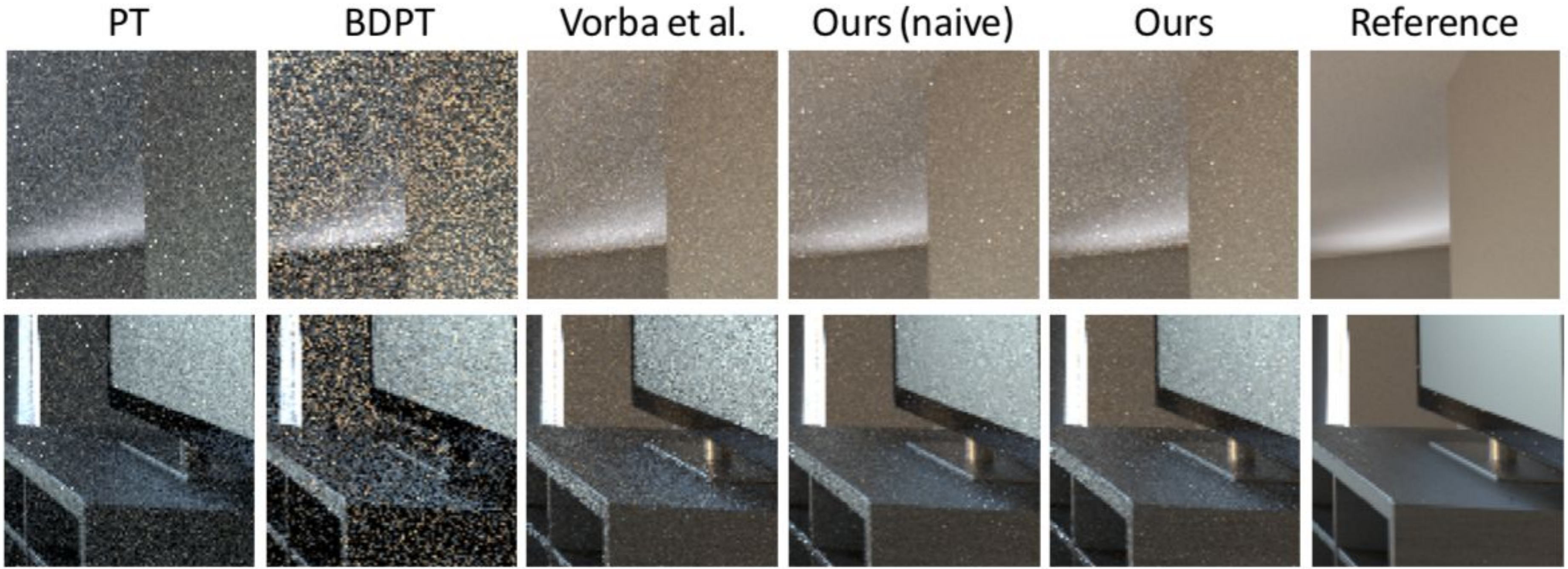
# Product Importance Sampling for Light Transport Guiding

**Herholtz et al. 2016**

presenter: Eunhyouk Shin

# It is all about convergence



# Contents

- Review on importance sampling
- Light transport guiding techniques
- Gaussian Mixture Model & EM
- Process overview
- Results & Discussion

# Source Materials

**Product Importance Sampling for Light Transport Path Guiding**

Sebastian Herholz[1] Oskar Elek[2] Jiří Vorba[2,3] Hendrik Lensch[1] Jaroslav Křivánek[2]

[1]Tübingen University [2]Charles University Prague [3]Weta Digital

[EUROGRAPHICS 2016]

- Main paper for this presentation

**On-line Learning of Parametric Mixture Models for Light Transport Simulation**

Jiří Vorba[1*] Ondřej Karlík[1*] Martin Šik[1*] Tobias Ritschel[2†] Jaroslav Křivánek[1‡]

[1]Charles University in Prague [2]MPI Informatik, Saarbrücken

[SIGGRAPH 2014]

- Baseline technology
- Useful presentation slides from the authors

# Importance Sampling

# Rendering Equation

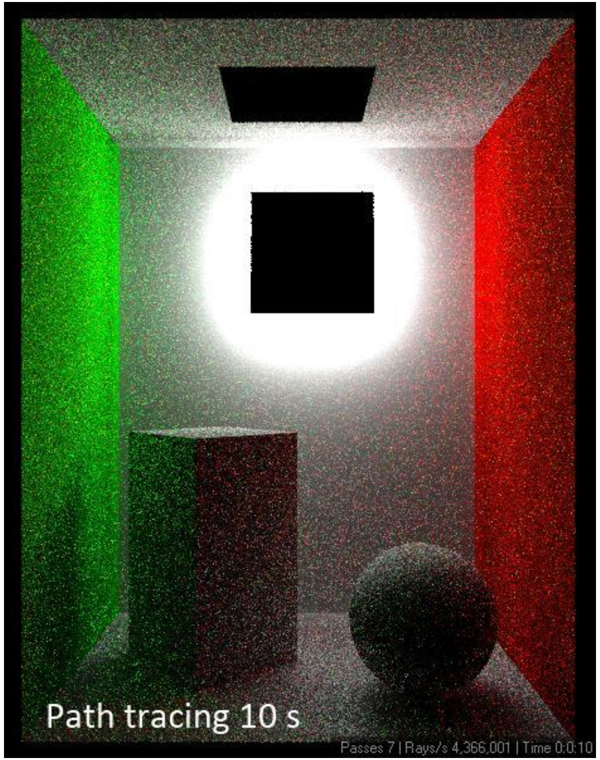
$$L(\mathbf{x}, \omega_o) = L_E(\mathbf{x}, \omega_o) + L_R(\mathbf{x}, \omega_o)$$

$$L_R = \int_\Omega \rho(\mathbf{x}, \omega_o, \omega_i) \cdot L(\mathbf{x}, \omega_i) \cdot \cos\theta \, d\omega_i$$

- Direct analytic integration is virtually impossible
- Recursive, due to the radiance term in the integrand

# Monte Carlo Ray Tracing

$$\widehat{L}_{\mathrm{R}} = \frac{\rho(\mathbf{x}, \omega_{\mathrm{o}}, \omega_{\mathrm{i}}) \cdot L(\mathbf{x}, \omega_{\mathrm{i}}) \cdot \cos\theta}{p(\omega_{\mathrm{i}})}$$




- Random sample direction from hemisphere to cast ray recursively
- Unbiased, even if sampling is not uniform

# Importance Sampling

Better to be...

$$p(\omega_i) \propto \rho(\mathbf{x}, \omega_o, \omega_i) \cdot \boxed{L(\mathbf{x}, \omega_i)} \cdot \cos\theta$$

- Lower variance when PDF is close to integrand distribution
- i.e. make more path that contributes more to radiance (light transport guiding)
- How can we make a good estimate for the integrand distribution?
  - BRDF (given)
  - **Illumination (unknown)**

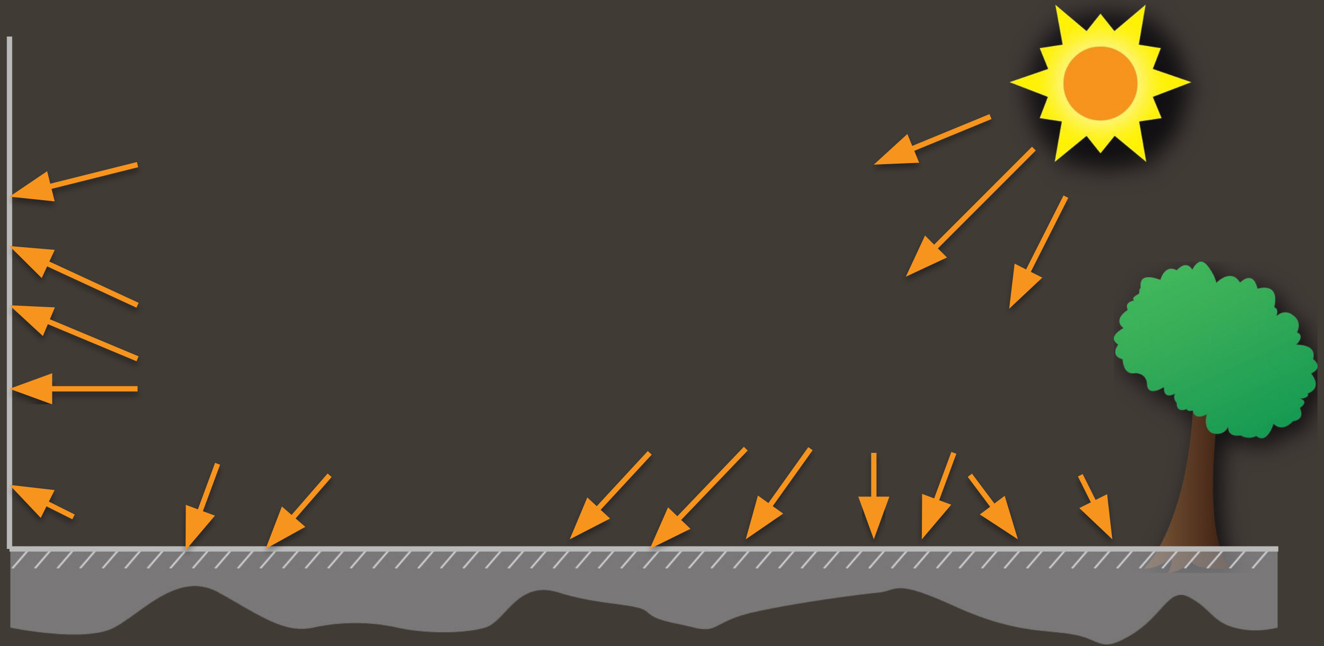
# Light Transport Guiding Techniques

**(slides from Vorba et al.)**

# Previous work

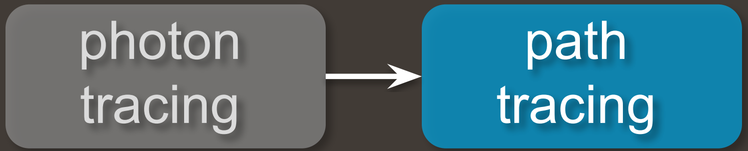
- Jensen *[1995]*

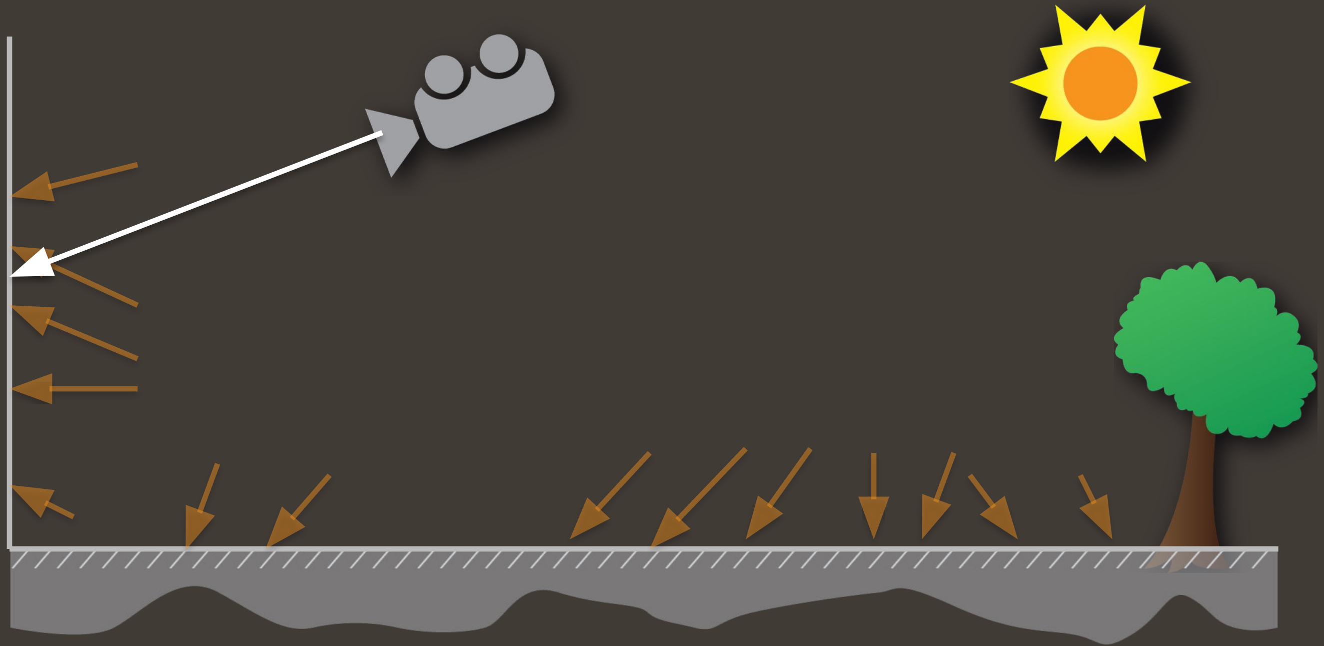
photon tracing



# Previous work

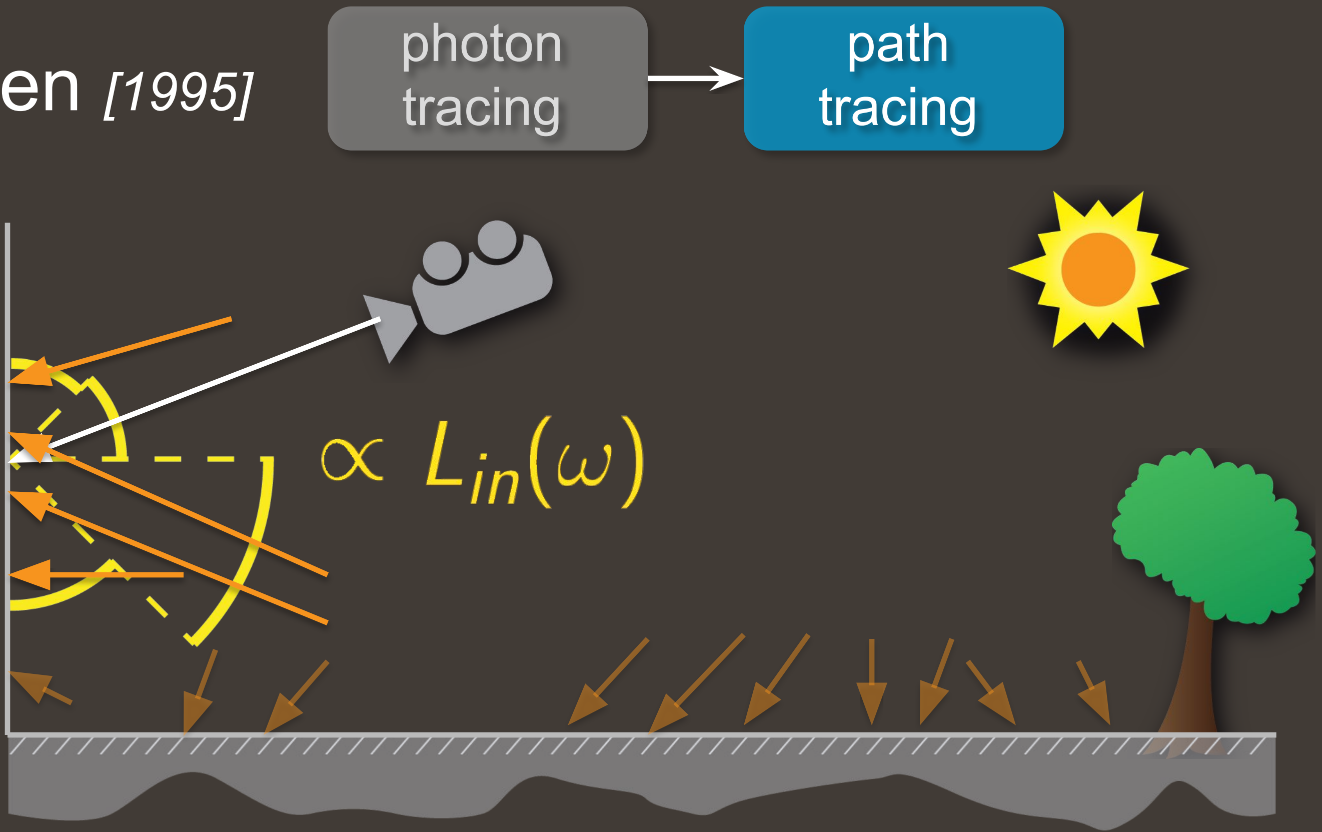
- Jensen *[1995]*



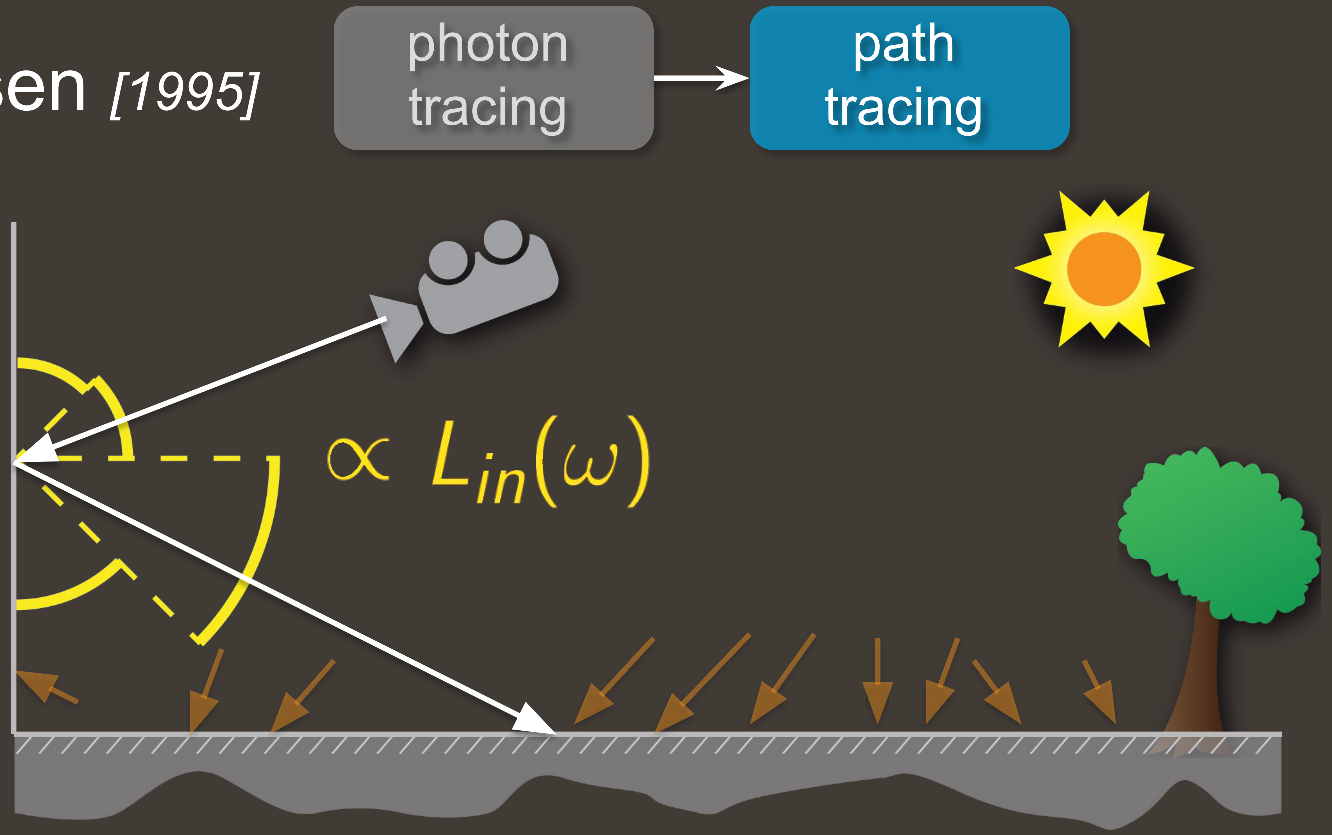
# Previous work

- Jensen *[1995]*



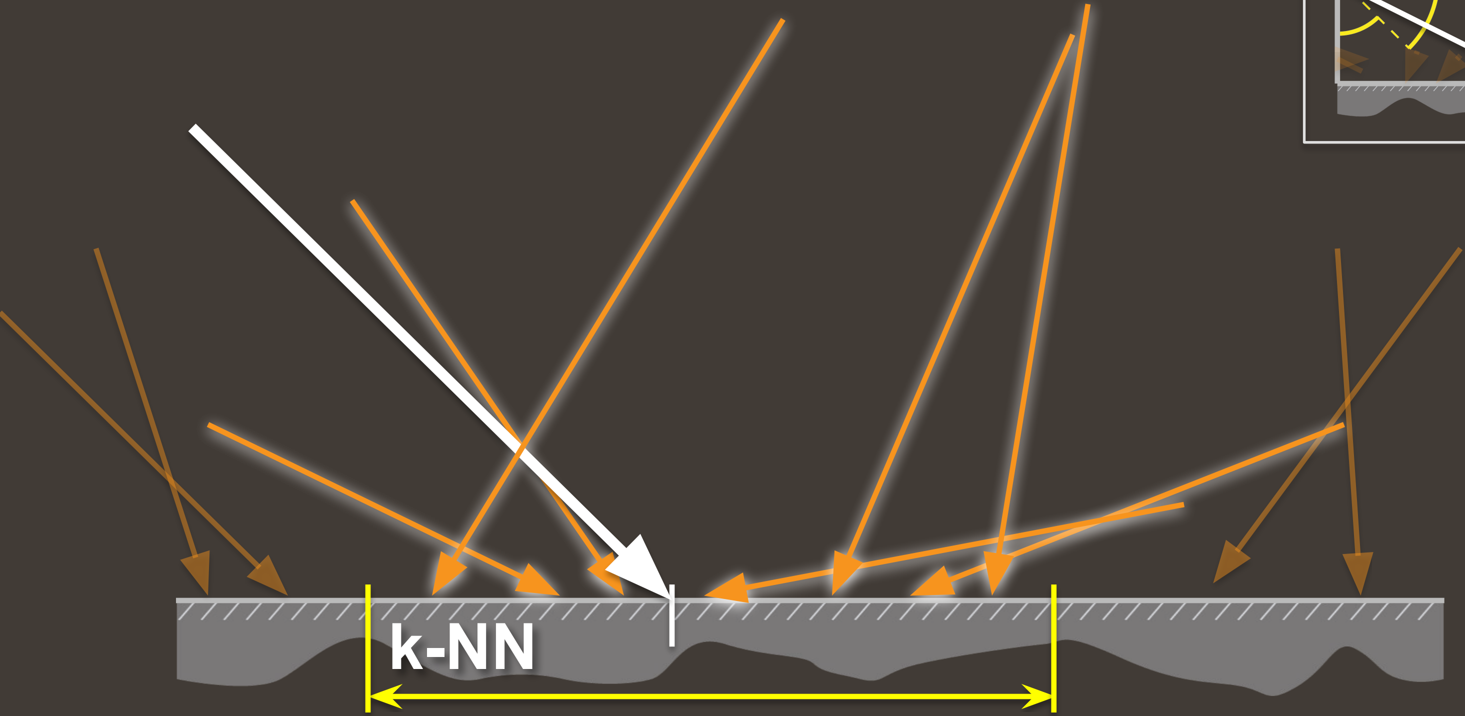
# Previous work

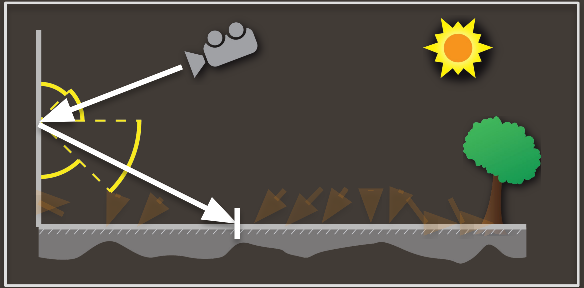
- Jensen *[1995]*



# Previous work

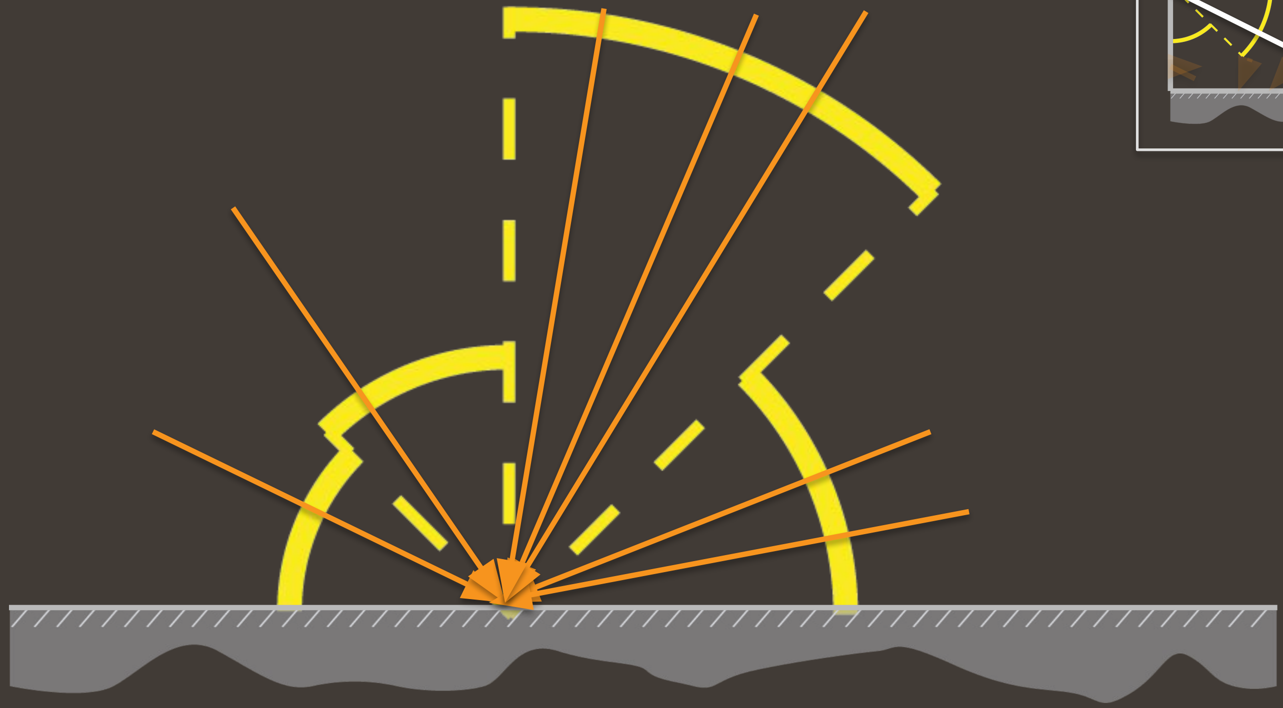
- Jensen *[1995]*: reconstruction

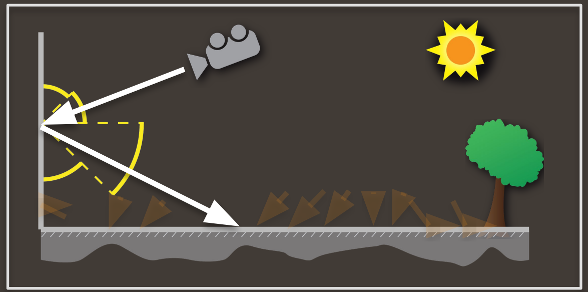


# Previous work

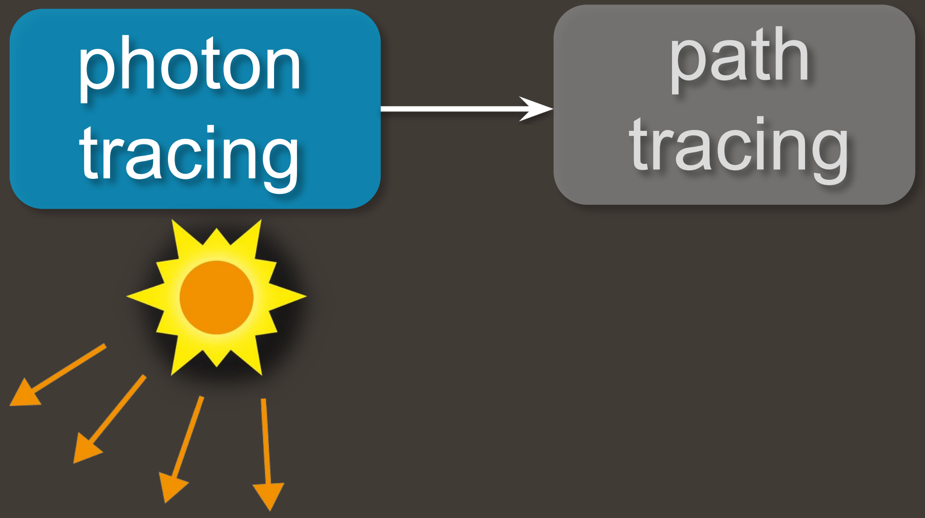
- Jensen *[1995]*: reconstruction





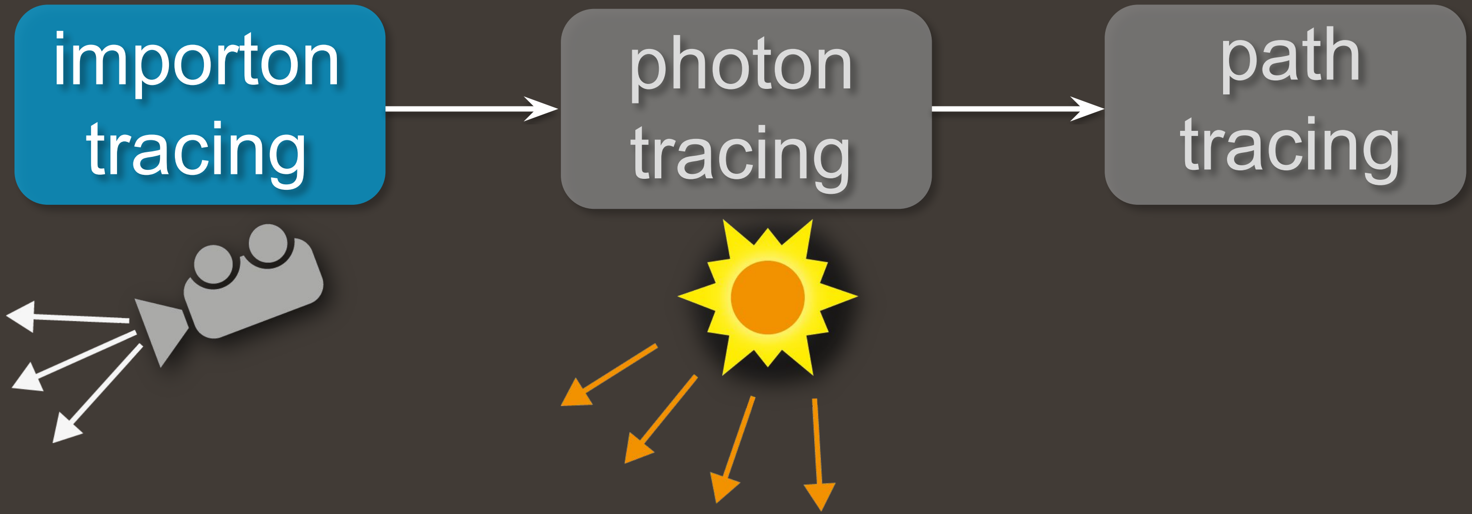
# Previous work

- Peter and Pietrek *[1998]*



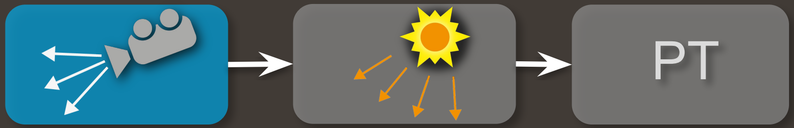
# Previous work

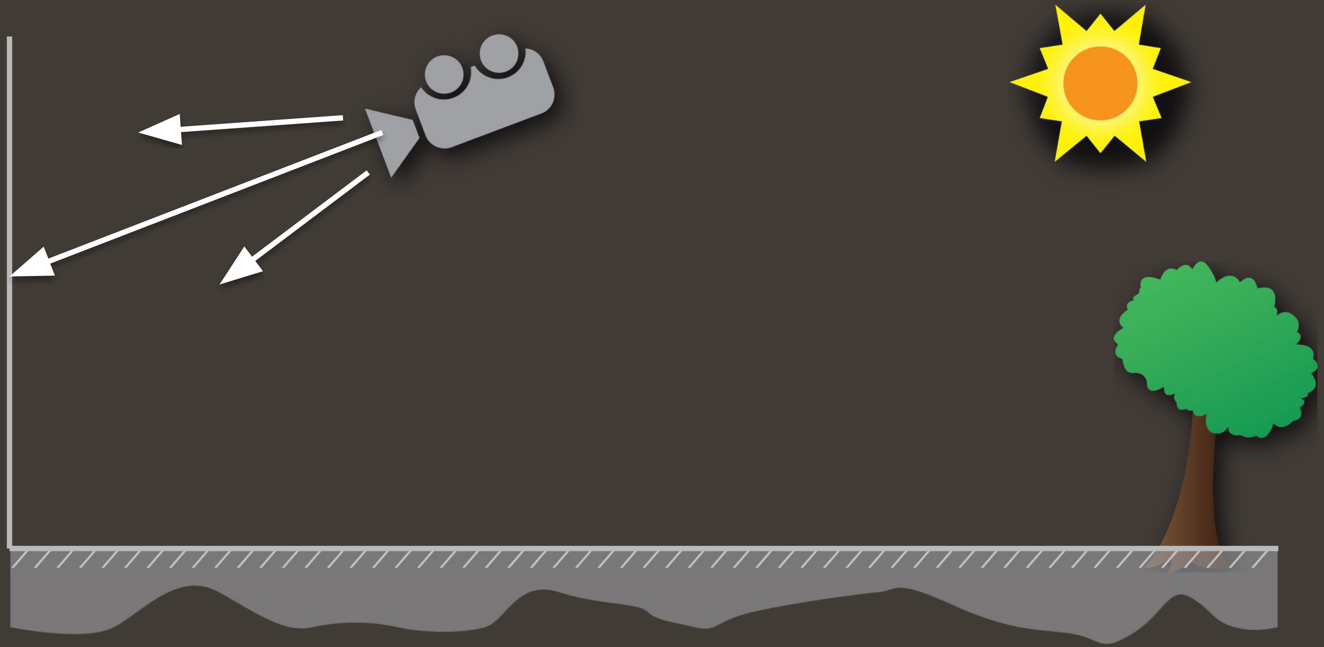
- Peter and Pietrek *[1998]*



# Previous work

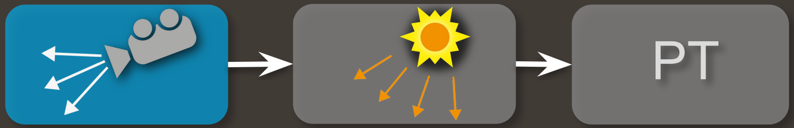
- Peter and Pietrek *[1998]*

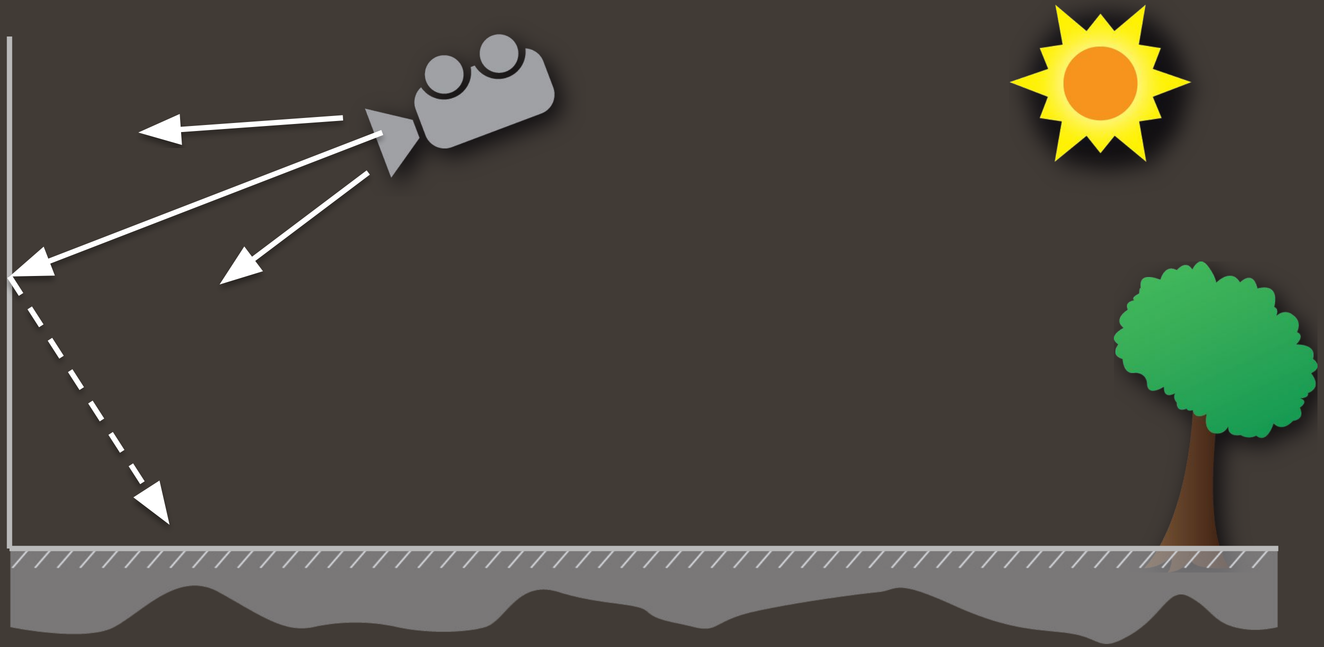


# Previous work

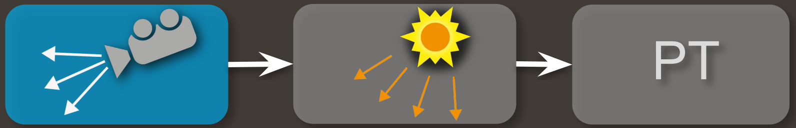
- Peter and Pietrek *[1998]*

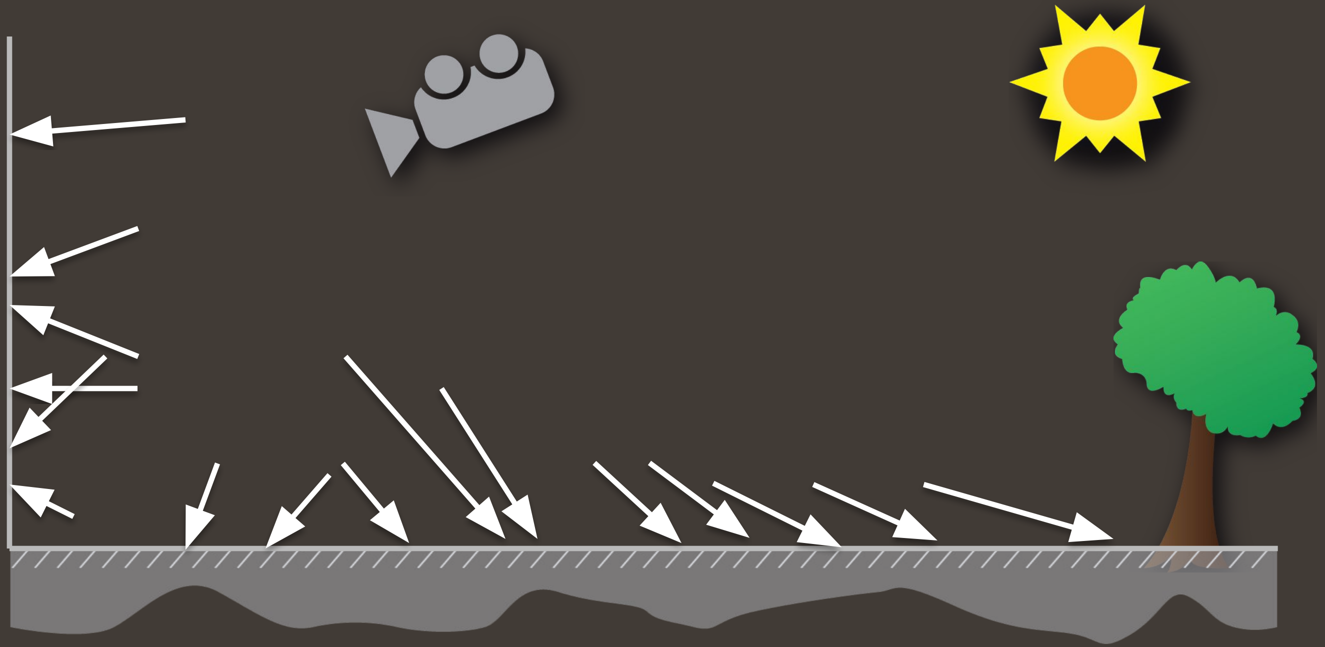


# Previous work

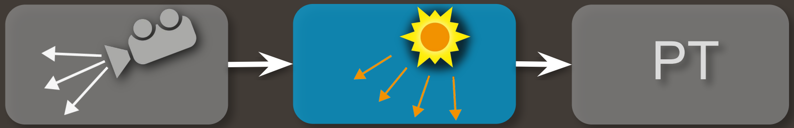
- Peter and Pietrek *[1998]*

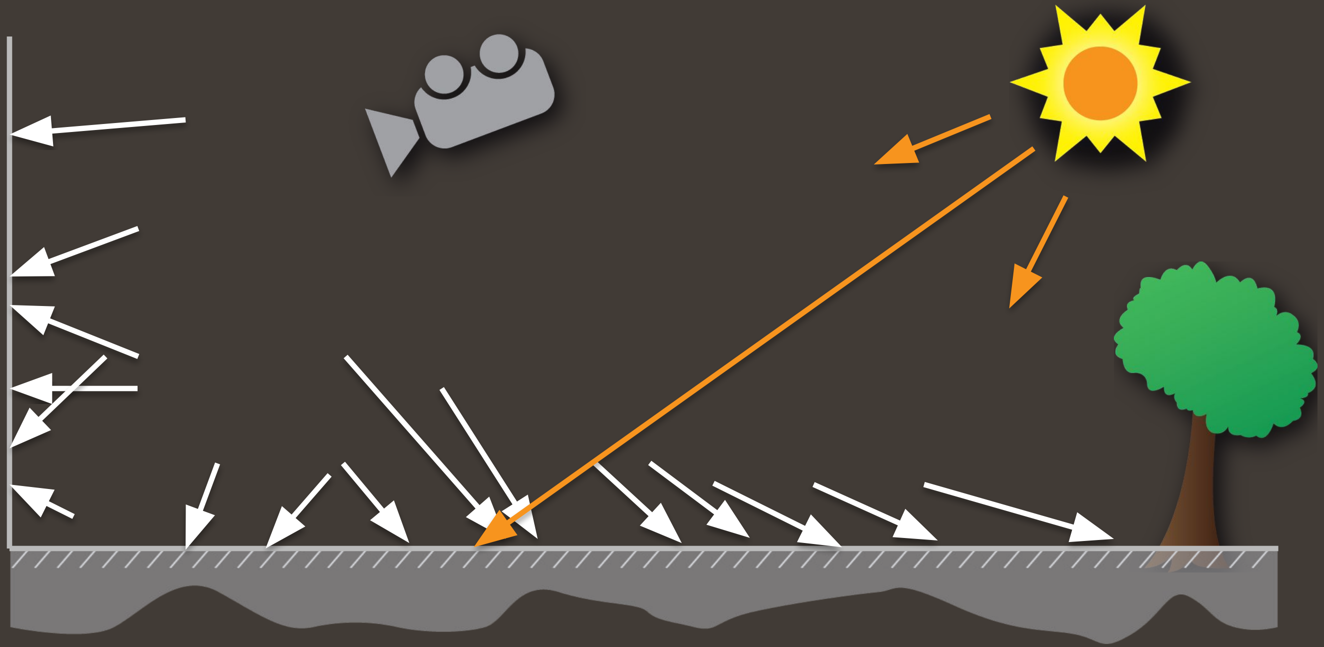


# Previous work

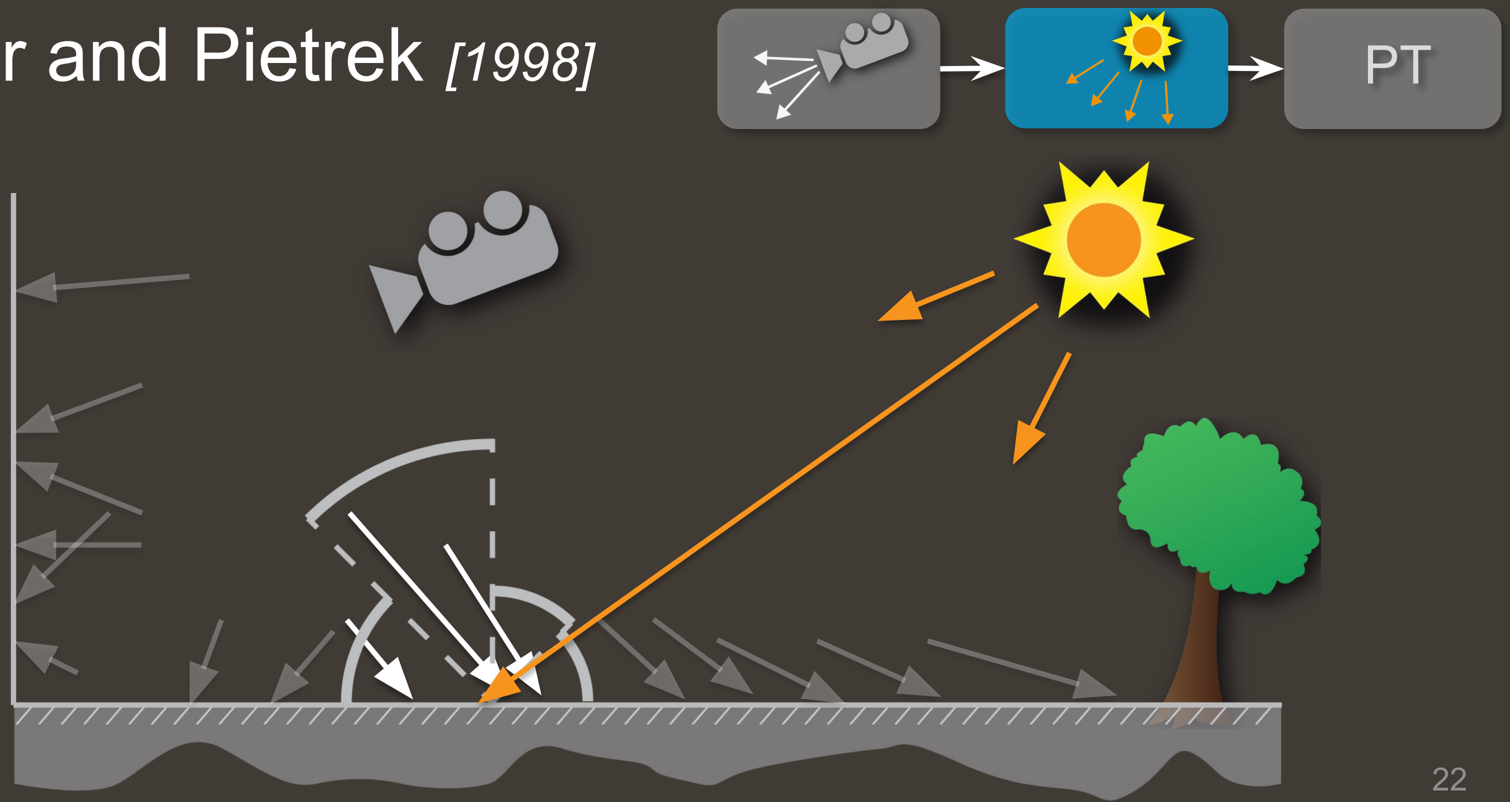
- Peter and Pietrek *[1998]*



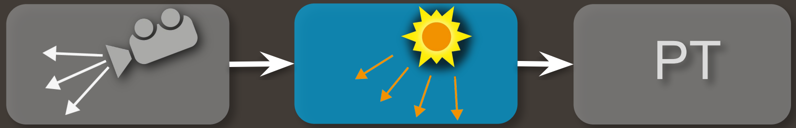
# Previous work

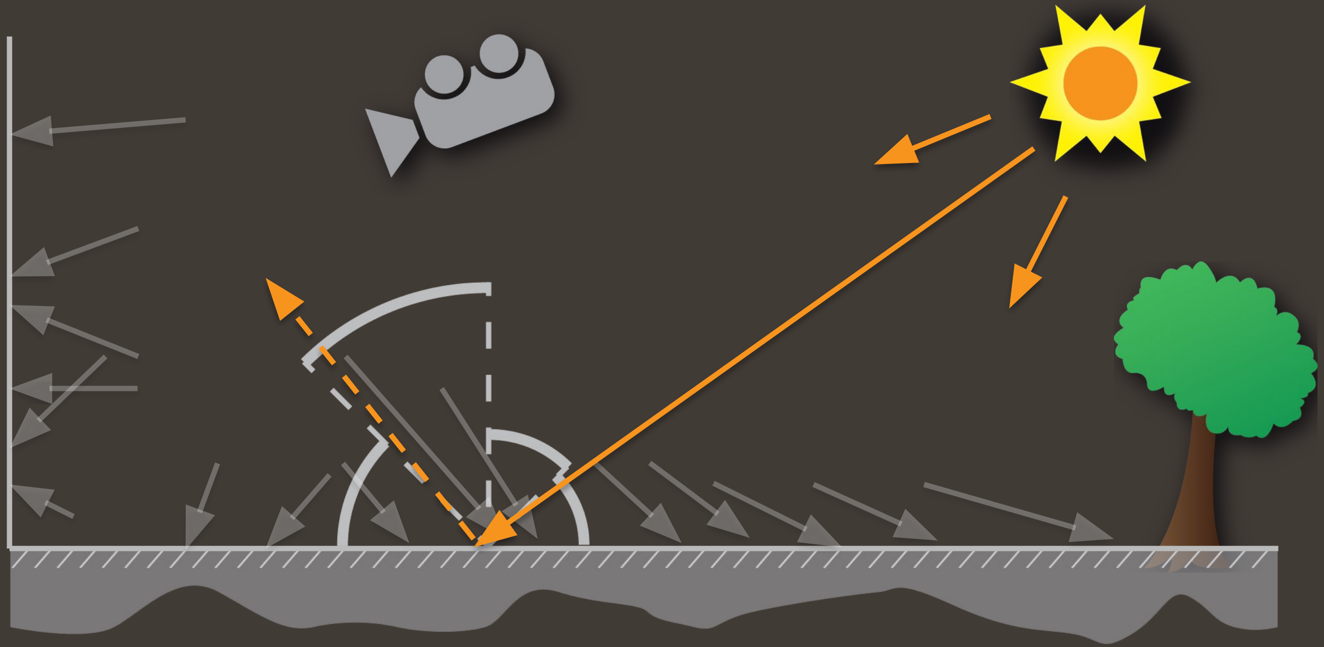
- Peter and Pietrek *[1998]*



# Previous work

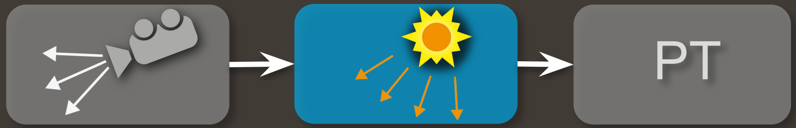
- Peter and Pietrek *[1998]*

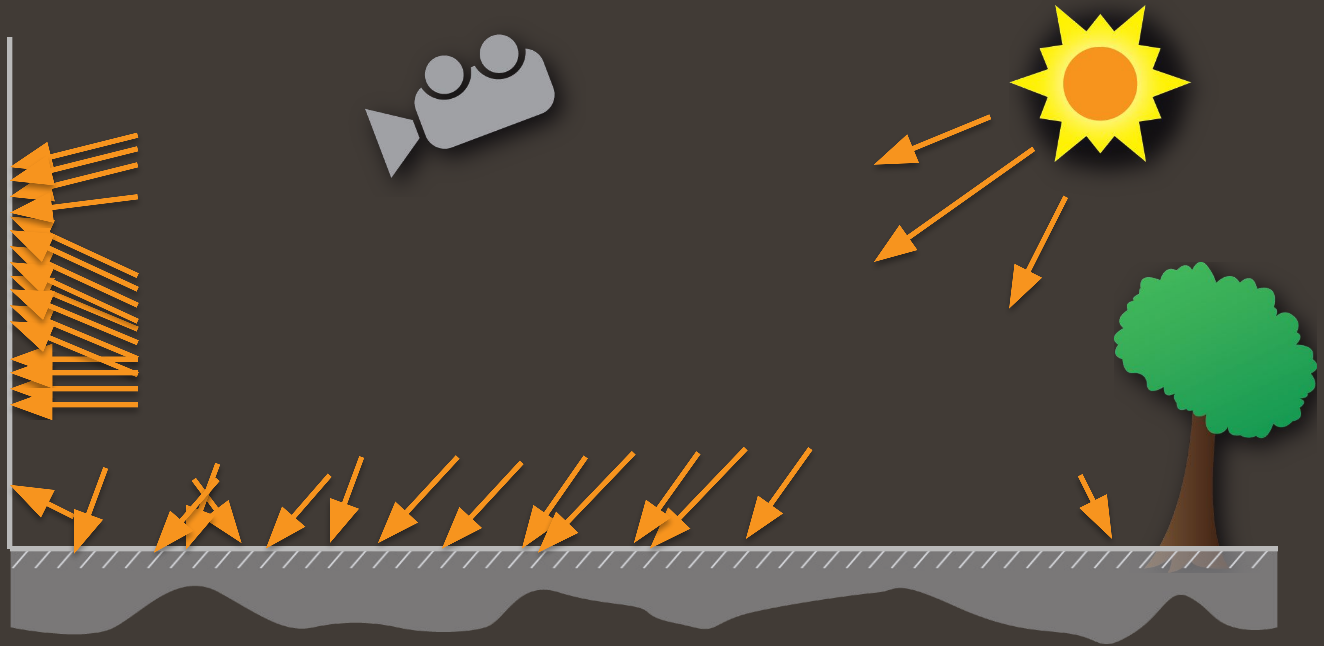


# Previous work

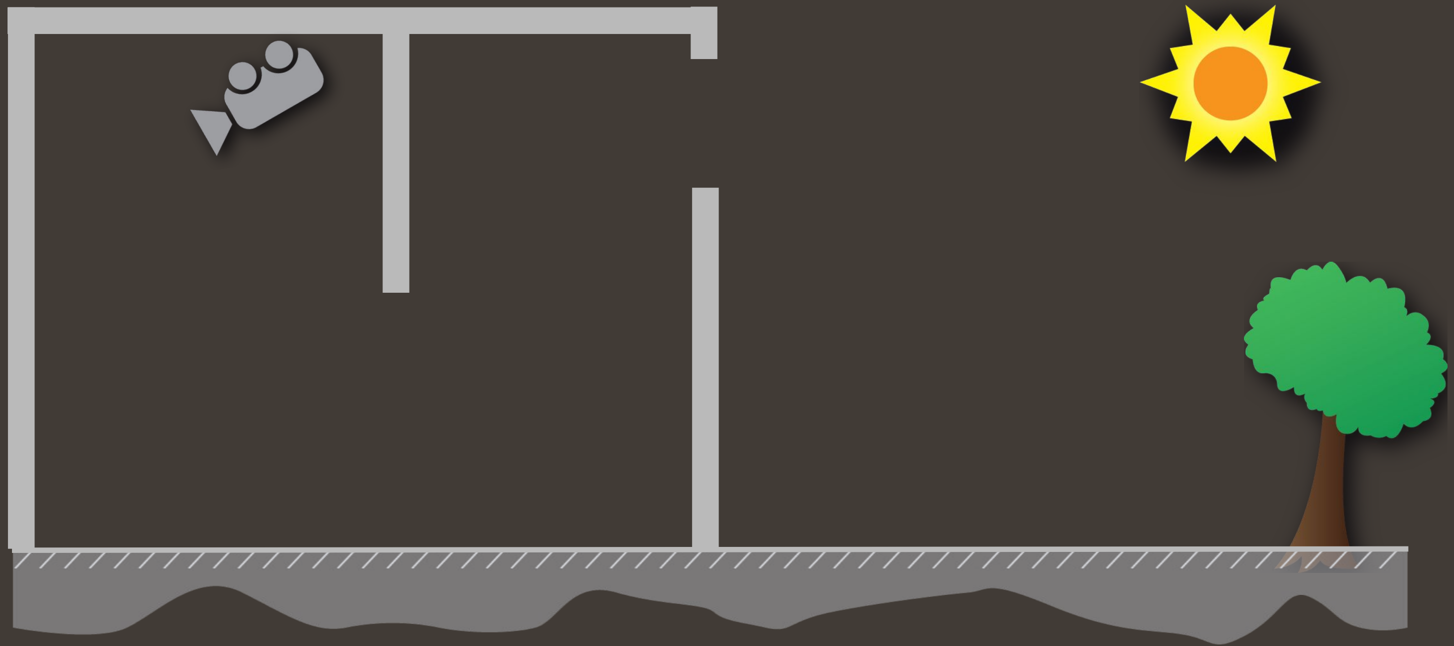
- Peter and Pietrek *[1998]*



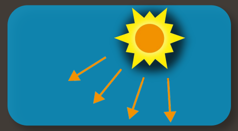



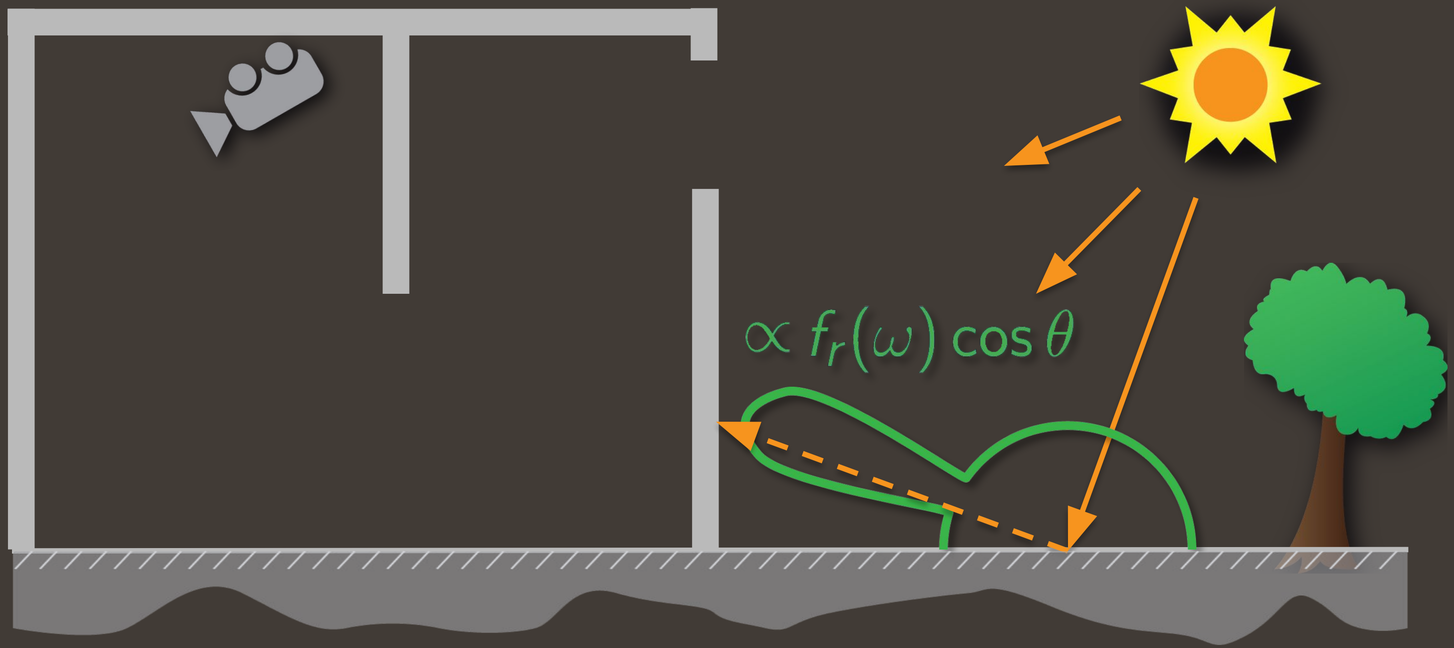
# Limitations of previous work

- Bad approximation of $L_{in}(\omega)$ in complex scenes

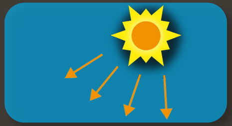


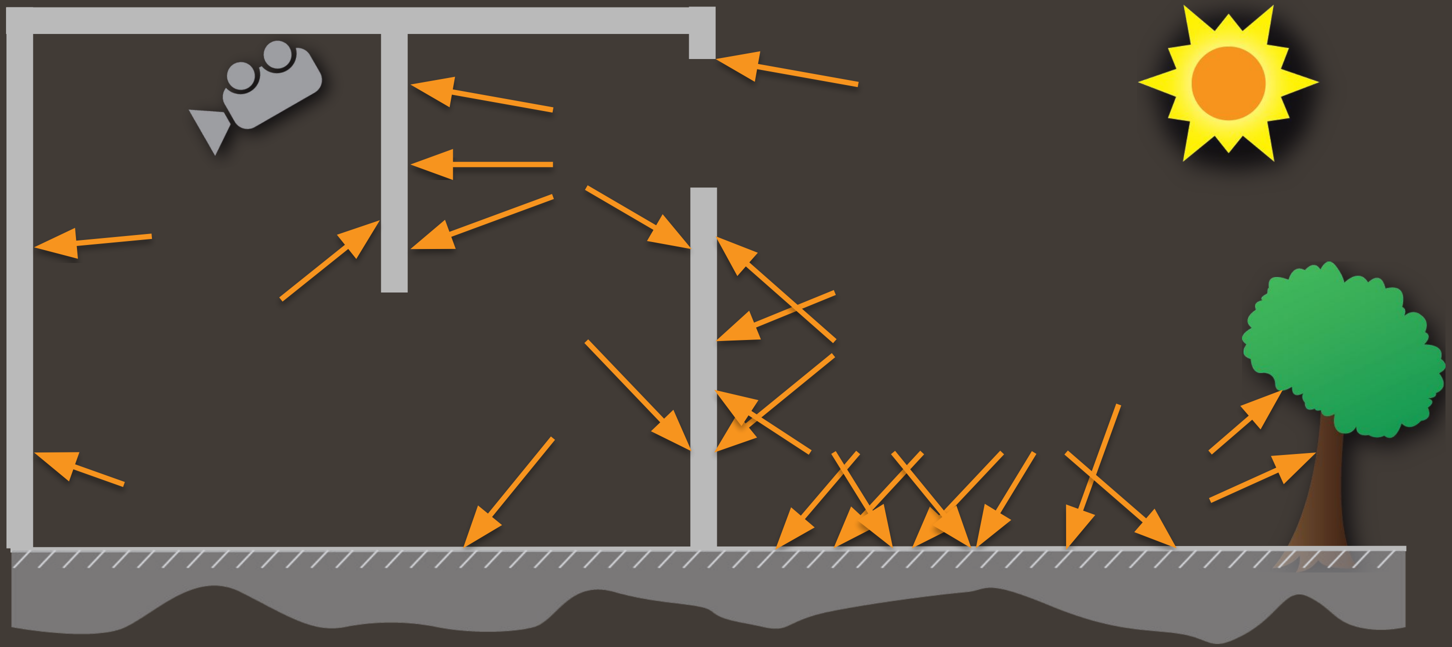
# Limitations of previous work



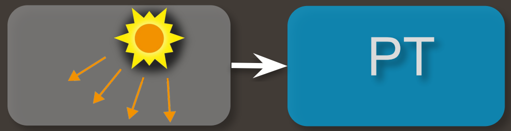


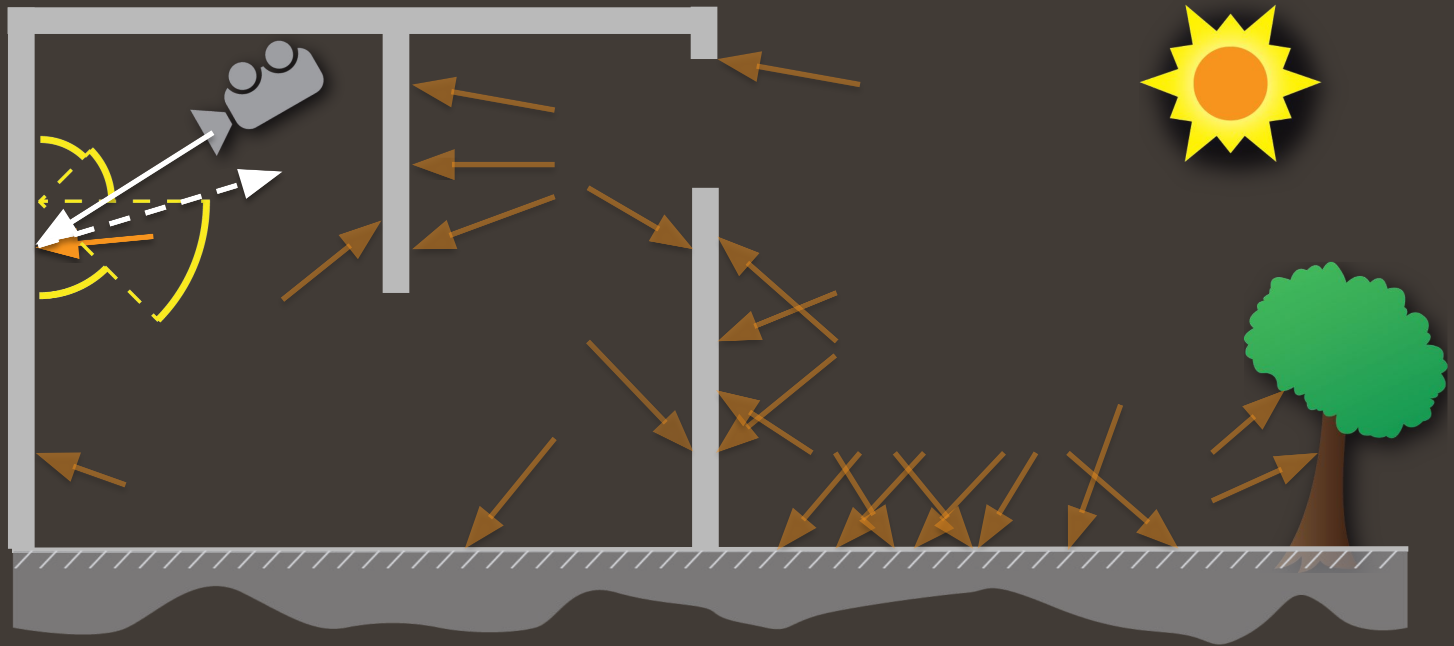
# Limitations of previous work





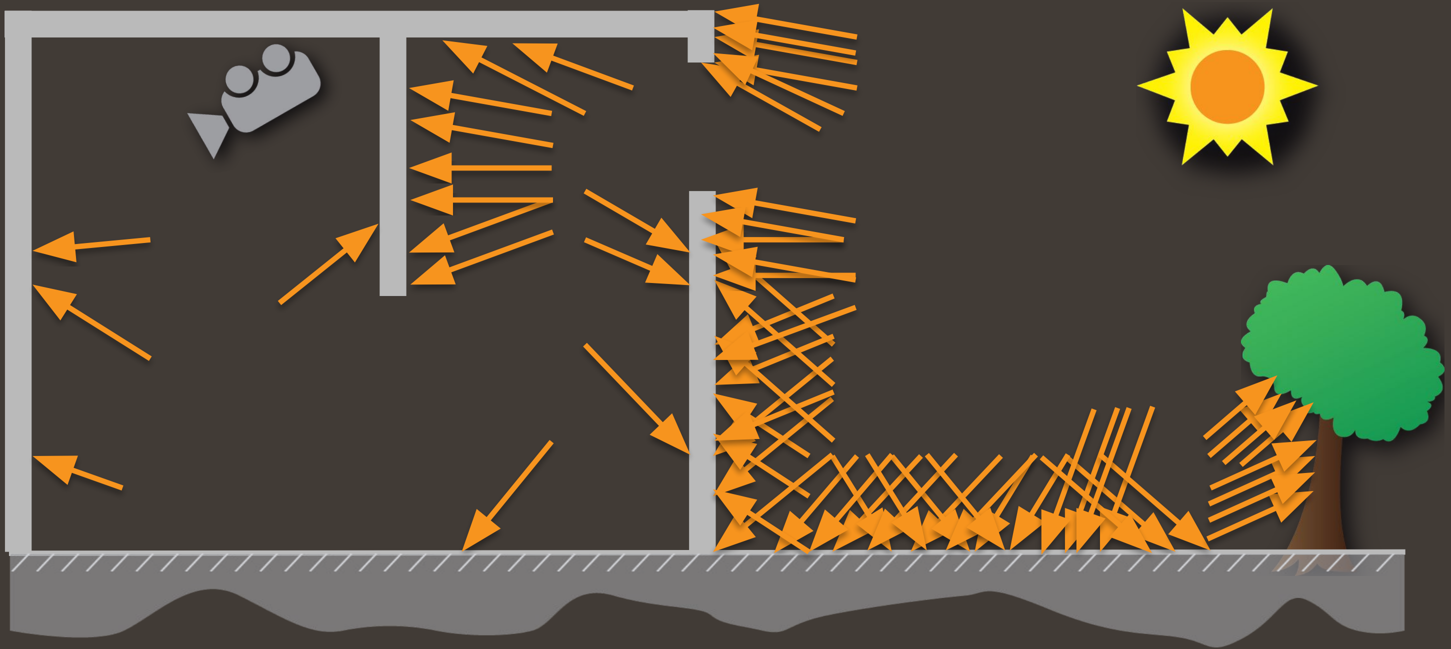
# Limitations of previous work



# Limitations of previous work

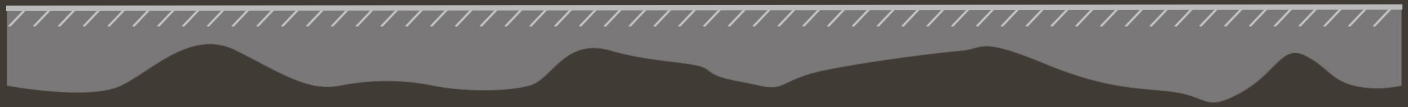
**Not enough memory!**



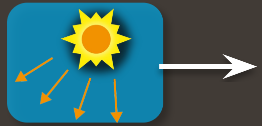
# Solution: On-line Learning of Parametric Model

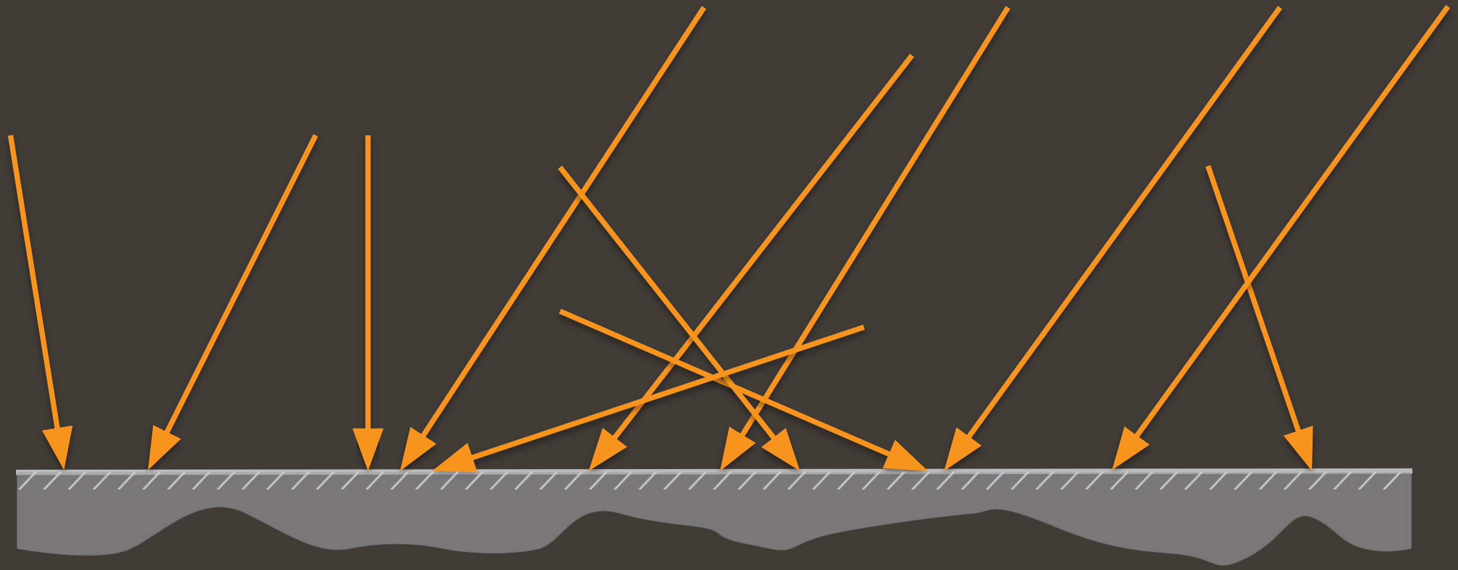
- Shoot a batch of photons, then summarize into a parametric model
  - GMM (Gaussian Mixture Model) is used
  - **Parametric model use less memory**
- Forget previous photon batch and shoot new batch
- Keep updating parameters of the model: **On-line learning**

# Overcoming the memory constraint

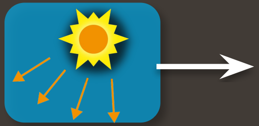


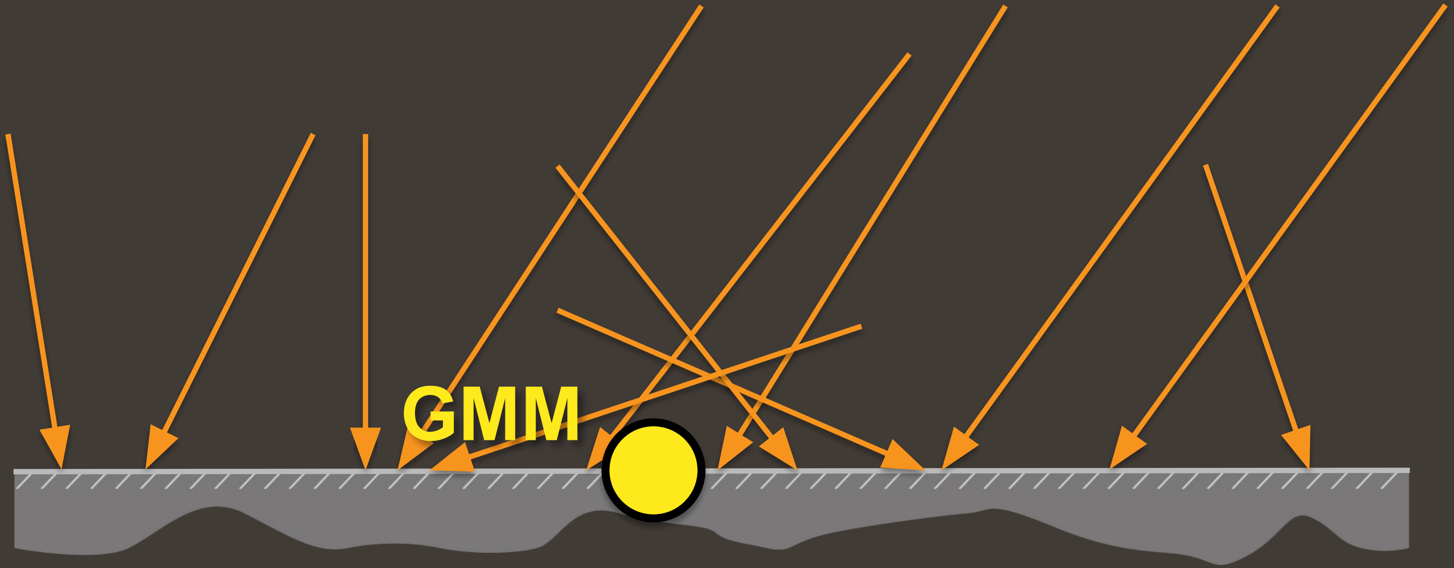
# Overcoming the memory constraint

$1^{st}$ pass 

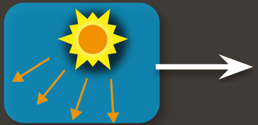


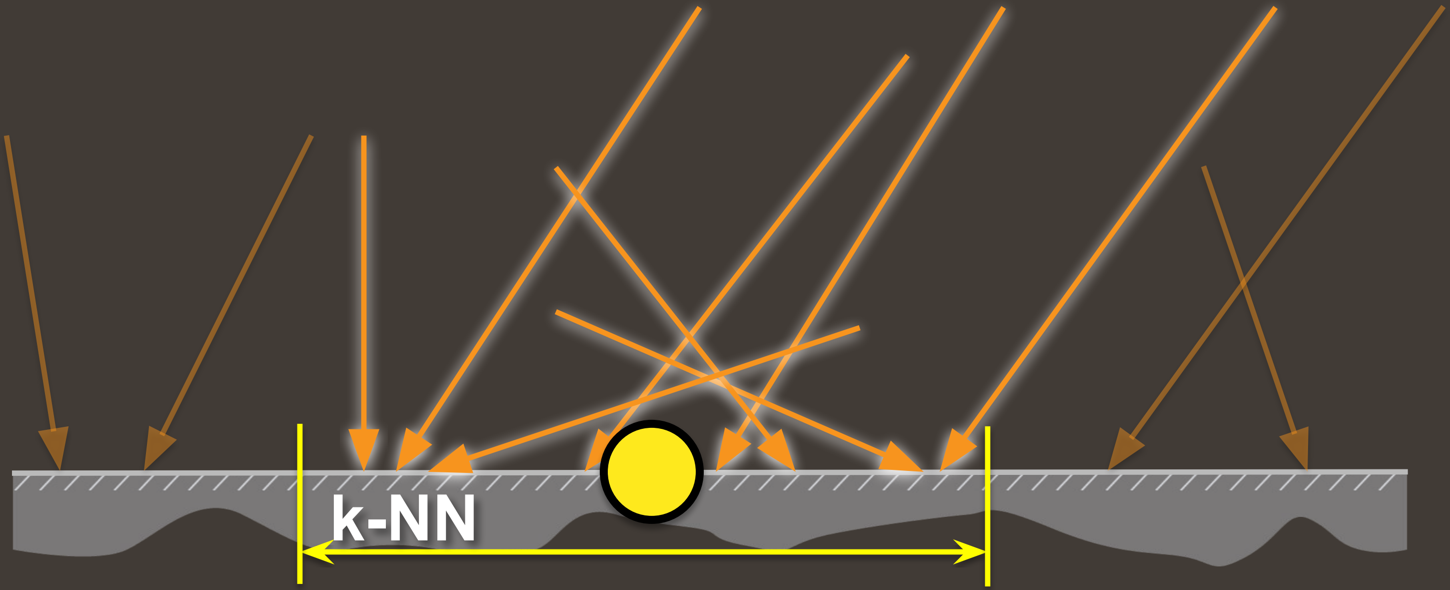
# Overcoming the memory constraint

1st pass 

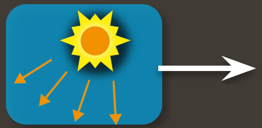


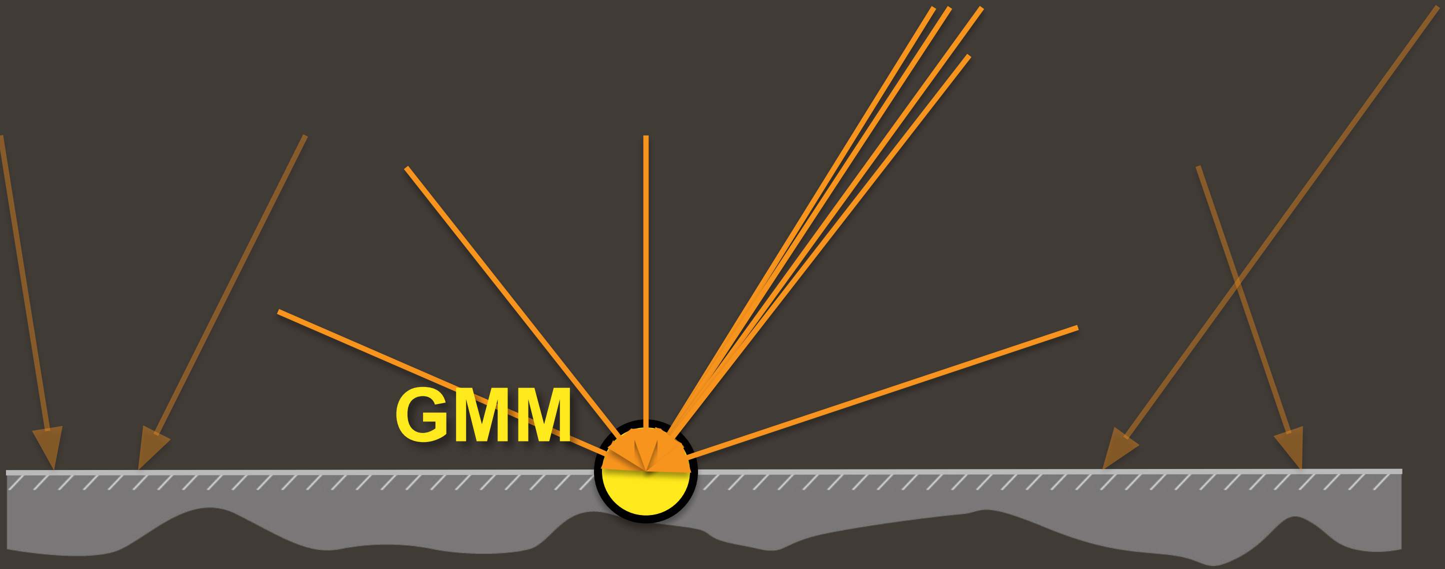
# Overcoming the memory constraint

$1^{st}$ pass 

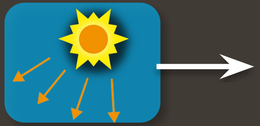


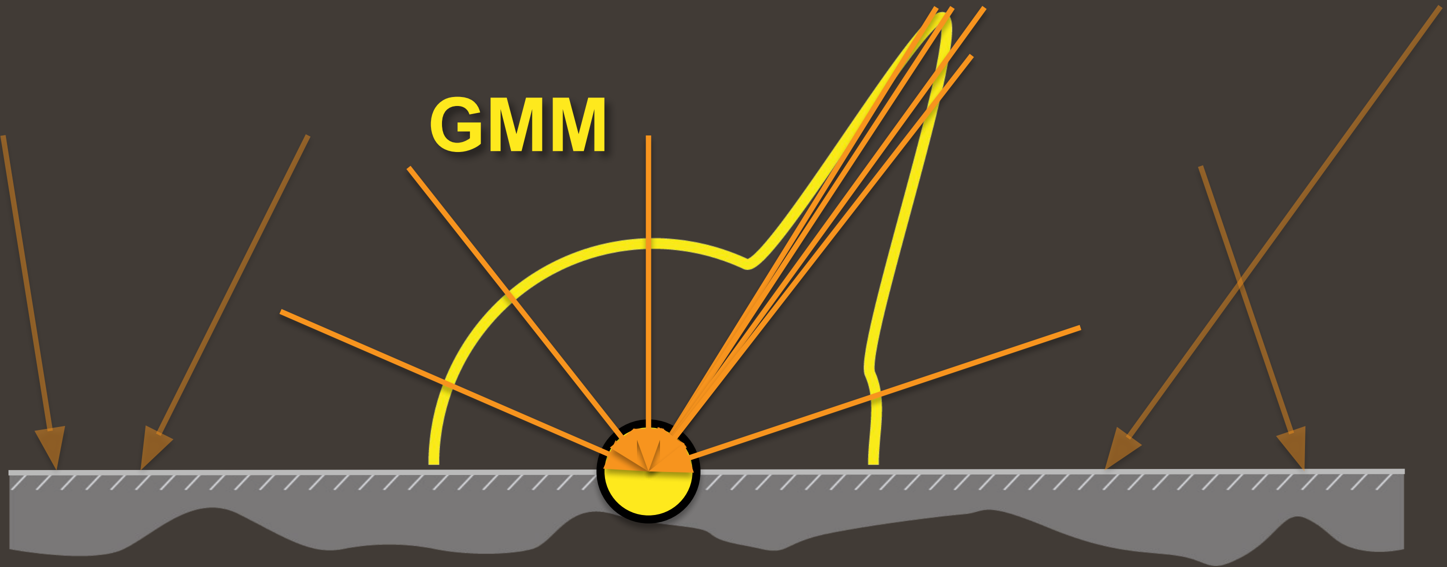
# Overcoming the memory constraint

1$^{st}$ pass 

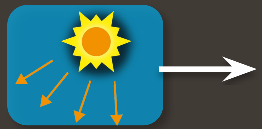


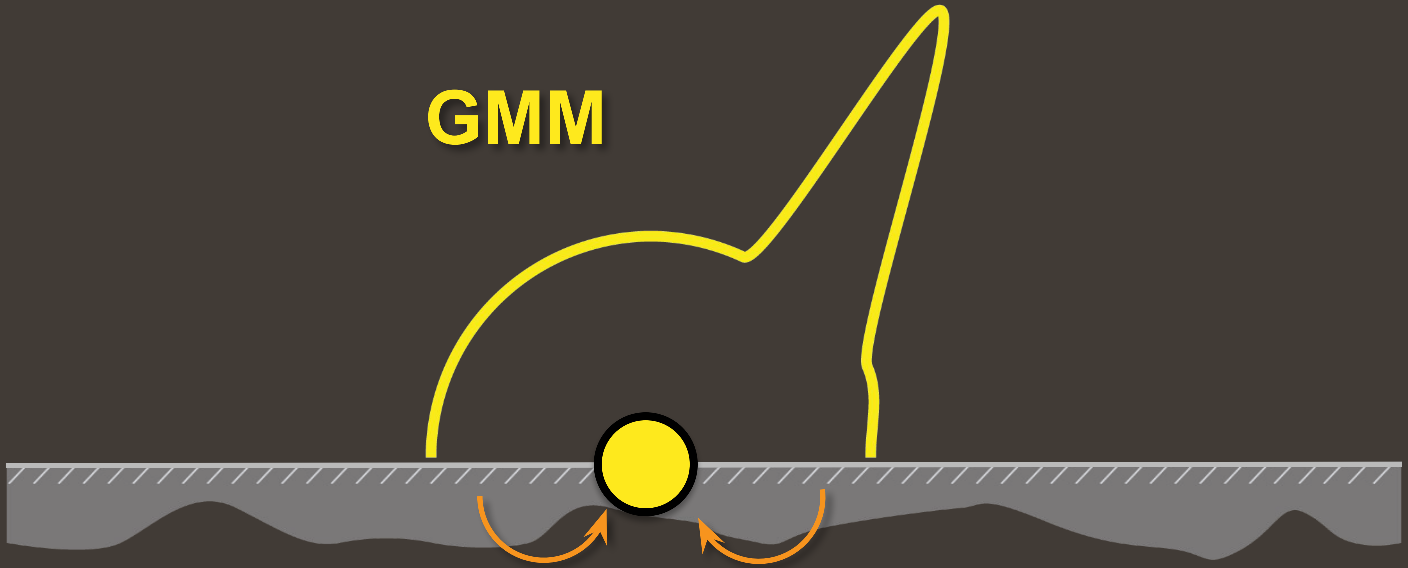
# Overcoming the memory constraint

1st pass 

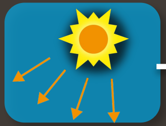
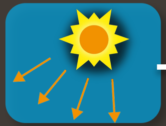


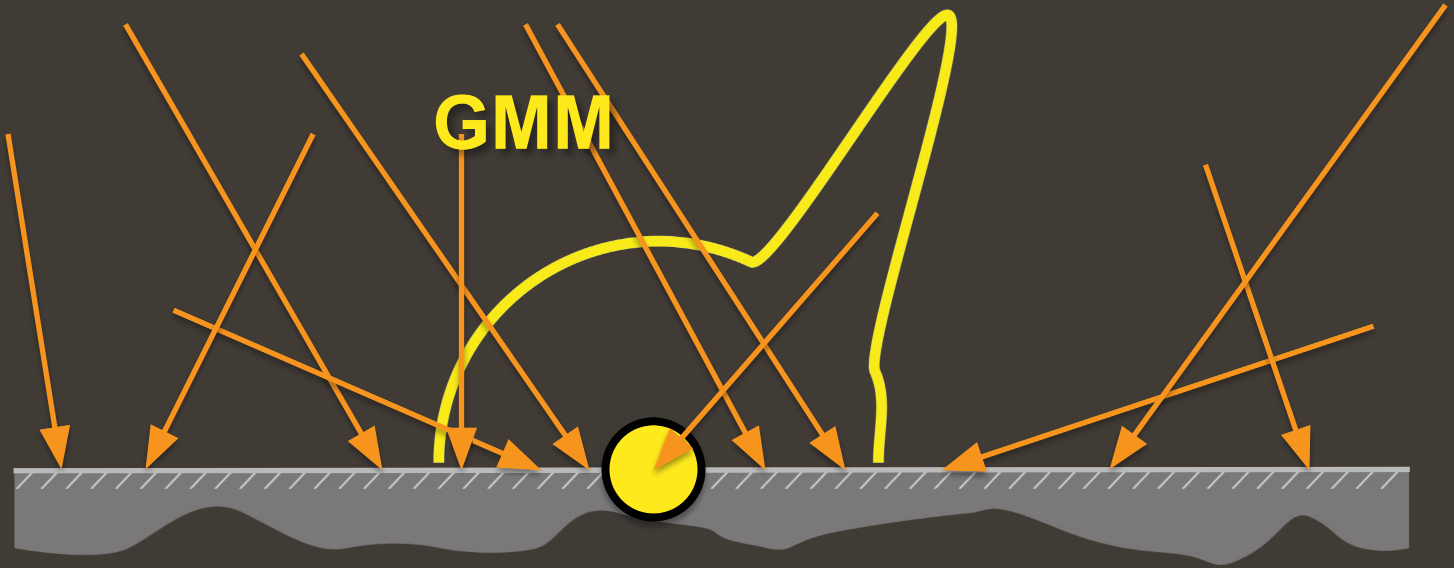
# Overcoming the memory constraint

1st pass 

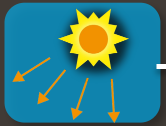
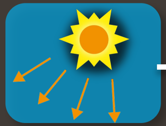


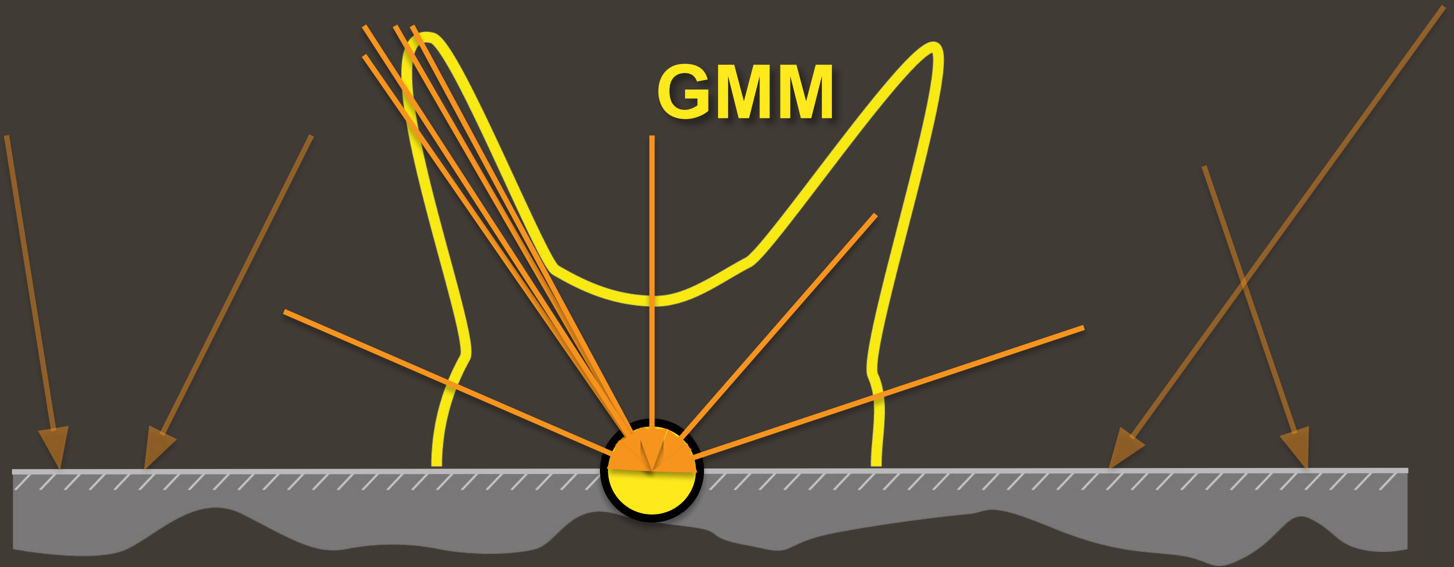
# Overcoming the memory constraint

$1^{st}$ pass  → $2^{nd}$ pass  →

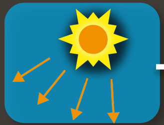
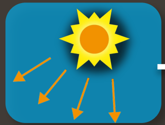
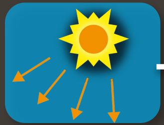


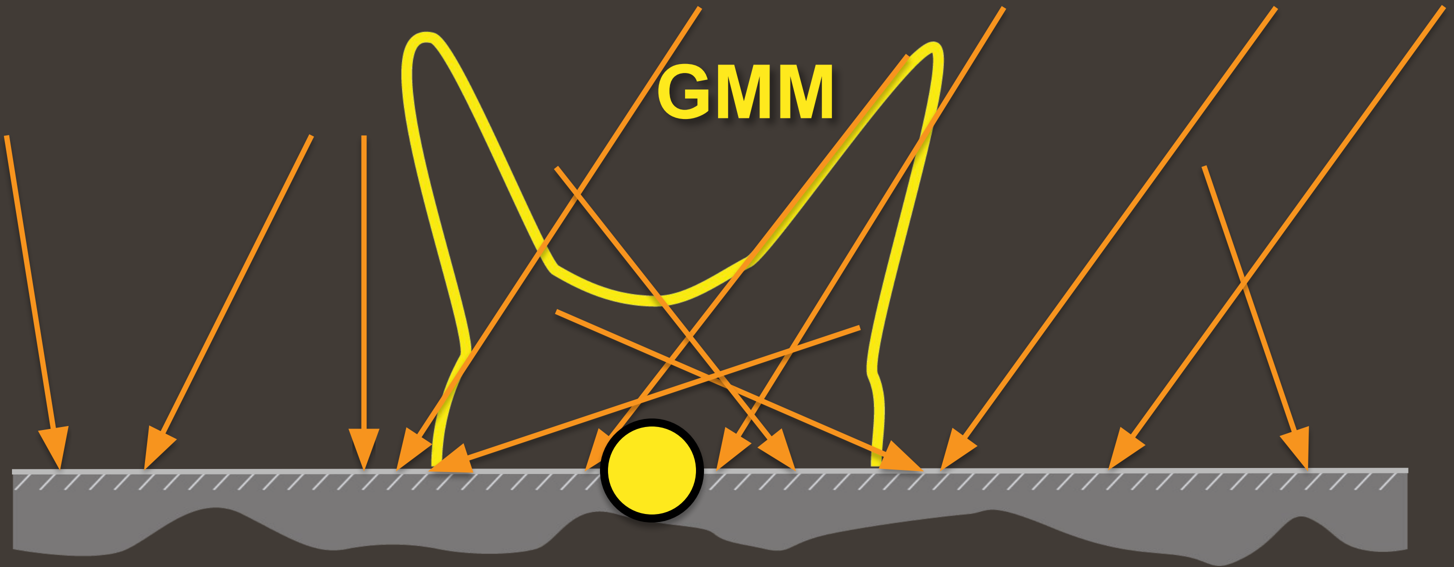
# Overcoming the memory constraint

1$^{st}$ pass  → 2$^{nd}$ pass  →

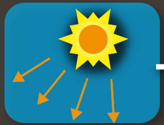
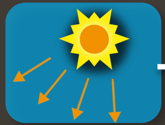
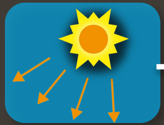


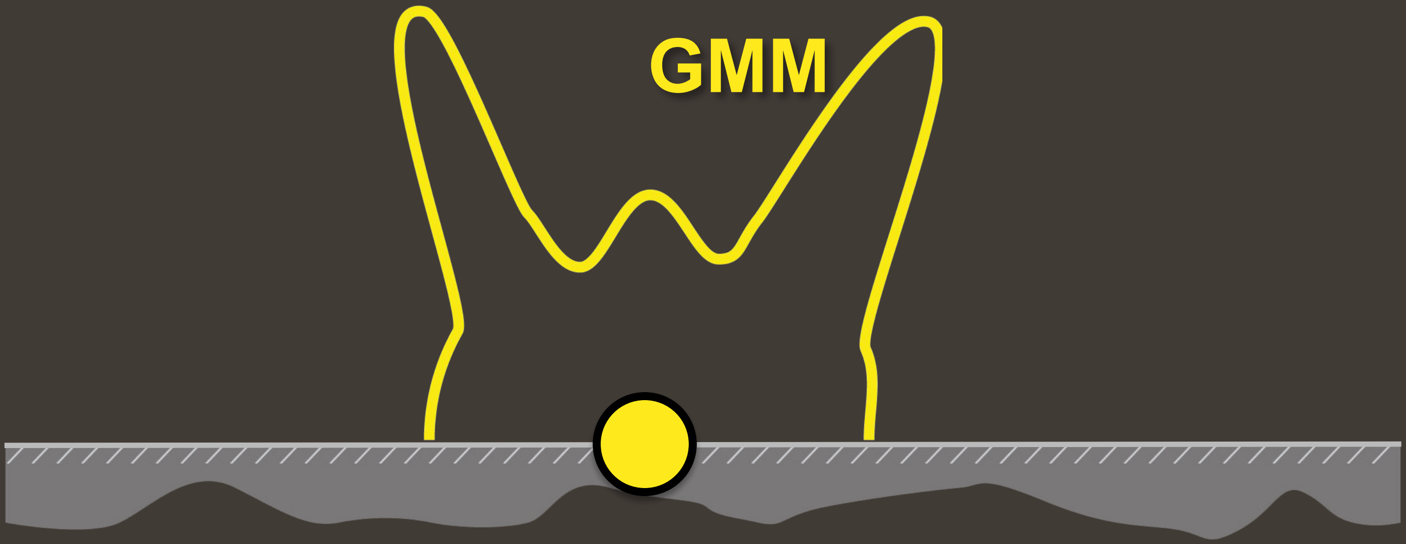
# Overcoming the memory constraint

1st pass  → 2nd pass  → 3rd pass  → ...



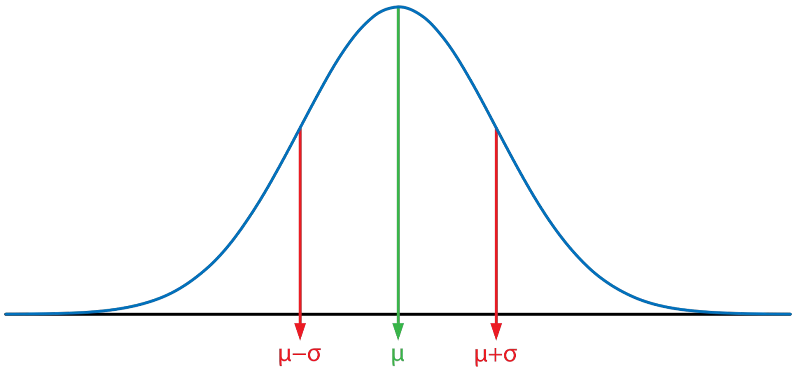
# Overcoming the memory constraint

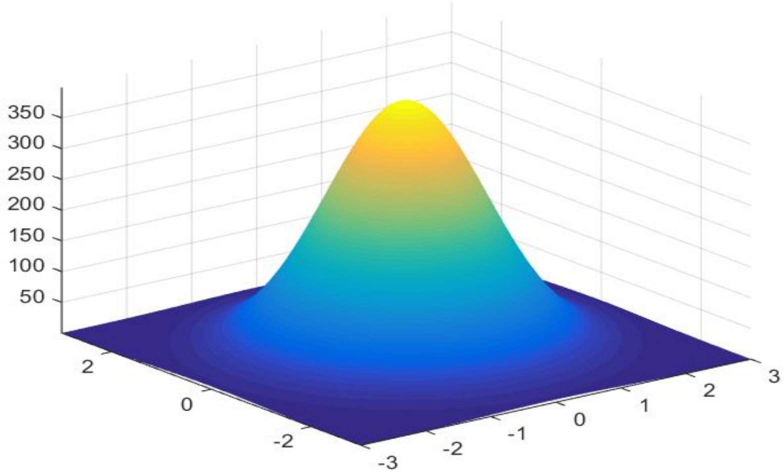
1st pass  → 2nd pass  → 3rd pass  → ...



# Gaussian Mixture Model

# Gaussian Distribution (Normal Distribution)

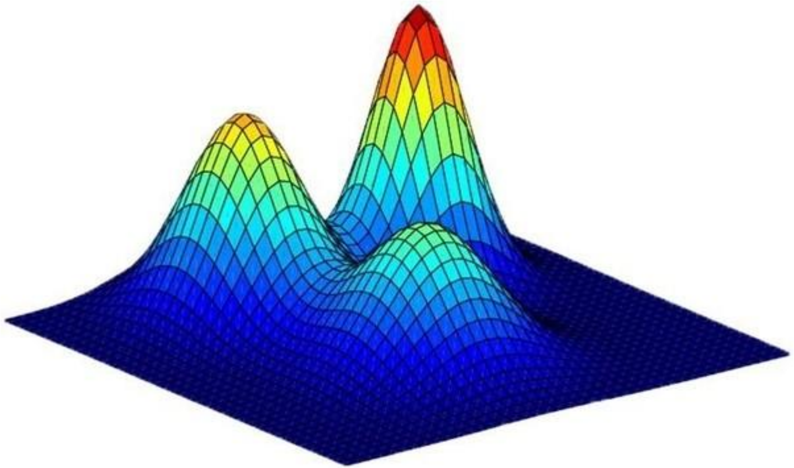



$$f_{\mathbf{X}}(x_1, \ldots, x_k) = \frac{\exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})\right)}{\sqrt{(2\pi)^k |\boldsymbol{\Sigma}|}}$$

Compact:
just 6 float
numbers for 2D

# Gaussian Mixture Model (GMM)



Convex combination of Gaussians:

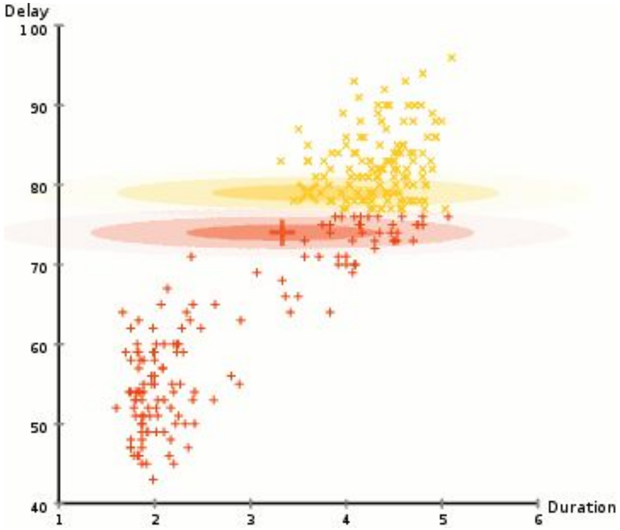
$$\mathrm{GMM}(\mathbf{s}|\theta) = \sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{s}|\mu_j, \mathbf{\Sigma}_j).$$

$\sum_k^K \pi_k = 1$

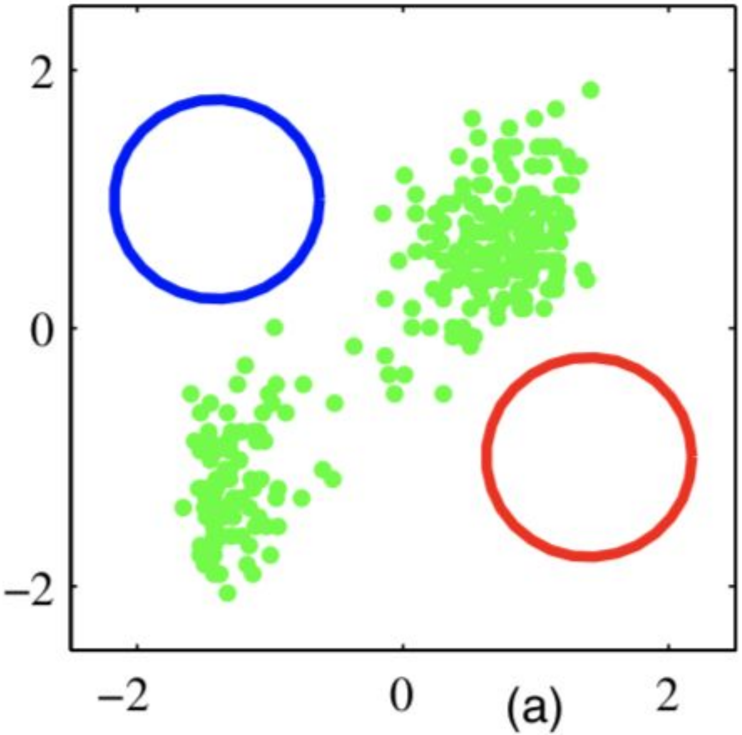
Used to approximate PDF

# Expectation Maximization (EM) Algorithm

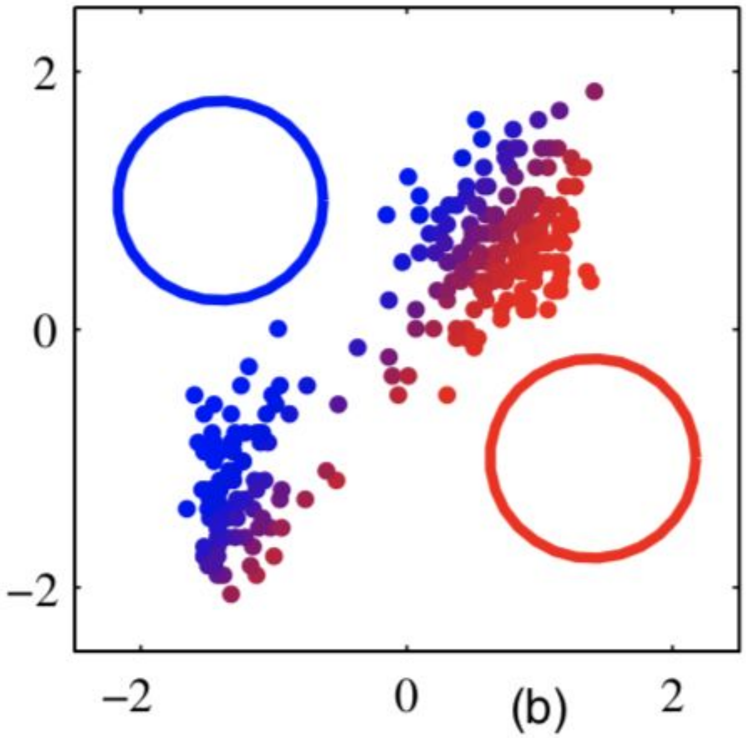
- Popular algorithm that can be used for fitting GMM to scattered data points
- Consists of 2 steps: E-step (expectation) and M-step (maximization)
- Converge to local maximum of likelihood



# EM: How It Works



# EM: Expectation Step

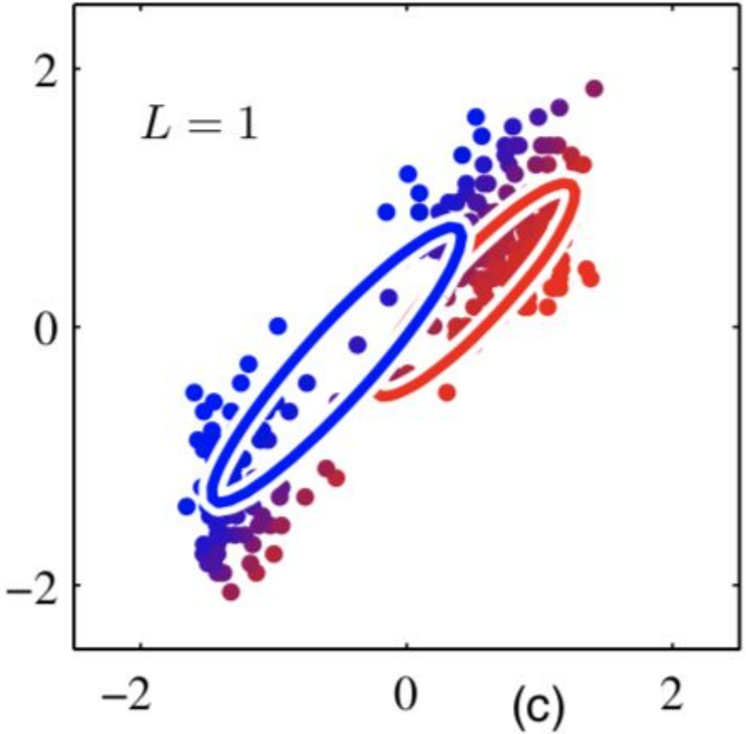



- For each sample, compute soft assignment weight to clusters

$$\gamma_{qj} = \frac{\pi_j \mathcal{N}(\mathbf{s}_q \mid \theta_j^{\text{old}})}{\sum_{h=1}^{K} \pi_h \mathcal{N}(\mathbf{s}_q \mid \theta_h^{\text{old}})}$$

Soft assignment using Bayes' rule

# EM: Maximization Step

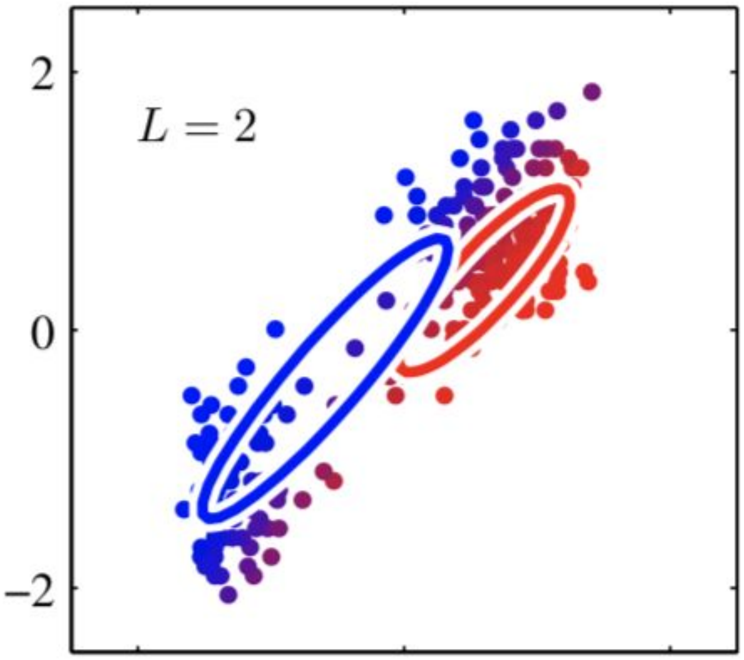



- Update each cluster parameters (mean, variance, weight) to fit the data assigned to it

$$\mathbf{u}^j_{N-1} = \frac{1}{N} \sum_{q=0}^{N-1} \gamma_{qj} \mathbf{u}(\mathbf{s}_q)$$

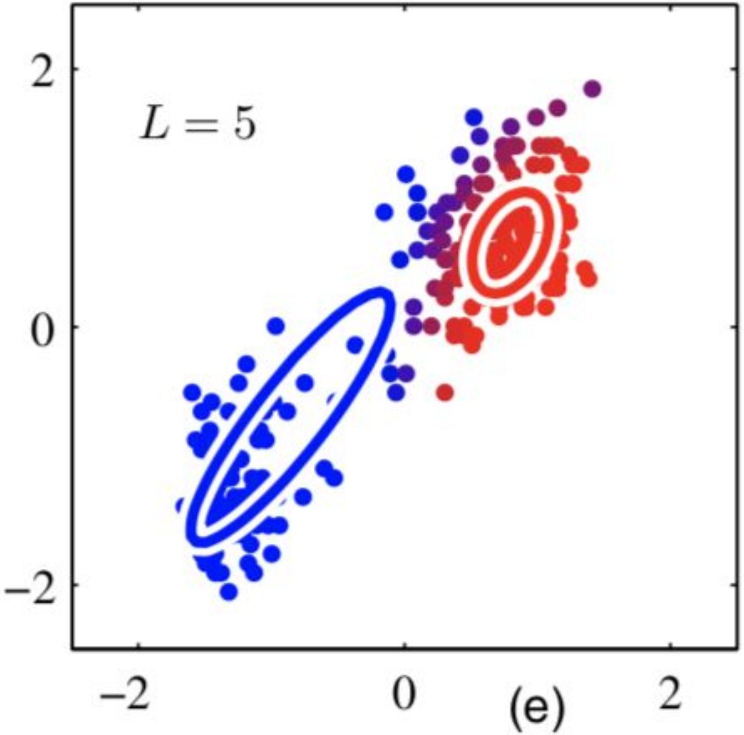
$$\theta^{\text{new}} = \overline{\theta}(\mathbf{u}^1_i, \ldots, \mathbf{u}^K_i)$$

# EM example

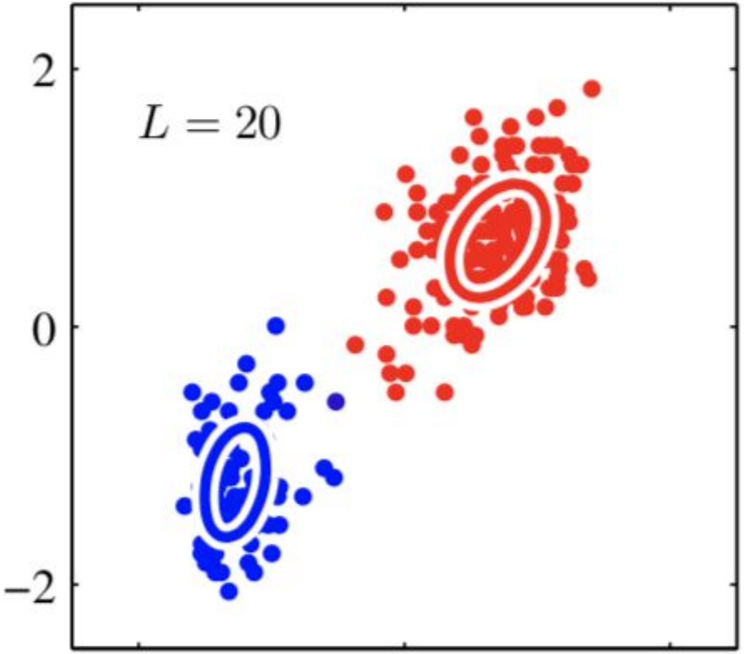


(d)

# EM example



# EM example



(f)

# On-line learning: Weighted Stepwise EM

**Original EM:**

$$\mathbf{u}_{N-1}^{j} = \frac{1}{N} \sum_{q=0}^{N-1} \gamma_{qj} \, \mathbf{u}(\mathbf{s}_q)$$

- Fit to density of finite set of samples, compute sufficient statistics at once

**Weighted stepwise EM: (variant used for this paper)**

$$\mathbf{u}_{i}^{j} = (1 - \eta_i)\mathbf{u}_{i-1}^{j} + \eta_i w_q \gamma_{qj} \, \mathbf{u}(\mathbf{s}_q)$$

- Use one sample for each step and extend to **infinite stream of samples**
- Use **weighted samples** (can be viewed as repeated samples)

# Process Overview

# Process Overview

1. Preprocessing
2. Training
3. Rendering

$$L_R = \int_\Omega \rho(\mathbf{x}, \omega_o, \omega_i) \cdot L(\mathbf{x}, \omega_i) \cdot \cos\theta \, d\omega_i$$

# Process Overview

1. Preprocessing
2. Training
3. Rendering

$$L_R = \int_\Omega \rho(\mathbf{x}, \omega_o, \omega_i) \cdot L(\mathbf{x}, \omega_i) \cdot \cos\theta \, d\omega_i$$

# Process Overview

1. Preprocessing
2. Training
3. Rendering

$$L_{\mathrm{R}} = \int_{\Omega} \rho(\mathbf{x}, \omega_{\mathrm{o}}, \omega_{\mathrm{i}}) \cdot L(\mathbf{x}, \omega_{\mathrm{i}}) \cdot \cos\theta \, d\omega_{\mathrm{i}}$$

# Process Overview

1. Preprocessing
2. Training
3. Rendering

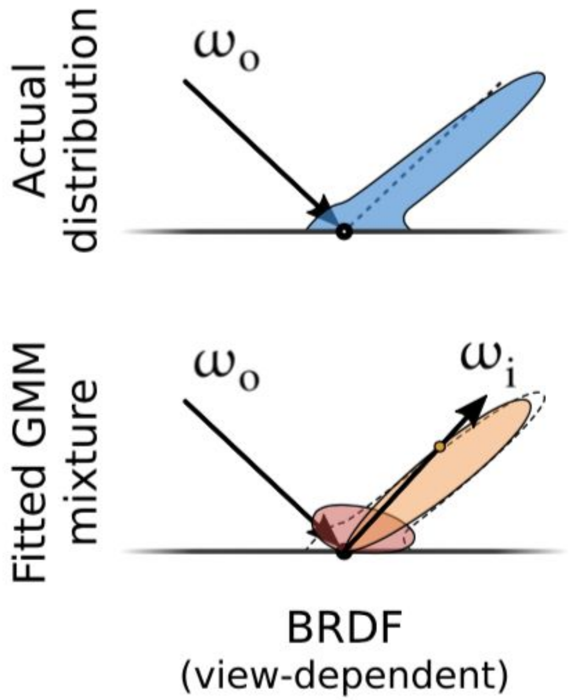
$$L_R = \int_{\Omega} \rho(\mathbf{x}, \omega_o, \omega_i) \cdot L(\mathbf{x}, \omega_i) \cdot \cos\theta \, d\omega_i$$

# 1. Preprocessing

- BRDF is approximated by GMM
- Cache GMM for each material, for each (viewing) direction

$$p_\rho(\omega_o|\omega_i, \mathbf{x}) \propto \rho(\mathbf{x}, \omega_i, \omega_o)$$

**BRDF:Given**

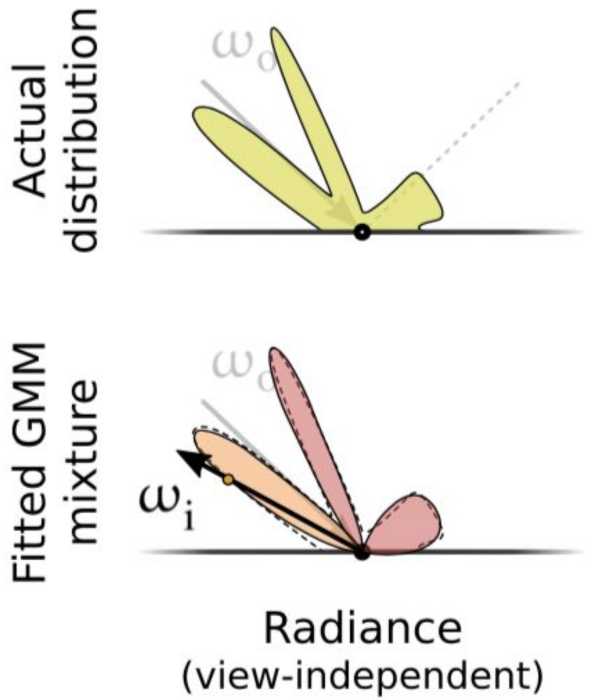


# 2. Training

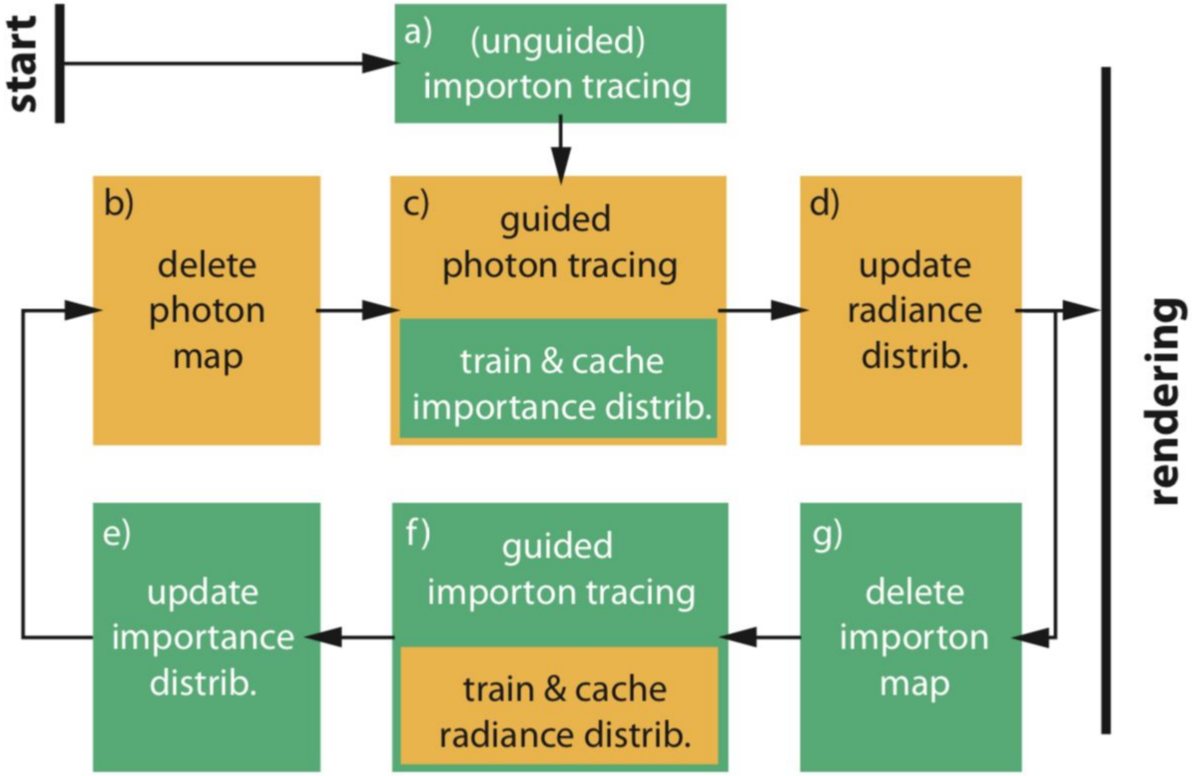
- Photon, importons guide each other in alternating fashion
- On-line learning with weighted step-wise EM
- Cache the learnt illumination GMMs

$$p_L(\omega_o|\mathbf{x}) \propto L(\mathbf{x}, \omega_o)\cos\theta$$

**Illumination: not known in advance**



# 2. Training

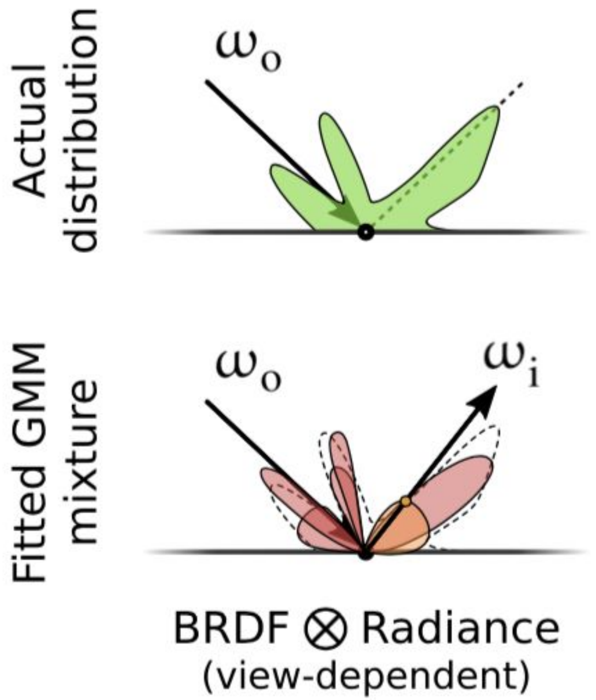

start
a) (unguided) importon tracing
b) delete photon map
c) guided photon tracing
train & cache importance distrib.
d) update radiance distrib.
rendering
e) update importance distrib.
f) guided importon tracing
train & cache radiance distrib.
g) delete importon map

# 3. Rendering

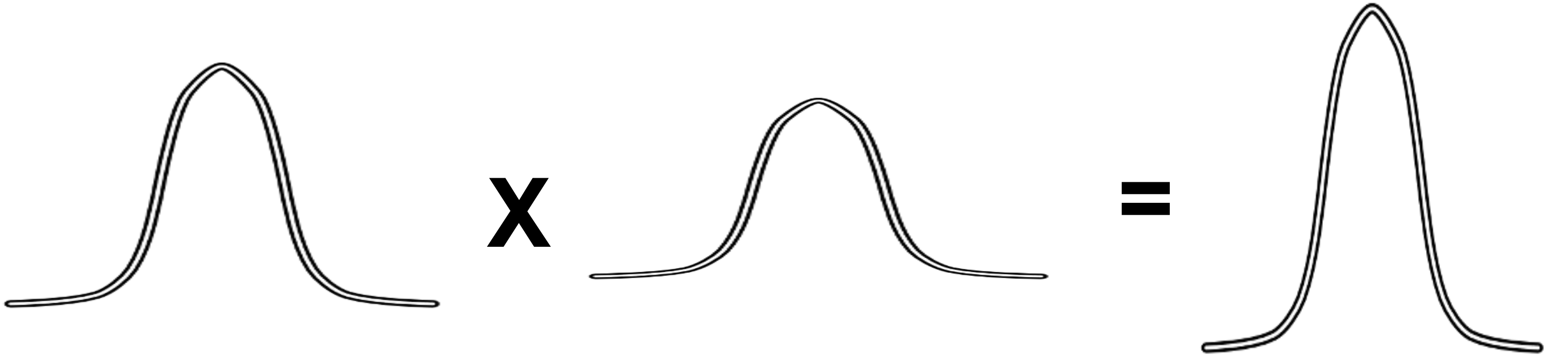
- For intersection point, query the cached BRDF, radiance GMM
- **Product distribution is calculated on-the-fly**
- Sampling based on product distribution

$$p \propto p_{\otimes} = p_{\rho} \otimes p_{L}$$

How can we calculate efficiently?



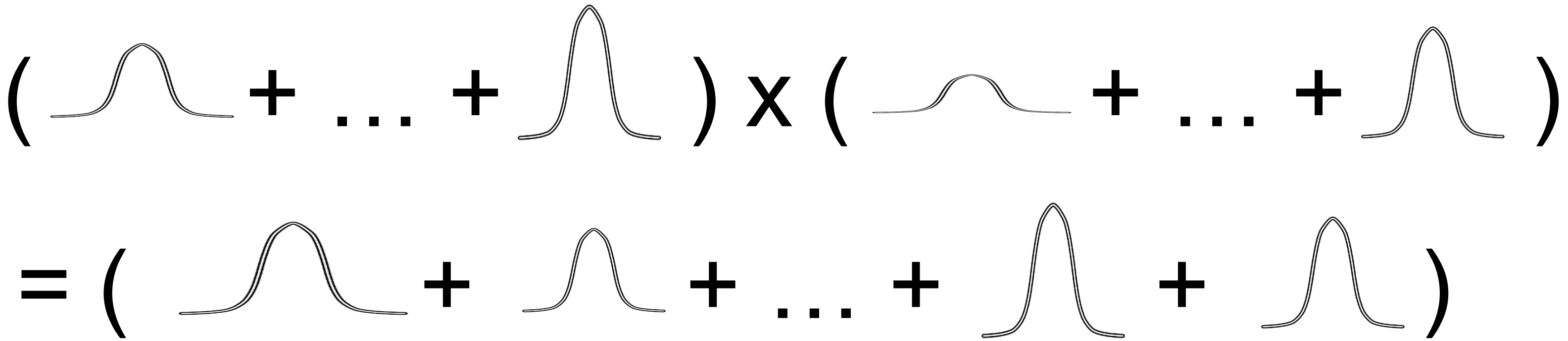
# Gaussian x Gaussian = Gaussian




- Extends to multi-dimensional Gaussian

# GMM x GMM = GMM

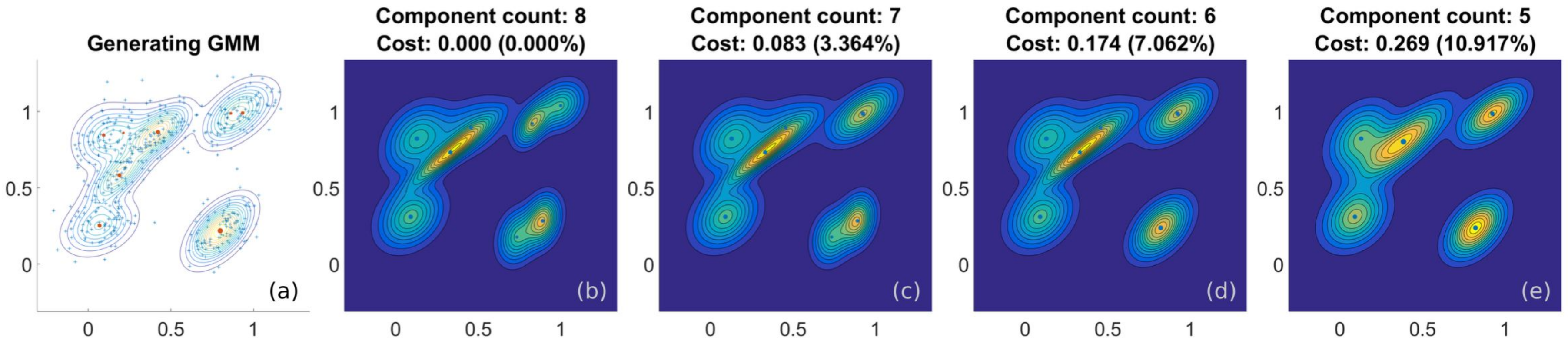
BRDF: GMM of N components Illumination: GMM of M components




Product distribution: GMM of M*N components

- **Parameters for product GMM can be computed directly from original parameters**

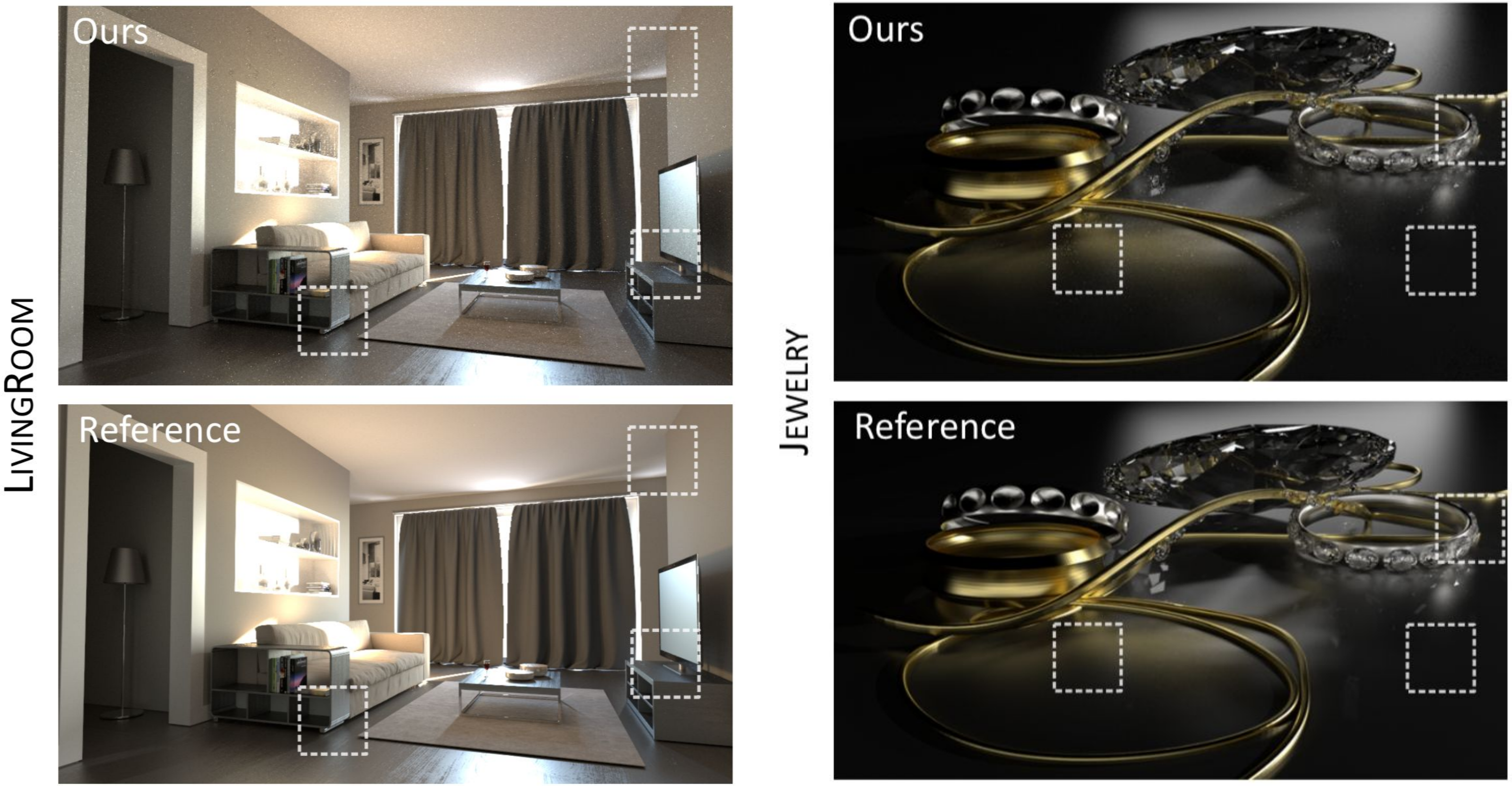
# Reduction of GMM components




- For the sake of efficiency, merge similar components

# Results & Discussion

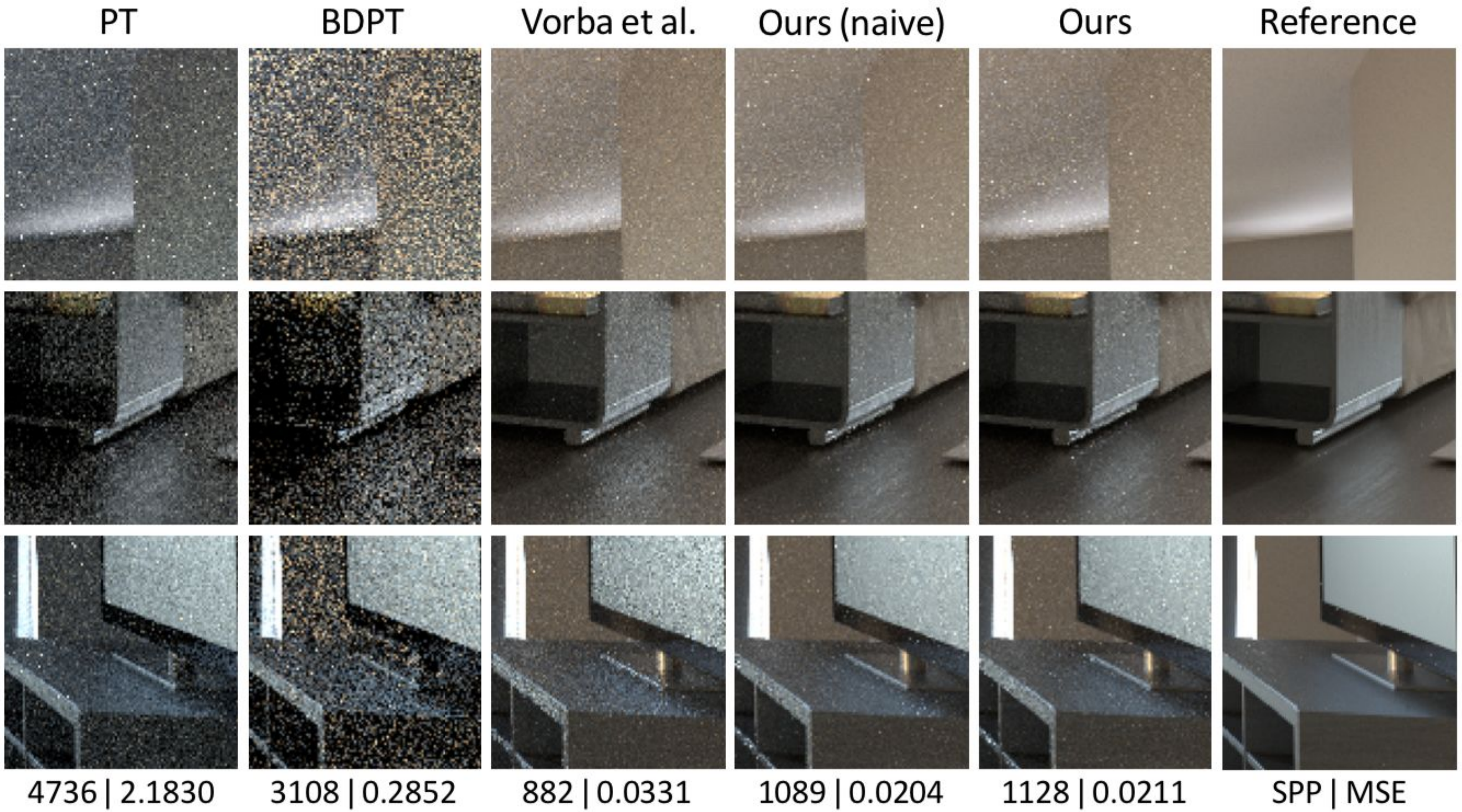
# Evaluation: 1 hour rendering



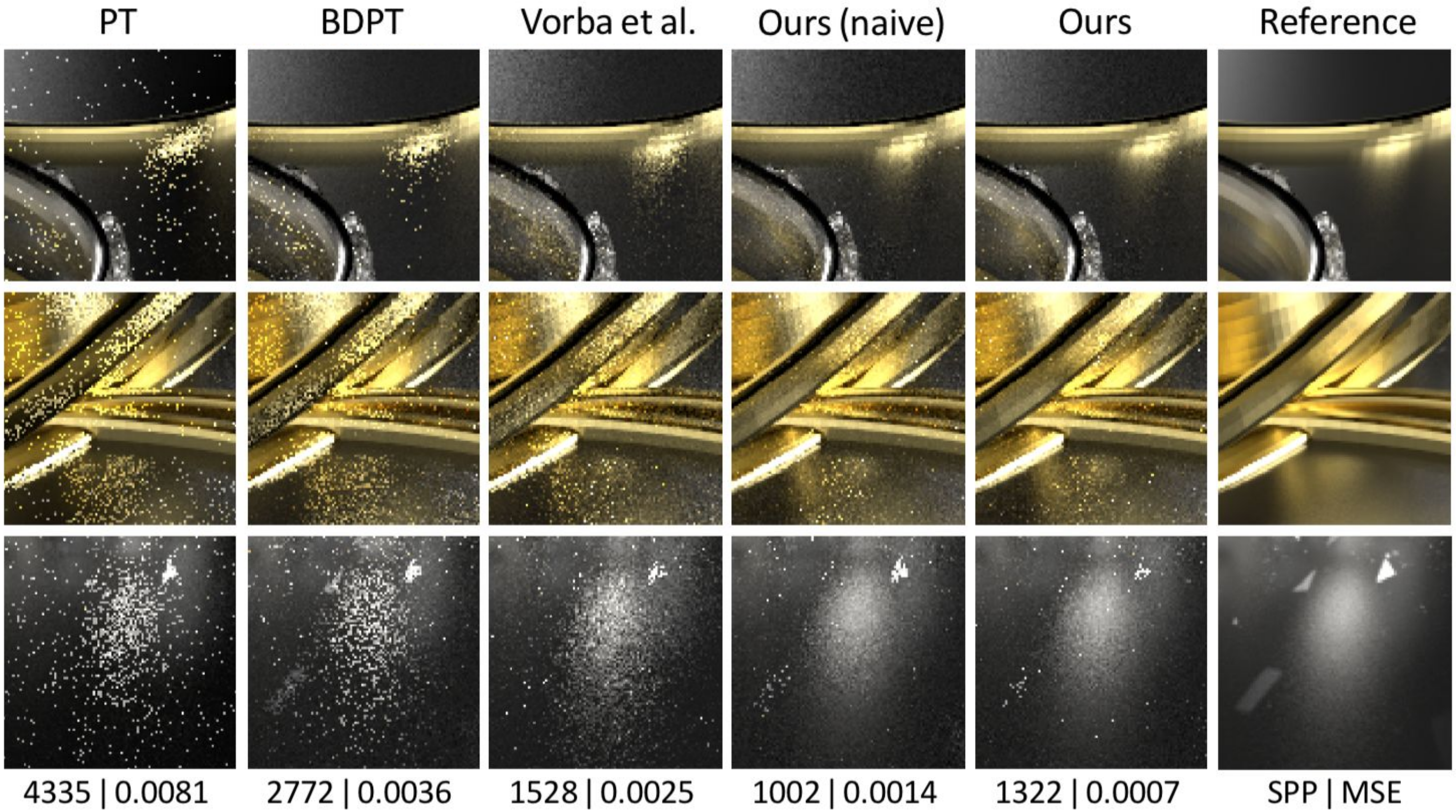
# Result

Multiple importance sampling instead of product dist.

No GMM reduction



# Result


PT
BDPT
Vorba et al.
Ours (naive)
Ours
Reference
4335 | 0.0081
2772 | 0.0036
1528 | 0.0025
1002 | 0.0014
1322 | 0.0007
SPP | MSE

# Discussion

- No memory issue indeed
  - < 10MB for GMM cache in typical scene
- Fast convergence for complex glossy-glossy reflection scene
  - Where product sampling is important
- Not efficient for spatially varying BRDF
  - GMM is cached per material
  - Possible extension using SVBRDF parameters

# Summary

- In order to perform importance sampling, we estimate illumination based on particles
- In complex scenes, we need more particles for better estimation
- **On-line learning of GMM by weighted stepwise EM, enables to generate particles without causing memory issues.**
- BRDF is also approximated as GMM so that we can use the **product GMM as direct approximation for the integrand** of the rendering equation
- Fast convergence for complex, glossy scenes