Fluid Volume Modeling from Sparse Multi-view Images by Appearance Transfer

Makoto Okabe[1,4] Yoshinori Dobashi[2,4] Ken Anjyo[3,4] Rikio Onai[1]
[1]The University of Electro-Communications [2]Hokkaido University [3]OLM Digital, Inc. [4]JST, CREST

Abstract

We propose a method of three-dimensional (3D) modeling of volumetric fluid phenomena from sparse multi-view images (e.g., only a single-view input or a pair of front- and side-view inputs). The volume determined from such sparse inputs using previous methods appears blurry and unnatural with novel views; however, our method preserves the appearance of novel viewing angles by transferring the appearance information from input images to novel viewing angles. For appearance information, we use histograms of image intensities and steerable coefficients. We formulate the volume modeling as an energy minimization problem with statistical hard constraints, which is solved using an expectation maximization (EM)-like iterative algorithm. Our algorithm begins with a rough estimate of the initial volume modeled from the input images, followed by an iterative process whereby we first render the images of the current volume with novel viewing angles. Then, we modify the rendered images by transferring the appearance information from the input images, and we thereafter model the improved volume based on the modified images. We iterate these operations until the volume converges. We demonstrate our method successfully provides natural-looking volume sequences of fluids (i.e., fire, smoke, explosions, and a water splash) from sparse multi-view videos. To create production-ready fluid animations, we further propose a method of rendering and editing fluids using a commercially available fluid simulator.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

Keywords: image-based modeling, natural phenomena animation, volume modeling, texture analysis/synthesis, single-view modeling

1 Introduction

Visual effects of fluid phenomena, such as fire, smoke, and explosion, are indispensable in modern movie production, and fluids have been important research topics in computer graphics for decades. However, the creation of fluid animations remains a difficult and time-consuming task, which often becomes a bottleneck in the production workflow.

There are three important methods that are commonly used to create fluid animations. The first is a physics-based fluid simulation, which allows users to design a wide variety of realistic fluids [Bridson and Müller-Fischer 2007]; however, this approach is often difficult to apply, even for experts, because the inputs to a fluid simulator (i.e., numerical parameters) are very different from the output (appearance, behavior, and shape of fluids). The second method is to make a composite of fluid videos [Bhat et al. 2004]. Production companies typically have databases of fluid videos, whereby the artist chooses an adequate video from it, and edits and overlays it onto the scene in the post-production process. However, because fluid videos are two-dimensional (2D), it is difficult to cope with 3D effects; camera paths or lighting cannot be edited, and stereoscopic effects are not possible. The third method is image-based modeling, which takes multi-view videos of fluid phenomena and models the volume sequence [Hasinoff and Kutulakos 2007; Ihrke and Magnor 2004; Gregson et al. 2012]. Here, the input data are videos (i.e., sequences of images), which clearly correspond to the output (i.e., a volume sequence). The resulting volume sequence appears realistic, and is suitable for 3D effects; however, problems result in that there are difficulties preparing multi-view inputs, which require a studio setup with a large number of cameras.

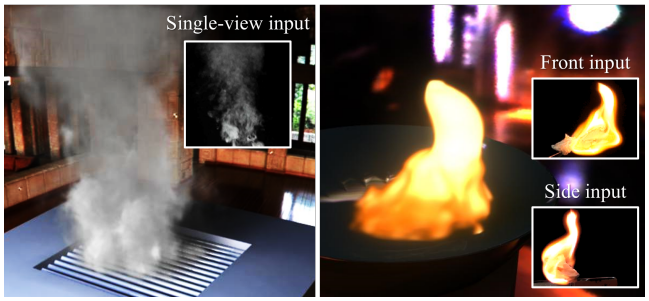


Figure 1: *We model the natural-looking volume of fluids from sparse multi-view images (e.g., only a single-view (left) or a pair of front and side views (right)). We create the production-ready fluid animation using a volume sequence of fluids modeled from sparse multi-view videos. A fluid simulator allows users to further edit the appearance, behaviour, and shape of fluids (e.g., adding more turbulence (left)).*

In this paper, we propose a method of image-based modeling of fluids, which takes sparse multi-view images (e.g., only a single-view or a pair of front and side views). Figures 1 and 2 illustrate typical results obtained by our method. As will be demonstrated in this paper, while significantly reducing the number of required cameras, our method still produces high-quality and editable 3D fluids. Our method can thus provide high-quality, efficiency and flexible usability at the same time, for modeling 3D fluids. This is the main feature of our method, which has never been achieved by previous work.

When reducing the number of inputs, the visual information may become insufficient to model the volume. The top row of Figure 2 shows the volume modeled from a pair of front- and side-view inputs using the least squares method (LSM) [Kak and Slaney 1988]. This appears blurry and unnatural with novel viewing angles. Here, we ask why these images appear blurry and unnatural. To answer this question, we assume the following: *We recall the appearance of the inputs (see the leftmost and rightmost panels of Figure 2), and expect that the volume should have a similar appearance, even with novel viewing angles.*

Our method is intended to live up to this expectation. Our technical

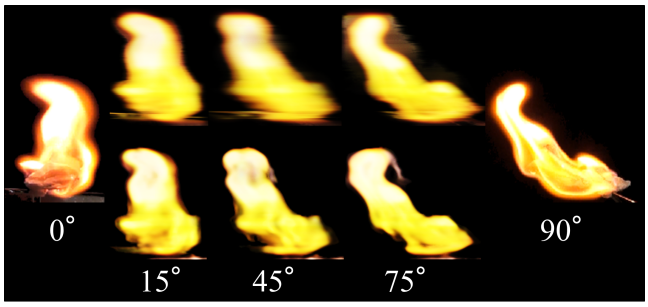


Figure 2: *The leftmost and rightmost images are a pair of front- and side-view inputs of fire. The top row shows the volume modeled using LSM, and the bottom row shows the volume modeled by our method. Since our method models the volume preserving the appearance with novel viewing angles, the volume appears more natural and more similar to the inputs.*

contribution lies in the novel energy function of volume modeling with appearance preservation and the simple and efficient method for minimizing it. As shown in the bottom row of Figure 2, the volume modeled using our method appears more similar to the inputs (i.e., sharper) with novel viewing angles. We perform a number of experiments, which clearly demonstrate the validity of the mathematical formulation. We find that the appearance information significantly improves the quality of the modeled volume, which is not possible by only pursuing the exactness of the volume.

We also note how artists use databases of fluid videos in a production workflow. The videos in the database are never used directly, and are always modified to obtain desired and never-seen-before animations. To satisfy such desires of the artists, we propose an efficient incorporation method using a commercially available fluid simulator, in which the user can further edit the fluid animation by modifying the rendering and simulation parameters. The resulting animations illustrate the potential of our approach in conjunction with graphics applications, including movie and video game production.

The purpose of our method is to not precisely reconstruct the volume, but rather, to effectively create a fluid animation using the volume sequence modeled from a sparse set of videos as a guide. We of course know that precise reconstruction from a single image or a pair of images is impossible. Our approach is particularly useful in cases where we use the input images as an initial reference, rather than the goal of precise reconstruction.

2 Related work

2.1 Image-based tomographic reconstruction

There are two major categories of tomographic reconstruction algorithm [Kak and Slaney 1988]. The first is direct methods (e.g., filtered back projection), which derive the analytical inverse conversion from the input images to a volume in advance using a Fourier transform and calculus. The second is iterative methods that formulate the relationship between the input images and a volume as an energy function that is minimized using various iterative algorithms. Direct methods typically require highly dense multi-view inputs; however, because these are often difficult to prepare for fluids, iterative methods are more commonly used for fluid volume modeling.

Hasinoff and Kutulakos proposed an image formation model for flames, and represented the volume as a linear combination of flame sheets; given the input images, the best combination is found in the least squares sense [2007]. Ihrke and Magnor modeled the volume of flames by minimizing the least squares energy using the conjugate gradient (CG) method [2004]. Atcheson et al. captured 3D gas flows by using the time-resolved Schlieren tomography system [2008]. Gregson et al. proposed the use of stochastic tomography, which minimizes the energy using random walks [2012]. This method empirically finds the pretty good solution to *convex* energy functions, which allows the inclusion of sophisticated regularization terms (e.g., L_1 energies) into the function (the convergence of the algorithm is discussed in more detail in [Gregson et al. 2013]). These iterative methods model fluid volumes with satisfactory appearance; however, they require a large number of images with multiple viewing angles as input data. We propose a novel *non-convex* energy function and an iterative algorithm to solve this problem, which achieves volumes with a satisfactory appearance, even from single-view images. Klehm et al. proposed tomography-based volume painting [2013] that takes an existing volume as input, and allows users to edit its color rather than its shape. Our goal, on the other hand, is to model the volumetric shape of fluids.

2.2 The other methods of fluid capture

Specialized hardware tools are proposed to capture fluid phenomena [Hawkins et al. 2005; Gu et al. 2013]; we wish to develop a more casual technique that non-experts can use. Liu et al. proposed the method of single-view modeling of smoke [2011], which represents smoke as a thin, smooth surface, and does not model volumetric fluid phenomena. Wenger et al. proposed the method for single-view modeling of astronomical nebulae, which models a maximally axisymmetric volume [2013]. Li et al. captures water phenomena from a single-view video, which are not volumetric but height field water surfaces [2013].

2.3 Solid texture synthesis

Solid texture synthesis methods model the volume from single or multiple example images. Pixel-based methods model volumetric textures with a satisfactory appearance [Heeger and Bergen 1995; Wei 2002; Qin and Yang 2007; Kopf et al. 2007; Dong et al. 2008]. Element-based methods model volumetric textures, while preserving the structures of example images or images that the user specifies [Jagnow et al. 2004; Takayama et al. 2008; Ma et al. 2013]. All of these methods are good at modeling volumetric textures in which high frequency texture patterns are dominant; however, they are difficult to use for modeling the overall shapes of fluids. Our method addresses this problem by filling the gap between texture synthesis and tomographic reconstruction approaches.

3 Our approach

We define a global energy function to mathematically formulate the assumption that *we recall the appearance of the inputs, and expect that the volume should have a similar appearance, even with novel viewing angles* (Section 1). Let $I = \{\mathbf{i}_\theta : \theta \in \Theta\}$ denote the set of inputs with multiple viewing angles, where Θ is the set of viewing angles at which the inputs are given. $\mathbf{i}_\theta$ represents the input image at a viewing angle of θ degrees. Let $\mathbf{v}$ denote the volume that we wish to model. We assume that each input $\mathbf{i}_\theta$ has dimensions of $w \times h$ pixels for all θ, and that the volume $\mathbf{v}$ has dimensions of $w \times h \times w$ voxels. We consider x-, y-, and z-axes in the volume space and u- and v-axes on the view plane. Since we consider the orthographic projection, the relations between the volume space and the view plane are $u = x\cos\theta + z\sin\theta$ and $v = y$, where θ is defined on the

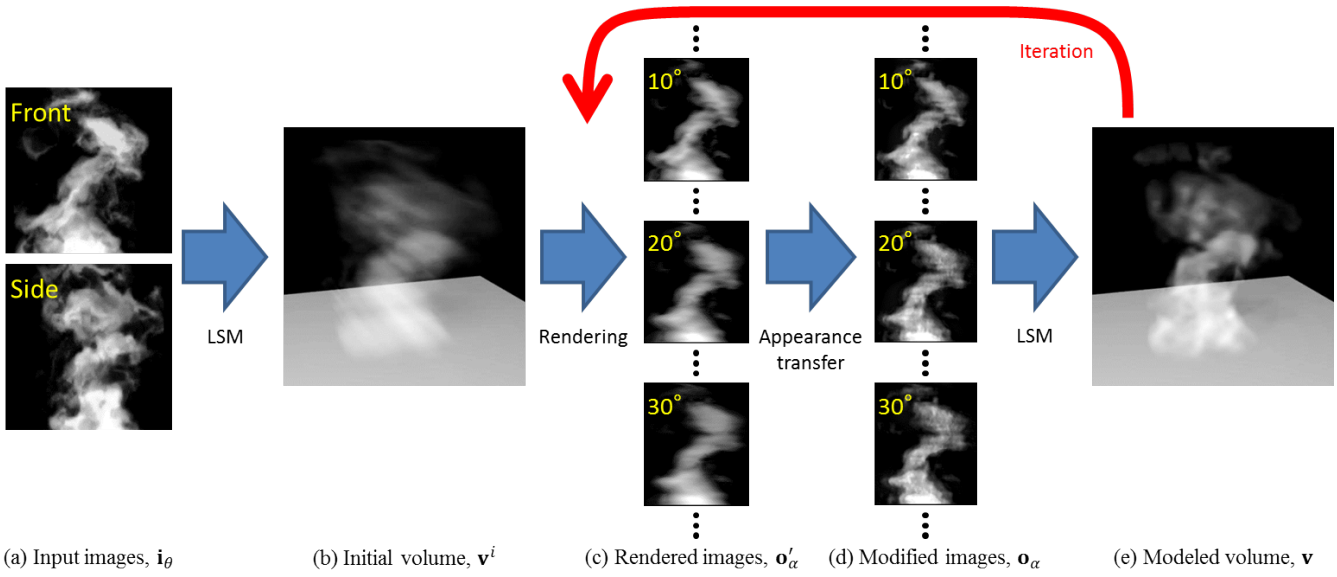


Figure 3: *Overview of our approach.*

x-z plane as the angle from the x-axis, rotating counter clockwise (when $\theta = 0$, the view plane is on x-axis).

Let $\mathcal{A}$ denote the set of viewing angles (in degrees) at which we want to measure the appearance similarity: we use 180 viewing angles horizontally around y-axis in all of our experiments (i.e., $\mathcal{A} = \{0, 1, 2, ..., 179\}$ and $\Theta \subset \mathcal{A}$). We consider a very large set of images T_α for each viewing angle of $\alpha (\in \mathcal{A})$ degrees, where $\mathbf{t} \in T_\alpha$ is an $w \times h$-pixel image that has a similar appearance to the input images I. To measure the similarity of the appearance of the images, we use the histogram of image intensities and steerable coefficients that is used for image analysis/synthesis [Heeger and Bergen 1995]. Let $\phi_k(\mathbf{t})$ denote the value in the k-th bin of the histogram of $\mathbf{t}$. We define $T_\alpha = \{\mathbf{t} : \phi_k(\mathbf{t}) = c_{\alpha,k}, \forall k\}$, where $c_{\alpha,k}$ is the value in the k-th bin of the histogram and is computed based on I in Section 3.2.

We define our energy function E as follows:

$$E = \sum_{\alpha \in \mathcal{A}} |B_\alpha \mathbf{v} - \mathbf{o}_\alpha|^2, \tag{1}$$

$$\text{subject to } \begin{cases} \mathbf{o}_\alpha = \mathbf{i}_\alpha & (\alpha \in \Theta) \\ \mathbf{o}_\alpha \in T_\alpha & (\alpha \notin \Theta) \end{cases} \tag{2}$$

where B_α is a matrix representing the ray-casting operation, and $B_\alpha \mathbf{v}$ renders the image of the volume $\mathbf{v}$ at a viewing angle of α degrees. Because we always consider fluids as emissive media, B_α adds a voxel value to each ray passing through it. We want the modeled volume $\mathbf{v}$ to appear the same as $\mathbf{i}_\alpha$ at the input views, and to appear similar to these inputs I at novel views, which Eq. 2 imposes as hard constraints (i.e., $\mathbf{o}_\alpha$ is equal to $\mathbf{i}_\alpha$ at the input views and belongs to T_α at novel views).

The energy function in Eq. 1 and Eq. 2 is non-convex and has an infinite number of local minima. To model the volume as a good local minimum, we begin with an initial volume, and estimate $\mathbf{v}$ iteratively by refining this volume to decrease the energy. Figure 3 shows an overview of our iterative algorithm. Given multi-view inputs (e.g., only front and side views (Figure 3-a)), we model the initial volume $\mathbf{v}^i$ (Figure 3-b) using LSM:

$$\mathbf{v}^i = \arg\min_{\mathbf{v}} \sum_{\theta \in \Theta} |B_\theta \mathbf{v} - \mathbf{i}_\theta|^2. \tag{3}$$

This initial volume appears the same as the input images with the input views but appears blurry and unnatural with novel views, similarly to that shown in the top row of Figure 2. Grid artifacts are visible when viewed from the top, as shown in Figure 3-b.

During each iteration, we alternate between $\mathbf{o}_\alpha$ and $\mathbf{v}$ as variables with respect to which the energy (see Eq. 1 and Eq. 2) is minimized. First, we fix $\mathbf{v}$, and update $\mathbf{o}_\alpha$ by minimizing the energy. We solve this minimization for each viewing angle independently, i.e., we update $\mathbf{o}_\alpha$ so that it minimizes $|B_\alpha \mathbf{v} - \mathbf{o}_\alpha|^2$ with the constraints of Eq. 2. Because this problem is difficult to precisely solve, we instead propose an approximate solution. We render the image of the current volume as $\mathbf{o}'_\alpha = B_\alpha \mathbf{v}$ (Figure 3-c). We modify the blurry and unnatural appearance of $\mathbf{o}'_\alpha$ so that the modified image $\mathbf{o}_\alpha$ has similar appearance to the inputs. We use a histogram-matching operation so that $\mathbf{o}_\alpha$ satisfies the statistical constraints $\phi_k(\mathbf{o}_\alpha) = c_{\alpha,k}$ for all k (Figure 3-d). Then, we fix $\mathbf{o}_\alpha$ and update $\mathbf{v}$: we model the volume from the modified images $\mathbf{o}_\alpha$ using LSM (Figure 3-e). The appearance of the modeled volume is improved compared to the previous iteration.

This iterative method is not identical to the expectation maximization (EM) algorithm; however, the two are algorithmically similar. Such EM-like approaches are commonly used in texture synthesis and video completion methods [Kwatra et al. 2005; Kopf et al. 2007; Wexler et al. 2007]. In our case, the E-step and M-step correspond to the updates of $\mathbf{v}$ and $\mathbf{o}_\alpha$, respectively. Our E-step performs an exact minimization via LSM; however, our M-step is an approximate solution. Now recall that T_α is the set of images prescribed by the histogram of the image intensities and steerable coefficients. If T_α were a convex manifold, and our modification corresponds to an orthogonal projection of $\mathbf{o}'_\alpha$ onto T_α, $\mathbf{o}_\alpha$ exactly minimizes the energy. As discussed in [Portilla and Simoncelli 2000], the histogram-matching operation does achieve an orthogonal projection, but T_α is non-convex. As a result, $\mathbf{o}_\alpha$ does not exactly minimize the energy, which is the major reason for difficulties in formally guaranteeing convergence of our iterative algorithm. Nevertheless, our

method did not fail to converge to a satisfactory solution for more than 1,600 examples that we have tested.

We describe the method to model the grayscale density volume from grayscale versions of input images, so that pixels and voxels have a single scalar value. The pixel values in the input images range from 0 to 255.

3.1 Volume modeling from images

In the E-step, or when modeling the initial volume $\mathbf{v}^i$, we model the volume using LSM, where we minimize the least squares energy of Eq. 1 or Eq. 3 using the method of [Ihrke and Magnor 2004]. This is performed by solving the linear system of equations $\frac{dE}{d\mathbf{v}} = \mathbf{0}$ using the CG method. According to the literature, we use the visual hull constructed from the input images to limit voxels that can have a non-zero value. Because the number of unknown variables whw is usually larger than the number of equations $wh|\mathcal{A}|$ or $wh|I|$, we require regularization: we terminate the iteration of the CG method to avoid overfitting according to the literature. We apply a 3D Gaussian blur operation with a small kernel to reduce the noise of the modeled volume at the end of each E-step, which we found models a smoother result. The CG method is suitable for implementation on a graphics processing unit (GPU), whereby the computation will become approximately 30 times faster than our implementation on a central processing unit (CPU).

We used the CG method, solely due to simplicity of implementation and low computational cost. However, other iterative methods can be used. We experimented with using stochastic tomography [Gregson et al. 2012]. The quality of the modeled volume was slightly better than with the CG method in our case; however, the computational cost was significantly greater.

3.2 Appearance transfer

We apply the pyramid-based texture analysis/synthesis method [Heeger and Bergen 1995] to transfer the appearance of input images to modify the blurry and unnatural appearance of the rendered images (see the top row of Figure 2). The original method takes two images as inputs: one is a texture exemplar and the other is a random dot image. The method transfers the appearance of the texture exemplar to the random dot image, and the random dot image is converted into an image with an appearance that is similar to the texture exemplar. This transfer process is achieved via a histogram-matching operation on the image intensities, and steerable coefficients between the images. Similarly with the original method, we use the third-order steerable pyramid, which decomposes an image into an image pyramid with four orientations. Figure 4 shows an example.

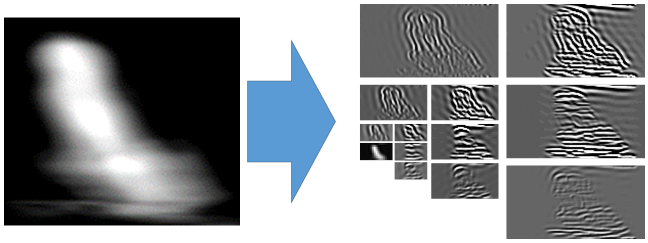


Figure 4: *The four scale, third-order steerable pyramid decomposition (the component of the highest frequency residual is omitted).*

Since we have sparse collections of images with multiple viewing angles, we do not use the histogram of one of them, but rather, we make a target histogram by interpolating the histograms of the two

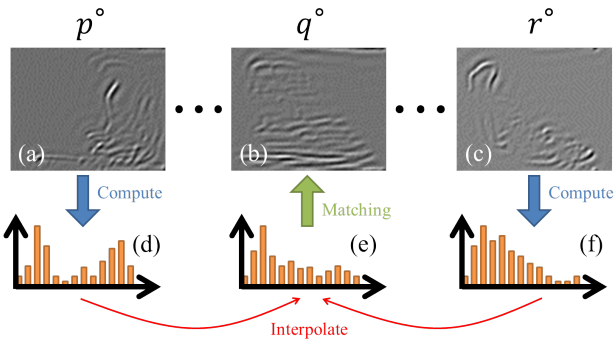


Figure 5: *Histogram interpolation and matching.*

nearest input images. Figure 5 shows an example of this process, where input images $\mathbf{i}_p$ and $\mathbf{i}_r$ are given at viewing angles of p degrees and r degrees. Figures 5-a and 5-c are the third-scale, fourth orientation sub-bands of the steerable pyramids of the inputs. Figure 5-b shows the corresponding sub-band of $\mathbf{o}'_q$, to which we transfer the appearance of $\mathbf{i}_p$ and $\mathbf{i}_r$ ($p < q < r$). Figures 5-d and 5-f show the histograms of the sub-bands. We linearly interpolate the value of each histogram bin to create the target histogram shown in Figure 5-e. Finally, we apply a histogram-matching operation to modify the sub-band of $\mathbf{o}'_q$ (Figure 5-b) so that it has the target histogram (Figure 5-e).

Here, we describe this process in mathematical form. We consider the histogram for an image $\mathbf{i}$, which is formed by concatenating all of the histograms of image intensities and steerable coefficients of $\mathbf{i}$. Let $\phi_k(\mathbf{i}_p)$ and $\phi_k(\mathbf{i}_r)$ denote the k-th bin of the concatenated histogram of $\mathbf{i}_p$ and $\mathbf{i}_r$. We linearly interpolate these to create the k-th bin of the target histogram:

$$c_{q,k} = \frac{(r-q)\phi_k(\mathbf{i}_p) + (q-p)\phi_k(\mathbf{i}_r)}{r-p}. \tag{4}$$

The histogram-matching operation modifies $\mathbf{o}'_q$ so that the modified image $\mathbf{o}_q$ satisfies $\phi_k(\mathbf{o}_q) = c_{q,k}$ for all k.

Our implementation of appearance transfer is similar to the original method [Heeger and Bergen 1995]. We independently apply the histogram-matching operation to each sub-band of the steerable pyramid of $\mathbf{o}'_q$. Then, we collapse the steerable pyramid to reconstruct the image to which we finally apply the histogram-matching operation of the image intensities to obtain $\mathbf{o}_q$.

3.2.1 Validity of linear interpolation

Histogram interpolation is based on the observation that histograms of rendered images of a volumetric scene change relatively smoothly as a function of the viewing angle. We render the images of the volume of the synthetic fire between viewing angles of 0 degrees and 90 degrees with 5-degree intervals, resulting in 19 rendered images. Figures 6-a and 6-c show the rendered images at 0 degrees and 90 degrees. Figures 6-b and 6-d show histograms of the image intensities. The transition of the overall shapes of the histograms between 0 degrees and 90 degrees is relatively smooth; however, it is often non-linear. For example, the black lines in Figures 6-e, 6-f, and 6-g show the observed transitions of the histograms at the 5th, 13th, and 21st bin, respectively. Whereas the transition at the 5th and 21st bins was approximately linear, the transition at the 13th bin was highly non-linear. Nevertheless, our method linearly interpolates all of the bins, as shown by the orange dotted lines. The reason for the choice of linear interpolation

is that it is simple to implement and guarantees smoothness of the transition. We have discussed histograms of image intensities here; however, the same discussion applies to histograms of steerable coefficients (the steerable pyramid is a linear image decomposition).

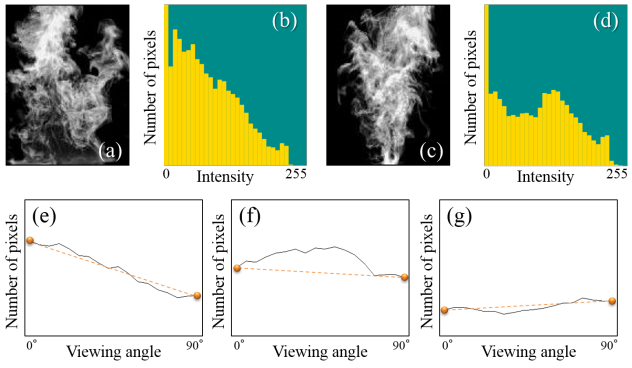


Figure 6: *Transitions of the histogram bins between viewing angles of 0 degrees and 90 degrees.*

Although linear interpolation is not always accurate, we demonstrate its effectiveness by comparing the results of appearance transfer with and without linear interpolation. The top row of Figure 7 shows the rendered images of the volume modeled using Figures 6-a and 6-c. We apply appearance transfer with linear interpolation, and the second row of Figure 7 shows the results: the rendered images are successfully modified. In the third row of Figure 7, we apply appearance transfer without linear interpolation (i.e., we transfer the appearance information of the front input (Figure 6-a) and modify the images of all of the viewing angles). The modified image at 15 degrees is similar to that of the second row; however, artifacts appeared around the fire at viewing angles of 45, 60, and 75 degrees. Figure 8 emphasizes the artifacts showing the same images as the red and green rectangles in Figure 7 using the jet colormap. Such artifacts introduce artifacts in the resulting volume in the E-step.

Even if a more sophisticated non-linear interpolation is used, we doubt that it will significantly improve the quality of appearance transfer. The fourth row of Figure 7 shows rendered images of the synthetic fire. The bottom row of Figure 7 shows the images modified using appearance transfer using the ground truth histograms (i.e., the histograms of the images in the fourth row of Figure 7 were used to modify the images). The qualities of the second and bottom rows of Figure 7 are comparable visually, meaning that even if a more sophisticated non-linear interpolation is used to produce the ground truth histograms, they will not improve the quality of appearance transfer significantly.

4 Animation with a fluid simulator

Given sparse multi-view video data as input, we apply our method described in Section 3 to the corresponding video frames, and model the volume sequence. To further create a production-ready fluid animation, we address the following two issues. First, because we model the volume independently on a frame-by-frame basis, temporal coherence is not guaranteed. Second, we wish to allow users to further edit the appearance, behavior, and shape of fluids. To improve the temporal coherence and editability, we present a simple, but efficient method to incorporate our method into a fluid simulator. In principle, any fluid simulator can be used; however, we chose Autodesk Maya 2015 and its built-in simulator. To use the simulator, we should estimate the velocity fields from the input

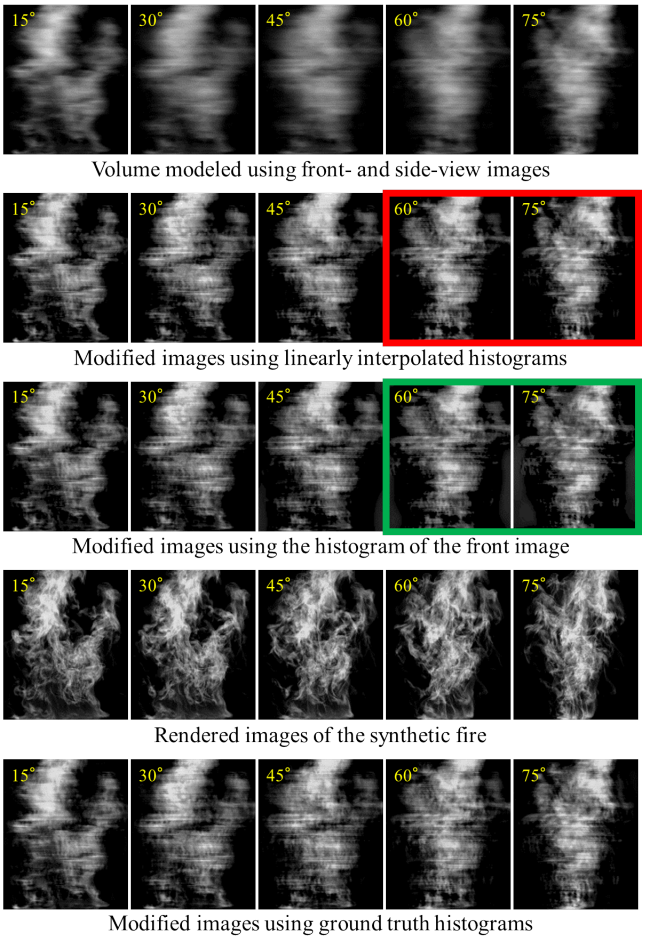


Figure 7: *Modified images using various histograms.*

videos. Gregson et al. proposed a sophisticated method to estimate the physically plausible dense velocity fields from multi-view video images [Gregson et al. 2014]: since our goal is not to estimate precise velocity fields, but rather to improve the quality of fluid animations, the simpler method will suffice.

We estimate the approximate 3D optical flow using the input videos, and then use this to control fluid motions using the simulation. Let $\boldsymbol{\omega}^t$ denote the approximate 3D optical flow at the t-th frame that we wish to estimate, and let $\boldsymbol{\omega}_x^t$, $\boldsymbol{\omega}_y^t$, and $\boldsymbol{\omega}_z^t$ denote the x-, y-, and z-components thereof. Let $\mathbf{f}_{u,\theta}^t$ and $\mathbf{f}_{v,\theta}^t$ denote horizontal and vertical components of the 2D optical flow computed using [Werlberger et al. 2010] between t-th frame and $(t+1)$-th frame in the input video at a viewing angle of θ degrees. The size of $\boldsymbol{\omega}^t$ is $w \times h \times w$

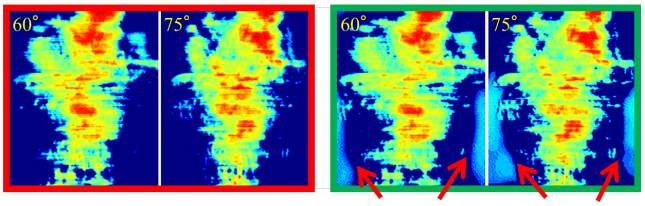


Figure 8: *Artifacts pointed by the arrows appear around fire without interpolating histograms between views.*

voxels, which is the same as all of the simulation grids in the simulator. The size of $\mathbf{f}^t_{u,\theta}$ and $\mathbf{f}^t_{v,\theta}$ is $w \times h$ pixels. We estimate $\boldsymbol{\omega}^t$ by taking the weighted average of the 2D optical flow for all the viewing angles of the input videos:

$$\boldsymbol{\omega}^t_x(\mathbf{x}) = \frac{\sum_{\theta\in\Theta} \mathbf{f}^t_{u,\theta}(x\cos\theta + z\sin\theta, y)\cos\theta}{\sum_{\theta\in\Theta}|\cos\theta|}, \quad (5)$$

$$\boldsymbol{\omega}^t_y(\mathbf{x}) = \frac{\sum_{\theta\in\Theta} \mathbf{f}^t_{v,\theta}(x\cos\theta + z\sin\theta, y)}{|\Theta|}, \quad (6)$$

$$\boldsymbol{\omega}^t_z(\mathbf{x}) = \frac{\sum_{\theta\in\Theta} \mathbf{f}^t_{u,\theta}(x\cos\theta + z\sin\theta, y)\sin\theta}{\sum_{\theta\in\Theta}|\sin\theta|}, \quad (7)$$

where $\mathbf{x} = (x, y, z)^T$ is a voxel position.

Let $\boldsymbol{\psi}^t$ denote the velocity field of the simulator at t-th frame, which we wish to modify in the manner similar to [Thürey et al. 2006]. While we wish to control the low frequency, overall fluid motions in $\boldsymbol{\psi}^t$ by the approximate 3D optical flow $\boldsymbol{\omega}^t$, we wish to preserve the high frequency, detailed fluid motions in $\boldsymbol{\psi}^t$:

$$\mathbf{r} = \boldsymbol{\psi}^t - G(\boldsymbol{\psi}^t), \quad (8)$$

$$\boldsymbol{\psi}' = G(\boldsymbol{\omega}^t) + \mathbf{r}. \quad (9)$$

Let $G()$ denote a 3D Gaussian blur operation, and the high frequency, detailed components of $\boldsymbol{\psi}^t$ are extracted as the residual $\mathbf{r}$ in Eq. 8. To control the overall fluid motions using $\boldsymbol{\omega}^t$, its low frequency components $G(\boldsymbol{\omega}^t)$ are added to $\mathbf{r}$ in Eq. 9. This preservation of these details is important: if we simply use the approximate 3D optical flow as the velocity field (i.e., $\boldsymbol{\psi}' = \boldsymbol{\omega}^t$), the interesting fluid phenomena created by the simulator (e.g., vortices and turbulence), would disappear. Also, note that the replacement of low frequency components in Eq. 8 and Eq. 9 (i.e., $G(\boldsymbol{\psi}^t)$ with $G(\boldsymbol{\omega}^t)$) does not guarantee temporal coherence, which means that the temporal coherence of the approximate 3D optical flow $\boldsymbol{\omega}^t$ is required.

Let $\mathbf{v}^t$ denote the volume modeled using our method at t-th frame, and let $\mathbf{d}^t$ denote the density field of the simulator at t-th frame. We modify the density field as follows:

$$\mathbf{d}'(\mathbf{x}) = \begin{cases} \mathbf{v}^t(\mathbf{x})\lambda + \mathbf{d}^t(\mathbf{x})(1-\lambda) & (\mathbf{d}^t(\mathbf{x}) < \mathbf{v}^t(\mathbf{x})) \\ \mathbf{d}^t(\mathbf{x}) & (\mathbf{d}^t(\mathbf{x}) \geq \mathbf{v}^t(\mathbf{x})) \end{cases} \quad (10)$$

where λ is a weight describing how much the density is updated by the volume (we used $\lambda = 0.2$). Eq. 10 adds the volume to the density at voxel positions where the density value is smaller than the voxel value. We perform addition, but do not perform subtraction in Eq. 10. Instead, we manipulate the dissipation parameter of the simulator, which defines the rate at which the density gradually vanishes with time. The resulting animation is smoother than when performing the subtraction in Eq. 10.

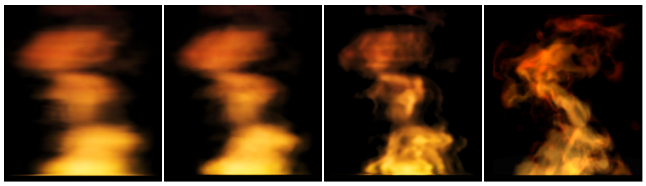
Given the modified velocity and density fields, $\boldsymbol{\psi}'$ and $\mathbf{d}'$, we perform the simulation using the built-in simulator of Autodesk Maya 2015, which solves the Navier–Stokes equations, and updates all of the simulation grids (i.e., we obtain $\boldsymbol{\psi}^{t+1}$ and $\mathbf{d}^{t+1}$). We start the simulation with $\boldsymbol{\psi}^0 = \boldsymbol{\omega}^0$ and $\mathbf{d}^0 = \mathbf{v}^0$, iterate these processes to create the animation frames. The smooth advections and gradual update of the density (Eq. 10) guarantee temporal coherence. The use of the simulator also allows the user to further edit the shading and physics parameters to control the fluid phenomena.

5 Results and discussion

We apply our method to real and synthetic multi-view input videos. The supplementary video shows the resulting fluid animations. To apply color to fluid animations, we use the built-in shader from Autodesk Maya 2015, where a density value is mapped to a color. For each fluid animation, we manually select appropriate parameters for shading and fluid simulation. We used the light probes from Paul Debevec's light probes gallery [Debevec 2001; Debevec 2006] to render some of the fluid animations.

We modeled the fluids using a machine with an Intel(R) Core(TM) i7–4820K CPU (3.70 GHz), 16.0 GB of memory, and an NVIDIA GeForce GTX 770 GPU. The computational expense was proportional to the volume resolution. For the real fire (Figure 2), since the size of the input images is 271×144 pixels, we modeled the volume whose size is $271 \times 144 \times 271$ voxels. We measured the average time for each step: initialization required 10.6 seconds, the E-step required 15.9 seconds, and the M-step required 14.0 seconds. We iterated the algorithm eight times, and the total time to model the volume was 250 seconds. Since our method described in Section 3 does not take temporal coherence into account, we can process video frames independently. We distributed video frames to multiple computers and modeled volumes in parallel. Speed-up is simply proportional to the number of computers.

Front- and side-view input videos of real fire Figures 1-right and 2 show the volume modeled from a pair of front- and side-view images of real fire. To capture videos of the fire, we used a simple studio setup shown in the supplementary video. Two Sony Handycam digital cameras were placed so that the optical axes were orthogonal. We created a fire by burning paper on a round chair. The background was covered using dark cloth to create efficient matting. Since the distance between each camera and fire was long enough to assume the orthogonal projection, we did not use camera calibration, and instead aligned the timing, scale, and translation manually between the two videos using commercially available video editing tools. Given these two aligned videos, our method automatically models the volume for each frame.



(a) LSM (b) ST-SAD (c) Our method (d) Ground truth

Figure 9: *Volumes modeled by LSM, ST-SAD, and our method from a pair of front- and side-view inputs of synthetic fire.*

Sparse multi-view images of synthetic fire Figure 9 shows the volume modeled from a pair of front- and side-view images of synthetic fire (Figure 3-a). We prepared the inputs using a fluid simulator and a volume rendering technique. Figure 9 shows the volumes modeled by LSM, stochastic tomography with an L_1 sum-of-absolute-differences regularization (ST-SAD) [Gregson et al. 2012], and our method (described in Section 3 without the use of a fluid simulator). For ST-SAD, we used 4 billion mutations and 10,000 sample chains. Figure 9 shows the volumes at an intermediate (45-degree) viewing angle. Compared with the volumes modeled using LSM and ST-SAD, our result appears sharper and more similar to the inputs (Figure 3-a) even with novel views. While ST-SAD models the volume constraining the internal structures to have smoothness and sparse sharp edges, our method does not constrain the internal structures, but it constrains only the external appearance. Figure 9-d is the ground truth image (i.e., the original simulation data seen at a viewing angle of 45 degrees). Compared

with the ground truth, our result appears different; however, the overall atmosphere and textures appear similar. Figure 10 shows another experiment using synthetic fire: we prepared six input images rendered with 30-degree intervals. We rendered the images of the volumes modeled by LSM, ST-SAD, and our method with a novel viewing angle of 15 degrees (see the bottom row of Figure 10). Our result still appears blurry compared with the ground truth, but captures the features better than LSM and ST-SAD.

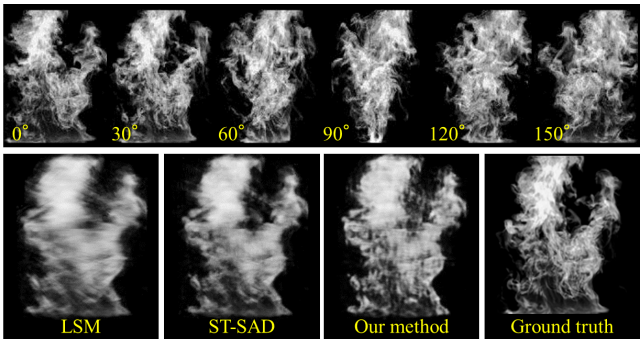


Figure 10: *We take the six-view inputs of synthetic fire (top) and model the volumes by LSM, ST-SAD, and our method (bottom).*

Single-view modeling Figures 1-a and 11 show volumes modeled from single-view videos of fluids. Figure 11 gives the information on the resolution and number of frames of each input video. With a single input, we applied our method assuming that a pair of front- and side-view inputs exists, where the two images are identical. The resulting volume is symmetrical, as observed in the horizontal slice of each volume (Figure 11). While the volumes modeled using ST-SAD are square and have grid structures, our method models round shapes and organic structures inside the resulting volumes. These input videos include not only emissive but also absorptive media. Since our method assumes fluids to be emissive media, absorptive media are treated incorrectly: specifically, dark smoke is usually ignored and holes are created there. Nevertheless, the modeled volumes can be used to create interesting fluid animations as shown in the supplementary video.

Edit of fluid animations We demonstrate the incorporation of a fluid simulator to add more detailed turbulence to our fluid animations. This is achieved by increasing the parameters for turbulence in Autodesk Maya 2015. Figures 12-a and 12-b show the results of an explosion without and with turbulence. We may use the velocity field based on the approximate 3D optical flow to describe the interaction with the objects around the fluids. Figure 12-c shows an example of the explosion that destroys a car. The car was passively advected by the velocity field, and we applied a rigid body simulation to describe the interaction between the car and ground.

Existing methods with the fluid simulator Even if we applied the fluid simulator to volume sequences modeled with LSM or ST-SAD, we could not create nice fluid animations (i.e., we still found significant grid artifacts in them; see Figure 13). Figure 14 shows the comparison, where we added relatively strong turbulence to the fluid animations: for all the animations, we used the same set of turbulence parameters in Autodesk Maya 2015. Procedural details like turbulence tend to relax grid artifacts; however, even when strong turbulence is added, grid artifacts are still visible as many straight lines in the animations with the existing methods.

Convergence As described in Section 3, it is difficult to formally guarantee the convergence of our algorithm. However, we successfully get close to convergence to a good solution for more than 1,600 examples that we have tested. Figure 15 shows plots of the energy (Eq. 1) during modeling of 2-view synthetic fire (Figure 9) and 6-view synthetic fire (Figure 10).

Figure 12: *Incorporation with a fluid simulator.*

Figure 13: *The fluid simulator and the technique of Section 4 is applied to volume sequences of LSM, ST-SAD, and our method.*

Comparison with [Hasinoff and Kutulakos 2007] Hasinoff et al. proposed two algorithms for modeling volumes using front- and side-view images: the flame sheet algorithm and the multiplication algorithm. The flame sheet algorithm is useful to model thin surfaces; however, it cannot model a volumetric structure. The volume modeled using the multiplication algorithm appears similar to a volume modeled using LSM (i.e., the volume appears blurry with novel viewing angles, and grid artifacts are visible viewed from the top). We show a comparison in the supplementary video.

Figure 14: *Turbulence is added to fluid animations created using LSM, ST-SAD, and our method.*

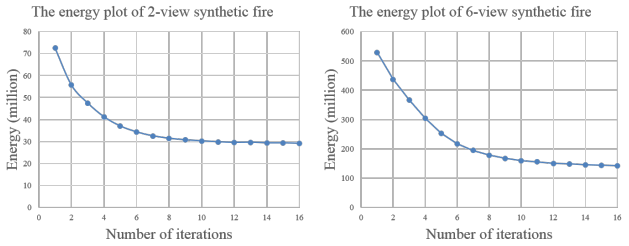


Figure 15: *Plots of the energy during modeling the volumes of synthetic fire as a function of number of iterations.*

Non-parametric texture synthesis for appearance transfer We have performed the parametric texture synthesis (PTS) [Heeger and Bergen 1995] for appearance transfer (Section 3), but it is still not clear what will happen when we perform non-parametric texture synthesis (NPTS). We tested *texture optimization* (TO) [Kwatra et al. 2005]. The original TO works similarly to PTS: the method takes a random dot image and a texture exemplar, and converts the random dot image into an image with an appearance that is similar to the texture exemplar. In our case, in the M-step of our iterative algorithm, TO takes a rendered image $\mathbf{o}'_\alpha$ and all the input images I as texture exemplars, and then converts $\mathbf{o}'_\alpha$ into the modified image $\mathbf{o}_\alpha$. Mathematically, replacing PTS with TO affects the definition of T_α in Eq. 2: we now consider $T_\alpha = \{\mathbf{t} : \mathbf{t} \text{ minimizes } E_t\}$, where E_t is the texture energy described in [Kwatra et al. 2005]. Note that T_α in TO is the same for all α, because we do not perform any interpolation between views (PTS allows linearly interpolating histograms readily; however, we have not found a good way to perform interpolation for TO). We chose TO because it is one of the methods that synthesizes textures with the highest quality. However, volume modeling using TO was unsuccessful (Figure 16). Since the size of local patches used for the nearest neighbor search is an important parameter in TO, we tested multiple sizes of local patches; however, all the volumes modeled by TO (Figures 16-c, 16-d, and 16-e) were too dense and had grid artifacts of many straight lines and arcs inside; the appearance at novel views is much less similar to the inputs compared with the PTS result (Figure 16-b). Furthermore, the energy of Eq. 1 did not decrease smoothly like the plots in Figure 15, but it often increased and oscillated. We believe that this failed for the following reason. In PTS case, T_α is a very large set of images, while T_α is much smaller in the TO case, because a local patch in $\mathbf{o}_\alpha$ has to appear similar to a patch in the exemplars, which results in tighter constraints on T_α than the appearance similarity described by histograms in PTS. As a result, the projection of $\mathbf{o}'_\alpha$ onto T_α, performed by TO, often modifies $\mathbf{o}'_\alpha$ too dramatically, which is far from the orthogonal projection.

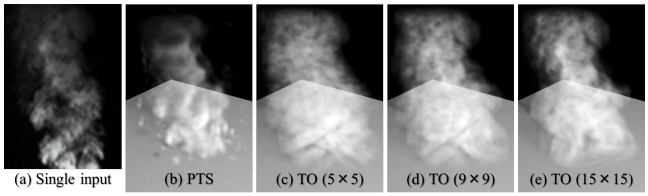


Figure 16: *We applied our method (Section 3) to the single-view input (a), whose size is* 141×100 *pixels, and modeled the volume (b). We replaced PTS with TO in our iterative algorithm and modeled volumes (c-e) with various sizes of local patches. Volumes modeled using TO were noisy and not convincing. Through all these experiments, the number of views for appearance transfer is 180, and we iterated the algorithm eight times.*

Limitations in preserving the global structure We attempted to model a volumetric tree by preparing front- and side-view images of a real tree, and applied our method to them. A tomographic approach was used for modeling the volumetric tree from multi-view images [Reche-Martinez et al. 2004]. However, we failed to model the branching structure of the tree in the resulting volume (Figure 17). Even when changing the viewpoint from the front of the tree only slightly, the portion of the trunk inside the red circle disappeared. This is because the texture synthesis method [Heeger and Bergen 1995] was not able to preserve the global structure, such as that along the branches. The trunk was scattered inside the volume. To solve this limitation, we plan to investigate other parametric texture synthesis methods that can better preserve the global structure [Portilla and Simoncelli 2000].

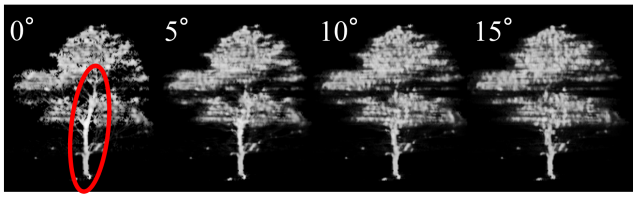


Figure 17: *The volumetric tree modeled by our method from front- and side-view images. The trunk inside the red circle immediately disappears when changing the viewpoint slightly.*

Viewing angles for appearance transfer So far we have only discussed side views of the volume (i.e., the viewing angle was rotated horizontally around the vertical axis). The reason for this is that most of the fluids we have investigated flow from bottom to top, and the side views appear similar; however, the top differs significantly. Where the top view also appears similar to the side views, it is possible to extend our method to a larger range of viewing angles.

The number of views for appearance transfer We have performed appearance transfer with 180 viewing angles (Section 3); however, this number should be changed according to the size of the target volume. For example, more viewing angles should be considered to model larger volumes. We tried to use the smaller number, e.g., 30, of viewing angles, but the quality of the resulting volume was no good: streak artifacts appeared in the volume after applying LSM in the E-step, which were further emphasized at subsequent iterations.

6 Conclusion and future work

We have proposed a method to create fluid animations efficiently using sparse multi-view videos as a guide. Our method described in Section 3 models the volume sequence from the videos. We proposed an energy function for volume modeling with appearance preservation. Our iterative algorithm successfully modeled more than 1,600 fluid volumes. The fluid animation of the modeled volume sequence can be edited by manipulating shading and physics parameters using the built-in fluid simulator in Autodesk Maya 2015. We have demonstrated that our method allows the user to create production-ready fluid animations from single-view videos.

An investigation on applying NPTS methods to our framework should be considered next. We expect that if we apply it successfully, it might solve the limitations in preserving the global structure, because NPTS methods are generally good at preserving global structures such as branches (Figure 17) in the texture and image synthesis contexts.

In this paper, we performed all the ray-casting operations using orthographic projections, because most of the input images tested were single-view images or we could assume an orthographic projection for the other examples. However, perspective projection and camera calibration should be considered in a more general situation. Extending our method to perspective projection is a future project.

Acknowledgements

We would like to thank the anonymous reviewers for their insightful and constructive comments. Many thanks also go to Hiroyuki Ochiai, Keisuke Mizutani, Takatsugu Yamaguchi, and Ayumi Kimura for discussions and encouragements. This work was supported by Japan Science and Technology Agency, CREST, and JSPS Grant-in-Aid for Young Scientists (B) Grant Number 25730071. This work was partially supported by the Joint Research Program (Short-term Collaborative Research) of the Institute of Mathematics for Industry, Kyushu University. Yoshinori Dobashi was partially supported by UEI Research.

References

ATCHESON, B., IHRKE, I., HEIDRICH, W., TEVS, A., BRADLEY, D., MAGNOR, M., AND SEIDEL, H.-P. 2008. Time-resolved 3d capture of non-stationary gas flows. *ACM Trans. Graph. 27*, 5, 132:1–132:9.

BHAT, K. S., SEITZ, S. M., HODGINS, J. K., AND KHOSLA, P. K. 2004. Flow-based video synthesis and editing. *ACM Trans. Graph. 23*, 3, 360–363.

BRIDSON, R., AND MÜLLER-FISCHER, M. 2007. Fluid simulation: Siggraph 2007 course notes. In *ACM SIGGRAPH 2007 Courses*, 1–81.

DEBEVEC, P. 2001. Light probe image gallery. Available at http://www.pauldebevec.com/Probes/.

DEBEVEC, P. 2006. High-resolution light probe image gallery. Available at http://gl.ict.usc.edu/Data/HighResProbes/.

DONG, Y., LEFEBVRE, S., TONG, X., AND DRETTAKIS, G. 2008. Lazy solid texture synthesis. In *Proc. of EGSR '08*, 1165–1174.

GREGSON, J., KRIMERMAN, M., HULLIN, M. B., AND HEIDRICH, W. 2012. Stochastic tomography and its applications in 3d imaging of mixing fluids. *ACM Trans. Graph. 31*, 4, 52:1–52:10.

GREGSON, J., HEIDE, F., HULLIN, M. B., ROUF, M., AND HEIDRICH, W. 2013. Stochastic deconvolution. In *Proc. of CVPR 2013*, 1043–1050.

GREGSON, J., IHRKE, I., THUEREY, N., AND HEIDRICH, W. 2014. From capture to simulation: Connecting forward and inverse problems in fluids. *ACM Trans. Graph. 33*, 4, 139:1–139:11.

GU, J., NAYAR, S., GRINSPUN, E., BELHUMEUR, P., AND RAMAMOORTHI, R. 2013. Compressive structured light for recovering inhomogeneous participating media. *IEEE Trans. Pattern Anal. Mach. Intell. 35*, 3, 555–566.

HASINOFF, S. W., AND KUTULAKOS, K. N. 2007. Photo-consistent reconstruction of semitransparent scenes by density-sheet decomposition. *IEEE Trans. Pattern Anal. Mach. Intell. 29*, 5, 870–885.

HAWKINS, T., EINARSSON, P., AND DEBEVEC, P. 2005. Acquisition of time-varying participating media. *ACM Trans. Graph. 24*, 3, 812–815.

HEEGER, D. J., AND BERGEN, J. R. 1995. Pyramid-based texture analysis/synthesis. In *Proc. of SIGGRAPH '95*, 229–238.

IHRKE, I., AND MAGNOR, M. 2004. Image-based tomographic reconstruction of flames. In *Proc. of SCA '04*, 365–373.

JAGNOW, R., DORSEY, J., AND RUSHMEIER, H. 2004. Stereological techniques for solid textures. *ACM Trans. Graph. 23*, 3, 329–335.

KAK, A. C., AND SLANEY, M. 1988. Principles of computerized tomographic imaging. IEEE Press.

KLEHM, O., IHRKE, I., SEIDEL, H.-P., AND EISEMANN, E. 2013. Volume stylizer: Tomography-based volume painting. In *Proc of I3D '13*, 161–168.

KOPF, J., FU, C.-W., COHEN-OR, D., DEUSSEN, O., LISCHINSKI, D., AND WONG, T.-T. 2007. Solid texture synthesis from 2d exemplars. *ACM Trans. Graph. 26*, 3, 2:1–2:9.

KWATRA, V., ESSA, I., BOBICK, A., AND KWATRA, N. 2005. Texture optimization for example-based synthesis. *ACM Trans. Graph. 24*, 3, 795–802.

LI, C., PICKUP, D., SAUNDERS, T., COSKER, D., MARSHALL, D., HALL, P., AND WILLIS, P. 2013. Water surface modeling from a single viewpoint video. *IEEE Trans. Vis. Comput. Graphics 19*, 7, 1242–1251.

LIU, Z., HU, Y., AND QI, Y. 2011. Modeling of smoke from a single view. In *Proc. of ICVRV '11*, 291–294.

MA, C., WEI, L.-Y., LEFEBVRE, S., AND TONG, X. 2013. Dynamic element textures. *ACM Trans. Graph. 32*, 4, 90:1–90:10.

PORTILLA, J., AND SIMONCELLI, E. P. 2000. A parametric texture model based on joint statistics of complex wavelet coefficients. *Int. J. Comput. Vis. 40*, 1, 49–70.

QIN, X., AND YANG, Y.-H. 2007. Aura 3d textures. *IEEE Trans. Vis. Comput. Graph. 13*, 2, 379–389.

RECHE-MARTINEZ, A., MARTIN, I., AND DRETTAKIS, G. 2004. Volumetric reconstruction and interactive rendering of trees from photographs. *ACM Trans. Graph. 23*, 3, 720–727.

TAKAYAMA, K., OKABE, M., IJIRI, T., AND IGARASHI, T. 2008. Lapped solid textures: Filling a model with anisotropic textures. *ACM Trans. Graph. 27*, 3, 53:1–53:9.

THÜREY, N., KEISER, R., PAULY, M., AND RÜDE, U. 2006. Detail-preserving fluid control. In *Proc. of SCA '06*, 7–12.

WEI, L.-Y. 2002. *Texture Synthesis by Fixed Neighborhood Searching*. PhD thesis.

WENGER, S., LORENZ, D., AND MAGNOR, M. 2013. Fast image-based modeling of astronomical nebulae. *Comput. Graph. Forum 32*, 7, 93–100.

WERLBERGER, M., POCK, T., AND BISCHOF, H. 2010. Motion estimation with non-local total variation regularization. In *Proc. of CVPR 2010*, 2464–2471.

WEXLER, Y., SHECHTMAN, E., AND IRANI, M. 2007. Space-time completion of video. *IEEE Trans. Pattern Anal. Mach. Intell. 29*, 3, 463–476.

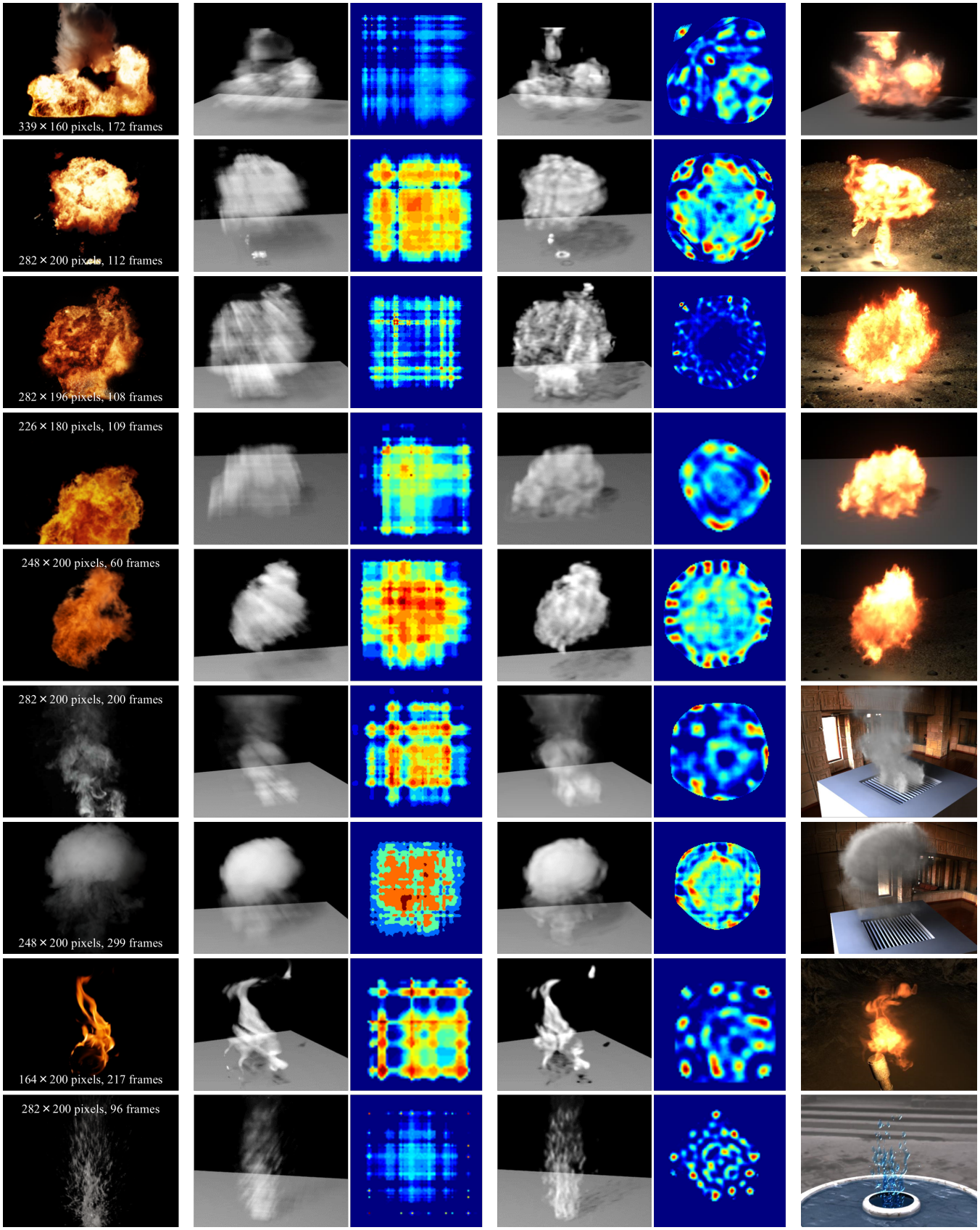


Figure 11: *Volumes modeled from single-view videos. From left to right: a frame of the input video, the volume modeled using ST-SAD and its horizontal slice, the volume modeled using our method (described in Section 3 without using any fluid simulator) and its horizontal slice, and the final rendering using Autodesk Maya 2015. The jet colormap is used to visualize the slices.*