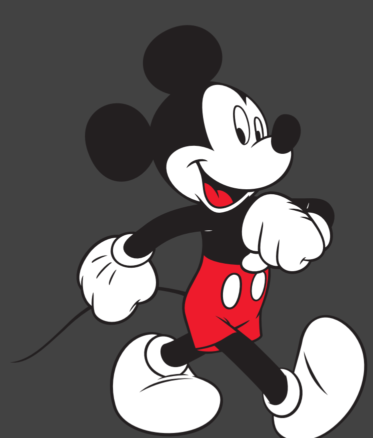
Progressive Expectation–Maximization for Hierarchical Volumetric Photon Mapping

Wenzel Jakob[1,2] Christian Regg[1,3] Wojciech Jarosz[1]

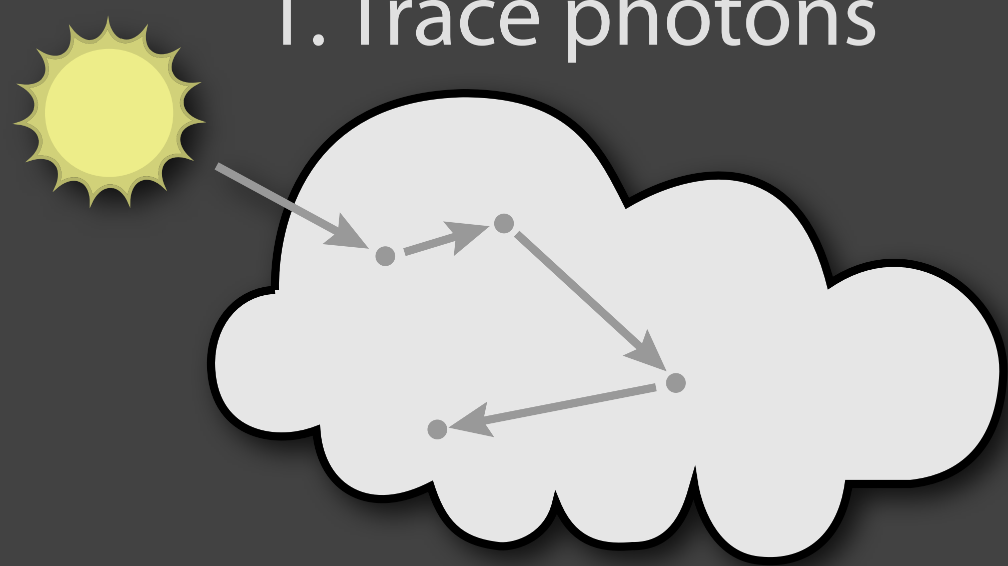


[1] Disney Research, Zürich
[2] Cornell University
[3] ETH Zürich

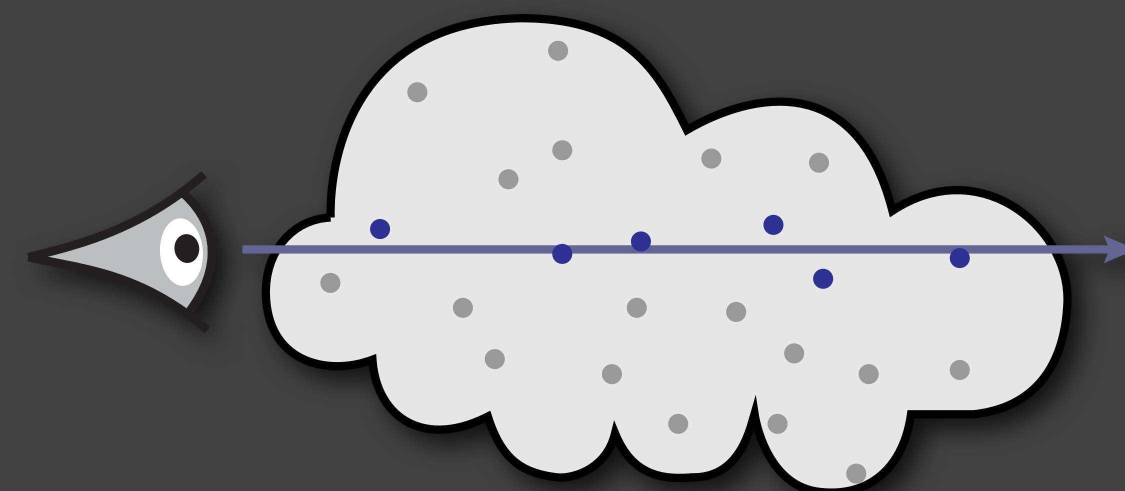
Motivation

Volumetric photon mapping

1. Trace photons



2. Radiance estimate



Issues

- high-frequency illumination requires *many* photons
- time spent on photons that contribute very little
- prone to temporal flickering

Motivation

Beam radiance estimate : 917K photons

Per-pixel render time

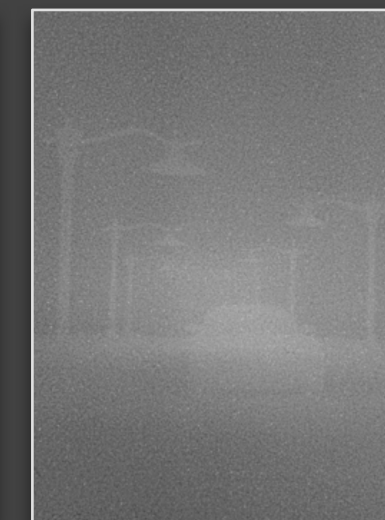
Motivation

Beam radiance estimate : 917K photons | **Our method: 4K Gaussians**

Render time: 281 s

Per-pixel render time

Render time: 125 s

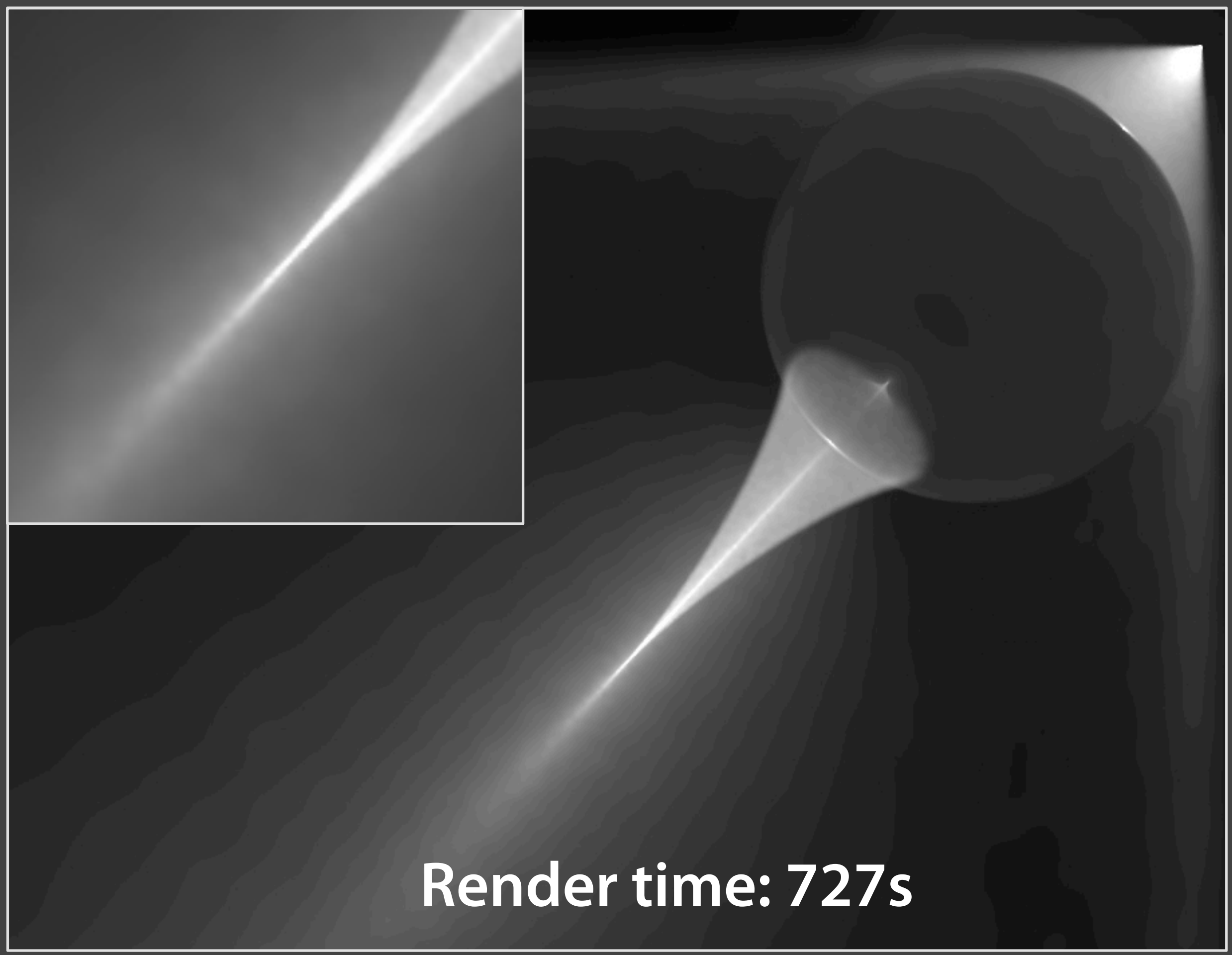


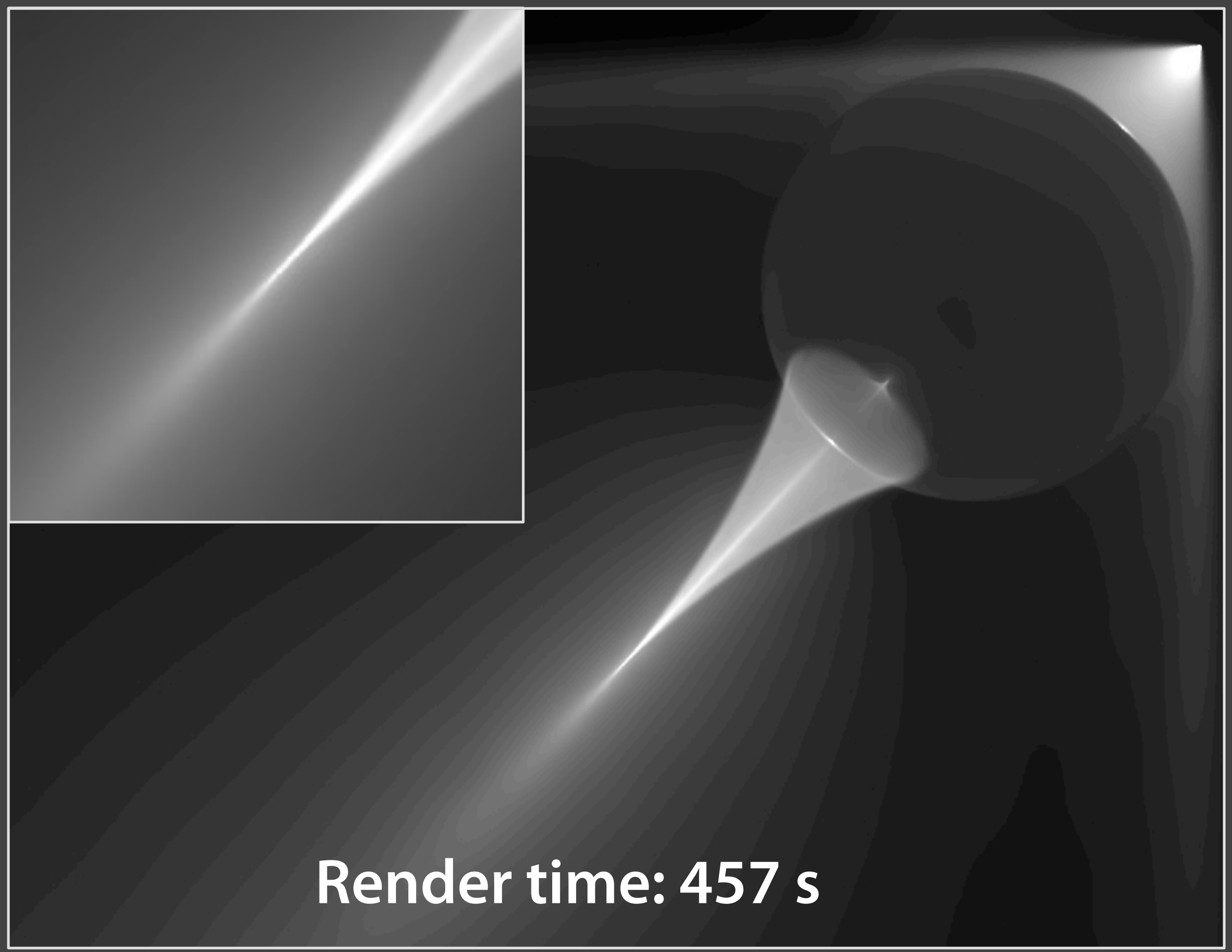
Per-pixel render time

Our approach:

- represent radiance using a Gaussian mixture model (**GMM**)
- fit using progressive expectation maximization (**EM**)
- render with multiple levels of detail

Motivation



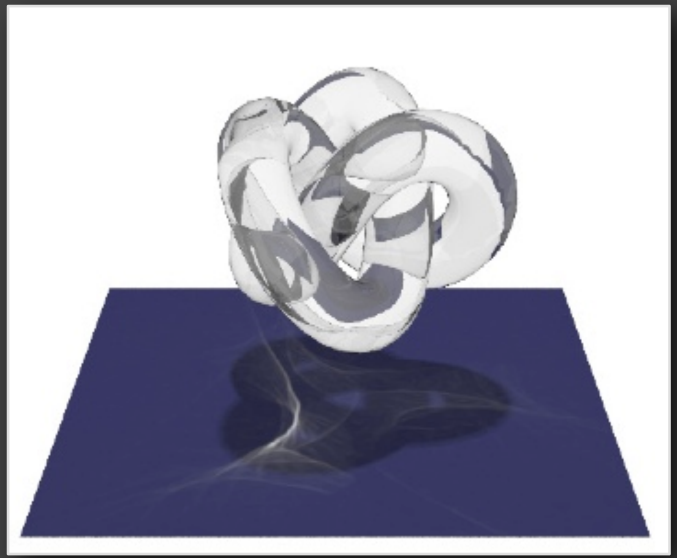


Our approach:

- represent radiance using a Gaussian mixture model (**GMM**)
- fit using progressive expectation maximization (**EM**)
- render with multiple levels of detail

Related work

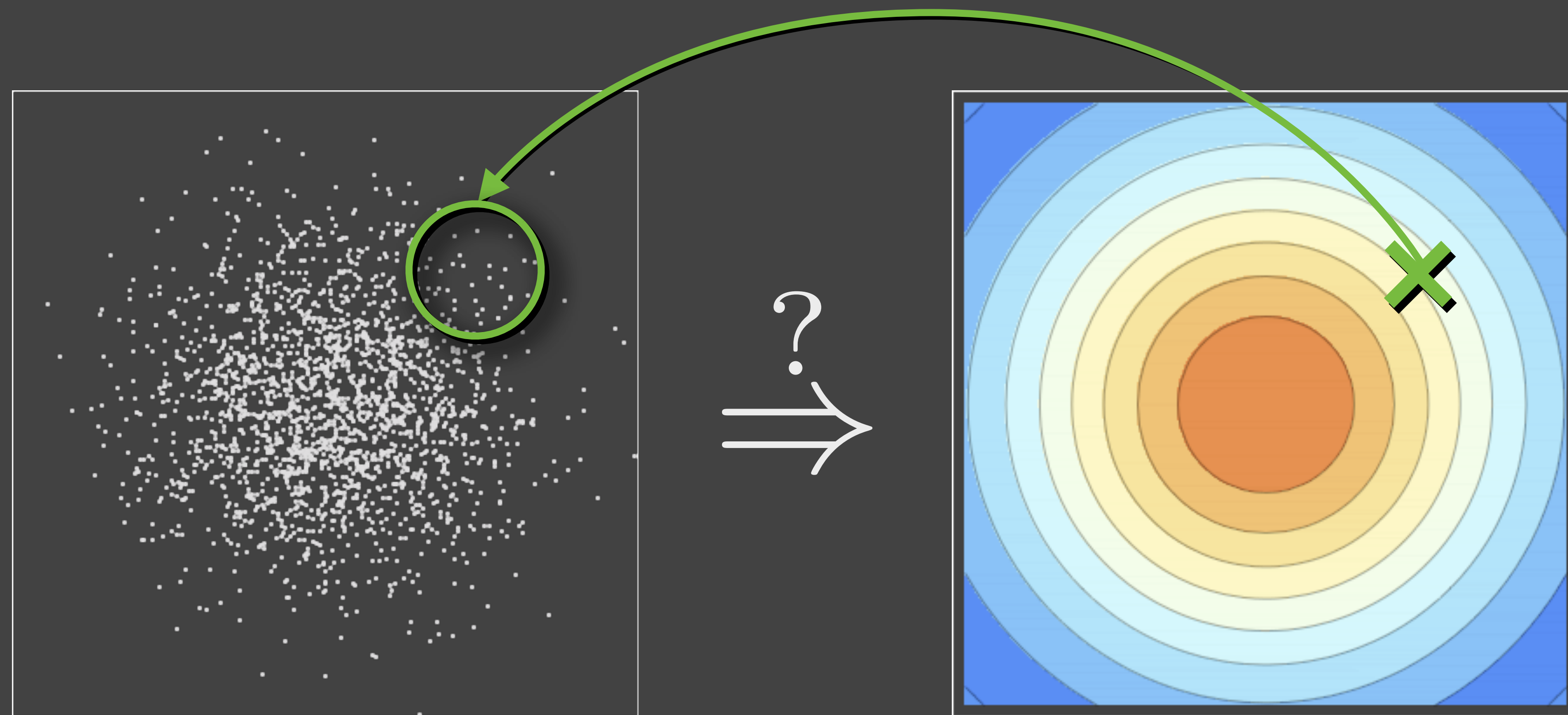
- Diffusion based photon mapping
  [Schjøth et al. 08]



- Photon relaxation
  [Spencer et al. 09]

- Hierarchical photon mapping
  [Spencer et al. 09]

Density estimation



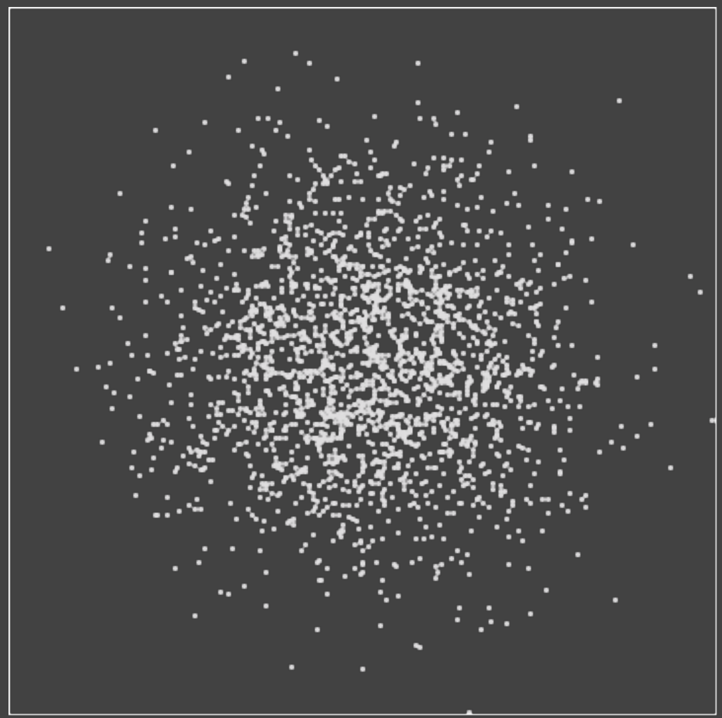
Given photons $x_1, x_2, \ldots$

approximately determine their density f

Nonparametric:

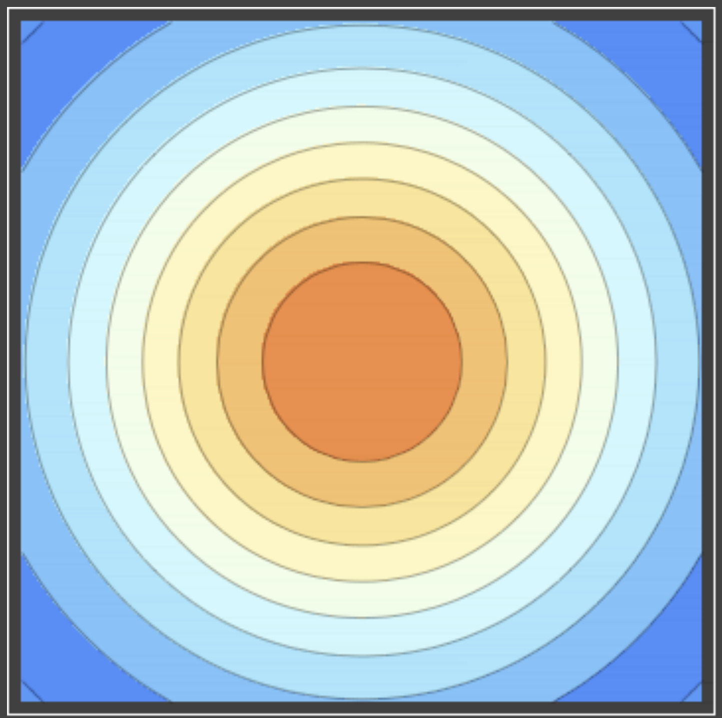
- Count the number of photons within a small region

Density estimation



Given photons

$x_1, x_2, \ldots$



approximately determine their density f

Nonparametric:

- Count the number of photons within a small region

Parametric:

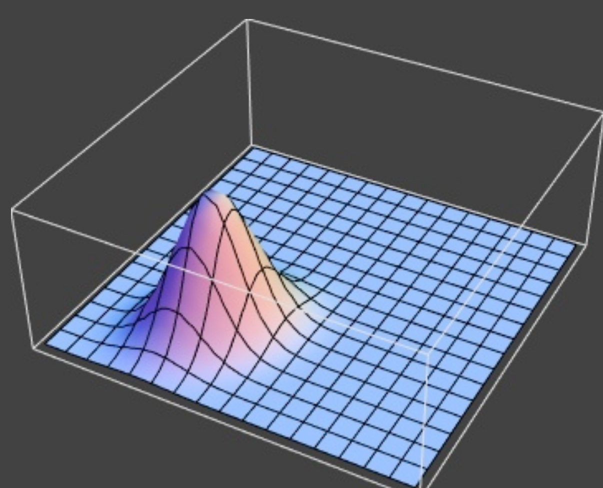
- Find suitable parameters for a **known** distribution

Gaussian mixture models

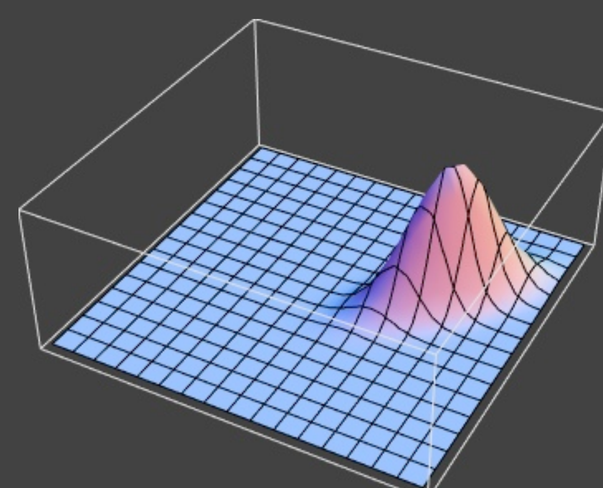
- Photon density modeled as a weighted sum of Gaussians:

$$f(\mathbf{x} \mid \Theta) = \sum_{i=1}^{k} w_i \; g(\mathbf{x} \mid \Theta_i)$$

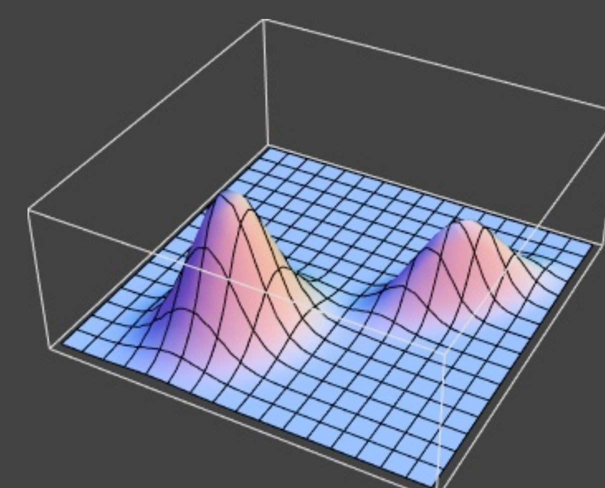
$0.7 \cdot$



$+\, 0.3 \cdot$



$=$



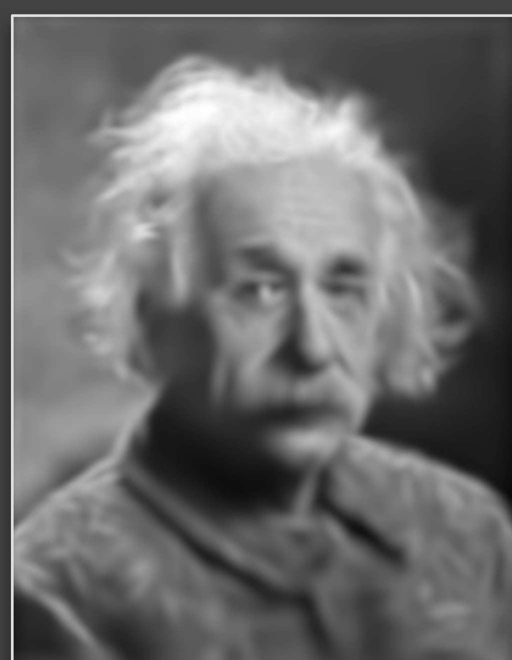
Gaussian mixture models

- Photon density modeled as a weighted sum of Gaussians:

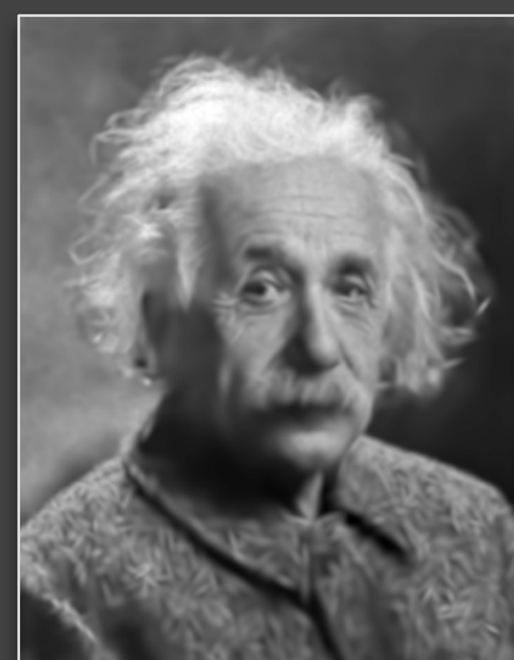
$$f\left(\mathbf{x} \mid \Theta\right) = \sum_{i=1}^{k} w_i \; g\left(\mathbf{x} \mid \Theta_i\right)$$

[Papas et al.]

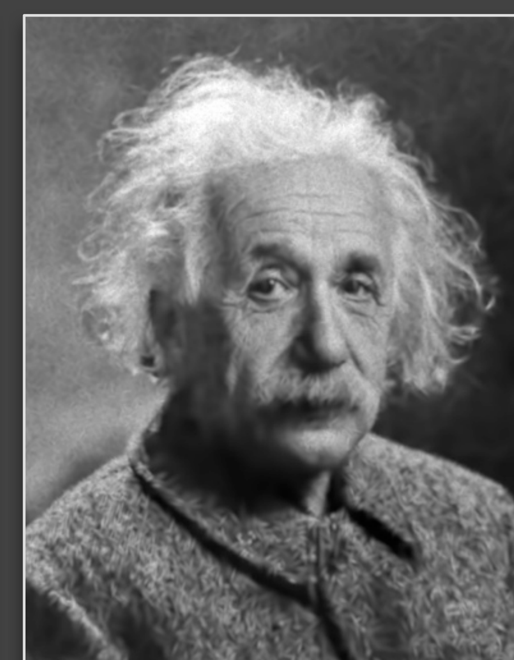
256 Gaussians



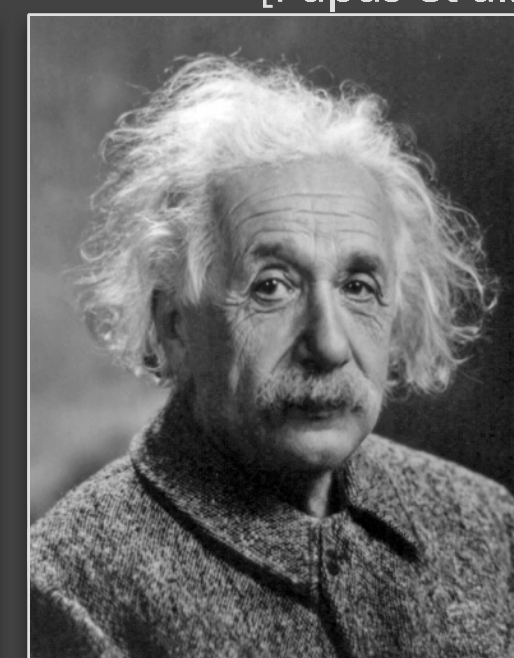
1024 Gaussians



4096 Gaussians



16384 Gaussians



Target density

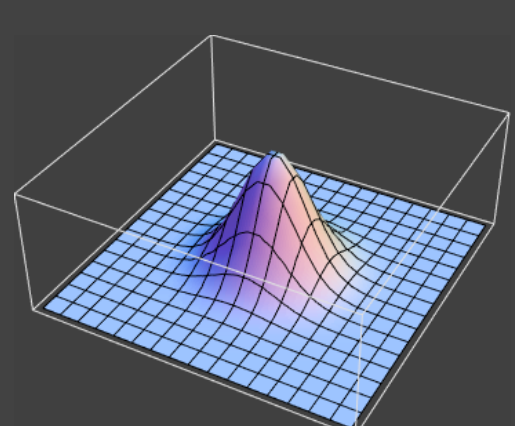
Gaussian mixture models

- Photon density modeled as a weighted sum of Gaussians:

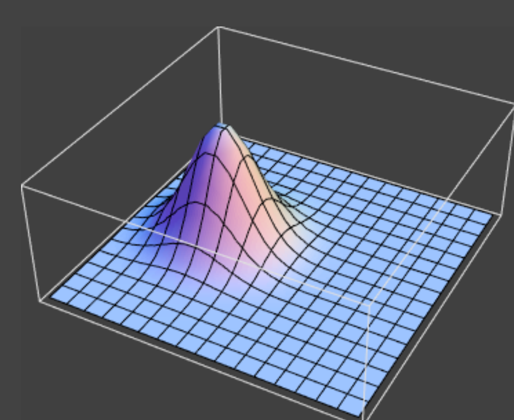
$$f\left(\mathbf{x} \mid \Theta\right) = \sum_{i=1}^{k} w_i \; g\left(\mathbf{x} \mid \Theta_i\right)$$

Unknown parameters Θ:

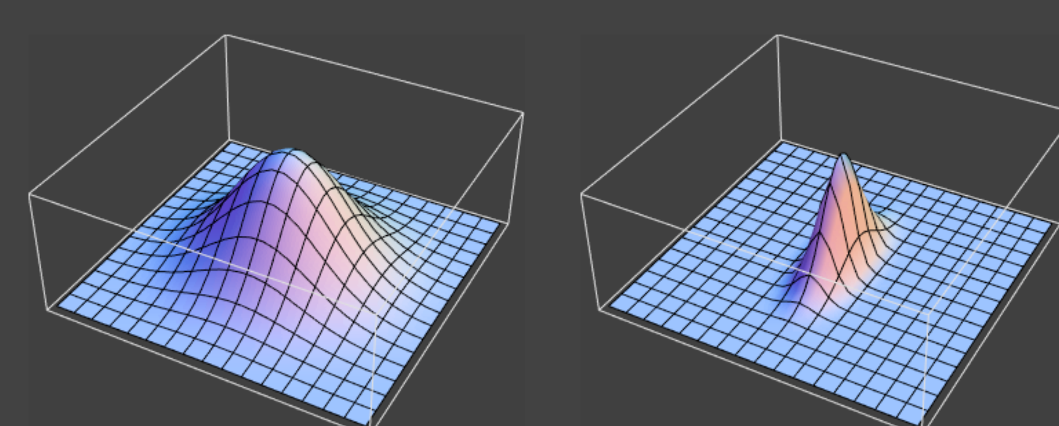
1. Weights



2. Means



3. Covariance matrices



Maximum likelihood estimation

Approach: find the "most likely" parameters, i.e.

$$\Theta^* := \underset{\Theta}{\operatorname{argmax}} \prod_{i=1}^{n} f\left(x_i \mid \Theta\right)$$

Mixture model (f)

Estimated parameters (Θ^*)

Photon locations (x_i)

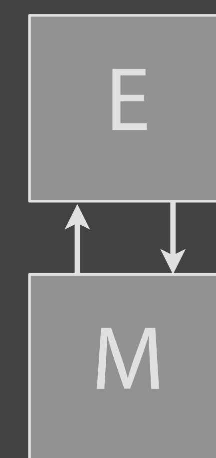
➔ **Expectation maximization**

Expectation maximization

- Two components:

  E-Step: establish soft assignment between photons and Gaussians

  M-Step: maximize the expected likelihood

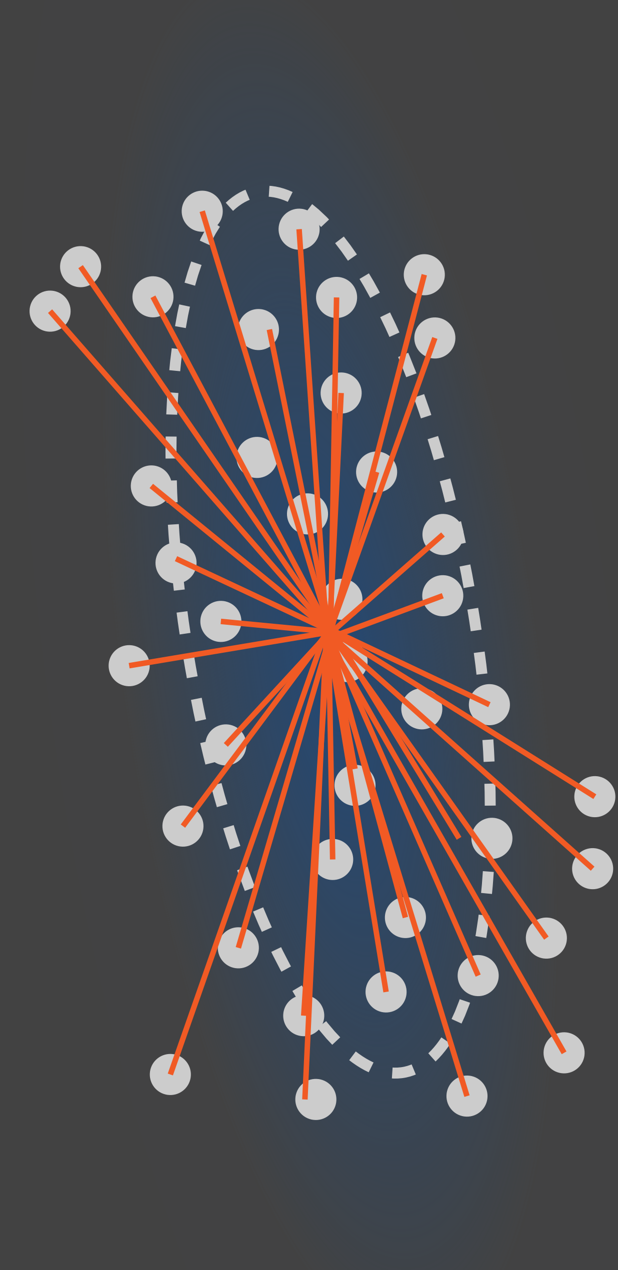


- Finds a locally optimal solution

  ➔ good starting guess needed!

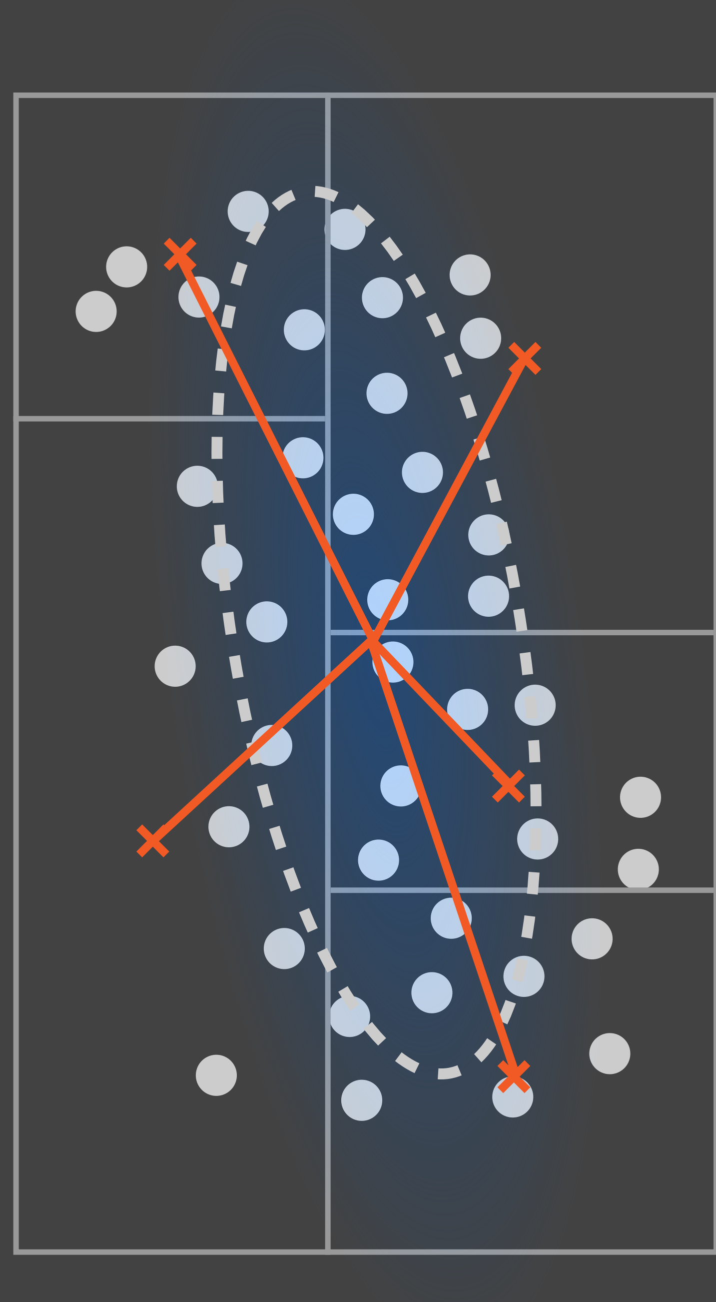
- Slow and scales poorly — $\mathcal{O}(n^2)$
  (where n: photon count)

Expectation maximization



Accelerated EM by [Verbeek et al. 06]

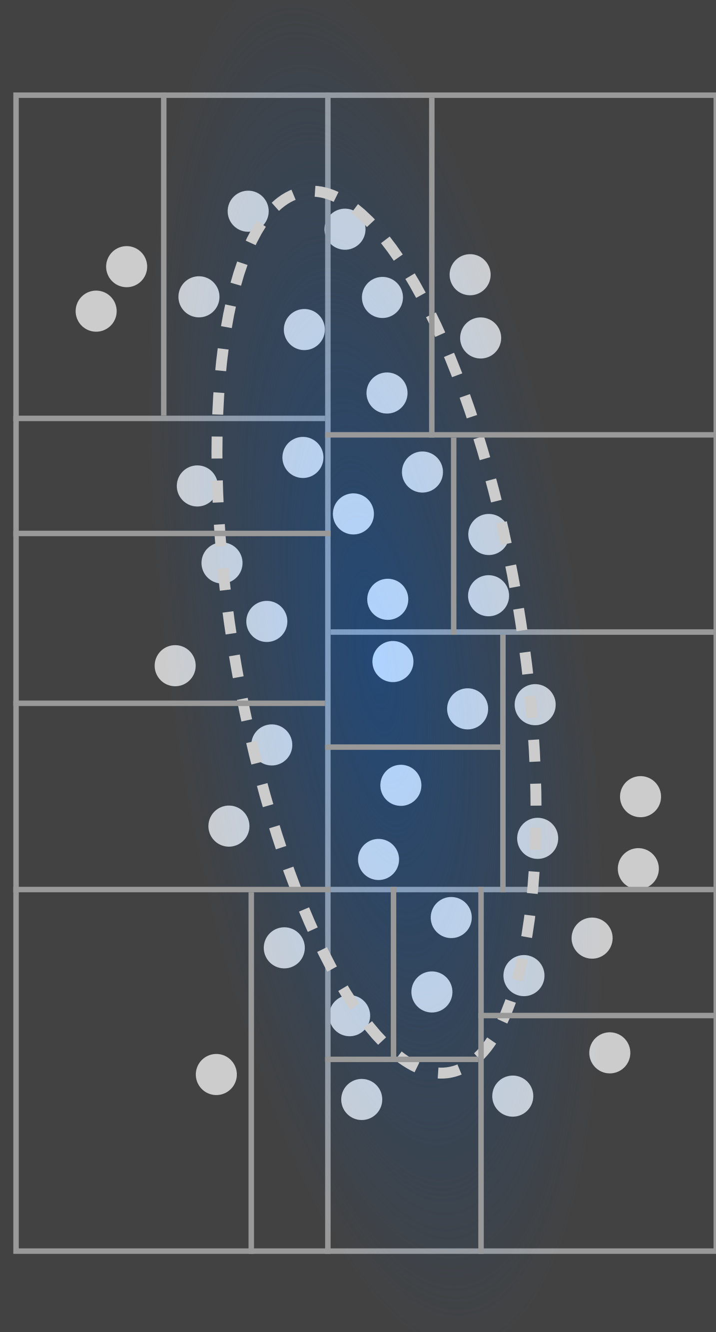
Accelerated EM



Stored cell statistics:

- photon count
- mean position
- average outer product

Progressive EM



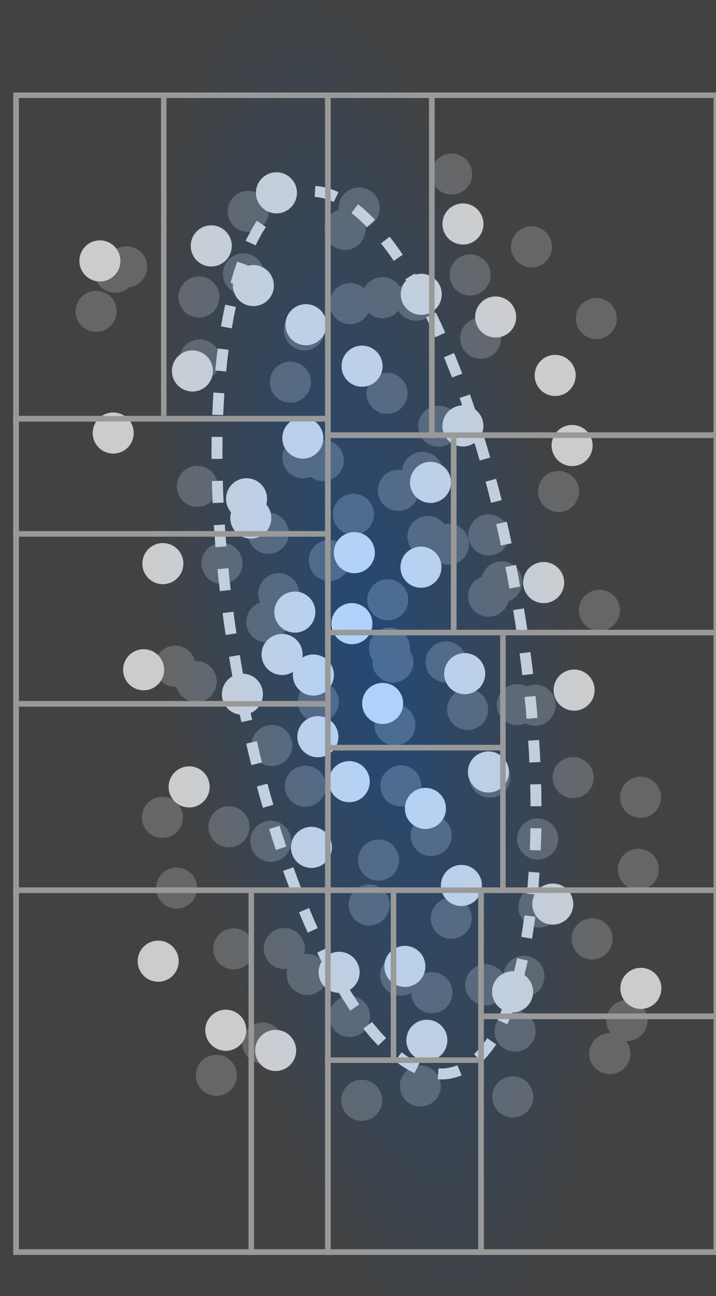
Stored cell statistics:

- photon count
- mean position
- average outer product

Our modifications:

- better cell refinement

Progressive EM



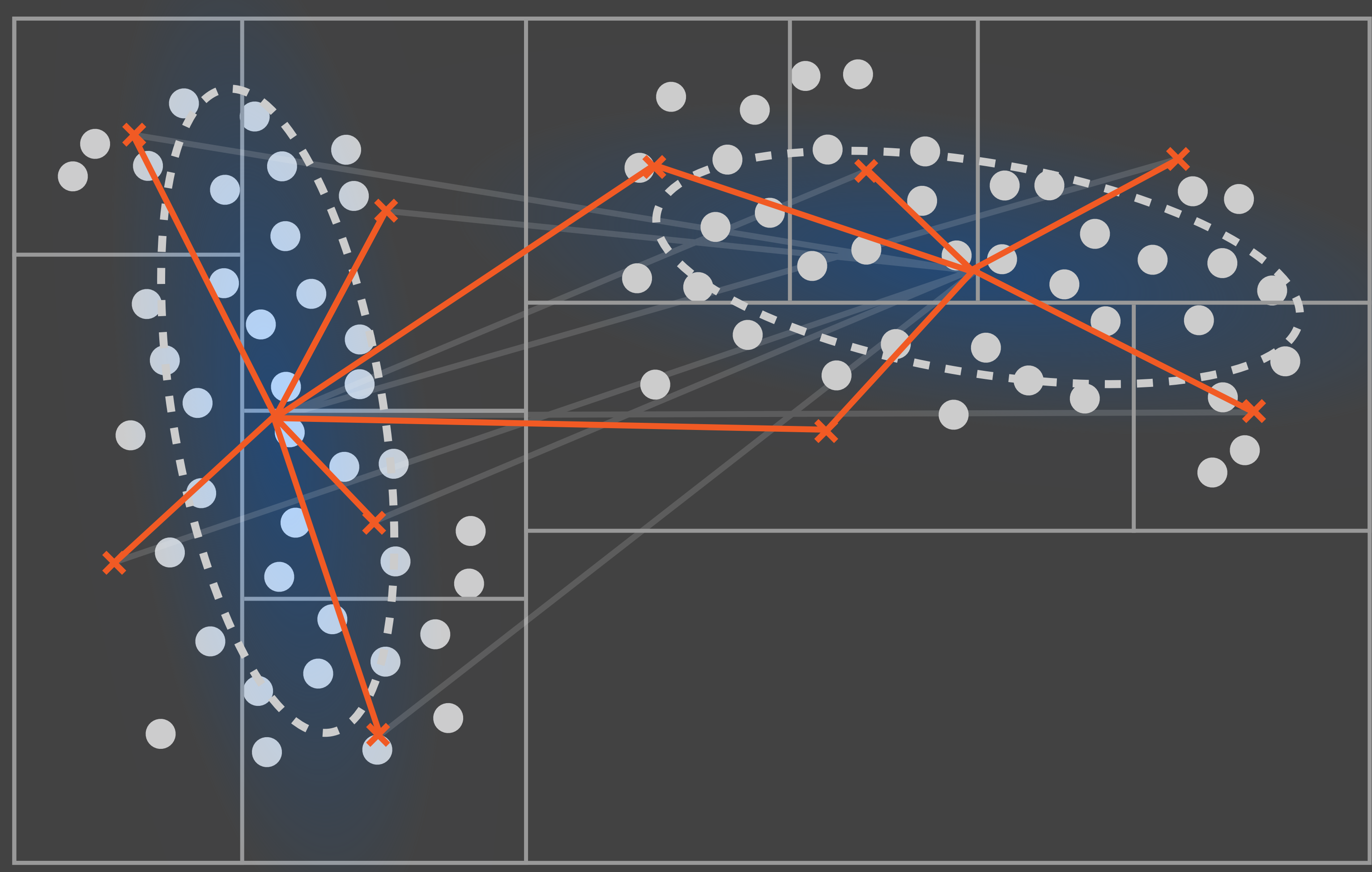
Stored cell statistics:

- photon count
- mean position
- average outer product

Our modifications:

- better cell refinement
- progressive photons shooting passes

Progressive EM



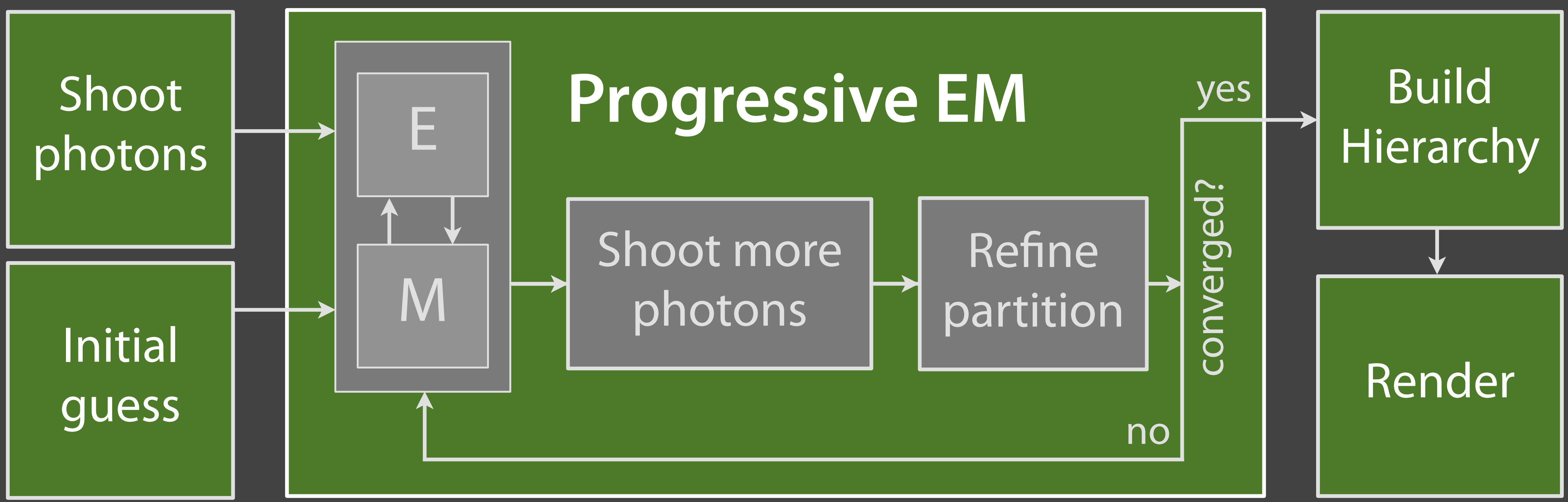
Stored cell statistics:

- photon count
- mean position
- average outer product

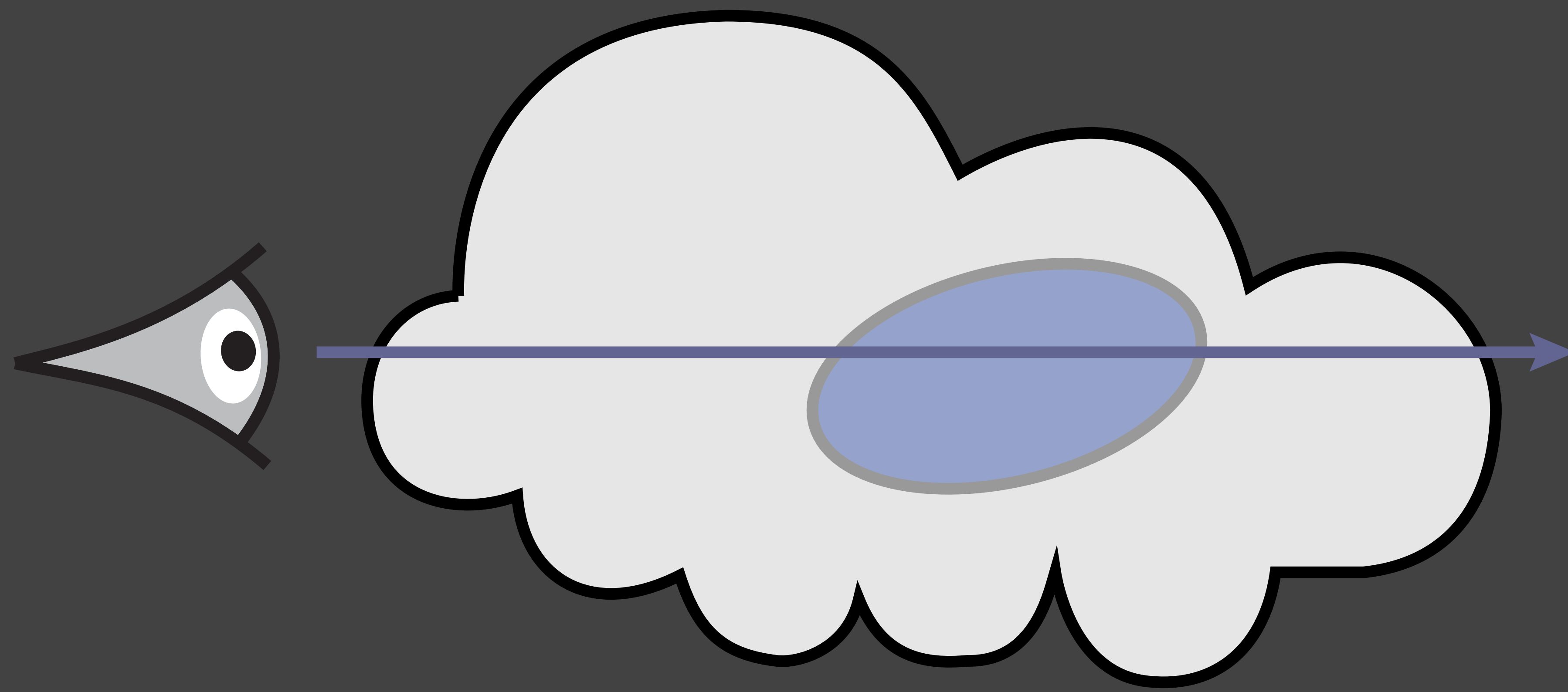
Our modifications:

- better cell refinement
- progressive photons shooting passes
- reduced complexity $\mathcal{O}(n^2) \rightarrow \mathcal{O}(n \log n)$

Pipeline overview



Rendering



$$\text{pixel value} = \sum_{i=1}^{k} \text{contrib}(i)$$

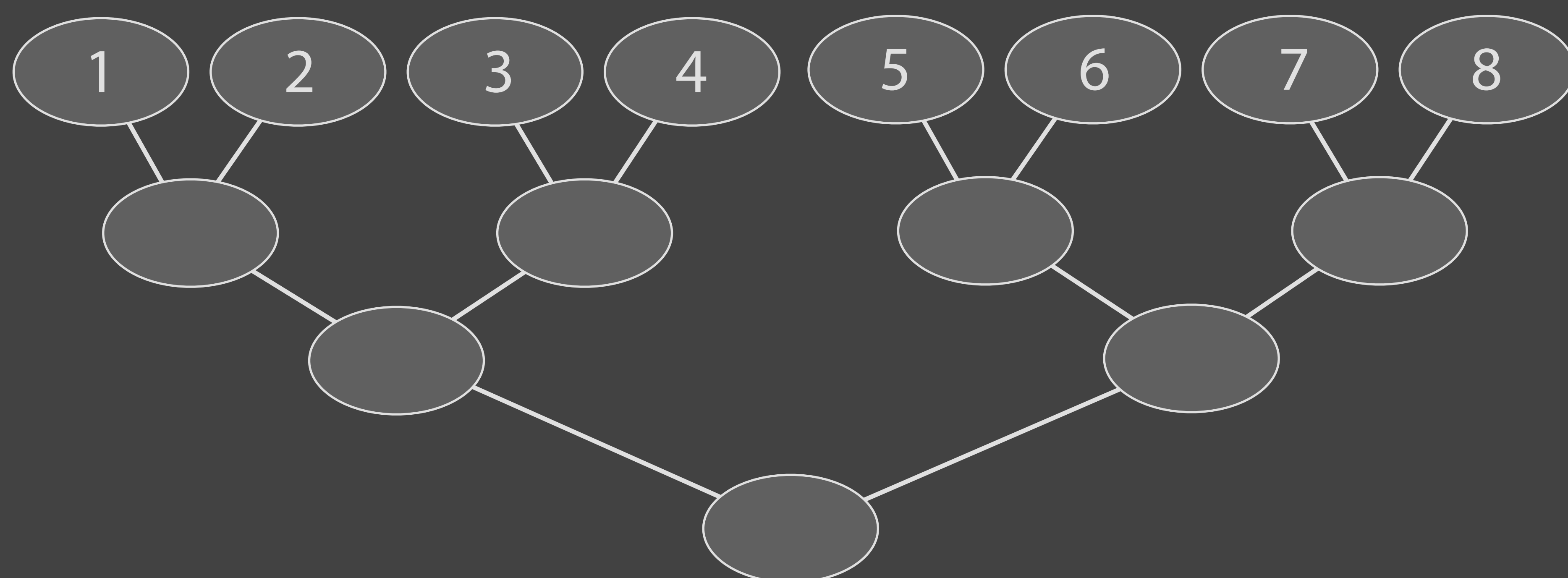
$$\text{contrib}(i) = \int_a^b g(\mathbf{r}(t)|\bar{\Theta}_i)\, e^{-\sigma_t\, t} dt = C_0 \left[\text{erf}\left(\frac{C_3 + 2C_2 b}{2\sqrt{C_2}} \right) - \text{erf}\left(\frac{C_3 + 2C_2 a}{2\sqrt{C_2}} \right) \right]$$

...

Level of detail hierarchy

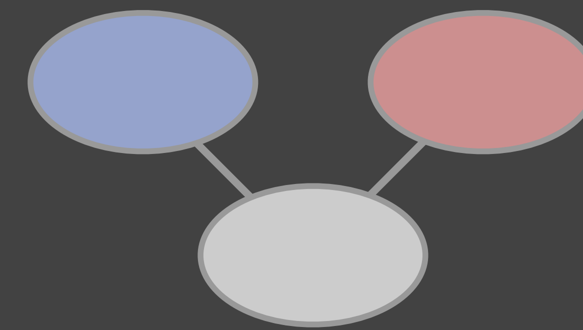
Agglomerative construction:

- Repeatedly merge nearby Gaussians based on their Kullback-Leibler divergence

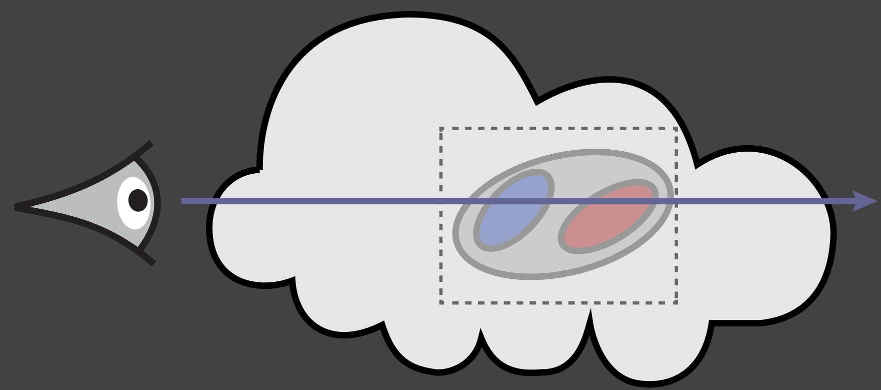


Rendering

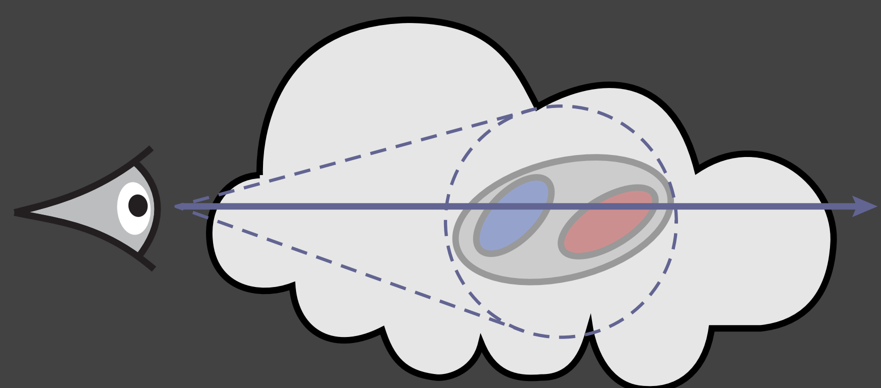
Example hierarchy:



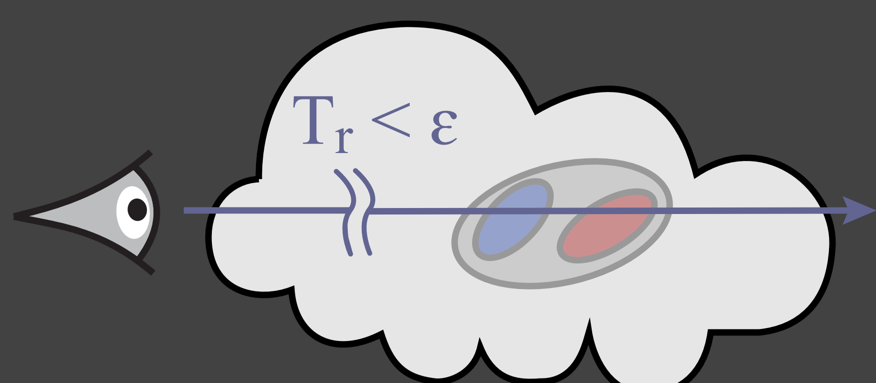
Criterion 1: bounding box intersected?

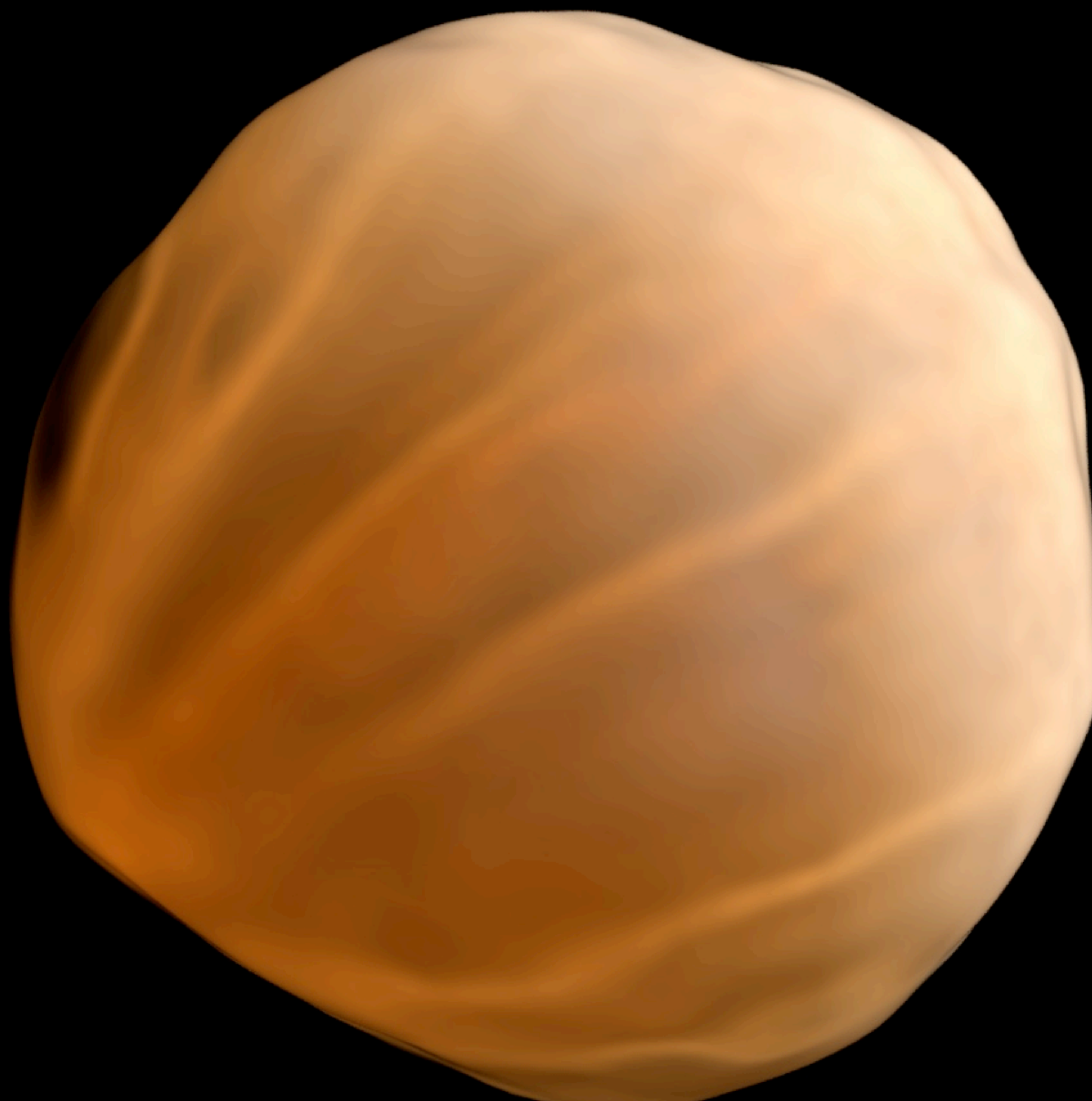


Criterion 2: solid angle large enough?



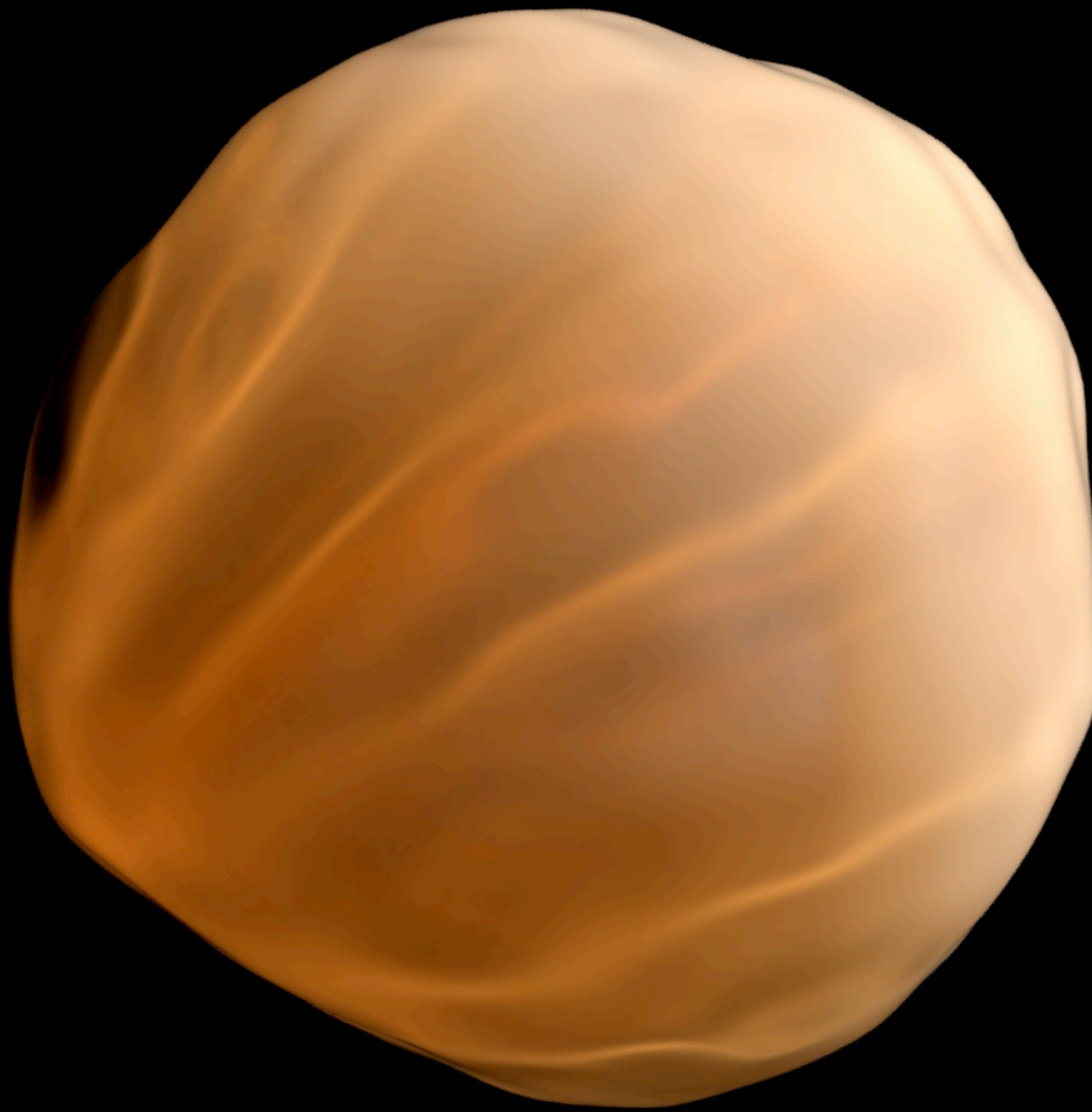
Criterion 3: attenuation low enough?





BRE: 1M Photons

23+192 = 215 s

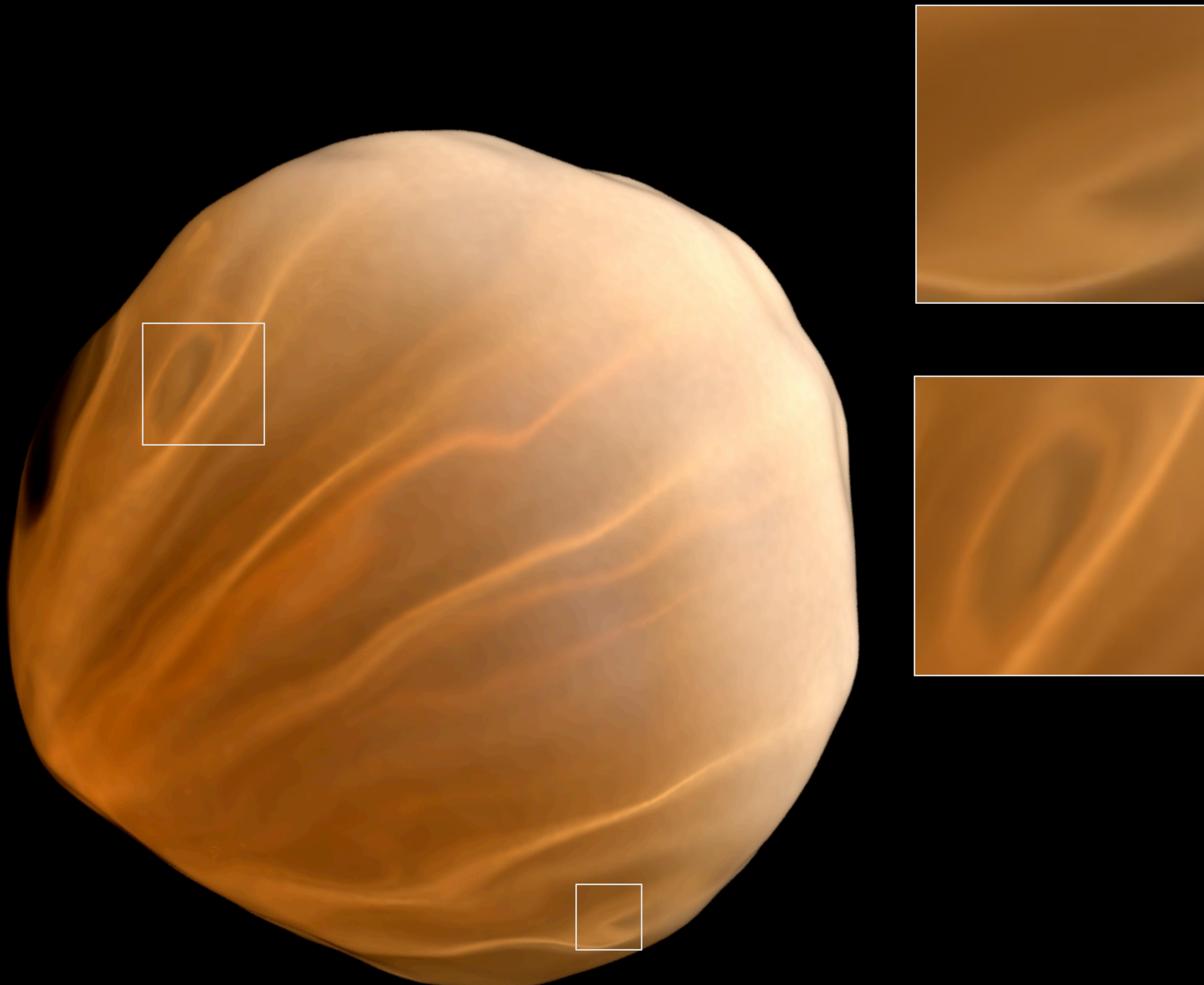


Our method: 4K Gaussians

(fit to 1M photons)

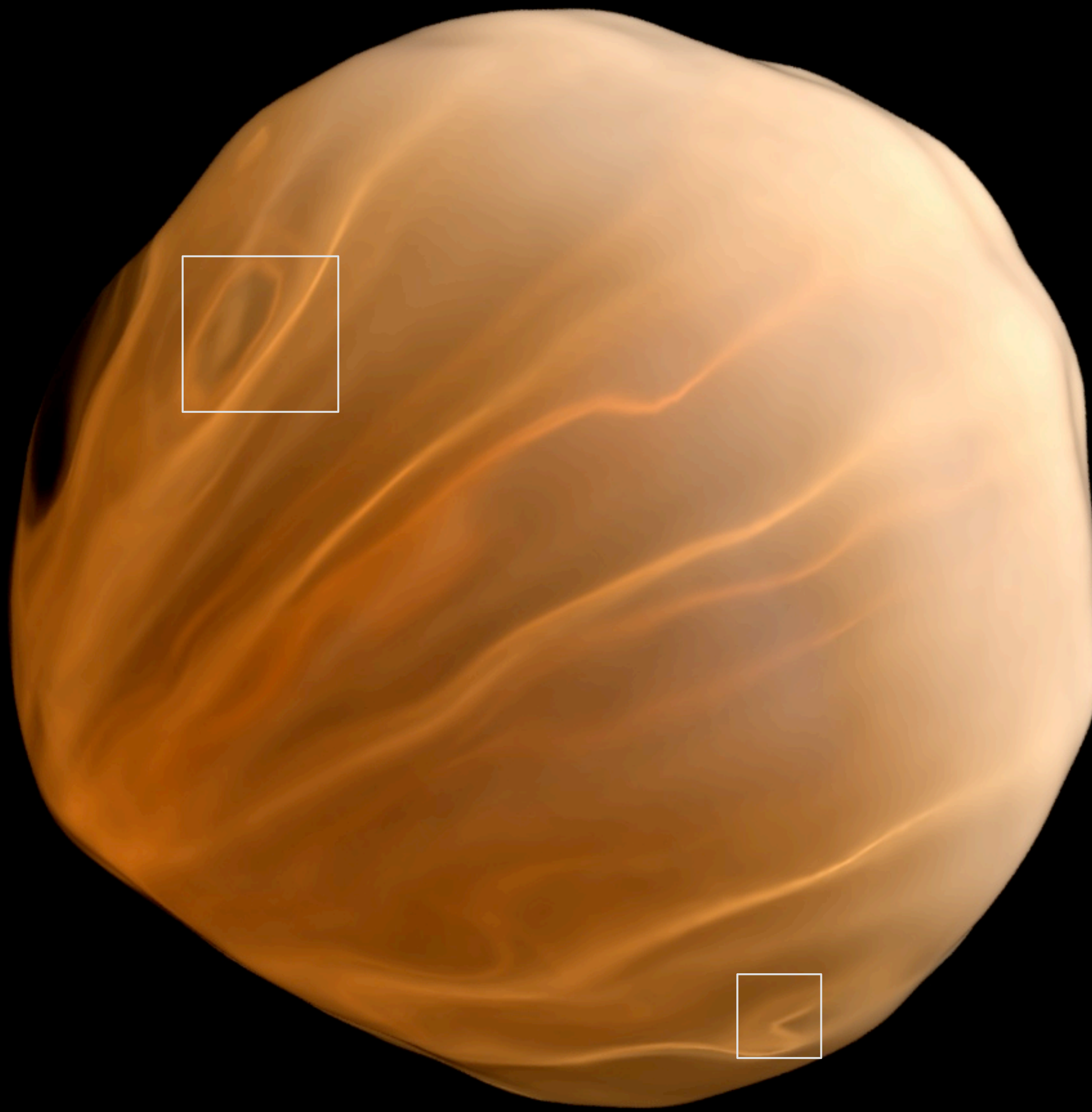
35+24 = 59 s

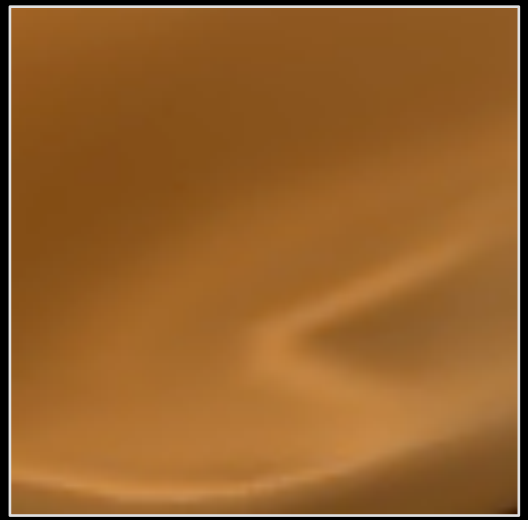
(3.6×)

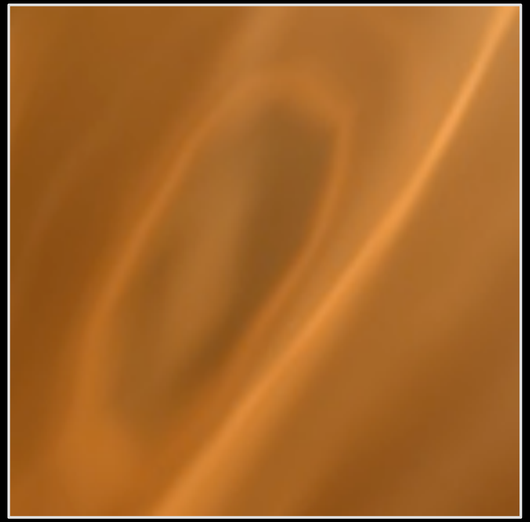


BRE: 18M Photons

507+609 = 1116 s

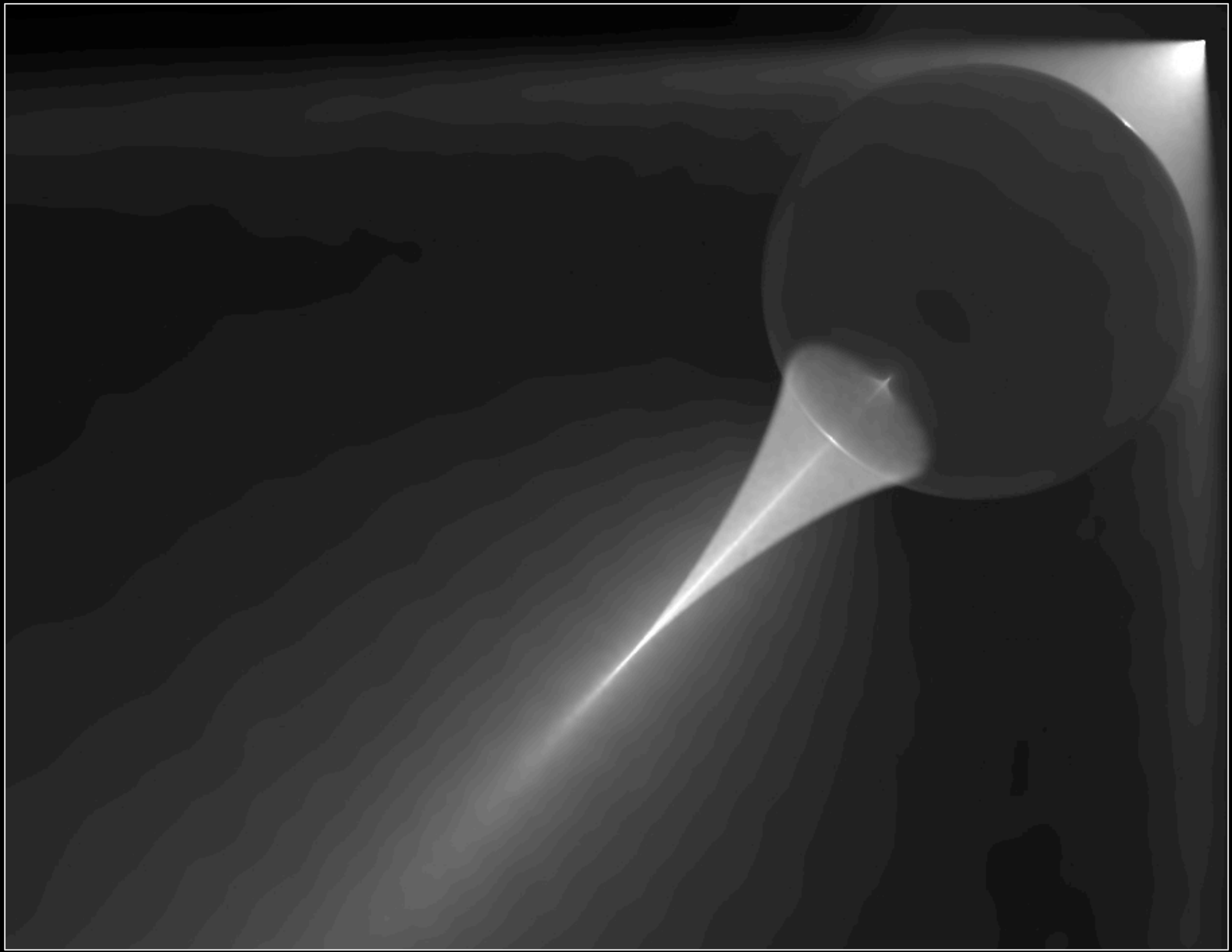






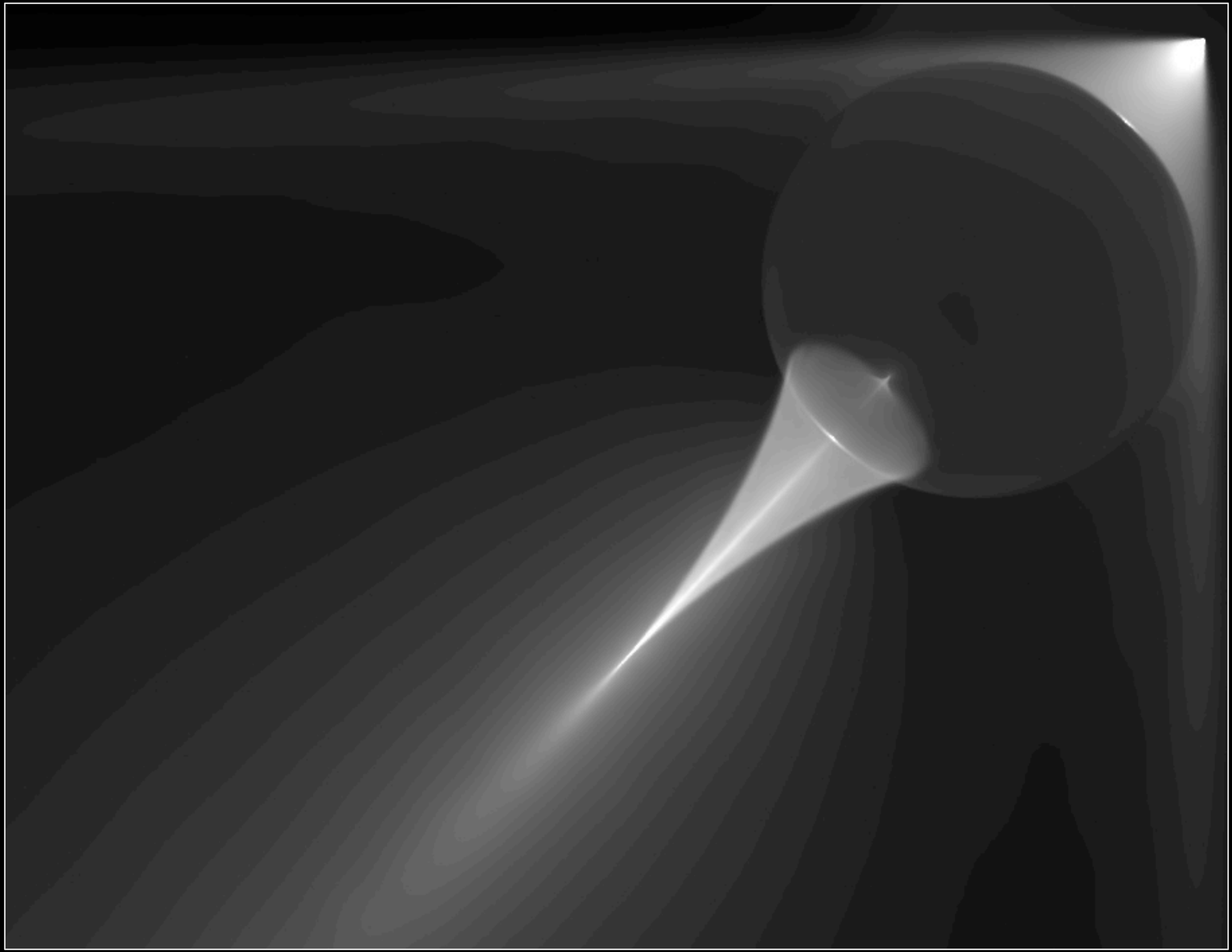
Our method: 64K Gaussians
(fit to 18M photons)

868+66 = 934 s
(1.2×)



BRE: 4M Photons

89 + 638 = 727 s

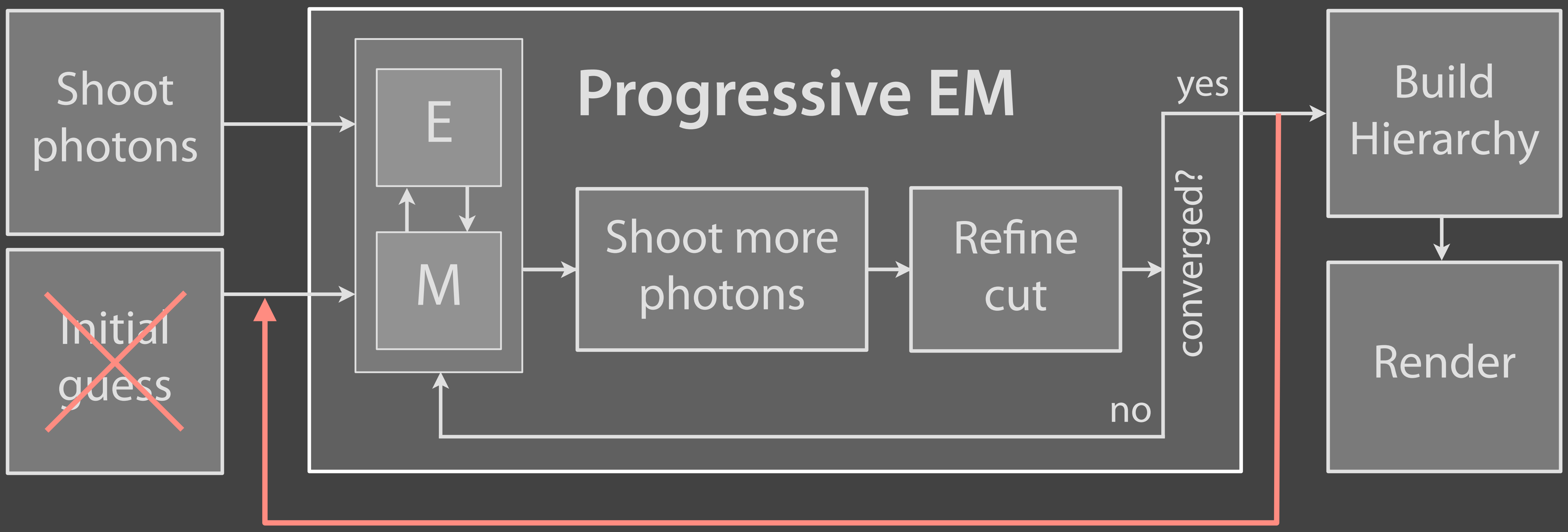


Our method: 16K Gaussians

330 + 127 = 457 s

(1.6×)

Temporal Coherence



- Feed the result of the current frame into the next one

→ Faster fitting, no temporal noise

[Video]

[Video]

GPU-based rasterizer:

- Anisotropic Gaussian splat shader: 30 lines of GLSL
- Gaussian representation is very compact (4096-term GMM requires only ~240KB of storage)

Conclusion

Contributions

- Rendering technique based on parametric density estimation
- Uses a progressive and optimized variant of accelerated EM
- Compact & hierarchical representation of volumetric radiance
- Extensions for temporal coherence and real-time visualization