Stroke Transfer for Participating Media: Supplementary Material

NAOTO SHIRASHIMA*, Aoyama Gakuin University (AGU), Japan
HIDEKI TODO*, Takushoku University, Japan
YUKI YAMAOKA, Aoyama Gakuin University (AGU), Japan
SHIZUO KAJI, Kyushu University, Japan, Kyoto University, Japan
KUNIHIKO KOBAYASHI, Aoyama Gakuin University (AGU), Japan
HARUNA SHIMOTAHIRA, Aoyama Gakuin University (AGU), Japan
YONGHAO YUE, Aoyama Gakuin University (AGU), Japan

CCS Concepts: • **Computing methodologies** → **Non-photorealistic rendering**.

Additional Key Words and Phrases: Non-photorealistic rendering, stroke-based rendering, example-based, stroke transfer, vector field generation, participating media, volumetric normals and curvatures, automatic exemplar selection

ACM Reference Format:
Naoto Shirashima, Hideki Todo, Yuki Yamaoka, Shizuo Kaji, Kunihiko Kobayashi, Haruna Shimotahira, and Yonghao Yue. 2025. Stroke Transfer for Participating Media: Supplementary Material. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '25), August 10–14, 2025, Vancouver, BC, Canada.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3721238.3730603

1 Motivation for $\mathbf{M}_{\mathrm{v}}$

Figure 1 illustrates that the orientations of artist-drawn strokes do not always align with predetermined basis directions, such as principal curvature directions or luminance gradients. Instead, we observe an *unconscious* combination of basis directions.

For example, beyond strokes following illumination cues, we see strokes annotated in green, orange, and red in the right panel of Figure 1. Compared to the principal curvature directions shown in green in the left panel, the orange directions deviate from the curvature directions, instead appearing as an intermediate blend between the red silhouette directions and the curvature directions in the interior.

The role of $\mathbf{M}_{\mathrm{v}}$ is to convert features into coefficients for basis fields, effectively capturing and learning this unconscious relationship.

*Joint first authors — authors contributed equally.

Authors' Contact Information: Naoto Shirashima, Aoyama Gakuin University (AGU), Kanagawa, Japan, koutei7penguin.jp@gmail.com; Hideki Todo, Takushoku University, Tokyo, Japan, htodo@cs.takushoku-u.ac.jp; Yuki Yamaoka, Aoyama Gakuin University (AGU), Kanagawa, Japan, ymok124@gmail.com; Shizuo Kaji, Kyushu University, Fukuoka, Japan, and Kyoto University, Kyoto, Japan, skaji@imi.kyushu-u.ac.jp; Kunihiko Kobayashi, Aoyama Gakuin University (AGU), Kanagawa, Japan, kuni.koba.onepiece@outlook.com; Haruna Shimotahira, Aoyama Gakuin University (AGU), Kanagawa, Japan, sanso.sugu@gmail.com; Yonghao Yue, Aoyama Gakuin University (AGU), Kanagawa, Japan, yonghao@it.aoyama.ac.jp.

SIGGRAPH Conference Papers '25, Vancouver, BC, Canada

ACM ISBN 979-8-4007-1540-2/2025/08
https://doi.org/10.1145/3721238.3730603

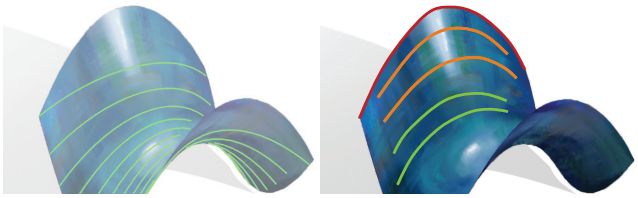

Fig. 1. **Stroke orientations in a saddle shape example**.

2 Curvature for Isosurfaces of a Scalar Field

Let us use subscripts after a comma to indicate partial derivatives (e.g., $\psi_{,x} := \frac{\partial \psi}{\partial x}$ and $\psi_{,xy} := \frac{\partial^2 \psi}{\partial x \partial y}$). Then, the gradient of a scalar field ψ is given by $\nabla\psi = (\psi_{,x}, \psi_{,y}, \psi_{,z})^\top$, and its Hessian $\mathbf{H}(\psi)$ and the adjugate of the Hessian $\mathbf{H}^*(\psi)$ are given by

$$\mathbf{H}(\psi) = \begin{pmatrix} \psi_{,xx} & \psi_{,xy} & \psi_{,xz} \\ \psi_{,yx} & \psi_{,yy} & \psi_{,yz} \\ \psi_{,zx} & \psi_{,zy} & \psi_{,zz} \end{pmatrix}, \quad \mathbf{H}^*(\psi) = \begin{pmatrix} \psi_{,yy}\psi_{,zz} - \psi_{,yz}\psi_{,zy} & \psi_{,yz}\psi_{,zx} - \psi_{,yx}\psi_{,zz} & \psi_{,yx}\psi_{,zy} - \psi_{,yy}\psi_{,zx} \\ \psi_{,xz}\psi_{,zy} - \psi_{,xy}\psi_{,zz} & \psi_{,xx}\psi_{,zz} - \psi_{,xz}\psi_{,zx} & \psi_{,xy}\psi_{,zx} - \psi_{,xx}\psi_{,zy} \\ \psi_{,xy}\psi_{,yz} - \psi_{,xz}\psi_{,yy} & \psi_{,yx}\psi_{,xz} - \psi_{,xx}\psi_{,yz} & \psi_{,xx}\psi_{,yy} - \psi_{,xy}\psi_{,yx} \end{pmatrix}, \tag{1}$$

respectively. Then, the formulae by Goldman [2005] gives the Gaussian curvature κ_G and mean curvature κ_m as

$$\kappa_G = \frac{(\nabla\psi)^\top \mathbf{H}^*(\psi) \nabla\psi}{|\nabla\psi|^4}, \quad \kappa_m = \frac{(\nabla\psi)^\top \mathbf{H}(\psi) \nabla\psi - |\nabla\psi|^2 \operatorname{Tr} \mathbf{H}(\psi)}{2|\nabla\psi|^3}. \tag{2}$$

In practice, the computed object-space curvatures κ_G and κ_m can be noisy in very thin regions where $\nabla\psi$ approaches zero. To address this, we apply a Gaussian filter to smooth ψ slightly before computing the curvatures. This smoothing causes non-constant regions of ψ to spread toward the vacuum region, effectively moving zero-gradient regions closer to the vacuum. Since the free-path distribution is zero in the vacuum, these regions contribute less during the integration along the line of sight, improving numerical stability.

3 Computing Distance from Silhouette for Medium

We discretize the following integral discussed in the main paper,

$$\xi_M(\boldsymbol{u}) = \int_0^1 \xi_M(\boldsymbol{u}, \eta) p_{\mathrm{dist}}(\eta) d\eta, \tag{3}$$

in an adaptive way. For $p_{\mathrm{dist}}(\eta)$, we use

$$p_{\mathrm{dist}}(\eta) \propto \exp\left(-\frac{(\eta - \eta_t)^2}{2\sigma^2}\right), \quad \text{for } 0 \le \eta \le 1, \tag{4}$$

with proper normalization, where we set $\eta_t = 0.9$, and $\sigma = 0.2$.

We approximate (3) in the following form:

$$\xi_M(\boldsymbol{u}) = \sum_{k=0}^{N_{\text{Int}}} \xi_M\left(\boldsymbol{u}, \frac{\eta_k + \eta_{k+1}}{2}\right)(F(\eta_{k+1}) - F(\eta_k)), \tag{5}$$

where $\eta_0 = 0$ and $\eta_{N_{\text{Int}}} = 1$, and η_k's are not evenly distributed. $F(\eta)$ is the cumulative distribution function of $p_{\text{dist}}(\eta)$. We obtain this form by performing Taylor expansion for $\xi_M(\boldsymbol{u}, \eta)$ up to first order, and use the first order term as an error metric to determine η_k's.

Suppose that η_k's are given. Then, we can estimate the error when we only use the zero-th order term for $\xi_M(\boldsymbol{u}, \eta)$ as follows. First, we have

$$\xi_M(\boldsymbol{u}) = \sum_{k=0}^{N_{\text{Int}}} \int_{\eta_k}^{\eta_{k+1}} \xi_M(\boldsymbol{u}, \eta) p_{\text{dist}}(\eta). \tag{6}$$

Expanding $\xi_M(\boldsymbol{u}, \eta)$ up to first order, we have

$$\xi_M(\boldsymbol{u}, \eta) \approx \xi_M(\boldsymbol{u}, \bar{\eta}) + (\eta - \bar{\eta}) \frac{\partial \xi_M(\boldsymbol{u}, \bar{\eta})}{\partial \eta}. \tag{7}$$

Let

$$\bar{\eta}_k := \frac{\eta_k + \eta_{k+1}}{2}. \tag{8}$$

Then, we have an first order approximation $\tilde{\xi}_M(\boldsymbol{u})$ (each term expanded at the middle point of the corresponding interval) for (6) as

$$\tilde{\xi}_M(\boldsymbol{u}) := \sum_{k=0}^{N_{\text{Int}}} \int_{\eta_k}^{\eta_{k+1}} \left\{ \xi_M(\boldsymbol{u}, \bar{\eta}_k) + (\eta - \bar{\eta}_k) \frac{\partial \xi_M(\boldsymbol{u}, \bar{\eta}_k)}{\partial \eta} \right\} p_{\text{dist}}(\eta) d\eta. \tag{9}$$

To compute the integration for the second term in the curly braces in (9) (and noting that $\frac{\partial \xi_M(\boldsymbol{u}, \bar{\eta}_k)}{\partial \eta}$ is a constant), we define $Q(a, b)$ as

$$Q(a, b) := \int_a^b \left(\eta - \frac{a+b}{2}\right) p_{\text{dist}}(\eta) d\eta. \tag{10}$$

Let the integral of F be G. With integral by parts, we have

$$\begin{aligned} Q(a, b) &= [\eta F(\eta)]_a^b - \int_a^b F(\eta) d\eta - \frac{a+b}{2} \int_a^b p_{\text{dist}}(\eta) d\eta \\ &= \frac{b-a}{2}(F(b) + F(a)) - (G(b) - G(a)). \end{aligned} \tag{11}$$

Plugging in (9), we have

$$\begin{aligned} \tilde{\xi}_M(\boldsymbol{u}) = &\sum_{k=0}^{N_{\text{Int}}} \xi_M(\boldsymbol{u}, \bar{\eta}_k)(F(\eta_{k+1}) - F(\eta_k)) \\ &+ \sum_{k=0}^{N_{\text{Int}}} \frac{\partial \xi_M(\boldsymbol{u}, \bar{\eta}_k)}{\partial \eta} Q(\eta_k, \eta_{k+1}). \end{aligned} \tag{12}$$

If we require the absolute value of each term in the second summation to be smaller than a threshold ε, we must have

$$\left| \frac{\partial \xi_M(\boldsymbol{u}, \bar{\eta}_k)}{\partial \eta} Q(\eta_k, \eta_{k+1}) \right| \leq \varepsilon. \tag{13}$$

Since $\xi_M(\boldsymbol{u}, \bar{\eta}_k)$ is a signed distance function, its partial derivative $\frac{\partial \xi_M(\boldsymbol{u}, \bar{\eta}_k)}{\partial \eta}$ is essentially bounded by how fast the inverse of the spatial derivative of transmittance $T_s^{(M)}(\boldsymbol{u})$ changes near the iso-contour $T_s^{(M)}(\boldsymbol{u}) = \eta$, or more specifically, let δ_k be

$$\delta_k := \max \left\{ \frac{1}{\left| \nabla_{\boldsymbol{u}} T_s^{(M)}(\boldsymbol{u}) \right|} \middle| \eta_k \leq T_s^{(M)}(\boldsymbol{u}) \leq \eta_{k+1} \right\}, \tag{14}$$

then

$$\left| \frac{\partial \xi_M(\boldsymbol{u}, \bar{\eta}_k)}{\partial \eta} \right| \leq \delta_k. \tag{15}$$

So,

$$|Q(\eta_k, \eta_{k+1})| \leq \frac{\varepsilon}{\delta_k}. \tag{16}$$

Starting from $k = 0$ with $\eta_0 = 0$, we find the largest η_{k+1} satisfying (16).

4 Standardization of Features

We apply standardization as in Todo et al. [2022] for features to equalize their scales. The standardization is done for all the frames simultaneously (to keep the procedure consistent over the frames).

4.1 $L^\star$, $a^\star$, $b^\star$

We first convert the raw screen intensity $\boldsymbol{I}_s(\boldsymbol{u})$ into the corresponding $L^\star(\boldsymbol{u})$, $a^\star(\boldsymbol{u})$, and $b^\star(\boldsymbol{u})$ components. Then, For the luminance $L^\star$, we apply a tone mapping $\mathcal{T}$ to obtain the tone mapped luminance $\tilde{I}_s$, which is treated as the intensity feature, using the same function as Todo et al. [2022] given below:

$$\tilde{I}_s(L^\star) = \mathcal{T}(L^\star; L_{\max}, \Theta_{\mathcal{T}}) := L_{\max}\left(2\frac{\exp(\Theta_{\mathcal{T}} L^\star)}{\exp(\Theta_{\mathcal{T}} L^\star) + 1} - 1\right), \tag{17}$$

where $L_{\max}$ controls the upper limit of the tone mapped luminance, and $\Theta_{\mathcal{T}}$ controls the curve. For standardization of the luminance, we rescale and shift $\tilde{I}_s$ from $[0, L_{\max}]$ to $[-1, 1]$.

For the $a^\star$ and $b^\star$ components, we first compute the maximum value $c^{\max} = \max(|a^\star|, |b^\star|)$, and then rescale $[-c^{\max}, c^{\max}]$ to $[-1, 1]$.

4.2 Apparent intensity gradient

For the apparent intensity gradient, we first compute its mean $\mu_{\tilde{I}_s^{\nabla_2}}$ and standard deviation $\Sigma_{\tilde{I}_s^{\nabla_2}}$. Then, we rescale and shift $[0, \mu_{\tilde{I}_s^{\nabla_2}} + 2\Sigma_{\tilde{I}_s^{\nabla_2}}]$ to $[-1, 1]$.

4.3 Apparent curvatures

We first compute the (top) 2 percentile values $\kappa_{Gs}^{(2\%)}$, $\kappa_{ms}^{(2\%)}$ of the absolute values of apparent Gaussian curvatures $|\kappa_{Gs}|$ and apparent mean curvatures $|\kappa_{ms}|$ over frames, respectively. Then, we compute their maximum $\kappa^{\max}$ as $\kappa^{\max} = \max(\kappa_{Gs}^{(2\%)}, (\kappa_{ms}^{(2\%)})^2)$. Note that the absolute maximum of the mean curvature is squared to align the unit with that of the Gaussian curvature. For the apparent Gaussian curvature, we rescale $[-\kappa^{\max}, \kappa^{\max}]$ to $[-1, 1]$, and for the apparent mean curvature, we rescale $[-\sqrt{\kappa^{\max}}, \sqrt{\kappa^{\max}}]$ to $[-1, 1]$. Note that larger values (outside of $[-1, 1]$) are not clamped.

4.4 Apparent normal

Since the apparent normals lie in the range $[-1, 1]$ by construction, we apply no standardization for them.

4.5 Temperature

We first compute the (top) 2 percentile value $C_s^{(2\%)}$ of the temperatures C_s over frames. Then, we rescale and shift $[0, C_s^{(2\%)}]$ to $[-1, 1]$. No clamp is applied for values exceeding this range.

4.6 Apparent relative velocity

We first compute the (top) 2 percentile values $v_s^{(x,2\%)}$ and $v_s^{(y,2\%)}$ of the absolute x- and y- components of the apparent relative velocities $|v_s^{(x)}|$ and $|v_s^{(y)}|$ over frames. We then compute the maximum value as $v^{\max} = \max(v_s^{(x,2\%)}, v_s^{(y,2\%)})$, and rescale $[-v^{\max}, v^{\max}]$ to $[-1, 1]$. No clamp is applied for values exceeding this range.

4.7 Transmittance

We rescale and shift $[0, 1]$ (the value range of transmittance) to $[-1, 1]$.

4.8 Apparent mean free-path

We compute the top and bottom 1 percentile values $d_s^{(\text{top},1\%)}$ and $d_s^{(\text{bottom},1\%)}$ over the frames. We then rescale $[d_s^{(\text{bottom},1\%)}, d_s^{(\text{top},1\%)}]$ to $[-1, 1]$. No clamp is applied for values exceeding this range.

4.9 Distance from silhouettes

We compute the maximum value $\xi^{\max}$ for the distance from silhouettes over frames (no absolute operator is applied). We then rescale $[-\xi^{\max}, \xi^{\max}]$ to $[-1, 1]$. Note that no clamp is applied for $\xi_s < -\xi^{\max}$.

5 Basis Fields

For the intensity gradient $\boldsymbol{I}^{(\parallel)}(\boldsymbol{u})$ and its 90° rotation $\boldsymbol{I}^{(\perp)}(\boldsymbol{u})$, we have

$$\boldsymbol{I}^{(\parallel)}(\boldsymbol{u}) := \mathfrak{N}(\nabla_2 \tilde{I}(\boldsymbol{u})), \tag{18}$$

and

$$\boldsymbol{I}^{(\perp)}(\boldsymbol{u}) := \mathrm{Rot}_{\pi/2}(\mathfrak{N}(\nabla_2 \tilde{I}(\boldsymbol{u}))), \tag{19}$$

where $\mathfrak{N}(\cdot)$ normalizes the vector in the screen space, and $\mathrm{Rot}_{\pi/2}$ is the 90° rotation in the screen space.

For silhouette guided direction $\boldsymbol{o}^{(\parallel)}(\boldsymbol{u})$ and its 90° rotation $\boldsymbol{o}^{(\perp)}(\boldsymbol{u})$, we have

$$\boldsymbol{o}^{(\parallel)}(\boldsymbol{u}) := \mathfrak{N}(\nabla_2 \xi_s(\boldsymbol{u})), \tag{20}$$

and

$$\boldsymbol{o}^{(\perp)}(\boldsymbol{u}) := \mathrm{Rot}_{\pi/2}(\mathfrak{N}(\nabla_2 \xi_s(\boldsymbol{u}))). \tag{21}$$

For apparent normal $\boldsymbol{n}^{(\parallel)}(\boldsymbol{u})$ and its 90° rotation $\boldsymbol{n}^{(\perp)}(\boldsymbol{u})$, we have

$$\boldsymbol{n}^{(\parallel)}(\boldsymbol{u}) := \mathfrak{N}(\boldsymbol{n}_s^{(x,y)}(\boldsymbol{u})), \tag{22}$$

and

$$\boldsymbol{n}^{(\perp)}(\boldsymbol{u}) := \mathrm{Rot}_{\pi/2}(\mathfrak{N}(\boldsymbol{n}_s^{(x,y)}(\boldsymbol{u}))). \tag{23}$$

For gradient of apparent mean free-path $\boldsymbol{m}^{(\parallel)}(\boldsymbol{u})$ and its 90° rotation $\boldsymbol{m}^{(\perp)}(\boldsymbol{u})$, we have

$$\boldsymbol{m}^{(\parallel)}(\boldsymbol{u}) := \mathfrak{N}(\nabla_2 d_s(\boldsymbol{u})) \tag{24}$$

and

$$\boldsymbol{m}^{(\perp)}(\boldsymbol{u}) := \mathrm{Rot}_{\pi/2}(\mathfrak{N}(\nabla_2 d_s(\boldsymbol{u}))). \tag{25}$$

Finally, for relative velocity $\boldsymbol{v}^{(\parallel)}(\boldsymbol{u})$ and its 90° rotation $\boldsymbol{v}^{(\perp)}(\boldsymbol{u})$, we have

$$\boldsymbol{v}^{(\parallel)}(\boldsymbol{u}) := \mathfrak{N}(\boldsymbol{v}_s(\boldsymbol{u})), \tag{26}$$

and

$$\boldsymbol{v}^{(\perp)}(\boldsymbol{u}) := \mathrm{Rot}_{\pi/2}(\mathfrak{N}(\boldsymbol{v}_s(\boldsymbol{u}))). \tag{27}$$

6 Orientation Smoothing

Let $\boldsymbol{d}(t, \boldsymbol{u})$ denote the orientation field obtained from the learned model for the target scene. The smoothed field, $\bar{\boldsymbol{d}}(t, \boldsymbol{u})$, is computed as:

$$\bar{\boldsymbol{d}} = \underset{\boldsymbol{d}'}{\mathrm{argmin}} \left(\int_T \int_\Omega \|\boldsymbol{d}' - \boldsymbol{d}\|^2 dA dt + \lambda_\mathrm{s} \int_T \int_\Omega \|\nabla_2 \boldsymbol{d}'\|^2 dA dt + \lambda_\mathrm{t} \int_T \int_\Omega \left(\frac{\partial \boldsymbol{d}'}{\partial t} \right)^2 dA dt \right), \tag{28}$$

where $(t, \boldsymbol{u})$ is omitted for brevity, λ_s and λ_t are spatial and temporal smoothing coefficients, and T and Ω denote the temporal domain (start to end of the animation) and spatial domain (screen region), respectively. ∇_2 denotes the spatial gradient in screen space.

The corresponding discrete formulation is:

$$\boldsymbol{V} = \underset{\boldsymbol{V}'}{\mathrm{argmin}}\, E(\boldsymbol{V}'), \tag{29}$$

where $\boldsymbol{V}$ encodes the spatially and temporally varying field $\boldsymbol{v}$ into a single long vector. The energy functional $E(\boldsymbol{V}')$ is expressed as:

$$E(\boldsymbol{V}') = (\boldsymbol{V} - \bar{\boldsymbol{V}})^\top \hat{\mathbf{I}} (\boldsymbol{V} - \bar{\boldsymbol{V}}) + \boldsymbol{V}^\top \mathbf{D}^\top \hat{\mathbf{\Lambda}} \mathbf{D} \boldsymbol{V}, \tag{30}$$

where $\hat{\mathbf{I}} = \mathrm{diag}(\Delta A \Delta t, ..., \Delta A \Delta t)$, $\Delta A = \Delta x \Delta y$, $\hat{\mathbf{\Lambda}}$ is a diagonal matrix consisting terms of $\lambda_\mathrm{s} \Delta A \Delta t$ and $\lambda_\mathrm{t} \Delta A \Delta t$, and $\mathbf{D}$ is the discrete differential operator consisting terms of $\frac{1}{\Delta x}$, $\frac{-1}{\Delta x}$, $\frac{1}{\Delta y}$, $\frac{-1}{\Delta y}$, $\frac{1}{\Delta t}$, $\frac{-1}{\Delta t}$, and 0. By differentiating the energy functional $E(\boldsymbol{V}')$ with respect to $\boldsymbol{V}'$ and setting the derivative to zero, the smoothed field $\boldsymbol{V}$ is obtained by solving the following sparse linear system:

$$(\hat{\mathbf{I}} + \mathbf{D}^\top \hat{\mathbf{\Lambda}} \mathbf{D}) \boldsymbol{V} = \hat{\mathbf{I}} \bar{\boldsymbol{V}}. \tag{31}$$

Unlike the discretization approach on curved surfaces employed in stroke transfer for surfaces [Todo et al. 2022], our method operates on a regular grid aligned with the screen resolution. This makes the implementation straightforward and computationally efficient. Consequently, we can solve for smoothing across all frames simultaneously, further enhancing scalability and performance.

7 Stroke Rendering

Our goal in stroke rendering is to determine a set of anchor points and corresponding strokes for each frame of the animation. The inputs to this process include the attribute samplers $S^{(n)}$ (one per frame), relative velocity fields $\hat{\boldsymbol{v}}^{(n)}$, optional binary alpha masks $M^{(n)}$, two stroke textures T_E and T_N (original and shrinked), and discretization parameters such as timestep Δt (the interval between consecutive frames) and stroke step length Δl.

An attribute sampler $S^{(n)}$ is a queryable structure that returns stroke attributes—color, orientation, length, and width—at a given screen-space position $\boldsymbol{u}$. It can also be extended to return other properties, such as region labels, which are transferred from exemplars in the same way as color, via regression. The relative velocity field $\hat{\boldsymbol{v}}^{(n)}$ is defined as described in the main paper and is used to advect anchor points across frames. The alpha mask $M^{(n)}$, if provided, represents a coarse binary segmentation of foreground regions where strokes are to be drawn; it can be generated, for instance, by thresholding transmittance. We use two stroke textures: the original texture T_E, and a shrinked version T_N, analogous to the use of reduced radii in particle resampling (e.g., [Yue et al. 2015]). This design helps newly added strokes persist across frames, even when undergoing slight motion. By applying overlap checks using the smaller texture but rendering with the original one, we encourage spatial overlap between neighboring strokes, which can help maintain visual density and continuity in animated sequences.

Stroke generation proceeds as described in Algorithm 1, which loops over all frames. For each frame, the system propagates anchor points and strokes from the previous frame (Algorithm 2), fills uncovered regions with new strokes (Algorithm 3), sorts the strokes (Algorithm 4), and removes strokes that are completely occluded (Algorithm 5).

In the update step (Algorithm 2), the active set is initialized using the alpha mask. Each anchor from the previous frame is advected using the velocity field via a Runge–Kutta integration scheme (`TVDRK`, Algorithm 6). A new stroke is then generated using `Generate_Stroke` (Algorithm 8), and its pixels are marked as covered in the active set using the original texture T_E.

In the addition step (Algorithm 3), new anchor points are sampled from the remaining active regions. Each new anchor is assigned randomized parameters and used to generate a stroke. The region covered by the stroke is deactivated using the shrinked texture T_N to allow strokes to overlap in rendering while preventing unnecessary density.

Strokes are sorted (Algorithm 4) in creation order for a consistent overdrawing effect, though other ordering strategies such as luminance-based sorting are also possible.

Hidden strokes are removed (Algorithm 5) efficiently by rendering strokes in reverse order (from front to back) and maintaining a transparency buffer. If no pixels of a stroke remain visible (based on alpha thresholds and accumulated transparency), it is flagged for removal. This approach avoids quadratic time complexity and allows for linear-time occlusion testing.

Each stroke is composed of two halves traced from an anchor point. The function `Generate_Stroke` calls `Generate_Stroke_Half` (Algorithm 7) in both forward and backward directions. Tracing follows the screen-space orientation field given by the attribute sampler and terminates based on stroke length, region boundary, or degeneracy. Stroke length is determined dynamically as the average of sampled target lengths along the traced path, allowing early or extended termination depending on the local field. The centerline generated from tracing is then thickened based on the width sampled at the anchor point to construct the final stroke geometry.

We use a three-stage Runge–Kutta method, `TVDRK` (Algorithm 6), for accurate integration along the given vector field. While the pseudocode presents `TVDRK` as a single step, the actual implementation subdivides it into multiple substeps for improved numerical stability and smooth visual output. A rotation matrix is passed to `TVDRK` to account for angular offsets or direction flipping needed when tracing in both forward and reverse directions.

Algorithm 1 `Generate_Strokes`

Input: The set of frame indices $\mathcal{N}$, texture T_E for existing strokes, shrinked texture T_N for new strokes, attribute samplers $\{S^{(n)}\}$, relative velocities $\hat{\boldsymbol{v}}_s^{(n)}$, alpha masks $\{M^{(n)}\}$, timestep (interval between consecutive frames) Δt, step length Δl

$\mathcal{A}^- \leftarrow \emptyset$ ▷ Set of anchor points from previous frame
$\mathcal{S}^- \leftarrow \emptyset$ ▷ Set of strokes from previous frame
for $n \in \mathcal{N}$ **do**
  $\mathcal{A}', \mathcal{S}', A \leftarrow \texttt{Update}(\mathcal{A}^-, \mathcal{S}^-, M^{(n)}, T_E, S^{(n)}, \hat{\boldsymbol{v}}_s^{(n)}, \Delta t, \Delta l)$
  $\mathcal{A}^+, \mathcal{S}^+ \leftarrow \texttt{Add}(\mathcal{A}', \mathcal{S}', A, T_N, S^{(n)}, \Delta l)$
  $\mathcal{A}, \mathcal{S} \leftarrow \texttt{Sort}(\mathcal{A}', \mathcal{S}', \mathcal{A}^+, \mathcal{S}^+)$
  $\mathcal{A}, \mathcal{S} \leftarrow \texttt{Remove\_Hidden}(\mathcal{A}, \mathcal{S}, T_E, T_N)$
  $\texttt{Save\_Data}(\mathcal{A}, \mathcal{S}, n)$
  $\mathcal{A}^- \leftarrow \mathcal{A}$
  $\mathcal{S}^- \leftarrow \mathcal{S}$
end for

Algorithm 2 `Update`

Input: Anchors $\mathcal{A}^-$ from previous frame, strokes $\mathcal{S}^-$ from previous frame, mask $M^{(n)}$ for the current frame, stroke texture T, attribute sampler $S^{(n)}$ for the current frame, relative velocity $\hat{\boldsymbol{v}}_s^{(n)}$ for the current frame, timestep Δt, step length Δl

Output: Advected anchors $\mathcal{A}'$, advected strokes $\mathcal{S}'$, active set A

$A \leftarrow M^{(n)}$
$\mathcal{A}' \leftarrow \emptyset$
$\mathcal{S}' \leftarrow \emptyset$
for $a^- \in \mathcal{A}^-$ **do**
  $a' \leftarrow \texttt{TVDRK}(a^-, \hat{\boldsymbol{v}}_s^{(n)}, \Delta t, \mathbf{I})$
  $\mathcal{A}' \leftarrow \mathcal{A}' \cup \{a'\}$
  $s' \leftarrow \texttt{Generate\_Stroke}(a', S^{(n)}, \Delta l)$
  $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{s'\}$
  $A \leftarrow \texttt{Deactivate}(A, s', T)$
end for
return $\mathcal{A}', \mathcal{S}', A$

Algorithm 3 `Add`

Input: Already generated anchors $\mathcal{A}'$ and strokes $\mathcal{S}'$, active set A, Texture T, attribute sampler $S^{(n)}$, step length Δl
Output: Anchors $\mathcal{A}^+$ and strokes $\mathcal{S}^+$

$\mathcal{A}^+ \leftarrow \mathcal{A}'$
$\mathcal{S}^+ \leftarrow \mathcal{S}'$
while A has active location **do**
  $\boldsymbol{u} \leftarrow$ `Find_Random_Location_from_Active_Set`(A)
  $\Xi \leftarrow$ `Determine_Random_Numbers_for_Anchor`$()$
  $a \leftarrow \{\boldsymbol{u}, \Xi\}$
  $\mathcal{A}^+ \leftarrow \mathcal{A}^+ \cup \{a\}$
  $s \leftarrow$ `Generate_Stroke`$(a, S^{(n)}, \Delta l)$
  $\mathcal{S}^+ \leftarrow \mathcal{S}^+ \cup \{s\}$
  $A \leftarrow$ `Deactivate`(A, s, T)
end while
return $\mathcal{A}^+, \mathcal{S}^+$

Algorithm 4 `Sort`

Input: Anchors $\mathcal{A}'$ and strokes $\mathcal{S}'$ that existed from the previous frame (and have been advected during the current frame), newly added anchors $\mathcal{A}^+$ and strokes $\mathcal{S}^+$
Output: Sorted anchors $\mathcal{A}$ and strokes $\mathcal{S}$

▷ For the overdrawing effect, a stroke generated earlier should be drawn onto the canvas before a stroke generated later. Since the stroke order is preserved throughout other operations, we simply append newly generated anchors and strokes to the end of their respective lists.
$\mathcal{A} \leftarrow \mathcal{A}' \cup \mathcal{A}^+$
$\mathcal{S} \leftarrow \mathcal{S}' \cup \mathcal{S}^+$

Algorithm 5 `Remove_Hidden`

Input: Anchors $\mathcal{A}$, strokes $\mathcal{S}$, texture T_E, texture T_N
Output: Anchors $\mathcal{A}$, strokes $\mathcal{S}$

▷ Strokes are drawn in reverse order. A buffer tracks the accumulated (reduced) transparency. If no pixels pass the transparency test for the current stroke, we set a remove flag for that stroke.
$n_S \leftarrow |\mathcal{S}|$
$B \leftarrow \{1\}$ ▷ Initialize the transparent buffer to 1 (transparent) everywhere
$\mathcal{J} \leftarrow \emptyset$ ▷ Indices of strokes to be removed
for $j \in \{n_S - 1, n_S - 2, \cdots, 0\}$ **do**
  if not `Visible`$(B, \mathcal{S}[j], T_E)$ **then**
    $\mathcal{J} \leftarrow \mathcal{J} \cup \{j\}$
  end if
  $B \leftarrow$ `Update_Transparency`$(B, \mathcal{S}[j], T_N)$
end for
$\mathcal{A} \leftarrow$ `Remove_from_List`$(\mathcal{A}, \mathcal{J})$
$\mathcal{S} \leftarrow$ `Remove_from_List`$(\mathcal{S}, \mathcal{J})$

Algorithm 6 `TVDRK`

Input: Position $\boldsymbol{u}$, vector field $\boldsymbol{w}$, timestep Δt, Rotation matrix $\mathbf{R}$
Output: Advected position $\boldsymbol{u}'$

$\boldsymbol{v_u} \leftarrow \boldsymbol{w}(\boldsymbol{u})$ ▷ velocity
$\boldsymbol{u}_1 \leftarrow \boldsymbol{u} + \mathbf{R}\boldsymbol{v_u}\Delta t$
$\boldsymbol{v}_1 \leftarrow \boldsymbol{w}(\boldsymbol{u}_1)$
$\boldsymbol{u}_2 \leftarrow \frac{3}{4}\boldsymbol{u} + \frac{1}{4}(\boldsymbol{u}_1 + \mathbf{R}\boldsymbol{v}_1\Delta t)$
$\boldsymbol{v}_2 \leftarrow \boldsymbol{w}(\boldsymbol{u}_2)$
$\boldsymbol{u}' \leftarrow \frac{1}{3}\boldsymbol{u} + \frac{2}{3}(\boldsymbol{u}_2 + \mathbf{R}\boldsymbol{v}_2\Delta t)$
▷ For simplicity, the above TVDRK is presented as a single step, but in our implementation, we decompose the single step into multiple sub-steps for more accurate advection
return $\boldsymbol{u}'$

Algorithm 7 `Generate_Stroke_Half`

Input: Anchor a, attribute sampler S, matrix factor $\mathbf{F}$, step length Δl
Output: Stroke s_h

$w \leftarrow S.w(a.\boldsymbol{u})$
$\mathbf{R} \leftarrow \mathbf{R}(a.\Xi)$ ▷ Rotation corresponding to random angular offset
$L \leftarrow S.L(a.\boldsymbol{u})$ ▷ Region label
$l_s \leftarrow [\frac{1}{2}S.l(a.\boldsymbol{u})]$
$\boldsymbol{u} \leftarrow a.\boldsymbol{u}$
$c \leftarrow \{\boldsymbol{u}\}$ ▷ Center line vertices
while $S.L(\boldsymbol{u}) = L$ **and** `Length`$(s) <$ `Average`(l_s) **do**
  find Δt **such that** $|\|\texttt{TVDRK}(\boldsymbol{u}, S.\boldsymbol{d}, \Delta t, \mathbf{FR}) - \boldsymbol{u}\| - \Delta l| \le \varepsilon_l$
  $\boldsymbol{u} \leftarrow \texttt{TVDRK}(\boldsymbol{u}, S.\boldsymbol{d}, \Delta t, \mathbf{FR})$
  $l_s \leftarrow l_s \cup \{\frac{1}{2}S.l(\boldsymbol{u})\}$
  $c \leftarrow c \cup \{\boldsymbol{u}\}$
end while
$s_h \leftarrow$ `Build_from_Centerline`(c, w)
return s_h

Algorithm 8 `Generate_Stroke`

Input: Anchor a, attribute sampler S, step length Δl
Output: Stroke s

$s^+ \leftarrow$ `Generate_Stroke_Half`$(a, S, \mathbf{I}, \Delta l)$
$s^- \leftarrow$ `Generate_Stroke_Half`$(a, S, -\mathbf{I}, \Delta l)$
return $\{s^- \cup s^+\}$

Algorithm 9 `Deactivate`

Input: Active set A, stroke s, texture T
Output: Updated active set A'

$A' \leftarrow A$
for $P \in s(T)$ **do** ▷ For each pixel drawn for s using T
  $A'(\boldsymbol{u}(P)) \leftarrow$ **inactive**
end for
return A'

8 Scene Statistics

We summarize scene statistics in Table 1, including the camera setting (dynamic or static), medium setting (dynamic or static), presence of surfaces, total number of frames, average number of strokes per frame, and the frame indices of selected exemplar frames.

Algorithm 10 `Find_Random_Location_from_Active_Set`

Input: Active set A
Output: Position $\boldsymbol{u}$

$\mathcal{P} \leftarrow$ list of all active pixels in A
$C(\mathcal{P}) \leftarrow$ cumulative distribution for $\mathcal{P}$
$\boldsymbol{u} \leftarrow$ sampling according to $C(\mathcal{P})$
▷ For efficiency, the operations listed above are accelerated using a quad-tree
return $\boldsymbol{u}$

Algorithm 11 `Visible`

Input: Transparency buffer B, stroke s, texture T
Output: **True** if stroke s is not hidden, **False** otherwise

for $P \in s(T)$ **do** ▷ For each pixel drawn for s using T
    ▷ Check if the pixel drawn is not transparent (using some threshold θ_α):
    $b_{\alpha(P)} \leftarrow \alpha(P) \geq \theta_\alpha$
    ▷ Check if the transparency buffer is not opaque enough to hide the current pixel (using some threshold θ_B):
    $b_{\alpha(B(\boldsymbol{u}(P)))} \leftarrow \alpha(B(\boldsymbol{u}(P))) \geq \theta_B$
    ▷ If both tests passed, then there is at least one pixel that is visible:
    if $b_{\alpha(P)}$ **and** $b_{\alpha(B(\boldsymbol{u}(P)))}$ **then**
        return true
    end if
end for
▷ If none of the pixels passed the tests, then the stroke is invisible:
return false

Algorithm 12 `Update_Transparency`

Input: Transparency buffer B, stroke s, texture T
Output: Updated transparency buffer B'

$B' \leftarrow B$
for $P \in s(T)$ **do** ▷ For each pixel drawn for s using T
    ▷ Multiply the transparency $1-\alpha(P)$ with the values stored in the buffer:
    $B'(\boldsymbol{u}(P)) \leftarrow (1-\alpha(P))B(\boldsymbol{u}(P))$
end for
return B'

Table 1. **Scene statistics.**

Scene	Camera	Medium	Surface	♯Animation frames	Avr. ♯Strokes per frame	Exemplar frames
Rising Smoke	dynamic	dynamic	-	240	57,706	114, 216
Clouds	static	dynamic	-	500	121,125	336, 60, 456
Ring Fire (Dense)	static	dynamic	-	240	56,424	144, 48, 198
Ring Fire (Thin)	static	dynamic	-	240	167,798	186, 72
Dense Static Medium	dynamic	static	-	150	125,605	42
Surface Only	dynamic	-	monkey	149	75,751	42
Wood and Fire	static	dynamic	wood	240	64,103	228, 54, 168, 90
Colliding Smokes	static	dynamic	-	240	36,030	216
Laminar to Turbulent	static	dynamic	-	229	50,484	72, 192, 24
Foggy Forest	dynamic	dynamic	trees, ground	390	55,957	66, 282, 186, 366
Fire	static	dynamic	-	240	56,389	228

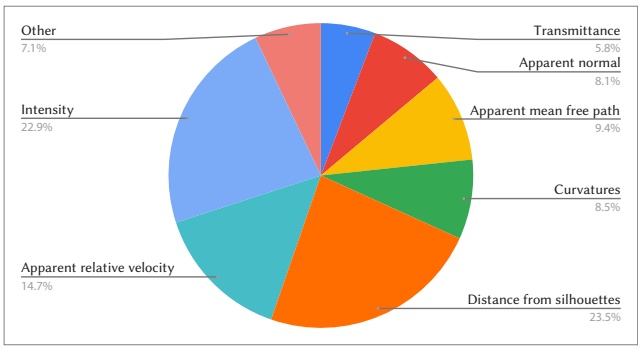

Fig. 2. **Breakdown of the computation time for features and basis fields.**

9 Detailed Timing

The computation of features and basis fields is embarrassingly parallel across frames. With Apple Silicon M4 Max CPU, computations can be distributed to the 12 Performance Cores at a time, and a scene with participating media at a volume resolution of $512 \times 512 \times 512$ and a screen resolution of 512×512 takes approximately 7 seconds per frame under this parallelism. Figure 2 presents a breakdown of the computation time for features and basis fields.

For exemplar frame selection, we downsample feature maps to 128×128 and sample every sixth frame, yielding 40 candidates for a 240-frame animation. Fitting a GMM takes 0.37 seconds per frame, and the exemplar selection algorithm adds 0.10, 0.20, 0.29, and 0.40 seconds per frame for selecting the first through fourth exemplars, respectively. If the process selects two exemplars, it incurs the cost of attempting a third and discarding it, resulting in a total of $40 \times (0.37 + 0.10 + 0.20 + 0.29) = 38.8$ seconds of computation.

The regression takes approximately 3 seconds, and the transfer takes 6.2 seconds per frame. The stroke generation time depends on the number of generated strokes (e.g., about 9 seconds for 30,000 strokes and 60 seconds for 130,000 strokes).

Preparing a single exemplar frame requires approximately 20 minutes for painting colors and 15 minutes for defining orientations, widths, and lengths.

10 Comparison to Previous Methods

We compare our method against two patch-based approaches [Fišer et al. 2016; Texler et al. 2020], a neural transfer method [Ghiasi et al. 2017], two stroke-based neural transfer methods [Hu et al. 2023; Kotovenko et al. 2021], and a diffusion-based text-to-video method [Liu et al. 2024]. These methods were selected based on the availability of source code and successful execution without errors.

For the methods by Fišer et al.[2016], Ghiasi et al.[2017], and Kotovenko et al.[2021], whose implementations accept only a single style image, we used the first exemplar frame selected by our method and drawn by the user. Texler et al.[2020] supports multiple style images; we provided all user-drawn exemplars selected by our method. For Hu et al. [2023], which requires a sequence of stylized images, we used color attributes obtained by applying the stroke transfer framework to image-based features.

Liu et al. [2024] did not release the implementation supporting conditional inputs. Therefore, we used our user-drawn exemplar frames as style images, along with a text prompt describing the scene.

Results are presented in our supplementary and additional videos. Participating media require structured, temporally coherent strokes to represent dynamic, non-rigid motion. Patch-based methods tend to introduce stitching artifacts and often fail to preserve stroke structure. Similar issues appear in the neural transfer method [Ghiasi et al. 2017], where brushstroke fidelity and motion coherence degrade significantly.

Stroke-based neural transfer methods [Hu et al. 2023; Kotovenko et al. 2021] struggled to reproduce the intended stroke shapes—often generating blocky patterns—and frequently altered stroke colors and styles. These inconsistencies become more pronounced in animations. The text-to-video method [Liu et al. 2024], when used without conditioning, was unable to generate stroke-based animations aligned with the desired scene.

In contrast, our method avoids these issues and produces temporally coherent stroke animations that faithfully reflect the user's intended style.

11 Importance of Features

In Figure 3, we present the learned model $\mathbf{M_v}$ for each scene, highlighting the diverse utilization of features across different scenarios. For instance, Ring-Fire (Dense) primarily relies on intensity-based features, while Laminar-to-Turbulence heavily utilizes velocity-based features. The Surface-only scene incorporates curvature and normal features alongside intensity-based, velocity-based, silhouette and transmittance[1] features, and Foggy-Forest also makes use of curvature features.

Our feature and basis field design is not aimed at minimizing the feature set; instead, we focus on expanding the model's degrees of freedom to simplify the learning process. This approach enables the system to effectively train models even with low-order (e.g., linear) regression.

12 Ablation Settings

We detail the five settings (no illuminations, no normals or curvatures, no silhouettes, no velocities, and no additional volumetric cues) in Table 2.

13 Performance of Transfer

To further validate the effectiveness of the attribute transfer, we conducted an additional test using validation exemplars provided by a human user for intermediate frames. These frames were not included during training. The user annotated stroke widths, lengths, and orientations on *rendered* images, which served as validation exemplars. We then compared these manually specified attributes with those predicted by our original model. As shown in Figures 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, and 14, the results closely align, demonstrating that our method robustly generalizes and reliably transfers stylistic attributes to unseen frames.

[1]For this surface only example, the transmittance is essentially a binary mask separating the foreground from the background.

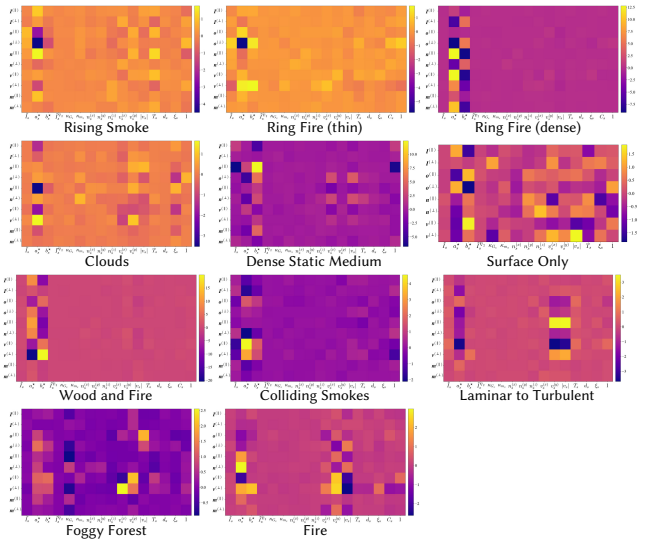

Fig. 3. **Learned model $\mathbf{M_v}$ for each scene.**

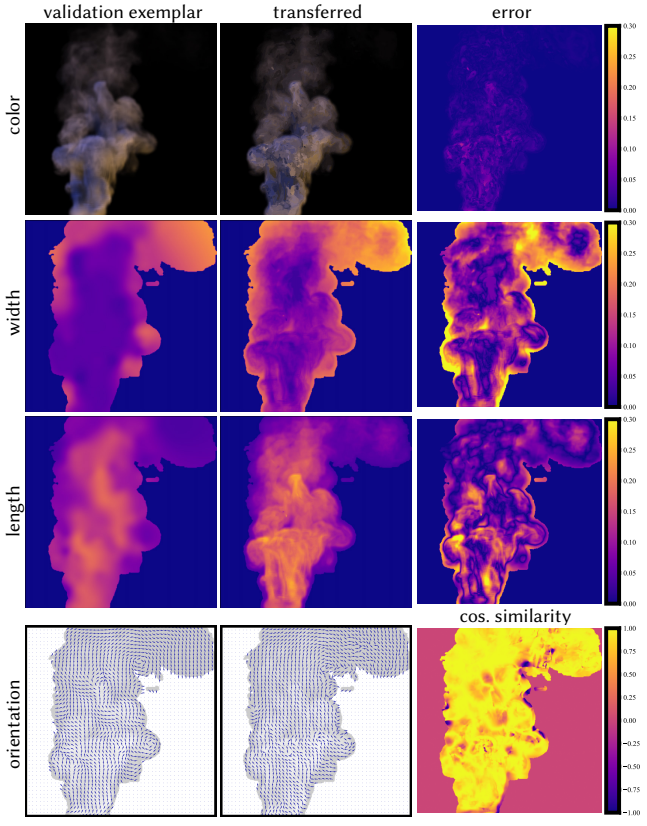

Fig. 4. **Validation of attributes for an intermediate frame (Rising Smoke).** Errors for colors, widths, and lengths are relative errors. For orientation, we show cosine similarity.

Table 2. **Features and basis fields included in each setting of the ablation study**. Sections refer to those in the main paper.

Features / basis fields	No illuminations	No normals or curvatures	No silhouettes	No velocities	No additional volumetric cues
Intensity and apparent intensity gradient (§5.2.1)	-	✓	✓	✓	✓
Apparent Gaussian and mean curvatures (§5.2.2)	✓	-	✓	✓	✓
Apparent normals (§5.2.3)	✓	-	✓	✓	✓
Temperature (§5.2.4)	✓	✓	✓	✓	-
Apparent relative velocity (§5.2.5)	✓	✓	✓	-	✓
Transmittance (§5.2.6)	✓	✓	✓	✓	-
Apparent mean free-path (§5.2.7)	✓	✓	✓	✓	-
Distance from silhouettes (§5.2.8)	✓	✓	-	✓	✓
Intensity gradient and its 90° rotation	-	✓	✓	✓	✓
Silhouette-guided direction and its 90° rotation	✓	✓	-	✓	✓
Apparent normal and its 90° rotation	✓	-	✓	✓	✓
Apparent relative velocity and its 90° rotation	✓	✓	✓	-	✓
The gradient of apparent mean free-path and its 90° rotation	✓	✓	✓	✓	-

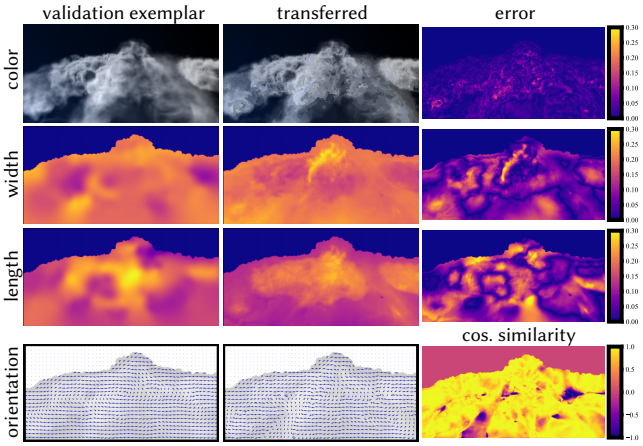

Fig. 5. **Validation of attributes for an intermediate frame (Clouds)**. Errors for colors, widths, and lengths are relative errors. For orientation, we show cosine similarity.

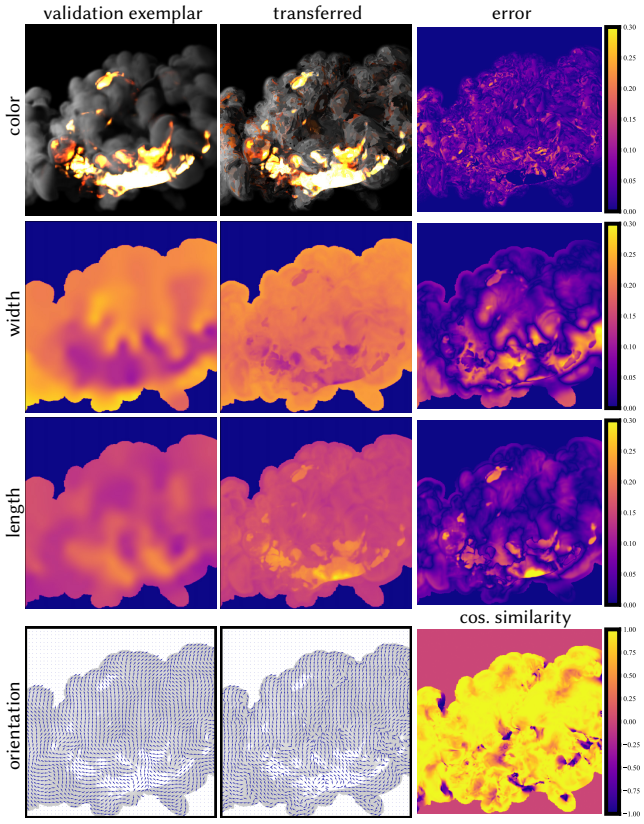

Fig. 6. **Validation of attributes for an intermediate frame (Ring Fire (Dense))**. Errors for colors, widths, and lengths are relative errors. For orientation, we show cosine similarity.

14 Proof for Free-Path Distribution Approaching Delta Function

Definition 1. *We say that $\sigma(x)$ is an extinction function if $\sigma(x) \geq 0$ for ${}^{\forall}x \in \mathbb{R}^{\not}$, where $\mathbb{R}^{\not} = [0, \infty)$. We assume that the viewpoint is located at $x = 0$, and the region ${}^{\forall}x \in \mathbb{R}^{\not}$ corresponds to the line of sight.*

Definition 2. *We say that an extinction function $\sigma(x)$ is reasonably mild if and only if $\sigma(x)$ is bounded (i.e., $0 < {}^{\exists}M_\sigma < \infty$, such that $|\sigma(x)| < M_\sigma$) and Riemann integrable.*

Definition 3. *We say that an extinction function $\sigma(x)$ has a vacuum-medium boundary at $x = x_0$, where $x_0 > 0$, if and only if $\sigma(x) = 0$ for $x < x_0$, $\sigma(x)$ is right-differentiable at $x = x_0$, and there exists $\gamma_\sigma > 0$ such that $\sigma(x) > 0$ for $x_0 < x \leq x_0 + \gamma_\sigma$. The last two conditions can be equivalently posed as 1) $\sigma(x)$ is right-continuous ($\lim_{x' \downarrow x} \sigma(x') = \sigma(x)$) for $x_0 < x \leq x_0 + \gamma_\sigma$, and 2) for $0 < {}^{\forall}\varepsilon \leq \gamma_\sigma$, $\sigma(x)$ is bounded in the following sense:*

$$\sigma_{x_0} + k_1(\varepsilon)\varepsilon \leq \sigma(x_0 + \varepsilon) \leq \sigma_{x_0} + k_2(\varepsilon)\varepsilon, \tag{32}$$

where γ_σ (and hence ε) can be taken small enough so that $\sigma_{x_0} + k_1(\varepsilon)\varepsilon > 0$ for $0 < x \leq \varepsilon$, and as $\varepsilon \downarrow 0$, the bound can be made tighter such that $\lim_{\varepsilon \downarrow 0} \frac{k_1(\varepsilon)}{k_2(\varepsilon)} = 1$. These conditions are for defining the boundary (0 extinction between the viewpoint at $x = 0$ and the boundary $x = x_0$, and a region with non-zero length of non-zero extinction right beyond the boundary) while ruling out pathological cases. We also

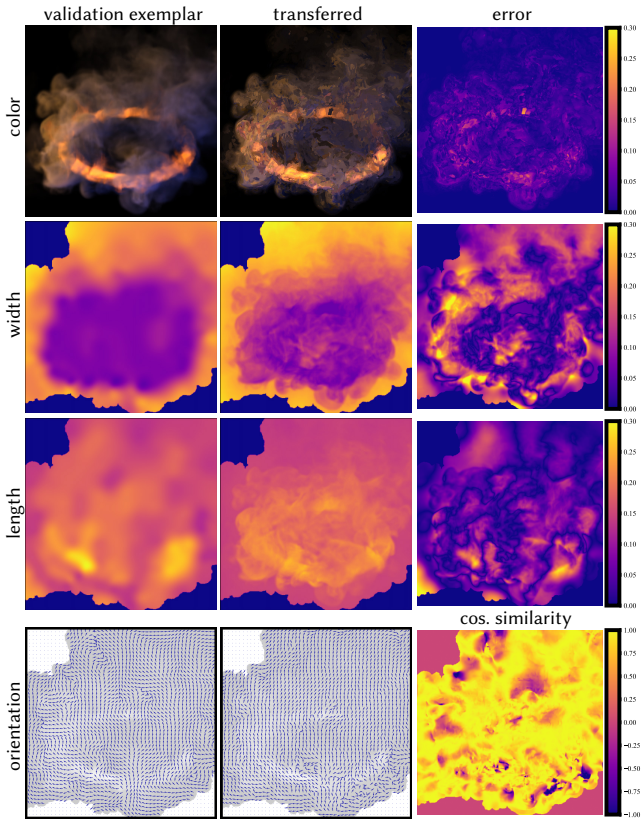

Fig. 7. **Validation of attributes for an intermediate frame (Ring Fire (Thin))**. Errors for colors, widths, and lengths are relative errors. For orientation, we show cosine similarity.

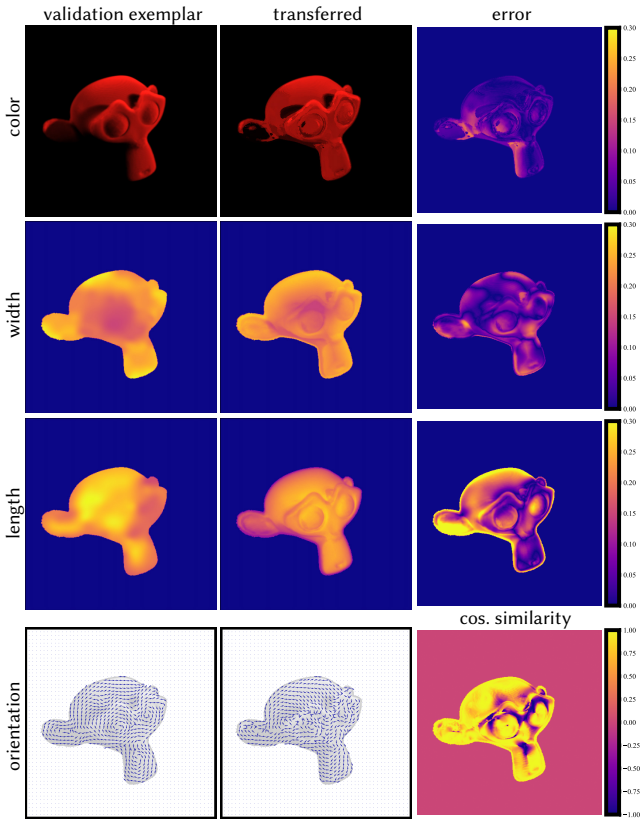

Fig. 8. **Validation of attributes for an intermediate frame (Dense Static Medium)**. Errors for colors, widths, and lengths are relative errors. For orientation, we show cosine similarity.

use the notation $\sigma(x;x_0)$ to imply the extinction function having a vacuum-medium boundary at $x = x_0$.

Remark 1. *Definition 3 allows $\sigma(x;x_0)$ to be discontinuous and have a step edge at $x = x_0$; while $\lim_{x'\uparrow x_0}\sigma(x') = 0$, $\lim_{x'\downarrow x_0}\sigma(x')$ can be either 0 or some non-zero positive value (due to the step edge). In addition, it is possible that k_1 and k_2 are both negative if beyond the step edge the extinction is decreasing.*

Definition 4. *For a reasonably mild extinction function $\sigma(x;x_0)$, we define its transmittance $T_\sigma(x;x_0)$ as*

$$T_\sigma(x;x_0) = \exp\left(-\int_0^x \sigma(x';x_0)dx'\right) = \exp\left(-\int_{x_0}^x \sigma(x';x_0)dx'\right). \tag{33}$$

For $x < x_0$, $T_\sigma(x;x_0) = 1$. We are using the subscript to T to indicate the associated extinction function. Note that since $\sigma(x) \geq 0$, we can easily see that $T_\sigma(x;x_0)$ is a non-increasing function (for $x \in \mathbb{R}^{\not\perp}$).

Definition 5. *For a reasonably mild extinction function $\sigma(x;x_0)$, we define its free path distribution (a probability density function) p^{fp}_σ as*

$$p^{fp}_\sigma(x;x_0) := \sigma(x;x_0)T_\sigma(x;x_0). \tag{34}$$

Again, we are using the subscript to indicate the associated extinction function.

Definition 6. *Let $\gamma_f > 0$. We say that a function $f(x)$ is reasonably smooth at $[x_0, x_0+\gamma_f]$ if and only if $f(x)$ is bounded in the following Lipschitz sense as*

$$f_{x_0} + k_3(x - x_0) \leq f(x) \leq f_{x_0} + k_4(x - x_0) \tag{35}$$

for $x_0 \leq x \leq x_0 + \gamma_f$, where $f_{x_0} = f(x_0)$.

Theorem 1. *Let $\sigma(x;x_0)$ be a reasonably mild extinction function (Definitions 2 and 3). Let $\sigma_a(x;x_0) = a\sigma(x;x_0)$ (note that $\sigma_a(x;x_0)$ is also a reasonably mild extinction function with a vacuum-medium boundary at $x = x_0$).*

Let a function $f(x) \in L^1$ be bounded (i.e., $0 < {}^\exists M_f < \infty$ such that $|f(x)| \leq M_f$), absolutely integrable (i.e., $0 < {}^\exists M_{\int|f|} < \infty$, such that $\int_{-\infty}^{\infty}|f(x)|dx < M_{\int|f|}$), and reasonably smooth at $[x_0, x_0+\gamma_f]$ for some $\gamma_f > 0$ (Definition 6).

Then,

$$\lim_{a\to\infty}\int_0^\infty p^{fp}_{\sigma_a}(x;x_0)f(x)dx = f(x_0). \tag{36}$$

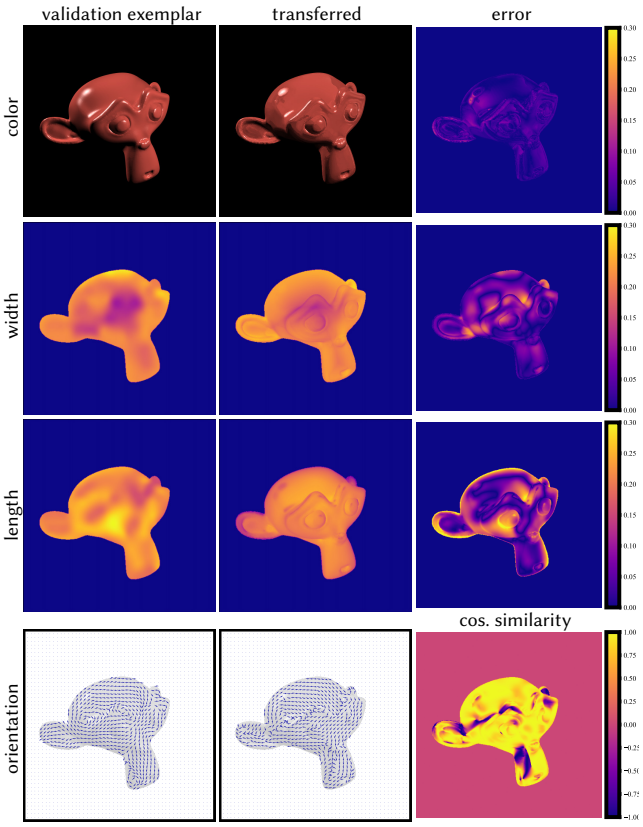

Fig. 9. **Validation of attributes for an intermediate frame (Surface Only)**. Errors for colors, widths, and lengths are relative errors. For orientation, we show cosine similarity.

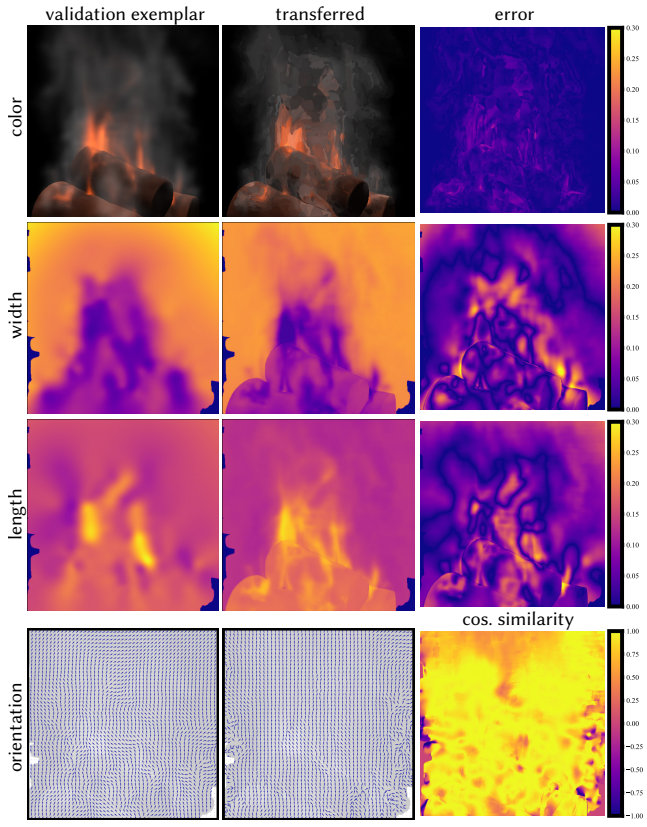

Fig. 10. **Validation of attributes for an intermediate frame (Wood and Fire)**. Errors for colors, widths, and lengths are relative errors. For orientation, we show cosine similarity.

Remark 2. *In the sense of Theorem 1, we say*

$$\lim_{a\to\infty} p^{fp}_{\sigma_a}(x;x_0) = \delta(x,x_0). \tag{37}$$

Remark 3. *For cases where $\sigma(x;x_0)$ is Lebesgue integrable but not Riemann integrable, if there exists $\tilde{\sigma}(x;x_0)$ such that $\tilde{\sigma}(x;x_0) = \sigma(x;x_0)$ except for 0 measure discontinuities and $\tilde{\sigma}(x;x_0)$ is right-continuous for $(x_0, x_0+\gamma_\sigma]$ (i.e., $\tilde{\sigma}(x;x_0)$ is a continuous version of $\sigma(x;x_0)$ in $(x_0, x_0+\gamma_\sigma]$), we may still apply the above theorem for $\tilde{\sigma}(x;x_0)$, provided that other conditions imposed by the theorem on $\tilde{\sigma}(x;x_0)$ hold.*

Proof. Let $\gamma = \min(\gamma_\sigma, \gamma_f)$. Note that $\gamma > 0$. We take ε as a function of a such that $0 < \varepsilon(a) \le \gamma$. To simplify notation, we write ε_a instead of $\varepsilon(a)$. We split the integral (36) as

$$\begin{aligned}
&\lim_{a\to\infty} \int_0^\infty p^{\mathrm{fp}}_{\sigma_a}(x;x_0) f(x) dx \\
&= \lim_{a\to\infty} \int_0^{x_0} p^{\mathrm{fp}}_{\sigma_a}(x;x_0) f(x) dx + \lim_{a\to\infty} \int_{x_0}^{x_0+\varepsilon_a} p^{\mathrm{fp}}_{\sigma_a}(x;x_0) f(x) dx \\
&\quad + \lim_{a\to\infty} \int_{x_0+\varepsilon_a}^\infty p^{\mathrm{fp}}_{\sigma_a}(x;x_0) f(x) dx.
\end{aligned} \tag{38}$$

The first term of (38) is clearly 0. The ideas behind splitting the integral into these three terms are that 1) as a is cranked up, the free path distribution will more condense into the region $[x_0, x_0+\varepsilon_a]$ of the middle term, and that 2) the last term will diminish if ε is driven to 0 in an appropriate speed (which cannot be too fast, otherwise the free path distribution will not be dominant in the middle term; the detailed condition will be clear later on).

For the last term of (38), we name it V and define

$$\hat{V}_a(\varepsilon_a;x_0) := \int_{x_0+\varepsilon_a}^\infty p^{\mathrm{fp}}_{\sigma_a}(x;x_0) f(x) dx, \tag{39}$$

so

$$V = \lim_{a\to\infty} \hat{V}_a(\varepsilon_a;x_0). \tag{40}$$

For any x satisfying $x_0 \le x \le x_0+\varepsilon_a \le x_0+\gamma$, we have

$$\begin{aligned}
0 \le \underbrace{T_{\sigma_a}(x;x_0) \le \exp\left(-\int_{x_0}^x a(\sigma_{x_0} + k_1(\varepsilon_a)(x'-x_0))dx'\right)}_{(33),(32)} \\
= \exp\left(-a(\sigma_{x_0}(x-x_0) + \frac{k_1(\varepsilon_a)}{2}(x-x_0)^2)\right).
\end{aligned} \tag{41}$$

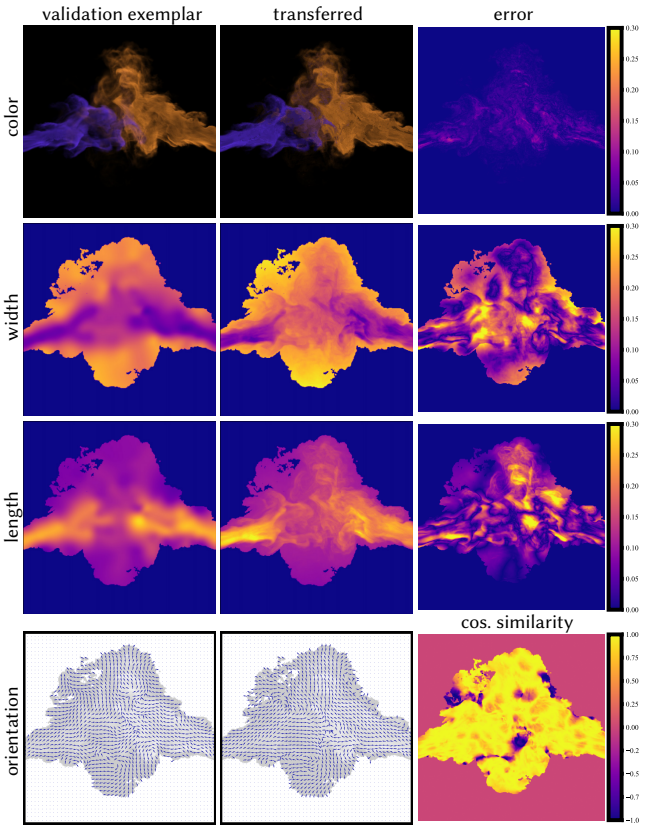

Fig. 11. **Validation of attributes for an intermediate frame (Colliding Smokes)**. Errors for colors, widths, and lengths are relative errors. For orientation, we show cosine similarity.

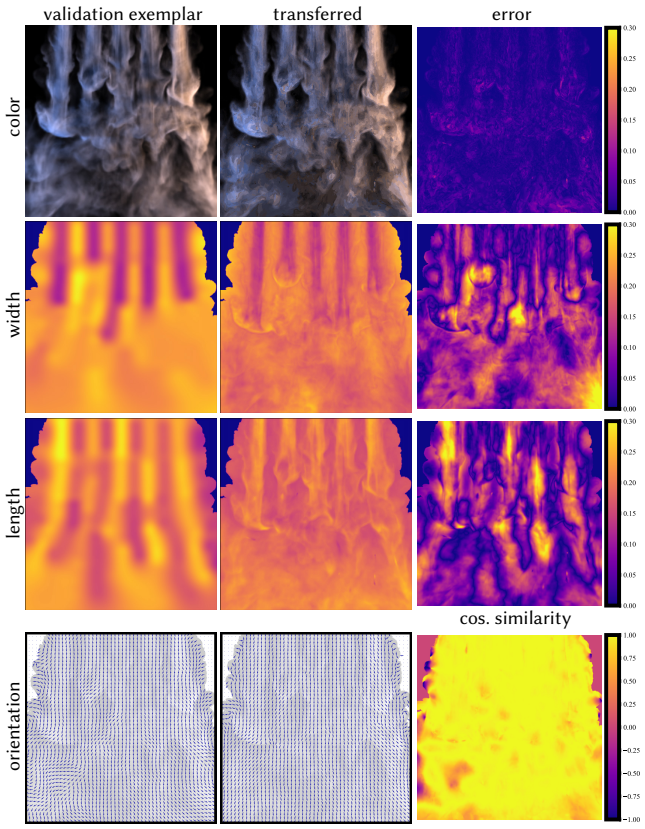

Fig. 12. **Validation of attributes for an intermediate frame (Laminar to Turbulent)**. Errors for colors, widths, and lengths are relative errors. For orientation, we show cosine similarity.

Because $T_{\sigma_a}(x;x_0)$ is a non-increasing function (for $x \in \mathbb{R}^{\prime}$), we can bound $T_{\sigma_a}(x;x_0)$ for $x \geq x_0 + \varepsilon_a$ by setting x to $x_0 + \varepsilon_a$ as

$$0 \leq T_{\sigma_a}(x;x_0) \leq \exp\left(-a(\sigma_{x_0}\varepsilon_a + \frac{k_1(\varepsilon_a)}{2}\varepsilon_a^2)\right). \tag{42}$$

Combined with

$$0 \leq \sigma_a(x;x_0) \leq aM_\sigma, \tag{43}$$

we have

$$0 \leq p^{\text{fp}}_{\sigma_a}(x;x_0) \leq aM_\sigma \exp\left(-a(\sigma_{x_0}\varepsilon_a + \frac{k_1(\varepsilon_a)}{2}\varepsilon_a^2)\right) \tag{44}$$

for $x \geq x_0 + \varepsilon_a$.

So, we have

$$\begin{aligned}
&|\hat{V}_a(\varepsilon_a;x_0)| = \left|\int_{x_0+\varepsilon_a}^{\infty} p^{\text{fp}}_{\sigma_a}(x;x_0)f(x)dx\right| \\
&\leq \int_{x_0+\varepsilon_a}^{\infty} |p^{\text{fp}}_{\sigma_a}(x;x_0)||f(x)|dx = \int_{x_0+\varepsilon_a}^{\infty} p^{\text{fp}}_{\sigma_a}(x;x_0)|f(x)|dx \\
&\leq \int_{x_0+\varepsilon_a}^{\infty} aM_\sigma \exp\left(-a(\sigma_{x_0}\varepsilon_a + \frac{k_1(\varepsilon_a)}{2}\varepsilon_a^2)\right)|f(x)|dx \\
&= aM_\sigma \exp\left(-a(\sigma_{x_0}\varepsilon_a + \frac{k_1(\varepsilon_a)}{2}\varepsilon_a^2)\right)\int_{x_0+\varepsilon_a}^{\infty} |f(x)|dx \\
&\leq aM_\sigma \exp\left(-a(\sigma_{x_0}\varepsilon_a + \frac{k_1(\varepsilon_a)}{2}\varepsilon_a^2)\right) M_{\int|f|}.
\end{aligned} \tag{45}$$

Hence,

$$\lim_{a\to\infty} |\hat{V}_a(\varepsilon_a;x_0)| \leq \lim_{a\to\infty} aM_\sigma \exp\left(-a(\sigma_{x_0}\varepsilon_a + \frac{k_1(\varepsilon_a)}{2}\varepsilon_a^2)\right) M_{\int|f|}. \tag{46}$$

If $a\exp\left(-a(\sigma_{x_0}\varepsilon_a + \frac{k_1(\varepsilon_a)}{2}\varepsilon_a^2)\right) \to 0$ as $a \to \infty$, then we have $\lim_{a\to\infty} |\hat{V}_a(\varepsilon_a;x_0)| = 0$, so $V(\varepsilon_a) = 0$ and the last term of (38) diminishes.

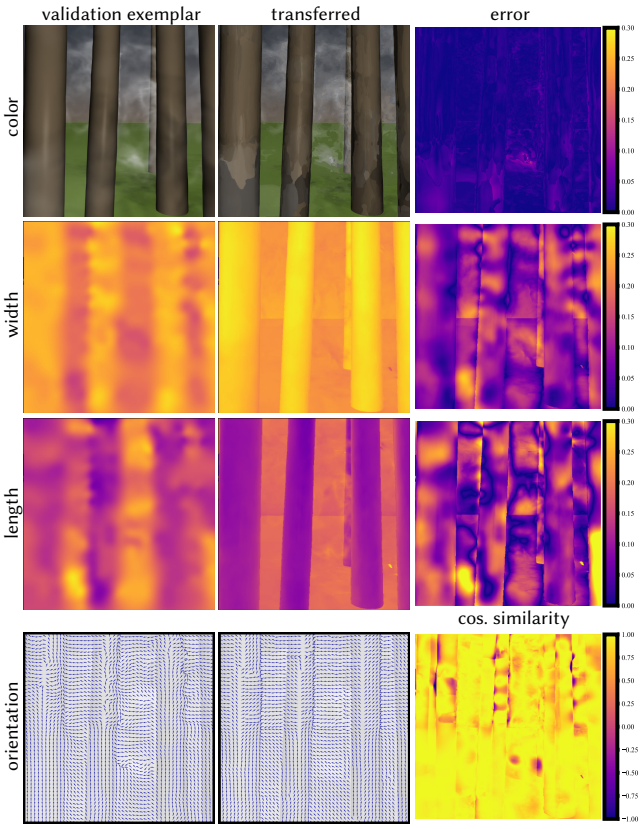

Fig. 13. **Validation of attributes for an intermediate frame (Foggy Forest)**. Errors for colors, widths, and lengths are relative errors. For orientation, we show cosine similarity.

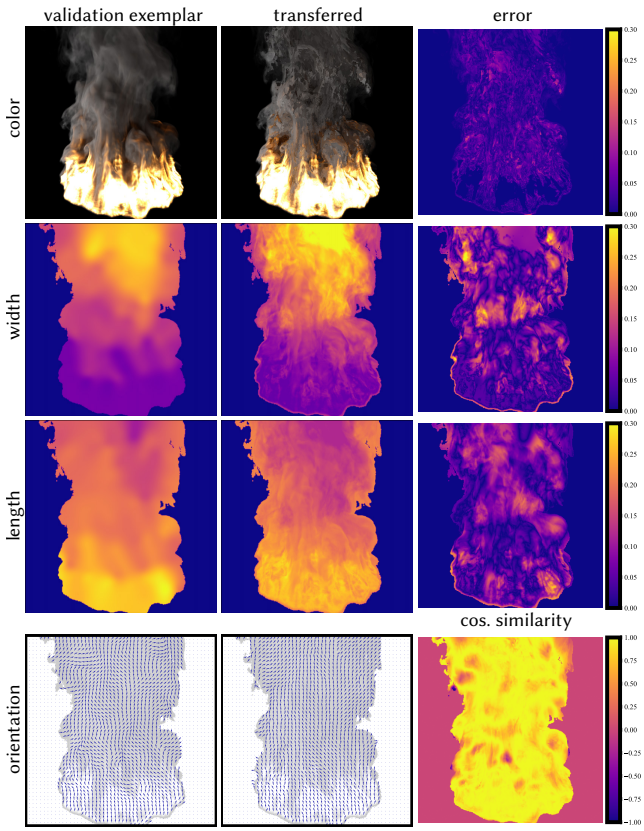

Fig. 14. **Validation of attributes for an intermediate frame (Fire)**. Errors for colors, widths, and lengths are relative errors. For orientation, we show cosine similarity.

Let us estimate the speed required for ε_a. From Lemma 1, we have $\lim_{a\to\infty} a\exp(-ca^s) = 0$ if $s > 0$ and $c > 0$. Hence, if $^\exists s > 0$ and $^\exists c > 0$ such that $a(\sigma_{x_0}\varepsilon_a + \frac{k_1(\varepsilon_a)}{2}\varepsilon_a^2) \geq ca^s$, we obtain the desired convergence. In fact, we can arbitrary pick a parameter δ from $(0, \frac{1}{2})$. Then, for the case of $\sigma_{x_0} > 0$, we can take $\varepsilon_a = a^{-1+\delta}$, which gives $\lim_{a\to\infty} \varepsilon_a = 0$, and

$$a\left(\sigma_{x_0}\varepsilon_a + \frac{k_1(\varepsilon_a)}{2}\varepsilon_a^2\right) = \sigma_{x_0}a^\delta + \frac{k_1(\varepsilon_a)}{2}\underbrace{a^{-1+2\delta}}_{\to 0 \text{ as } a\to\infty}. \tag{47}$$

On the other hand, if $\sigma_{x_0} = 0$, condition (32) requires $k_1(\varepsilon_a) > 0$. For this case we can take $\varepsilon_a = a^{-\frac{1}{2}+\delta}$, which gives $\lim_{a\to\infty} \varepsilon_a = 0$ and

$$a\left(\sigma_{x_0}\varepsilon_a + \frac{k_1(\varepsilon_a)}{2}\varepsilon_a^2\right) = \frac{k_1(\varepsilon_a)}{2}a^{2\delta}. \tag{48}$$

Thus, $a\exp\left(-a(\sigma_{x_0}\varepsilon_a + \frac{k_1(\varepsilon_a)}{2}\varepsilon_a^2)\right) \to 0$ as $a \to \infty$, and the last term of (38) diminishes. Later on, we will find that the same choices of ε_a work for the computation of the middle term of (38).

Next, we compute the middle term of (38). Again, we consider two cases: i) $\sigma_{x_0} > 0$ and ii) $\sigma_{x_0} = 0$. For i), we can bound $\sigma(x; x_0)$ as

$$0 < \sigma_m(\varepsilon_a) \leq \sigma(x; x_0) \leq \sigma_M(\varepsilon_a) < \infty, \tag{49}$$

for $x_0 \leq x \leq x_0+\varepsilon_a \leq x_0+\gamma$, with $lim_{\varepsilon_a\downarrow 0}\sigma_m(\varepsilon_a) = lim_{\varepsilon_a\downarrow 0}\sigma_M(\varepsilon_a) = \sigma(x_0; x_0) = \sigma_{x_0}$. Hence, for $x_0 \leq x \leq x_0 + \varepsilon_a$,

$$0 < a\sigma_m(\varepsilon_a) \leq \sigma_a(x; x_0) \leq a\sigma_M(\varepsilon_a) < \infty, \tag{50}$$

and

$$\exp(-a\sigma_M(\varepsilon_a)(x - x_0)) \leq T_{\sigma_a}(x; x_0) \leq \exp(-a\sigma_m(\varepsilon_a)(x - x_0)), \tag{51}$$

so

$$\tilde{p}^{\mathrm{fp}}_{\sigma_a}(x; x_0, \sigma_m(\varepsilon_a), \sigma_M(\varepsilon_a)) \leq p^{\mathrm{fp}}_{\sigma_a}(x; x_0) \leq \tilde{p}^{\mathrm{fp}}_{\sigma_a}(x; x_0, \sigma_M(\varepsilon_a), \sigma_m(\varepsilon_a)), \tag{52}$$

where

$$\tilde{p}^{\mathrm{fp}}_{\sigma_a}(x; x_0, s, t) := as\exp(-at(x - x_0)) = \frac{s}{t}\frac{d}{dx}\left(1 - \exp(-at(x - x_0))\right). \tag{53}$$

Recall that $f(x)$ is bounded as

$$f_{x_0} + k_3(x - x_0) \leq f(x) \leq f_{x_0} + k_4(x - x_0), \tag{54}$$

according to (6). Then,

$$\begin{aligned}
&\int_{x_0}^{x_0+\varepsilon_a} p^{\mathrm{fp}}_{\sigma_a}(x;x_0)f(x)dx \geq V_a(\varepsilon_a,\sigma_m(\varepsilon_a),\sigma_M(\varepsilon_a),f_{x_0},k_3;x_0)\\
&:= \int_{x_0}^{x_0+\varepsilon_a} \tilde{p}^{\mathrm{fp}}_{\sigma_a}(x;x_0,\sigma_m(\varepsilon_a),\sigma_M(\varepsilon_a))(f_{x_0}+k_3(x-x_0))dx\\
&= \left[\tilde{P}^{\mathrm{fp}}_{\sigma_a}(x;x_0,\sigma_m(\varepsilon_a),\sigma_M(\varepsilon_a))(f_{x_0}+k_3(x-x_0))\right]_{x_0}^{x_0+\varepsilon_a}\\
&\quad - \int_{x_0}^{x_0+\varepsilon_a} \tilde{P}^{\mathrm{fp}}_{\sigma_a}(x;x_0,\sigma_m(\varepsilon_a),\sigma_M(\varepsilon_a))\frac{d}{dx}(f_{x_0}+k_3(x-x_0))dx,
\end{aligned} \tag{55}$$

where $\tilde{P}^{\mathrm{fp}}_{\sigma_a}(x;x_0,s,t) = \frac{s}{t}(1-\exp(-at(x-x_0)))$. Continuing the computation, we have

$$\begin{aligned}
&V_a(\varepsilon_a,\sigma_m(\varepsilon_a),\sigma_M(\varepsilon_a),f_{x_0},k_3;x_0)\\
&= \tilde{P}^{\mathrm{fp}}_{\sigma_a}(x_0+\varepsilon_a;x_0,\sigma_m(\varepsilon_a),\sigma_M(\varepsilon_a))(f_{x_0}+k_3\varepsilon_a)\\
&\quad - \underbrace{\tilde{P}^{\mathrm{fp}}_{\sigma_a}(x_0;x_0,\sigma_m(\varepsilon_a),\sigma_M(\varepsilon_a))}_{=0} f_{x_0}\\
&\quad - \int_{x_0}^{x_0+\varepsilon_a} \frac{k_3\sigma_m(\varepsilon_a)}{\sigma_M(\varepsilon_a)}(1-\exp(-a\sigma_M(\varepsilon_a)(x-x_0)))dx\\
&= \tilde{P}^{\mathrm{fp}}_{\sigma_a}(x_0+\varepsilon_a;\sigma_m(\varepsilon_a),\sigma_M(\varepsilon_a))(f_{x_0}+k_3\varepsilon_a)\\
&\quad - \frac{k_3\sigma_m(\varepsilon_a)}{\sigma_M(\varepsilon_a)}\left[x+\frac{\exp(-a\sigma_M(\varepsilon_a)(x-x_0))}{a\sigma_M(\varepsilon_a)}\right]_{x_0}^{x_0+\varepsilon_a}\\
&= \frac{\sigma_m(\varepsilon_a)}{\sigma_M(\varepsilon_a)}(1-\exp(-a\sigma_M(\varepsilon_a)\varepsilon_a))(f_{x_0}+k_3\varepsilon_a)\\
&\quad - \frac{k_3\sigma_m(\varepsilon_a)}{\sigma_M(\varepsilon_a)}\varepsilon_a + \frac{k_3\sigma_m(\varepsilon_a)}{a(\sigma_M(\varepsilon_a))^2}(1-\exp(-a\sigma_M(\varepsilon_a)\varepsilon_a)).
\end{aligned} \tag{56}$$

If we take δ from $(0,\frac{1}{2})$ (which aligns with the condition required for the case when computing the last term of (38)), and set $\varepsilon_a = a^{-1+\delta}$, then, $\lim_{a\to\infty}\varepsilon_a = 0$ and $\lim_{a\to\infty} a\varepsilon_a = \lim_{a\to\infty} a^{\delta} = \infty$, thus $\lim_{a\to\infty}(1-\exp(-a\sigma_M(\varepsilon_a)\varepsilon_a)) = 1$.

Hence, in terms of $V_a(\varepsilon_a,\sigma_m(\varepsilon_a),\sigma_M(\varepsilon_a),f_{x_0},k_3;x_0)$, we have

$$\begin{aligned}
&\lim_{a\to\infty} V_a(\varepsilon_a,\sigma_m(\varepsilon_a),\sigma_M(\varepsilon_a),f_{x_0},k_3;x_0)\\
&= \lim_{a\to\infty} \underbrace{\frac{\sigma_m(\varepsilon_a)}{\sigma_M(\varepsilon_a)}}_{\to 1}\underbrace{(1-\exp(-a\sigma_M(\varepsilon_a)\varepsilon_a))}_{\to 1}\underbrace{(f_{x_0}+k_3\varepsilon_a)}_{\to f_{x_0}}\\
&\quad - \lim_{a\to\infty}\underbrace{\frac{k_3\sigma_m(\varepsilon_a)}{\sigma_M(\varepsilon_a)}}_{\to 0}\varepsilon_a + \lim_{a\to\infty}\underbrace{\frac{k_3\sigma_m(\varepsilon_a)}{a(\sigma_M(\varepsilon_a))^2}}_{\to 0}\underbrace{(1-\exp(-a\sigma_M(\varepsilon_a)\varepsilon_a))}_{\to 1}\\
&= f_{x_0}.
\end{aligned} \tag{57}$$

Likewise,

$$\int_{x_0}^{x_0+\varepsilon_a} p^{\mathrm{fp}}_{\sigma_a}(x;x_0)f(x)dx \leq V_a(\varepsilon_a,\sigma_M(\varepsilon_a),\sigma_m(\varepsilon_a),f_{x_0},k_4;x_0). \tag{58}$$

Taking the limit of $a\to\infty$, we have

$$\lim_{a\to\infty} V_a(\varepsilon_a,\sigma_M(\varepsilon_a),\sigma_m(\varepsilon_a),f_{x_0},k_4;x_0) = f_{x_0}. \tag{59}$$

In summary, we have

$$\lim_{a\to\infty}\int_{x_0}^{x_0+\varepsilon_a} p^{\mathrm{fp}}_{\sigma_a}(x;x_0)f(x)dx = f_{x_0}. \tag{60}$$

For case ii), we can bound $\sigma(x;x_0)$ as

$$k_1(\varepsilon_a)(x-x_0) \leq \sigma(x;x_0) \leq k_2(\varepsilon_a)(x-x_0), \tag{61}$$

with $0 < k_1(\varepsilon_a) \leq k_2(\varepsilon_a)$. Then,

$$ak_1(\varepsilon_a)(x-x_0) \leq \sigma_a(x;x_0) \leq ak_2(\varepsilon_a)(x-x_0), \tag{62}$$

and

$$\exp\left(-\frac{ak_2(\varepsilon_a)}{2}(x-x_0)^2\right) \leq T_a(x;x_0) \leq \exp\left(-\frac{ak_1(\varepsilon_a)}{2}(x-x_0)^2\right), \tag{63}$$

so

$$\hat{p}^{\mathrm{fp}}_{\sigma_a}(x;x_0,k_1(\varepsilon_a),k_2(\varepsilon_a)) \leq p^{\mathrm{fp}}_{\sigma_a}(x;x_0) \leq \hat{p}^{\mathrm{fp}}_{\sigma_a}(x;x_0,k_2(\varepsilon_a),k_1(\varepsilon_a)), \tag{64}$$

where

$$\begin{aligned}
&\hat{p}^{\mathrm{fp}}_{\sigma_a}(x;x_0,k_1(\varepsilon_a),k_2(\varepsilon_a))\\
&= ak_1(\varepsilon_a)(x-x_0)\exp\left(-\frac{ak_2(\varepsilon_a)}{2}(x-x_0)^2\right)\\
&= \frac{k_1(\varepsilon_a)}{k_2(\varepsilon_a)}\frac{d}{dx}\left(1-\exp\left(-\frac{ak_2(\varepsilon_a)}{2}(x-x_0)^2\right)\right).
\end{aligned} \tag{65}$$

Again, recall that $f(x)$ is bounded as

$$f_{x_0}+k_3(x-x_0) \leq f(x) \leq f_{x_0}+k_4(x-x_0), \tag{66}$$

according to (6). So,

$$\begin{aligned}
&\int_{x_0}^{x_0+\varepsilon_a} p^{\mathrm{fp}}_{\sigma_a}(x;x_0)f(x)dx \geq \hat{V}_a(\varepsilon_a,k_1(\varepsilon_a),k_2(\varepsilon_a),f_{x_0},k_3;x_0)\\
&:= \int_{x_0}^{x_0+\varepsilon_a} \hat{p}^{\mathrm{fp}}_{\sigma_a}(x;x_0,k_1(\varepsilon_a),k_2(\varepsilon_a))(f_{x_0}+k_3(x-x_0))dx\\
&= \left[\hat{P}^{\mathrm{fp}}_{\sigma_a}(x;x_0,k_1(\varepsilon_a),k_2(\varepsilon_a))(f_{x_0}+k_3(x-x_0))\right]_{x_0}^{x_0+\varepsilon_a}\\
&\quad - \int_{x_0}^{x_0+\varepsilon_a} \hat{P}^{\mathrm{fp}}_{\sigma_a}(x;x_0,k_1(\varepsilon_a),k_2(\varepsilon_a))\frac{d}{dx}(f_{x_0}+k_3(x-x_0))dx,
\end{aligned} \tag{67}$$

where $\hat{P}^{\mathrm{fp}}_{\sigma_a}(x;x_0,s,t) = \frac{s}{t}(1-\exp(-\frac{at}{2}(x-x_0)^2))$.

Note that

$$\int_0^x \exp(-t^2)dt = \frac{\sqrt{\pi}}{2}\mathrm{erf}(x). \tag{68}$$

For $\int_0^x \exp(-st^2)dt$, with $s>0$, using the change of variable $T = \sqrt{s}t$, we have

$$\int_0^x \exp(-st^2)dt = \int_0^{\sqrt{s}x} \exp(-T^2)\frac{1}{\sqrt{s}}dT = \sqrt{\frac{\pi}{4s}}\mathrm{erf}(\sqrt{s}x). \tag{69}$$

So,

$$\begin{aligned}
&\int_{x_0}^{x_0+\varepsilon_a} \hat{P}^{\mathrm{fp}}_{\sigma_a}(x; x_0, k_1(\varepsilon_a), k_2(\varepsilon_a)) \frac{d}{dx}(f_{x_0} + k_3(x - x_0))dx \\
&= \int_0^{\varepsilon_a} \frac{k_1(\varepsilon_a)}{k_2(\varepsilon_a)} k_3 dx - \int_0^{\varepsilon_a} \frac{k_1(\varepsilon_a)}{k_2(\varepsilon_a)} k_3 \exp\left(-\frac{ak_2(\varepsilon_a)}{2}x^2\right)dx \\
&= \frac{k_1(\varepsilon_a)k_3\varepsilon_a}{k_2(\varepsilon_a)} - \frac{k_1(\varepsilon_a)k_3}{k_2(\varepsilon_a)}\sqrt{\frac{\pi}{2ak_2(\varepsilon_a)}}\mathrm{erf}\left(\sqrt{\frac{ak_2(\varepsilon_a)}{2}}\varepsilon_a\right), \qquad (70)
\end{aligned}$$

and

$$\begin{aligned}
&\hat{V}_a(\varepsilon_a, k_1(\varepsilon_a), k_2(\varepsilon_a), f_{x_0}, k_3) \\
&= \frac{k_1(\varepsilon_a)}{k_2(\varepsilon_a)}\left(1 - \exp\left(-\frac{ak_2(\varepsilon_a)}{2}\varepsilon_a^2\right)\right)(f_{x_0} + k_3\varepsilon_a) - \frac{k_1(\varepsilon_a)k_3\varepsilon_a}{k_2(\varepsilon_a)} \\
&\quad + \frac{k_1(\varepsilon_a)k_3}{k_2(\varepsilon_a)}\sqrt{\frac{\pi}{2ak_2(\varepsilon_a)}}\mathrm{erf}\left(\sqrt{\frac{ak_2(\varepsilon_a)}{2}}\varepsilon_a\right). \qquad (71)
\end{aligned}$$

If we take δ from $(0, \frac{1}{2})$ (again, which aligns with the condition required for the case when computing the last term of (38)), and set $\varepsilon_a = a^{-\frac{1}{2}+\delta}$, then, $\lim_{a\to\infty}\varepsilon_a = 0$ and $\lim_{a\to\infty} a\varepsilon_a^2 = \lim_{a\to\infty} a^{2\delta} = \infty$, thus $\lim_{a\to\infty}(1 - \exp(-\frac{ak_2(\varepsilon_a)}{2}\varepsilon_a^2)) = 1$. So, we have

$$\begin{aligned}
&\lim_{a\to\infty} \hat{V}_a(\varepsilon_a, k_1(\varepsilon_a), k_2(\varepsilon_a), f_{x_0}, k_3; x_0) \\
&= \lim_{a\to\infty} \underbrace{\frac{k_1(\varepsilon_a)}{k_2(\varepsilon_a)}}_{\to 1}\underbrace{\left(1 - \exp\left(-\frac{ak_2(\varepsilon_a)}{2}\varepsilon_a^2\right)\right)}_{\to 1}\underbrace{(f_{x_0} + k_3\varepsilon_a)}_{\to f_{x_0}} \\
&\quad - \lim_{a\to\infty} \underbrace{\frac{k_1(\varepsilon_a)k_3\varepsilon_a}{k_2(\varepsilon_a)}}_{\to 0} + \lim_{a\to\infty} \underbrace{\frac{k_1(\varepsilon_a)k_3}{k_2(\varepsilon_a)}}_{\to k_3}\underbrace{\sqrt{\frac{\pi}{2ak_2(\varepsilon_a)}}}_{\to 0}\underbrace{\mathrm{erf}\left(\sqrt{\frac{ak_2(\varepsilon_a)}{2}}\varepsilon_a\right)}_{-1\le\ \ \le 1} \\
&= f_{x_0}. \qquad (72)
\end{aligned}$$

Likewise,

$$\int_{x_0}^{x_0+\varepsilon_a} p^{\mathrm{fp}}_{\sigma_a}(x; x_0) f(x)dx \le \hat{V}_a(\varepsilon_a, k_2(\varepsilon_a), k_1(\varepsilon_a), f_{x_0}, k_4; x_0), \qquad (73)$$

and the limit of the right hand side is

$$\lim_{a\to\infty} \hat{V}_a(\varepsilon_a, k_2(\varepsilon_a), k_1(\varepsilon_a), f_{x_0}, k_4; x_0) = f_{x_0}. \qquad (74)$$

Hence,

$$\lim_{a\to\infty} \int_{x_0}^{x_0+\varepsilon_a} p^{\mathrm{fp}}_{\sigma_a}(x; x_0) f(x)dx = f_{x_0}. \qquad (75)$$

□

Lemma 1. *For* $^\forall s > 0$ *and* $^\forall c > 0$

$$\lim_{a\to\infty} a\exp(-ca^s) = 0. \qquad (76)$$

Proof. Using the L'Hôpital's rule, we have

$$\lim_{a\to\infty} \frac{a}{\exp(ca^s)} \overset{\text{L'Hôpital's rule}}{=} \lim_{a\to\infty} \frac{1}{csa^{s-1}\exp(ca^s)}. \qquad (77)$$

If $s \ge 1$, this limit is clearly 0, and if $0 < s < 1$, we further have

$$\begin{aligned}
\lim_{a\to\infty} \frac{1}{csa^{s-1}\exp(ca^s)} &= \lim_{a\to\infty} \frac{a^{1-s}}{cs\exp(ca^s)} \\
&\overset{\text{L'Hôpital's rule}}{=} \lim_{a\to\infty} \frac{(1-s)a^{-s}}{c^2s^2a^{s-1}\exp(ca^s)} = \lim_{a\to\infty} \frac{(1-s)a^{1-2s}}{c^2s^2\exp(ca^s)}, \qquad (78)
\end{aligned}$$

which is 0 if $s \ge \frac{1}{2}$. For smaller and smaller s, we have

$$\begin{aligned}
&\lim_{a\to\infty} \frac{(1-s)a^{1-2s}}{a^2s^2\exp(ca^s)} \overset{\text{L'Hôpital's rule}}{=} \lim_{a\to\infty} \frac{(1-s)(1-2s)a^{-2s}}{c^3s^3a^{s-1}\exp(ca^s)} \\
&= \lim_{a\to\infty} \frac{(1-s)(1-2s)a^{1-3s}}{c^3s^3\exp(ca^s)} \\
&= \cdots = \lim_{a\to\infty} \frac{a^{1-ns}\prod_{k=1}^{n-1}(1-ks)}{c^ns^n\exp(ca^s)}, \qquad (79)
\end{aligned}$$

which is 0 for $s \ge \frac{1}{n}$. So, for an arbitrary $s > 0$, if we take $n = \lceil \frac{1}{s} \rceil$, then

$$\lim_{a\to\infty} a\exp(-ca^s) = \lim_{a\to\infty} \frac{a^{1-ns}\prod_{k=1}^{n-1}(1-ks)}{c^ns^n\exp(ca^s)} = 0. \qquad (80)$$

□

References

Jakub Fišer, Ondřej Jamriška, Michal Lukáč, Eli Shechtman, Paul Asente, Jingwan Lu, and Daniel Sýkora. 2016. StyLit: Illumination-Guided Example-Based Stylization of 3D Renderings. *ACM Transactions on Graphics* 35, 4 (Proc. of SIGGRAPH 2016), Article 92 (jul 2016), 11 pages. doi:10.1145/2897824.2925948

Golnaz Ghiasi, Honglak Lee, Manjunath Kudlur, Vincent Dumoulin, and Jonathon Shlens. 2017. Exploring the structure of a real-time, arbitrary neural artistic stylization network. arXiv:1705.06830 [cs.CV] https://arxiv.org/abs/1705.06830

Ron Goldman. 2005. Curvature formulas for implicit curves and surfaces. *Comput. Aided Geom. Des.* 22, 7 (oct 2005), 632–658.

Teng Hu, Ran Yi, Haokun Zhu, Liang Liu, Jinlong Peng, Yabiao Wang, Chengjie Wang, and Lizhuang Ma. 2023. Stroke-based Neural Painting and Stylization with Dynamically Predicted Painting Region. In *Proceedings of the 31st ACM International Conference on Multimedia* (Ottawa ON, Canada) *(MM '23)*. Association for Computing Machinery, New York, NY, USA, 7470–7480. doi:10.1145/3581783.3611766

Dmytro Kotovenko, Matthias Wright, Arthur Heimbrecht, and Bjorn Ommer. 2021. Rethinking Style Transfer: From Pixels to Parameterized Brushstrokes . In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Los Alamitos, CA, USA, 12191–12200. doi:10.1109/CVPR46437.2021.01202

Gongye Liu, Menghan Xia, Yong Zhang, Haoxin Chen, Jinbo Xing, Yibo Wang, Xintao Wang, Ying Shan, and Yujiu Yang. 2024. StyleCrafter: Taming Artistic Video Diffusion with Reference-Augmented Adapter Learning. *ACM Trans. Graph.* 43, 6, Article 251 (Nov. 2024), 10 pages. doi:10.1145/3687975

Ondřej Texler, David Futschik, Michal kučera, Ondřej jamriška, Šárka Sochorová, Menclei Chai, Sergey Tulyakov, and Daniel SÝkora. 2020. Interactive Video Stylization Using Few-Shot Patch-Based Training. *ACM Transactions on Graphics* 39, 4 (Proc. of SIGGRAPH 2020), Article 73 (jul 2020), 11 pages. doi:10.1145/3386569.3392453

Hideki Todo, Kunihiko Kobayashi, Jin Katsuragi, Haruna Shimotahira, Shizuo Kaji, and Yonghao Yue. 2022. Stroke Transfer: Example-based Synthesis of Animatable Stroke Styles. In *ACM SIGGRAPH 2022 Conference Proceedings* (Vancouver, BC, Canada) *(SIGGRAPH '22)*. Association for Computing Machinery, New York, NY, USA, Article 54, 10 pages. doi:10.1145/3528233.3530703

Yonghao Yue, Breannan Smith, Christopher Batty, Changxi Zheng, and Eitan Grinspun. 2015. Continuum Foam: A Material Point Method for Shear-Dependent Flows. *ACM Transactions on Graphics* 34, 5 (2015), 160:1–20.