Supplemental Document

This is the supplemental material for the CVPR 2021 paper *Global Transport for Fluid Reconstruction with Learned Self-Supervision*. The accompanying video as well as the source code can be found on the project website.

A. Details on Method and Implementation

A.1. Algorithm

Here we detail the different algorithms used in the ablation. Algorithm 1 shows how the density of a single frame is reconstructed. This variant corresponds to the *single* version using the single pass of Section 4. The *forward* reconstruction is summarized in Algorithm 2; it uses the density reconstruction of Algorithm 1. In Algorithm 3 we depict the *coupling* (C) over time, and Algorithm 4 shows our full method with global transport optimization *global transport* (Glob-Trans).

Algorithm 1: Update Steps

Function `UpdateDensity` $(\rho, \mathcal{L}_\rho)$ **:**
  $\boldsymbol{L} \leftarrow$ `BuildLighting` $(\rho$, *lights*$)$;
  $\hat{\boldsymbol{I}}_c \leftarrow \mathcal{R}(\rho, \boldsymbol{L}, c)$;
  $\nabla\rho \leftarrow \partial\mathcal{L}_\rho / \partial\rho$;
  Update ρ with $\nabla\rho$ using Adam;
  $\rho \leftarrow$ `max` $(\rho, 0)$; //clip negative density
Function `UpdateVelocity` $(\rho, \mathcal{L}_\mathbf{u})$ **:**
  $\nabla\mathbf{u} \leftarrow \partial\mathcal{L}_\mathbf{u} / \partial\mathbf{u}$;
  Update $\mathbf{u}$ with $\nabla\mathbf{u}$ using Adam;
Function `UpdateDiscriminator` (ρ) **:**
  $\boldsymbol{I}_r \sim R$;
  $f \sim \Omega$; //sample random views
  $\hat{\boldsymbol{I}}_f \leftarrow \mathcal{R}(\rho, \boldsymbol{L}, f)$;
  $\hat{\boldsymbol{I}}_h \sim$ history;
  history $\leftarrow$ history $\circ\, \hat{\boldsymbol{I}}_f$;
  $\hat{\boldsymbol{I}}_f \leftarrow \hat{\boldsymbol{I}}_f \circ \hat{\boldsymbol{I}}_h$;
  Update $\Theta_\mathcal{D}$ with $\partial\mathcal{L}_\mathcal{D}(\rho, 1) / \partial\Theta_\mathcal{D}$;
   //Equation (11)

Algorithm 2: Forward Pass

Function `OptimizeDensity` (ρ) **:**
  for n **do**
    `UpdateDensity` $(\rho, \mathcal{L}_{\text{tar}})$;
Function `OptimizeVelocity` $(\mathbf{u}, \rho_{from}, \rho_{to}, n_{MS})$ **:**
  for $i = 0$ **to** n **do**
    if $i \in n_{MS}$ **then**
      $\mathbf{u} \leftarrow$ `ResizeGrid` $(\mathbf{u})$;
    `UpdateVelocity` $(\mathbf{u}, \mathcal{L}_{\mathcal{A}(\rho)} + \mathcal{L}_{\text{div}})$;
Function `ReconstructForward` $(\{\rho\}, \{\mathbf{u}\})$ **:**
  $\rho^0 \leftarrow \boldsymbol{H}^0 \cdot c$;
  $\mathbf{u}^0 \leftarrow \boldsymbol{H}^1 \cdot$ rand;
  `OptimizeDensity` (ρ^0) ; //Algorithm 1
  $\rho^1 \leftarrow \rho^0$;
  `OptimizeDensity` (ρ^1) ;
  `OptimizeVelocity` $(\mathbf{u}^0, \rho^0, \rho^1, n_{MS})$;
  for $t = 1$ **to** $\boldsymbol{F} - 1$ **do**
    $\mathbf{u}^t \leftarrow \mathcal{A}(\mathbf{u}^{t-1}, \mathbf{u}^{t-1})$;
    $\rho^{t+1} \leftarrow \mathcal{A}(\rho^t, \mathbf{u}^t)$;
    `OptimizeDensity` (ρ^{t+1}) ;
    `OptimizeVelocity` $(\mathbf{u}^t, \rho^t, \rho^{t+1}, \emptyset)$;
  $\mathbf{u}^{\boldsymbol{F}-1} \leftarrow \mathcal{A}(\mathbf{u}^{\boldsymbol{F}-2}, \mathbf{u}^{\boldsymbol{F}-2})$;

Algorithm 3: Coupled Optimization

`ReconstructForward` $(\{\rho\}, \{\mathbf{u}\})$;
for $i = 0$ **to** n **do**
  for $t = 0$ **to** $\boldsymbol{F}$ **do**
    `UpdateDensity` $(\rho^t, \mathcal{L}_{\text{tar}} + \mathcal{L}_{\mathcal{A}(\rho)} + \mathcal{L}_\mathcal{D})$;
    `UpdateVelocity` $(\mathbf{u}, \mathcal{L}_{\mathcal{A}(\rho)} + \mathcal{L}_{\mathcal{A}(\mathbf{u})} + \mathcal{L}_{\text{div}})$;
  if *useDisciminator* **then**
    `UpdateDiscriminator` $(\{\rho\})$;

A.2. Discriminator

Our discriminator $\mathcal{D}$ has a fully convolutional architecture with 14 convolution layers (filters, strides): (8,1) (16,2) (16,1) (24,2) (24,1) (32,2) (32,1) (32,1) (64,2) (64,1) (64,1) (16,1) (4,1) (1,1). All layers use a filter size of 4×4 and are followed by LReLU activations, with leak of 0.2. We use SAME padding, but instead of padding with zeros we use the mirrored input to avoid issues described in [31]. For data augmentation we apply random crop, scale, rotation, intensity and gamma scale. We keep a history of rendered fake samples $\hat{\boldsymbol{I}}_f$ to reuse as additional fake samples later for training the discriminator which improves the quality of the reconstruction and reduces rendering operations. The batch size for each discriminator or density update is 8 real samples and 4 rendered fake samples, with 4 additional fake samples from the history when training the discriminator. The 'average' formulation of the relativistic GAN [34] allows to use different resolutions and batch sizes for real and fake samples, while the least-squares variant gives empirically better results in our case. We also use an L_2 regularization for the weights of the discriminator $\Theta_\mathcal{D}$.

Algorithm 4: Full Method

```
ReconstructForward({ρ̃}, {u});
ρ^0 ← ρ̃^0;
for i = 0 to n do
    if i ∈ n_MS then
        ρ^0 ← ResizeGrid(ρ^0);
        for t = 0 to F do
            u^t ← ResizeGrid(u^t);
    // build sequence from first frame ρ^0
    for t = 1 to F do
        ρ^t ← A(ρ^{t-1}, u^{t-1});
    ∇ρ^F_A ← 0; // init gradient EMA
    for t = F − 1 to −1 do
        ∇ρ^t ← ∂L_ρ(ρ^t)/∂ρ^t;
        ∇ρ^t_A ← ∂A(ρ^t, u^t)/∂ρ^t · ∇ρ^{t+1}_A;
        if t == 0 then
            Update ρ^t with ∇ρ^t + λ_{ρA} ∇ρ^t_A;
        ∇u^t_A ← ∂A(ρ^t, u^t)/∂u^t · ∇ρ^{t+1}_A;
        Update u^t with ∇u^t + λ_{uA} ∇u^t_A;
        ∇ρ^t_A ← β ∗ ∇ρ^t_A + (1 − β) ∗ ∇ρ^t;
    if useDisciminator then
        UpdateDiscriminator({ρ});
```

A.3. Differentiable Warping

For the discretized operator $\mathcal{A}$ we use a second-order transport scheme in order to preserve small-scale content on the warped fields [65]. This MacCormack advection is based on Semi-Langrangian advection $\mathcal{A}_{SL}$, where the advected quantity s^{t+1} at position $\vec{x}$ is the tri-linear interpolation of s^t at $\vec{x} - \vec{u} * \Delta t$. With this, the MacCormack-advection is defined as

$$\begin{aligned} \hat{s}^{t+1} &:= \mathcal{A}_{SL}(s^t, \mathbf{u}^t), \\ \hat{s}^{t} &:= \mathcal{A}_{SL}(\hat{s}^{t+1}, -\mathbf{u}^t), \\ s^{t+1} &:= \hat{s}^{t+1} + 0.5(s^t - \hat{s}^t). \end{aligned} \tag{13}$$

However, we use a variant that reverts to $\mathcal{A}_{SL}$ when the corrected term s^{t+1} would exceed the values used for interpolation in the first $\mathcal{A}_{SL}$:

$$s^{t+1} := \begin{cases} s^{t+1} & \text{if } \hat{s}^{t+1}_{min} \leq s^{t+1} \leq \hat{s}^{t+1}_{max} \\ \hat{s}^{t+1} & \text{else} \end{cases}. \tag{14}$$

This reversion is necessary as the un-clamped MacCormack correction causes issues like negative density and escalating inflow from the open boundaries. The result is smoother than the one obtained without clamping, but still more detailed than $\mathcal{A}_{SL}$.

The advected generic quantity s can be both ρ and $\mathbf{u}$, in case of $\mathbf{u}$ the components are handled individually and treated as scalars.

Differentiation We implemented a differentiable SL advection as tensorflow CUDA operation. Since the MacCormack advection is composed of two SL steps and a condition, its gradients are simply handled by automatic differentiation. Since we optimize ρ and $\mathbf{u}$, both $\partial\mathcal{A}(s, \mathbf{u})/\partial s$ and $\partial\mathcal{A}(s, \mathbf{u})/\partial\mathbf{u}$ are required. For $\partial\mathcal{A}(s, \mathbf{u})/\partial s$, the gradients are just scattered to the grid positions used for the interpolation in the forward step, after being multiplied with the interpolation weights. $\partial\mathcal{A}(s, \mathbf{u})/\partial\mathbf{u}$ is $\mathcal{A}(\nabla s, \mathbf{u})$, *i.e.*, during back-propagation the gradients are multiplied component wise with the advected spatial gradients of the scalar grid s.

A.3.1 Boundary Handling

We use open boundaries for both density and velocity. Technically that means that velocities at the boundaries are not constrained (or handled differently from the inner velocities in any way). In the advection, look-ups that fall outside of the volume are clamped to the boundaries, thus using the values there. Their gradients are also accumulated at the boundaries accordingly.

A.3.2 Inflow

Inflow is only handled explicitly for the density, the velocity has to solve all inflow via the open boundaries. As we target rising smoke plumes we place an inflow region at the lower end of the visual hull $\boldsymbol{H}$, with some overlap, although it could be composed of arbitrary regions of the volume without changing the algorithm (we use a simple mask to indicate inflow cells). These cells are additional degrees of freedom and are optimized alongside the density. They are individual for every frame, and still used when using our global transport formulation. The advection then becomes $\mathcal{A}(\rho^t + \text{inflow}, \mathbf{u}^t) = \rho^{t+1}$ and is used as such in all losses and the global transport. This is also how the inflow receives gradients. The only constraint on the inflow is that $\rho^t + inflow \geq 0$, meaning that the inflow itself can be (partially) negative, and thus serve as outflow. Density can also enter and leave the volume via the boundaries, just like the velocity.

A.4. Differentiable Rendering

For rendering we use a discrete ray-marching scheme that essentially replaces the integrals in Equation (8) with discrete sums, stepping though the volume along the view-

depth (z) with a fixed step size:

$$\sum_{z=0}^{Z} \boldsymbol{L}(x_z) e^{-\sum_{a=0}^{x_z} \rho(a)}, e^{-\sum_{z=0}^{Z} \rho(x_z)}. \quad (15)$$

To implement the single-scattered self-shadowing we use a shadow volume, instead of casting shadow rays from every sample point. Thus, the shadow computation is like rendering the transparency from the point of view of the light, but storing every z-slice to create the shadow volume.

We implement our custom ray-marching kernel and its analytic gradient as tensorflow CUDA operation. The backwards pass inside this operation is what standard auto-differentiation would yield: go backwards along the ray and scatter gradients to the locations used for interpolation in the forward pass.

A.4.1 Sampling Gradients

The naive back-propagation of gradients through the grid-sampling, *i.e.*, scattering the gradients of a sample point to all used interpolants, weighted with the original interpolation weights, yields regular artifacts in the reconstruction, likely due to the regular grid sampling. Using an intermediate grid allows us to invert the sampling for back-propagation with another sampling operation as in the forward pass, which can then employ mip-mapping, and thus reduces aliasing in both directions. Conceptually, this approach does not differentiate the sampling, but rather transforms the volume of gradients into another space, resulting in spatially smooth gradients. It is, however, very memory intensive as all samples have to be stored. An equivalent gradient can also be obtained with scattering, by normalizing the gradients with their accumulated interpolation weights. This also avoids aliasing, as long as the rendering resolution is large enough, and is the method we use for our final results.

A.4.2 Comparison to Linear Image Formation

The comparisons in Figures 8 and 9 highlight the importance of the non-linear IF for visible light capturing. While a linear model reduces densities to account for effects like shadowing, the differentiable rendering in our optimization yields correct gradients to recover the original density, (b) *vs.* (c) in Figure 8.

A.4.3 Comparison to Path-tracer

We compare our renderer (Section 3.2) to a path-traced reference in Figure 10. Using 0 light bounces in the path-tracer (*i.e.* only single-scattering shadow rays) produces results very similar to our renderer, while the results with more bounces (multi-scattering) change gracefully.

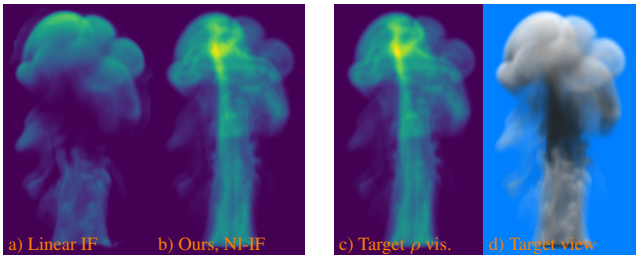


Figure 8: Non-linear IF optimization: a linear model "bakes" lighting into the density reconstruction (a), while our pipeline can recover the correct density distribution (b) via a differentiable renderer. (c,d) show target density as visualization and with lighting, respectively.

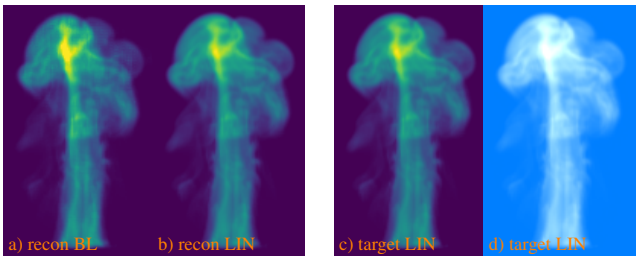


Figure 9: A target rendered with a linear (LIN) IF can be recovered by the attenuated model (BL), as long as sufficient lighting is provided.

Figure 10: Our renderer vs. Blender 2.91 [5] Cycles path-tracer (128 SPP, denoised), 0 and 4 light bounces.

A.4.4 Backgrounds and Lighting Optimization

When using a black background or monochrome target image there is an ambiguity between high light intensity in conjunction with low density and lower light intensity with more density as these two cases cannot be distinguished in a target image. With a black background the transparency does not matter, and with a monochrome target (with any background) contributions from reflected light and background are indistinguishable. Using a background with a color distinguishable from the smoke, the transparency, and therefore density, along a ray can be determined success-

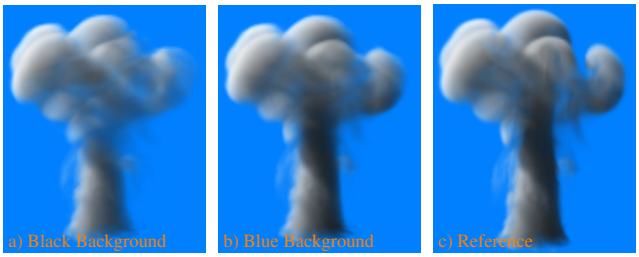


Figure 11: Side view (90°) of a density reconstructed with black (a) and light blue RGB(0,127,255) (b) background. A colored background helps to recover the correct density in shadowed regions.

fully.

The reconstruction consequently yields better results when using a colored background, as seen in Figure 11. However, as our real target dataset [9] uses only grey-scales, we likewise used black backgrounds for our synthetic tests.

We also experimented with optimizing the light intensities (i_a and i_p), which worked better with colored backgrounds. Because this adds additional degrees of freedom, making the reconstruction harder, we ultimately determined our light intensities empirically.

A.5. Visual Hull

For our multi-view reconstruction experiments with 5 target views, the visual hull $\boldsymbol{H}$ is constructed only from the 5 available views, as described in Section 3.4. After the background subtraction the target images are turned into a binary mask, using a threshold of $\epsilon = 0.04$ to cut of residual noise. These masks are slightly blurred with $\sigma = 1$ (in pixels) before being projected into the volume (Eq. (12)). The hulls are then blurred again with $\sigma = 0.5$ (in cells).

To construct a visual hull from a single view, we create 4 additional, evenly spread views from the available view by rotating it around the central y-axis (up) of the volume. The masks created from these auxiliary views are additionally mirrored at the projected rotation axis to reduce cutoffs in the original view when constructing the volume hull by intersection. This works well for relatively symmetric rising plumes, but is not guaranteed to work for arbitrary shapes. Using a visual hull to guide the densities noticeably improves the quality of the reconstruction, but constructing a hull for sparse or single-view reconstructions that adheres to physically correct motion remains a topic for future work.

A.6. Multi-Scale Approach

We employ a multi-scale approach during reconstruction, increasing the grid resolution while the spatial size of the domain stays the same. The up-sampling is done using tri-linear interpolation. In our full method we use a scaling factor per step of 1.2. Thus, for a reference resolution of 128 (see also Appendix B), the velocity of the first frame is scaled 4 times ($14 \rightarrow 18 \rightarrow 21 \rightarrow 25 \rightarrow 30$) in the pre-optimization, and density and velocity synchronously in the coupled phase an additional 8 times ($30 \rightarrow 36 \rightarrow 43 \rightarrow 51 \rightarrow 62 \rightarrow 74 \rightarrow 88 \rightarrow 106 \rightarrow 128$). As the first frame velocity optimization operates on a lower resolution than the densities ρ^t and ρ^{t+1} used therein, the densities are temporarily down-sampled (with filtering) to match the resolution of the velocity.

A.7. Gradient Descent with Global Transport

Naive automatic differentiation of a full 120 frame sequence via back-propagation would require a lot of memory as all intermediate results of the forward pass have to be recorded for the backwards pass. To reduce the memory requirements we split the sequence to only do automatic back-propagation for one frame at a time. The back-propagation between frames is handled more explicitly (see also line 19 in Algorithm 4). The result is the same, at least when using Equation (4) instead of the EMA version (Eq. (5)). We keep all resources in (CPU) RAM, only the current frame and intermediate steps needed for automatic differentiation are moved to the GPU.

A.8. Hyperparameters

The hyperparameters, which can be set individually for the first frame in the pre-pass, the remaining frames of the pre-pass, and the main optimization, are reported in Table 2. Some benefit from a linear ($\xrightarrow{lin}$) or exponential ($\xrightarrow{exp}$) fade-in or growth. We do not use any smoothness regularization for density or velocity.

A.9. Hardware and Runtime Statistics

We used the following hard- and software configuration for our experiments: Hardware: CPU: Intel(R) Core(TM) i7-6850K, GPU: Nvidia GeForce GTX 1080 Ti 11GB, RAM: 128GB; Software (versions): Python 3.6.9, Tensorflow 1.12 (GPU), CUDA 9.2. We implemented custom tensorflow operation in CUDA for rendering (ray-marching and shading) and advection as well as their gradient operations.

For our full method, using the resolution detailed in Appendix B, we have measured an average reconstruction time of 43 - 65 hours ($\sim$ 30 minutes per frame, variation based on system load) for 120 frames of single view reconstruction from real targets with our current implementation and hardware. The pre-optimization (Sec. 4) takes only 2% of the total time. The majority of the time is spent on the density optimization (60%, 38% of which is needed for the discriminator loss and rendering the needed fake images), while Calculating and back-propagating though the losses for $\mathbf{u}$ makes up 30% of the time. The manual back-propagation

Loss \Pass	Fwd first	Fwd	Main
Density			
n	600	600	4200
n_{MS}	-	-	400×8
η_ρ	3	$3 \xrightarrow{lin} 1$	2.4
$\mathcal{L}_{\text{tar}}$	1.74e-5		
$\mathcal{L}_{\mathcal{A}(\rho)}$	0	0	2.7e-10 $\xrightarrow{lin}$ 5.4e-10
$\mathcal{L}_{\mathcal{D}}(\rho, -1)$	0	0	1.5e-5
Velocity			
n	6000	600	4200
n_{MS}	1000×4	-	400×8
$\eta_{\mathbf{u}}$	0.04	0.02	$\xrightarrow{exp}$ 0.016
$\mathcal{L}_{\mathcal{A}(\rho)}$	4.1e-10		
$\mathcal{L}_{\mathcal{A}(\mathbf{u})}$	0	0	4e-11 $\xrightarrow{lin}$ 8e-11
$\mathcal{L}_{\text{div}}$	8.6e-10 $\xrightarrow{exp}$	2.6e-9	$\xrightarrow{exp}$ 1.7e-8
Discriminator†			
$\eta_{\mathbf{u}}$	-	-	2e-4
$\lvert\Theta_{\mathcal{D}}\rvert^2$	-	-	2e-3

Table 2: Hyperparameters for the first (lines 12 and 15 in Algorithm 2) and remaining frames of the forward reconstruction and our coupled global transport optimization (Algorithm 4). learning rate η, iterations n with multi-scale intervals n_{MS} with a resize factor of 1.2.

though the global transport via Equation (5) takes 4% of the total runtime.

B. Additional Evaluation

The Reference has a grid size of $128 \times 196 \times 128$. As base grid size for our reconstruction we use $128 \times 227 \times 128$, which is cropped to the union of the AABBs of the visual hulls of all frames plus a padding of 4 cells on all sides. For our reconstructions the grid sizes are therefore reduced to $125 \times 187 \times 93$, $94 \times 186 \times 95$, $97 \times 164 \times 77$ and $86 \times 161 \times 76$ for synthetic multi-view, synthetic single-view, real multi-view and real single-view for the base plume, respectively. The spatial resolution remains the same. As lighting model we use a single white point light with single-scattering, hand-placed above the central camera, and white ambient light to approximate multi-scattering. The light intensities are $i_p = 0.85$ and $i_a = 0.64$, respectively.

B.1. Additional Multi-View Ablations

An extended evaluation of the motion of the synthetic case is shown in Figure 12. The *forward* reconstruction removes the streak-like artifacts, but does so at the expense of spatial reconstruction accuracy regarding the metrics. The step to *coupled* improves the motion it results in only minor improvements in ρ. The multi-scale approach (C-MS) brings the metrics for ρ on-par with the *single* version. The bottom row show a continuously improving transport of a test density, compared to the reference (Figure 12 f), seen in the resulting distribution of the density. While ScalarFlow might appear closer to the reference it shows a mismatch in shape at the stem and backside (right side, shown in the detail) of the resulting plume.

An ablation of multi-view reconstructions of the SF data is shown in Figure 14. Here, C-MS shows an unphysical accumulations of density at the top which are resolved by our *global transport* (Glob-Trans). The discriminator (Full) has only little effect in the multi-view reconstruction, sometimes causing slight halo-artifacts, and is thus not used here. Previous methods give a more diffuse result.

B.2. Warp Loss in Global Transport

In Figure 13 we show an additional ablation that highlights the effects of keeping the per-frame warp loss $\mathcal{L}_{\mathcal{A}(\rho)}$ for both density ($\partial\mathcal{L}_{\mathcal{A}(\rho)}/\partial\rho$) and velocity ($\partial\mathcal{L}_{\mathcal{A}(\rho)}/\partial\mathbf{u}$), even when global transport (Sec. 3.1) is used. Our global transport already provides gradients from the advection of the density for both ρ (Eq. (5)) and $\mathbf{u}$, meaning that the explicit density warping loss seems redundant for both. In fact, since the sequence is defined via transport (Eq. (2)), the warp loss, and therefore its gradients, should be zero. However, since the visual hull is applied after every advection step (in the initial transport, not the warp loss), the error stems from the parts removed by the hull. Using the hull as a loss, instead or in addition to a hard constraint, does not change anything about the results in Figure 13.

As visible in the figure, $\partial\mathcal{L}_{\mathcal{A}(\rho)}/\partial\rho$ is still mostly redundant, only in the real data case it reduces the divergence of the reconstructed density grid (not visible in the rendering). However, it also has no negative influence on the reconstruction. Removing $\partial\mathcal{L}_{\mathcal{A}(\rho)}/\partial\mathbf{u}$ from the optimization has adverse effect in the real case, resulting in visible divergence in the lower part of the plume, turning the inflow into an outflow. In the Synthetic case the reconstruction still works well, showing only less inflow. We conjecture that the differences between the synthetic and real cases are caused by a mismatch of the physical model to the real case. Using $\mathcal{L}_{\mathcal{A}(\rho)}$ puts more emphasis (gradients) on *local* correctness of the transport itself. $\partial\mathcal{L}_{\mathcal{A}(\rho)}/\partial\rho$ adapts the density to fit the existing transport. However, because the density also has to fit the observations, which have more weight, there is no change, and thus removing these gradients has no big impact. $\partial\mathcal{L}_{\mathcal{A}(\rho)}/\partial\mathbf{u}$, on the other hand, adapts the velocity, *i.e.*, the transport, to fit the existing density, including the hull constraint. These results seem to suggest that explicit adaption of the local transport to the densities is still necessary when using the global transport.

B.3. State of the Art

We re-implemented TomoFluid (TF) in our own Framework as no original source-code is publicly available as of the time of writing. We use the loss functions and view-interpolations as described in the original work, but the hy-

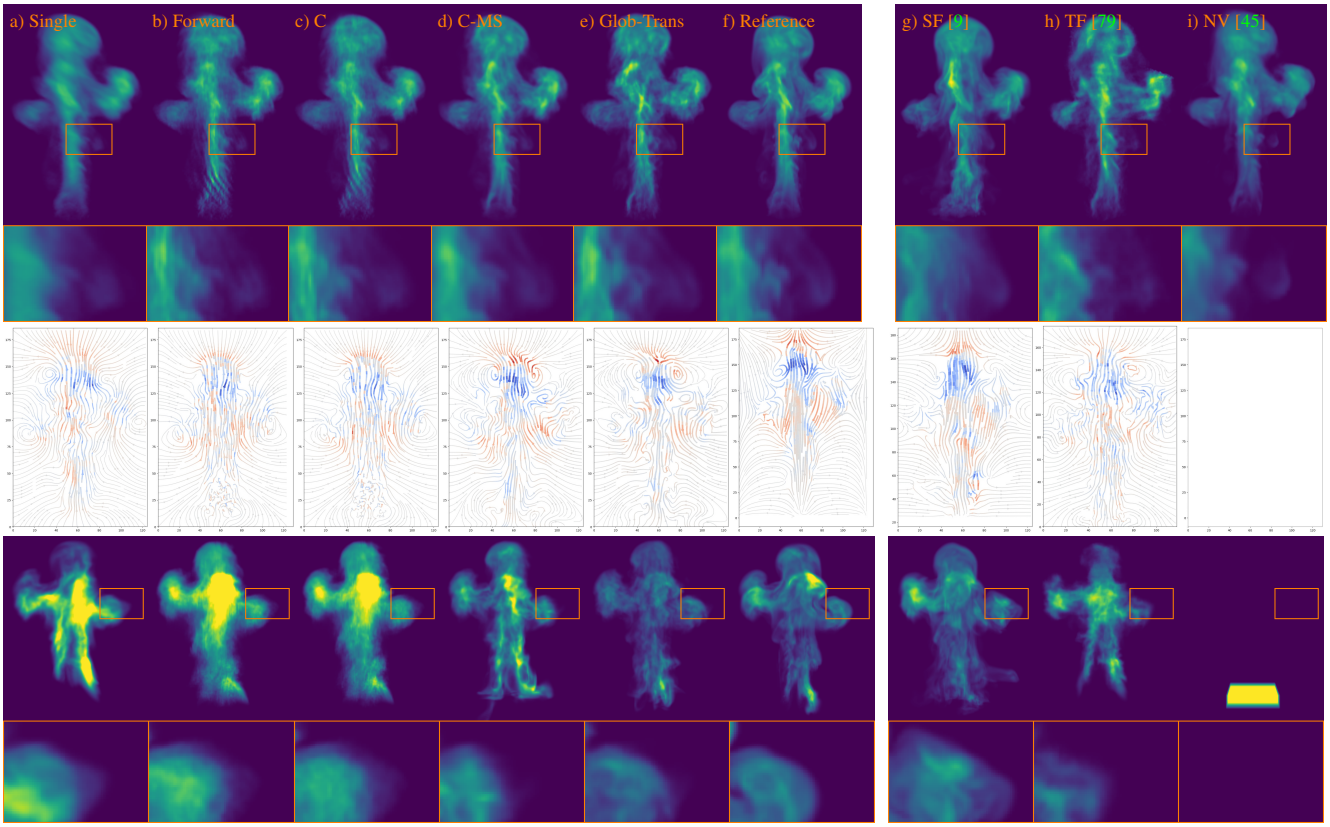
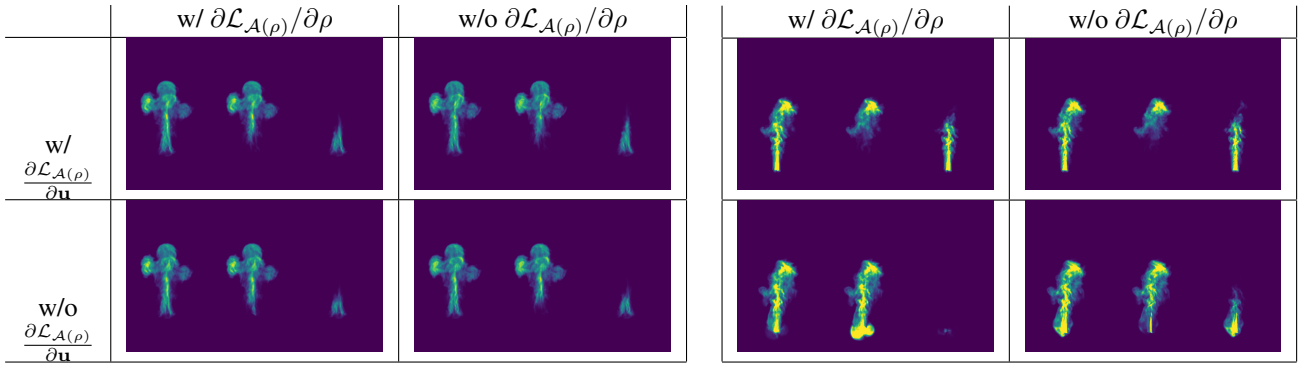


Figure 12: Multi-view evaluation with synthetic data: (a-e) Ablation with reference shown in (f). The different versions of the ablation continually improve density reconstruction (top) and motion accuracy, illustrated by advecting the initial state shown in (i) with the reconstructed velocity sequence, in the bottom row. (g-i) Comparison with previous work: ScalarFlow [9], TomoFluid [79], and NeuralVolumes [45]. Our method in (e) yields an improved density reconstruction, in addition to a coherent, and physical transport. Note that (i) does not produce velocities.



(a) Synthetic (b) Real

Figure 13: Evaluation of the influence of the density warp loss $\mathcal{L}_{\mathcal{A}(\rho)}$ when *global transport* is also used. The tables show an ablation of the $\mathcal{L}_{\mathcal{A}(\rho)}$ applied to density and velocity, for both synthetic and real data. Every image shows the following triplet: left, the final result of the optimization, obtained by advecting ρ^0 with the velocity sequence and adding the inflow in every step; the center shows the same, but without any inflow; on the right, only the inflow is advected, *i.e.*, ρ^0 is set to all zero. The top left images are the desired results. While the gradients provided by this loss are are almost redundant in the synthetic case (a), the gradients w.r.t. $\mathbf{u}$ are necessary when using real data(b), even though the same dependencies are modeled by the global transport. Gradients w.r.t. ρ are sill redundant, but do not hurt the reconstruction either.

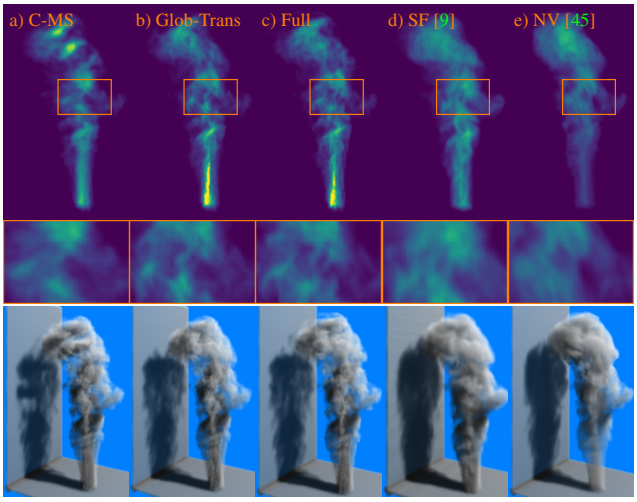


Figure 14: Multi-views (5 targets) reconstruction with data from [9]. Top: perspective density from a 90° view. Bottom: rendering with lighting and inc. density from 60°. Global transport prevents artifacts, *e.g.*, visible in (a), while previous work (d,e) is smoother.

perparameters had to be adapted to work in our setting. As TF is implemented in our framework its grid is also cropped.

We use the original ScalarFlow code, which is based on mantaflow, the solver we used to create our synthetic test case. Due to this shared code-base SF has a certain advantage in the synthetic case. It shows in the visual similarity of the reconstructed motions compared to the reference, as seen in the stream-plots of Figure 14 (f,g). Nevertheless, our method matches the targets better than the SF version. The high transport error of SF in Table 1a might be caused, in part, by the missing inflow as the ScalarFlow algorithm removes the synthetic inflow after reconstruction. SF runs on the full $128 \times 196 \times 128$ grid in the synthetic test case while the real reconstructions from the provided dataset use a grid size of $100 \times 178 \times 100$.

NeuralVolumes typically works with many more viewpoints than we provide. This results in artifacts in our sparse-view setting, *i.e.*, colored or black background is reconstructed within the volume. The reconstruction is however still very good. To compute metrics and rendering we sample a RGBα volume at the resolution of the reference and extract a density via $\rho_{NV} := 0.03[\alpha]_0^1(\mathrm{R}+\mathrm{G}+\mathrm{B})/3$, which removes black opaque artifacts, thus giving NV an advantage in these comparisons.

B.4. Details for the Single-View Ablation

For the single-view ablation, Figure 5, we focus on the key versions of our previous, multi-view ablation and evaluate them on real targets as the effect of the discriminator is more prominent here. The tomographic reconstruction (a) yields a strong smearing out of information along the unconstrained direction of the viewing rays, despite the existence of transport initialization. Although the qualitative examples from a 90° angle in Figure 5 top illustrate this behavior, it is even more clearly visible in the supplemental videos. The *coupling* version (b) refines the reconstructed volumes, but still contains noticeable striping artifacts as well as a lack of details. Our *global transport* formulation (c) resolves these artifacts and yields a plausible motion, which, however, does not adhere to the motions observed in the input views. In particular, it yields relatively strong single streaks of transported density which are more diffusive in the real world flow. The discriminator matches the appearance of the reconstruction to the real world smoke, thus guiding the motion reconstruction to adhere to features of real flows (d).