

# Volume Scattering Probability Guiding

**Kehan Xu**<sup>1</sup>,  
Sebastian Herholz<sup>2</sup>,  
Marco Manzi<sup>3</sup>,  
Marios Papas<sup>3</sup>,  
Markus Gross<sup>1,3</sup>

<sup>1</sup>ETH Zürich, <sup>2</sup>Intel Corporation, <sup>3</sup>DisneyResearch|Studios

Sponsored by



Organized by



1

Thank you for the introduction.

This paper is a joint work between me, Sebastian Herholz, Marco Manzi, Marios Papas and Markus Gross.

And let's jump right in!



Gerda Arendt - Wikimedia Commons (CC BY-SA 2.0)

As we all know, volumetric effects can bring realism to the scenes.



Naturally we would like to render these effects, but adding volumes usually increases the noise and thereby render times.





Jungle, 32 SPP, Surface Only

Let's demonstrate this on a simple example.

Here we have a nice Jungle scene that almost converged after 32 samples per pixel.



Jungle, 32 SPP, With Volume

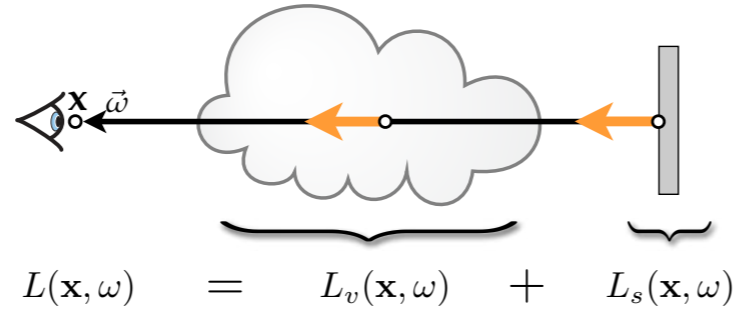
Adding a volume to it increases the noise of our rendering at the same number of samples.

The key insight we are going to show in this talk is that, a significant amount of this noise depends on the binary decision of whether a path scatters inside the volume or onto the (next) surface.



# Monte Carlo Estimator of the VRE

## Volume Rendering Equation



## Monte Carlo Estimator

$$\langle L(\mathbf{x}, \omega) \rangle = \begin{cases} \frac{1}{P_{\text{vol}}} \langle L_v(\mathbf{x}, \omega) \rangle \\ \frac{1}{1-P_{\text{vol}}} \langle L_s(\mathbf{x}, \omega) \rangle \end{cases}$$

$P_{\text{vol}}$ : the binary sample distribution between surface and volume

Figure source: Wojciech Jarosz

3-6 December 2024 Tokyo International Forum, Japan ASIA.SIGGRAPH.ORG/2024

Sponsored by   Organized by 

6

To understand this, let us first recap the basics of volume rendering.

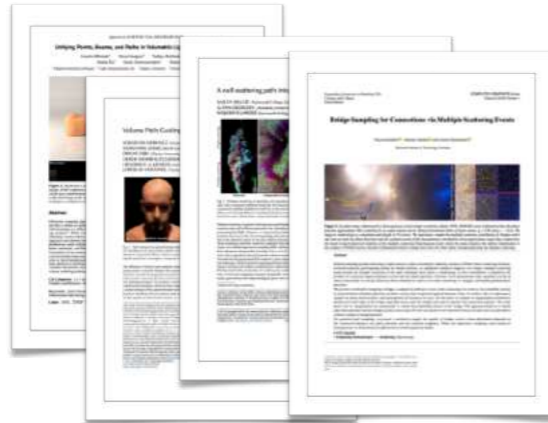
Our goal is to solve the volume rendering equation which is the sum of the volume contribution  $L_v$  and the (attenuated) surface contribution  $L_s$ .

In practice, we use a Monte Carlo path tracer which generates a path that either explores the surface or the volume component. This decision is made based on the volume scattering probability  $P_{\text{vol}}$ , as shown in the estimator.

# Many Works on Optimizing $\langle L_v \rangle$ and $\langle L_s \rangle$

$$\langle L_v(\mathbf{x}, \omega) \rangle$$

$$\langle L_s(\mathbf{x}, \omega) \rangle = \langle T_r(\mathbf{x}, \mathbf{x}_z) \rangle \langle L_o(\mathbf{x}, \omega) \rangle$$



3-6 December 2024 Tokyo International Forum, Japan ASIA.SIGGRAPH.ORG/2024

Sponsored by   Organized by 

A lot of the existing works have been focused on optimizing either the **nested** volume or surface contribution estimator.

# Many Works on Optimizing $\langle L_v \rangle$ and $\langle L_s \rangle$

$$\langle L(\mathbf{x}, \omega) \rangle = \begin{cases} \frac{1}{P_{\text{vol}}} \langle L_v(\mathbf{x}, \omega) \rangle \\ \frac{1}{1-P_{\text{vol}}} \langle L_s(\mathbf{x}, \omega) \rangle \end{cases}$$

$\langle L_o(\mathbf{x}, \omega) \rangle$



But not many previous works have been focused on optimizing the **volume scattering probability (VSP)** directly.



## Many Works on Optimizing $\langle L_v \rangle$ and $\langle L_s \rangle$

$$\langle L(\mathbf{x}, \omega) \rangle = \begin{cases} \frac{1}{P_{\text{vol}}} \langle L_v(\mathbf{x}, \omega) \rangle \\ \frac{1}{1 - P_{\text{vol}}} \langle L_s(\mathbf{x}, \omega) \rangle \end{cases}$$

### Delta Tracking

$$P_{\text{vol}}^{\Delta} = 1 - T_r(x, x_z)$$

This value is usually a byproduct of distance sampling, which in the case of delta tracking is equal to 1 - volume transmittance.

Since this decision is only based on local volume properties, this is often not optimal, and the VSP can become an important source of the estimator's variance.

# Many Works on Optimizing $\langle L_v \rangle$ and $\langle L_s \rangle$

$$\langle L(\mathbf{x}, \omega) \rangle = \begin{cases} \frac{1}{P_{\text{vol}}} \langle L_v(\mathbf{x}, \omega) \rangle \\ \frac{1}{1-P_{\text{vol}}} \langle L_s(\mathbf{x}, \omega) \rangle \end{cases}$$

$\langle L_o(\mathbf{x}, \omega) \rangle$

Our key insight is that directly importance sampling the VSP is actually simple and can effectively improve the volume rendering efficiency.



Let's demonstrate the effect of VSP guiding on our previous example.

Here we have 32 spp with the standard transmittance based sampling.

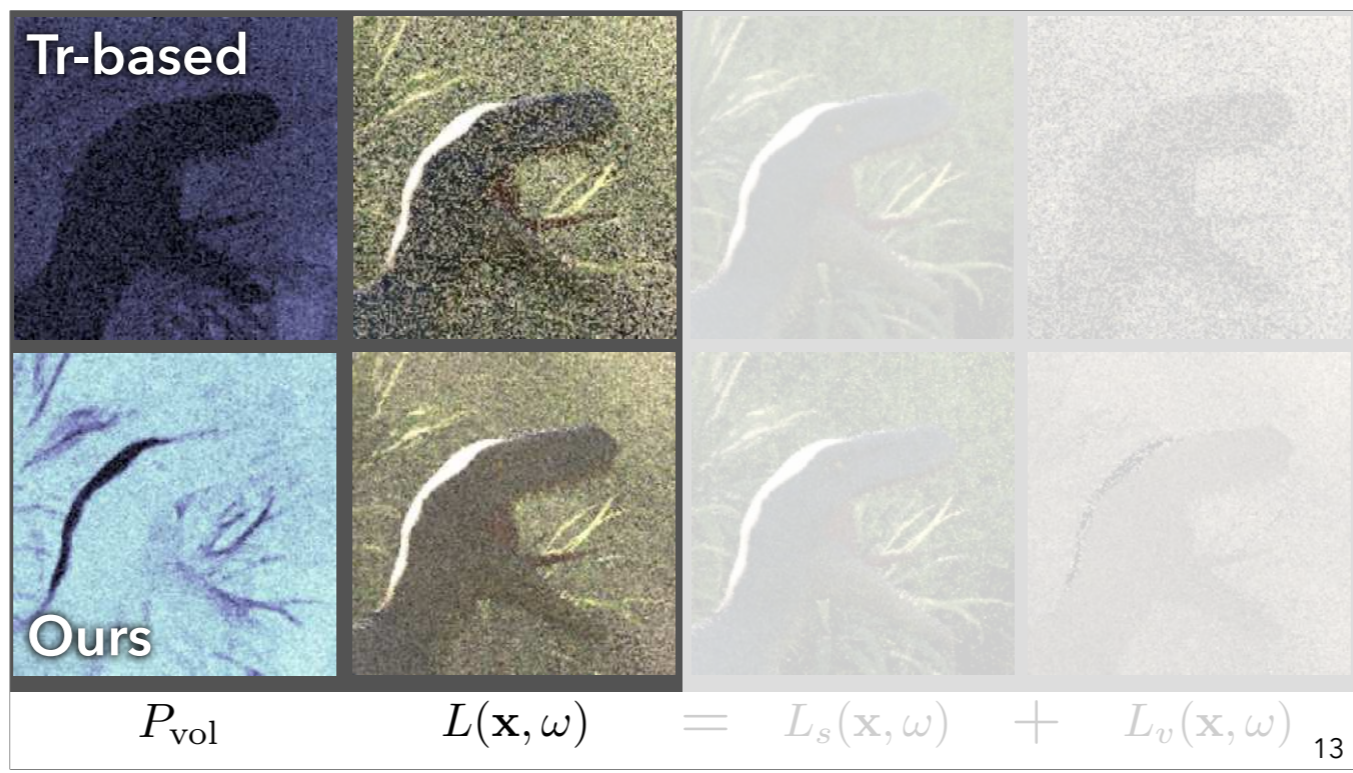




Jungle, 32 SPP, VSP Guiding (Ours)

And by simply guiding the VSP towards its optimum, this significantly decreases the noise level.



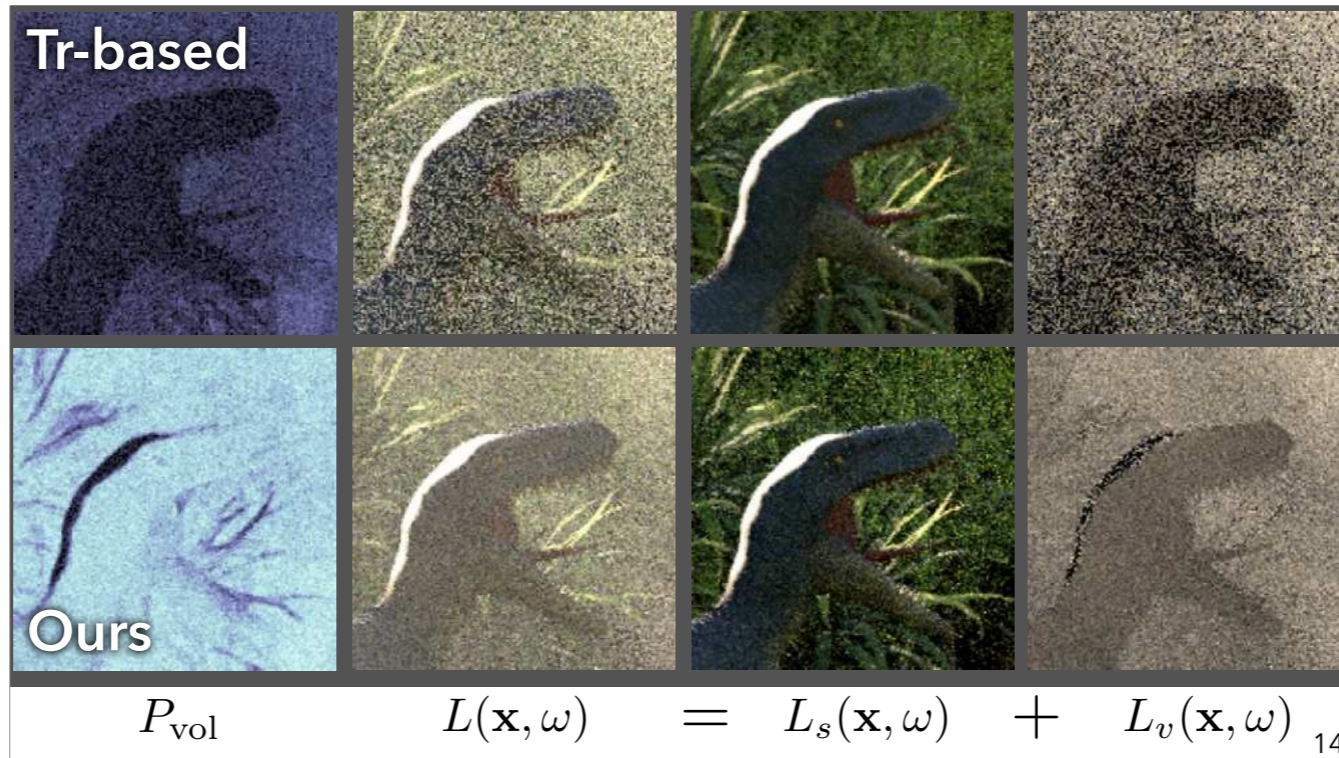


To give a better intuition on what is happening, the top row is transmittance-based sampling and the bottom row is our method.

The first column is a visualization of the VSP, where brighter values indicate higher VSP.

In this example, transmittance-based sampling is undersampling the volume. By increasing the ratio of volume samples, our method achieves noise reduction.

(32 spp)



If we look at separate surface and volume components:

In transmittance-based sampling, the surface component is almost nicely converged, while the volume component is still noisy.

Our method properly adjusts the VSP so that the surface and the volume contributions have a similar amount of noise. This indicates the workload has been distributed reasonably between surface and volume.

This is an example of increasing the VSP. Later in the evaluation section, we will show another example of decreasing the VSP.

# The Optimal VSP: Two Types

- Zero-Variance-based [Herholz et al. 2019]:
  - Assuming the nested estimators have **no variance**

$$P_{\text{vol}}^{\text{1st}} = \frac{\mathbb{E}[\langle L_v(x, \omega) \rangle]}{\mathbb{E}[\langle L_v(x, \omega) \rangle] + \mathbb{E}[\langle L_s(x, \omega) \rangle]}$$

- Variance-based [Rath et al. 2020]:
  - Considering the **variance** of the nested estimators

$$P_{\text{vol}}^{\text{2nd}} = \frac{\mathbb{E}[\langle L_v(x, \omega) \rangle^2]}{\mathbb{E}[\langle L_v(x, \omega) \rangle^2] + \mathbb{E}[\langle L_s(x, \omega) \rangle^2]}$$

To look into our method, the first question is naturally, what is the optimal VSP value.

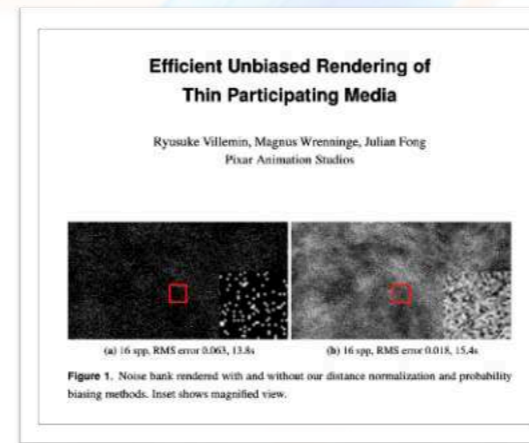
In our paper, we propose two types of optimal values:

The first one is based on the zero-variance theory, which assumes that the nested surface and volume estimators have no variance; the derived target value is based on the ratio of the first moments.

The second criterion takes the variance of the nested estimators into account, and the resulting optimal VSP is the ratio of the second moments.

# Normalized Distance Sampling

- Need to **manually** set the VSP per scene / per volume
- Only support increasing the VSP
- Not reaching the target VSP



Now with the optimal VSP, we would like to modify the distance sampling algorithm to reach this target value.

There's only one existing method we found that is directly comparable to ours, which is **normalized distance sampling (NDS)** by Villemin et al. This work requires manually setting the VSP, therefore they can only set this value per scene / per volume. Besides this, NDS also only supports increasing but not decreasing the VSP, and there is no guarantee that it will reach the target VSP.



- Homogeneous
- Heterogeneous

# Our Distance Sampling Method

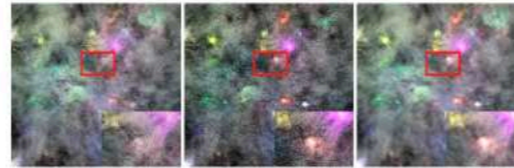
We present our own distance sampling algorithm that is capable of exactly achieving the target VSP.

For homogeneous volume, the method is simple, and we would like to refer you to the paper (we know the exact PDF and CDF, use the inversion method). In this talk, we focus on solving for the heterogeneous case.

# Delta Tracking as Resampling

Product Importance Sampling of the Volume Rendering Equation  
using Virtual Density Segments  
(Pixar Technical Memo 20-01)

MAGNUS WRENNINGE, Pixar Animation Studios  
RYUSUKE VILLEMEN, Pixar Animation Studios



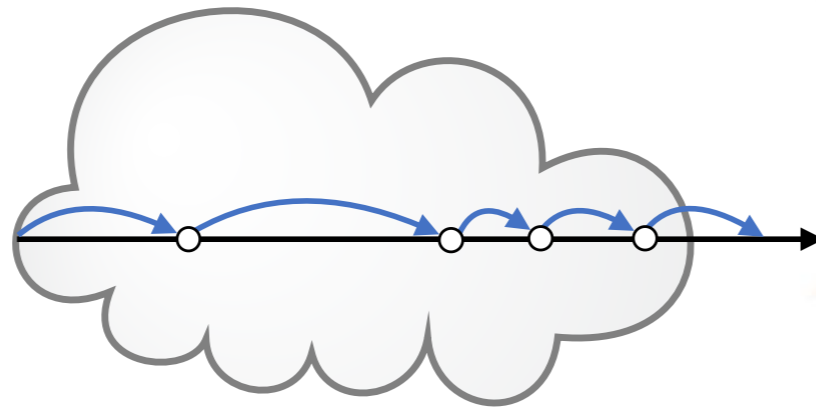
(a) Delta tracking SMAPE: 0.316  
(b) Equiangular sampling SMAPE: 0.535  
(c) Resampled product importance sampling SMAPE: 0.185

Fig. 1. Noise bark with 50 colored light sources at fixed render time of 1m per image. Delta tracking can resolve the heterogeneous density (left) and equiangular sampling improves areas around light sources (middle). Our resampled product importance sampling method (right) handles efficient sampling of both the density and light distributions, even in a many-light situation.



For this purpose, we utilize the insight from Wrennige & Villemin (2020) that delta tracking can be reinterpreted as a resampling process.

# Delta Tracking as Resampling



3-6 December 2024 Tokyo International Forum, Japan ASIA.SIGGRAPH.ORG/2024

Sponsored by

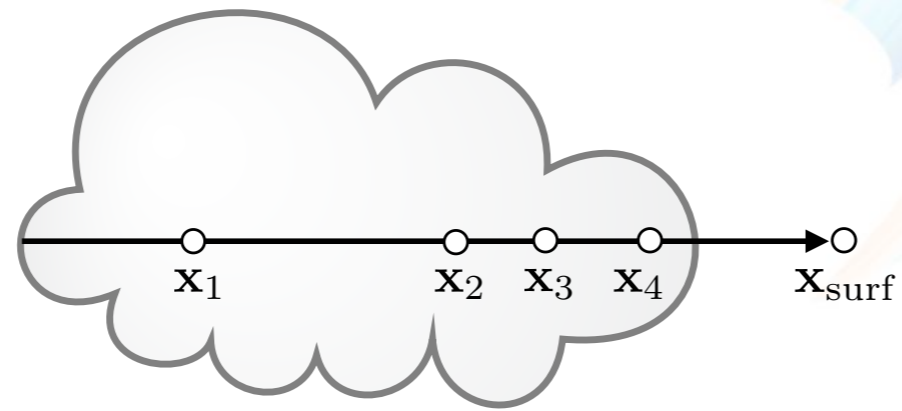


Organized by



The first step is the same as in ratio tracking:  
We do continuous distance sampling until passing the volume boundary.

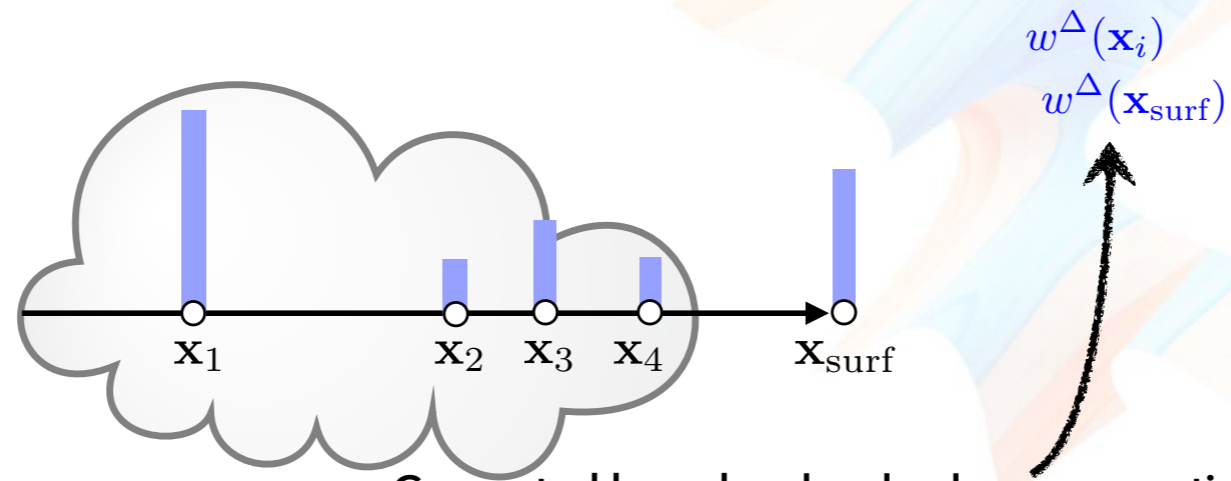
# Candidate Samples



The samples we generate inside the volume, here  $x_1$ - $x_4$ , and an addition surface sample  $x_{surf}$ , are used as the candidate samples for resampling.




# Resampling Weights for Delta Tracking



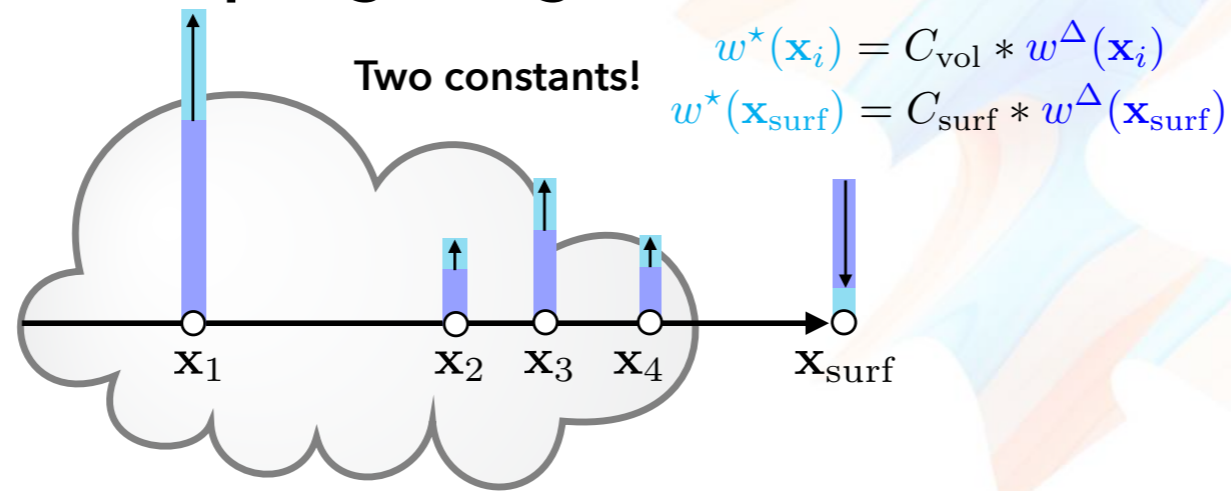
Computed based on local volume properties  
(details in the paper)

3-6 December 2024 Tokyo International Forum, Japan ASIA.SIGGRAPH.ORG/2024

Sponsored by  Organized by 

Omitting many details, we simply state that resampling weights can be computed based on the local volume properties of the generated candidates - this leads to the same sample distribution as with delta tracking.

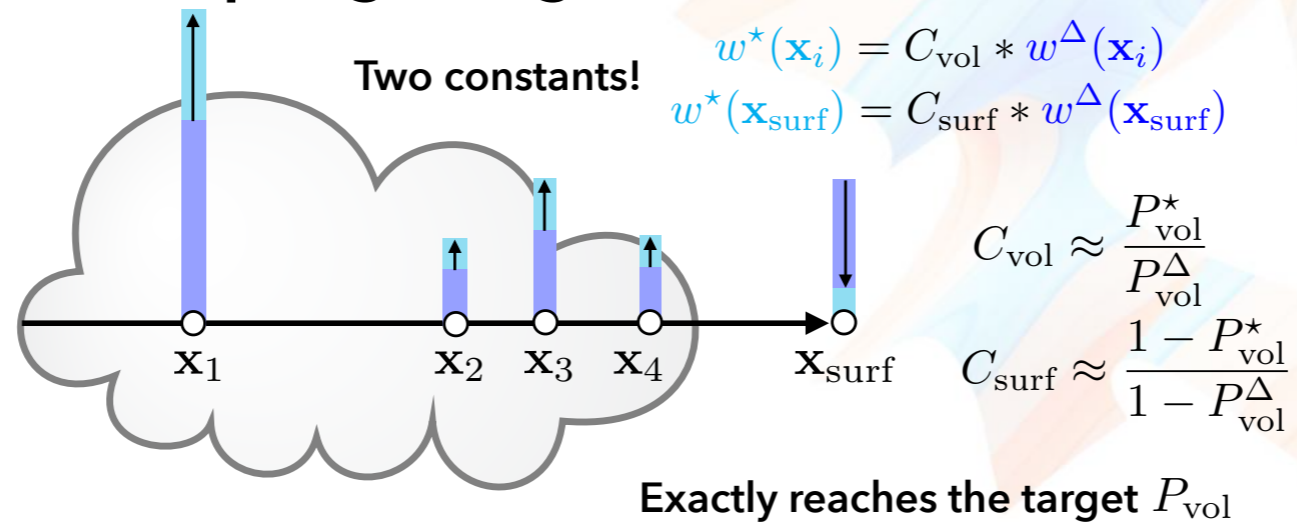
# Resampling Weights for Our Method



Extending this to our method, the high-level idea is pretty clear.

We simply apply two constants, one to all the volume resampling weights ( $C_{\text{vol}}$ ), one to the only one surface resampling weight ( $C_{\text{surf}}$ ).

# Resampling Weights for Our Method



These two scaling factors are approximately the ratio of the volume or surface scattering probability of our method to that of transmittance-based sampling.

And again, these two values can be computed only based on the local volume properties of the candidates. And our algorithm enables exactly reaching the target VSP.

# Resampling Weights for Our Method

## Reservoir Sampling :)



$$\mathbb{P}[\text{Accept } i] = \frac{w_i}{\sum_{j \leq i} w_j}$$

$$\frac{P_{\text{vol}}^*}{P_{\text{vol}}^\Delta} \frac{1 - P_{\text{vol}}^*}{1 - P_{\text{vol}}^\Delta}$$

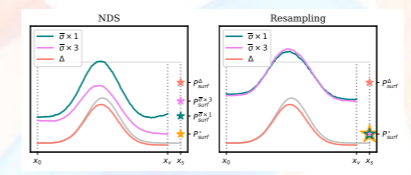
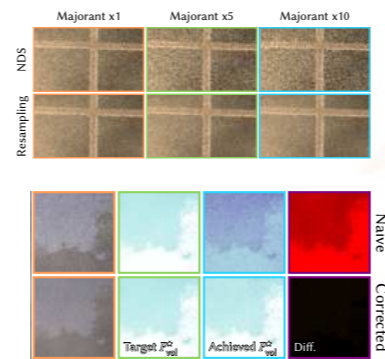
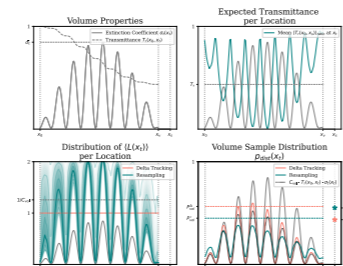
Exactly reaches the target  $P_{\text{vol}}$

To make the whole process memory efficient, we use reservoir sampling.



# Details in the Paper!

- Zero volume event candidate
- Defensive resampling
- Volume sample distribution analysis
- Increase majorant
- ...



```

ALGORITHM 1: Volume Scattering Probability Guiding
1 Function VSP(G, t_e, P_vol^0, sigma):
2   Reservoir
3   f ← 0, (T_e)_min ← 1, w_max ← 0
4   P_vol^1 ← ZeroVolumeCandidateCompensation(G, t_e, P_vol^0, sigma)
5   while true do
6     f ← f + (1 - f) * (1 - sigma) // Distance sampling, Eq. 11
7     x_i ← 2 * f * t_e // Generate a volume candidate
8     if f >= t_e then
9       break
10    (T_e)_min ← P_vol(x_i)(T_e)_min
11    w_max ← w_max + w^2(x_i) // Eq. 21
12    resample(x_i, w^2(x_i))
13  end
14  x_sur ← x + t_e * t_e // Generate the surface candidate
15  /* Defensive resampling */
16  w_sur ← w(1 - P_vol^1) + (1 - w) * w_max // Eq. 27
17  w^2(x_sur) ← w_sur^2 + (1 - sigma) * w^2(x_sur) // Eq. 22, Eq. 27
18  w_sur ← w_sur + w^2(x_sur) // w_sur = 1
19  resample(x_sur, w_sur^2(x_sur))
20  /* Set path segment throughput */
21  P_vol^1 ← w_sur^2 + (1 - sigma) * (1 - (T_e)_min) // Resulting VSP
22  P_vol^2 ← (1 - sigma) * (1 - (T_e)_min) // Eq. 28
23  return P_vol^1
24 Function ZeroVolumeCandidateCompensation(G, t_e, P_vol^0, sigma):
25   if P_vol^0 <= max(x_i, -ln(1 - P_vol^0)/t_e) // Eq. 25
26     P_vol^1 ← P_vol^0 / (1 - exp(-sigma * t_e)) // Eq. 26
27   return P_vol^1
    
```

While the high-level idea might seem very straightforward, but to make the algorithm practical, a lot of nitty-gritty details are involved. If you're interested, we would like to refer you to the paper.

# Our VSP Guiding Framework

- Structures to query the optimal  $P_{vol}$  for every path segment
- Incremental training during rendering



Figure source: Wojciech Jarosz

Finally, to integrate our algorithm into a render, we developed our VSP guiding framework that:

- allows us to query the optimal VSP for every path segment,
- and learns all the important quantities incrementally during rendering (i.e., no pre-processing needed).

# Our VSP Guiding Framework

Primary rays:

Auxiliary image space buffer

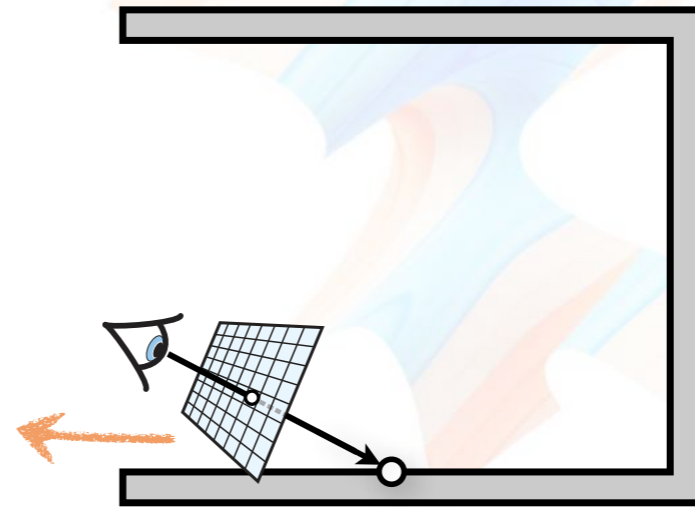


Figure source: Wojciech Jarosz

For primary rays, we use a screen-space buffer data structure to store the estimates of VSP.

# Our VSP Guiding Framework

Secondary rays:

5D spatial-directional data structure

**Piggyback existing data structures**

**(e.g., from path guiding)!**

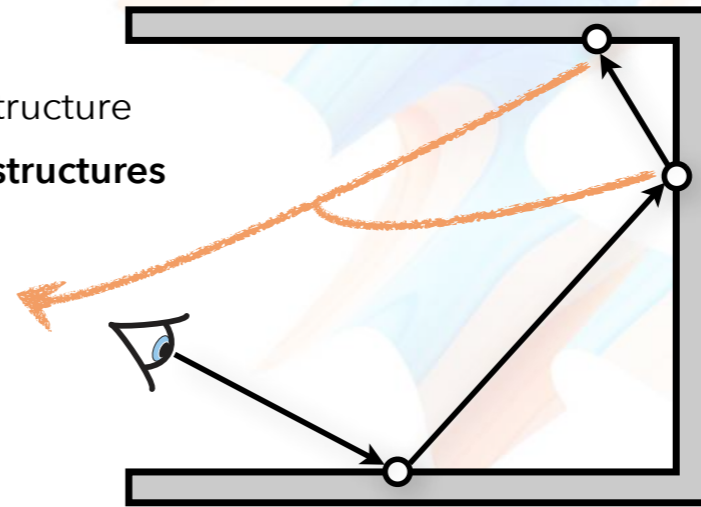
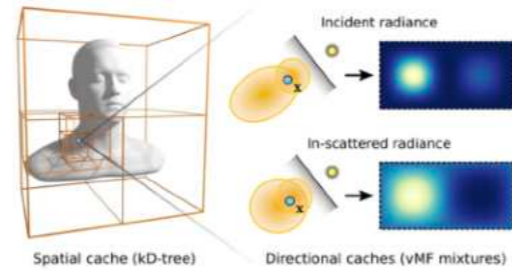


Figure source: Wojciech Jarosz

For secondary rays, we can piggyback existing 5D spatial-directional data structures (for example, the ones used for path guiding), as we actually did.



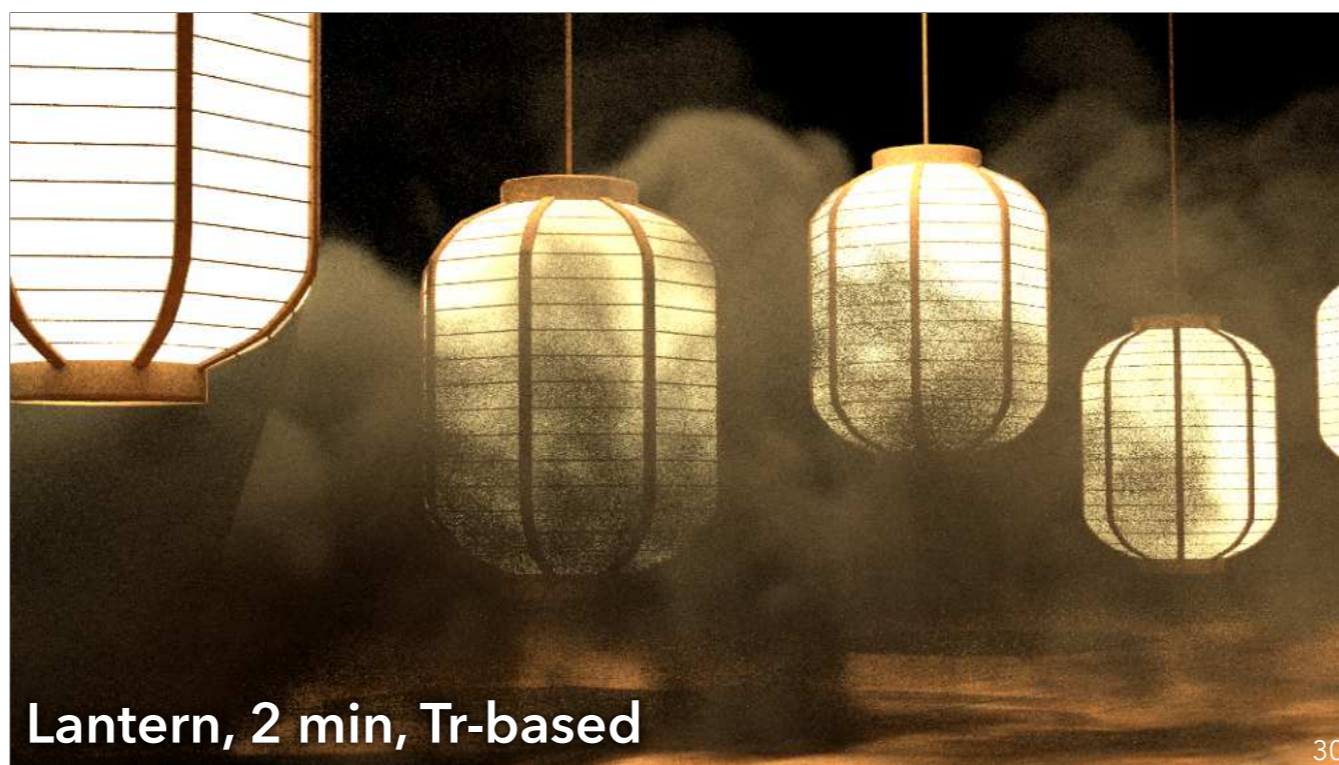
## Evaluation

- Equal time
- Directional guiding enabled
- Distance sampling methods:
  - Transmittance-based  $P_{vol}$
  - The VSP Guiding framework (**Ours**):
    - NDS
    - Resampling (**Ours**)
      - MIS: 0.75

Now let's look at some evaluations. All evaluations are equal-time, with directional guiding always enabled.

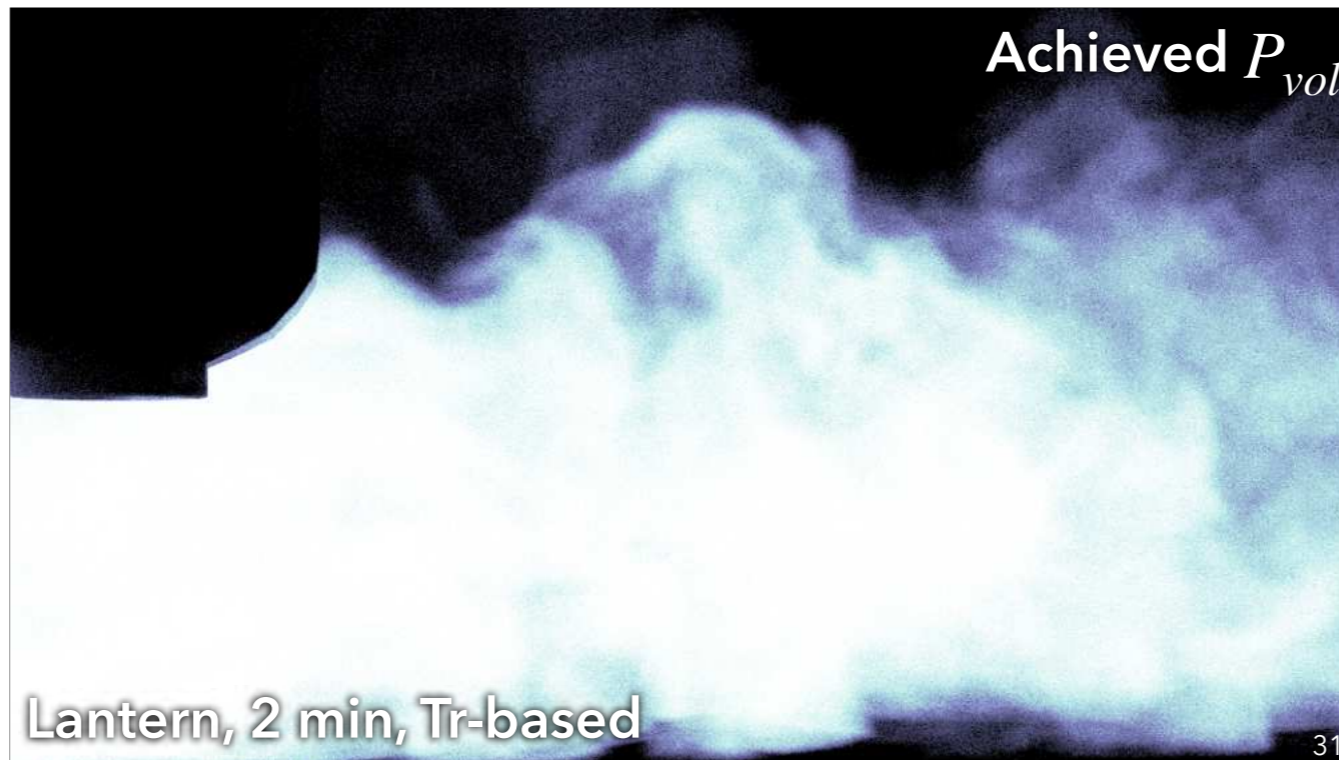
We compare between three distance sampling methods:

- Transmittance-based sampling
- The other two methods utilize the optimal VSP value derived from the VSPG framework, these two methods are:
  - 1. Normalized distance sampling (NDS)
  - 2. Our resampling method
  - Both methods uses 0.75 MIS with tr-based sampling as a kind of defensive scheme

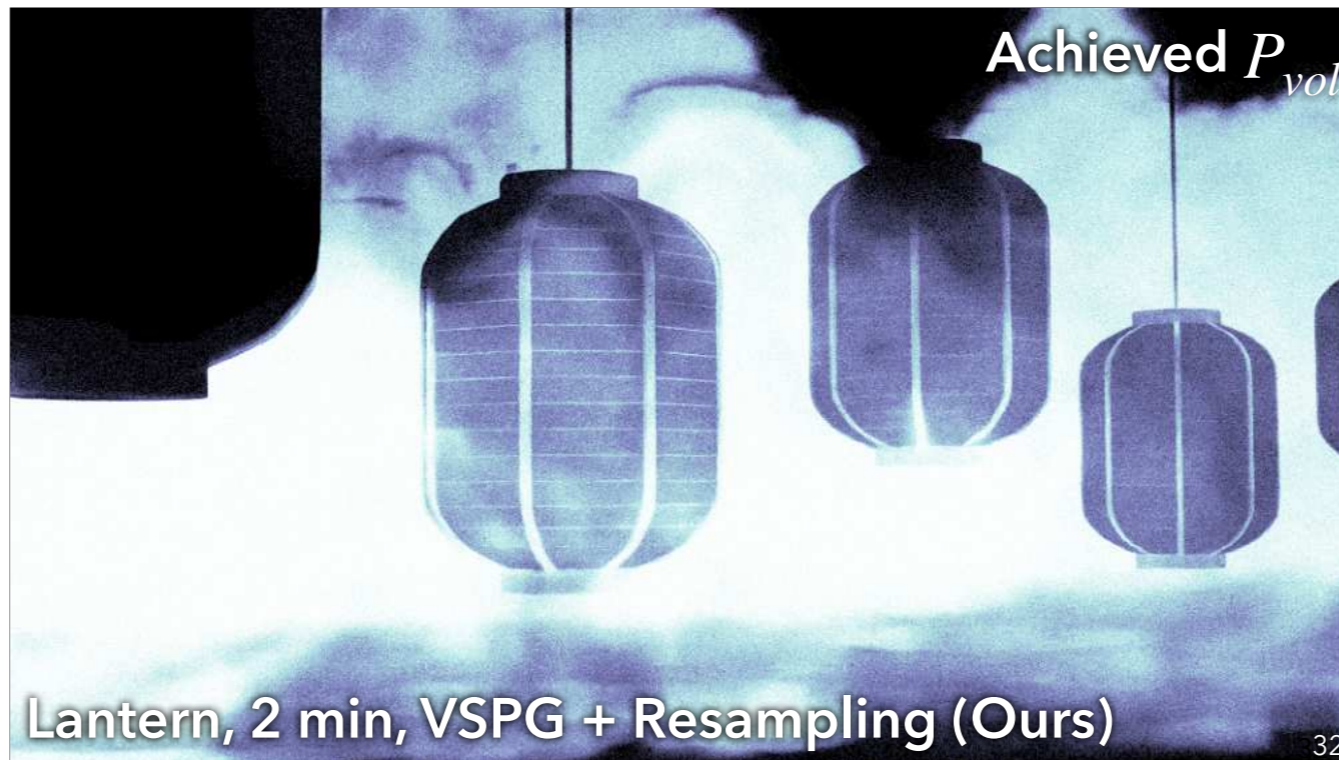


This is a scene of lanterns rendered with transmittance-based sampling

(47 spp)



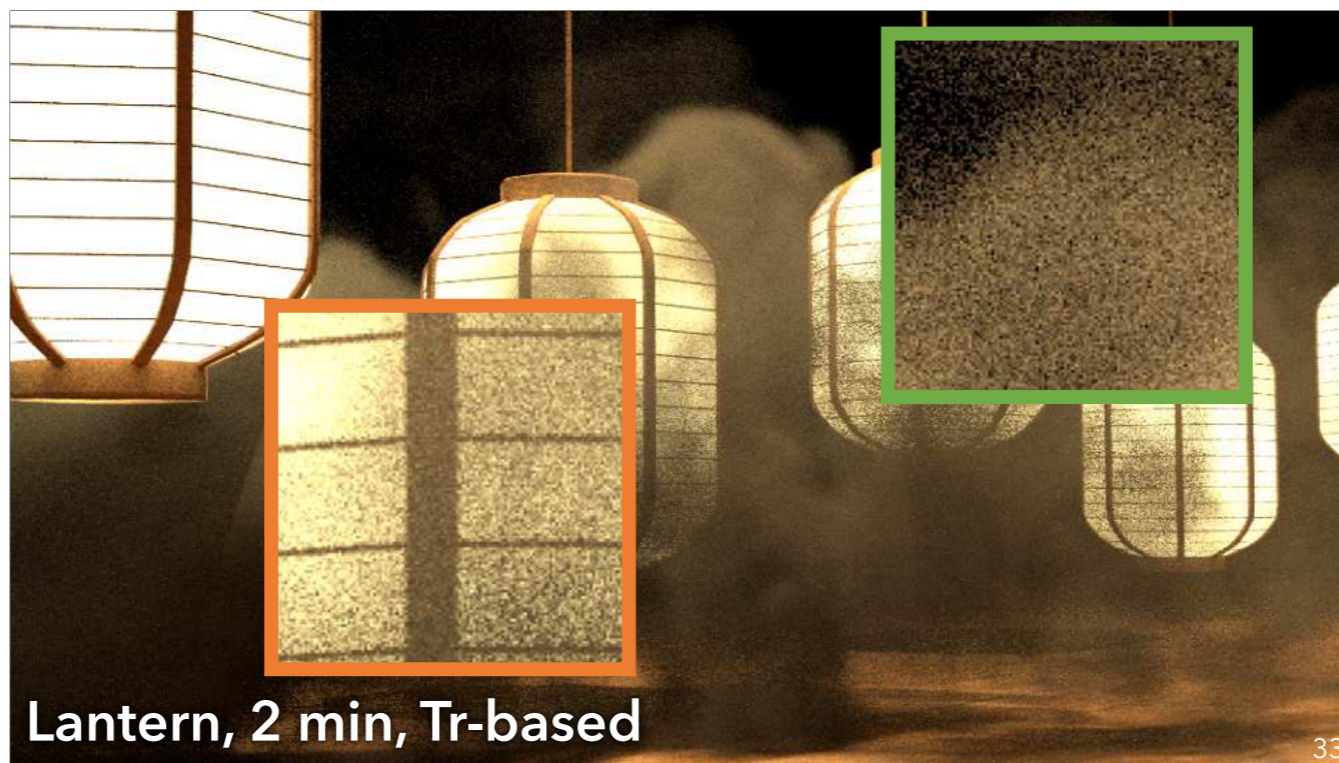
Here is the visualization of the achieved VSP. The achieved value is quite high because we have a thick media.



Comparing this with our method:  
In regions of lanterns, we're decreasing the VSP.  
In some regions of the cloud, we're increasing the VSP.

If you flip two images back and forth, you can see the difference.

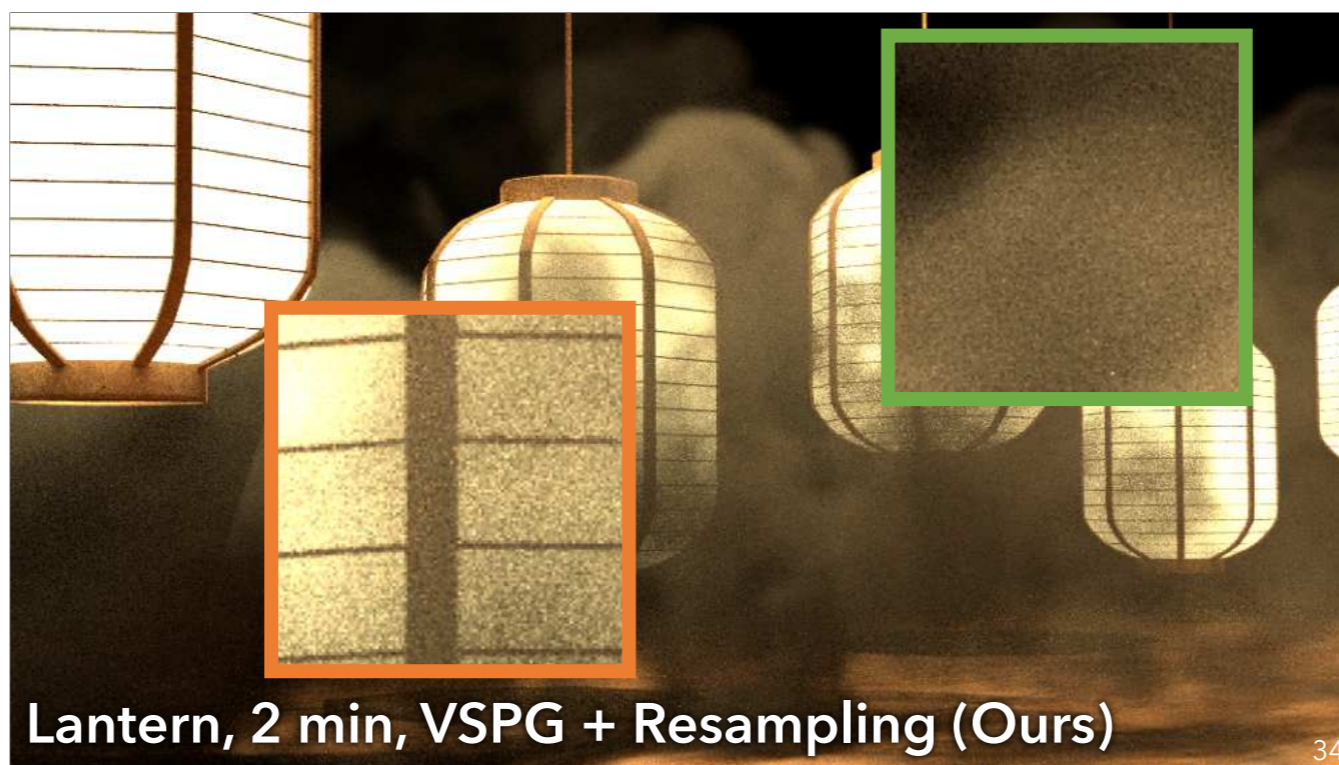




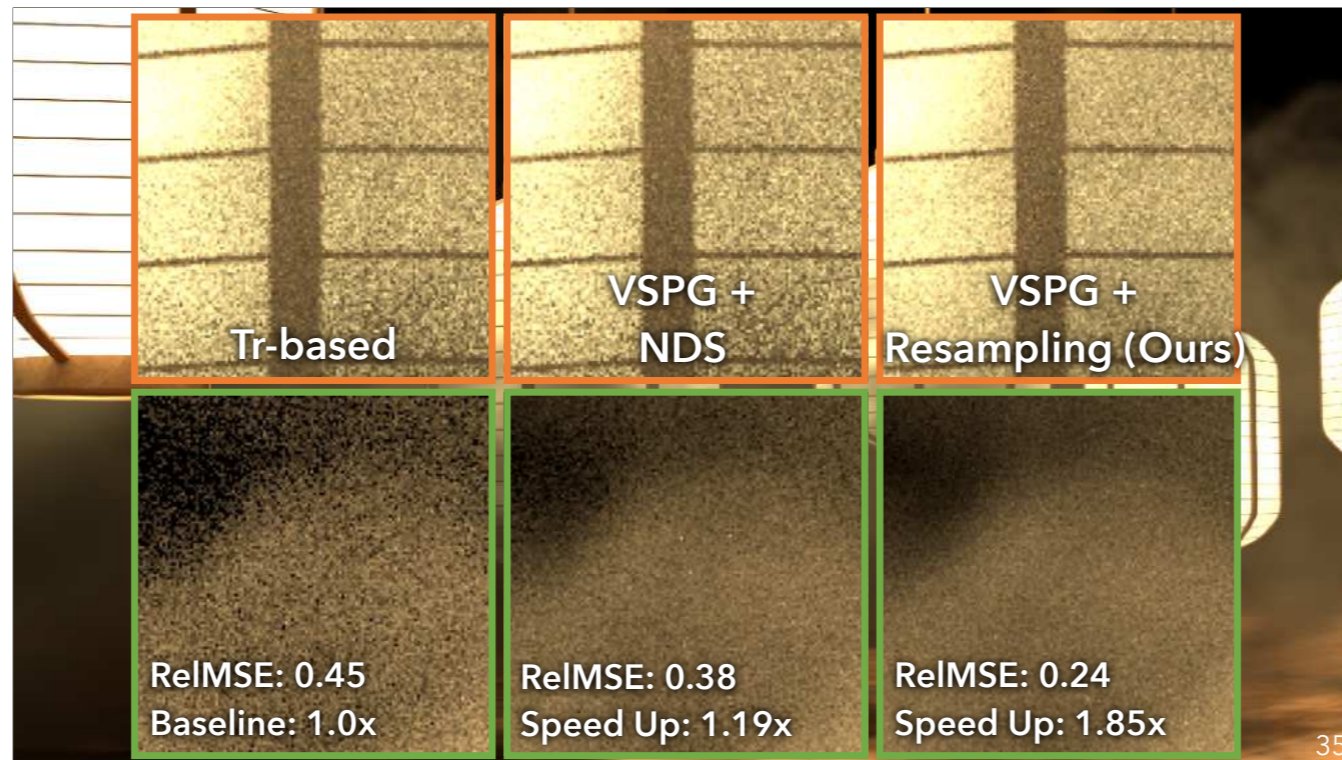
Lantern, 2 min, Tr-based

Here is the comparison of the renderings, and the difference is remarkable.

(47 spp)



(46 spp)

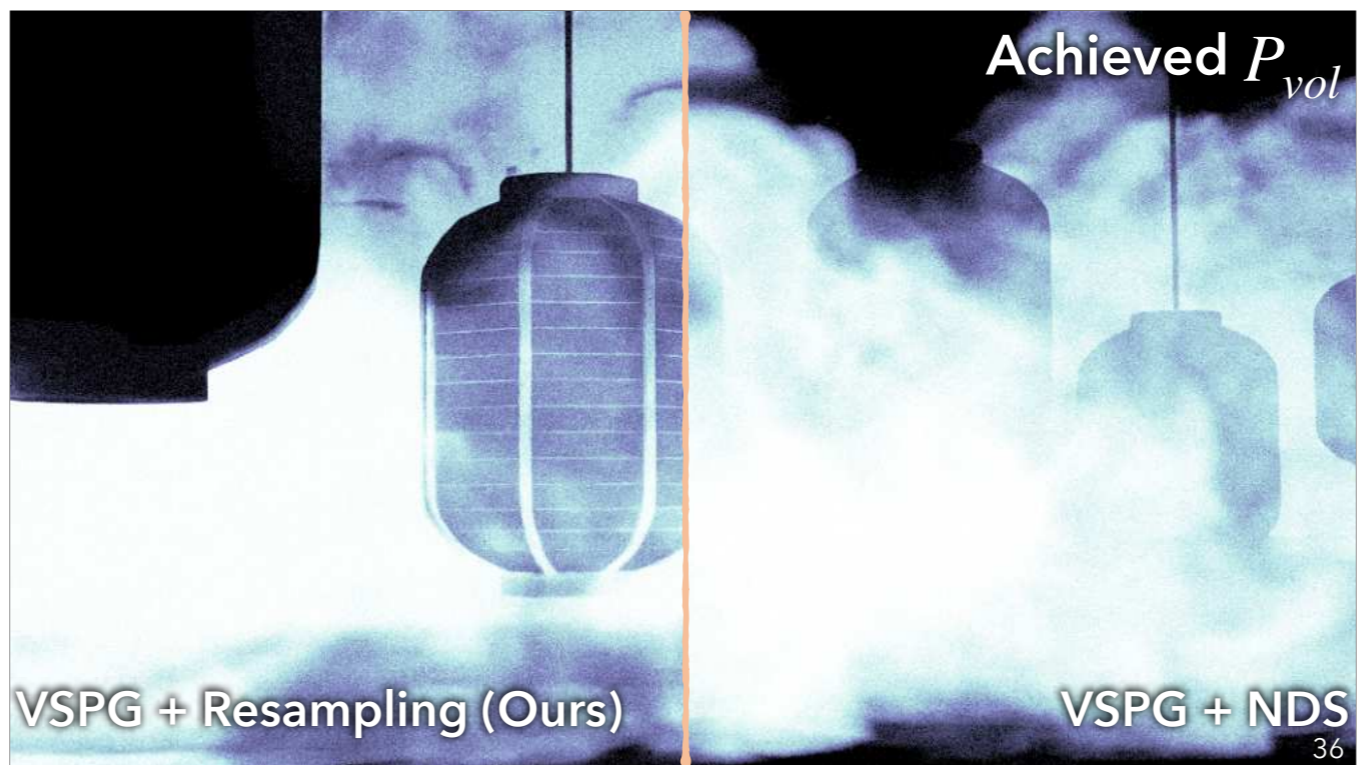


Here we see a side-by-side comparison of the insets. In the middle, we add an additional column to compare with NDS.

In the top row, we want the VSP to decrease, and NDS is not helpful because it does not support VSP decrease.

In the bottom row, even where we want VSP increase, NDS does not behave as well as ours because, though given the same target VSP, there is no guarantee that NDS reaches this target, while our methods always do.





Here is a comparison of the achieved VSP of ours vs. NDS.  
We can clearly see the difference in both the Lantern and the cloud areas (on the top).





We've tested our VSPG framework on an extensive set of scenes, and achieved an average speedup of more than two times by simply modifying the quantity of VSP.

If you're interested in seeing more results, you're welcomed to find me at the interactive discussion session.

# Summary

- **Key insight:** explicitly controlling  $P_{vol}$  can improve efficiency
- A practical framework for:
  1. Computing the optimal  $P_{vol}$
  2. Achieve precise control over  $P_{vol}$

- ✓ Unbiased
- ✓ Easy to implement & minimal overhead
  - Perfect combination with directional guiding!
- ✓ Fully automatic, no user parameter

To summarize, our key insight is that explicitly controlling the VSP can improve the volume rendering efficiency.

We provide a practical framework for both computing the optimal VSP and achieving precise control over this value.

Our framework is unbiased; easy to implement and has minimal overhead when combined with directional guiding; it is fully automatic, and no user parameter is required.

## Thank you!



intel.  
**OPENPGL**

- VSPG framework included in upcoming v0.8.0 release



**Project Page**

- Interactive viewer
- PBRT source code (soon)

That concludes my talk, thank you all for listening!

The VSP Guiding framework will be included in the upcoming release of the Intel Open Path Guiding Library, and here is the QR code of the project webpage, where we provide an interactive viewer. The implementation in PBRT V4 will be available soon.