Modeling and Rendering Glow Discharge—Supplementary Material

VENKATARAM SIVARAM, University of California San Diego, USA
RAVI RAMAMOORTHI, University of California San Diego, USA
TZU-MAO LI, University of California San Diego, USA

CCS Concepts: • **Computing methodologies → Volumetric models**; **Ray tracing**.

ACM Reference Format:
Venkataram Sivaram, Ravi Ramamoorthi, and Tzu-Mao Li. 2025. Modeling and Rendering Glow Discharge—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, 2 pages. https://doi.org/10.1145/3721238.3730674

1 Monte Carlo PDE Methods

Recent Monte Carlo PDE methods [Sawhney and Crane 2020; Sawhney et al. 2022] have demonstrated the possibility of solving certain classes of PDEs using grid-free methods. Specifically, these PDEs are of the form

$$\begin{aligned} \nabla \cdot (\alpha \nabla u) + \boldsymbol{\omega} \cdot \nabla u - \sigma u &= -f, \ \Omega \\ u &= g, \ \partial\Omega \end{aligned} \tag{1}$$

where $\alpha : \Omega \to \mathbb{R}$ and $\sigma : \Omega \to \mathbb{R}_{\geq 0}$ are scalar fields and $\boldsymbol{\omega}$ is a vector field that is the gradient of a scalar field.

Under time-invariant conditions, it is possible to write Equation 3a similarly to the above. Recall that, under time-invariance, this equation is:

$$0 = (\alpha - \eta)\|\vec{v}_E\|E - \beta EP - \nabla \cdot (E\vec{v}_E) + \nabla \cdot (\mathcal{D}\nabla E) \tag{2}$$

We take $\alpha, \beta, \eta, \mathcal{D}$ and P to be spatially varying coefficients concerning the primary variable E. It follows that:

$$\begin{aligned} 0 &= \nabla \cdot (\mathcal{D}\nabla E) + (\alpha - \eta)\|\vec{v}_E\|E - \beta EP - \nabla \cdot (E\vec{v}_E) \\ 0 &= \nabla \cdot (\mathcal{D}\nabla E) + [(\alpha - \eta)\|\vec{v}_E\| - \beta P]\, E - [\vec{v}_E \cdot \nabla E + (\nabla \cdot \vec{v}_E)E] \\ 0 &= x\nabla \cdot (\mathcal{D}\nabla E) - \vec{v}_E \cdot \nabla E + [(\alpha - \eta)\|\vec{v}_E\| - \beta P - (\nabla \cdot \vec{v}_E)]\, E \end{aligned}$$

The terms corresponding to the fields in Equation 1 shown below:

$$\alpha = \mathcal{D}, \quad \boldsymbol{\omega} = \vec{v}_E, \quad \sigma = (\nabla \cdot \vec{v}_E) + \beta P - (\alpha - \eta)\|\vec{v}_E\|$$

In our method, $\vec{v}_E$ is approximately a potential flow, which is compliant with the restrictions on $\boldsymbol{\omega}$. The core issue that obstructs the application of Monte Carlo PDE approaches to our problem lies in the σ coefficient. In Equation 1, it is required that σ is strictly non-negative. However, the corresponding term above does not necessarily adhere to this. When σ becomes negative, the PDE becomes a Helmholtz equation, for which there currently is no stable Monte Carlo-based solver.

Authors' Contact Information: Venkataram Sivaram, ves223@ucsd.edu, University of California San Diego, USA; Ravi Ramamoorthi, ravir@cs.ucsd.edu, University of California San Diego, USA; Tzu-Mao Li, tzli@ucsd.edu, University of California San Diego, USA.

2 Glow Discharge Strand Generation

Our strand-generation algorithm is built on the observation that the origins of Corona discharge filaments are concentrated where there are surface imperfections, i.e. sharp edges and scratches. At a high level, our pipeline evaluates a sharpness metric for each vertex, samples vertices by the corresponding sharpness distribution, and finally constructs strands extruding from the vertex and following its normal vector.

For each vertex $\boldsymbol{x}$, consider the set of face normals $\{\boldsymbol{n}^i\}$ in its one-ring. Our sharpness metric is measured as the sum of the variances along each coordinate of the normal vectors:

$$\mathcal{S}(\boldsymbol{v}) = \sum_{j=1}^{3} \mathrm{Var}(\{\boldsymbol{n}^i_j\}) \tag{3}$$

To avoid sampling flat regions, we mask out this sharpness value if it is less than the mean $\overline{\mathcal{S}(\boldsymbol{v})}$. Next, we normalize this distribution to get a discrete probability distribution over all vertices. For each sampled vertex $\boldsymbol{v}$, we generate a strand as a Bezier curve originating at $\boldsymbol{v}$ and extruding along the mean normal vector of the adjacent faces. To add diversity to the resulting curves, we jitter the control and end points of the curve. Figure 1 demonstrates the result of this process on a particular mesh.

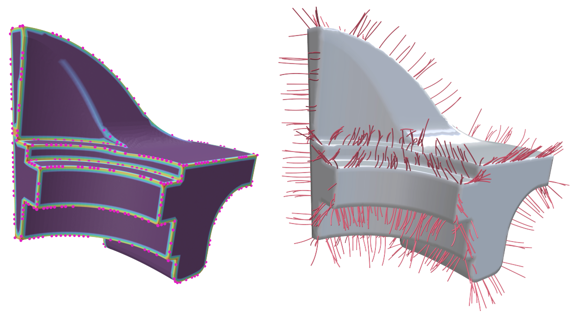

Fig. 1. **Generating glow discharge strands.** Our strand generation algorithm can reliably generate strands of glow discharge protruding from an arbitrary mesh. We demonstrate this above for the Fandisk mesh. On the left, we show a heatmap of the sharpness distribution over the mesh along with the corresponding point samples. On the right, visualize the completed strand; note that nearly all strands originate from a sharp edge. Scene adapted from Horse Statue 01 by Rico Cilliers ©Poly Haven

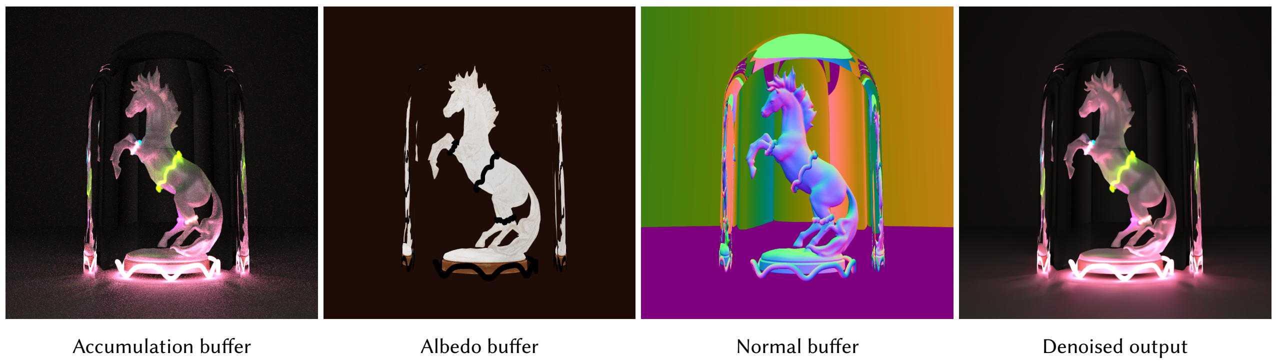

Fig. 2. **Denoising rendered outputs.** To improve the fidelity of our denoised outputs, we feed the denoiser with auxiliary albedo and normal buffers. In constrast with typical volumetric rendering settings, where assigning values to these buffers can be ambiguous, we use intersection tests with the active regions of glow discharge to locate and evaluate normal vectors. Note that we set the albedo value to a zero valued vector.

3 Denoising Rendered Images

We use the Intel Open Image Denoise [Áfra 2025] library to denoise our rendered outputs. As with typical denoiser interfaces, users are able to provide auxiliary buffers along with the base accumulation buffer to provide more context to the denoiser. However, in volumetric rendering, assigning values to auxiliary buffers, especially normal buffers, is ambiguous [Zhu et al. 2023]. Fortunately, in our case, this decision simplified by the active region in each glow discharge primitive. Specifically, for a given pixel, if the first intersection of the corresponding ray – not including the bounding box of the glow discharge primitives – is the active region defined by the primitives, then the normal vector at the active region is recorded in the normal buffer. In such cases, we also assign a zero valued vector to the albedo buffer. We show an example of this in Figure 2. Since our glow discharge primitives have active regions specified by level sets of Bezier curves, we use sphere tracing to pinpoint such intersections and then compute the appropriate normal vectors.

4 Alternative Numerical Integrators

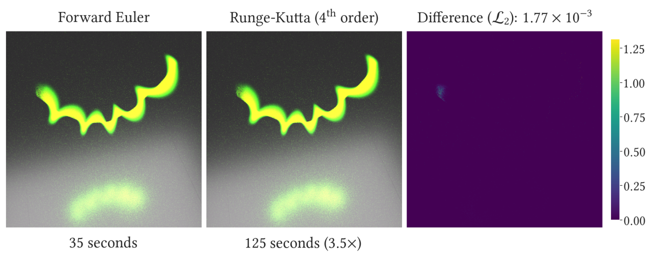

Fig. 3. **Using alternative numerical integrators.** We compare our implementation of Algorithm 1 using other numerical integration schemes in place of forward Euler. Above, we show the comparison between rendered results when using fourth order Runge-Kutta. While Runge-Kutta takes more than three times as long to render 1024 samples, the difference between these renderings is negligible.

While the numerical solver presented Algorithm 1 uses forward Euler integration, our method is agnostic to the specific numerical integration scheme used. In Figure 3, we provide a comparison between renderings using forward Euler and Runge-Kutta methods (fourth order). Ultimately, the differences in the rendered outputs between the different methods are minimal. However, the Runge-Kutta variant takes significantly longer to render, which is why we prefer using forward Euler.

References

Attila T. Áfra. 2025. Intel® Open Image Denoise. https://www.openimagedenoise.org.

Rohan Sawhney and Keenan Crane. 2020. Monte Carlo Geometry Processing: A Grid-Free Approach to PDE-Based Methods on Volumetric Domains. *ACM Trans. Graph.* 39, 4 (2020).

Rohan Sawhney, Dario Seyb, Wojciech Jarosz, and Keenan Crane. 2022. Grid-Free Monte Carlo for PDEs with Spatially Varying Coefficients. *ACM Trans. Graph.* XX, X (2022).

Shilin Zhu, Xianyao Zhang, Gerhard Röthlin, Marios Papas, and Mark Meyer. 2023. Denoising production volumetric rendering. In *ACM SIGGRAPH 2023 Talks*. 1–2.