derivatives. Hence, these methods' performance was affected by the number of samplings. Unlike these methods, the volumetric data size significantly impacted our renderer's performance. To characterize this relationship, validations were performed on the rabbit scene in Figure 3 under four resolutions: 32^3 , 64^3 , 128^3 , and 256^3 voxels. Table 1 presents the average runtime and GPU memory usage across 100 iterative optimizations at different volumetric data resolutions. As anticipated, the runtime of our renderer demonstrated an increasing trend with higher-resolution volumetric data. The recorded runtimes for the resolutions of 32^3 , 64^3 , 128^3 , and 256^3 voxels were 9.4 ms, 17.6 ms, 82.4 ms, and

545.1 ms, respectively. Despite the increase in runtime, our method maintained real-time performance even at higher resolutions, which could be crucial for interactive applications and dynamic scenes.

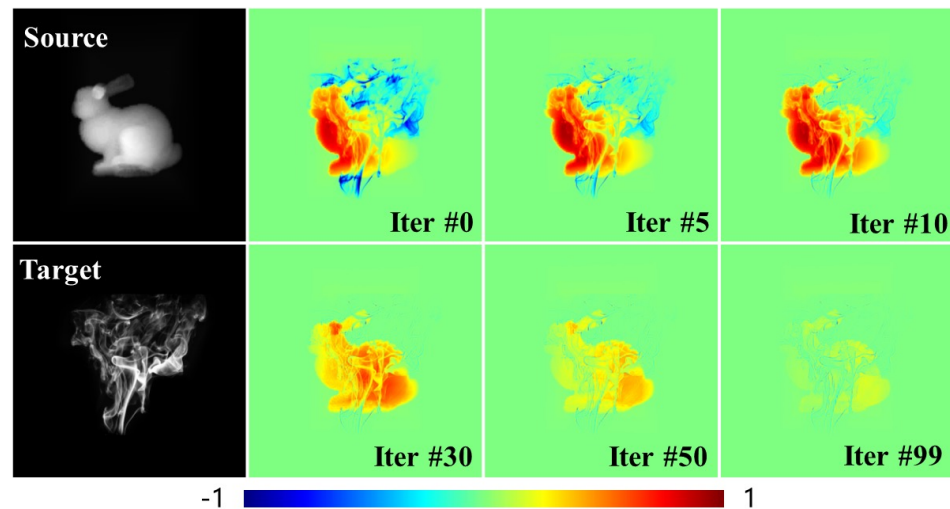


Figure 3. Derivative analysis of density optimization. The initial volume was configured in the shape of a rabbit, with the density field optimized to match a target smoke image. The derivative images show an increasing density in blue and decreasing density in red. This color mapping accurately captures the evolution of the derivatives throughout the entire density optimization process.

The memory usage of our differentiable renderer was another essential aspect of its performance. Similar to the runtime analysis, we observed a direct correlation between the resolution of the volumetric data and the GPU memory usage. As the resolution increased, so did the memory requirement. The memory usages for the resolutions 32^3 , 64^3 , 128^3 , and 256^3 voxels were 612 MB, 648 MB, 851 MB, and 2813 MB, respectively. This increase was consistent with the greater amount of data that needs to be processed and stored in higher-resolution volumetric data. Despite this growth, our method remained memory-efficient even at the highest resolution, which was a testament to its practical viability for a wide range of applications.

Table 1. Performance validation at different volumetric data resolutions. The average runtime and GPU memory utilization across 100 optimization iterations are presented for varying volumetric data resolutions. We observed increasing runtimes and memory usage with higher resolutions. However, our differentiable renderer maintained real-time performance and memory efficiency even at elevated resolutions.

Volumetric Resolution	Average Runtime (ms)	Memory Usage (MB)
32^3	9.4	612
64^3	17.6	648
128^3	82.4	851
256^3	545.1	2813

Evaluation of density reconstruction. To evaluate the reconstruction accuracy of our proposed method, we conducted an experiment using synthetic fluid motion data. The ground-truth fluid flows were generated using the *Mantaflow* simulator [38]. The density fields were then rendered into a monocular video sequence using *Mitsuba 3* [39]. This synthetic video served as the target for guiding the reconstruction using our approach. To enhance the shape constraints, we re-used the monocular video to constrain the reconstruction from orthogonal angles (e.g., angle 0° and 90°). As shown in Figure 4, we conducted an ablation experiment to study the effectiveness of the proposed *coupled density and velocity*

estimation component. Qualitatively, the physically constrained result in Figure 4c more faithfully captured the true fluid evolution compared to the unconstrained reconstruction in Figure 4b. The results demonstrated that our proposed framework could accurately reconstruct fluid motions from monocular video while maintaining physically consistent temporal evolutions.

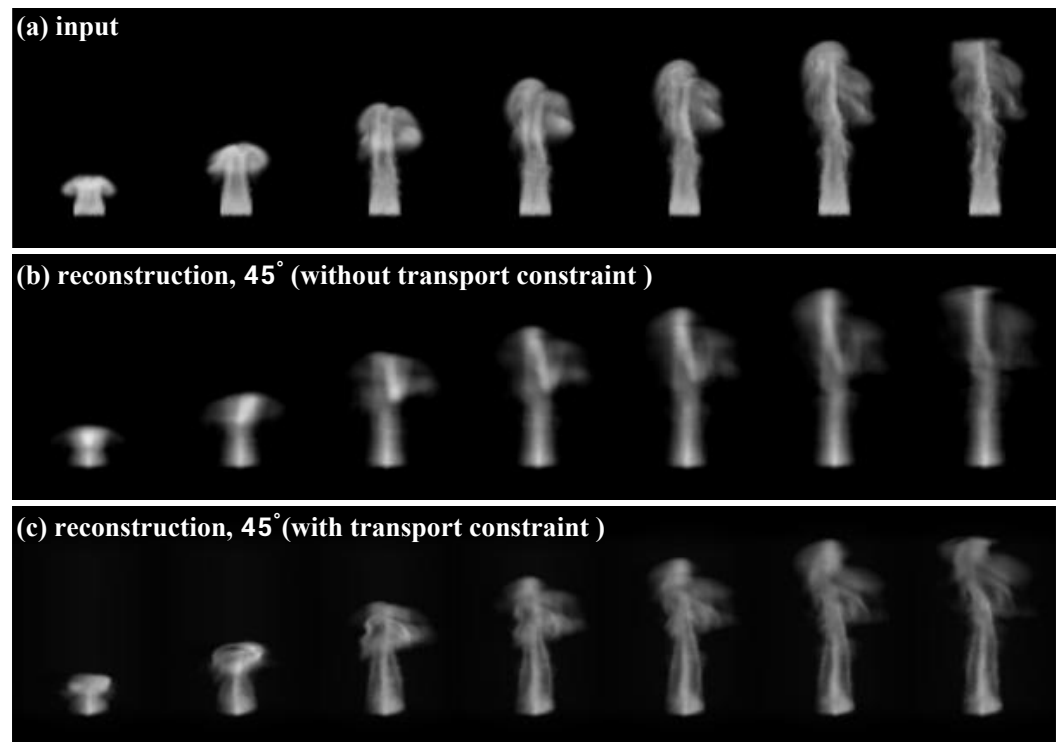


Figure 4. Validation of the physical constraint. (a) The input image sequence used to constrain the shape from orthogonal views. (b) The reconstructed results without the transport constraint. (c) The proposed constraint. We observed that adding the transport constraint significantly improved the reconstruction quality and temporal consistency.

Evaluation of the velocity estimation. The evaluation of the velocity estimation is of utmost importance in reconstructing fluid motions, as it significantly impacts the reconstruction quality and temporal consistency of fluid motions. We compared estimated velocity fields obtained from reconstructing Figure 4 with the ground truth. As Figure 5 shows, the estimated velocity was somewhat minimal at the base because no specialized processing was implemented for the inflow zones. Despite the inherent challenges in precisely estimating inflow velocity from only a monocular video and transport constraints, the overall velocity field could still be coarsely estimated to a reasonable degree. These findings validated the feasibility of estimating fluid velocities from limited visual information while preserving efficiency. In applications where computational efficiency is not the primary concern, one potential area of investigation could be devising techniques to better model the inflow velocity profile, which would likely improve estimated accuracy.

Comparison to previous work. While we demonstrated our method's fluid motion reconstruction capabilities above, further comparative evaluations were essential to fully validate its abilities against previous approaches. To this end, we conducted fluid motion reconstruction from the same initial configuration using both our method and Franz et al.'s approach [5]. Our method followed similar overall workflows to Franz et al.'s approach but had key differences in the following critical aspects: First, we proposed an efficient differentiable renderer for participating media, which was successfully integrated into our framework. Second, our method utilized an improved joint density and velocity estimation strategy. These improvements aimed to enable efficient and temporally consistent

fluid motion reconstruction. As depicted in Figure 6, both our method and Franz et al.'s method achieved a high similarity between visualizations and the ground truth. This demonstrated that our framework had a comparable fluid motion reconstruction capability to Franz et al.'s method.

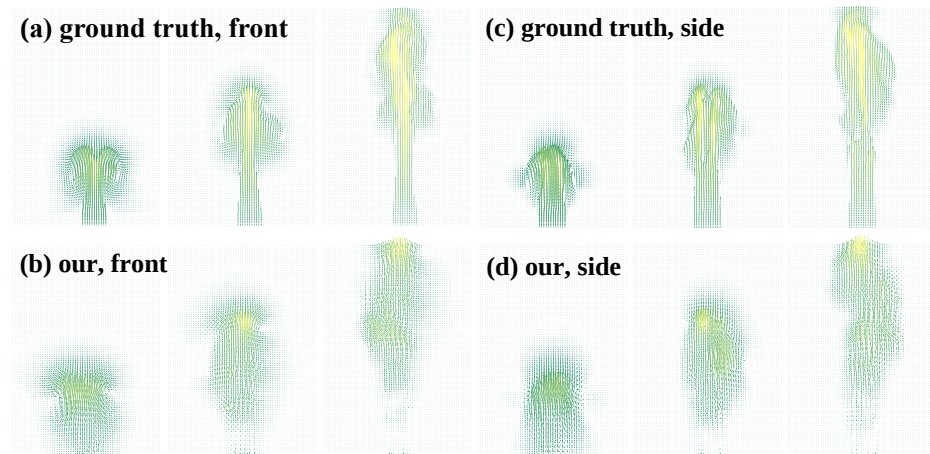


Figure 5. Validating the velocity estimation. We compared center-slice velocities from our estimation to the ground truth at frames $t = 70, 110$, and 147 along the front and side views. The results exhibited some bias near the base, as no specialized inflow treatment was implemented. However, overall, the estimated velocities reasonably matched the ground truth, suggesting that our method could effectively and reliably characterize the fluid flow dynamics.

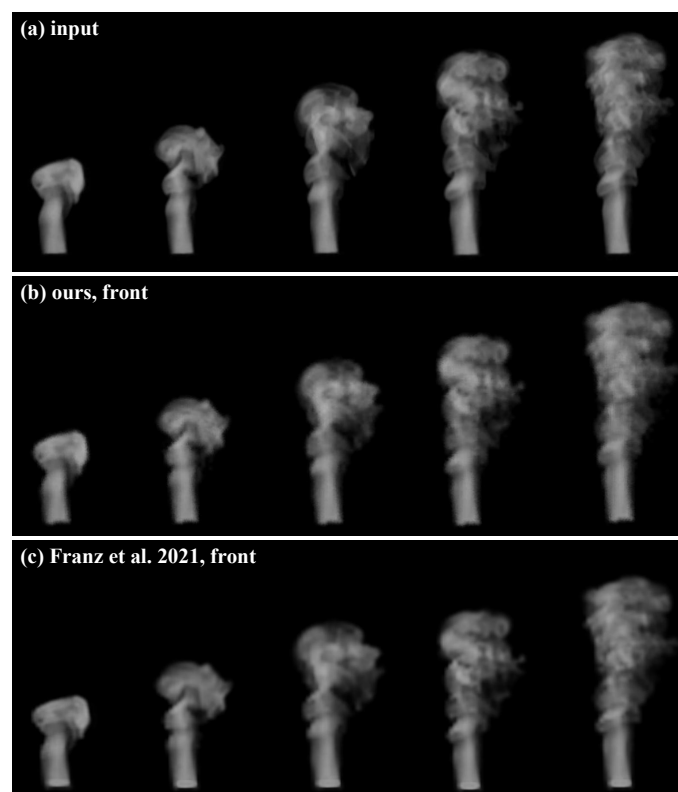


Figure 6. Orthogonal view (angle 0° and angle 90°) reconstruction. The results demonstrate that our approach could produce highly realistic fluid motion that matched the ground truth closely. Our comparison with Franz et al. [5]'s method revealed that our method's reconstructed ability was comparable to theirs.

Furthermore, run-time efficiency is an important indicator to evaluate a reconstruction framework. To validate the efficiency of our proposed framework, we compared the average runtime of our method against Franz et al.’s method for reconstructing fluid densities from video inputs. As shown in Table 2, we evaluated three volumetric resolutions of $64 \times 96 \times 64$, $128 \times 192 \times 128$, and $256 \times 288 \times 256$ voxels.

For the $64 \times 96 \times 64$ resolution, our complete reconstruction time per frame was 6.13 s, over $59\times$ faster than Franz et al.’s 363 s. More significantly, our differentiable rendering time was just 0.08 s, demonstrating a speedup of over $870\times$ compared to their 69.6 s for this key computational stage.

For the higher resolutions of $128 \times 192 \times 128$ and $256 \times 288 \times 256$, our method outperformed Franz et al.’s by factors of approximately 172 and 173, respectively, in terms of average reconstruction time. Furthermore, our differentiable rendering stage exhibited speedup factors of around 2655 and 1676 compared to Franz et al.’s methodology.

Table 2. A comparison of efficiency with Franz et al.’s works. “Time” indicates the average reconstruction time per frame.

Method	Resolution	DR Time	Total Time
Franz et al. [5]	$64 \times 96 \times 64$	1.16 m	6.06 m
	$128 \times 192 \times 128$	5.31 m	25.2 m
	$256 \times 288 \times 256$	21.8 m	61.6 m
Our method	$64 \times 96 \times 64$	0.08 s	6.13 s
	$128 \times 192 \times 128$	0.12 s	8.75 s
	$256 \times 288 \times 256$	0.78 s	21.31 s

4.2. Reconstruction from Real-World Data

Reconstructing fluid flows from real-world video captures presents significant challenges due to ambient interference and computationally intensive processing requirements. Previous approaches have required several minutes per frame for reconstruction, severely limiting applicability in interactive contexts. Our proposed method aimed to address this limitation by enabling the efficient modeling of fluid motions from monocular videos of real fluids. To validate the practical applicability of our method, we reconstructed fluid motions from a monocular video capturing real fluid flows from a fixed viewpoint. As depicted in Figure 7, the reconstructed results were rendered by *Mitsuba 3* [39] from a perspective offset by 20 degrees away from the frontal view. Our results demonstrated that our method could reconstruct fluid motions realistically with high shape fidelity compared to the input video, even for complex backgrounds. This showcased the potential of our method as a useful and efficient tool for artistic creation and content generation.

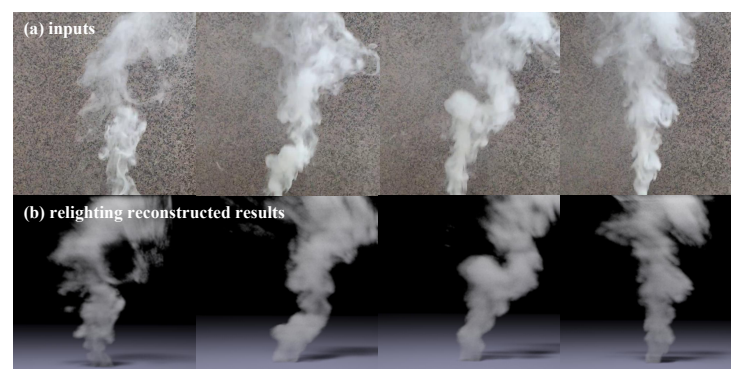


Figure 7. Reconstruction from real-world data. We utilized a monocular video captured from real-world fluid flows as input into our method, reconstructing at a volumetric resolution of $64 \times 96 \times 64$. As illustrated in (b), the reconstructed flows rendered via *Mitsuba 3* demonstrated that our proposed approach could reconstruct physically plausible fluid motions from real-world video.

5. Conclusions

We introduced a novel method to efficiently reconstruct fluid density fields from monocular video data. The key contribution was our newly proposed efficient differentiable rendering framework for participating media and the improved transport constraints. Our approach outperformed previous methods by considering both computational efficiency and visual quality. Looking ahead, worthwhile future work includes reducing current limitations, such as incorporating the differentiable rendering of surface models and developing advanced sampling strategies to further improve efficiency. Overall, this work represents an important advance in performing monocular video-based fluid reconstructions, providing a valuable new tool for digital content creation.

Author Contributions: Conceptualization, Y.C.; Methodology, Y.C.; Software, Y.C. and Q.Z.; Validation, Q.Z.; Formal analysis, Y.C.; Investigation, Q.Z. and X.L.; Data curation, Q.Z.; Writing—original draft, Y.C.; Writing—review & editing, X.L.; Visualization, Q.Z.; Supervision, X.L.; Project administration, X.L.; Funding acquisition, X.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China, grant number 62272019.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ihrke, I.; Magnor, M. Image-based tomographic reconstruction of flames. In Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer Animation, Grenoble, France, 27–29 August 2004; pp. 365–373.
2. Okabe, M.; Dobashi, Y.; Anjyo, K.; Onai, R. Fluid volume modeling from sparse multi-view images by appearance transfer. *ACM Trans. Graph. (TOG)* **2015**, *34*, 1–10. [\[CrossRef\]](#)
3. Eckert, M.L.; Um, K.; Thuerey, N. ScalarFlow: A large-scale volumetric data set of real-world scalar transport flows for computer animation and machine learning. *ACM Trans. Graph. (TOG)* **2019**, *38*, 1–16. [\[CrossRef\]](#)
4. Eckert, M.L.; Heidrich, W.; Thuerey, N. Coupled Fluid Density and Motion from Single Views. *Comput. Graph. Forum* **2018**, *37*, 47–58. [\[CrossRef\]](#)
5. Franz, E.; Solenthaler, B.; Thuerey, N. Global Transport for Fluid Reconstruction with Learned Self-Supervision. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 1632–1642.
6. Kak, A.C.; Slaney, M. *Principles of Computerized Tomographic Imaging*; SIAM: Philadelphia, PA, USA, 2001.
7. Ihrke, I.; Magnor, M. Adaptive grid optical tomography. *Graph. Model.* **2006**, *68*, 484–495. [\[CrossRef\]](#)
8. Atcheson, B.; Ihrke, I.; Heidrich, W.; Tevs, A.; Bradley, D.; Magnor, M.; Seidel, H.P. Time-resolved 3d capture of non-stationary gas flows. *ACM Trans. Graph. (TOG)* **2008**, *27*, 1–9. [\[CrossRef\]](#)
9. Elsinga, G.E.; Scarano, F.; Wieneke, B.; van Oudheusden, B.W. Tomographic particle image velocimetry. *Exp. Fluids* **2006**, *41*, 933–947. [\[CrossRef\]](#)
10. Belden, J.; Truscott, T.T.; Axiak, M.C.; Techet, A.H. Three-dimensional synthetic aperture particle image velocimetry. *Meas. Sci. Technol.* **2010**, *21*, 125403. [\[CrossRef\]](#)
11. Xiong, J.; Idoughi, R.; Aguirre-Pablo, A.A.; Aljedaani, A.B.; Dun, X.; Fu, Q.; Thoroddsen, S.T.; Heidrich, W. Rainbow particle imaging velocimetry for dense 3D fluid velocity imaging. *ACM Trans. Graph. (TOG)* **2017**, *36*, 1–14. [\[CrossRef\]](#)
12. Xiong, J.; Fu, Q.; Idoughi, R.; Heidrich, W. Reconfigurable rainbow PIV for 3D flow measurement. In Proceedings of the 2018 IEEE International Conference on Computational Photography (ICCP), Salt Lake City, UT, USA, 18–22 June 2018; pp. 1–9.
13. Aguirre-Pablo, A.; Aljedaani, A.B.; Xiong, J.; Idoughi, R.; Heidrich, W.; Thoroddsen, S.T. Single-camera 3D PTV using particle intensities and structured light. *Exp. Fluids* **2019**, *60*, 25. [\[CrossRef\]](#)
14. Fahringer, T.W.; Lynch, K.P.; Thurow, B.S. Volumetric particle image velocimetry with a single plenoptic camera. *Meas. Sci. Technol.* **2015**, *26*, 115201. [\[CrossRef\]](#)
15. Tan, Z.P.; Thurow, B.S. Time-resolved 3D flow-measurement with a single plenoptic-camera. In Proceedings of the AIAA Scitech 2019 Forum, San Diego, CA, USA, 7–11 January 2019; p. 0267.
16. Goldhahn, E.; Seume, J. The background oriented schlieren technique: Sensitivity, accuracy, resolution and application to a three-dimensional density field. *Exp. Fluids* **2007**, *43*, 241–249. [\[CrossRef\]](#)
17. Atcheson, B.; Heidrich, W.; Ihrke, I. An evaluation of optical flow algorithms for background oriented schlieren imaging. *Exp. Fluids* **2009**, *46*, 467–476. [\[CrossRef\]](#)

18. Corpetti, T.; Mémín, É.; Pérez, P. Dense estimation of fluid flows. *IEEE Trans. Pattern Anal. Mach. Intell.* **2002**, *24*, 365–380. [\[CrossRef\]](#)
19. Herlin, I.; Béréziat, D.; Mercier, N.; Zhuk, S. Divergence-free motion estimation. In Proceedings of the European Conference on Computer Vision, Florence, Italy, 7–13 October 2012; Springer: Berlin/Heidelberg, Germany, 2012; pp. 15–27.
20. Yuan, J.; Schörr, C.; Steidl, G. Simultaneous higher-order optical flow estimation and decomposition. *SIAM J. Sci. Comput.* **2007**, *29*, 2283–2304. [\[CrossRef\]](#)
21. Ruhnau, P.; Stahl, A.; Schnörr, C. Variational estimation of experimental fluid flows with physics-based spatio-temporal regularization. *Meas. Sci. Technol.* **2007**, *18*, 755. [\[CrossRef\]](#)
22. Loper, M.M.; Black, M.J. OpenDR: An approximate differentiable renderer. In Proceedings of the European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2014; Springer: Berlin/Heidelberg, Germany, 2014; pp. 154–169.
23. Chen, W.; Ling, H.; Gao, J.; Smith, E.; Lehtinen, J.; Jacobson, A.; Fidler, S. Learning to predict 3d objects with an interpolation-based differentiable renderer. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 9609–9619.
24. Genova, K.; Cole, F.; Maschinot, A.; Sarna, A.; Vlastic, D.; Freeman, W.T. Unsupervised training for 3d morphable model regression. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 8377–8386.
25. Henderson, P.; Ferrari, V. Learning to generate and reconstruct 3d meshes with only 2d supervision. *arXiv* **2018**, arXiv:1807.09259.
26. Kato, H.; Ushiku, Y.; Harada, T. Neural 3d mesh renderer. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 3907–3916.
27. Liu, S.; Chen, W.; Li, T.; Li, H. Soft rasterizer: Differentiable rendering for unsupervised single-view mesh reconstruction. *arXiv* **2019**, arXiv:1901.05567.
28. Li, T.M.; Aittala, M.; Durand, F.; Lehtinen, J. Differentiable monte carlo ray tracing through edge sampling. *ACM Trans. Graph. (TOG)* **2018**, *37*, 222. [\[CrossRef\]](#)
29. Nimier-David, M.; Speierer, S.; Ruiz, B.; Jakob, W. Radiative backpropagation: An adjoint method for lightning-fast differentiable rendering. *ACM Trans. Graph. (TOG)* **2020**, *39*, 146. [\[CrossRef\]](#)
30. Vicini, D.; Speierer, S.; Jakob, W. Path replay backpropagation: Differentiating light paths using constant memory and linear time. *ACM Trans. Graph. (TOG)* **2021**, *40*, 108. [\[CrossRef\]](#)
31. Weiss, S.; Westermann, R. Differentiable direct volume rendering. *IEEE Trans. Vis. Comput. Graph.* **2021**, *28*, 562–572. [\[CrossRef\]](#) [\[PubMed\]](#)
32. Flanders, H. Differentiation under the integral sign. *Am. Math. Mon.* **1973**, *80*, 615–627. [\[CrossRef\]](#)
33. Leal, L.G. *Advanced Transport Phenomena: Fluid Mechanics and Convective Transport Processes*; Cambridge University Press: Cambridge, UK, 2007; Volume 7.
34. Wedel, A.; Cremers, D. *Stereoscopic Scene Flow for 3D Motion Analysis*; Springer: Berlin/Heidelberg, Germany, 2011; Volume 10, p. 978.
35. Chambolle, A.; Pock, T. A first-order primal-dual algorithm for convex problems with applications to imaging. *J. Math. Imaging Vis.* **2011**, *40*, 120–145. [\[CrossRef\]](#)
36. Hu, Y.; Li, T.M.; Anderson, L.; Ragan-Kelley, J.; Durand, F. Taichi: A language for high-performance computation on spatially sparse data structures. *ACM Trans. Graph. (TOG)* **2019**, *38*, 201. [\[CrossRef\]](#)
37. Hu, Y.; Anderson, L.; Li, T.M.; Sun, Q.; Carr, N.; Ragan-Kelley, J.; Durand, F. DiffTaichi: Differentiable Programming for Physical Simulation. *arXiv* **2020**, arXiv:1910.00935.
38. Pfaff, T.; Thuerey, N. Mantaflow. 2017. Available online: <http://mantaflow.com> (accessed on 10 December 2022).
39. Jakob, W.; Speierer, S.; Roussel, N.; Nimier-David, M.; Vicini, D.; Zeltner, T.; Nicolet, B.; Crespo, M.; Leroy, V.; Zhang, Z. Mitsuba 3 Renderer. 2022. Available online: <https://mitsuba-renderer.org> (accessed on 5 February 2023).

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.