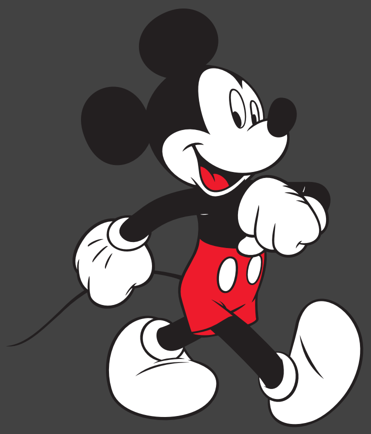
# Progressive Expectation–Maximization for Hierarchical Volumetric Photon Mapping

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

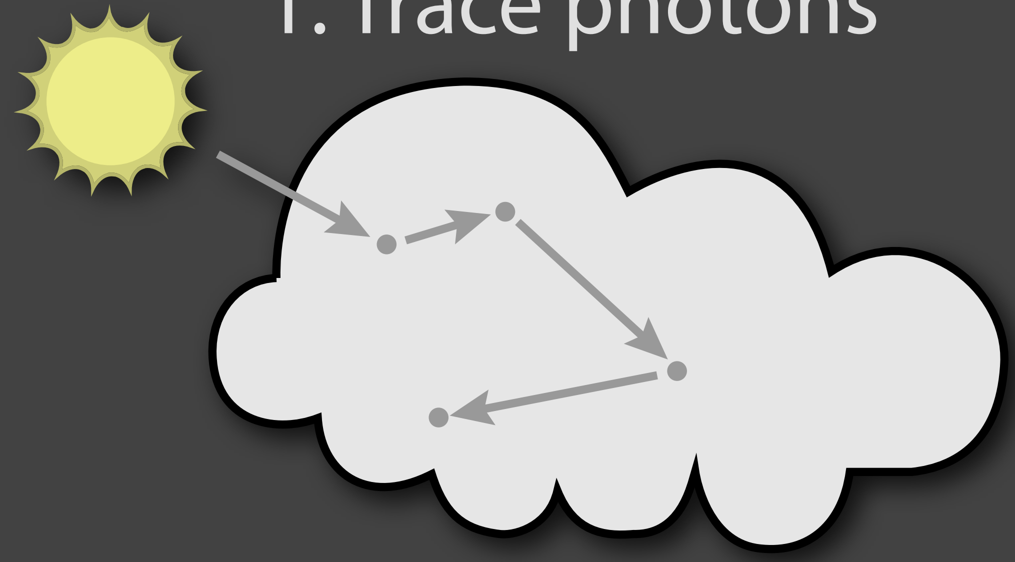
[1] Disney Research, Zürich
[2] Cornell University
[3] ETH Zürich



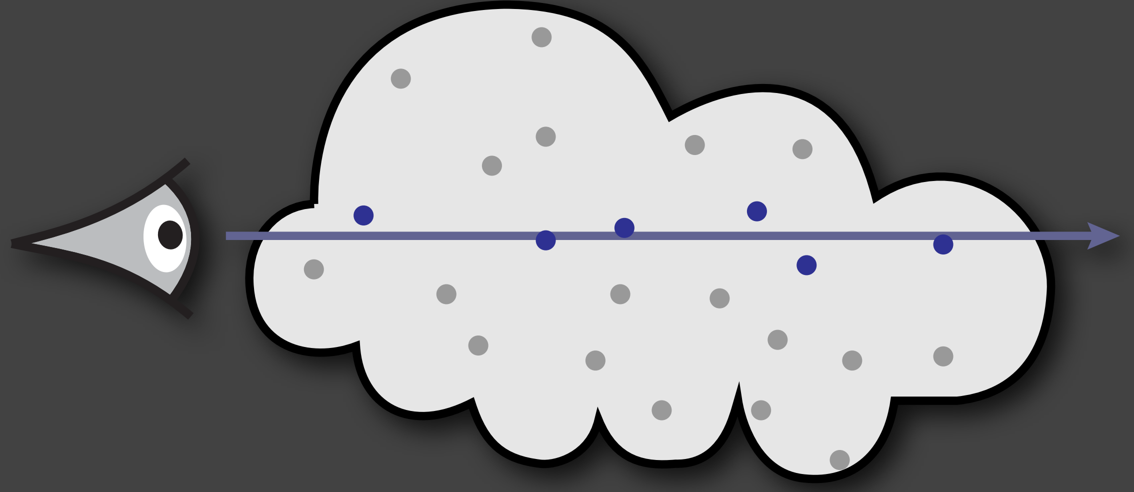
## Motivation

### Volumetric photon mapping

1. Trace photons



2. Radiance estimate



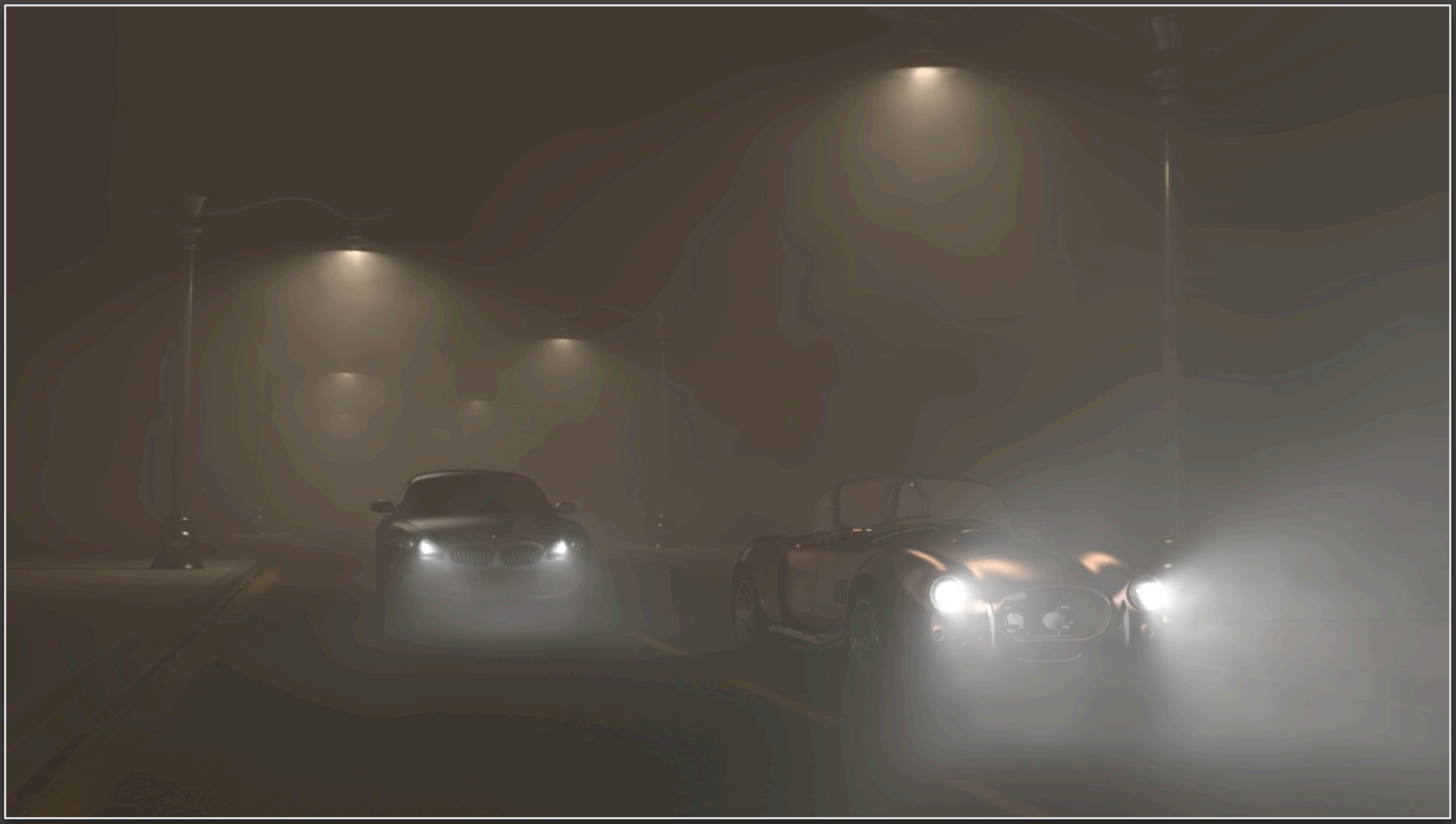
### Issues

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

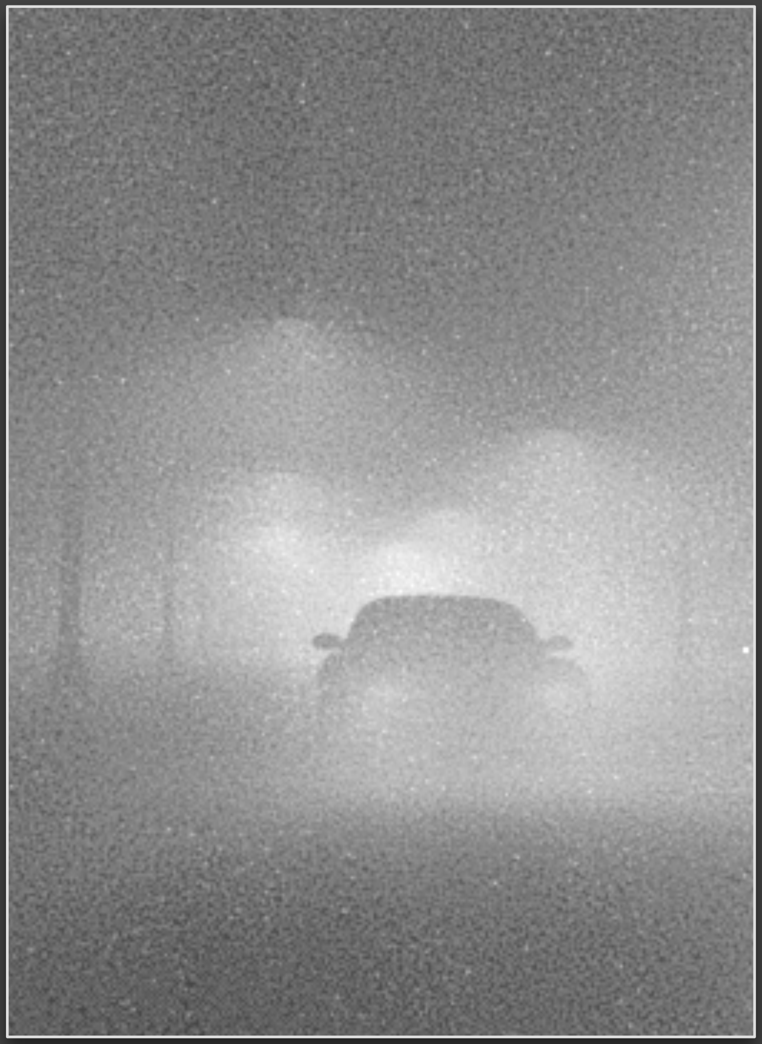
Downsides of volumetric photonmapping / BRE:

- To capture fine illumination details, one needs a vast number of photons (curse of dimensionality)
- high memory consumption & rendering time
- much time is spent processing photons that contribute little to the rendered image
- flickering in animations

## Motivation



Beam radiance estimate : 917K photons

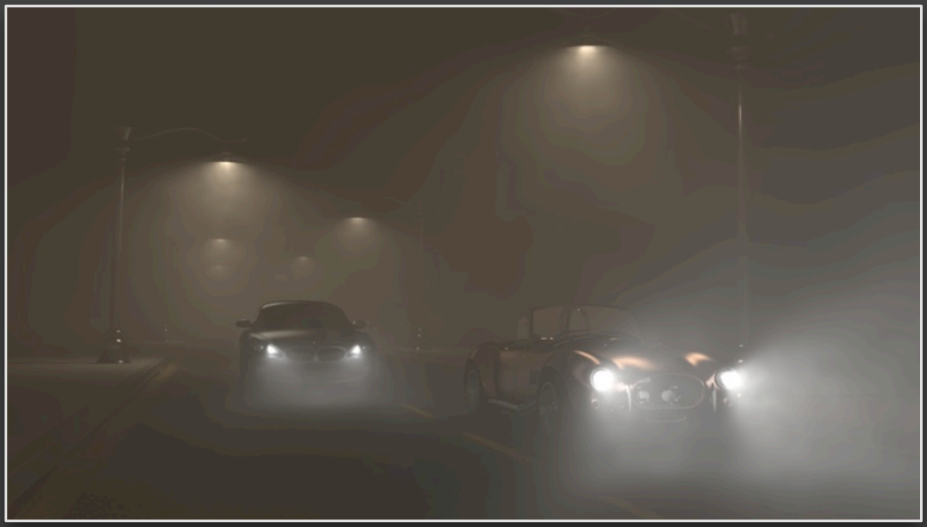


Per-pixel render time

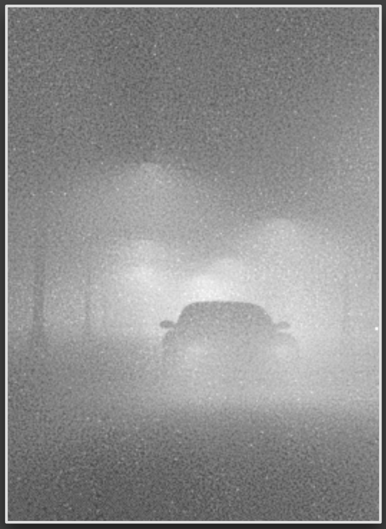
Problematic BRE case: a significant amount of time is spent in areas that are actually strongly attenuated in the rendering.

## Motivation

Beam radiance estimate : 917K photons

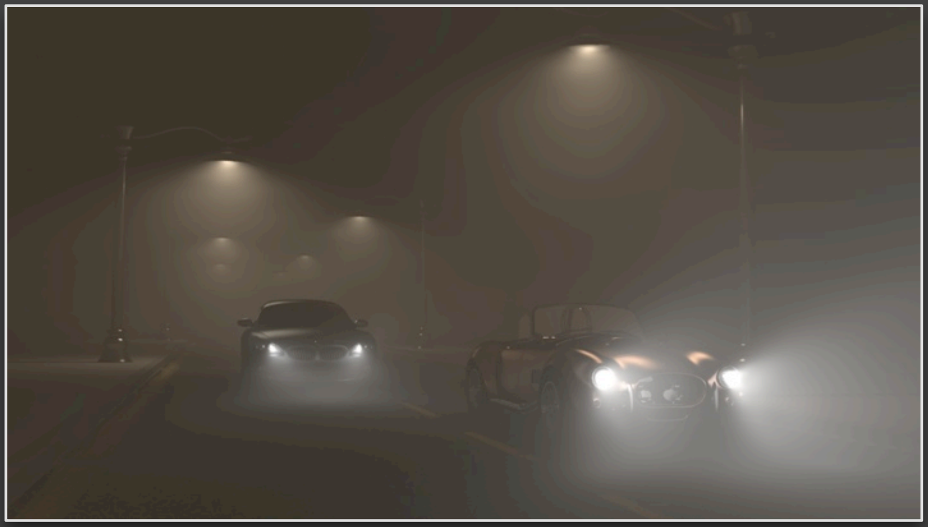


Render time: 281 s



Per-pixel render time

Our method: 4K Gaussians



Render time: 125 s

Per-pixel render time

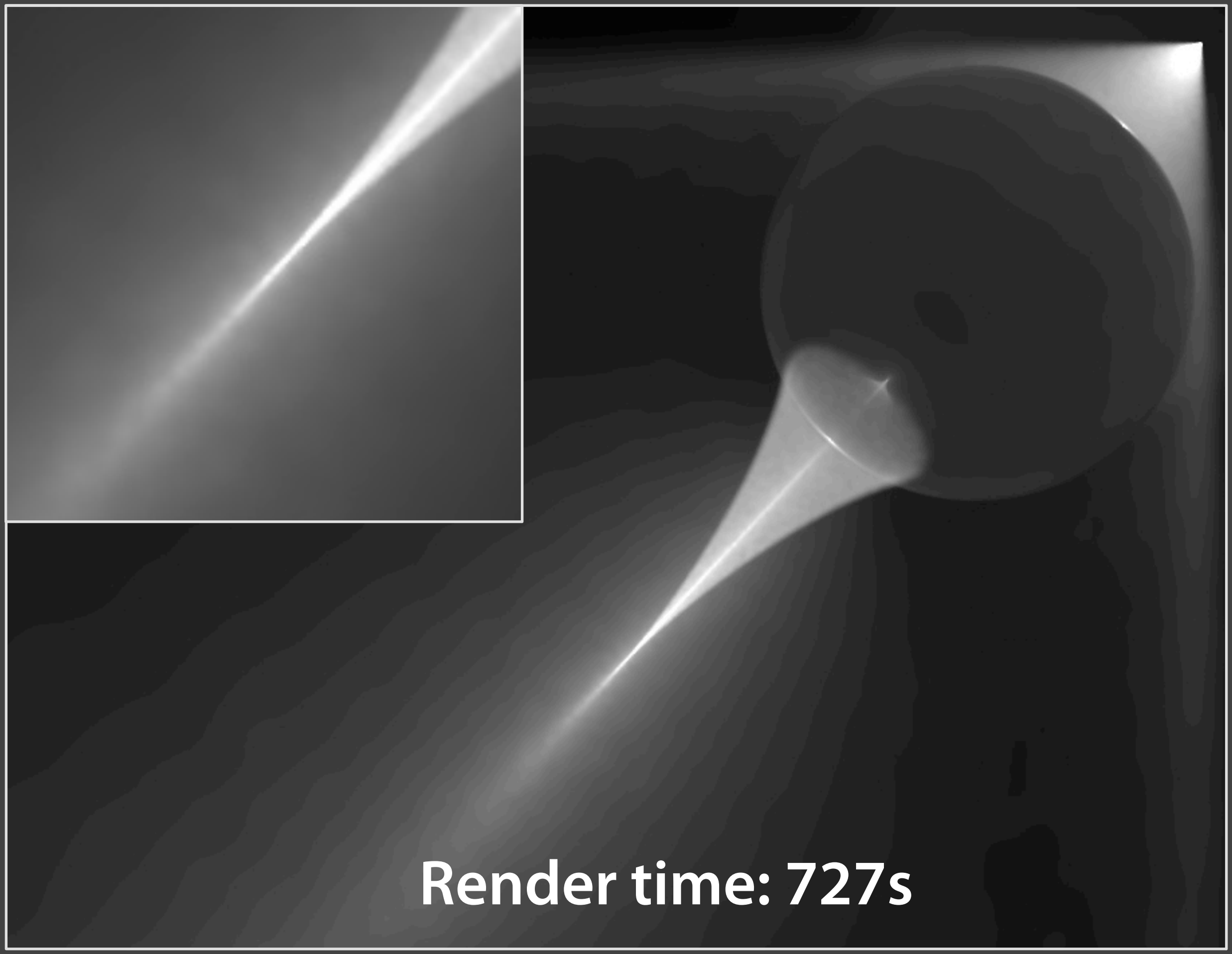
## Our approach:

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

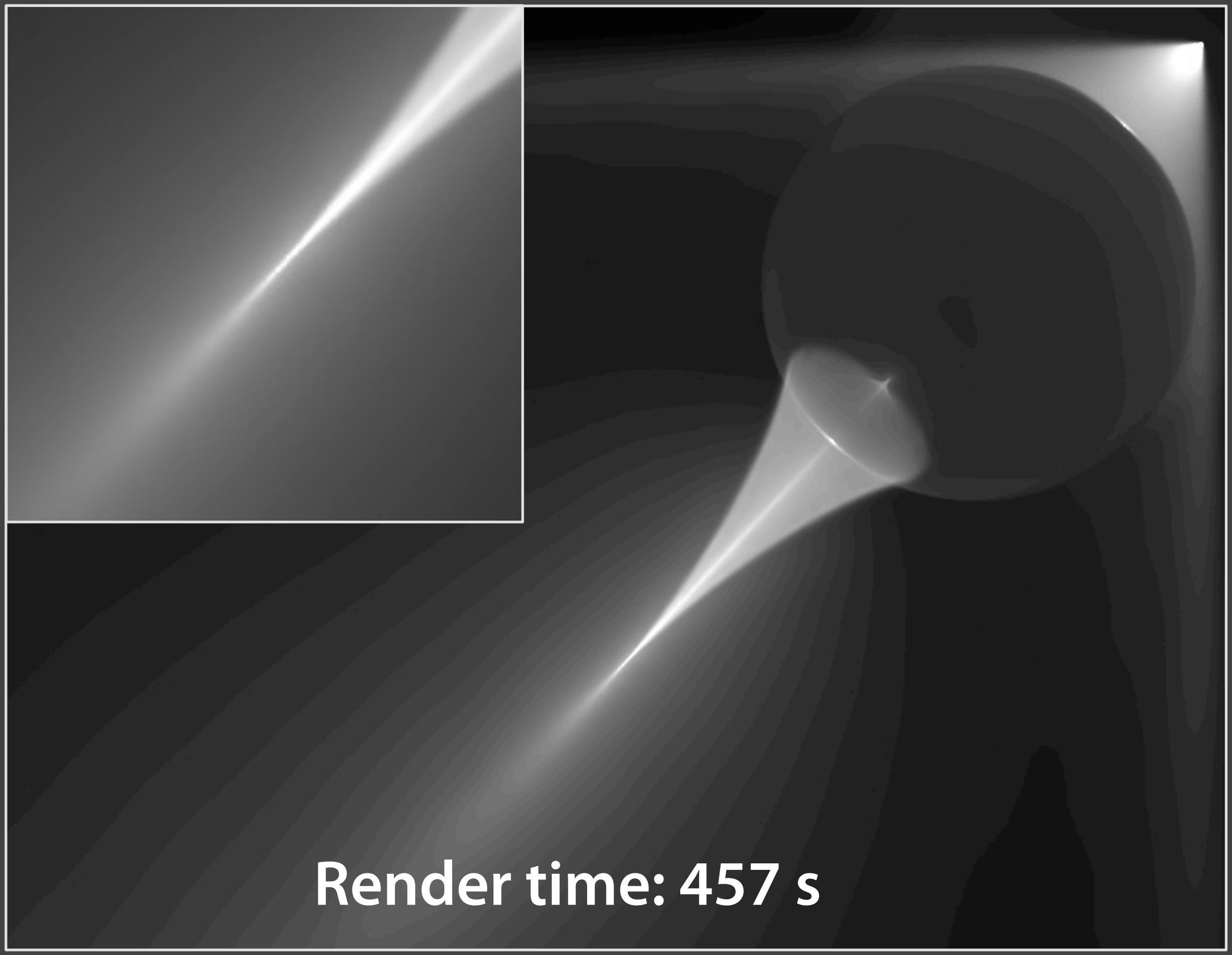
The presented method renders this scene faster by relying on a different radiance representation that only requires four thousand terms in this example. The technique can switch to a lower levels of detail where it makes sense -- for instance, when drawing the attenuated distant light sources.

## Motivation

Beam radiance estimate : 4M photons



Our method: 16K Gaussians



### Our approach:

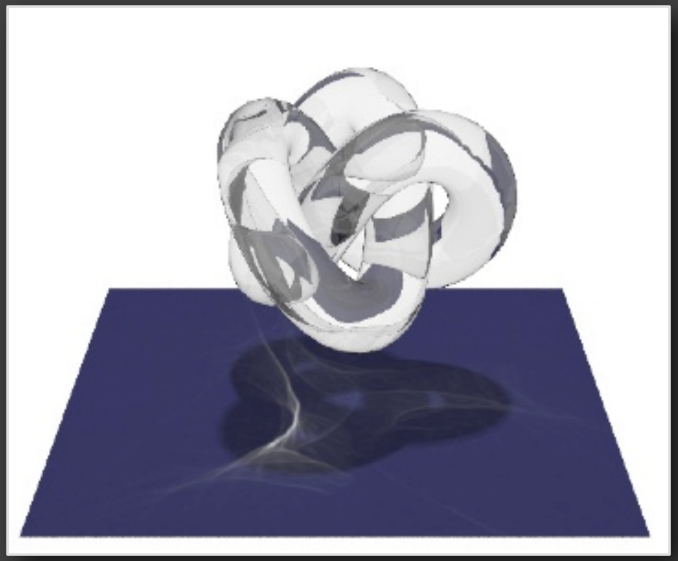
- represent radiance using a Gaussian mixture model (**GMM**)
- fit using progressive expectation maximization (**EM**)
- render with multiple levels of detail

Saturday, August 4, 12

The Gaussian-based representation is not only faster, but can also blur noise in way so that important image features are retained.

## Related work

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



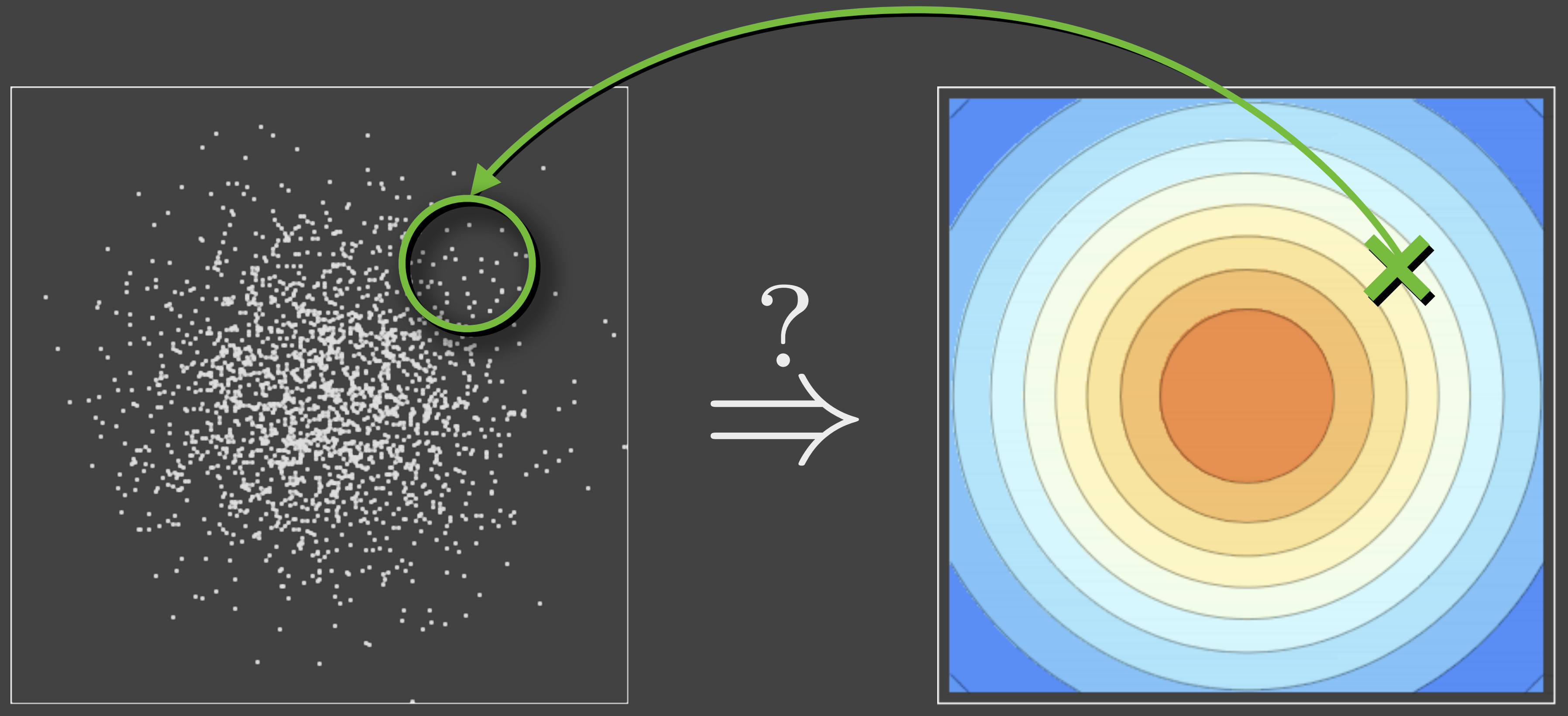
- Photon relaxation
  [Spencer et al. 09]

- Hierarchical photon mapping
  [Spencer et al. 09]

Saturday, August 4, 12

There have been a number of techniques that solve related problems in the surface rendering setting (these were already covered by previous speakers).

## Density estimation



Given photons

$x_1, x_2, \dots$

approximately determine their density $f$

**Nonparametric:**

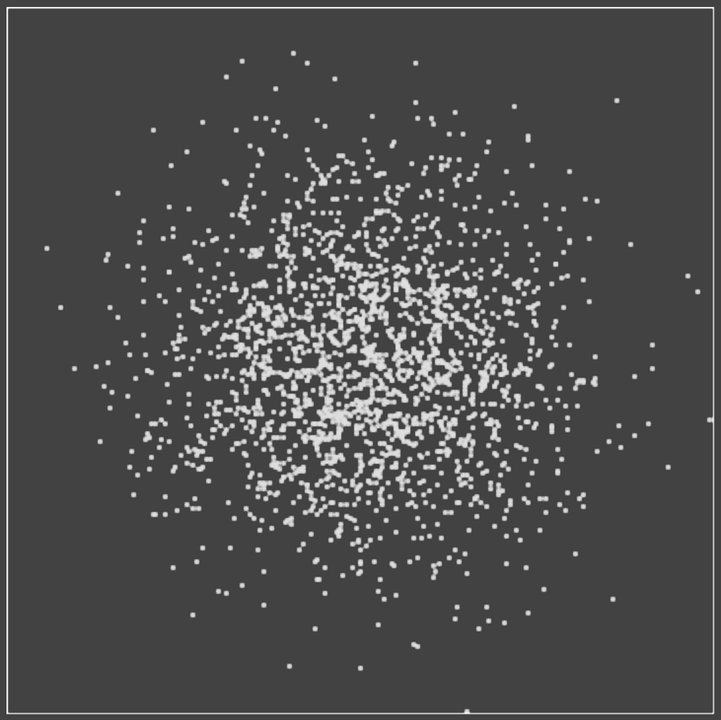
- Count the number of photons within a small region

Saturday, August 4, 12

take-away message: a radiance representation other than photons can be valuable not only to reduce rendering time, but also to improve quality. To see how we can find such a representation, let's start with a bit of review:

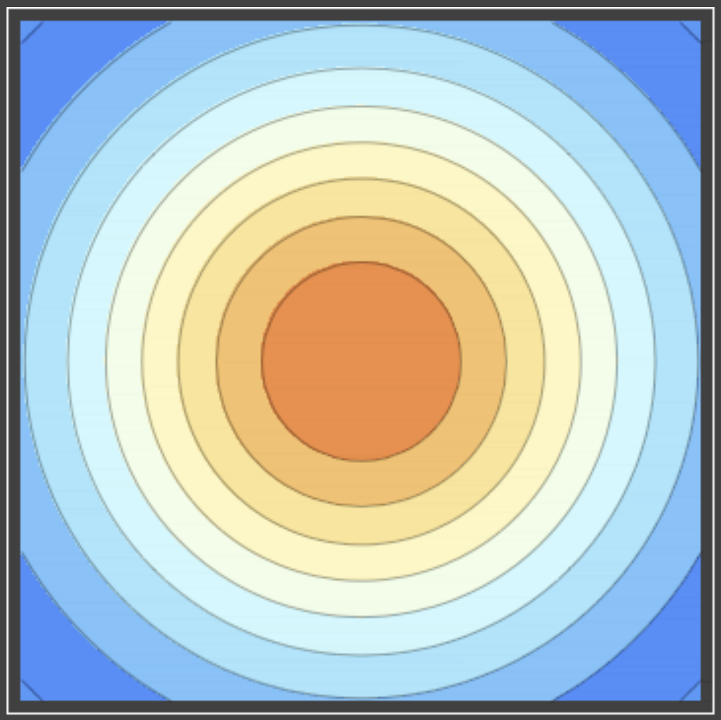
Photon mapping (non-parametric density estimation): count the number of photons that fall into a small region around the point in question, repeat for every evaluation of the density function.

# 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

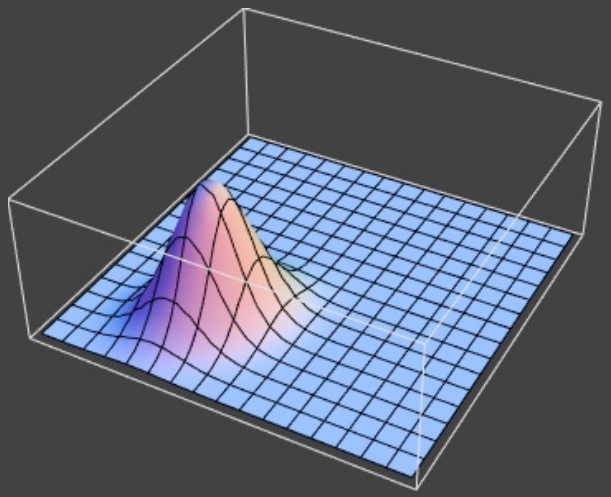
Parametric density estimation: assume that the photons are drawn from a certain **known** distribution -- for instance, a 2D Normal distribution. The goal is to find the most suitable parameter values. This is the approach used in this project.

## Gaussian mixture models

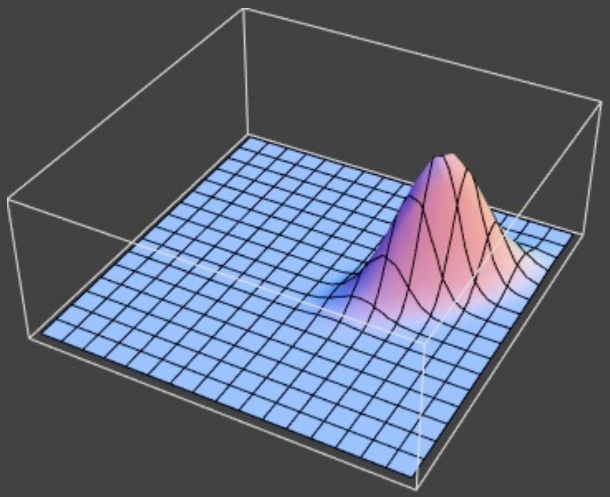
- Photon density modeled as a weighted sum of Gaussians:

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

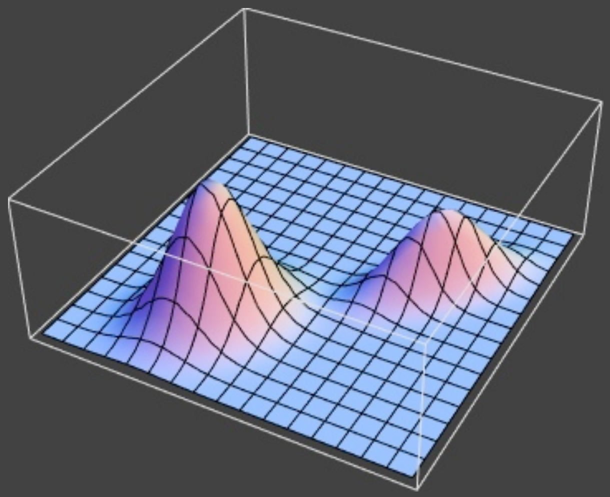
$0.7\cdot$



$+0.3\cdot$



$=$



Need a distribution that is general enough:

Gaussian mixture model (used in AI/data minig/statistics): weighted sum of gaussian functions.

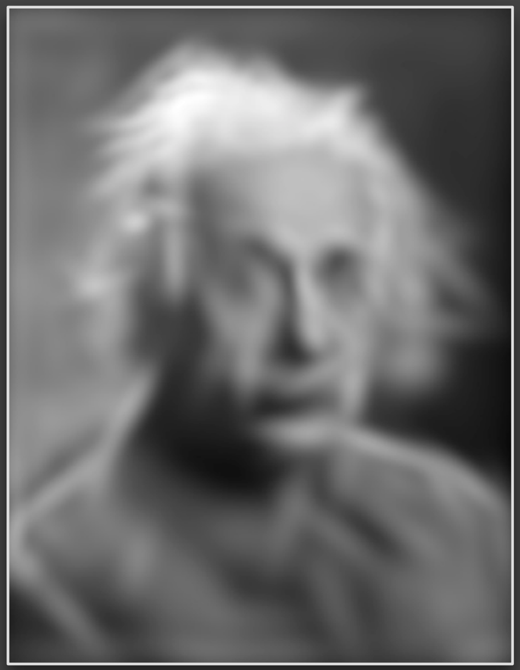
Of course, the real radiance distribution is usually not a gaussian mixture model

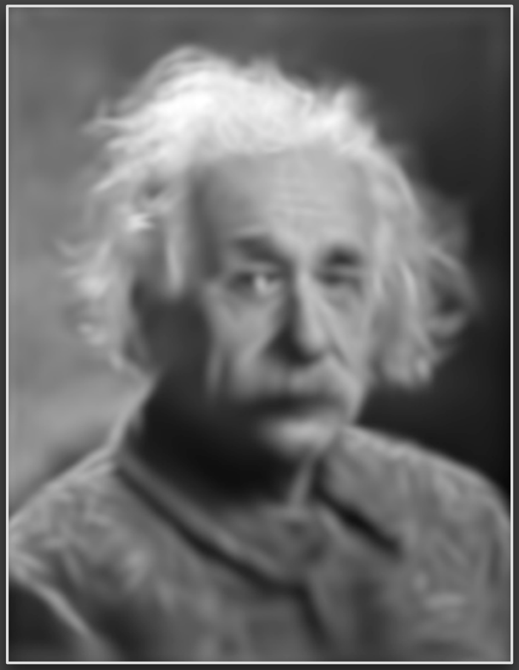
# Gaussian mixture models

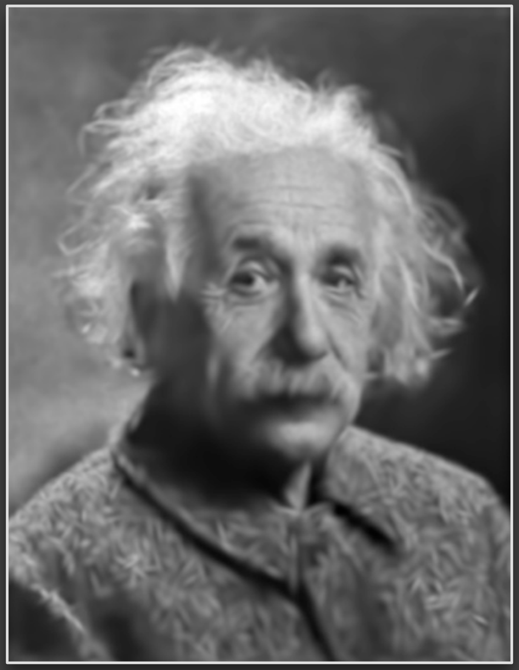
- Photon density modeled as a weighted sum of Gaussians:

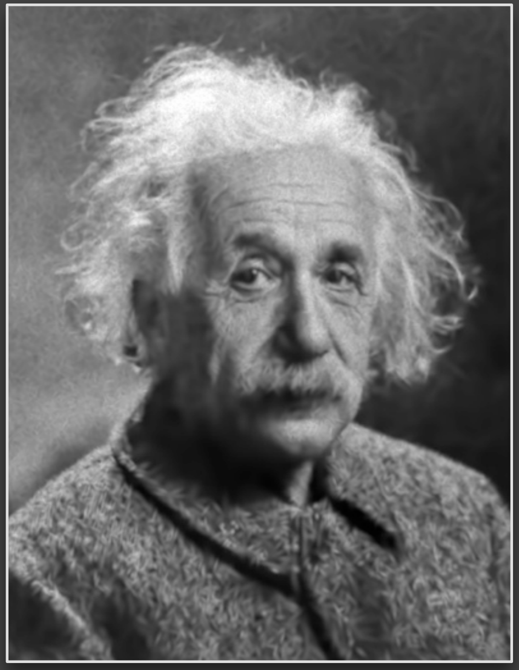
$$f\left(\mathbf{x} \mid \Theta\right) = \sum_{i=1}^{k} w_i \; g\left(\mathbf{x} \mid \Theta_i\right)$$

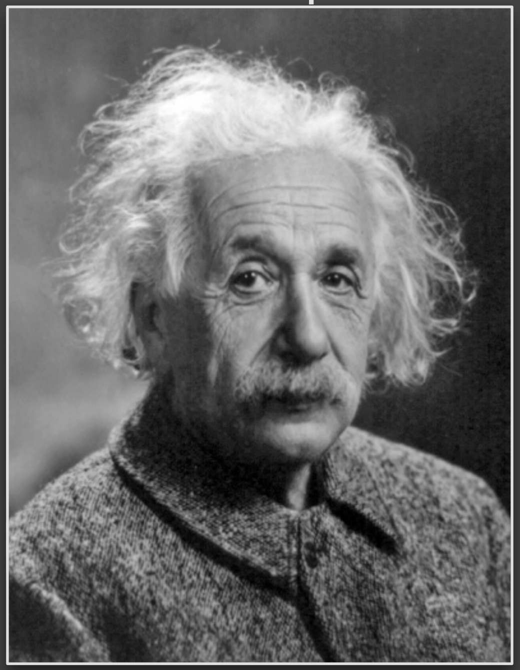
[Papas et al.]


256 Gaussians


1024 Gaussians


4096 Gaussians


16384 Gaussians


Target density

.. but with an increasingly large number of terms, it can be approximated arbitrarily well by one.

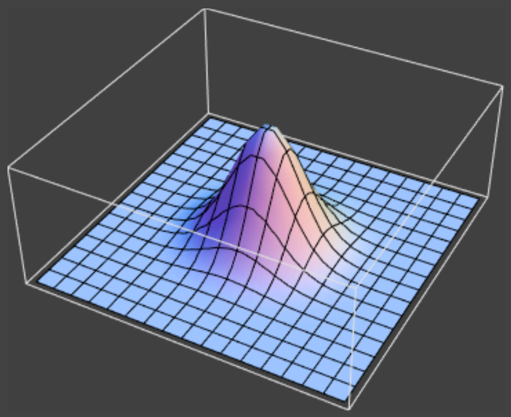
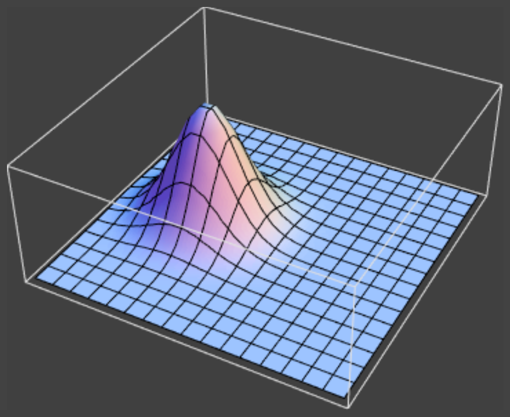
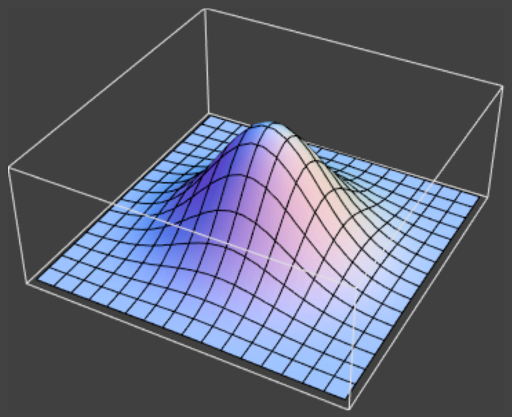
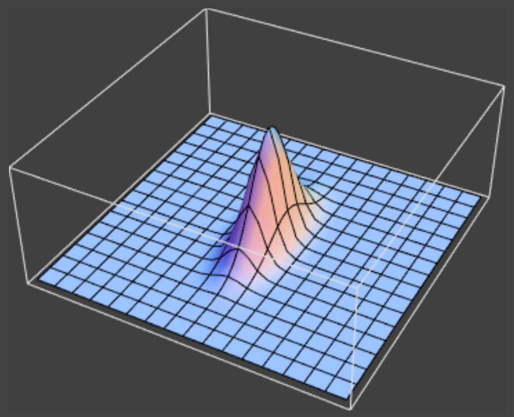
# Gaussian mixture models

- 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 $\Theta$:**

1. Weights
2. Means
3. Covariance matrices

Saturday, August 4, 12

The missing parameters then that need to be estimated are the weights, means, and covariance matrices (4–64K of them!).

## Maximum likelihood estimation

Approach: find the “most likely” parameters, i.e.

$$\underset{\text{Estimated parameters}}{\underset{\uparrow}{\Theta^*}} := \underset{\Theta}{\operatorname{argmax}} \prod_{i=1}^{n} \overset{\text{Mixture model}}{\overset{\downarrow}{f}} \left( \underset{\text{Photon locations}}{\underset{\uparrow}{x_i}} \mid \Theta \right)$$

$\Rightarrow$ **Expectation maximization**

Having decided on parametric density estimation and Gaussian mixtures, we need some way of finding their parameters.

-> Find parameters that are the most likely given the photon observations. This is a maximum likelihood estimation problem.

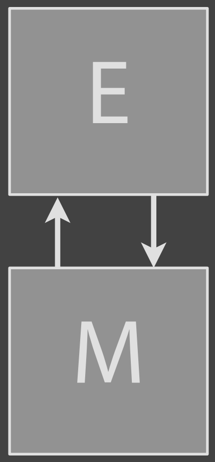
For gaussian mixture models, the algorithm of choice is expectation maximization.

# 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)

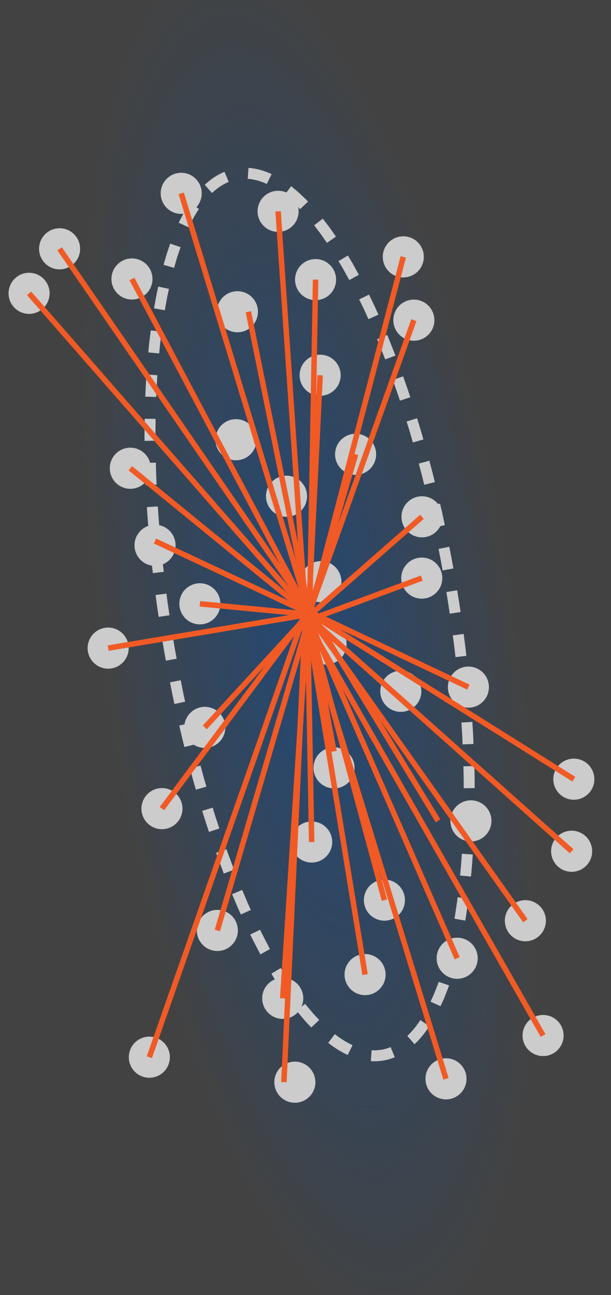
Saturday, August 4, 12

This is essentially a soft version of K-means.

E-step: creates soft assignment between every photon and every Gaussian in the mixture model
M-step: uses these assignments to maximize a likelihood function.
Parameters improve steadily until convergence.

Two issues: EM only finds locally optimal solutions. Also, it does not scale.

## Expectation maximization



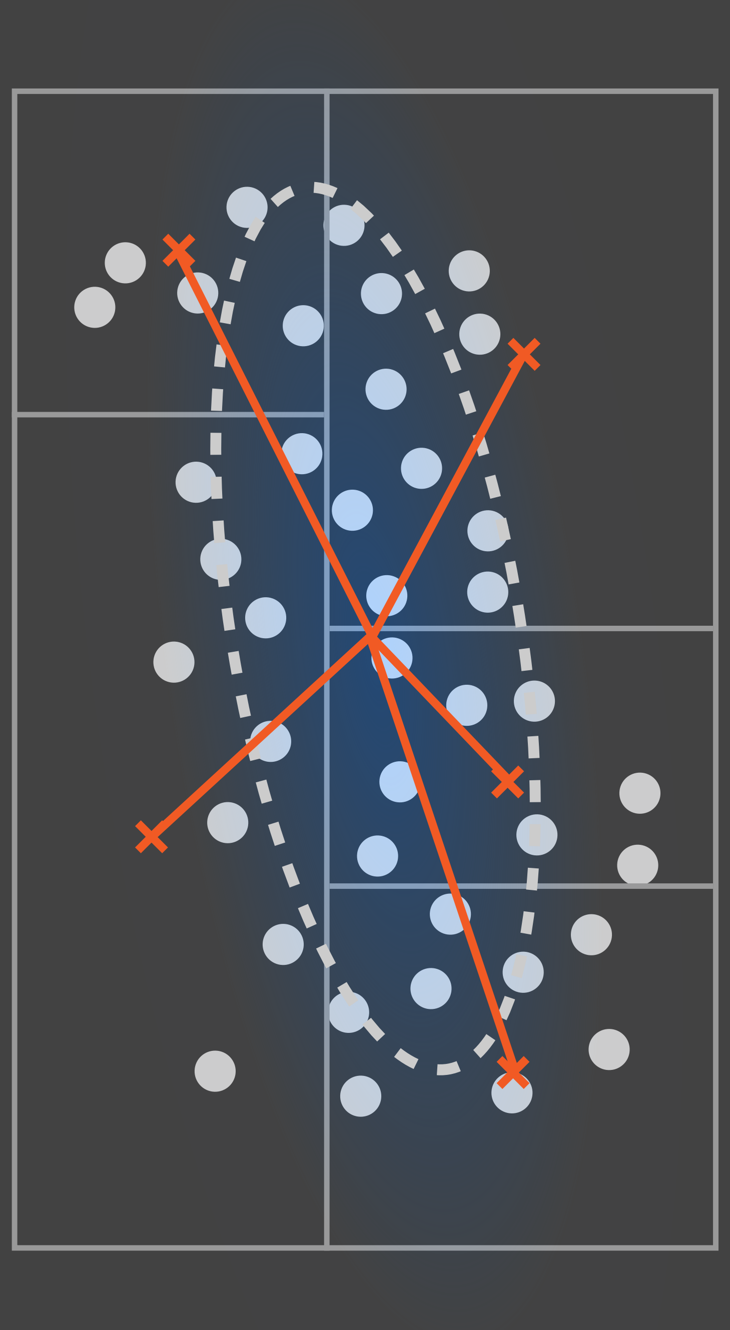
Accelerated EM by [Verbeek et al. 06]

Intuitive interpretation of EM:

Each photon exerts a “force” onto nearby Gaussian components, pulling them towards it as to be covered by the density function. This process is iterated until an equilibrium is reached.

To improve performance we rely on accelerated EM and make several modifications to make it scale to the problems of our size.

# Accelerated EM



**Stored cell statistics:**

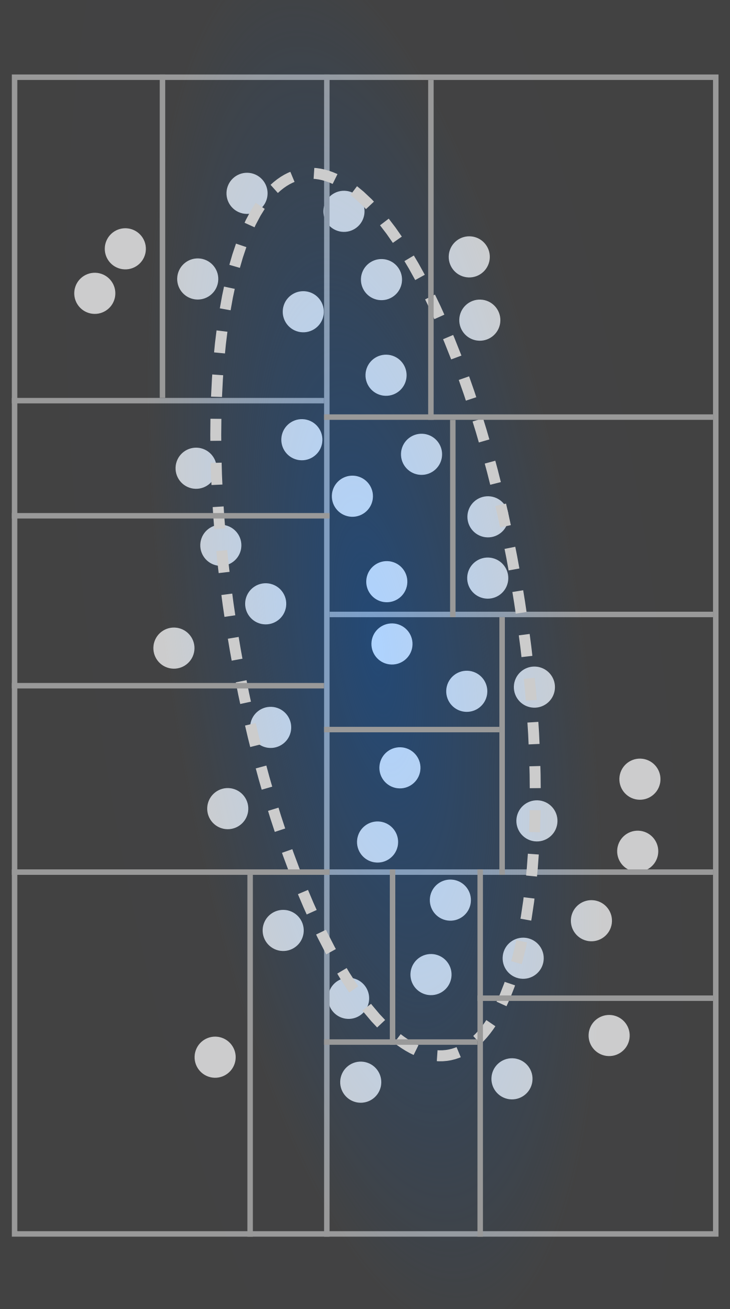
- photon count
- mean position
- average outer product

Saturday, August 4, 12

Accelerated EM: operates on cells instead of the individual data points. Each cell summarizes the photons that fall inside it using their count, mean, and average outer product (i.e. 0th, 1st, and 2nd order statistics).

The entire EM algorithm can be formulated only in terms of cell-Gaussian interactions, which is much faster.

## Progressive EM



**Stored cell statistics:**

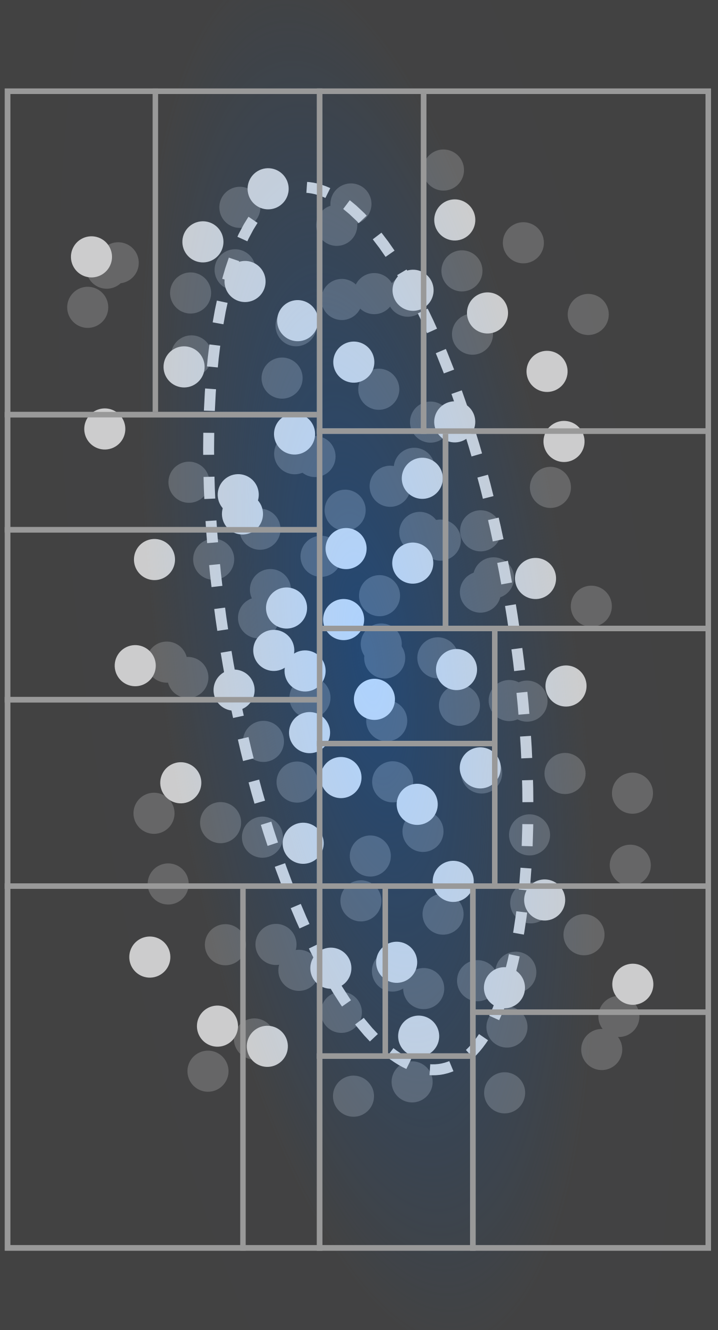
- photon count
- mean position
- average outer product

**Our modifications:**

- better cell refinement

Can also switch to increasingly finer cells partitions as the algorithm runs.

# Progressive EM



**Stored cell statistics:**

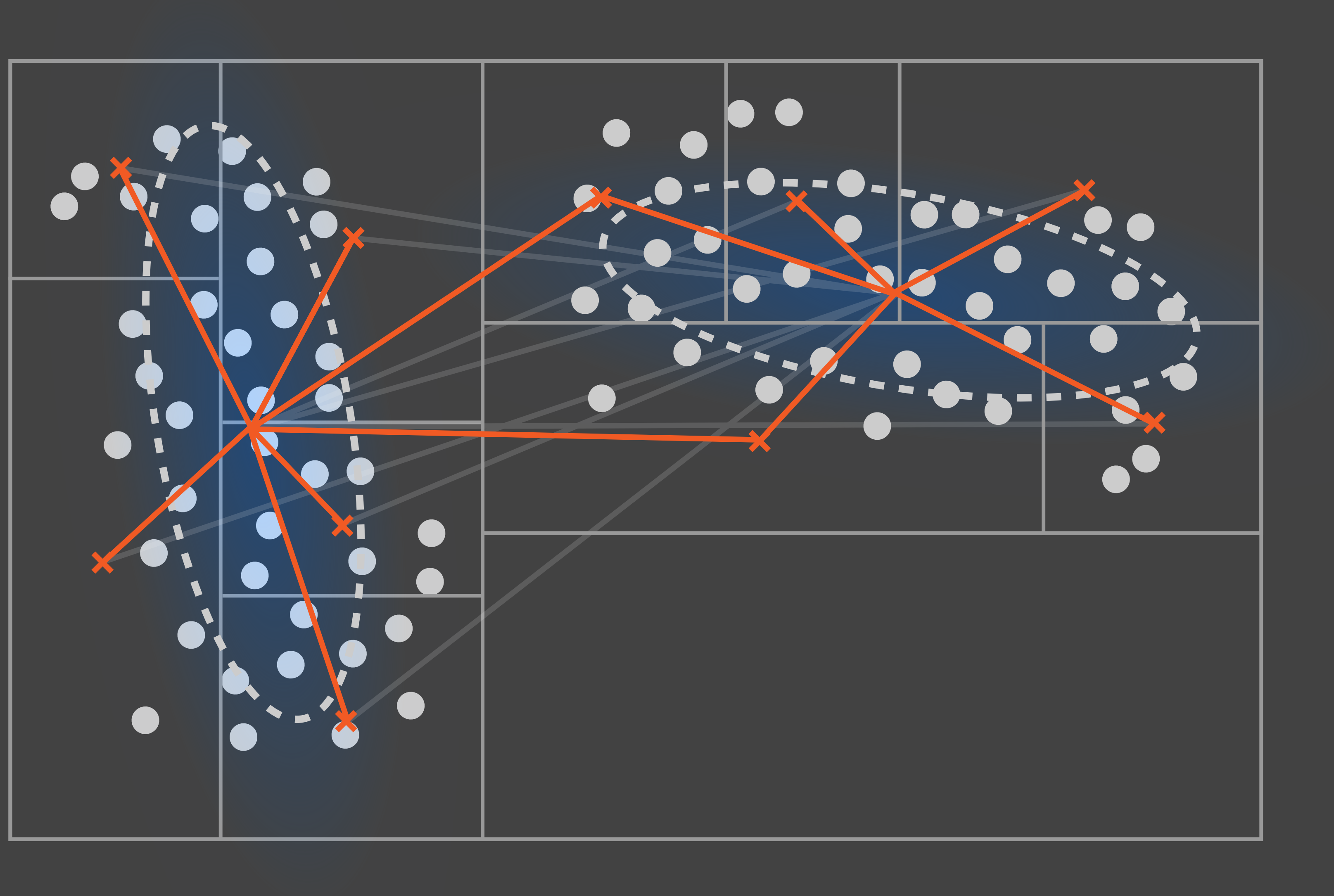
- photon count
- mean position
- average outer product

**Our modifications:**

- better cell refinement
- progressive photons shooting passes

– After storing photon statistics in cells, the photons themselves are not needed anymore and can be discarded.
– Can trace additional photons later on and incorporate them into the cell statistics, without having to store them.

# Progressive EM



**Stored cell statistics:**

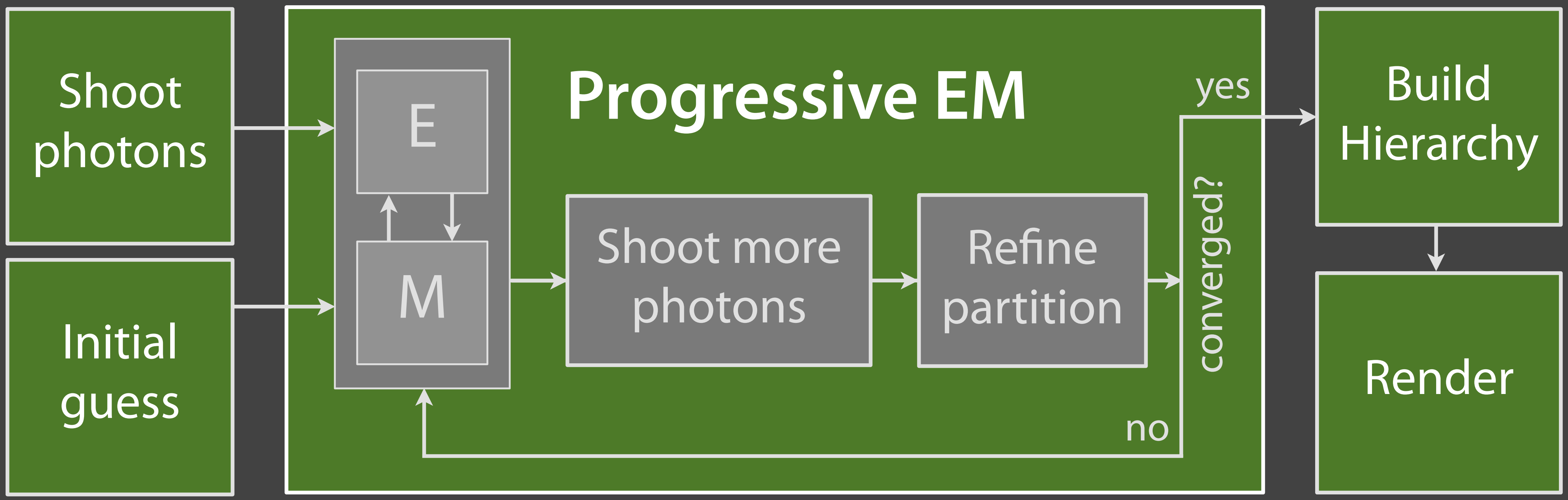
- photon count
- mean position
- average outer product

**Our modifications:**

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

Exponential decay in Gaussians -> we can cull away interactions between distant Gaussians and cells. This improves running time from O(n^2) to O(n log n)!

## Pipeline overview

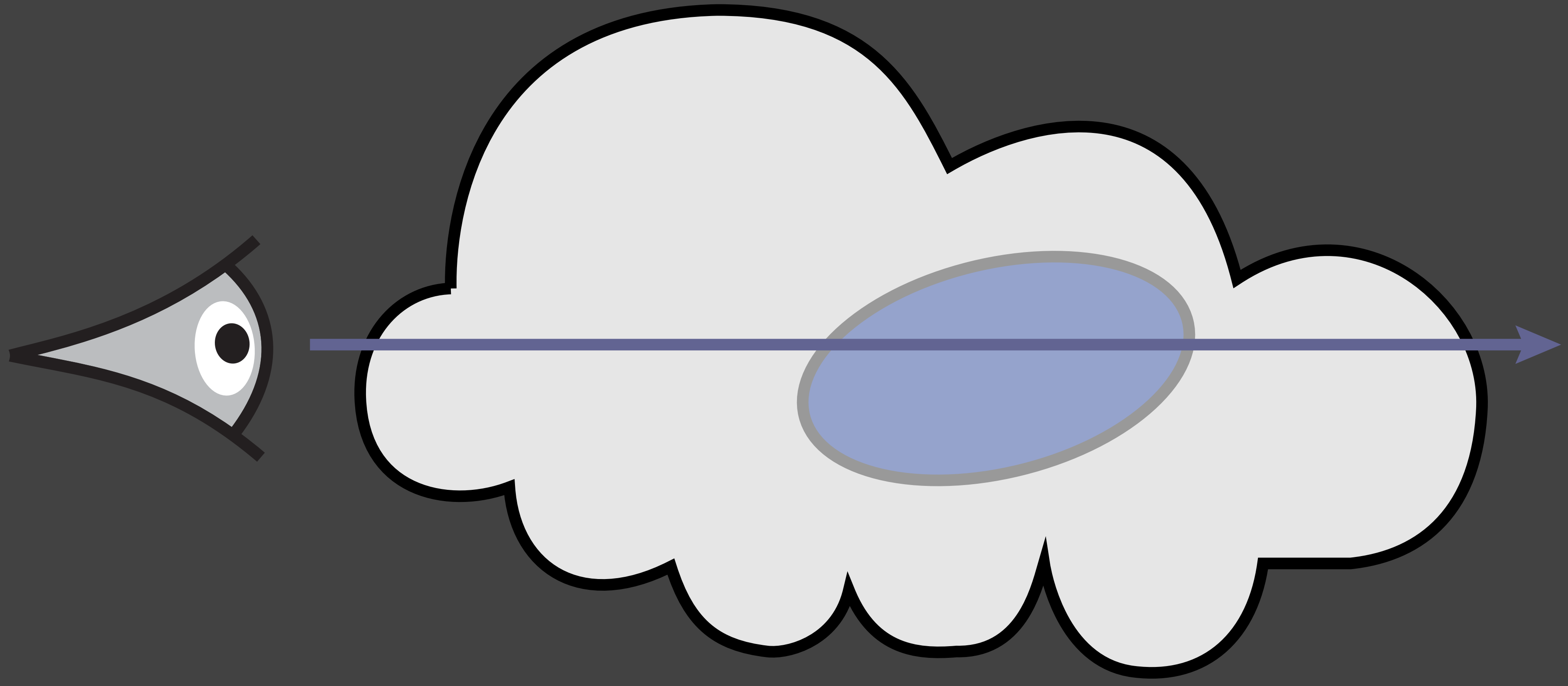


Saturday, August 4, 12

Pipeline diagram of the entire method.  Input: photon map + initial guess

Progressive EM is executed until convergence (and further photons may be traced to improve quality). Finally, a level of detail representation is created for rendering.

## Rendering



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

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

...

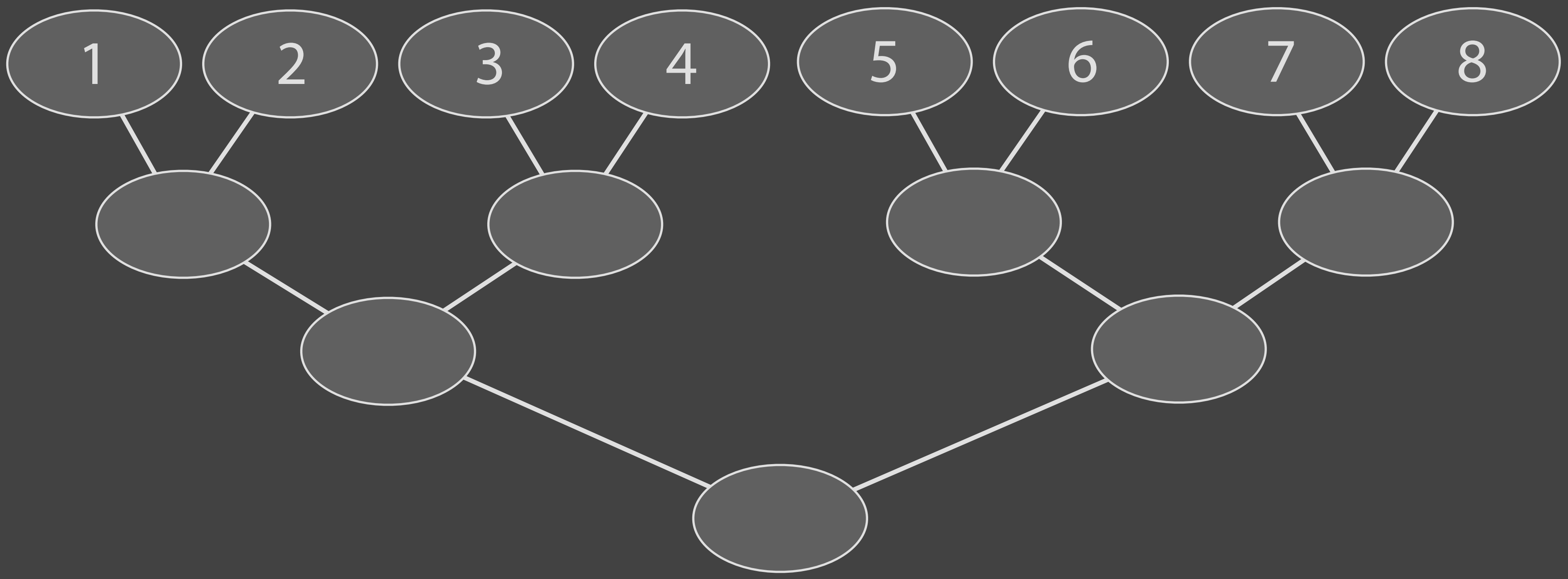
Saturday, August 4, 12

In theory, we could write a brute-force rendering algorithm, that just sums over the contributions of every single Gaussian to a camera ray. For homogeneous media, we have derived an analytic solution that gives the exact result in this case.
The problem with this brute-force summation is that when $k$ is large, each pixel takes a long time to render.

## Level of detail hierarchy

**Agglomerative construction:**

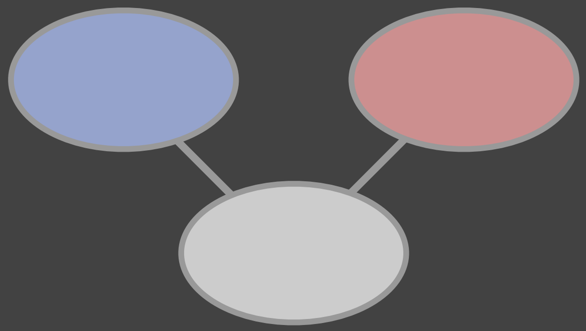
- Repeatedly merge nearby Gaussians based on their Kullback-Leibler divergence



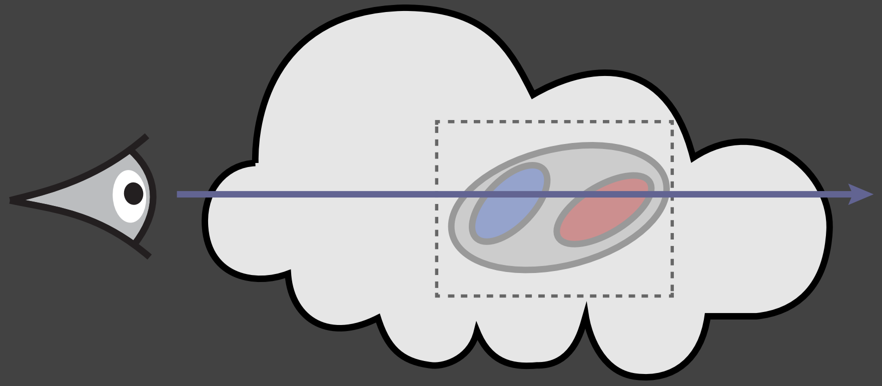
To get better performance, we turn the mixture model into a hierarchical representation similar to a MIP map.

## Rendering

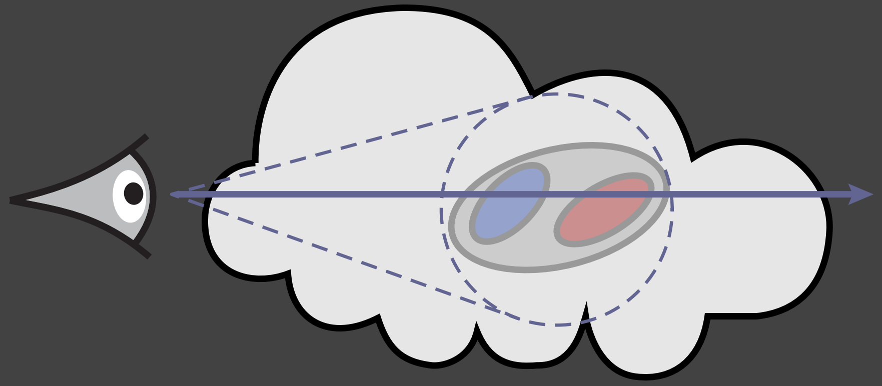
Example hierarchy:



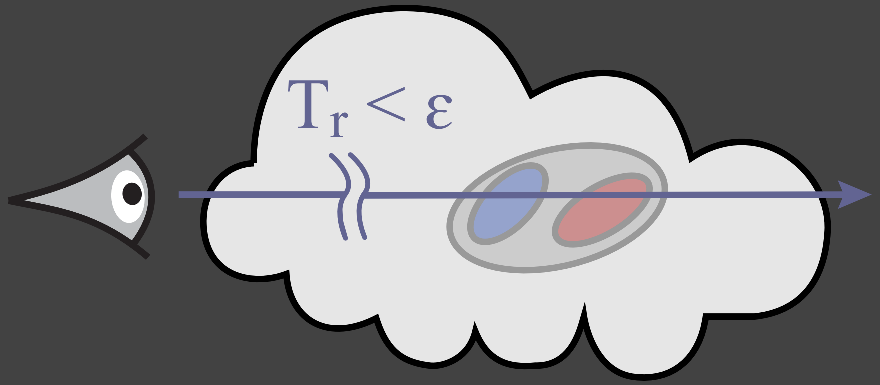
**Criterion 1:** bounding box intersected?



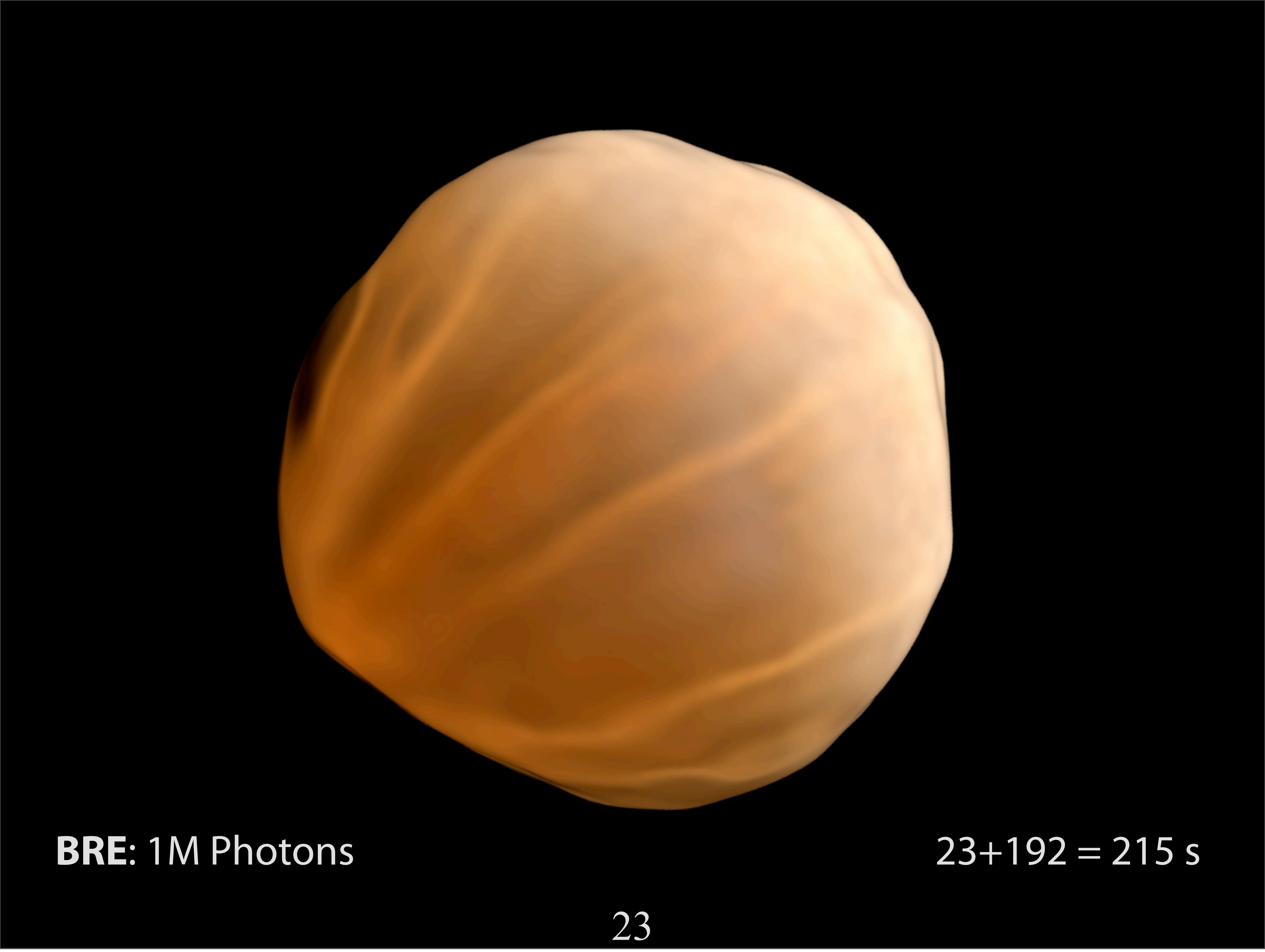
**Criterion 2:** solid angle large enough?



**Criterion 3:** attenuation low enough?

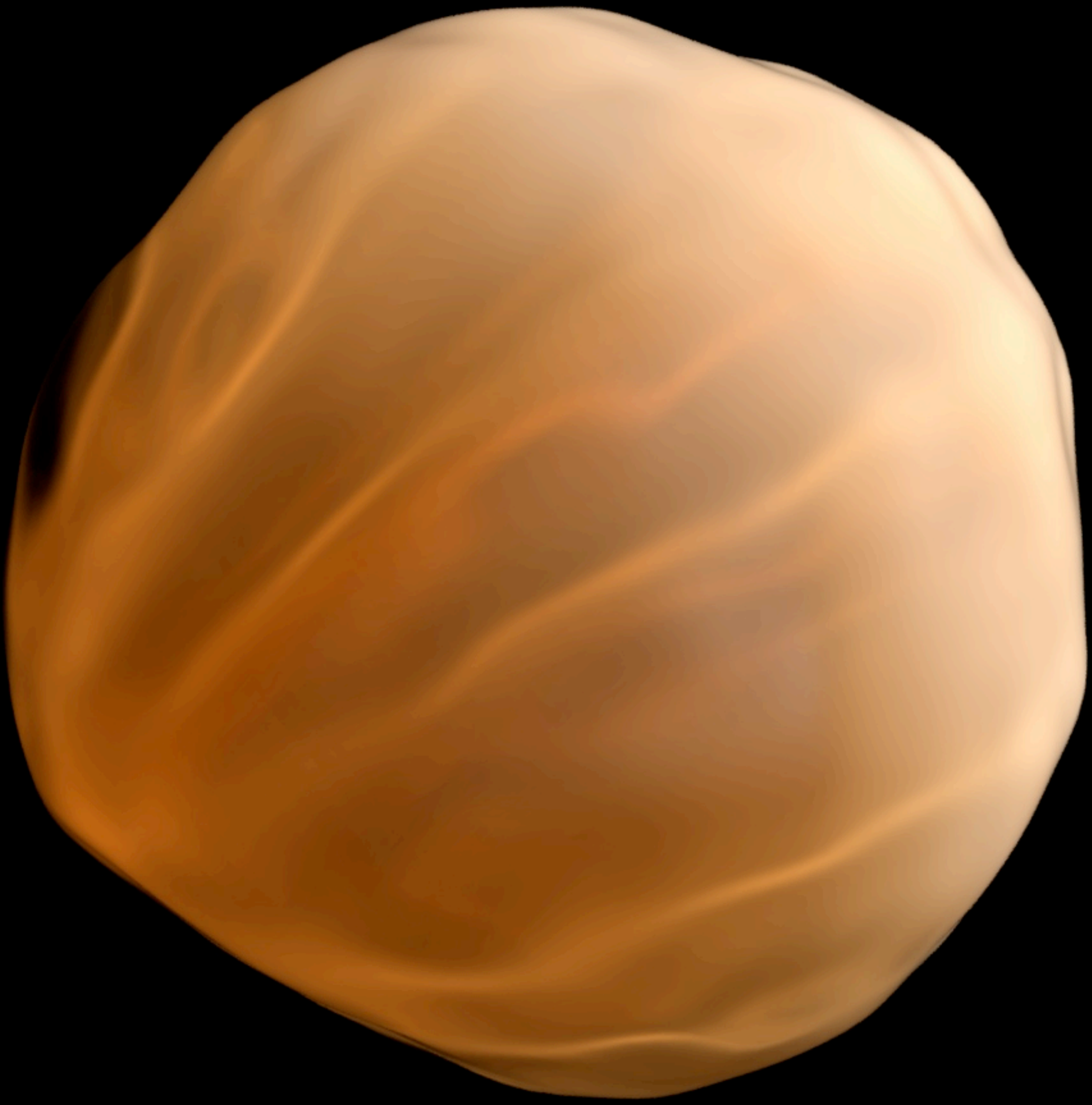


Can use lower resolutions when one of these three criteria is satisfied. This leads to a significantly shorter rendering times.



Examples:

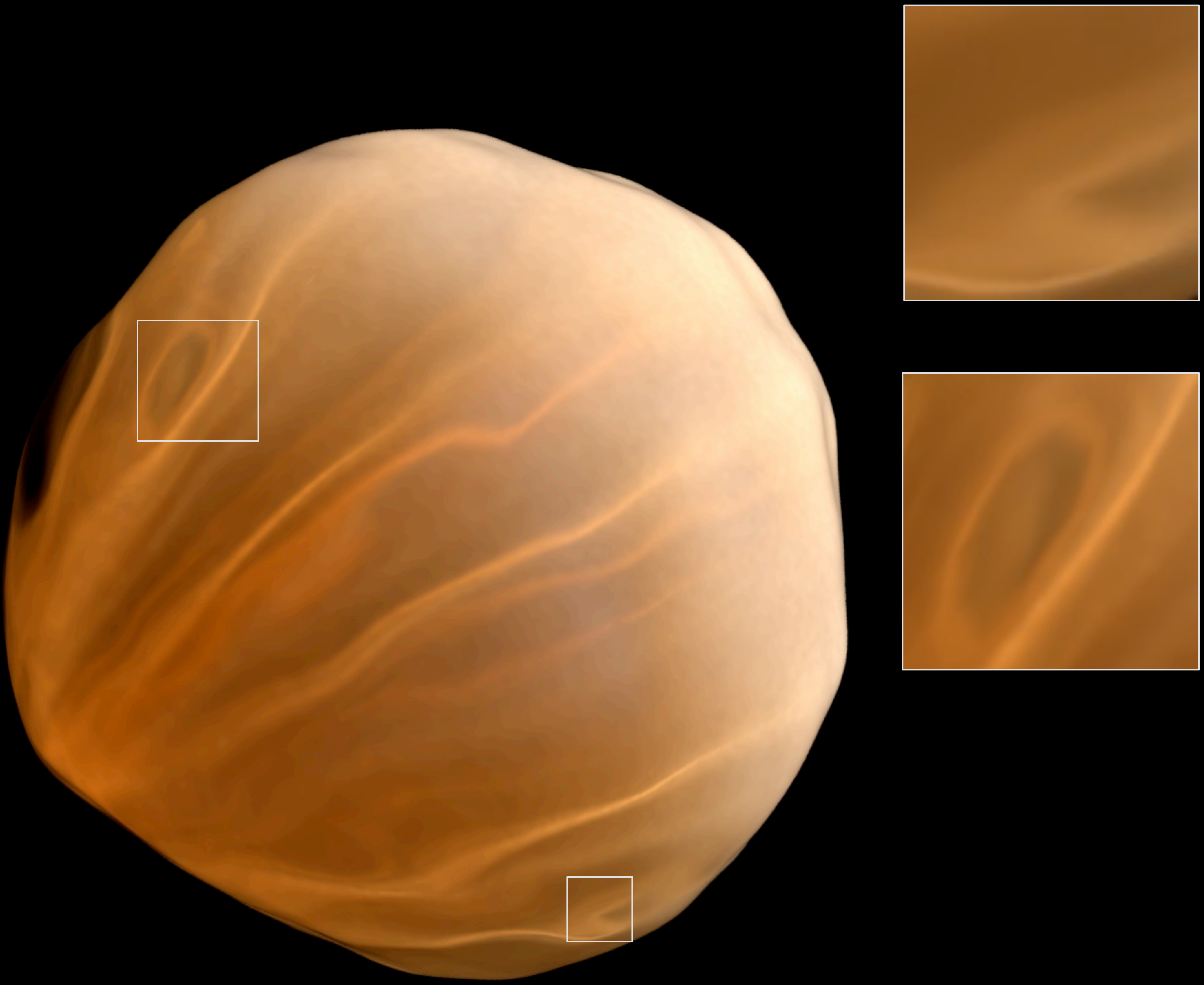
Bumpy sphere scene courtesy of Bruce Walter (BRE)



**Our method:** 4K Gaussians
(fit to 1M photons)

35+24 = 59 s
(3.6×)

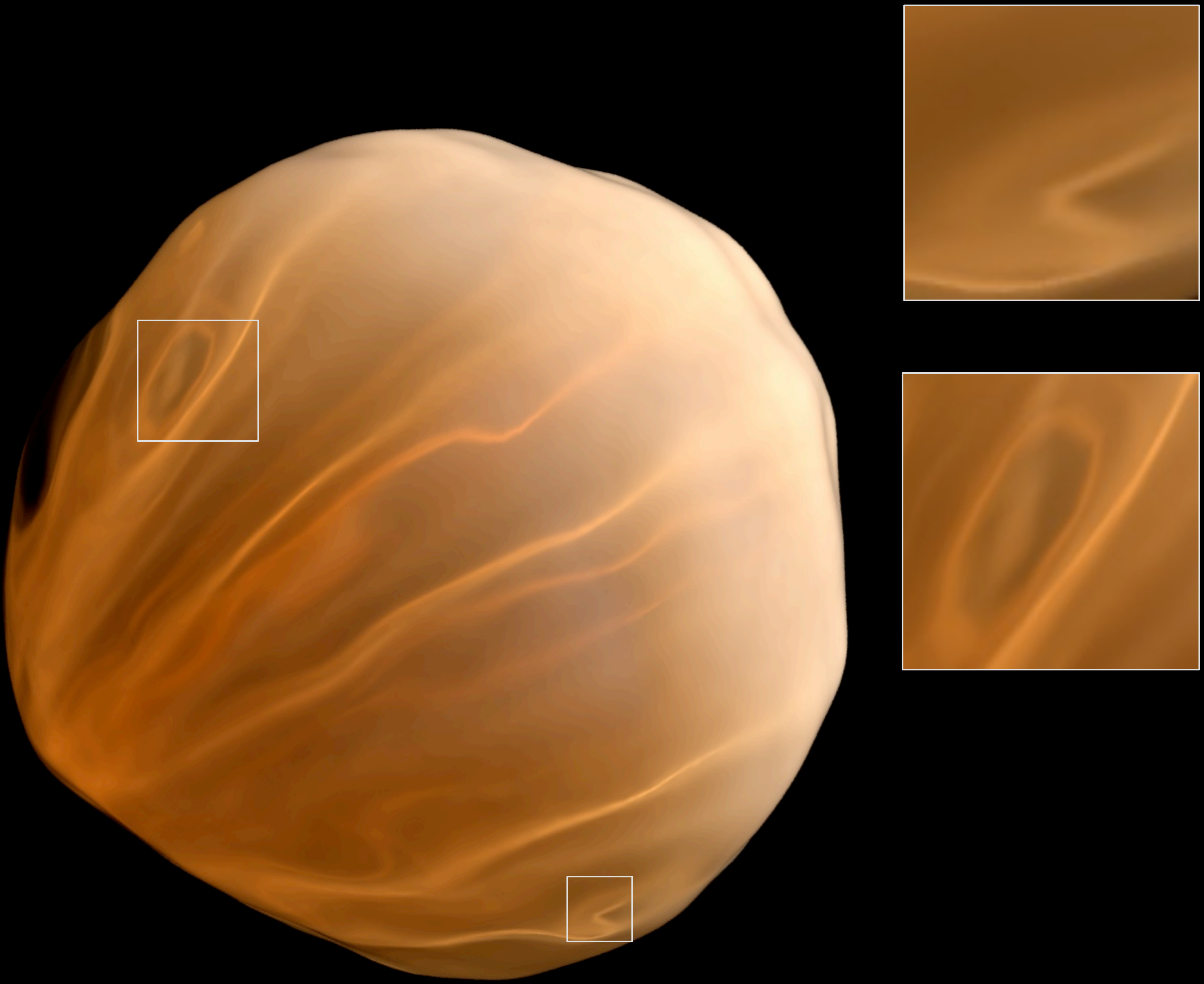
4K Gaussian fit using 1M photons as an input



**BRE:** 18M Photons

507+609 = 1116 s

BRE rendering with 18 million photons.



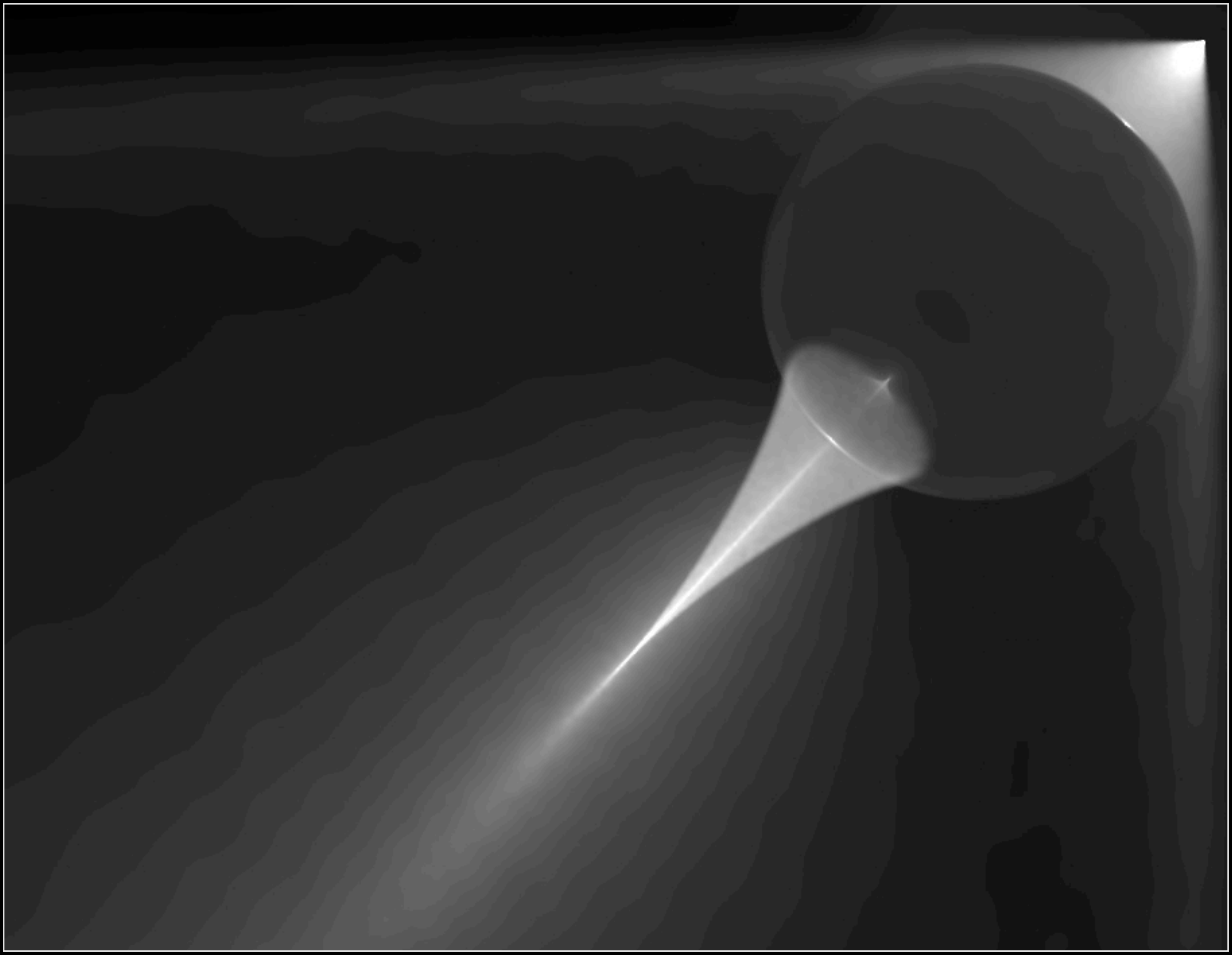
**Our method:** 64K Gaussians

(fit to 18M photons)

868+66 = 934 s

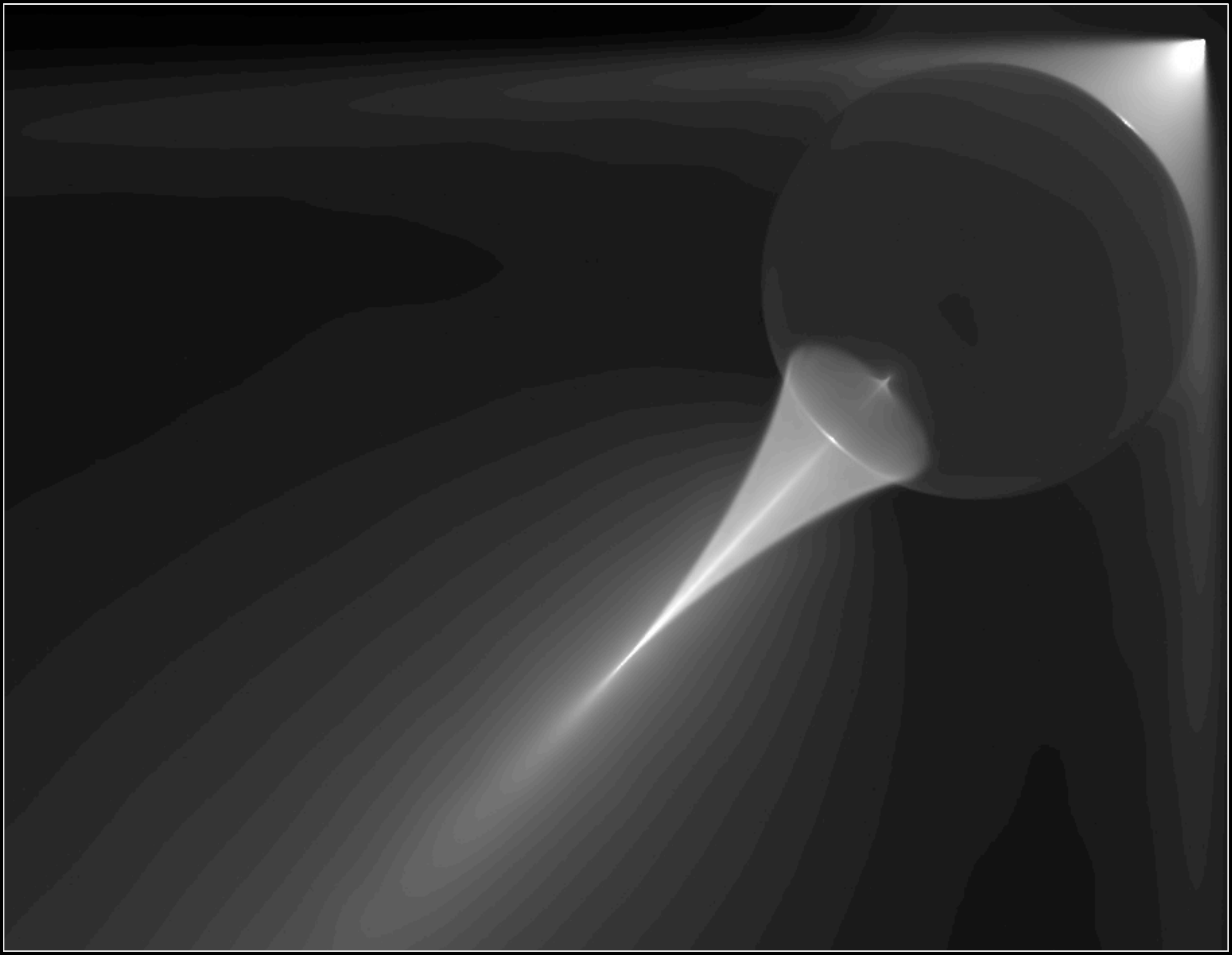
(1.2×)

64K Gaussian fit (to 18M photons)



**BRE**: 4M Photons

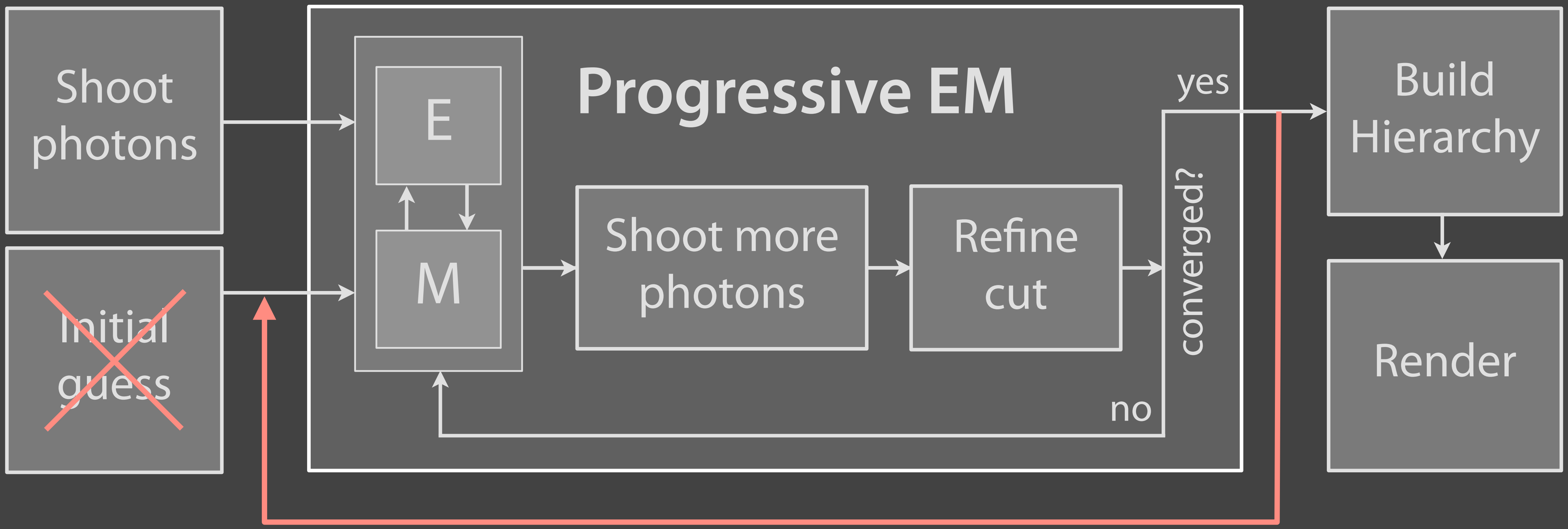
89 + 638 = 727 s



**Our method:** 16K Gaussians

330 + 127 = 457 s
(1.6×)

## Temporal Coherence



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

  $\longrightarrow$ Faster fitting, no temporal noise

Temporal coherence: can pass the solution of one frame as the initial guess of the next frame when rendering animations.

[ Video ]

Video showing the bumpy sphere scene with a rotating light source.

[ 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)

Realtime visualization: It is possible to code up the contribution from an anisotropic Gaussian in a simple GLSL shader so that large numbers of them can be drawn in realtime using splatting.

The storage requirements are tiny -- might be useful for games.

## 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