DOI: 10.1111/cgf.142638
EUROGRAPHICS 2021 / N. Mitra and I. Viola
(Guest Editors)

Volume 40 (2021), Number 2

Two-step Temporal Interpolation Network Using Forward Advection for Efficient Smoke Simulation

Young Jin Oh and In-Kwon Lee

Dept. of Computer Science, Yonsei University

Abstract

In this paper, we propose a two-step temporal interpolation network using forward advection to generate smoke simulation efficiently. By converting a low frame rate smoke simulation computed with a large time step into a high frame rate smoke simulation through inference of temporal interpolation networks, the proposed method can efficiently generate smoke simulation with a high frame rate and low computational costs. The first step of the proposed method is optical flow-based temporal interpolation using deep neural networks (DNNs) for two given smoke animation frames. In the next step, we compute temporary smoke frames with forward advection, a physical computation with a low computational cost. We then interpolate between the results of the forward advection and those of the first step to generate more accurate and enhanced interpolated results. We performed quantitative analyses of the results generated by the proposed method and previous temporal interpolation methods. Furthermore, we experimentally compared the performance of the proposed method with previous methods using DNNs for smoke simulation. We found that the results generated by the proposed method are more accurate and closer to the ground truth smoke simulation than those generated by the previous temporal interpolation methods. We also confirmed that the proposed method generates smoke simulation results more efficiently with lower computational costs than previous smoke simulation methods using DNNs.

CCS Concepts

• ***Computing methodologies*** → ***Physical simulation;***

1. Introduction

Physics-based simulation is an essential research topic for realistic computer graphics content, and many studies have attempted to achieve accurate physics-based simulation results with low computational costs. As recent computer graphics content is frequently executed in environments in which real-time interaction between users and objects is key, efficient physics-based simulation technology has become increasingly important. In particular, in the field of flow simulation, which aims to compute the motion of smoke and liquids, studies have been proposed to reduce the high computational costs and increase the accuracy.

Specifically, various studies have been proposed to improve the simulation algorithm [SG11, GNS*12, ATW15, YJL*16] or to use pre-computed examples to generate flow simulations for new environmental conditions [Thu16, SDN18, RKEW19]. Recently, efficient flow simulation methods using deep neural networks (DNNs) have been introduced. These include accelerated simulation methods using DNNs [TSSP17, UHT18, WBT19, KAT*19] and example-based simulation methods using DNNs [CT17, PBT18]. Moreover, super-resolution methods for flow [XFCT18, WXCT19] and the flow upsampling method [BLDL20] have been proposed to convert low-resolution smoke simulations into high-resolution ones. Nevertheless, since previous efficient smoke simulation methods have focused on reducing the cost of computing one frame in a smoke simulation, it is necessary to repeat the simulation and DNN inference for every frame in order to generate a flow simulation result with a high frame rate. Therefore, unlike in previous methods, we propose a method to efficiently generate a high frame rate simulation via temporal interpolation of smoke simulation results computed with large time steps.

Research on temporal interpolation in computer vision for use in general videos has been actively conducted [Che02, ESH06, RRBW12], and various temporal interpolation methods using DNNs have been proposed. For example, there are methods of generating interpolation results by warping the input frames using the optical flow estimated from a DNN [LYT*17, JSJ*18, GWC*19]. However, challenges arise when applying previous temporal interpolation methods directly to smoke frames. Unlike real objects in general videos, as the contours of the smoke flow are frequently deformed even in small time steps, we cannot easily predict the exact optical flow between given input frames via DNNs. In addition, translucent smoke is limited in the use of additional features, such as depth information. Therefore, when applying previous temporal

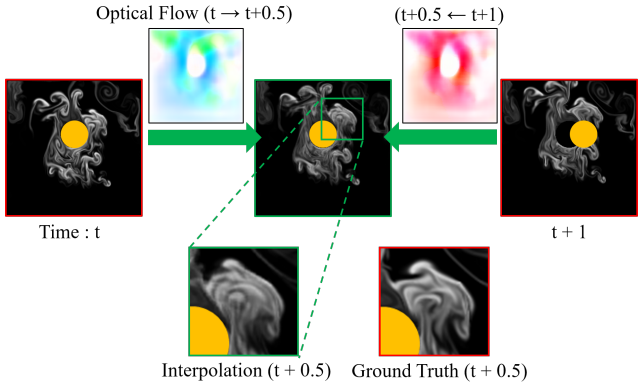


Figure 1: *Example of a temporally interpolated smoke frame generated by the optical flow-based temporal interpolations method. Compared to the ground truth (red), inaccurate and blurry smoke frames (green) are generated.*

interpolation methods directly to smoke simulations, the interpolation results can be inaccurate and blurry (see Figure 1).

In this paper, we propose a two-step temporal interpolation network using forward advection for efficient smoke simulation. Since the proposed method converts a low frame rate smoke simulation computed with large time steps into a high frame rate smoke simulation through the inference of a temporal interpolation network, it is able to efficiently generate a high frame rate smoke simulation with a lower computational cost than the conventional simulation method. The first step of the proposed network is to perform optical flow-based temporal interpolation for two input smoke frames. In the second step, to reduce the inaccurate and blurry artifacts that arise in the first step, we compute temporary smoke frames between the two input smoke frames by forward advection, a low-cost physical computation. Finally, we interpolate between the results of forward advection and those of the first step. The temporary frames computed with forward advection are inaccurate with respect to the ground truth, so we cannot use them directly as temporal interpolation results. However, since blurry artifacts are not generated by forward advection, the interpolation step between the results of the forward advection and those of the first step corrects for first step artifacts and generates enhanced temporal interpolation results.

As a result of our experiment, we found that the results generated by the proposed method are more accurate and closer to the ground truth smoke simulation than those generated by previous temporal interpolation methods. We also confirmed that the proposed method generates smoke simulation results more efficiently, with lower computational costs, than previous smoke simulation methods using DNNs.

2. Related work

Efficient flow simulation using DNNs. As physics-based simulation methods play an important role in generating high-quality computer graphics content, efficient physics-based simulation research has been conducted for a long time [SG11, GNS*12, ATW15, YJL*16]. For efficient flow simulations in particular, various methods that generate accurate results with low computational costs and efficient flow simulation methods based on machine learning and DNNs have been proposed. Ladický et al. [LJS*15] proposed a fluid simulation method using Regression Forests and handcrafted features. Tomshon et al. [TSSP17] and Xiao et al. [XYY18] proposed DNN models that replace the pressure projection, which is a simulation stage with a high computational cost. Methods that generate visually enhanced flow simulations using DNNs have also been proposed. For example, Chu and Thuerey [CT17] proposed a convolutional neural network(CNN) model that computes the similarity between low-resolution and high-resolution flow patches to synthesize a pre-computed high-resolution simulation into a low-resolution simulation. Prantl et al. [PBT18] developed a deformation-aware DNN model to generate simulation results for new conditions using a pre-computed simulation set. A generative model applying a FLIP simulation has been proposed to improve the details of liquid splashing [UHT18], and generative models for super-resolution have been developed to convert low-resolution flow simulation results into high-resolution results [XFCT18, WXCT19]. Bai et al. [BLDL20] proposed a multiscale neural network that can upsample a coarse animation into a high-resolution smoke animation via dictionary-based learning. Moreover, DNN models that encode a flow simulation as a simplified representation and simulation methods using the simplified representation have also been advanced. Kim et al. [KAT*19] developed an auto-encoder model that encodes a flow simulation as a latent variable and then restores it to a velocity field; they also proposed an integration model to advance the latent variable. Wiewel et al. [WBT19, WKA*20] presented an LSTM-based DNN model to generate a stable and controllable temporal evolution of a fluid simulation from a latent variable space. Whereas most previous studies have focused on reducing the computational cost of one frame, the proposed method generates a high frame rate flow simulation by temporal interpolation of a low frame rate simulation. In addition, since the proposed method uses a low frame rate flow simulation that is computed by a physics-based simulation, it is possible to generate a stable high frame rate flow simulation without the cumulative errors caused by iterative DNN inference.

Video frame interpolation. Temporal interpolation for video is a long-standing research topic in the computer vision field, and various studies have been conducted on accurate interpolation [CHKK07, JKJS05, HN08, MWZ*15]. Recently, with the development of machine learning and DNN technology, many studies based on DNNs have appeared. Long et al. [LKA*16] developed a CNN model that directly generates an in-between frame for two input frames, while Liu et al. [LYT*17] proposed Deep Voxel Flow, which predicts the 3D optical flow with a DNN and then warps the input frames according to the optical flow. In the AdaConv [NML17a] and SepConv [LYT*17] studies, spatial kernel-based interpolation methods that synthesize neighboring pixels to generate interpolation results were presented. Bao et al. [BLZ*19] developed MEMC-NET, which utilizes both an optical flow-based method and a spatial kernel-based interpolation method. To improve the quality of the temporal interpolation results, the use of additional features of the input frames in DNN-based methods has also been proposed. For example, SuperSloMo [JSJ*18] interpo-

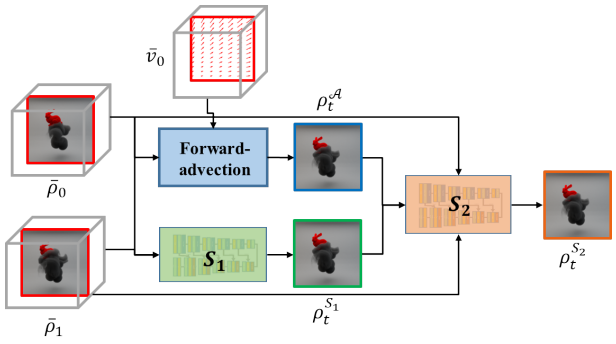


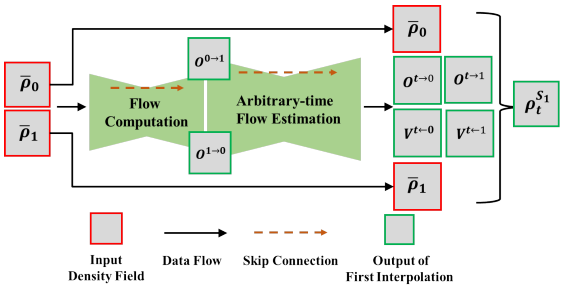
Figure 2: *Overview of the proposed method. The proposed temporal interpolation network consists of two interpolation networks: S_1 interpolates between two consecutive smoke frames using an optical flow-based temporal interpolation network. In order to improve the results of the first step, S_2 interpolates between those results and the results of forward advection.*

lates multiple in-between frames via optical flow-based interpolation and enhances the results with an occlusion mask predicted by a DNN. Moreover, Niklaus et al. [NL18] proposed a contextual extractor network to enhance the optical flow-based interpolation results, and DAIN [BLM*19] was proposed to warp input frames based on the optical flow, context features, and depth maps. CyclicGen [LLLC19] improved the interpolation result through a two-stage training model that performs temporal interpolation once more using the predicted result from the interpolation network. Our proposed temporal interpolation method deals with smoke simulation result data differently from previous temporal interpolation studies, which target general video. We propose a two-step interpolation method that uses forward advection to correct the inaccurate and blurry parts of interpolated smoke frames that occur when smoke frames are interpolated with a large time step using previous methods.

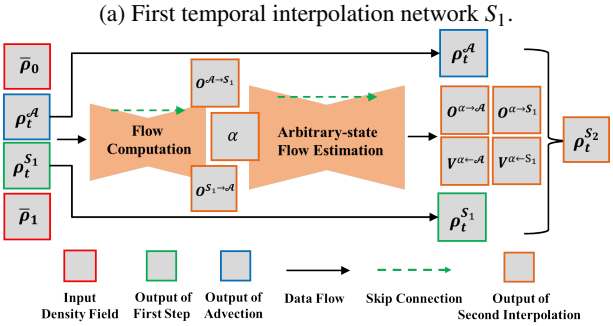
3. Proposed method

The proposed two-step temporal interpolation network converts a low frame rate smoke simulation with a large time step into a high frame rate smoke simulation. Unlike previous optical flow-based interpolation methods using one DNN model, the proposed method uses a two-step interpolation network to generate more accurate and less blurry interpolation results.

Figure 2 is an overview of the proposed method, which generates an interpolated smoke frame between two input smoke frames. The first temporal interpolation network S_1 receives two consecutive smoke frames $\bar{\rho}_0$ and $\bar{\rho}_1$, where $\bar{\rho}(\in \mathbb{R}^{H\times W\times D})$ is a density field frame from the ground truth simulation with height H, width W, and depth D ($D = 1$ for 2D smoke). S_1 generates a temporally interpolated smoke frame $\rho_t^{S_1}$ at arbitrary time $t \in (0,1)$ by conventional optical flow-based temporal interpolation. The second interpolation network S_2 then interpolates between the results of the first step and temporary smoke frames $\rho_t^{\mathcal{A}}$, which are computed with the forward advection at arbitrary time $t \in (0,1)$ using $\bar{\rho}_0$ and



(a) First temporal interpolation network S_1.



(b) Second interpolation network S_2.

Figure 3: *Overview of the structure of the two-step interpolation network.*

$\bar{v}_0$, where $\bar{v} \in \mathbb{R}^{H\times W\times D\times 3}$ ($\bar{v} \in \mathbb{R}^{H\times W\times D\times 2}$ for 2D smoke) is a velocity vector field in the ground truth simulation.

3.1. First step: temporal interpolation between two given smoke density grids

In the first step, we use SuperSloMo [JSJ*18], which is an optical flow-based temporal interpolation method that uses a DNN to generate $\rho_t^{S_1}$ between two input smoke frames. As suggested for SuperSloMo, the first temporal interpolation network S_1 consists of two U-Net [RFB15] structures, a flow computation network, and an arbitrary-time flow estimation network. The flow computation network generates the forward flow $O^{0\to1}$ and backward flow $O^{1\to0}$, where $O \in \mathbb{R}^{H\times W\times D\times 3}$, between the two input smoke frames $\bar{\rho}_0$ and $\bar{\rho}_1$. The arbitrary-time flow estimation network generates backward flow $O^{t\to0}$ and forward flow $O^{t\to1}$. $O^{t\to0}$ and $O^{t\to1}$ refer to flows from an arbitrary smoke frame $\rho_t^{S_1}$ to $\bar{\rho}_0$ and to $\bar{\rho}_1$, respectively. In addition, the arbitrary-time flow estimation network generates visibility maps $V^{t\leftarrow0}$ and $V^{t\leftarrow1}$, where $V \in [0,1]^{H\times W\times D}$. $V^{t\leftarrow0}$ ($V^{t\leftarrow1}$) indicates whether the smoke density in $\bar{\rho}_0$ ($\bar{\rho}_1$) remains visible in smoke frame ρ_t (0 denotes full occlusion). If the smoke in one voxel in $\bar{\rho}_0$ is not in the same voxel in $\bar{\rho}_1$, the corresponding voxel in $\rho_t^{S_1}$ of these two frames may not include smoke. Thus, the $V^{t\leftarrow1}$ becomes 0, preventing the warping result of $\bar{\rho}_1$ from appearing in the corresponding voxel at $\rho_t^{S_1}$. The first temporally interpolated smoke frame $\rho_t^{S_1}$ is computed as

$$\rho_t^{S_1} = \frac{1}{Z^{S_1}} \otimes \Big((1-t)V^{t\leftarrow0} \otimes \mathcal{W}(\bar{\rho}_0, O^{t\to0}) + tV^{t\leftarrow1} \otimes \mathcal{W}(\bar{\rho}_1, O^{t\to1}) \Big), \tag{1}$$

where $Z^{S_1} = (1-t)V^{t\to 0} + tV^{t\to 1}$ is a normalization factor, $\mathcal{W}$ is a backward warping function, and $\otimes$ denotes element-wise multiplication.

The loss function L^{S_1} for training the first temporal interpolation network includes with three loss terms:

$$L^{S_1} = \lambda_{\mathcal{R}} L^{S_1}_{\mathcal{R}} + \lambda_{\mathcal{W}} L^{S_1}_{\mathcal{W}} + \lambda_{\mathcal{S}} L^{S_1}_{\mathcal{S}}, \tag{2}$$

where $L^{S_1}_{\mathcal{R}}$ is the reconstruction loss, $L^{S_1}_{\mathcal{W}}$ is the warping loss, and $L^{S_1}_{\mathcal{S}}$ is the smoothness loss. $\lambda_{\mathcal{R}}$, $\lambda_{\mathcal{W}}$, and $\lambda_{\mathcal{S}}$ are the weights of the three loss terms. The reconstruction loss $L^{S_1}_{\mathcal{R}}$ quantifies the difference between the temporally interpolated smoke frame and the ground truth smoke frame. When we generate N-interpolated frames $\left\{\rho^{S_1}_{t_i}\right\}_{i=1}^{N}$ at arbitrary time $t_i \in (0,1)$, the reconstruction loss is

$$L^{S_1}_{\mathcal{R}} = \frac{1}{N}\sum_{N}\left\|\rho^{S_1}_{t_i} - \bar{\rho}_{t_i}\right\|_1, \tag{3}$$

where $\bar{\rho}_{t_i}$ is the ground truth smoke frame corresponding to $\rho^{S_1}_{t_i}$. The warping loss $L^{S_1}_{\mathcal{W}}$ denotes the difference between the backward warping results using predicted optical flows and the ground truth smoke frame; it is defined by

$$\begin{aligned} L^{S_1}_{\mathcal{W}} &= \left\|\mathcal{W}(\bar{\rho}_1, O^{0\to 1}) - \bar{\rho}_0\right\|_1 + \left\|\mathcal{W}(\bar{\rho}_0, O^{1\to 0}) - \bar{\rho}_1\right\|_1 \\ &+ \frac{1}{N}\sum_{N}\left(\left\|\mathcal{W}(\bar{\rho}_0, O^{t_i\to 0}) - \bar{\rho}_{t_i}\right\|_1 + \left\|\mathcal{W}(\bar{\rho}_1, O^{t_i\to 1}) - \bar{\rho}_{t_i}\right\|_1\right). \end{aligned} \tag{4}$$

The goal of the smoothness loss $L^{S_1}_{\mathcal{S}}$ is to encourage neighboring grids to have similar optical flow values; it is defined as

$$L^{S_1}_{\mathcal{S}} = \left\|\nabla O^{0\to 1}\right\|_1 + \left\|\nabla O^{1\to 0}\right\|_1, \tag{5}$$

where ∇ denotes the gradient operation.

3.2. Second step: interpolation between the results of the first interpolation and the forward advection results

The optical flow-based temporal interpolation of the first step trained based on a large number of smoke simulation scenes can generate accurate smoke frame results when the time step between the two input smoke frames is relatively small. This is because the range of expected in-between smoke states is small and limited if the time step between the two input smoke frames is small. Therefore, the temporal interpolation network can more easily and accurately estimate in-between smoke frames. However, as the time step between the two input smoke frames increases, the range of expected in-between smoke states also increases. Therefore, it becomes difficult for the temporal interpolation network to handle all possible in-between smoke states, resulting in inaccurate and blurry interpolation results (see Figure 1 and the first row of the left column of Figure 4). The proposed method proceeds with the second interpolation step to compensate for artifacts generated by the first interpolation.

The second interpolation network S_2 interpolates between the results of the first step and the temporary smoke frame $\rho^{\mathcal{A}}_t$. $\rho^{\mathcal{A}}_t$ is computed with the forward advection function $\mathcal{A}$ using $\bar{\rho}_0$, $\bar{v}_0$, and

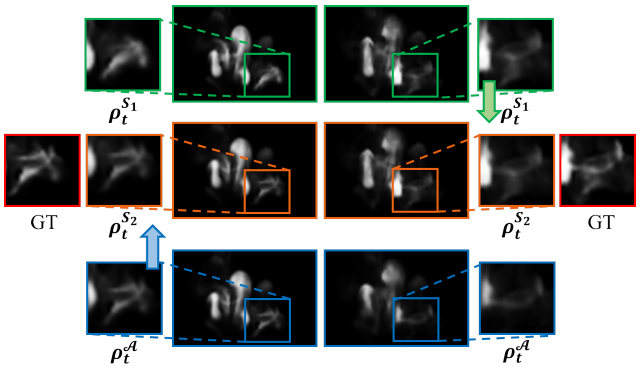


Figure 4: *Example results of the second interpolation step. With the arbitrary-state α interpolation, the results of the second step (orange) can be adaptively interpolated between the results of the first step (green) and the forward advection (blue).*

arbitrary time t ($\rho^{\mathcal{A}}_t = \mathcal{A}(\bar{\rho}_0, \bar{v}_0, t)$). The result of forward advection $\rho^{\mathcal{A}}_t$ is not appropriate as a temporal interpolation result because the difference between $\rho^{\mathcal{A}}_t$ and the ground truth increases as the arbitrary time t increases. However, since all of the forward advection result frames are generated through physical computation, they contain no irregular smoke shapes or blurry artifacts. We therefore use the forward advection results in the second interpolation process to correct for the inaccurate smoke shapes and blurry artifacts that appear after the first interpolation step. The smoke frame in the second row of Figure 4 shows the result of the second interpolation. We can see that the inaccurate and blurry part (first row of the left column of Figure 4) in the result of the first step has been replaced by the sharp part of the forward advection result (third row of the left column of Figure 4) after the second interpolation step. The final interpolation result is thus more accurate and less blurry compared to the result of the first step.

The second interpolation network S_2 is composed of two U-Nets, like S_1. However, since the input to the second step interpolation, $\rho^{S_1}_t$ and $\rho^{\mathcal{A}}_t$, is two smoke frames for the same arbitrary time t, interpolation for t is not performed. In place of the arbitrary time t, an arbitrary-state field α (where $\alpha \in (0,1)^{H\times W\times D}$) is generated through the flow computation network of S_2. The final interpolation result is then generated through the arbitrary-state flow estimation network using the arbitrary-state field (see Figure 3(b)). $O^{S_1\to\mathcal{A}}$ and $O^{\mathcal{A}\to S_1}$ are respectively forward and backward flows between the two input smoke frames $\rho^{S_1}_t$ and $\rho^{\mathcal{A}}_t$. $O^{\alpha\to S_1}$ and $O^{\alpha\to\mathcal{A}}$ are flows from the arbitrary-state field α to $\rho^{S_1}_t$ and $\rho^{\mathcal{A}}_t$, respectively. $V^{\alpha\leftarrow S_1}$ and $V^{\alpha\leftarrow\mathcal{A}}$ are visibility maps. The final interpolated smoke frame $\rho^{S_2}_t$ is computed as

$$\begin{aligned} \rho^{S_2}_t = \frac{1}{Z^{S_2}} \otimes \Big(&(1-\alpha)V^{\alpha\leftarrow S_1} \otimes \mathcal{W}(\rho^{S_1}_t, O^{\alpha\to S_1}) \\ &+ \alpha V^{\alpha\leftarrow\mathcal{A}} \otimes \mathcal{W}(\rho^{\mathcal{A}}_t, O^{\alpha\to\mathcal{A}})\Big), \end{aligned} \tag{6}$$

where $Z^{S_2} = (1-\alpha)V^{\alpha\leftarrow S_1} + \alpha V^{\alpha\leftarrow\mathcal{A}}$ is a normalization factor. The final result $\rho^{S_2}_t$ is generated by adaptive interpolation in the direction between $\rho^{S_1}_t$ and $\rho^{\mathcal{A}}_t$ that is closer to the ground truth.

As shown in the left column of Figure 4, the result of $\rho_t^{\mathcal{A}}$ (blue) is more accurate and less blurry than the result of $\rho_t^{S_1}$ (green). The final result $\rho_t^{S_2}$ is produced closer to $\rho_t^{\mathcal{A}}$, and it can be confirmed that $\rho_t^{S_2}$ is closest to the ground truth. On the other hand, in the right column of Figure 4, the result of $\rho_t^{S_1}$ (green) is more accurate and less blurry than the result of $\rho_t^{\mathcal{A}}$ (blue), so that the final result $\rho_t^{S_2}$ is produced closer to $\rho_t^{S_1}$. The loss function L^{S_2} for training the second interpolation network is defined as

$$L^{S_2} = \lambda_{\mathcal{R}} L_{\mathcal{R}}^{S_2} + \lambda_{\mathcal{W}} L_{\mathcal{W}}^{S_2} + \lambda_{\mathcal{S}} L_{\mathcal{S}}^{S_2} + \lambda_{\mathcal{T}} L_{\mathcal{T}}^{S_2}, \tag{7}$$

where $L_{\mathcal{R}}^{S_2}$, $L_{\mathcal{W}}^{S_2}$, and $L_{\mathcal{S}}^{S_2}$ are defined as follows:

$$L_{\mathcal{R}}^{S_2} = \frac{1}{N}\sum_N \left\| \rho_{t_i}^{S_2} - \bar{\rho}_{t_i} \right\|_1, \tag{8}$$

$$\begin{aligned} L_{\mathcal{W}}^{S_2} = \frac{1}{N}\sum_N \Big(& \left\| \mathcal{W}(\rho_{t_i}^{\mathcal{A}}, O_{t_i}^{S_i \to \mathcal{A}}) - \rho_{t_i}^{S_1} \right\|_1 \\ & + \left\| \mathcal{W}(\rho_{t_i}^{S_1}, O_{t_i}^{\mathcal{A} \to S_i}) - \rho_{t_i}^{\mathcal{A}} \right\|_1 \\ & + \left\| \mathcal{W}(\rho_{t_i}^{\mathcal{A}}, O_{t_i}^{\alpha \to \mathcal{A}}) - \bar{\rho}_{t_i} \right\|_1 \\ & + \left\| \mathcal{W}(\rho_{t_i}^{S_1}, O_{t_i}^{\alpha \to S_i}) - \bar{\rho}_{t_i} \right\|_1 \Big), \end{aligned} \tag{9}$$

$$L_{\mathcal{S}}^{S_2} = \left\| \nabla O^{S_1 \to \mathcal{A}} \right\|_1 + \left\| \nabla O^{\mathcal{A} \to S_1} \right\|_1, \tag{10}$$

where $L_{\mathcal{T}}^{S_2}$ is a loss function for temporal coherence of consecutive frames, and $\lambda_{\mathcal{T}}$ is a weight for the temporal coherence loss. As presented in [KAT*19], simply minimizing the reconstruction loss $L_{\mathcal{R}}^{S_2}$ using the L_1 distance does not guarantee that the temporal coherence matches the ground truth. We therefore include $L_{\mathcal{T}}^{S_2}$ to encourage temporal coherence between the generated temporal interpolation results and the input ground truth smoke frames, $\bar{\rho}_0$ and $\bar{\rho}_1$. $L_{\mathcal{T}}^{S_2}$ is defined as

$$L_{\mathcal{T}}^{S_2} = \frac{1}{N}\sum_N \left(\left\| \frac{d}{dt}\mathcal{C}(\bar{\rho}_0, \rho_{t_i}^{S_2}, \bar{\rho}_1) - \frac{d}{dt}\mathcal{C}(\bar{\rho}_0, \bar{\rho}_{t_i}, \bar{\rho}_1) \right\|_1 \right), \tag{11}$$

where $\mathcal{C}$ is a function that concatenates the three given smoke frames along the time t axis.

3.3. Implementation details

Training and test data preparation. We prepared ground truth smoke simulations via a physics-based simulation method for use in the training phase of the proposed method. We used MantaFlow [TP18], an open-source flow simulation library, to compute an accurate smoke simulation. The second through fifth columns in Table 1 list the simulation grid size, the number of smoke simulations used in the training phase, the number of frames for each simulation, and the average computation time for one frame of the smoke simulation. Each smoke simulation in the training data was generated with a small time step by randomly setting the smoke source location and the initial conditions of the velocity field. We computed each smoke simulation for the training data by setting the time step parameter in the MantaFlow simulator to 0.5. Additionally, we prepared test scenes for each smoke scenario by setting the time step parameter in the MantaFlow simulator to 4.0. Each training batch for temporal interpolation consisted of 17 frames of the smoke simulation. In other words, when we used the first and 17th simulation frames as the input to the proposed temporal interpolation network, 15 temporal interpolation frames (from the second to 16th frames) were generated by the proposed method. We used an Intel i5-8400K CPU at 2.80 GHz with 32 GB of memory to compute the physics-based simulation. The DNN models for the interpolation were trained on a GTX 1080 Ti GPU with 11 GB of memory.

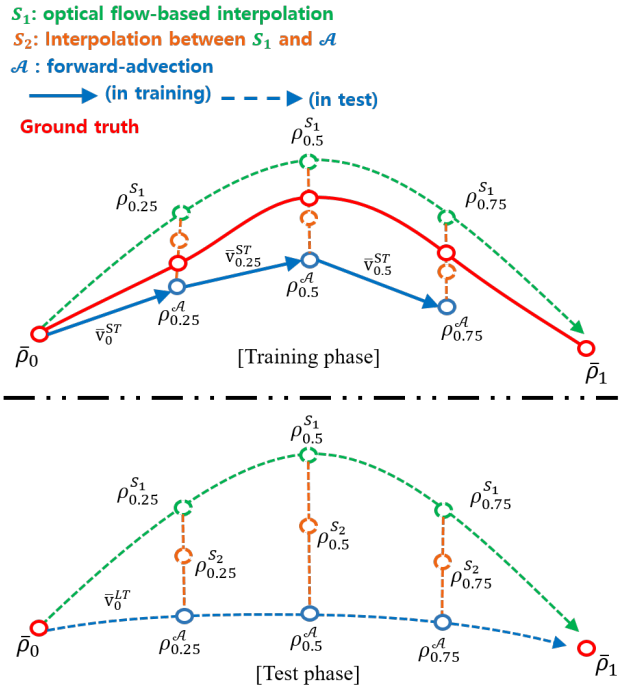


Figure 5: *A visual illustration of the training and test phases of the proposed method. The first interpolation (green dotted line) generates in-between smoke frames from two given smoke frames in the same manner in the training and test phases. However, the computation manners of the forward advection in the training and test phases (blue solid and dotted line) are different due to the difference in the velocity fields, $\bar{v}_0^{ST}$ and $\bar{v}_0^{LT}$.*

Training strategy for the second interpolation network. Unlike the first interpolation network that generates in-between smoke frames from two given smoke frames in the same manner in the training and test phases, the training and test phases of the second interpolation network were different in terms of the computation manner of the forward advection. Figure 5 is a visualization of the training and test phases of the proposed method. As shown in the test phase of Figure 5, the proposed method only has a physics-based simulation result computed with a large time step to convert to a high frame rate simulation, at which time the only velocity field available for the forward advection is $\bar{v}_0^{LT}$. Therefore, as described in Section 3.2, the forward advection in the second interpolation is computed using $\bar{\rho}_0$, $\bar{v}_0^{LT}$, and arbitrary time t. However, if the

Scenes	Simulation Grid Size	# of Simulations	# of Frames per Simulation	Simulation Time (s)	Training Time (h)	Inference Time(s)
Smoke2D & Fixed Circles	256×256	300	140	0.081	24	0.010
Smoke3D Plumes	80×80×80	100	120	0.266	25	0.159
Smoke3D & Fixed Bunny	128×192×128	50	160	6.909	46	0.584
Smoke3D & Moving Sphere	128×192×128	45	240	15.880	46	0.584
Smoke3D Plumes Bigger-res	240×320×240	-	-	67.168	-	9.483

Table 1: *Statistics on the quantity of training data used for each scenario in the experiment, the training time for the two-step interpolation network model, and the inference time needed for the proposed network to interpolate one frame of a smoke simulation.*

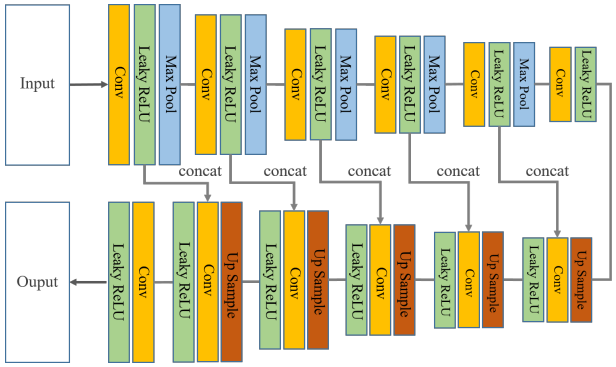


Figure 6: *Detail of the U-Net structure.*

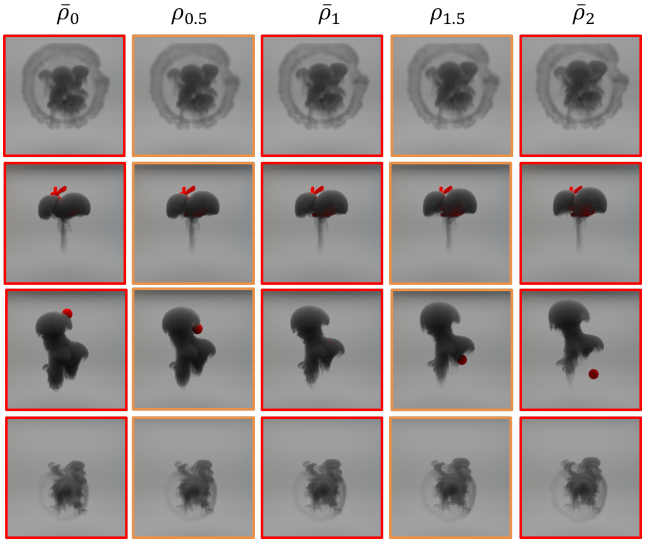


Figure 7: *Temporal interpolation result of the 3D smoke scenario. The in-between smoke frames (orange) are generated by interpolating between the two given smoke frames (red) using the proposed method.*

forward advection of the training phase is computed in the same way as in the test phase, the forward advection results are considerably different from the corresponding ground truth smoke as the arbitrary time t increases. This is because $\bar{v}_0^{ST}$, which is computed with a small time step, is different from the velocity field $\bar{v}_0^{LT}$ in the test step (see the training phase of Figure 5). When the forward advection results computed in the same way as the test step are considerably different from the ground truth smoke frames, it is difficult to train the second interpolation network and it generates worse interpolation results than those of the first step. Therefore, the forward advection results in the training phase are computed by using $\bar{\rho}_0$; the ground truth velocity fields computed with a small time step ($\bar{v}_0^{ST}$, $\bar{v}_{0.25}^{ST}$, and $\bar{v}_{0.5}^{ST}$); and a fixed arbitrary time t ($t = 0.25$ in case of Figure 5).

Network structure and hyperparameters. The proposed two-step temporal interpolation network uses U-Net [RFB15], which has been widely used as a base neural network model in image and video processing research. The encoder part of U-Net consists of 5 consecutively connected encoder blocks, a convolutional layer, and the leaky ReLU activation function. The encoder block consists of a convolutional layer, the leaky-ReLU activation function, and a max-pooling layer. The decoder part of U-Net consists of 5 consecutively connected decoder blocks, a convolutional layer, and the leaky-ReLU activation function. The decoder block consists of an up-sampling layer, a convolutional layer, and the leaky-ReLU activation function. Figure 6 shows the detailed network structure of U-Net. The proposed network models were implemented in Python with TensorFlow [AAB*15]. The Adam optimization method [KB15] was used for backward propagation with $\beta_1 = 0.9$ and a learning rate of 0.0001. The batch size of one epoch was 8 for 2D smoke and 2 for 3D smoke, with 30,000 and 10,000 epochs for 2D smoke and 3D smoke, respectively. The weighting of the loss terms $\lambda_{\mathcal{R}}$, $\lambda_{\mathcal{W}}$, $\lambda_{\mathcal{S}}$, and $\lambda_{\mathcal{G}}$ were empirically set to 0.1, 1.0, 100.0, and 100.0 when we trained the proposed method to generate the experimental results.

4. Experiments

The sixth and seventh columns of Table 1 show the training time of the proposed method for each smoke scene and the inference time required to generate one smoke frame with the proposed method. The temporal interpolation results of the *Smoke3D Plumes Bigger-res* scene were generated by applying the temporal interpolation model trained using *Smoke3D Plumes* simulation data. Therefore, columns related to the training data and training time of the *Smoke3D Plumes Bigger-res* are not presented in Table 1. The proposed method can generate smoke frames more efficiently than the physics-based simulation method. Figure 7 shows the results of the temporal interpolation of 3D smoke scenes. As can be seen in

Method	MSE(↓) ×1000	SSIM(↑)	LPIPS(↓) ×10	tLP(↓) ×10000
Butterflow	3.973	0.805	0.634	0.224
Sepconv	0.578	0.953	0.191	0.121
SuperSloMo	1.440	0.906	0.290	0.290
CycleGen	0.748	0.945	0.137	0.137
DAIN	1.275	0.927	0.255	0.135
Ours⊖	0.203	0.969	0.127	0.117
Ours	**0.190**	**0.970**	**0.120**	**0.089**

Table 2: *Statistics for comparison of the interpolation quality of our method with previous temporal interpolation methods. Ours⊖ denotes the proposed method without the temporal coherence loss, $L_{\mathcal{G}}^{S_2}$.*

Method	Simulation time (s)	Inference time (s)	Total time (s)
MantaFlow	103.635	-	103.635
TempoGAN	0.240	50.704	50.944
Multi-pass GAN	0.015	16.928	16.943
Ours	6.909	8.760	**15.669**
MantaFlow	3.990	-	3.990
DeepFluids	-	3.632	3.632
Ours	0.266	2.385	**2.651**

Table 3: *Statistics for comparison of the time performance of our method with that of previous smoke simulation generation methods.*

Figure 7, the proposed method was able to generate high-quality temporal interpolation results from the two input smoke frames.

In order to analyze the accuracy of the temporal interpolation results of the proposed method compared to that of previous temporal interpolation methods, we measured the mean square error (MSE), SSIM, and the LPIPS [ZIE*18] value. LPIPS is a perceptual similarity measure based on DNNs. In addition, in order to analyze the temporal coherence of the temporal interpolation results, we measured the tLP (tLPIPS) [CXM*20] score, which is the difference in the perceptual changes between consecutive smoke frames of the ground truth and consecutive smoke frames of the temporal interpolation results. Lower LPIPS and tLP values indicate that the temporal interpolation results are close to the ground truth in perceptual similarity and temporal coherency. A lower MSE value and a higher SSIM value indicate that the temporal interpolation results are close to the ground truth. In addition to the qualitative analysis of the proposed temporal interpolation method, we also compared its efficiency to that of previous smoke generation methods using DNNs. The efficiency was compared by measuring the computing time required to generate smoke simulations of the same grid resolution for the same number of frames.

4.1. Comparison with previous temporal interpolation methods

Since all previous temporal interpolation methods proposed in the computer vision field deal with general 2D video, it is difficult to apply them directly to interpolation between 3D smoke density frames. Therefore, we trained the previous temporal interpolation methods using the Smoke2D example dataset. The temporal interpolation results of the previous methods were then compared with the results of the proposed method. Among the comparison methods, Butterflow [Pha] is an open-source program implemented to increase the frame rate of a flow simulation by applying a motion-compensated frame interpolation method [LN10] rather than a DNN. On the other hand, Sepconv [NML17b], SuperSloMo [JSJ*18], CyclicGen [LLLC19], and DAIN [BLM*19] are methods for frame interpolation of general video using DNNs.

Figure 8 compares the temporal interpolation results of smoke frames generated by previous methods with the results generated by the proposed method. The proposed and previous methods, except for Butterflow, generated appropriate results that are similar to the ground truth (see the green boxes in the first row and the red box in the second row of Figure 8). However, some of the temporal interpolation results generated by the previous methods were not similar to the ground truth (see the red boxes in the first row of Figure 8). In the results of Butterflow, CycleGen, and SuperSloMo it appears that two different smoke shape clouds are overlapping. In addition, the results of Butterflow and CycleGen exhibited a different smoke texture than the ground truth (see the green boxes in the second row of Figure 8). In the results of Sepconv and DAIN, some parts of the smoke disappeared, and the results of Sepconv, SuperSloMo, and DAIN have a blurry texture when compared to the ground truth. In contrast, the results generated by the proposed method are the most similar to ground truth in all areas, and they contain few artifacts.

Table 2 provides the average MSE, SSIM, LPIPS, and tLP values of the smoke frames generated by the proposed method and previous temporal interpolation methods for comparison. The values for the SuperSloMo method were computed after applying the first step interpolation S_1 of the proposed method. As can be seen from Table 2, the SSIM value of the proposed method were found to be the highest among the methods used for comparison, and the values of MSE, LPIPS and tLP were the lowest. We also measured the MSE, SSIM, LPIPS, and tLP values of the proposed method without the temporal coherence loss $L_{\mathcal{T}}^{S_2}$ (Ours⊖ in Table 2). Compared to the proposed method using the temporal coherence loss, the values of MSE, SSIM, and LPIPS were not significantly different. However, we found that the tLP value, which measures perceptual changes in consecutive frames, improved substantially when the temporal coherence loss was used.

To sum up the first comparison experiment, the proposed temporal interpolation method produced interpolation results between the input smoke frames that were closer to the ground truth than those generated by the comparison temporal interpolation methods. In addition, we found that using temporal coherence loss in the proposed method generated improved results in terms of time coherency.

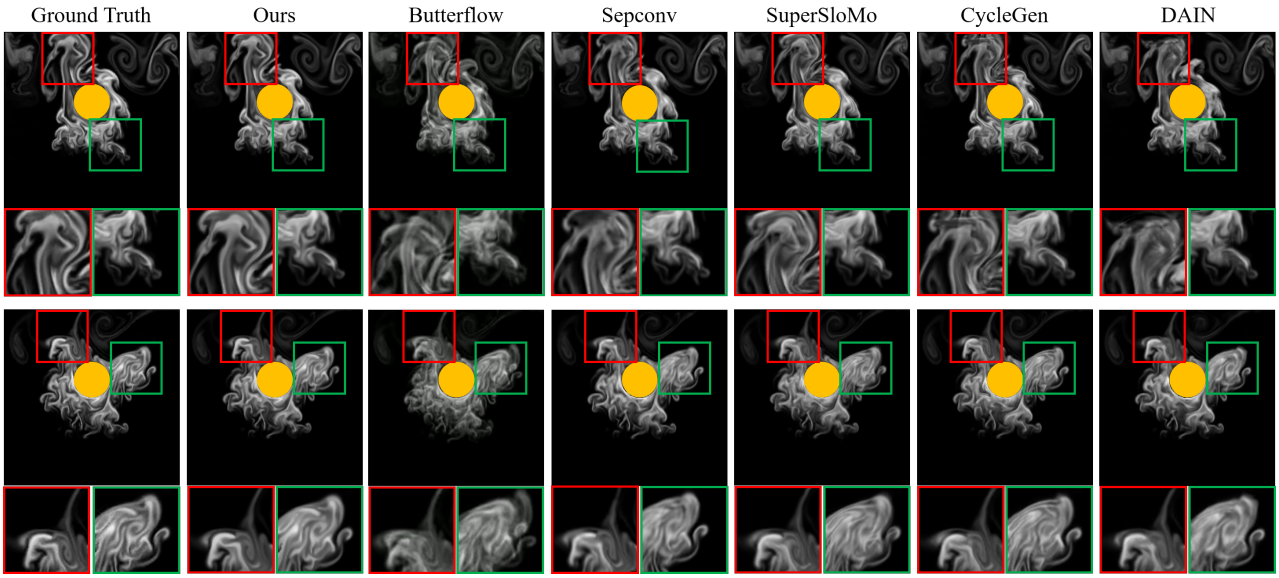


Figure 8: *Comparison of the interpolation results generated by the proposed method and those generated by previous temporal interpolation methods.*

4.2. Comparison of the efficiency of the proposed method with previous smoke generation methods using DNNs

In order to compare the performance of the proposed method with that of previous smoke generation methods that use DNNs, we implemented and trained the previous methods to generate a smoke density field of the same size as the smoke scene used in the experiment. Among the previous methods used for comparison, TempoGAN [XFCT18] and Multi-pass GAN [WXCT19] are super-resolution methods of smoke simulation that convert a low-resolution simulation into a high-resolution simulation. DeepFluids [KAT*19] is a flow movement generation method that uses auto-encoder networks without physics-based simulation computation.

The second to the fifth rows in Table 3 show the computation time required by MantaFlow, TempoGAN, Multi-pass GAN, and the proposed method to generate 16 frames of the *Smoke3D & Fixed Bunny* scenes. MantaFlow, which computes a physics-based simulation using a CPU-based solver, took 103.635 sec to compute 16 frames of the smoke scene. For TempoGAN and Multi-pass GAN, the computation of low-resolution simulations of $32 \times 48 \times 32$ and $16 \times 24 \times 16$, respectively, is required to generate a high-resolution smoke frame of $128 \times 192 \times 128$. TempoGAN required 0.24 sec for the low-resolution simulation computation and 50.704 sec for the network inference. Multi-pass GAN took 0.015 sec for the low-resolution simulation computation and 16.928 sec for the network inference. For the proposed method, one step of physics-based simulation computation was required to make two ground truth smoke frames for the temporal interpolation. The proposed method took 6.909 sec and 8.760 sec for the physics-based simulation computation and the network inference, respectively. These results confirm that the computation time required by the proposed method is smaller than that of the previous smoke generation methods.

Since the DeepFluids method was limited to training a $128 \times 192 \times 128$ smoke simulation grid using one graphic card, we performed the same experiment on the *Smoke3D Plumes* smoke scene, which has a smaller grid. The sixth to the eighth rows of Table 3 show the computation time needed for MantaFlow, DeepFluids, and the proposed method to generate 16 frames of the *Smoke3D Plumes* scene. MantaFlow required 3.990 sec to compute 16 frames of the smoke scene, and DeepFluids took 3.632 sec for the network inference. The proposed method took 0.266 sec and 2.385 sec for physics-based simulation computation and network inference, respectively. To sum up the second comparison experiment, the computation time required by the proposed method is smaller than that required by the previous smoke generation methods. The results confirm that the proposed method can more efficiently generate a smoke simulation than the physics-based simulation method or the methods using DNNs.

4.3. Comparison with the high frame rate simulation generated with a small time step

As the proposed method converts a low frame rate simulation computed with a large time step into a high frame rate simulation, its results are different from those of the physics-based simulation computed with a small time step under the same initial conditions. We compared a high frame rate simulation generated using the proposed method with a high frame rate simulation computed with a small time step under the same initial conditions. The first row of Figure 9 shows smoke frames generated by the proposed method, and the second row of Figure 9 shows the physics-based simulation results computed with a small time step. As shown in the second column of Figure 9, the interpolated frame is not significantly

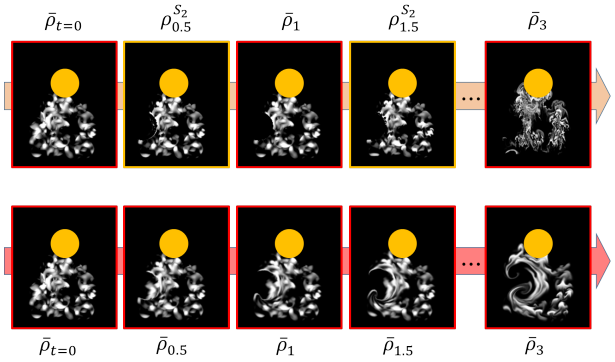


Figure 9: *Comparison with the high frame rate simulation generated with a small time step. Temporal interpolation results of the proposed method (top) are different from the physics-based simulation results computed with a small time step (bottom) due to the low frame rate simulation computed with a large time step.*

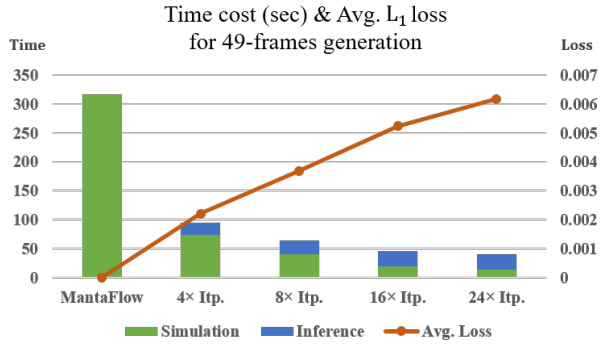


Figure 10: *Time cost (sec) and average L_1 loss values for generating* 49 *frames of the Smoke3D & Fixed Bunny scene when using the proposed method and increasing the frame rate by* 4, 8, 16, *and* 24 *times. The MantaFlow column corresponds to the case in which all frames are computed by the physics-based simulation.*

different from the corresponding frame computed by the physics-based simulation with a small time step. However, as can be seen in the third column of Figure 9, there is a difference between the simulation frame computed with a large time step and the corresponding frame computed with a small time step; this difference increases as time t increases (see the fourth and the fifth columns of Figure 9). As a result, the proposed method has a limitation in generating high frame rate simulations that are the same as the results of physics-based simulations computed with a small time step; this is owing to the difference between physics-based simulations that are computed with a large time step and with a small time step.

5. Discussion

When using the proposed method to generate smoke simulations of a specific frame length, it is important to appropriately choose the increased frame rate of the physics-based simulation based on the temporal interpolation. This is because the ratio between the number of physics-based simulation frames and the number of temporal interpolation frames varies according to the increased frame rate. For example, the results presented in Experiments Section for the proposed method are for 15 temporal interpolation frames being generated between two given input frames, which means that the interpolation effectively increases the frame rate of the physics-based simulation by about 16 times. To generate 49 frames of a smoke simulation using the proposed method with a frame rate that has been effectively increased 16 times, 4 frames of the physics-based simulation will be augmented by 45 frames of temporal interpolation.

In order to determine the most efficient increased frame rate for generating a smoke simulation of a specific frame length, we also experimented on trained models that increase the frame rate by 4, 8, 16, and 24 times. Figure 10 shows the time costs of the proposed method when increasing the frame rate by 4, 8, 16, and 24 times to generate 49 frames of the *Smoke3D & Fixed Bunny* scene. The MantaFlow column in Figure 10 corresponds to the case in which the physics-based simulation computes all frames. As the increased frame rate goes up, we can see the time costs required for the physics-based simulation decrease sharply when compared to MantaFlow (see the green blocks in Figure 10). On the other hand, the time costs required for network inference increase as the increased frame rate goes up (see the blue blocks in Figure 10).

When considering the change in the combined time cost of the physics-based simulation and the network inference, we can see that the time cost can be reduced most when the increased frame rate is highest. However, it is not best to choose the highest increased frame rate because the average L_1 loss of the temporal interpolation frames increases linearly as the increased frame rate goes up (see the orange graph in Figure 10). This is because the time step between the physics-based simulation frames increases when the increased frame rate goes up, reducing the accuracy of the temporal interpolation method. In fact, when increasing the frame rate by 24 times, we distinguished inaccurate temporal interpolation frames and physics-based simulation frames in the result due to the increasing loss (see the smoke simulation results for the Discussion in the supplementary video). As a result of this experiment, we determined that increasing the frame rate by 16 times is most efficient for reducing time cost and producing stable simulation results when generating a high frame rate simulation

As shown in Table 3, the inference time of the DNN model in the proposed method is lower than that of the efficient smoke generation methods that use DNNs. However, since one step of physics-based simulation computation for temporal interpolation in the proposed method requires more computation time than the low-resolution simulation computation time of the Multi-pass GAN method, the total time required by the proposed method to generate the smoke simulation is not significantly different from that required by Multi-pass GAN. The proposed method requires a simulation computation with the same grid resolution as the target simulation conditions for the temporal interpolation, whereas the TempoGAN and Multi-pass GAN methods use a simulation computation at a grid resolution that is 4 to 8 times smaller than the target simulation. Therefore, if the proposed method is combined

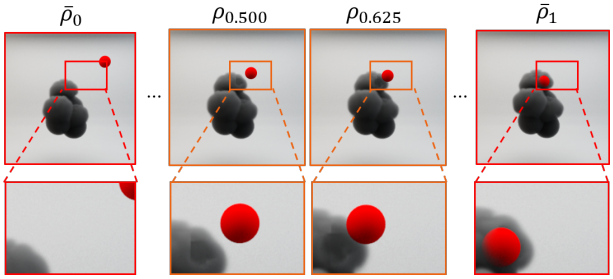


Figure 11: *Limitation of the proposed method. In the temporal interpolation results (orange) of the two given smoke frames (red), the smoke shapes are deformed before the external object (sphere) penetrates the smoke volume.*

with one of these super-resolution smoke simulation methods, even more efficient smoke simulation generation is possible. By using the Multi-pass GAN method for the one-step physics-based simulation computation in the proposed method, 4.793 sec of the physics-based simulation computation time can be saved. That is, it can increase the efficiency of the smoke simulation generation by about 30% compared to the current results.

Through the *Smoke3 & Moving Sphere* scene example, we can see that the proposed method is capable of appropriate temporal interpolation even when a linearly moving object passes through the smoke. However, since the temporal interpolation network does not have additional information to accurately predict the motion of the external object other than the two smoke density field frames, inaccurate temporal interpolation results may be generated in situations in which a moving object passes through the smoke. Figure 11 shows examples of inaccurate temporal interpolation. Among the temporal interpolation results (orange) of the two given smoke frames (red), we can see that the smoke shapes are deformed before the external object (sphere) penetrates the smoke volume (see the second and third columns in Figure 11). To address this limitation, further study of a temporal interpolation network that separately considers the movement features of external objects is needed. Like the approach presented in [KAT*19], defining latent variables for the external objects and environment and using the latent variables in temporal interpolation could enable the proposed method to be used in various kinds of smoke scenes. In addition, although the proposed method performs a second interpolation step to reduce artifacts that occur in the first interpolation step, it is sometimes difficult to generate detailed flows of in-between smoke frames due to the large range of the expected in-between smoke states. To address this limitation, further research into a temporal interpolation network that considers the detailed flows of the smoke frame is required. Similar to the approach presented in [CT17], proceeding with the second interpolation with the high-resolution smoke frame database could enable the proposed method to generate more efficient and high-quality interpolation results.

6. Conclusion

We have proposed a two-stage temporal interpolation network using forward advection for efficient smoke simulation generation. Since the proposed method converts a low frame rate smoke simulation computed with a large time step into a high frame rate smoke simulation through the inference of a temporal interpolation network, it can efficiently generate a high frame rate smoke simulation with a lower computational cost than the conventional simulation method. We performed quantitative analyses of the results generated by the proposed method and previous temporal interpolation methods. We also compared the computation time of the proposed method with that of previous methods that use DNNs to generate efficient smoke simulations. The experimental results indicate that the results generated by the proposed method are more accurate and closer to the ground truth smoke simulation than those generated by the previous temporal interpolation methods. The proposed method also generates smoke simulation results more efficiently than previous smoke simulation methods that use DNNs.

Acknowledgments. This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2020-2018-0-01419) supervised by the IITP (Institute for Information and Communications Technology Planning and Evaluation) and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2017R1A2B4005469).

References

[AAB*15] ABADI M., AGARWAL A., BARHAM P., BREVDO E., CHEN Z., CITRO C., CORRADO G. S., DAVIS A., DEAN J., DEVIN M., GHEMAWAT S., GOODFELLOW I., HARP A., IRVING G., ISARD M., JIA Y., JOZEFOWICZ R., KAISER L., KUDLUR M., LEVENBERG J., MANÉ D., MONGA R., MOORE S., MURRAY D., OLAH C., SCHUSTER M., SHLENS J., STEINER B., SUTSKEVER I., TALWAR K., TUCKER P., VANHOUCKE V., VASUDEVAN V., VIÉGAS F., VINYALS O., WARDEN P., WATTENBERG M., WICKE M., YU Y., ZHENG X.: TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. URL: https://www.tensorflow.org/. 6

[ATW15] ANDO R., THÜREY N., WOJTAN C.: A dimension-reduced pressure solver for liquid simulations. *Computer Graphics Forum 34*, 2 (2015), 473–480. 1, 2

[BLDL20] BAI K., LI W., DESBRUN M., LIU X.: Dynamic upsampling of smoke through dictionary-based learning. *ACM Trans. Graph. 40*, 1 (Sept. 2020). 1, 2

[BLM*19] BAO W., LAI W.-S., MA C., ZHANG X., GAO Z., YANG M.-H.: Depth-aware video frame interpolation. In *IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 3703–3712. 3, 7

[BLZ*19] BAO W., LAI W.-S., ZHANG X., GAO Z., YANG M.-H.: Memc-net: Motion estimation and motion compensation driven neural network for video interpolation and enhancement. *IEEE transactions on pattern analysis and machine intelligence* (2019). 2

[Che02] CHEN T.: Adaptive temporal interpolation using bidirectional motion estimation and compensation. In *IEEE International Conference on Image Processing* (2002), vol. 2. 1

[CHKK07] CHOI B., HAN J., KIM C., KO S.: Motion-compensated frame interpolation using bilateral motion estimation and adaptive overlapped block motion compensation. *IEEE Transactions on Circuits and Systems for Video Technology 17*, 4 (2007), 407–416. 2

[CT17] CHU M., THUEREY N.: Data-driven synthesis of smoke flows with cnn-based feature descriptors. *ACM Transactions on Graphics 36*, 4 (2017), 1–14. 1, 2, 10

[CXM*20] CHU M., XIE Y., MAYER J., LEAL-TAIXE L., THUEREY N.: Learning Temporal Coherence via Self-Supervision for GAN-based

Video Generation (TecoGAN). *ACM Transactions on Graphics 39*, 4 (2020). 7

[ESH06] EHRHARDT J., SÄRING D., HANDELS H.: Optical flow based interpolation of temporal image sequences. In *Medical Imaging 2006: Image Processing* (2006), vol. 6144. 1

[GNS*12] GOLAS A., NARAIN R., SEWALL J., KRAJCEVSKI P., DUBEY P., LIN M.: Large-scale fluid simulation using velocity-vorticity domain decomposition. *ACM Transactions on Graphics 31*, 6 (2012), 1–9. 1, 2

[GWC*19] GU D., WEN Z., CUI W., WANG R., JIANG F., LIU S.: Continuous bidirectional optical flow for video frame sequence interpolation. In *2019 IEEE International Conference on Multimedia and Expo (ICME)* (2019), pp. 1768–1773. 1

[HN08] HUANG A., NGUYEN T. Q.: A multistage motion vector processing method for motion-compensated frame interpolation. *IEEE Transactions on Image Processing 17*, 5 (2008), 694–708. 2

[JKJS05] JIEFU ZHAI, KEMAN YU, JIANG LI, SHIPENG LI: A low complexity motion compensated frame interpolation method. In *IEEE International Symposium on Circuits and Systems* (2005), vol. 5, pp. 4927–4930. 2

[JSJ*18] JIANG H., SUN D., JAMPANI V., YANG M.-H., LEARNED-MILLER E., KAUTZ J.: Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 9000–9008. 1, 2, 3, 7

[KAT*19] KIM B., AZEVEDO V. C., THUEREY N., KIM T., GROSS M., SOLENTHALER B.: Deep fluids: A generative network for parameterized fluid simulations. *Computer Graphics Forum 38*, 2 (2019), 59–70. 1, 2, 5, 8, 10

[KB15] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. In *3rd International Conference for Learning Representations* (2015). 6

[LJS*15] LADICKÝ L., JEONG S., SOLENTHALER B., POLLEFEYS M., GROSS M.: Data-driven fluid simulations using regression forests. *ACM Transactions on Graphics 34*, 6 (2015). 2

[LKA*16] LONG G., KNEIP L., ALVAREZ J. M., LI H., ZHANG X., YU Q.: Learning image matching by simply watching video. In *European Conference on Computer Vision* (2016), pp. 434–450. 2

[LLLC19] LIU Y.-L., LIAO Y.-T., LIN Y.-Y., CHUANG Y.-Y.: Deep video frame interpolation using cyclic frame generation. In *AAAI Conference on Artificial Intelligence* (2019), vol. 33, pp. 8794–8802. 3, 7

[LN10] LEE Y.-L., NGUYEN T.: High frame rate motion compensated frame interpolation in high-definition video processing. In *IEEE International Conference on Acoustics, Speech and Signal Processing* (2010), pp. 858–861. 7

[LYT*17] LIU Z., YEH R. A., TANG X., LIU Y., AGARWALA A.: Video frame synthesis using deep voxel flow. In *IEEE International Conference on Computer Vision* (2017), pp. 4463–4471. 1, 2

[MWZ*15] MEYER S., WANG O., ZIMMER H., GROSSE M., SORKINE-HORNUNG A.: Phase-based frame interpolation for video. In *IEEE Conference on Computer Vision and Pattern Recognition* (2015). 2

[NL18] NIKLAUS S., LIU F.: Context-aware synthesis for video frame interpolation. In *IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 1701–1710. 3

[NML17a] NIKLAUS S., MAI L., LIU F.: Video frame interpolation via adaptive convolution. In *IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 670–679. 2

[NML17b] NIKLAUS S., MAI L., LIU F.: Video frame interpolation via adaptive separable convolution. In *IEEE International Conference on Computer Vision* (2017), pp. 261–270. 7

[PBT18] PRANTL L., BONEV B., THUEREY N.: Generating liquid simulations with deformation-aware neural networks. In *International Conference on Learning Representations* (2018). 1, 2

[Pha] PHAM D.: Butterflow. URL: https://https://github.com/dthpham/butterflow. 7

[RFB15] RONNEBERGER O., FISCHER P., BROX T.: U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (2015), pp. 234–241. 3, 6

[RKEW19] REINHARDT S., KRAKE T., EBERHARDT B., WEISKOPF D.: Consistent shepard interpolation for sph-based fluid animation. *ACM Transactions on Graphics 38*, 6 (2019), 189. 1

[RRBW12] RAKÊT L. L., ROHOLM L., BRUHN A., WEICKERT J.: Motion compensated frame interpolation with a symmetric optical flow constraint. In *International Symposium on Visual Computing* (2012), pp. 447–457. 1

[SDN18] SATO S., DOBASHI Y., NISHITA T.: Editing fluid animation using flow interpolation. *ACM Transactions on Graphics 37*, 5 (2018), 1–12. 1

[SG11] SOLENTHALER B., GROSS M.: Two-scale particle simulation. *ACM Transactions on Graphics 30*, 4 (2011), 1–8. 1, 2

[Thu16] THUEREY N.: Interpolations of smoke and liquid simulations. *ACM Transactions on Graphics 36*, 1 (2016), 1–16. 1

[TP18] THUEREY N., PFAFF T.: MantaFlow, 2018. URL: http://mantaflow.com. 5

[TSSP17] TOMPSON J., SCHLACHTER K., SPRECHMANN P., PERLIN K.: Accelerating eulerian fluid simulation with convolutional networks. In *International Conference on Machine Learning* (2017), pp. 3424–3433. 1, 2

[UHT18] UM K., HU X., THUEREY N.: Liquid Splash Modeling with Neural Networks. *Computer Graphics Forum 37*, 8 (2018), 171–182. 1, 2

[WBT19] WIEWEL S., BECHER M., THUEREY N.: Latent space physics: Towards learning the temporal evolution of fluid flow. *Computer Graphics Forum 38*, 2 (2019), 71–82. 1, 2

[WKA*20] WIEWEL S., KIM B., AZEVEDO V. C., SOLENTHALER B., THUEREY N.: Latent space subdivision: Stable and controllable time predictions for fluid flow. *arXiv:2003.08723* (2020). 2

[WXCT19] WERHAHN M., XIE Y., CHU M., THUEREY N.: A multi-pass gan for fluid flow super-resolution. In *ACM Computer Graphics and Interactive Techniques* (2019), vol. 2, pp. 1–21. 1, 2, 8

[XFCT18] XIE Y., FRANZ E., CHU M., THUEREY N.: tempogan: A temporally coherent, volumetric gan for super-resolution fluid flow. *ACM Transactions on Graphics 37*, 4 (2018), 1–15. 1, 2, 8

[XYY18] XIAO X., YANG C., YANG X.: Adaptive learning-based projection method for smoke simulation. *Computer Animation and Virtual Worlds 29*, 3-4 (2018), e1837. 2

[YJL*16] YAN X., JIANG Y.-T., LI C.-F., MARTIN R. R., HU S.-M.: Multiphase sph simulation for interactive fluids and solids. *ACM Transactions on Graphics 35*, 4 (2016), 1–11. 1, 2

[ZIE*18] ZHANG R., ISOLA P., EFROS A. A., SHECHTMAN E., WANG O.: The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 586–595. 7