

Metropolis light transport (a.k.a. Markov chain Monte Carlo rendering)



15-468, 15-668, 15-868
Physics-based Rendering
Spring 2023, Lecture 15

Course announcements

- Take-home quiz 9 posted, due 4/5.
- Programming assignment 4 due tomorrow.

Slide credits

Most of these slides were directly adapted from:

- Shuang Zhao (UC Irvine).
- Toshiya Hachisuka (University of Waterloo)

Today's Lecture

- Metropolis light transport (MLT)
 - A Markov chain Monte Carlo (MCMC) framework implementing the Metropolis-Hastings method first proposed by Veach and Guibas in 1997
 - Capable of efficiently constructing “difficult” transport paths
 - Lots of ongoing research along this direction
- MLT is capable of solving both the rendering equation (RE) and the radiative transfer equation (RTE). We will focus on the former

Metropolis-Hastings Method

- A Markov-Chain Monte Carlo technique
- Given a non-negative function f , generate a chain of **correlated** samples $X_1, X_2, X_3, \dots$ that follow a probability density proportional to f
- **Main advantage:** f does not have to be a PDF (i.e., unnormalized)

Metropolis-Hastings Method

- Input
 - Non-negative function f
 - Probability density $g(y \rightarrow x)$ suggesting a candidate for the next sample value x , given the previous sample value y
- The algorithm: given current sample X_i
 1. Sample X' from $g(X_i \rightarrow X')$
 2. Let $a = \frac{f(X')}{f(X_i)} \frac{g(X_i \rightarrow X')}{g(X' \rightarrow X_i)}$ and draw $\xi \sim U(0, 1]$
 3. If $\xi \leq a$, set X_{i+1} to X' ; otherwise, set X_{i+1} to X_i
- Start with arbitrary initial state X_0
- *Eventually*, samples will be drawn proportionally to f !

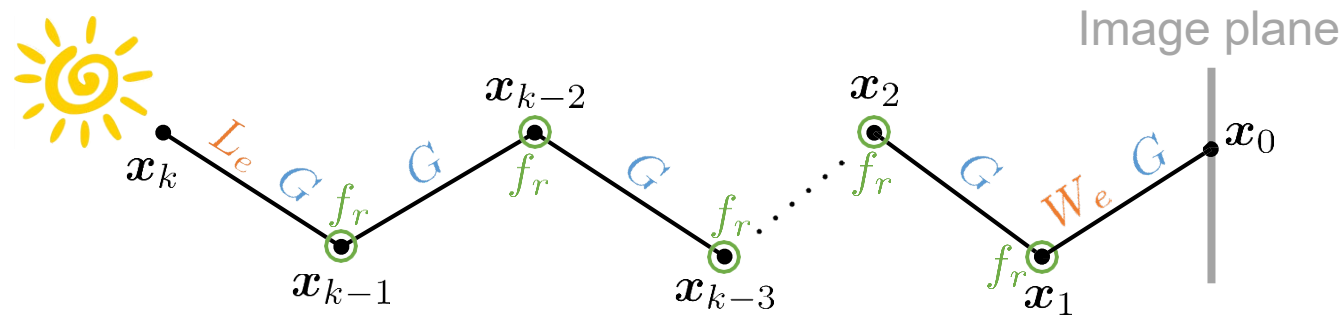
The Problem

- We focus on estimating the pixel values of a virtual image where intensity $I^{(j)}$ of pixel j is

$$I^{(j)} = \int_{\Omega} f^{(j)}(\bar{x}) d\bar{x} = \int_{\Omega} h^{(j)}(\bar{x}) f(\bar{x}) d\bar{x}, \text{ where}$$

$$h^{(j)}(\bar{x}) = W_e^{(j)}(\mathbf{x}_1 \rightarrow \mathbf{x}_0)$$

$$f(\bar{x}) = L_e(\mathbf{x}_k \rightarrow \mathbf{x}_{k-1}) \left[\prod_{j=0}^{k-1} G(\mathbf{x}_{j+1} \leftrightarrow \mathbf{x}_j) \right] \left[\prod_{j=1}^{k-1} f_r(\mathbf{x}_{j+1} \rightarrow \mathbf{x}_j \rightarrow \mathbf{x}_{j-1}) \right]$$



The Problem

- We focus on estimating the pixel values of a virtual image where intensity $I^{(j)}$ of pixel j is

$$I^{(j)} = \int_{\Omega} f^{(j)}(\bar{x}) \, d\bar{x} = \int_{\Omega} h^{(j)}(\bar{x}) f(\bar{x}) \, d\bar{x}, \text{ where}$$

$$h^{(j)}(\bar{x}) = W_e^{(j)}(\mathbf{x}_1 \rightarrow \mathbf{x}_0)$$

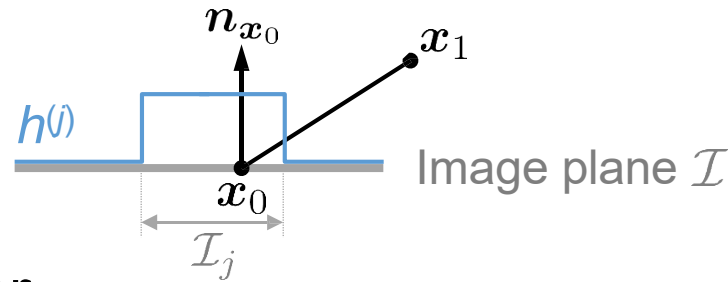
$$f(\bar{x}) = L_e(\mathbf{x}_k \rightarrow \mathbf{x}_{k-1}) \left[\prod_{j=0}^{k-1} G(\mathbf{x}_{j+1} \leftrightarrow \mathbf{x}_j) \right] \left[\prod_{j=1}^{k-1} f_r(\mathbf{x}_{j+1} \rightarrow \mathbf{x}_j \rightarrow \mathbf{x}_{j-1}) \right]$$

- $h^{(j)}$ varies per pixel and is called the *filter function*
- f stays identical for all pixels

Example Filter Functions

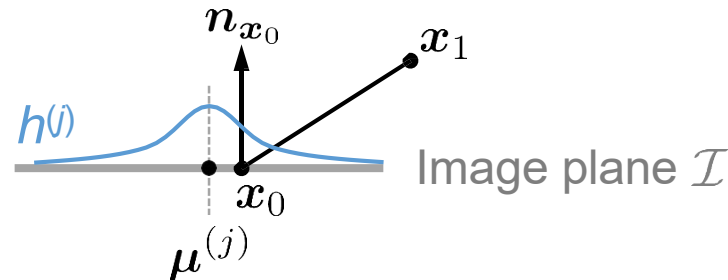
- Box Filter

$$h^{(j)}(\bar{x}) = \frac{\mathbb{1}[\mathbf{x}_0 \in \mathcal{I}_j]}{|\mathcal{I}_j|} \mathbb{1}[\langle \mathbf{x}_0 \rightarrow \mathbf{x}_1, \mathbf{n}_{\mathbf{x}_0} \rangle > 0]$$



- Gaussian Filter

$$h^{(j)}(\bar{x}) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{\|\mathbf{x}_0 - \boldsymbol{\mu}^{(j)}\|^2}{2\sigma^2}\right) \mathbb{1}[\langle \mathbf{x}_0 \rightarrow \mathbf{x}_1, \mathbf{n}_{\mathbf{x}_0} \rangle > 0]$$



Estimating Pixel Values

$$I^{(j)} = \int_{\Omega} h^{(j)}(\bar{x}) f(\bar{x}) \mathrm{d}\bar{x}$$

- We have seen that if we can draw N path samples $\bar{x}_1, \dots, \bar{x}_N$ according to some density function p , then

$$I^{(j)} = \mathbb{E} \left[\frac{1}{N} \sum_{i=1}^N \frac{h^{(j)}(\bar{x}_i) f(\bar{x}_i)}{p(\bar{x}_i)} \right]$$

- Particularly, if we take $p \propto f$, namely $p(\bar{x}) = f(\bar{x})/b$ with b being the normalization factor, then

$$I^{(j)} = \mathbb{E} \left[\frac{1}{N} \sum_{i=1}^N b h^{(j)}(\bar{x}_i) \right]$$

Estimating Pixel Values

$$I^{(j)} = \mathbb{E} \left[\frac{1}{N} \sum_{i=1}^N b h^{(j)}(\bar{x}_i) \right]$$

- Challenges

- How to obtain $b = \int_{\Omega} f(\bar{x}) d\bar{x}$?

Monte Carlo integration

- How to draw samples from $p(\bar{x}) = f(\bar{x})/b$?

Metropolis-Hastings method

$$I^{(j)} = \mathbb{E} \left[\frac{1}{N} \sum_{i=1}^N b h^{(j)}(\bar{x}_i) \right]$$

Metropolis Light Transport (MLT)

- Overview

- **Phase 1:** initialization (estimating b)

- Draw N' “seed” paths $\bar{x}_1^{\text{seed}}, \dots, \bar{x}_{N'}^{\text{seed}}$ from some known density p_0 (e.g., using bidirectional path tracing)

- Set $\langle b \rangle = \frac{1}{N'} \sum_{i=1}^{N'} \frac{f(\bar{x}_i^{\text{seed}})}{p_0(\bar{x}_i^{\text{seed}})}$

- Pick a small number (e.g., one) of representatives from $\bar{x}_1^{\text{seed}}, \dots, \bar{x}_{N'}^{\text{seed}}$ and apply Phase 2 to each of them

- **Phase 2:** Metropolis

- Starting with a seed path, apply the Metropolis-Hastings method to generate samples according to f

Metropolis Phase

- Overview (pseudocode)

Metropolis_Phase(image, $\mathbf{x}^{\text{seed}}$):

$\mathbf{x} = \mathbf{x}^{\text{seed}}$

for $i = 1$ to N :

$\mathbf{y} = \text{mutate}(\mathbf{x})$

$a = \text{acceptanceProbability}(\mathbf{x} \rightarrow \mathbf{y})$

if $\text{rand}() < a$:

$\mathbf{x} = \mathbf{y}$

recordSample(image, $\mathbf{x}$)

Path Mutations

- The key step of the Metropolis phase
- Given a transport path $\bar{x}$, we need to define a **transition probability** $g(\bar{x} \rightarrow \bar{y})$ to allow sampling mutated paths $\bar{y}$ based on $\bar{x}$
 - Given this transition density, the acceptance probability is then given by

$$a(\bar{x} \rightarrow \bar{y}) = \min \left\{ 1, \frac{f(\bar{y})}{f(\bar{x})} \frac{g(\bar{y} \rightarrow \bar{x})}{g(\bar{x} \rightarrow \bar{y})} \right\}$$

Desirable Mutation Properties

- High acceptance probability
 - $a(\bar{x} \rightarrow \bar{y})$ should be large with high probability
- Both small and large changes to the path
- **Ergodicity** (never stuck in some-region of the path space)
 - $g(\bar{x} \rightarrow \bar{y})$ should be non-zero for all $\bar{x}, \bar{y}$ with $f(\bar{x}) > 0, f(\bar{y}) > 0$
- Low cost

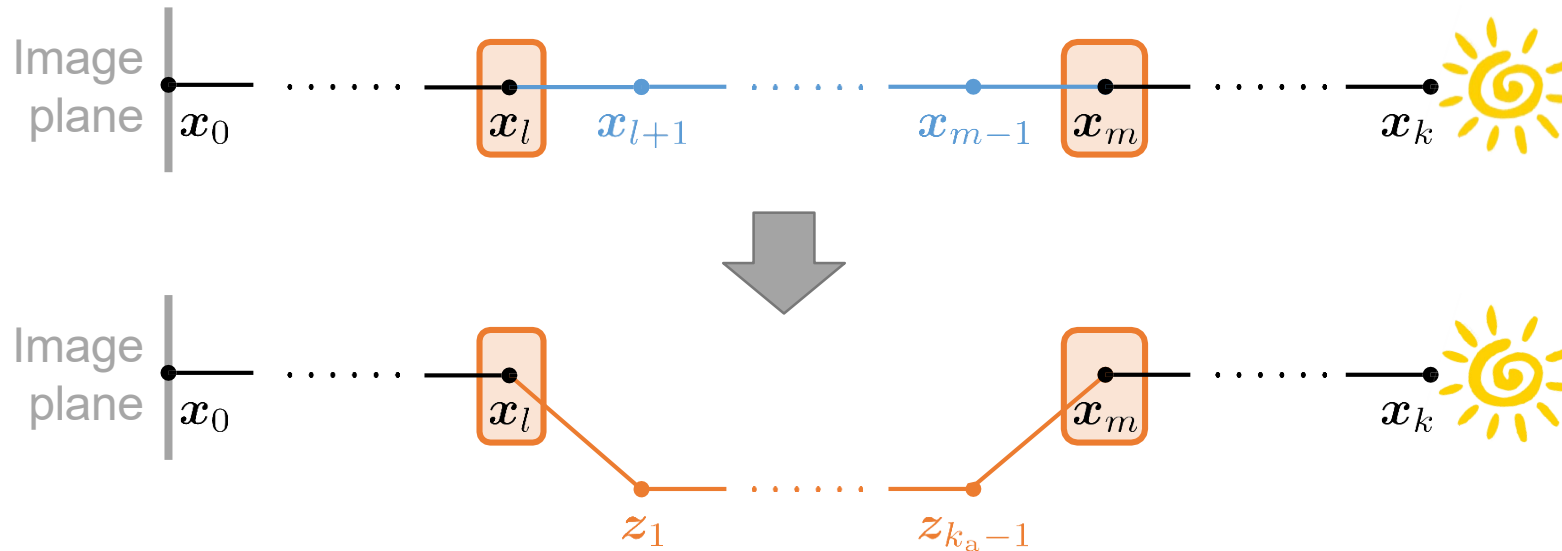
Path Mutation Strategies

- [Veach & Guibas 1997]
 - Bidirectional mutation
 - Path perturbations
 - Lens sub-path mutation
- [Jakob & Marschner 2012]
 - Manifold exploration
- [Li et al. 2015]
 - Hamiltonian Monte Carlo

...

Bidirectional Path Mutations

- Basic idea
 - Given a path $\bar{x} = (x_0, \dots, x_k)$, pick l, m and replace the vertices $x_{l+1}, \dots, x_{m-1}$ with $z_1, \dots, z_{k_a-1}$
 - l and m satisfies $-1 \leq l < m \leq k + 1$

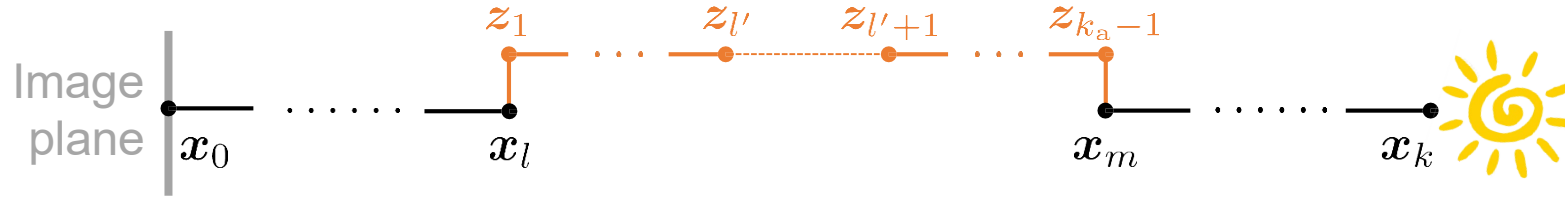


Deletion Probability



- l and m are sampled as follows:
 - Draw integer k_d from some probability mass function $p_{d,1}[k_d]$. This number captures the length of deleted sub-path (i.e., $m - l$)
 - Draw l from another probability mass function $p_{d,2}[l \mid k_d]$ to avoid low acceptance probability and set m to $l + k_d$ (more on this at the end of today's lecture)
 - The joint probability p_d for drawing (l, m) is
$$p_d[l, m] = p_{d,1}[m - l] p_{d,2}[l \mid m - l]$$

Addition Probability

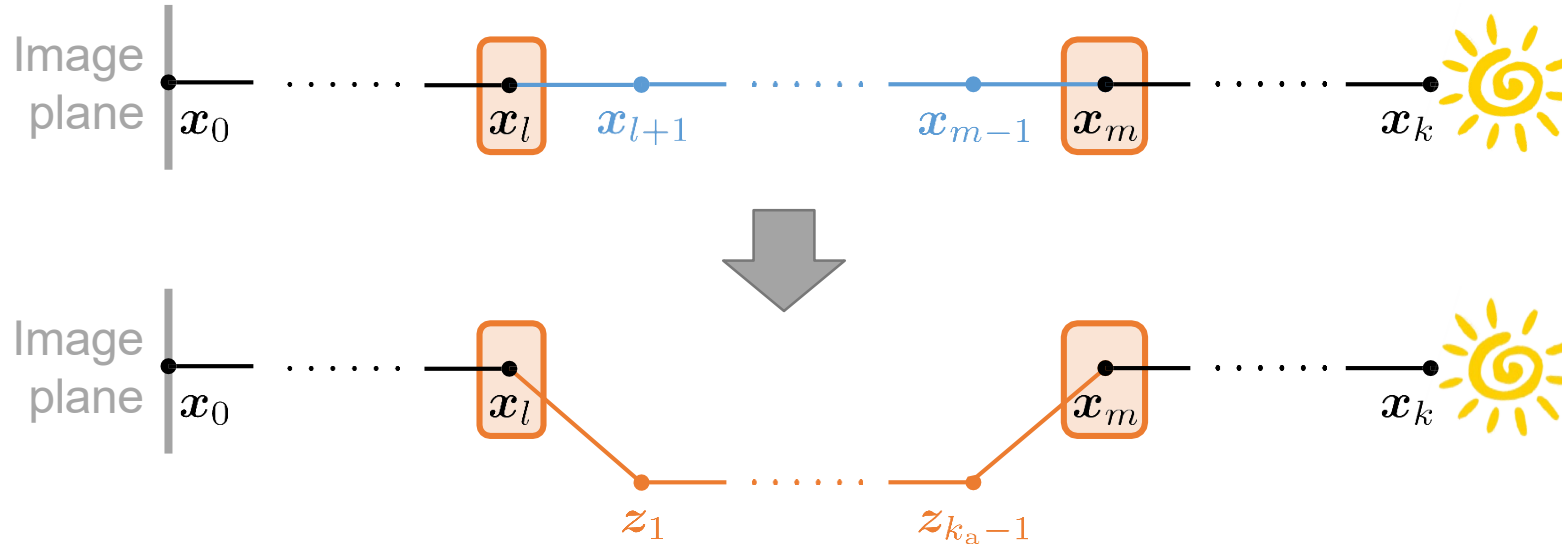


- The deleted sub-path is then replaced by adding l' and m' vertices on each side. To determine l' and m' :
 - Draw integer k_a from $p_a[k_a]$. This integer determines the new sub-path length (i.e., $k_a = l' + m' + 1$)
 - Draw l' uniformly from $\{0, 1, \dots, k_a - 1\}$ and set m' to $k_a - 1 - l'$
 - Let $p_a[l', m']$ denote the joint probability for drawing (l', m')
- After obtaining l' and m' , the two corresponding sub-paths are generated via local path sampling, yielding the new path $\bar{y} = (x_0, \dots, x_l, z_1, \dots, z_{k_a-1}, x_m, \dots, x_k)$

Parameter Values

- Veach [1997] proposed the following parameters:
- Deletion parameters
 - $p_{d,1}[1] = 0.25$, $p_{d,1}[2] = 0.5$, $p_{d,1}[k] = 2^{-k}$ for $k > 2$
(before normalization)
 - $p_{d,2}[l | k_d]$ to be discussed later
- Addition parameters (given k_d)
 - $p_{a,1}[k_d] = 0.5$, $p_{a,1}[k_d \pm 1] = 0.15$, $p_{a,1}[k_d \pm j] = 0.2(2^{-j})$ for $j > 2$
(before normalization)

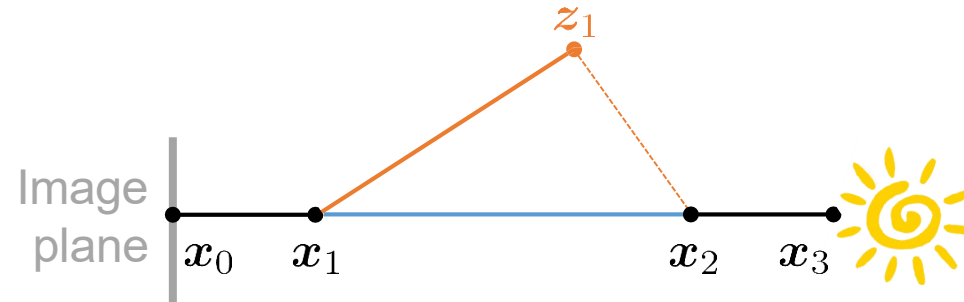
Evaluating Transition Probability



- The probability for transitioning from $\bar{x}$ to $\bar{y}$ is

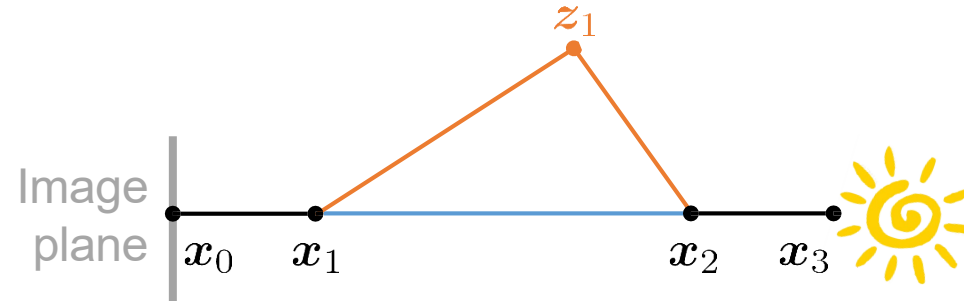
$$\begin{aligned}
 g(\bar{x} \rightarrow \bar{y}) &= p_d[l, m] \sum_{l'=0}^{k_a-1} p_a[l', k_a - 1 - l'] p((z_1, \dots, z_{l'})) p((z_{k_a-1}, \dots, z_{l'+1})) \\
 &= p_d[l, m] \sum_{l'=0}^{k_a-1} p_a[l', k_a - 1 - l'] \left(\prod_{i=1}^{l'} p_A(z_i) \right) \left(\prod_{i=k_a-1}^{l'+1} p_A(z_i) \right)
 \end{aligned}$$

Bidirectional Mutation: Example



- Original path: $\bar{x} = (x_0, x_1, x_2, x_3)$
- Mutation parameters:
 - $l = 1, m = 2$ (deletion); $l' = 1, m' = 0$ (addition)
- Mutated path: $\bar{y} = (x_0, x_1, z_1, x_2, x_3)$
- The probability to accept $\bar{y}$ equals
$$a(\bar{x} \rightarrow \bar{y}) = \min \left\{ 1, \frac{f(\bar{y})}{f(\bar{x})} \frac{g(\bar{y} \rightarrow \bar{x})}{g(\bar{x} \rightarrow \bar{y})} \right\} = \min \left\{ 1, \frac{q(\bar{y} \rightarrow \bar{x})}{q(\bar{x} \rightarrow \bar{y})} \right\},$$
where $q(\bar{u} \rightarrow \bar{v}) := g(\bar{u} \rightarrow \bar{v}) / f(\bar{v})$

Bidirectional Mutation: Example



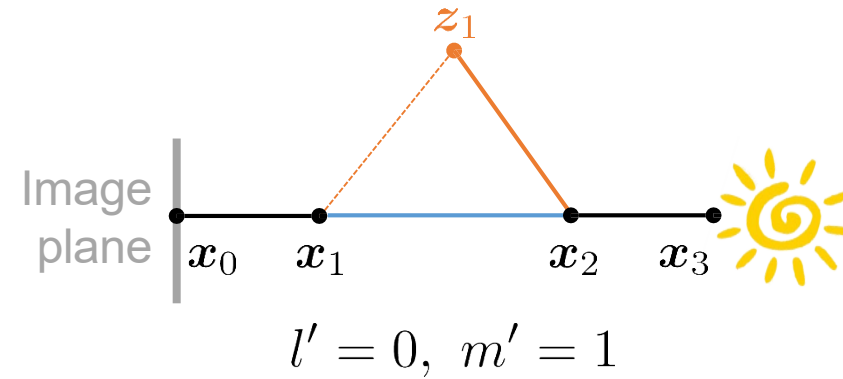
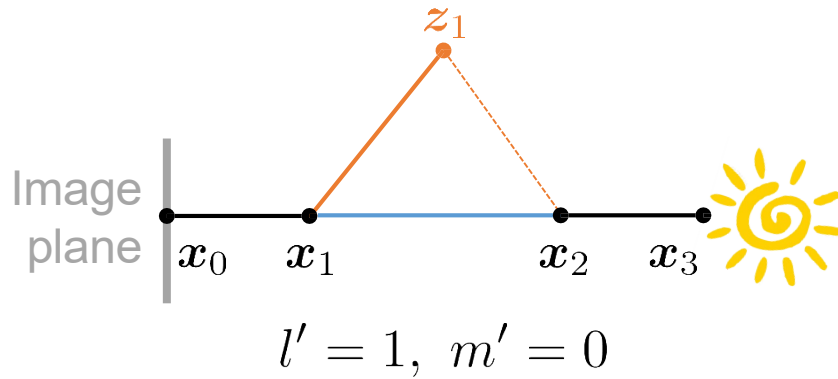
- $q(\bar{x} \rightarrow \bar{y}) := g(\bar{x} \rightarrow \bar{y}) / f(\bar{y})$, where

$$f(\bar{y}) = G(x_0 \leftrightarrow x_1) f_s(z_1 \rightarrow x_1 \rightarrow x_0) G(x_1 \leftrightarrow z_1) f_s(x_2 \rightarrow z_1 \rightarrow x_1) \\ G(z_1 \leftrightarrow x_2) f_s(x_3 \rightarrow x_2 \rightarrow z_1) G(x_2 \leftrightarrow x_3) L_e(x_3 \rightarrow x_2)$$

- Recall that $f(\bar{y})$ does not involve $W_e(x_1 \rightarrow x_0)$ as it is captured by the filter function $h^{(i)}$

Bidirectional Mutation: Example

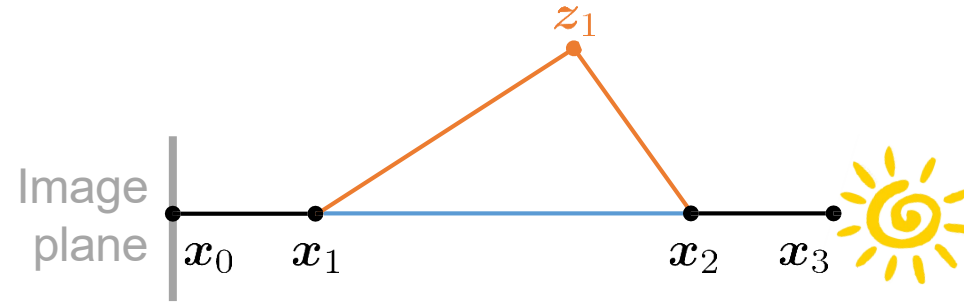
- $\bar{y}$ can be generated from $\bar{x}$ in two ways



- Thus,

$$\begin{aligned}
 g(\bar{x} \rightarrow \bar{y}) &= p_d[1, 2] (p_a[1, 0] p_A(z_1 \mid \mathbf{x}_1) + p_a[0, 1] p_A(z_1 \mid \mathbf{x}_2)) \\
 &= p_d[1, 2] (p_a[1, 0] p_{\sigma^\perp}(\mathbf{x}_1 \rightarrow z_1) G(\mathbf{x}_1 \leftrightarrow z_1) + \\
 &\quad p_a[0, 1] p_{\sigma^\perp}(\mathbf{x}_2 \rightarrow z_1) G(\mathbf{x}_2 \leftrightarrow z_1))
 \end{aligned}$$

Bidirectional Mutation: Example



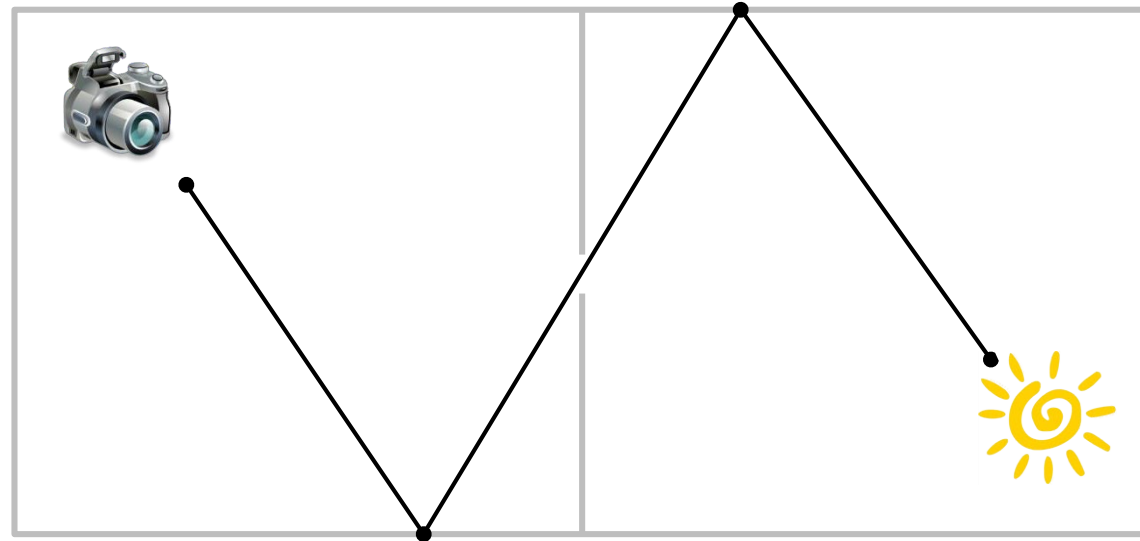
- $q(\bar{y} \rightarrow \bar{x}) := g(\bar{y} \rightarrow \bar{x}) / f(\bar{x})$, where

$$f(\bar{x}) = G(x_0 \leftrightarrow x_1) f_s(x_2 \rightarrow x_1 \rightarrow x_0) G(x_1 \leftrightarrow x_2) f_s(x_3 \rightarrow x_2 \rightarrow x_1) \\ G(x_2 \leftrightarrow x_3) L_e(x_3 \rightarrow x_2)$$
- To obtain $\bar{x}$ from $\bar{y}$ using bidirectional path mutation, we need $l = 1$, $m = 3$ and $l' = m' = 0$. Thus,

$$g(\bar{y} \rightarrow \bar{x}) = p_d[1, 3] p_a[0, 0]$$

Path Perturbations

- “Smaller” mutations
- Useful for finding “nearby” paths with high contribution

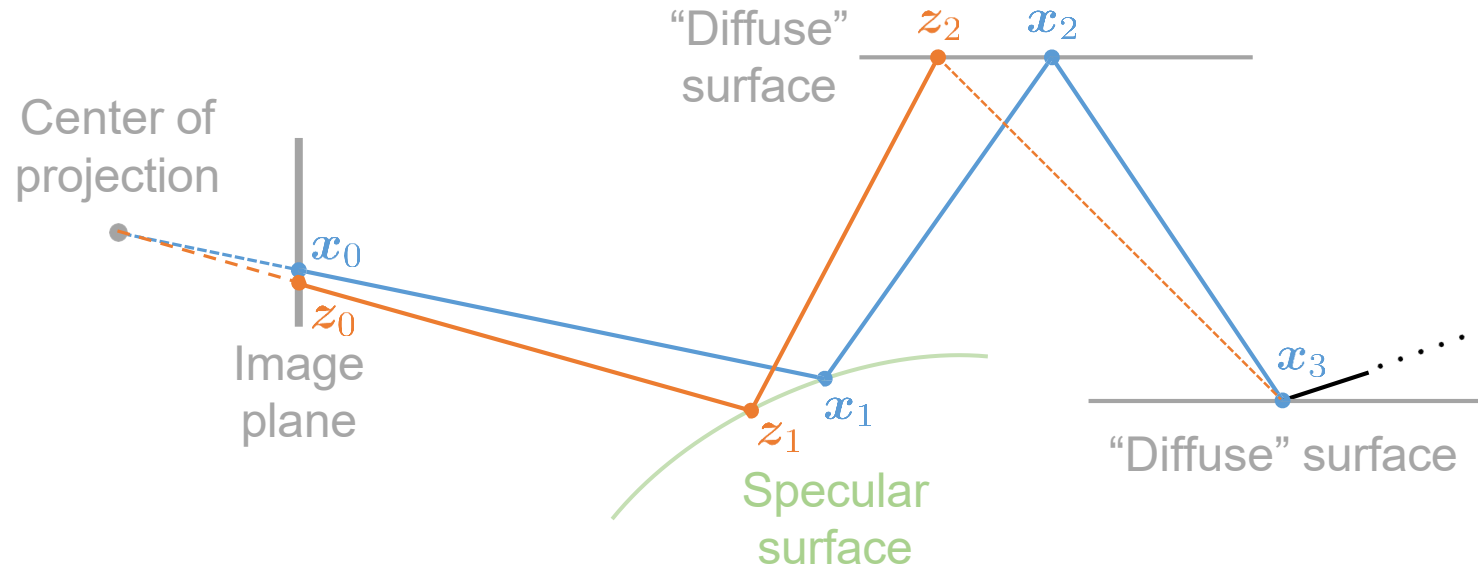


Path Perturbations

- Basic idea: choosing a sub-path and moving the vertices slightly
- Three types of perturbations
 - Lens
 - Caustic
 - Multi-chain

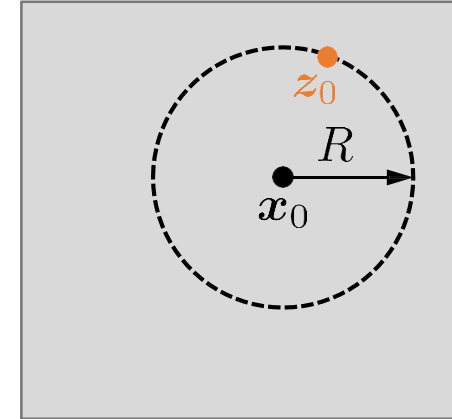
Path Perturbation: Lens

- Replace sub-paths $(\mathbf{x}_0, \dots, \mathbf{x}_m)$ of the form ES*D(D|L)
- Randomly move the endpoint $\mathbf{x}_0$ on the image plane to $\mathbf{z}_0$
- Trace a ray through $\mathbf{z}_0$ to form the new sub-path



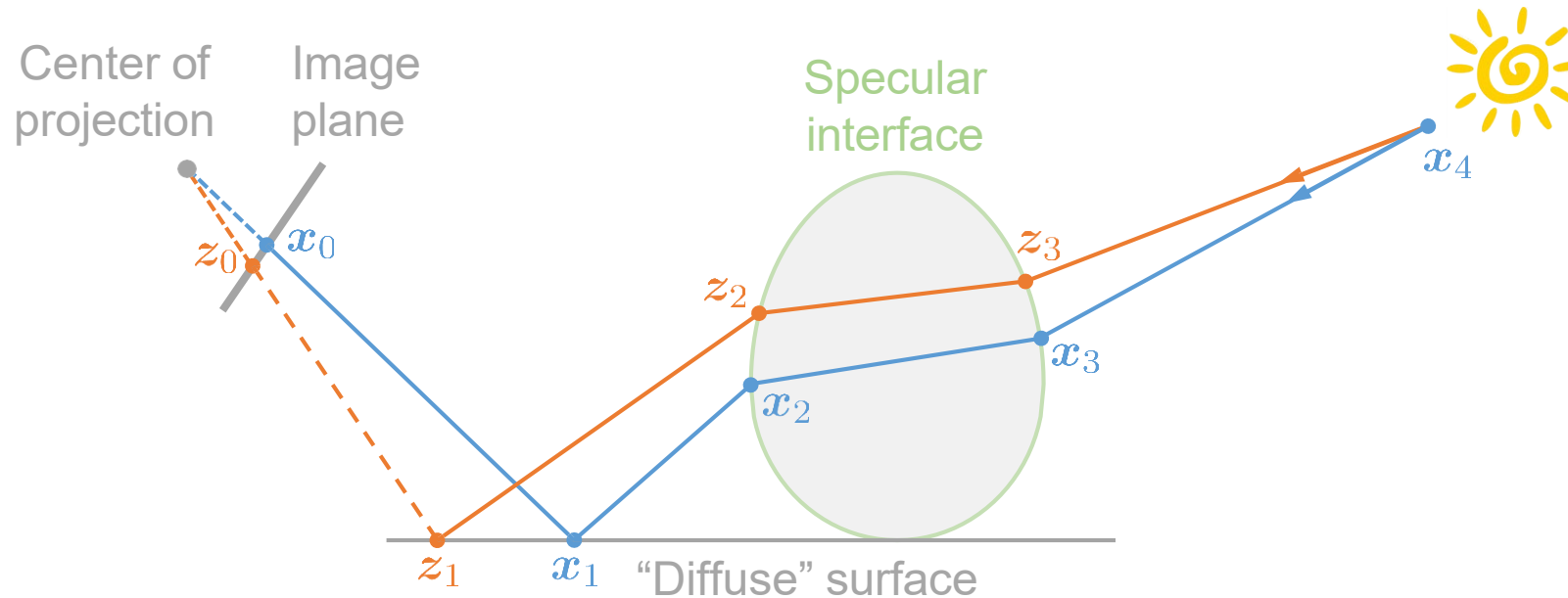
Path Perturbation: Lens

- To draw $\mathbf{z}_0$:
 - First, sample a distance R using $R = r_2 \exp(-\log(r_2/r_1) \xi)$ where $\xi \sim U(0, 1)$
 - Then, uniformly sample $\mathbf{z}_0$ from the circle which is center at $\mathbf{x}_0$ and has radius R
- The mutation is immediately rejected if ray tracing through $\mathbf{z}_0$ fails to generate a new sub-path with exactly the same form (i.e., $ES^*D(D|L)$)
- Otherwise, the acceptance probability is evaluated in a way similar to the bidirectional mutation case



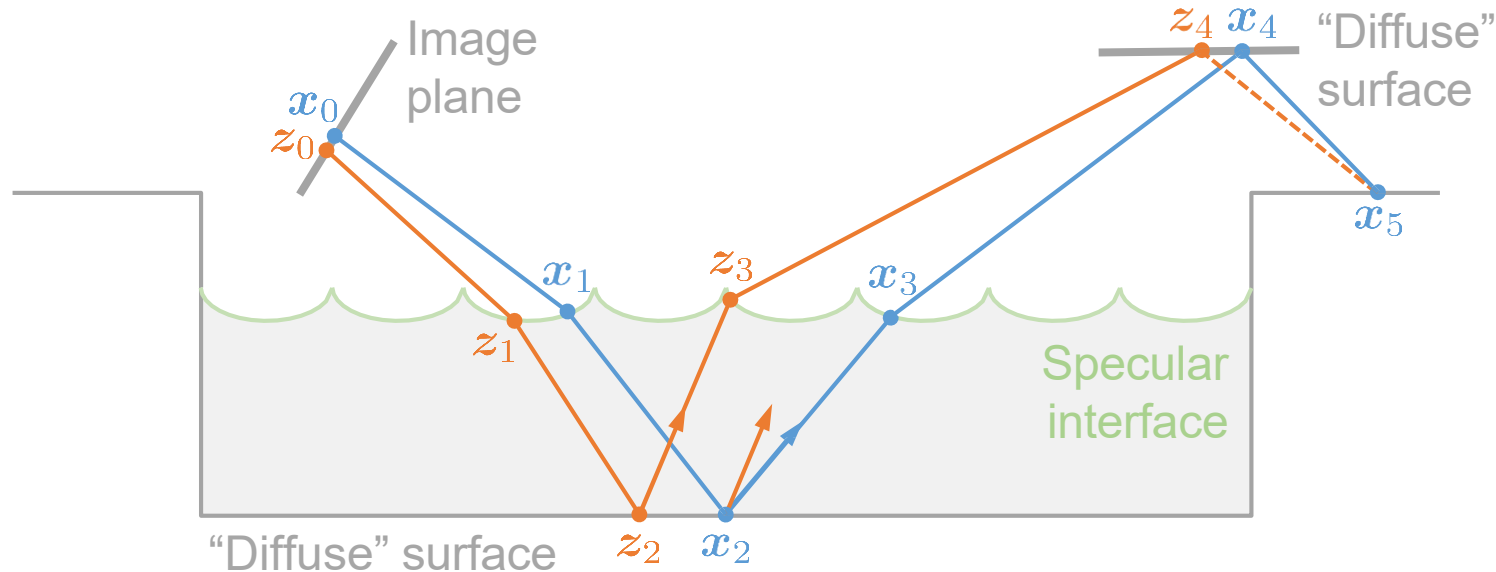
Path Perturbation: Caustic

- Replace **sub-paths** ($\mathbf{x}_0, \dots, \mathbf{x}_m$) of the form EDS*(D|L)
- Slightly modify the direction $\mathbf{x}_m \rightarrow \mathbf{x}_{m-1}$ (at random)
- Trace a ray from $\mathbf{x}_m$ with this new direction to form the new **sub-path**



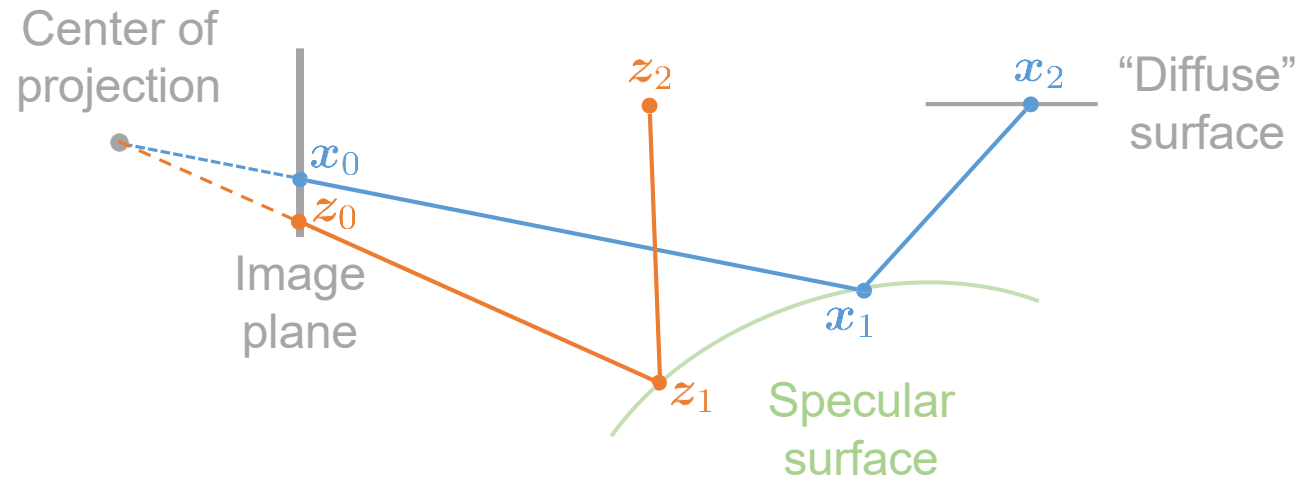
Path Perturbation: Multi-Chain

- Replace **sub-paths** of the form $ES+DS+D(D|L)$
- Lens perturbation is applied for $ES+D$
- The direction of the DS^+ edge in the **original sub-path** is perturbed
- The new direction is then used to complete the $DS^+D(D|L)$ segment of the **new sub-path** (using ray tracing)



Lens Sub-Path Mutation

- Used to stratify samples over the image plane
 - Each pixel should get enough sample paths
- Replace **lens sub-paths** of the form $ES^*(D|L)$
 - Similar to lens perturbation, but draw z_0 from a different density



Selecting Between Mutation Types

- Path mutations strategies introduced so far:
 - Bidirectional mutation
 - Lens, caustic, multi-chain perturbations
 - Lens sub-path mutation
- Choose one randomly in each iteration
- Or, make mutation selection part of sampling!

Refinements

- Direct lighting
 - It is more efficient to estimate direct illumination with standard methods (e.g., area & BSDF sampling combined using MIS) and apply MLT only for indirect illumination
- Importance sampling for mutation probabilities
 - For increasing the average acceptance probability $a(\bar{x} \rightarrow \bar{y})$

Improving Acceptance Rates

- Recall:

$$a(\bar{x} \rightarrow \bar{y}) = \min \left\{ 1, \frac{q(\bar{y} \rightarrow \bar{x})}{q(\bar{x} \rightarrow \bar{y})} \right\}, \text{ where } q(\bar{u} \rightarrow \bar{v}) := \frac{g(\bar{u} \rightarrow \bar{v})}{f(\bar{v})}$$

- **Observation:** given a path $\bar{x} = (x_0, \dots, x_k)$ and $-1 \leq l < m \leq k + 1$, $q(\bar{y} \rightarrow \bar{x}) = g(\bar{y} \rightarrow \bar{x}) / f(\bar{x})$ can be partially evaluated without constructing $\bar{y}$
- $f(\bar{x})$ can be fully evaluated
- $g(\bar{y} \rightarrow \bar{x})$ can be partially evaluated

Improving Acceptance Rates



- Let $k_a = m - l - 1$, then

$$g(\bar{y} \rightarrow \bar{x}) = \underbrace{p_d}_{\text{Unknown}} \sum_{l'=0}^{k_a-1} \underbrace{p_a}_{\text{Known}} p((\mathbf{x}_{l+1}, \dots, \mathbf{x}_{l+l'})) p((\mathbf{x}_{m-1}, \dots, \mathbf{x}_{l+l'+1}))$$

- Set the unknown term to one and get a weight $w_{l,m}$ for each mutation

$$w_{l,m}(\bar{x}) = \frac{\sum_{l'=0}^{k_a-1} p((\mathbf{x}_{l+1}, \dots, \mathbf{x}_{l+l'})) p((\mathbf{x}_{m-1}, \dots, \mathbf{x}_{l+l'+1}))}{f(\bar{x})}$$

Improving Acceptance Rates

$$w_{l,m}(\bar{x}) = \frac{\sum_{l'=0}^{k_a-1} p((\mathbf{x}_{l+1}, \dots, \mathbf{x}_{l+l'})) p((\mathbf{x}_{m-1}, \dots, \mathbf{x}_{l+l'+1}))}{f(\bar{x})}$$

- Given a path $\bar{x} = (\mathbf{x}_0, \dots, \mathbf{x}_k)$, we can evaluate the weights for several possible mutation strategies and use these weights to sample one
- Can be used to obtain $p_{d,2}$ for bidirectional mutations
 - Given k_d , simply make $p_{d,2}[l \mid k_d] \propto w_{l,l+k_d}(\bar{x})$

Results

[Veach 1997]



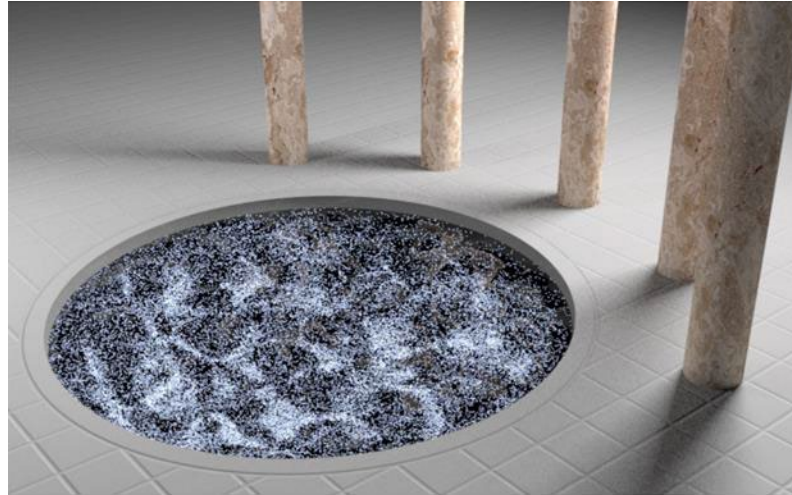
BDPT



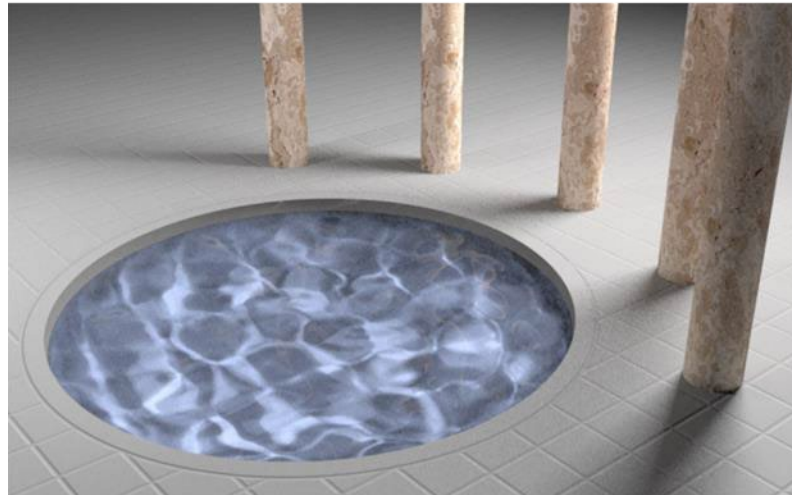
MLT
(equal-time)

Results

[Veach 1997]

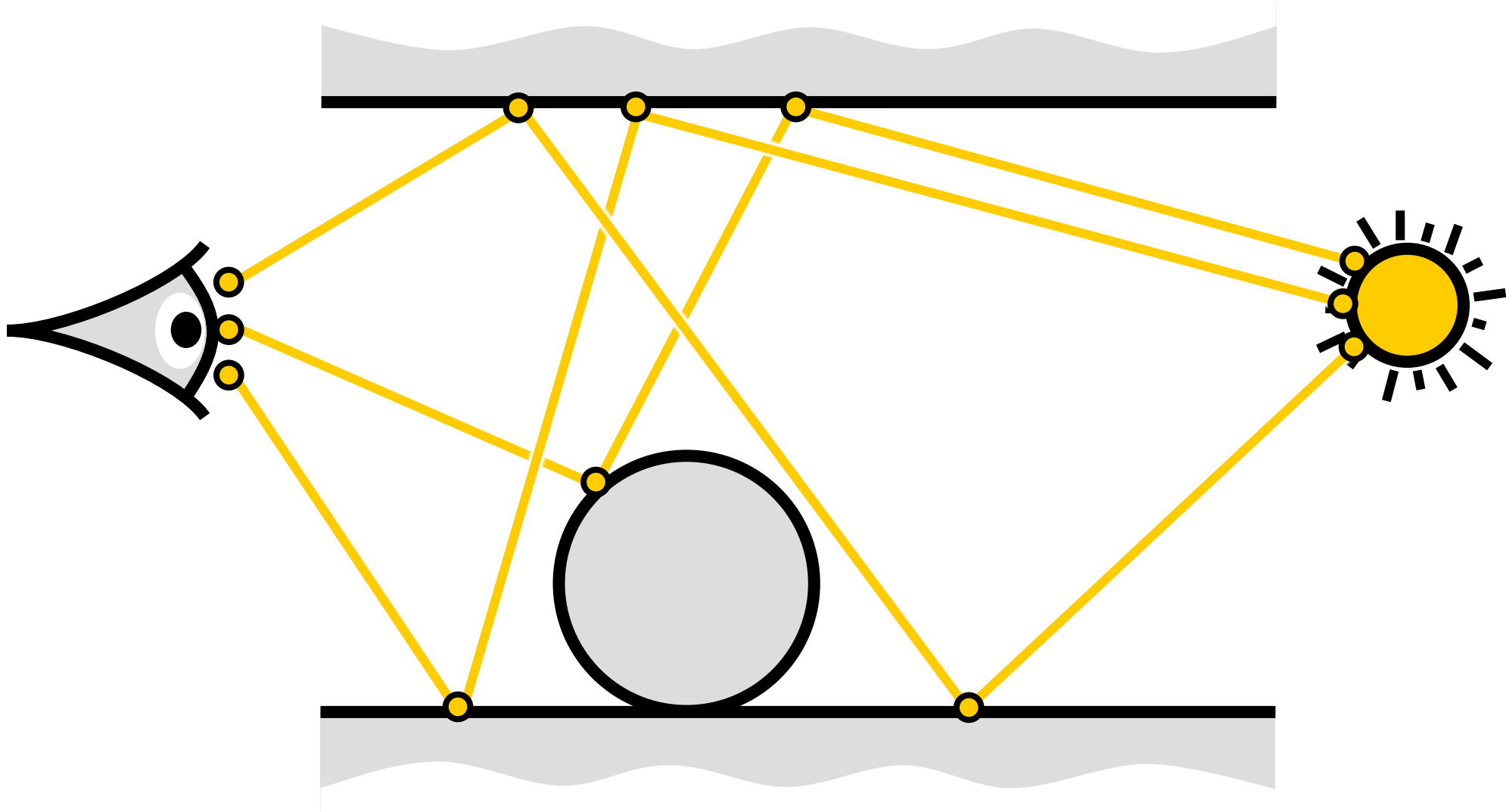


BDPT

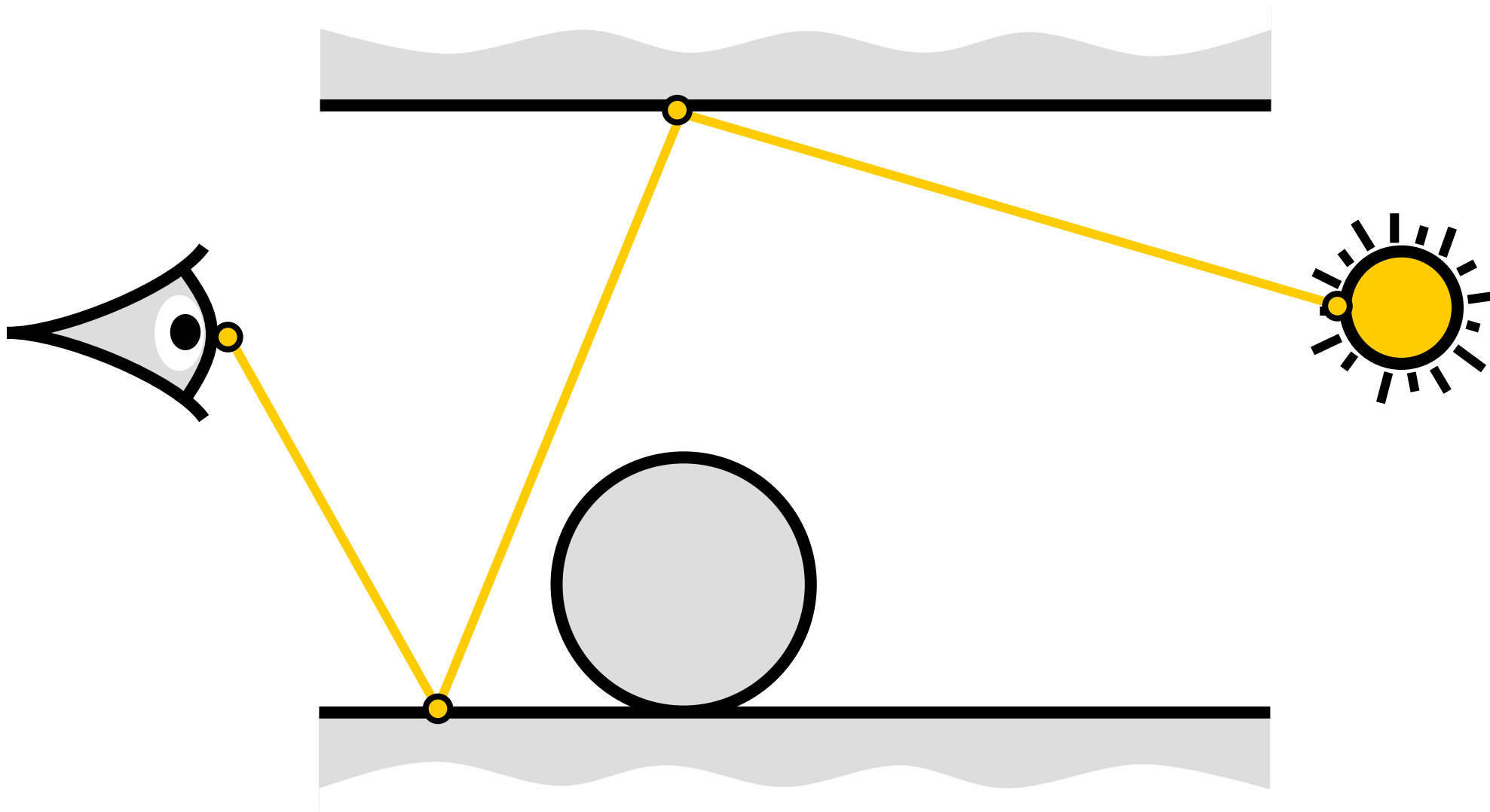


MLT
(equal-time)

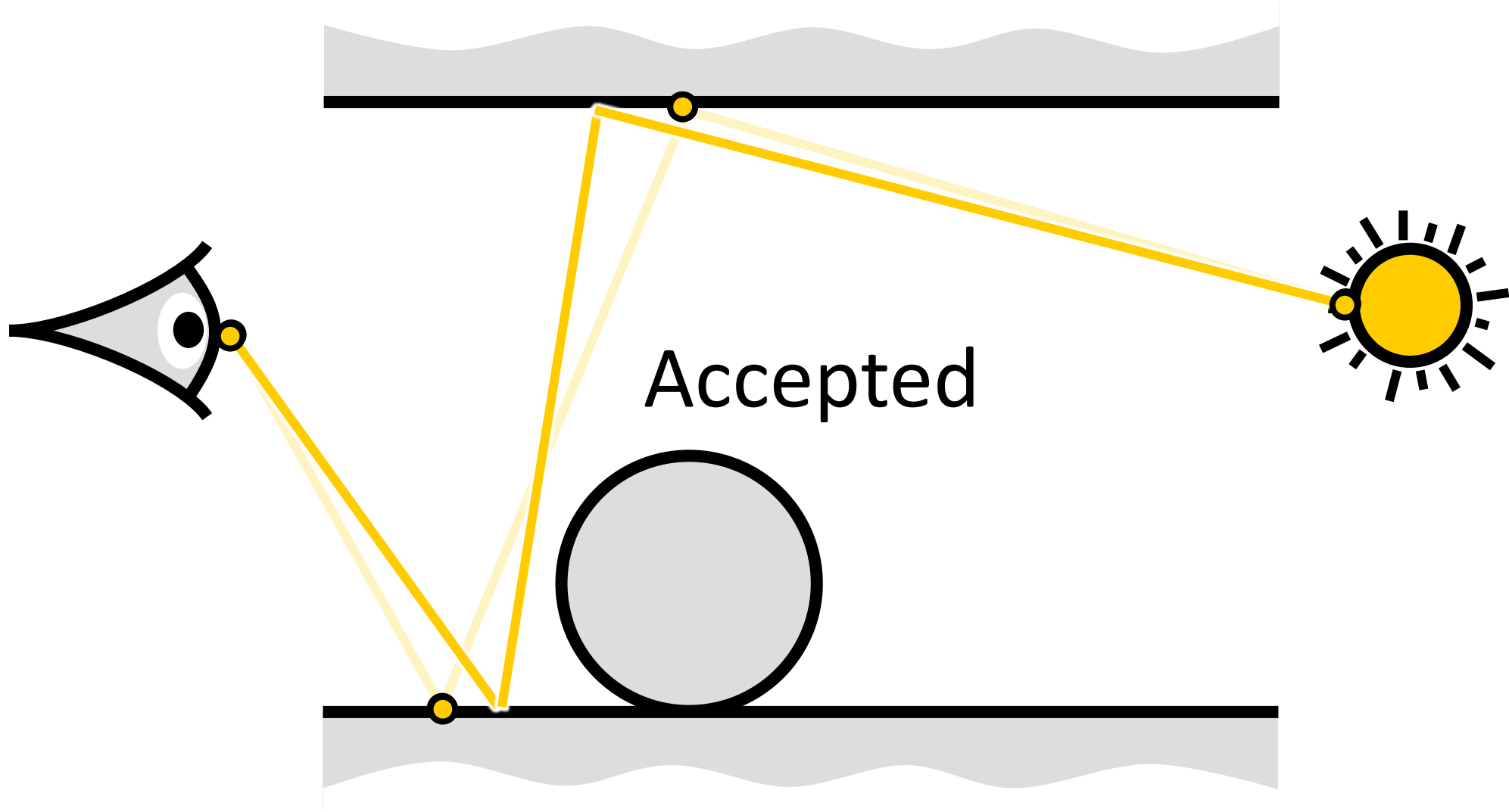
Monte Carlo (MC) rendering



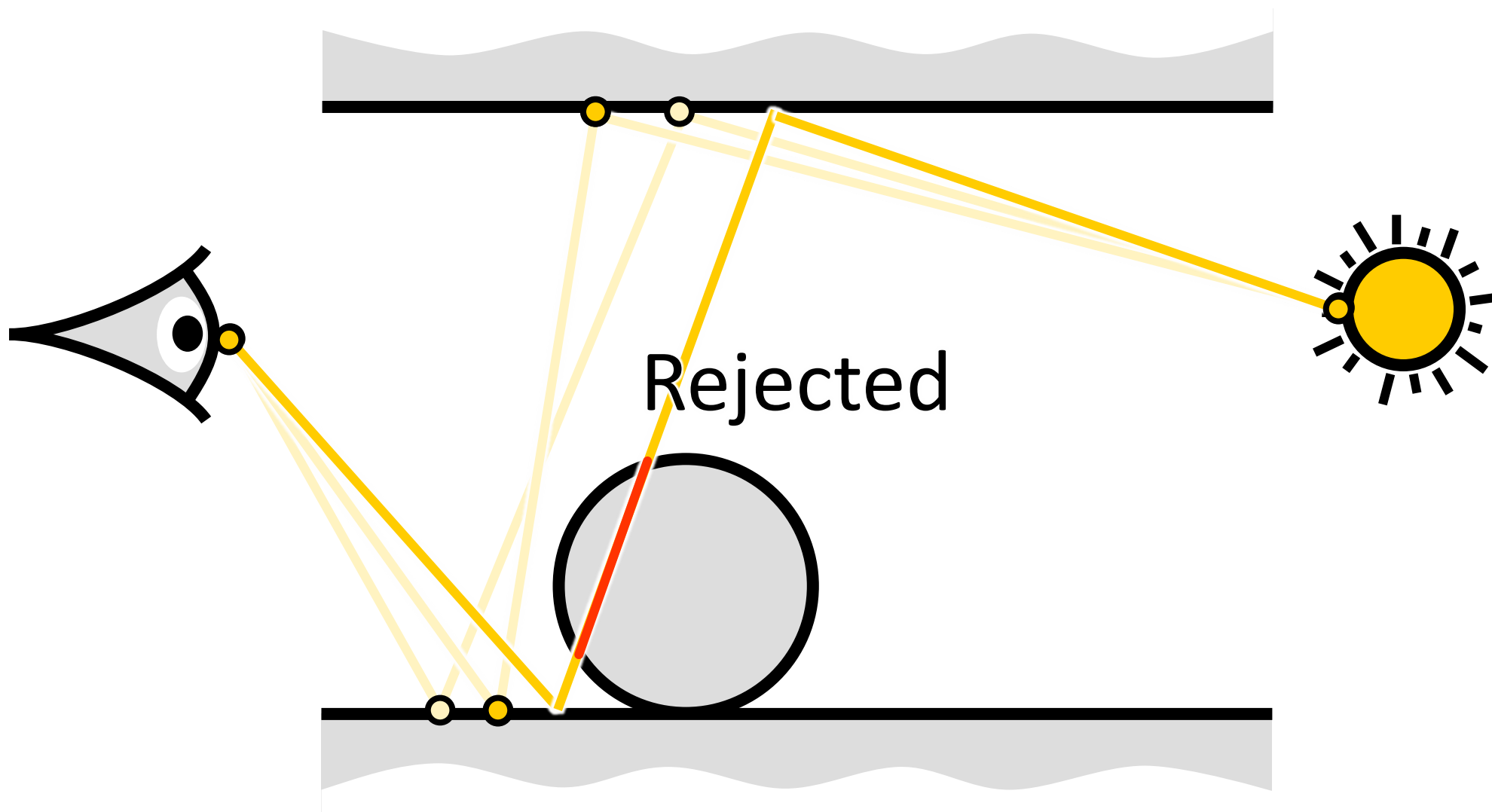
Markov chain Monte Carlo (MCMC) rendering



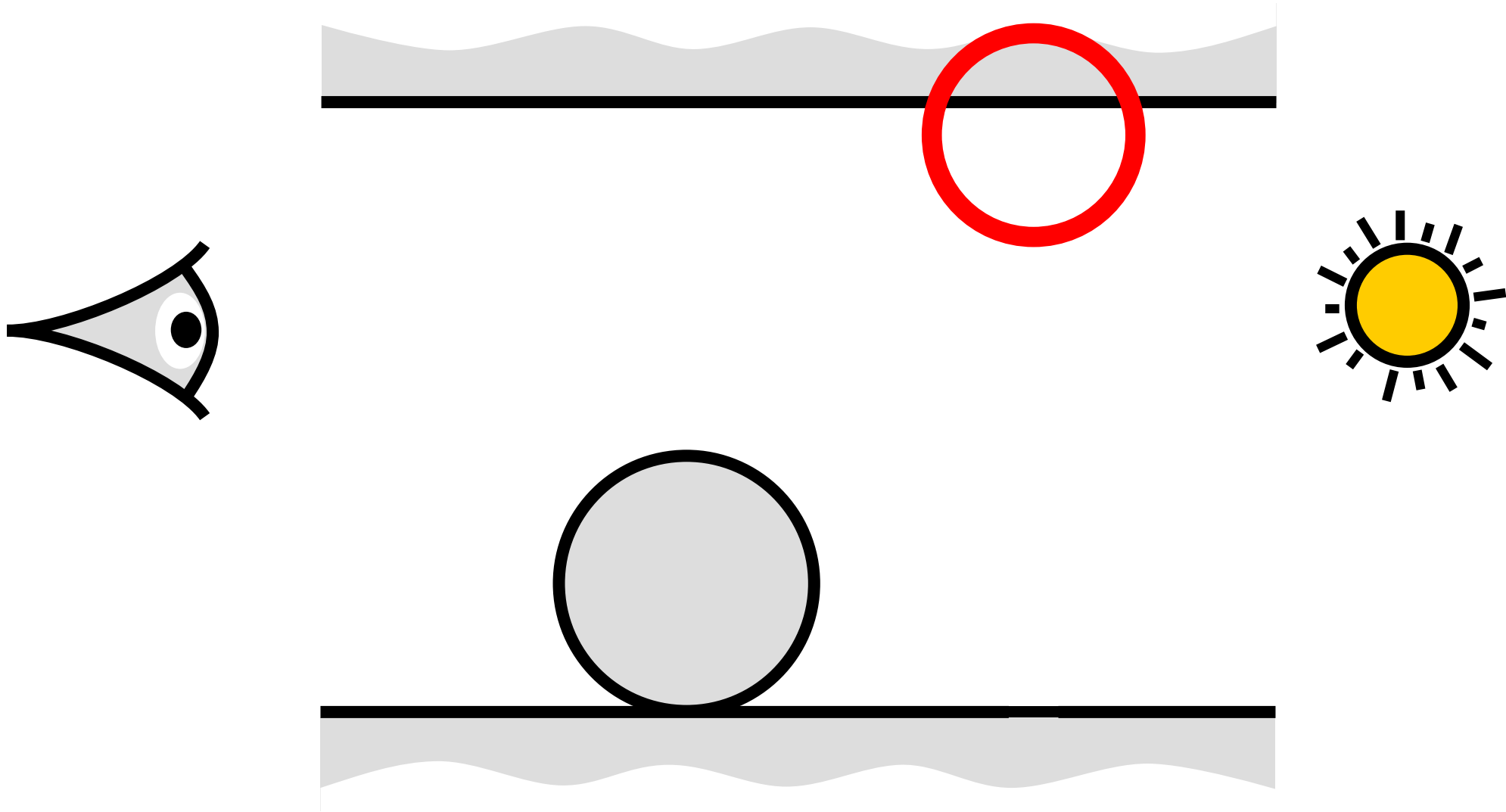
Markov chain Monte Carlo (MCMC) rendering



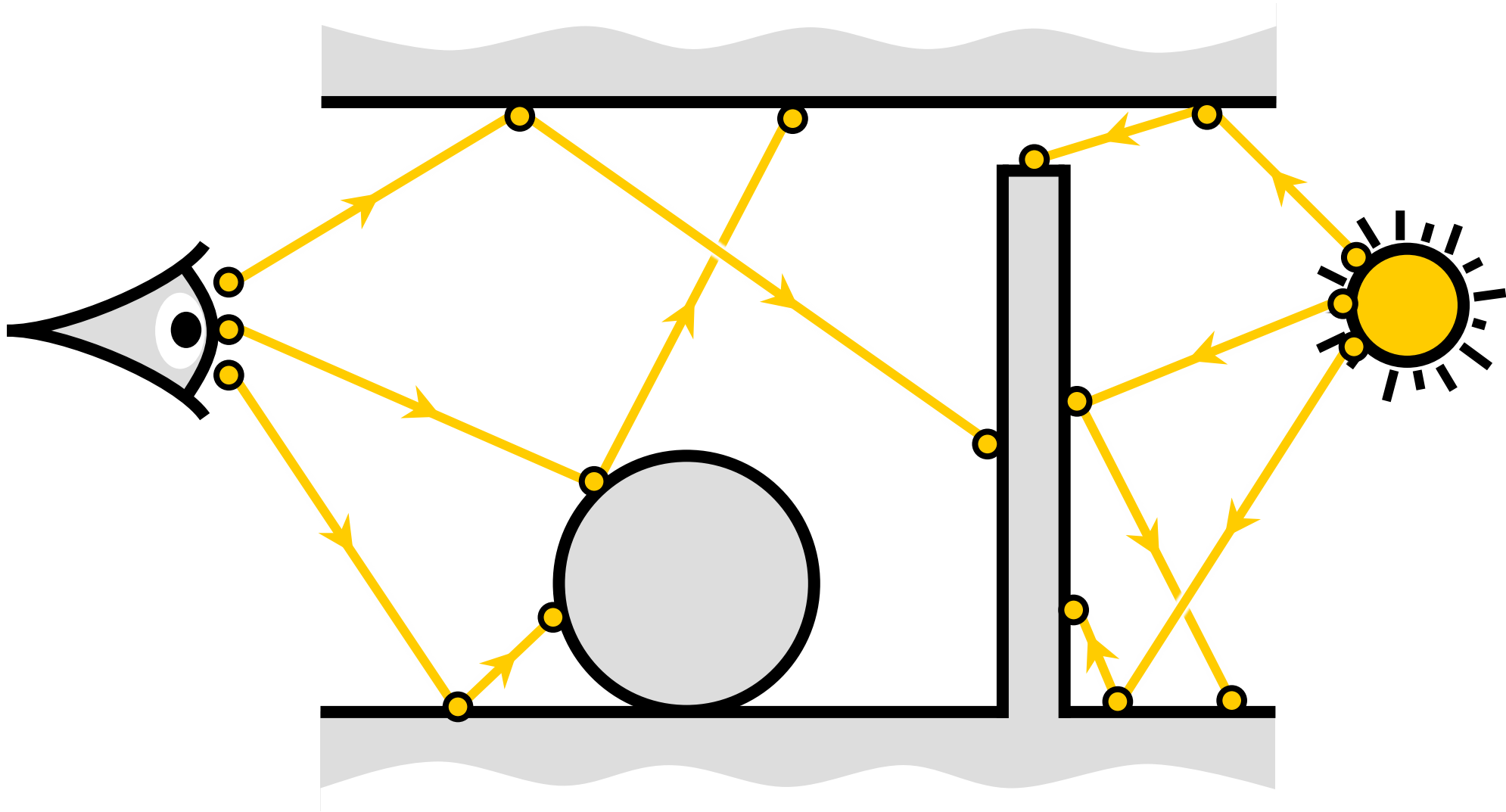
Markov chain Monte Carlo (MCMC) rendering



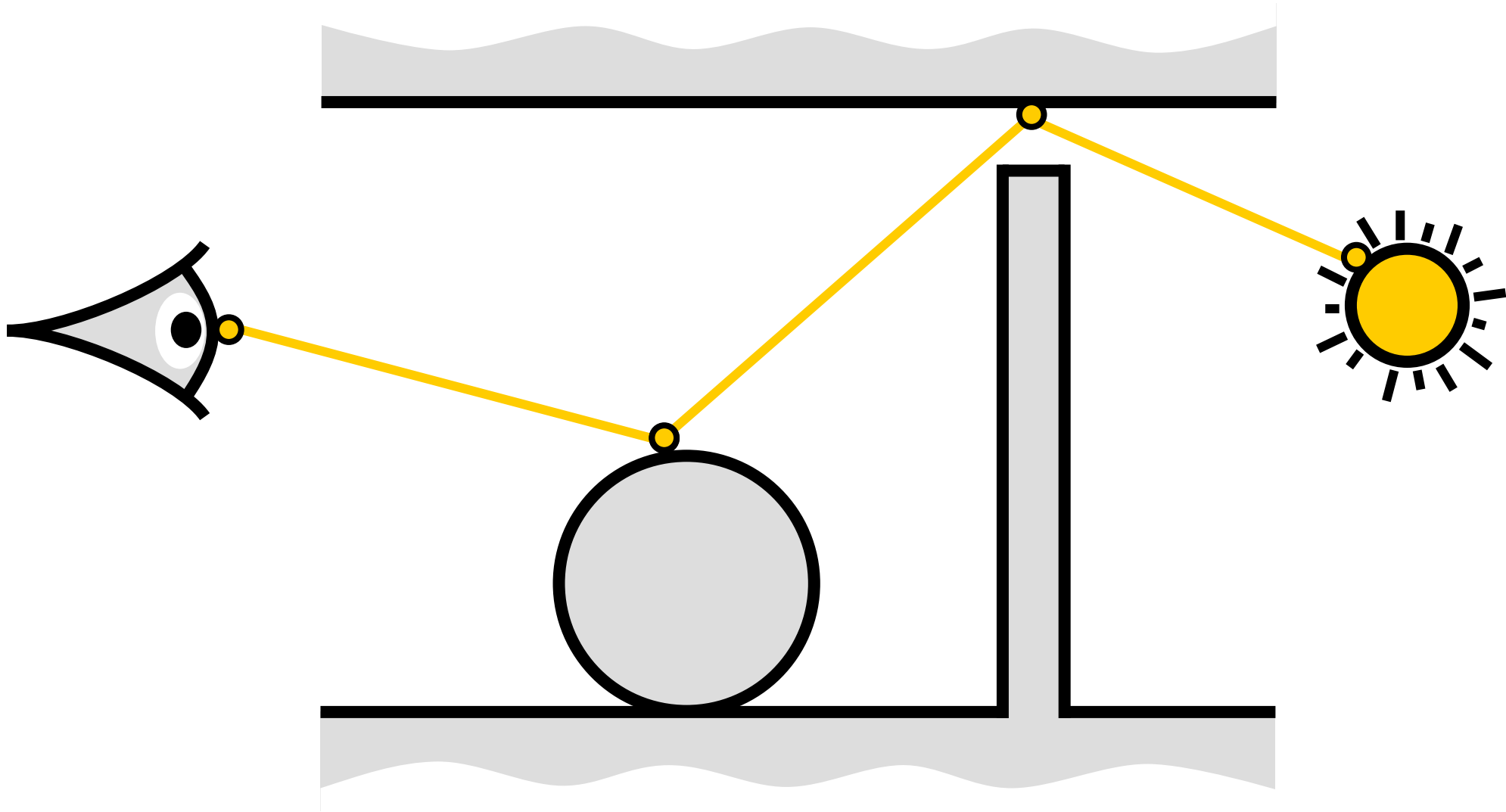
Efficiency of MCMC rendering



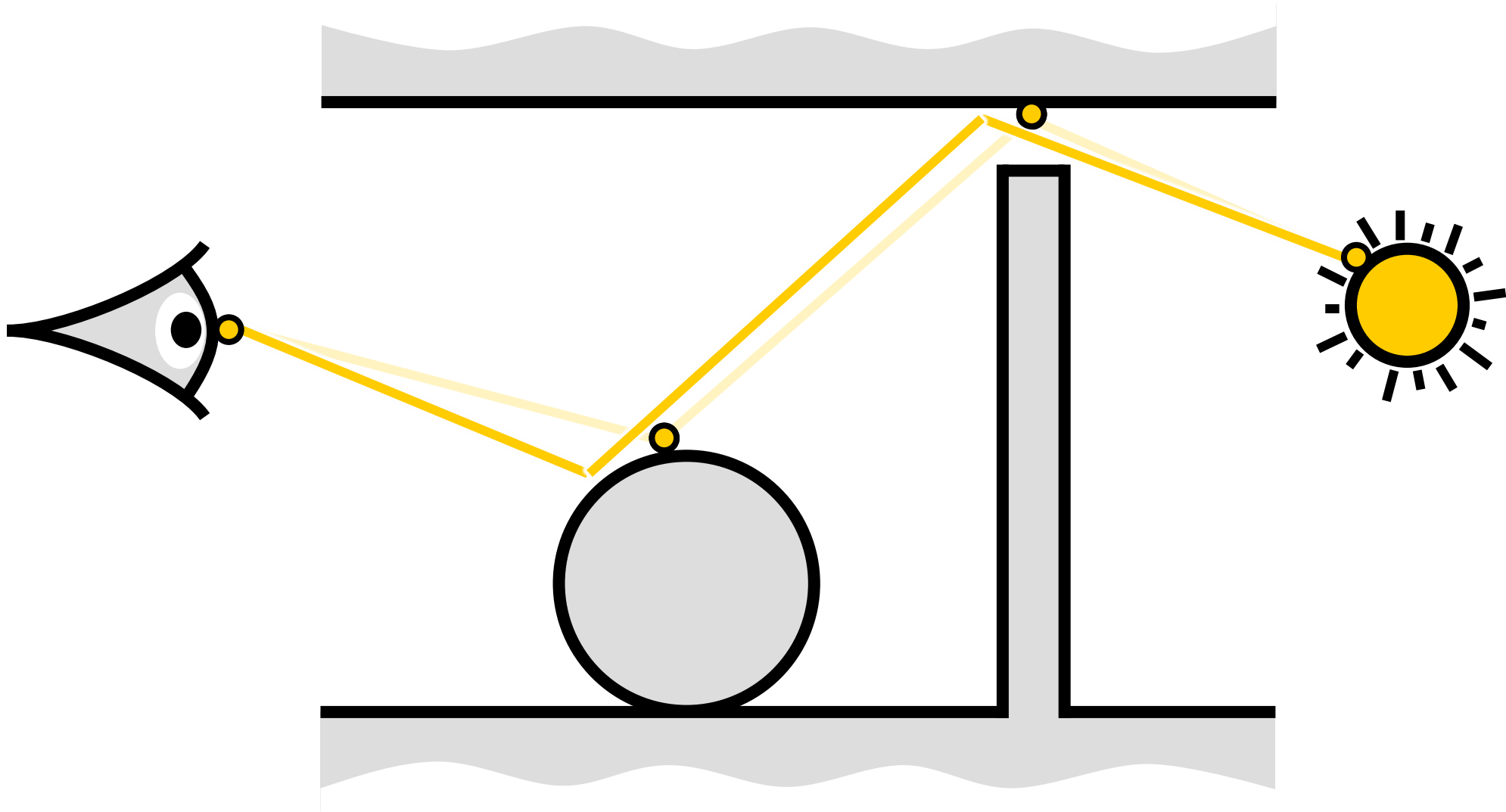
Efficiency of MCMC rendering



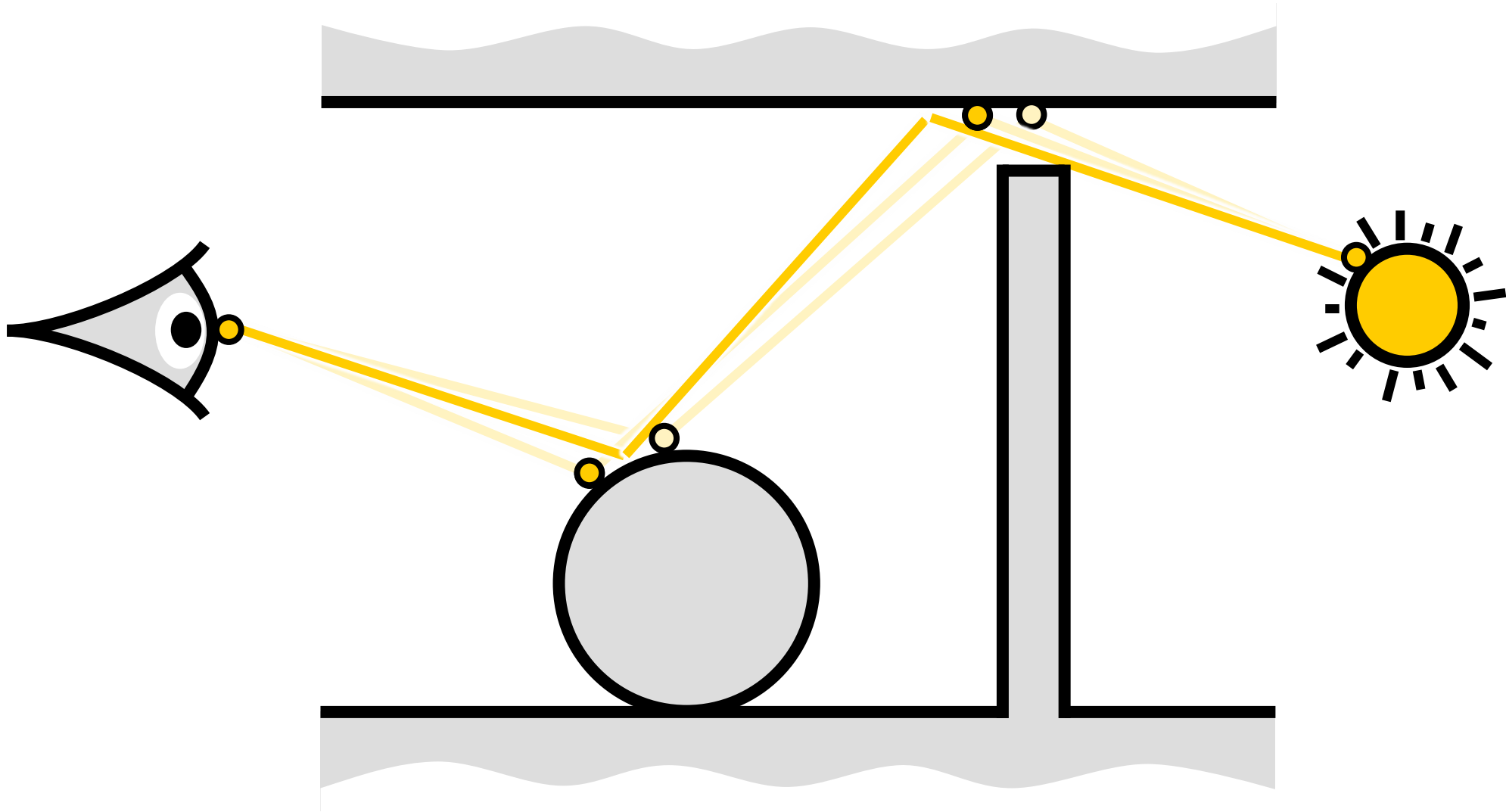
Efficiency of MCMC rendering



Efficiency of MCMC rendering



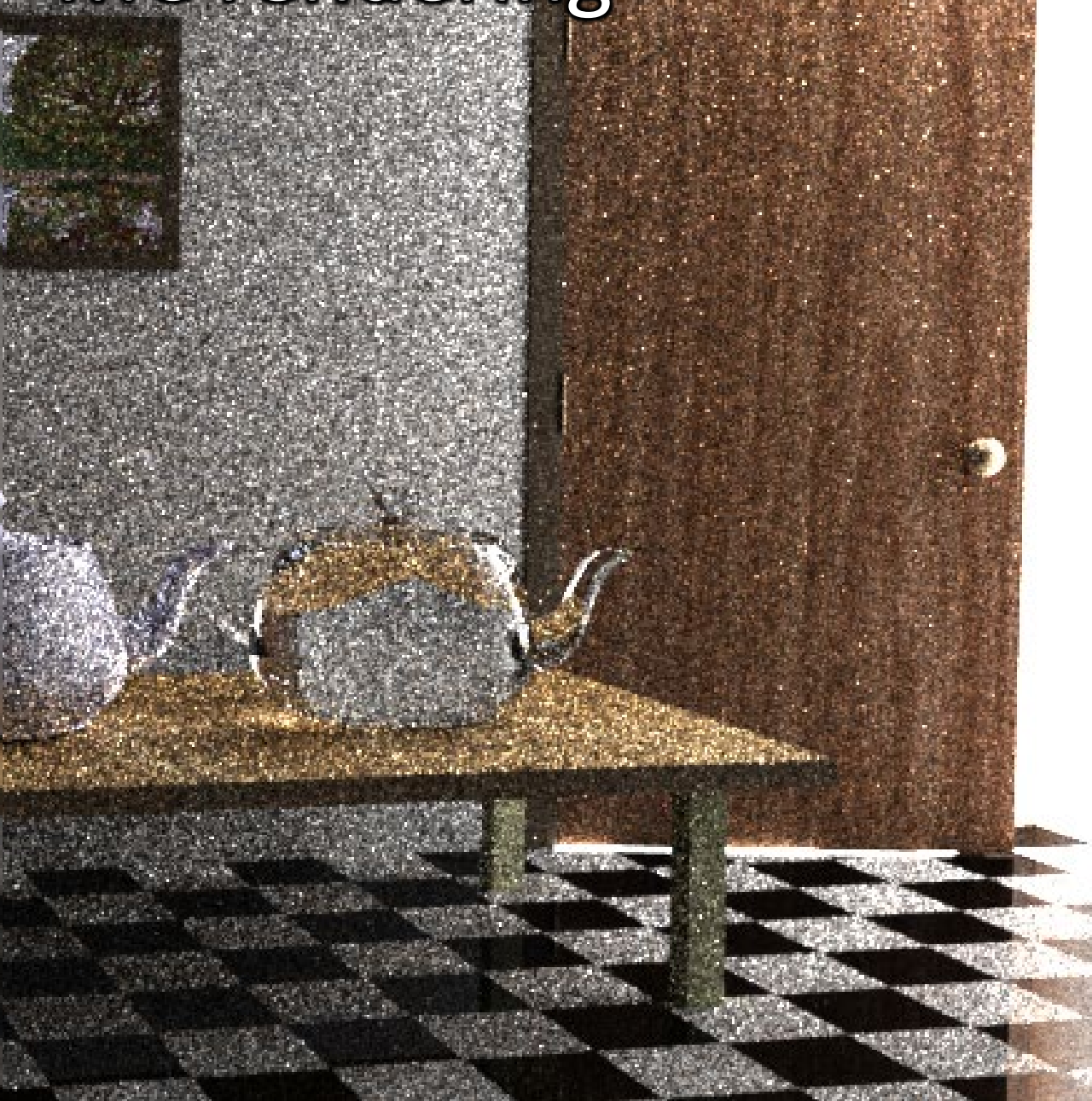
Efficiency of MCMC rendering



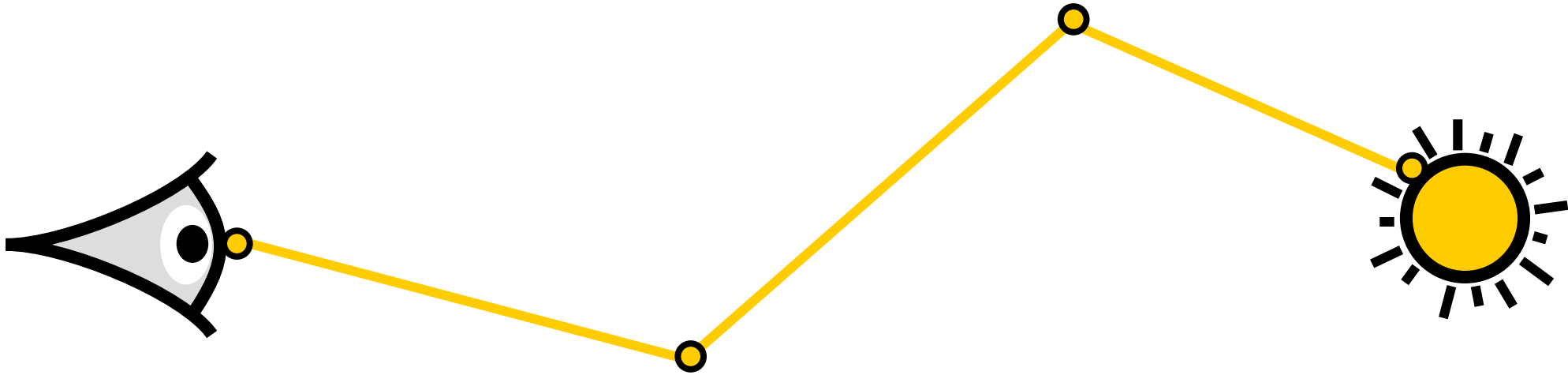
MCMC rendering



MC rendering

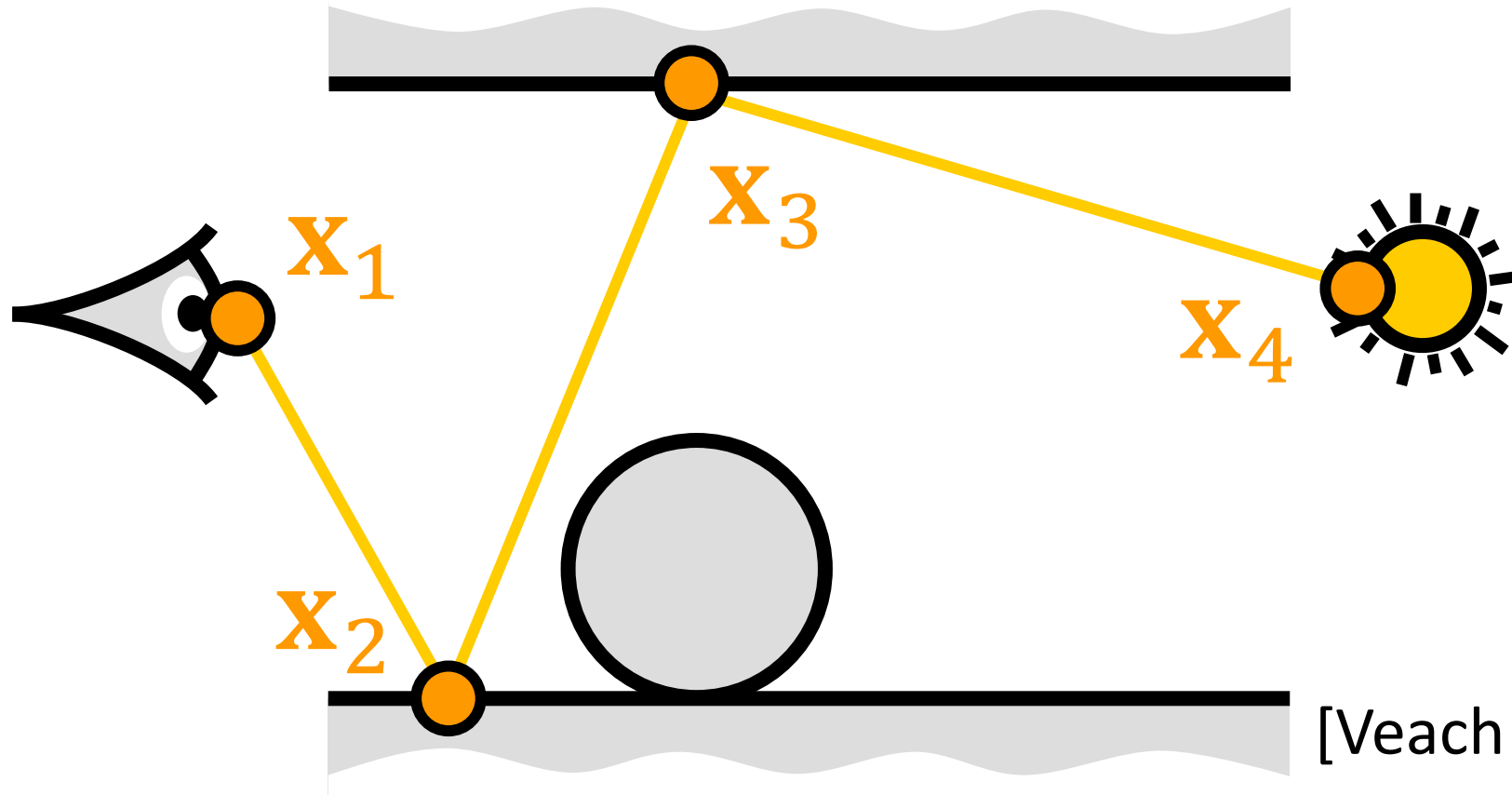


State spaces for MCMC rendering



Describes how a ***path*** is mathematically defined

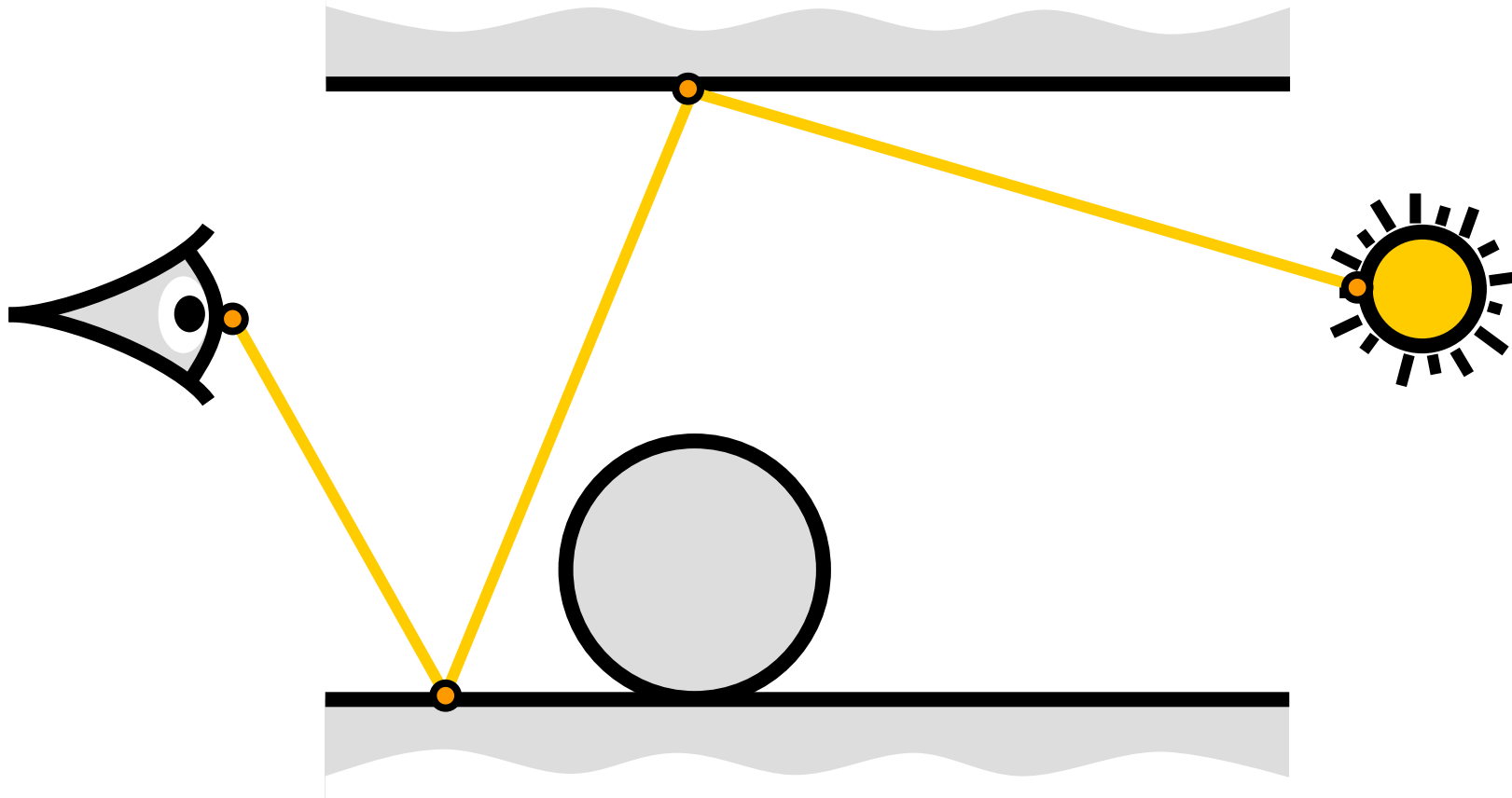
Path space



[Veach & Guibas 1997]

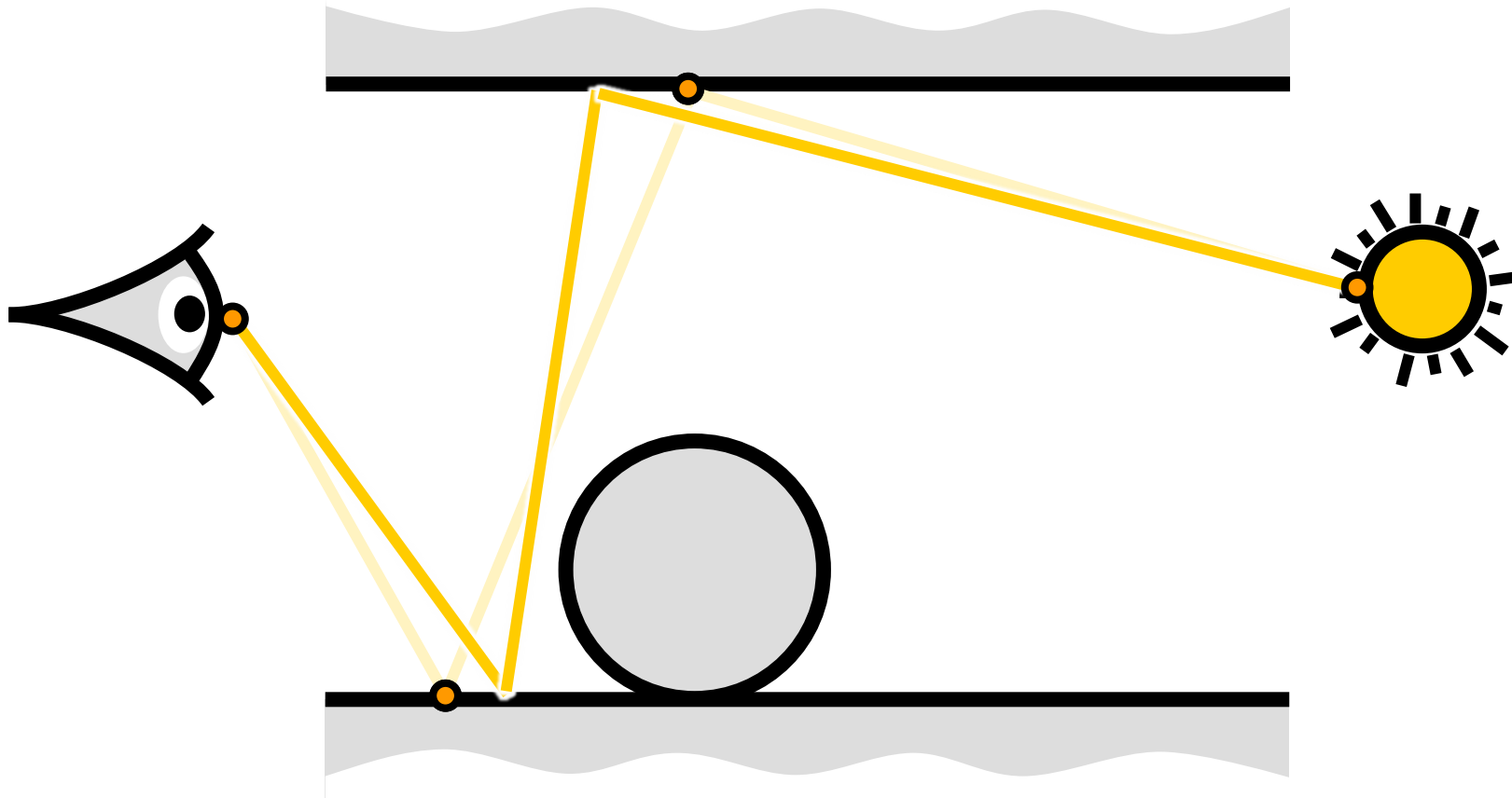
Path is a sequence of **surface points**

Mutation in path space



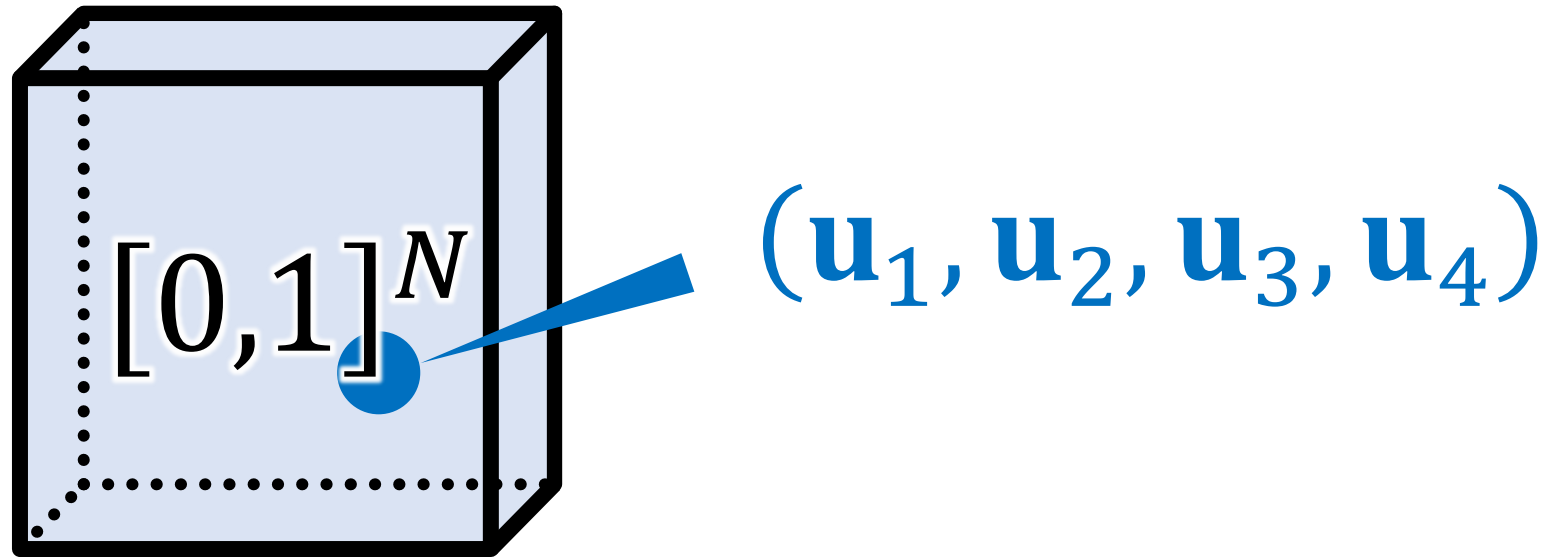
Changes **surface points** *directly*

Mutation in path space



Changes **surface points** *directly*

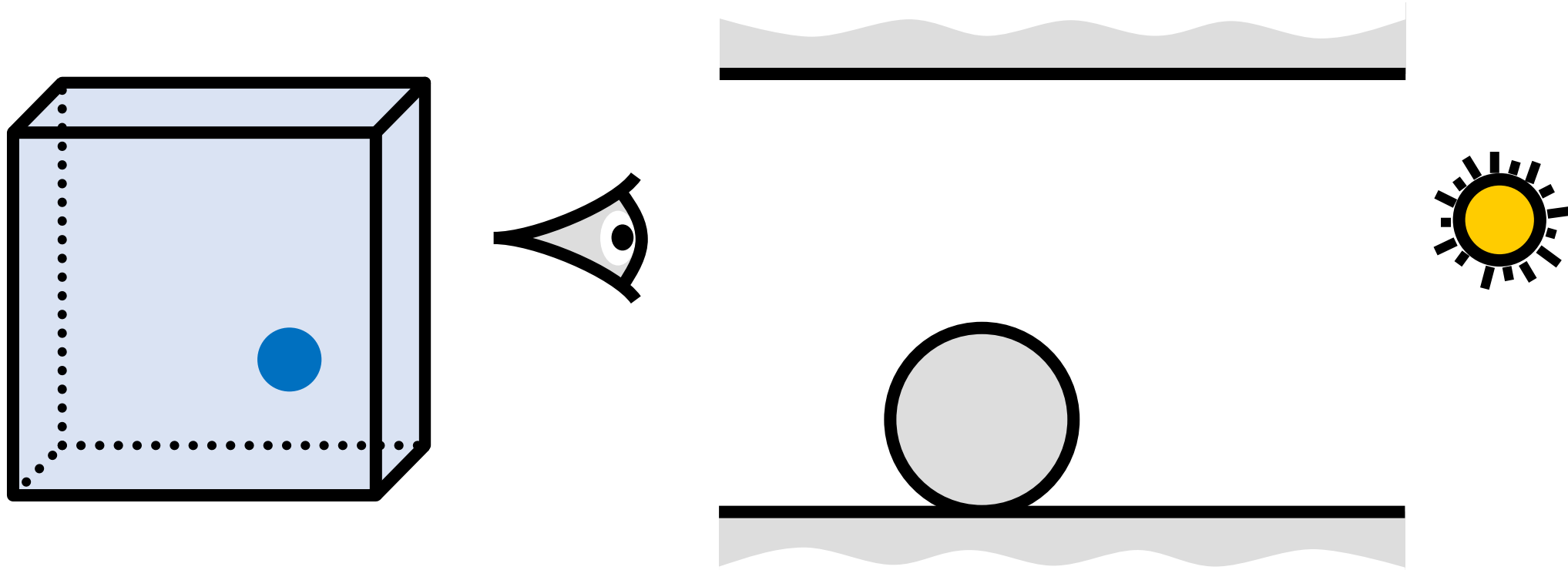
Primary sample space



[Kelemen et al. 2002]

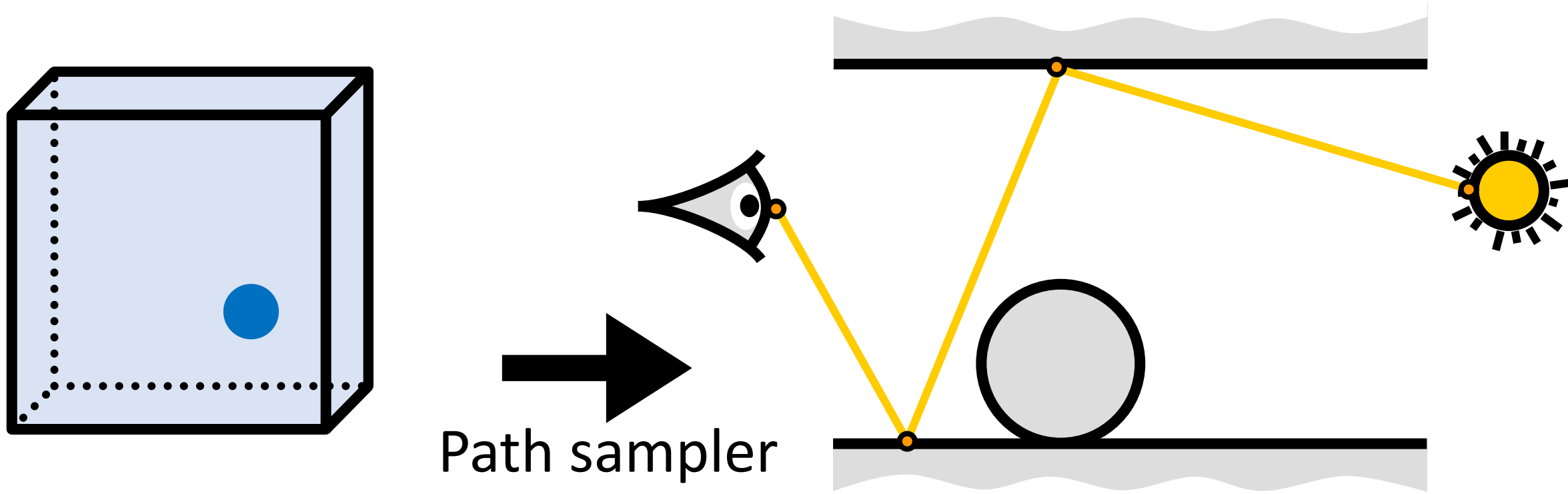
Path is a sequence of **numbers**

Primary sample space



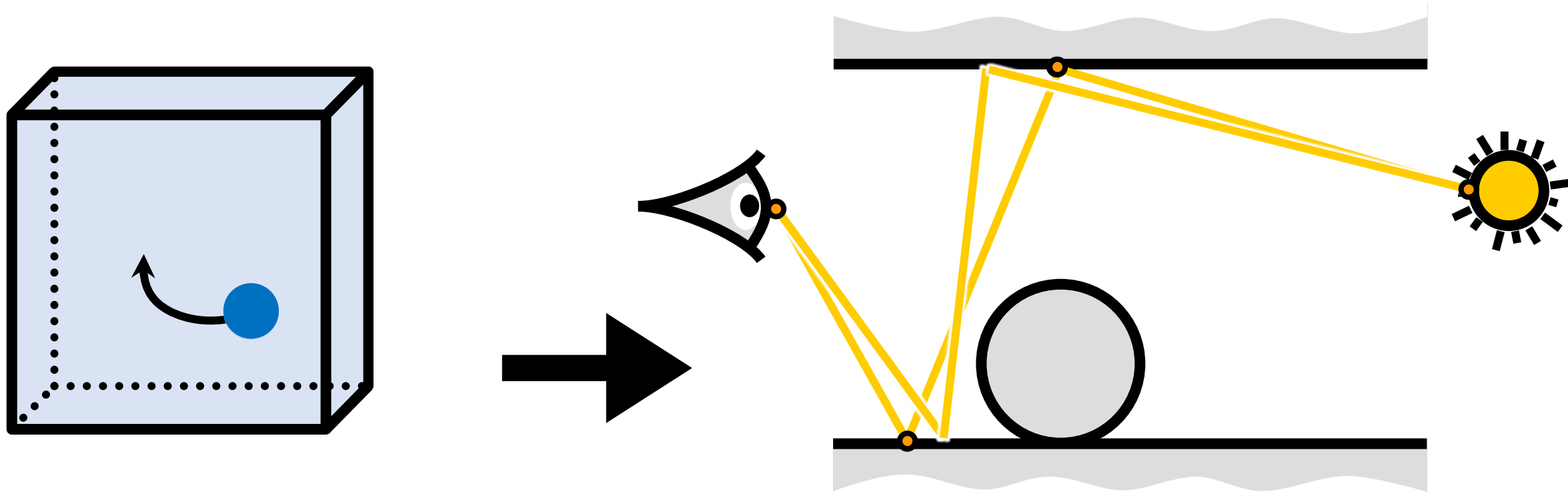
Path is a sequence of **numbers**
mapped to a sequence of **surface points**

Primary sample space



Path is a sequence of **numbers**
mapped to a sequence of **surface points**

Mutation in primary sample space



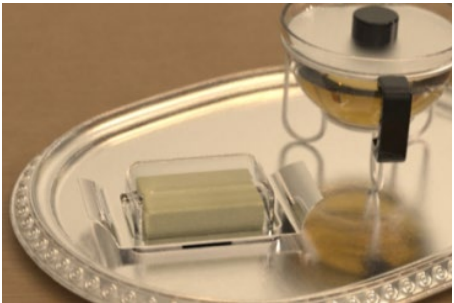
Changes **surface points** *indirectly*
by changing corresponding **numbers**

Existing mutation techniques

Path space



[Veach & Guibas 1997]



[Jacob & Marschner 2012]



[Kaplanyan et al. 2014]

+ local exploration
- global exploration

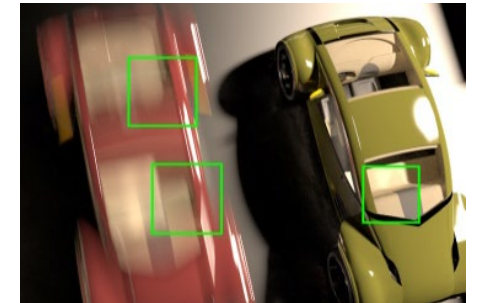
Primary sample space



[Kelemen et al. 2002]



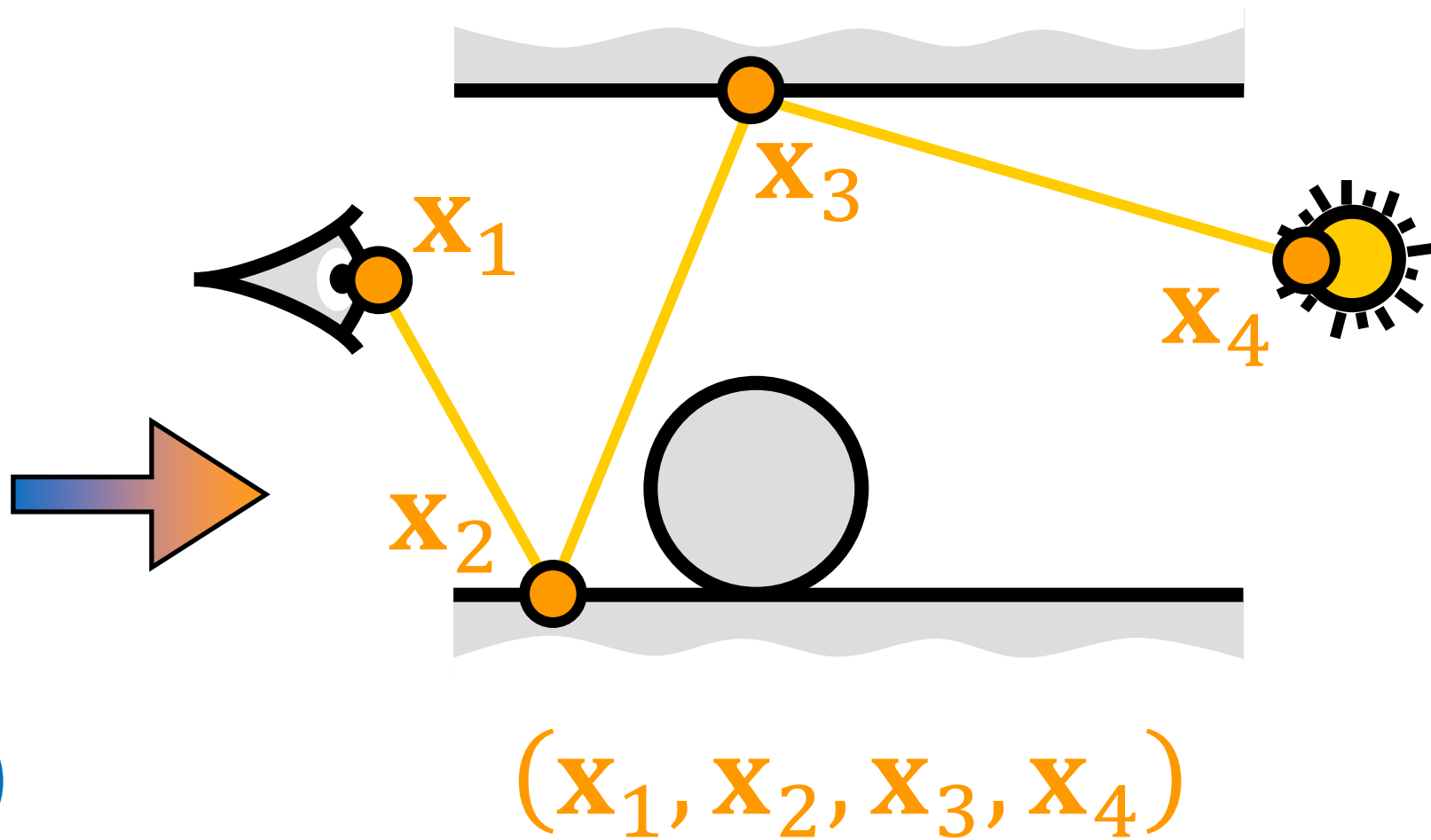
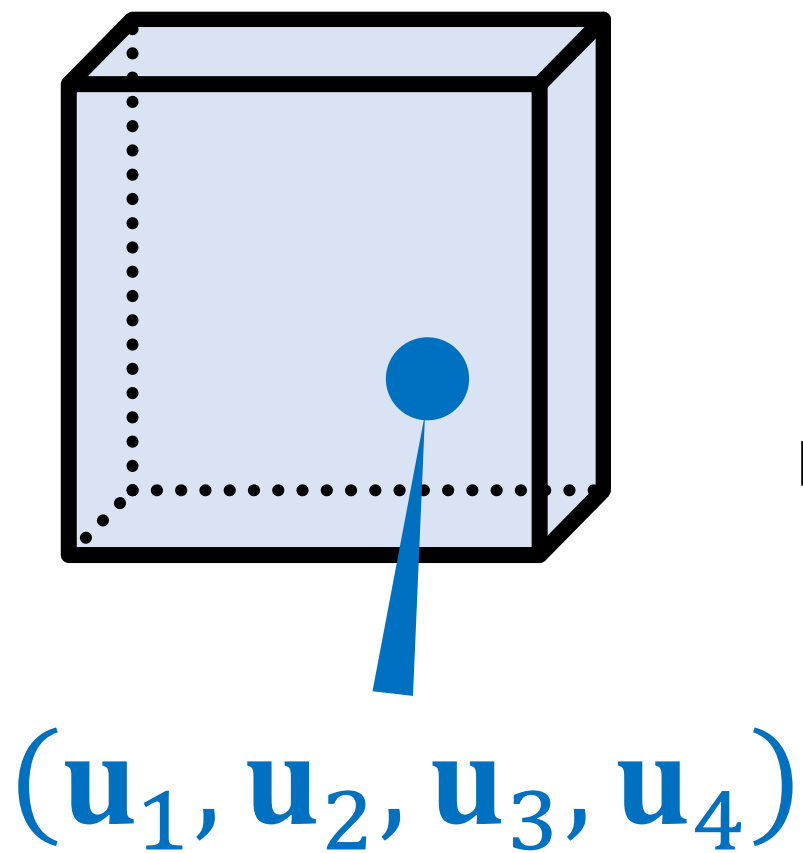
[Hachisuka et al. 2014]



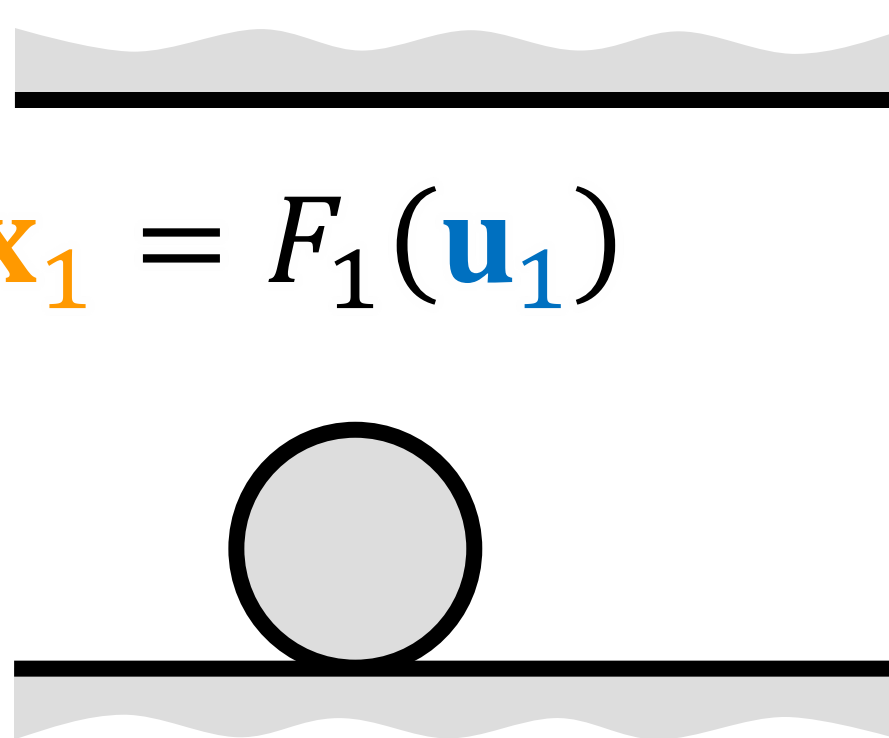
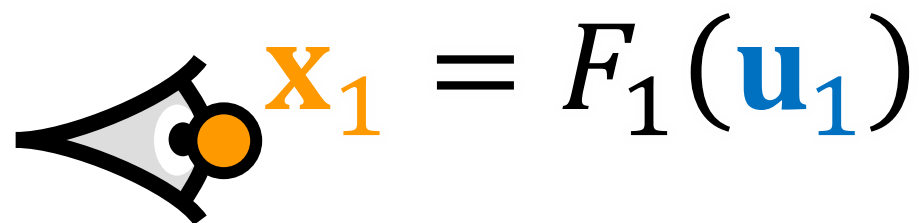
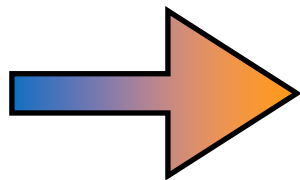
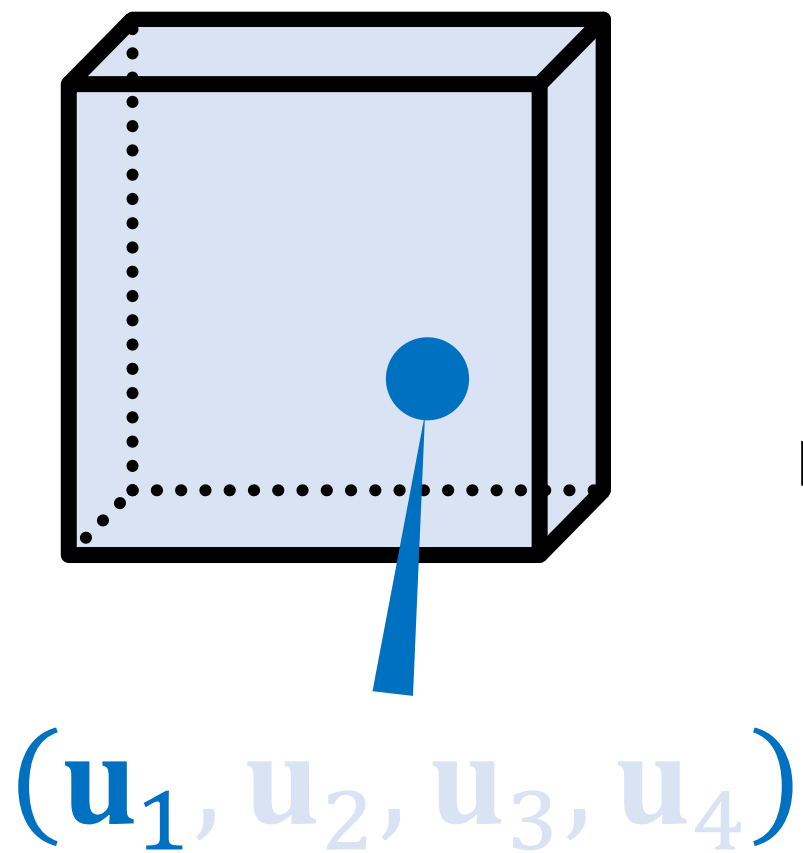
[Li et al. 2015]

- local exploration
+ global exploration

Path sampler

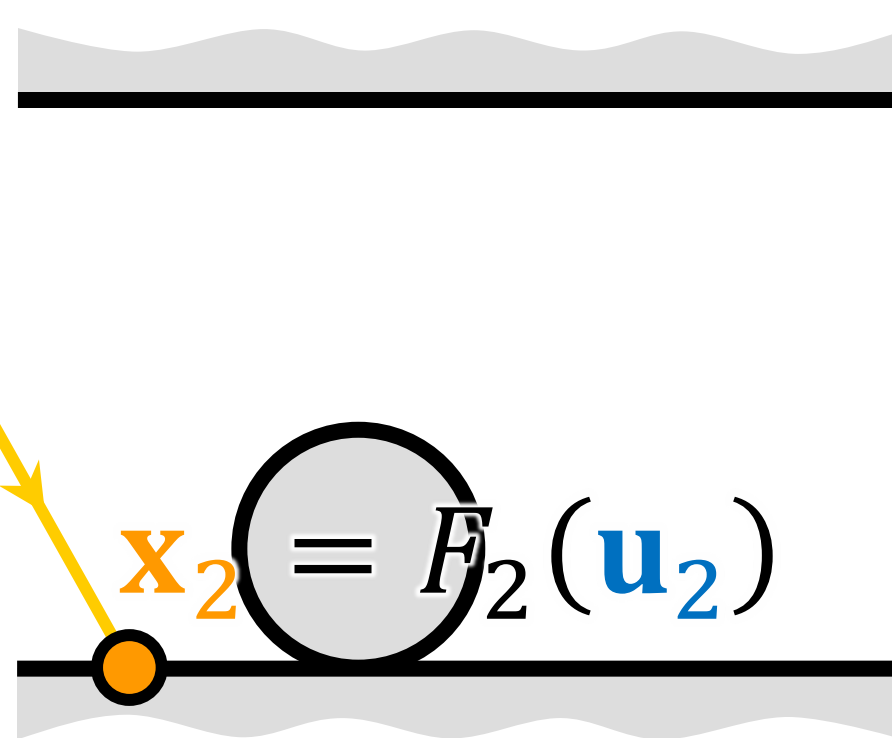
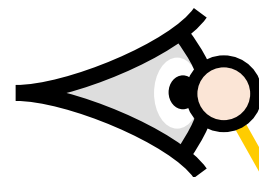
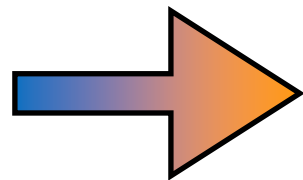
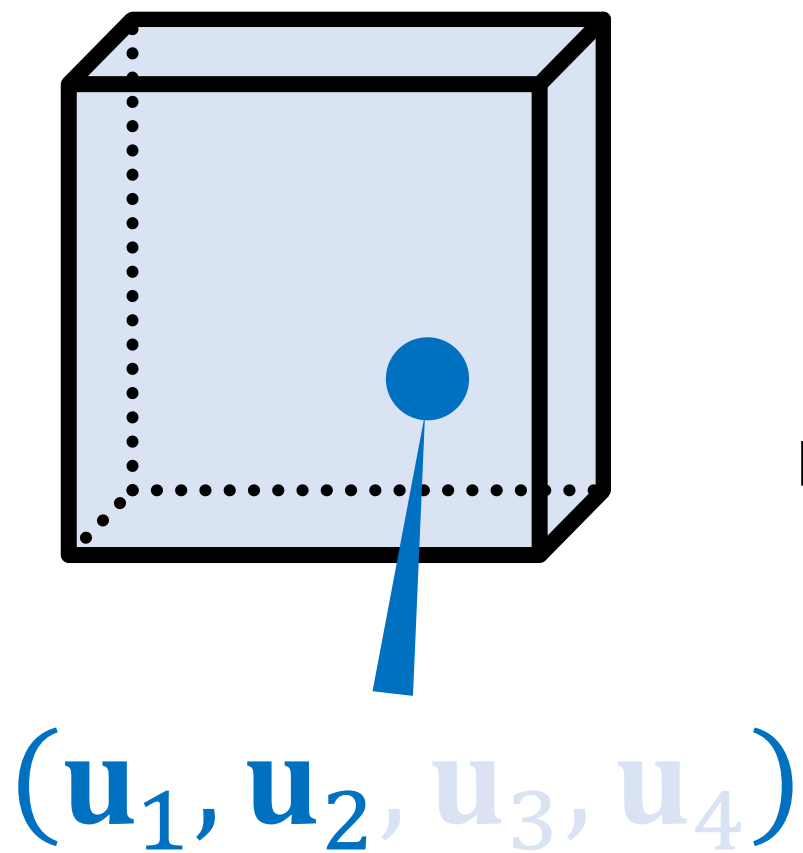


Path sampler



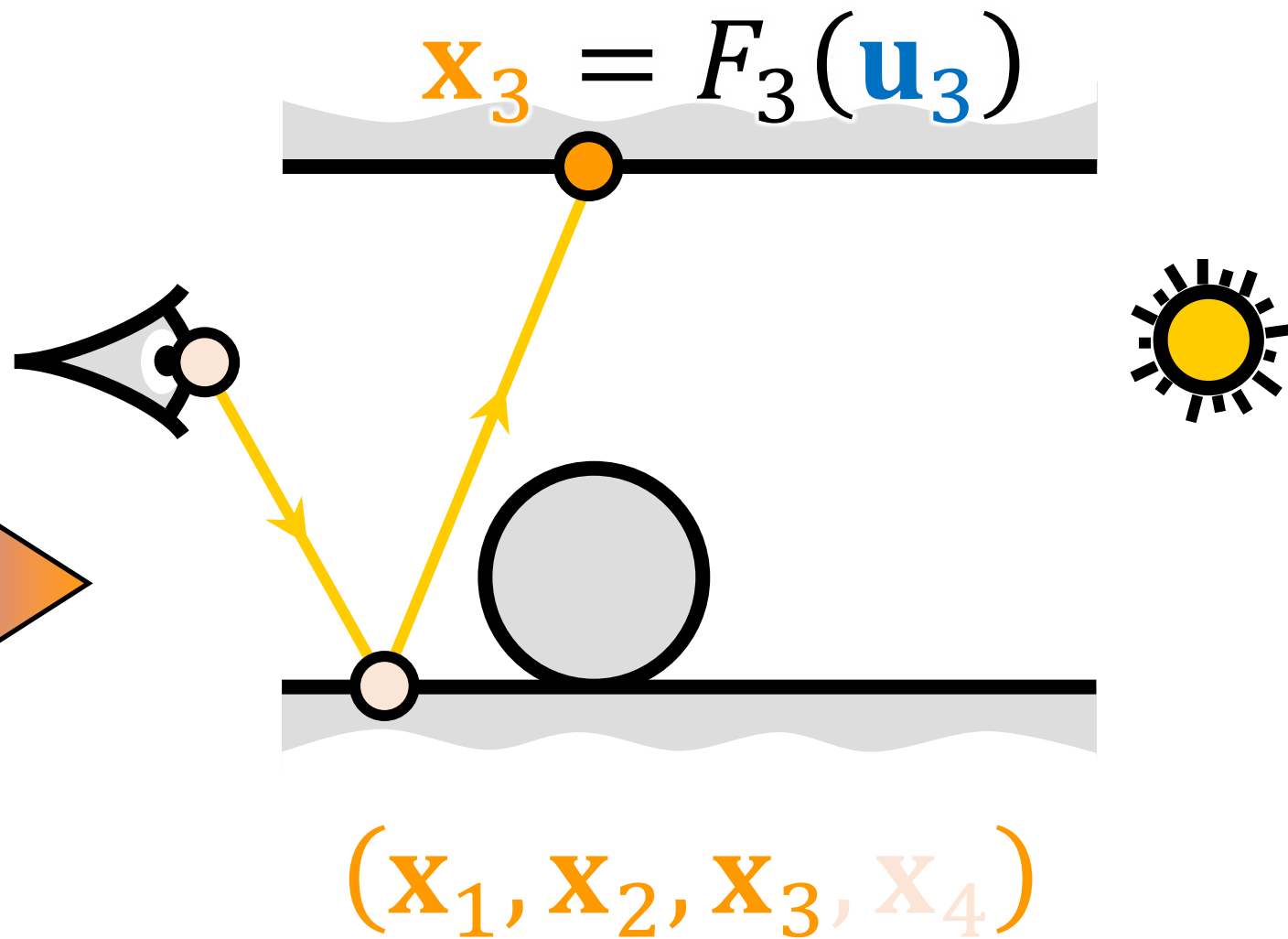
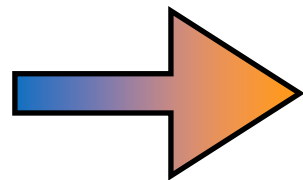
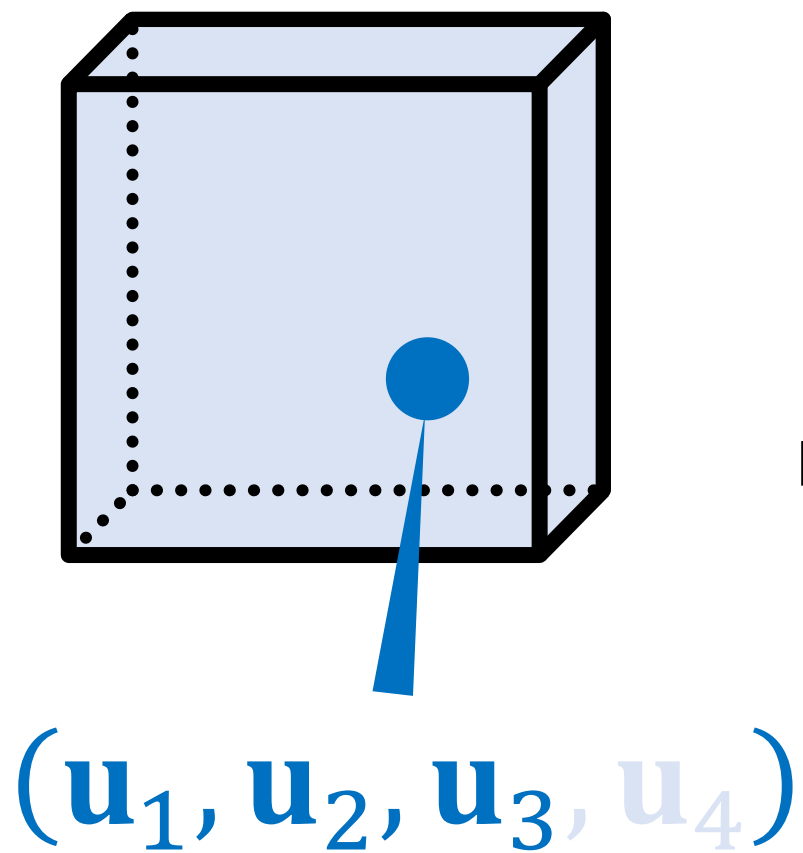
$(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4)$

Path sampler

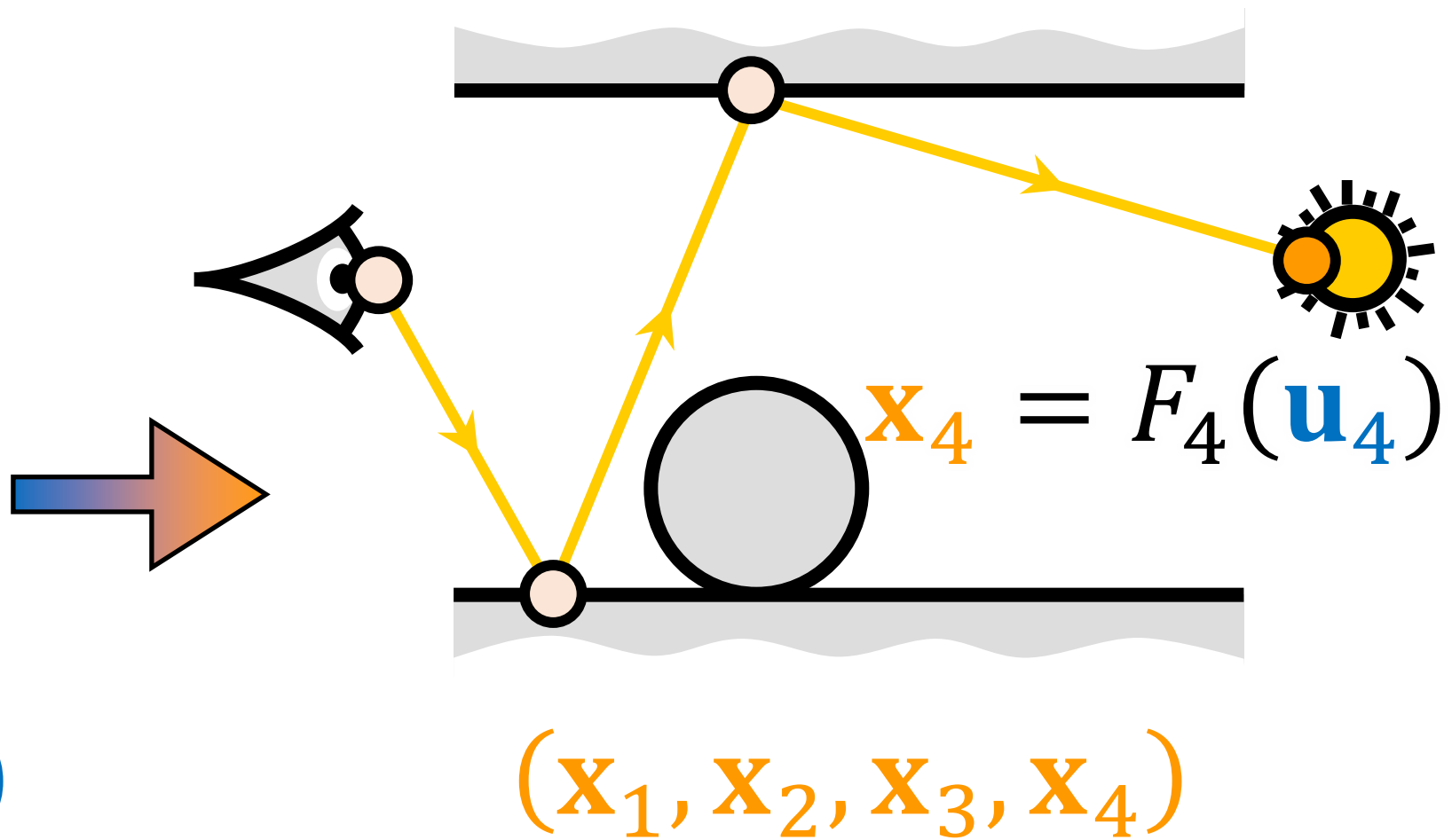
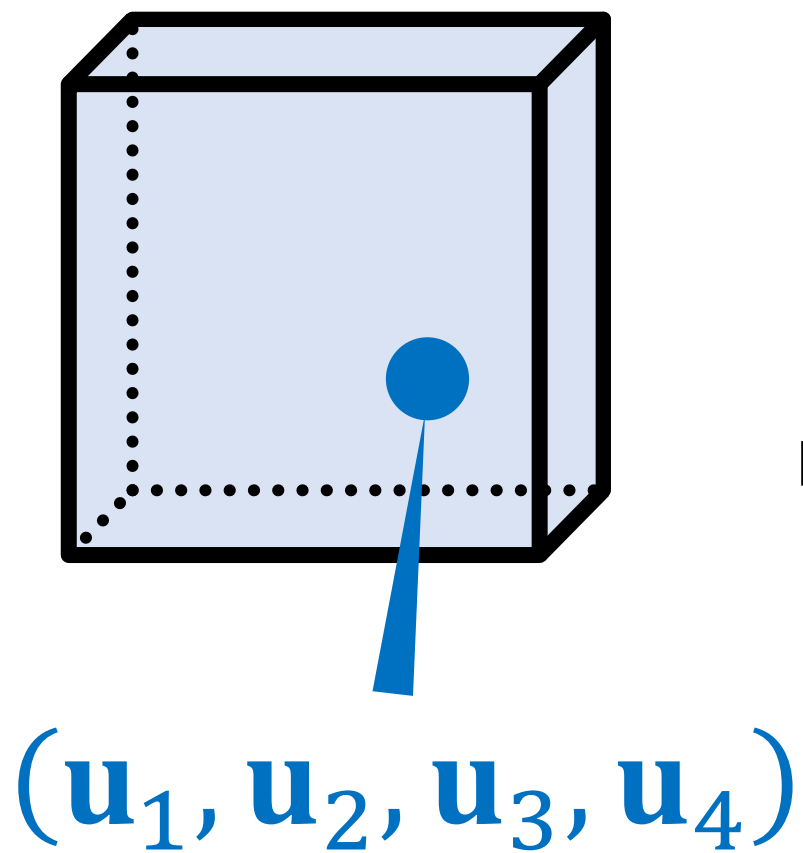


$(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4)$

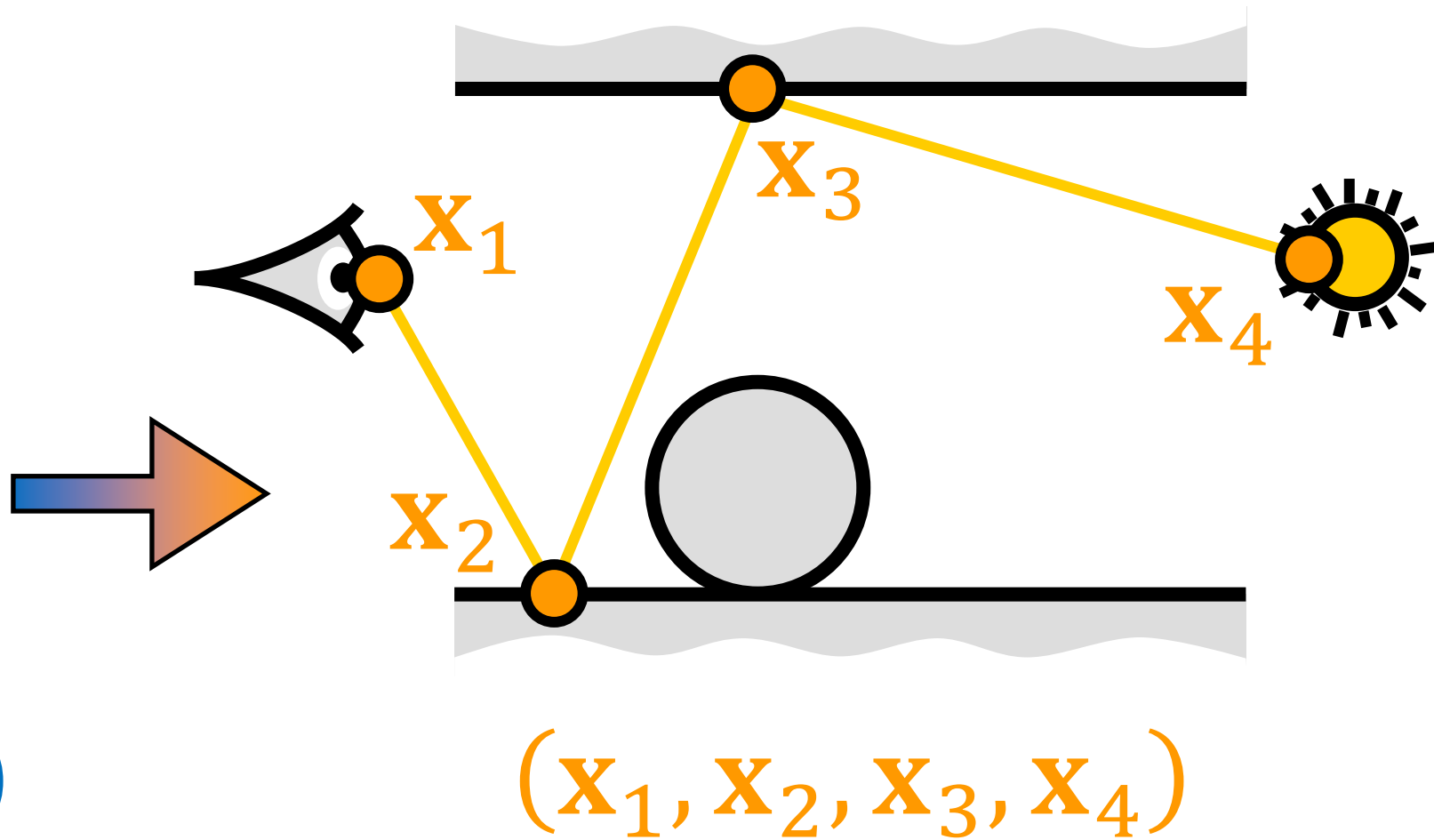
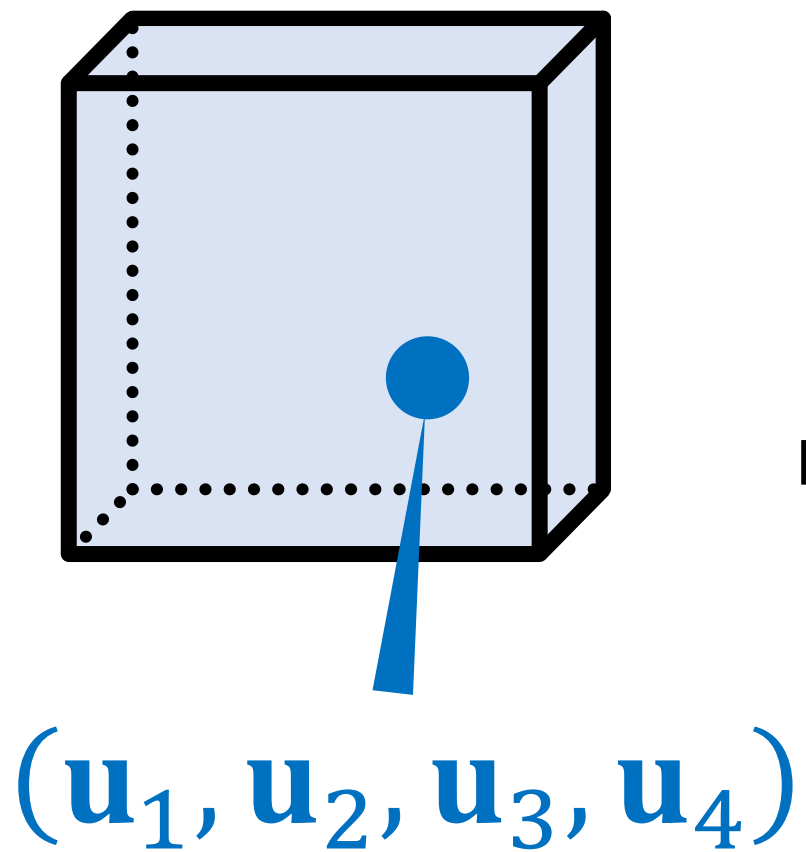
Path sampler



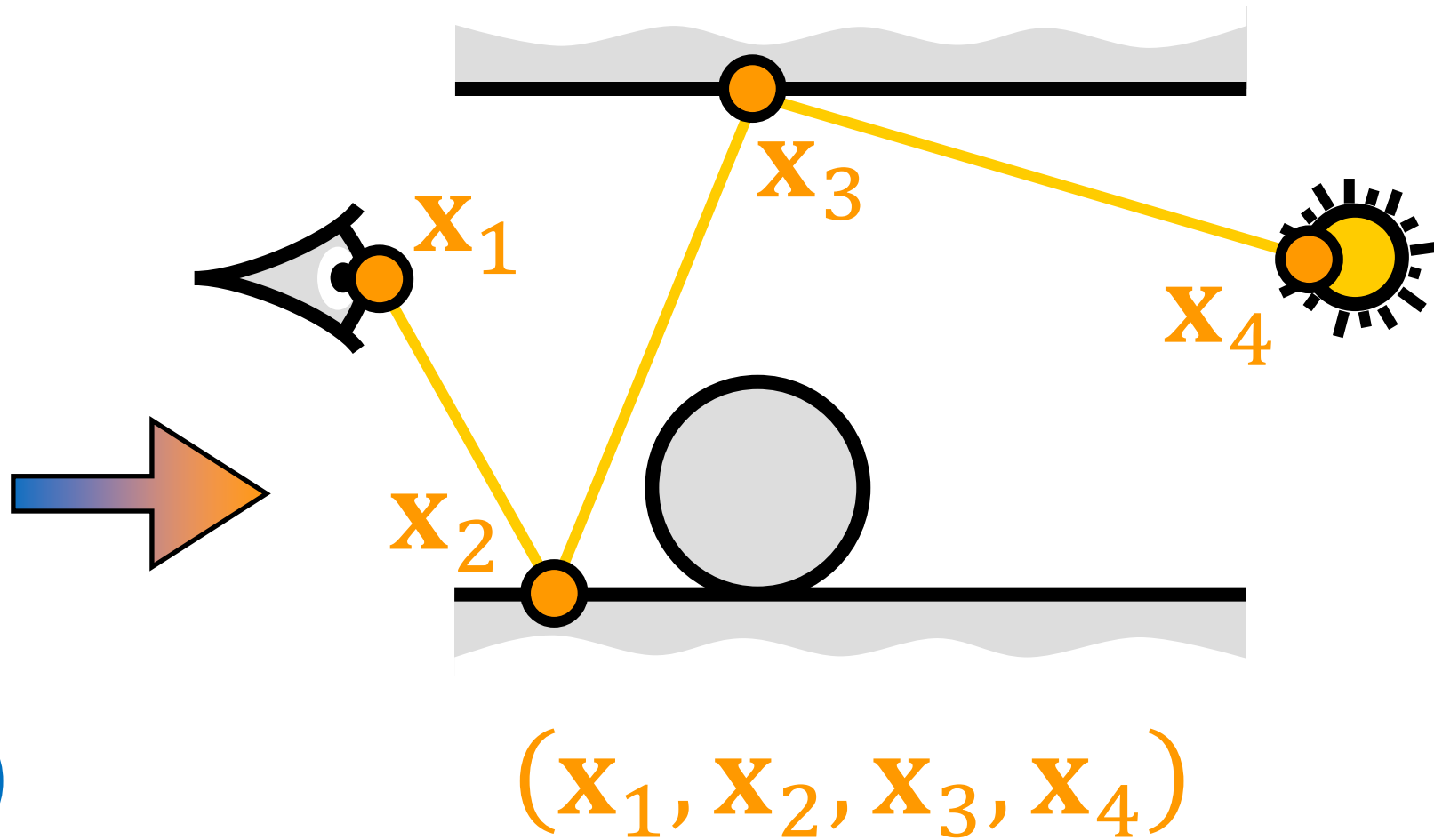
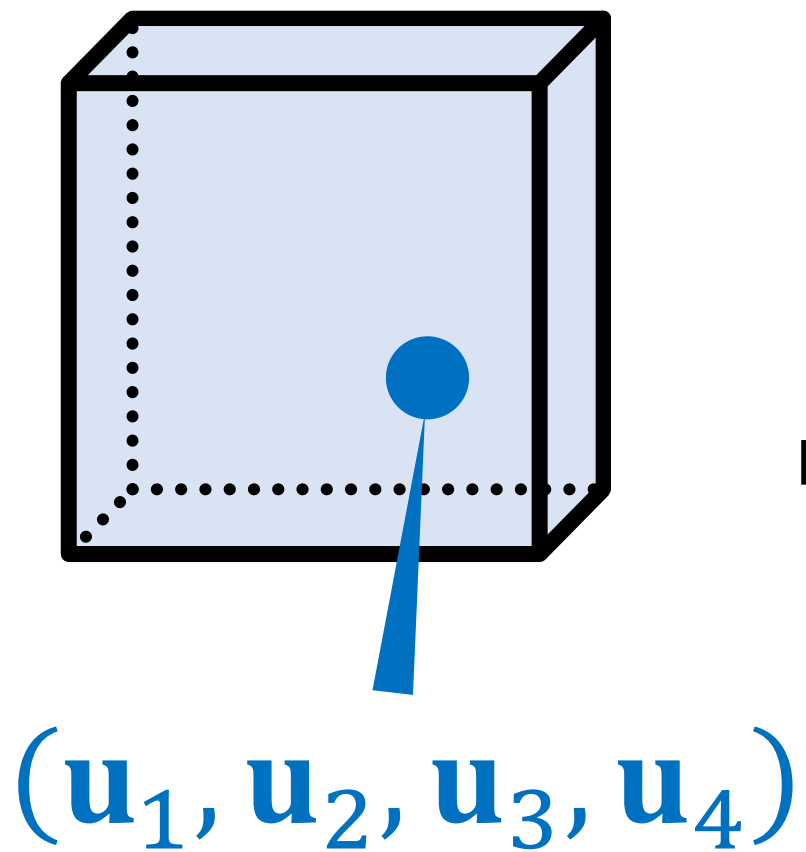
Path sampler



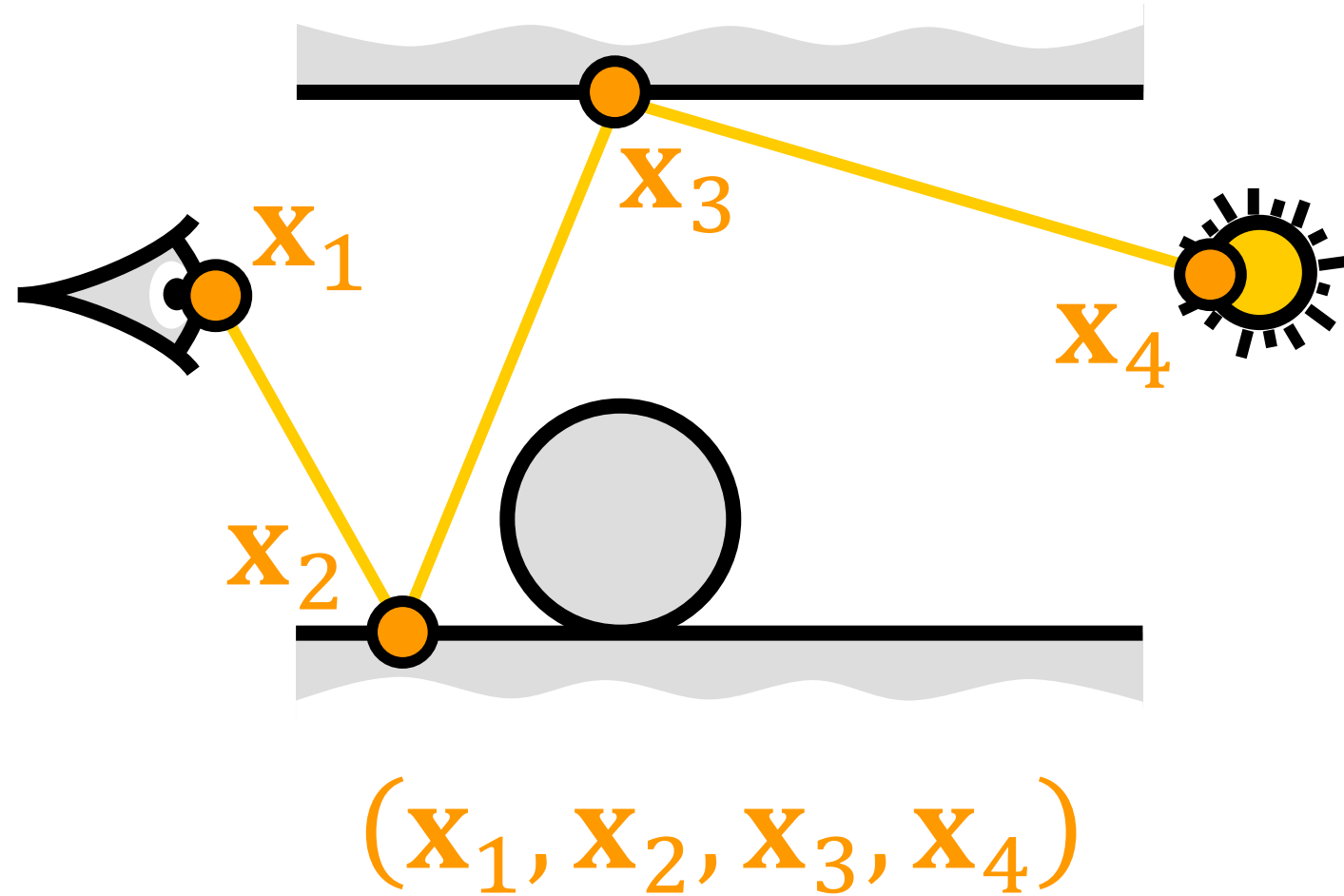
Path sampler



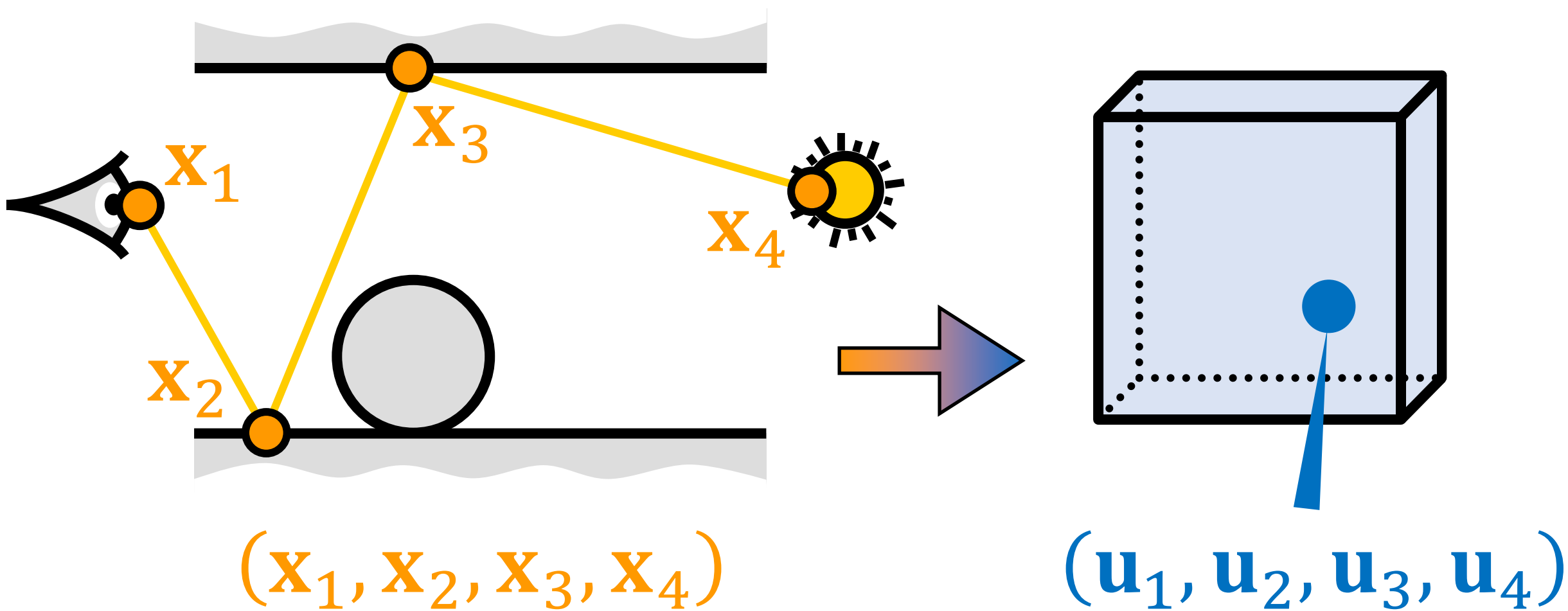
Path sampler



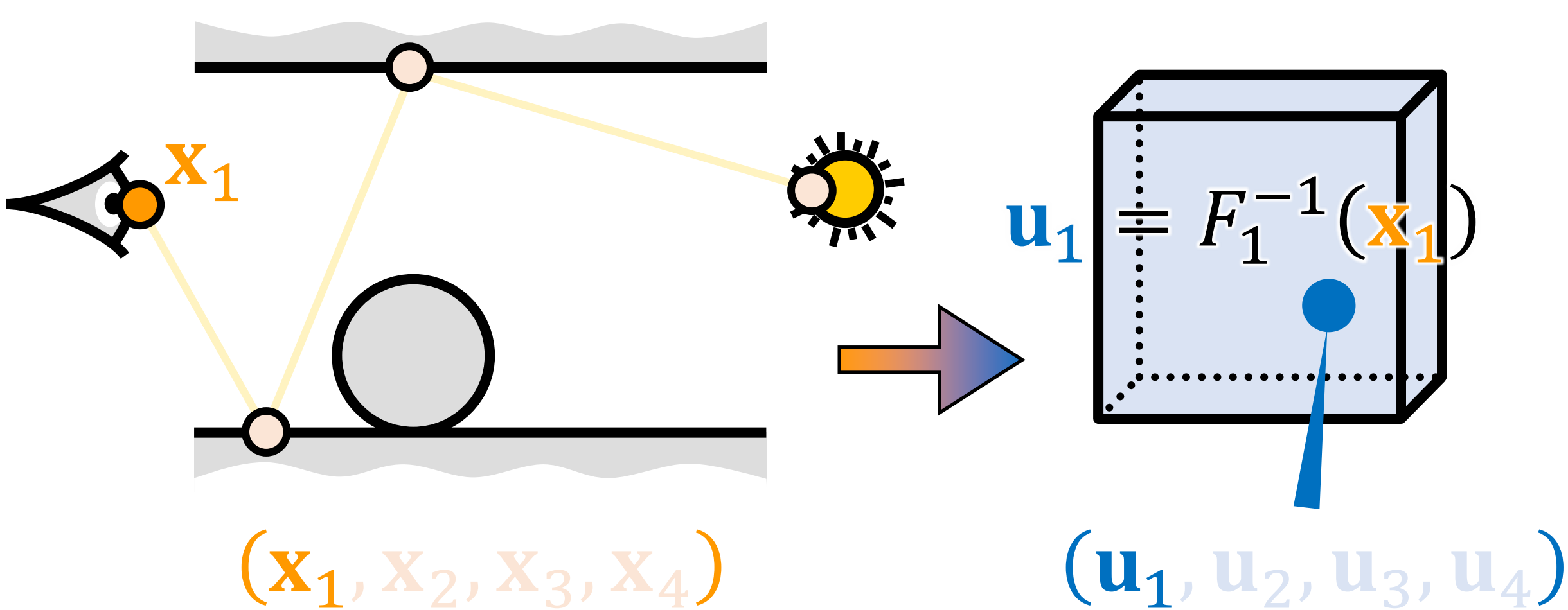
Inverse path sampler



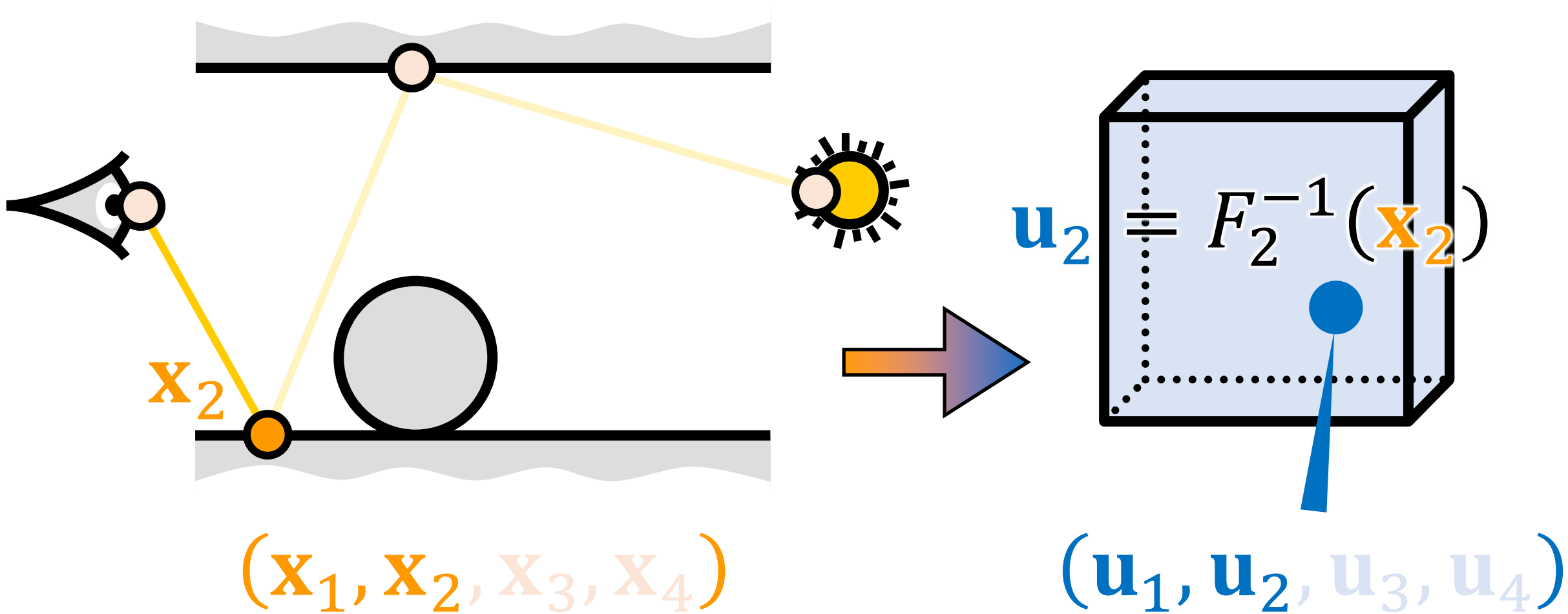
Inverse path sampler



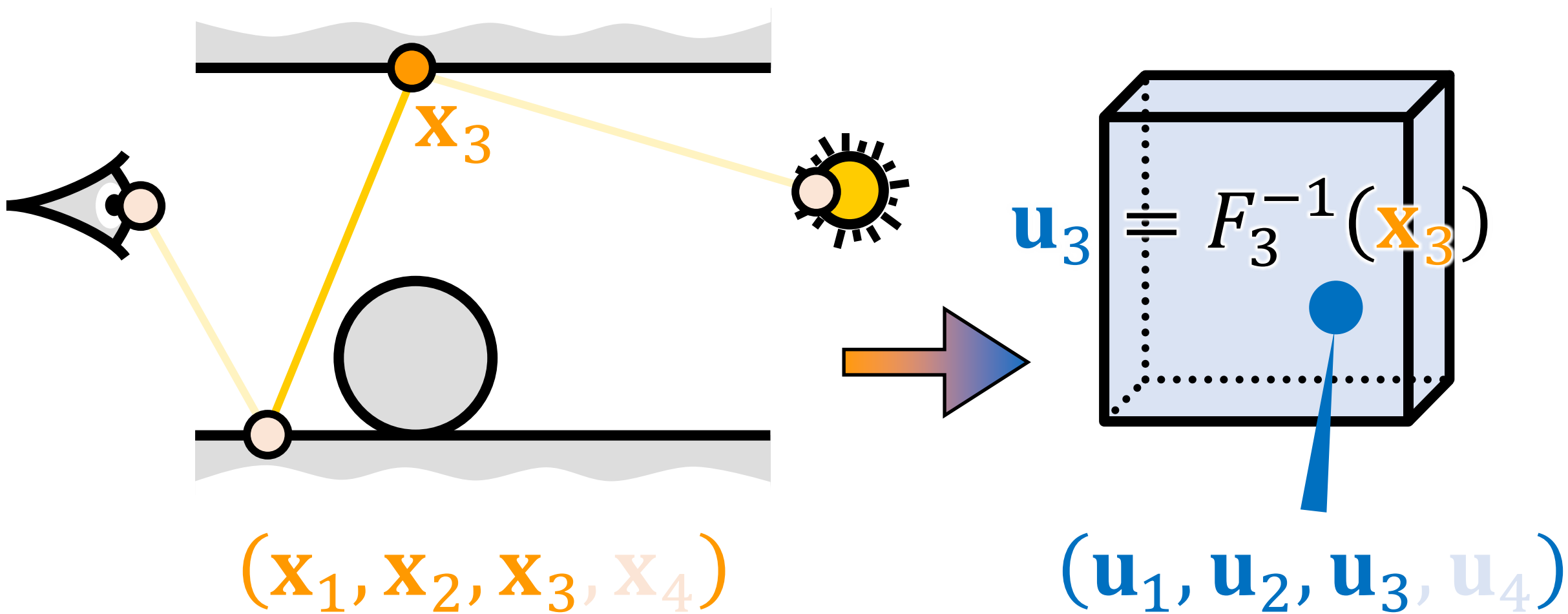
Inverse path sampler



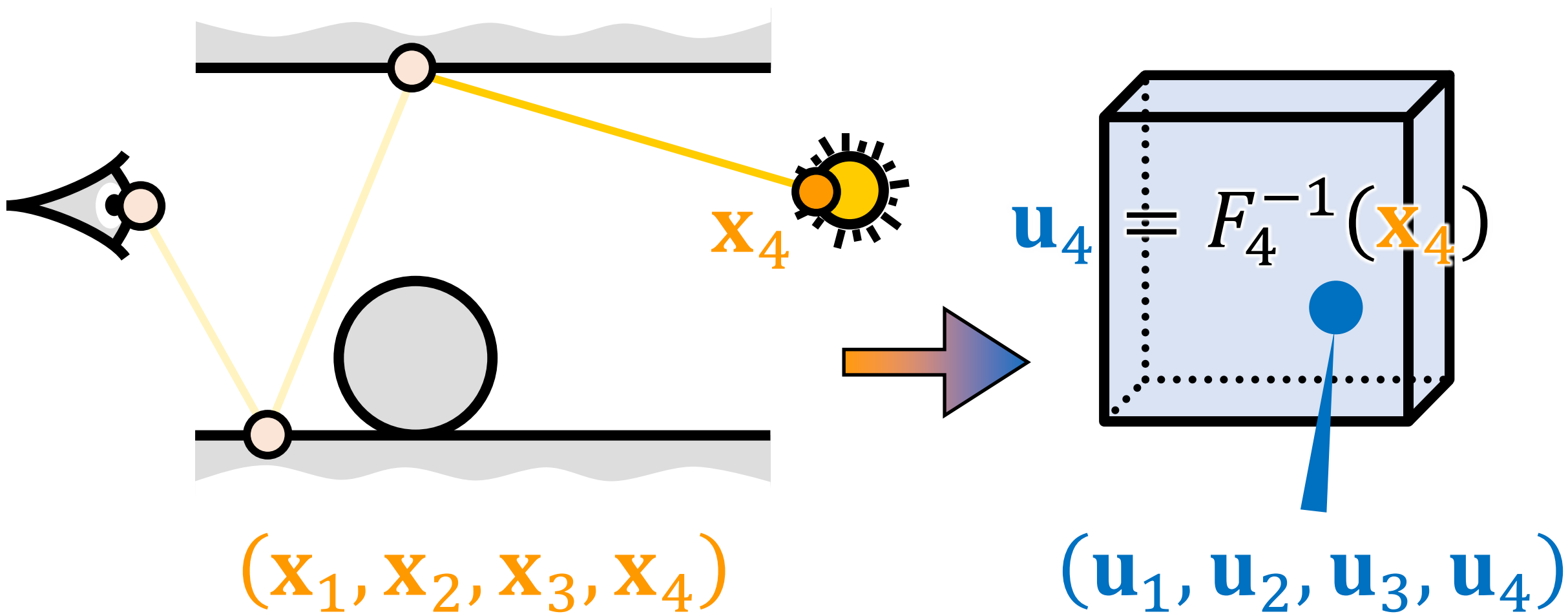
Inverse path sampler



Inverse path sampler



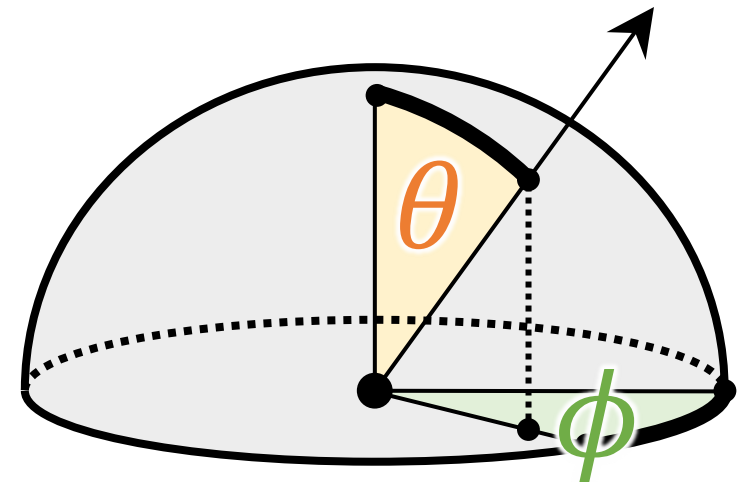
Inverse path sampler



Inverse path sampler

- F is the inverse cumulative distribution function (CDF)
- **Obtaining F^{-1} (CDF) is straightforward** because it is needed in the derivation of the inverse CDF
- Example: GGX distribution

$$\theta = \arccos\left(\frac{\tan^{-1}\left(\frac{\alpha \sqrt{u_1}}{\sqrt{1+\tan^2 u_1}}\right)}{\alpha}\right), \phi = 2\pi u_2$$

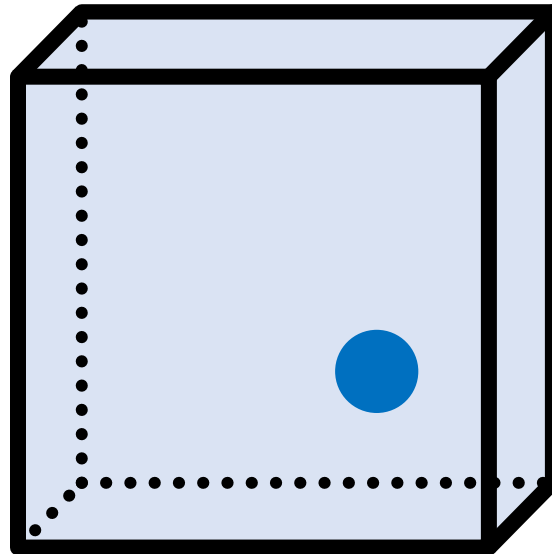


Handling non-invertibility

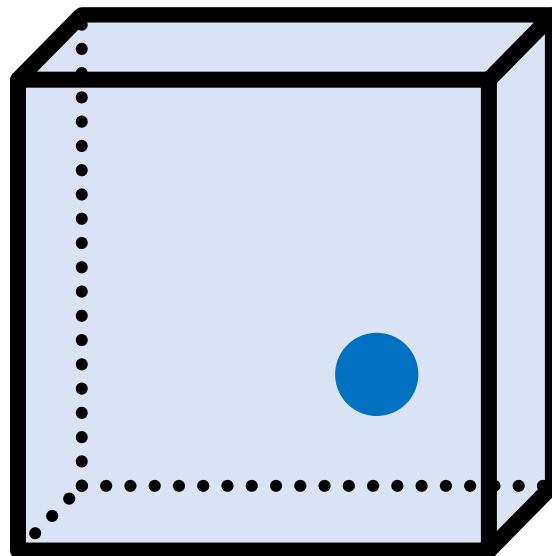
- Handling the case of F^{-1} does not exist
 - e.g., perfect reflectors, layered material
- Use of a mapping in ***lower dimensional subspace*** to define invertible mapping

Fusing mutations

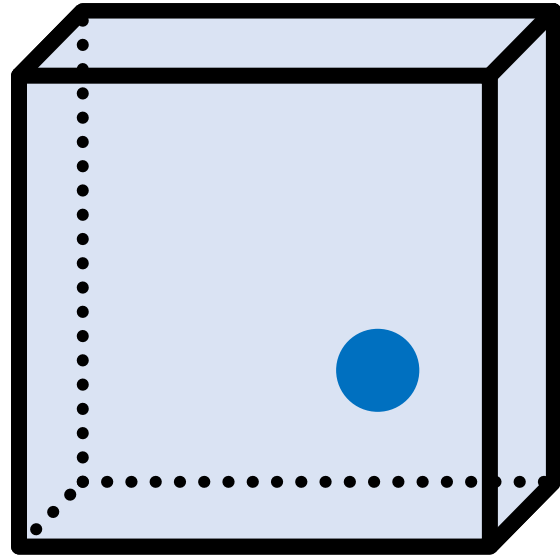
- With an *inverse path sampler*, we can fuse mutations in **path space** and **primary sample space**
- Our *state* is defined in **primary sample space**



Fusing mutations



Primary sample space mutations



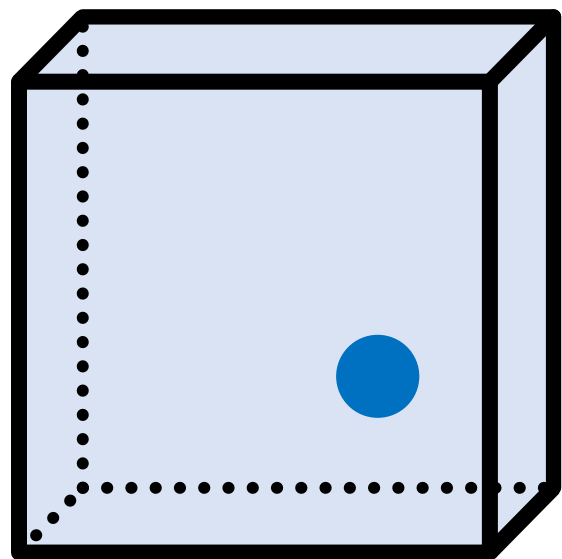
Current state

Primary sample space
mutation



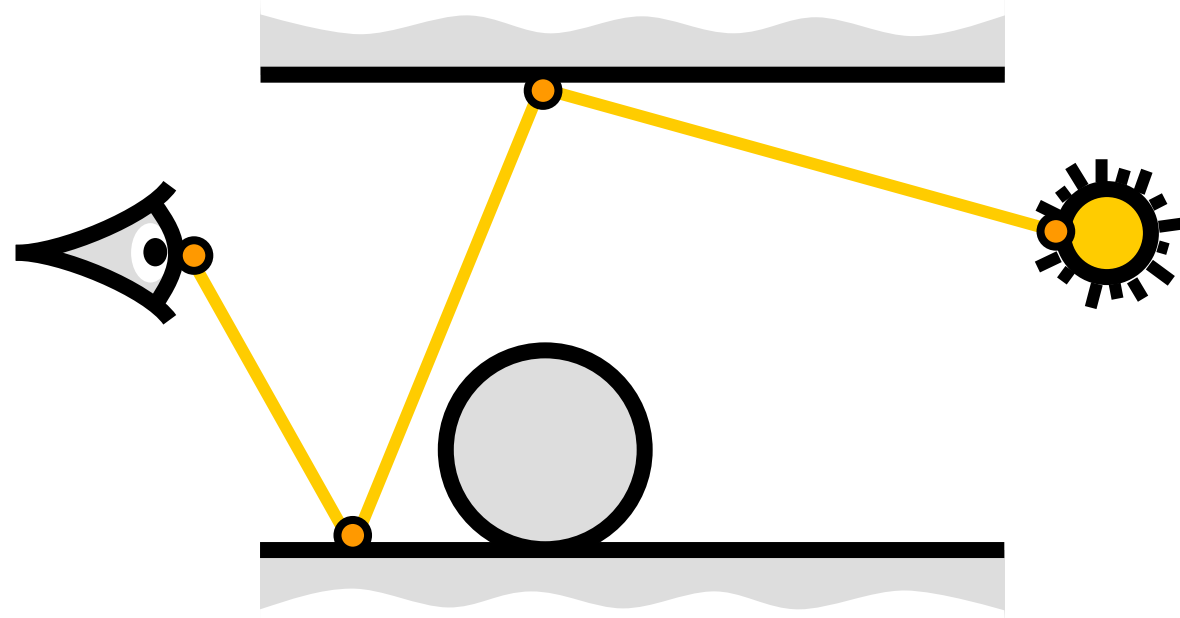
Next state

Path space mutations

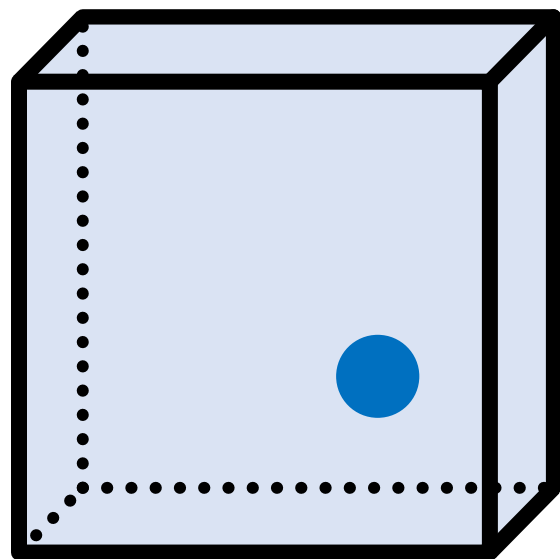


Current state

Map to
path space

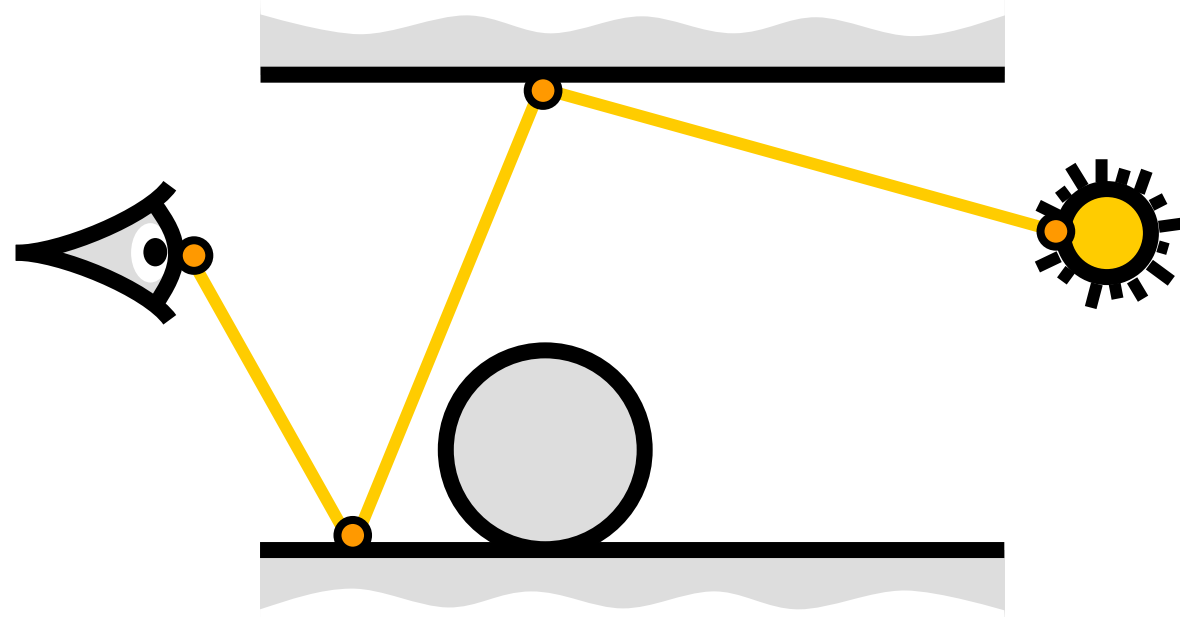


Path space mutations

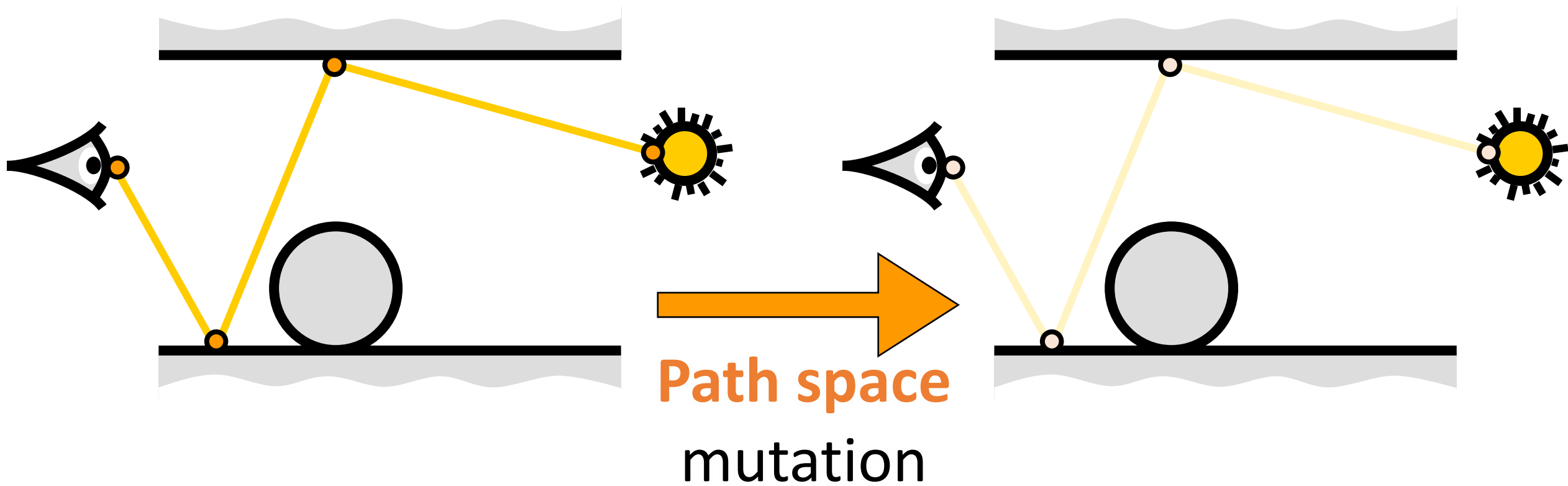


Current state

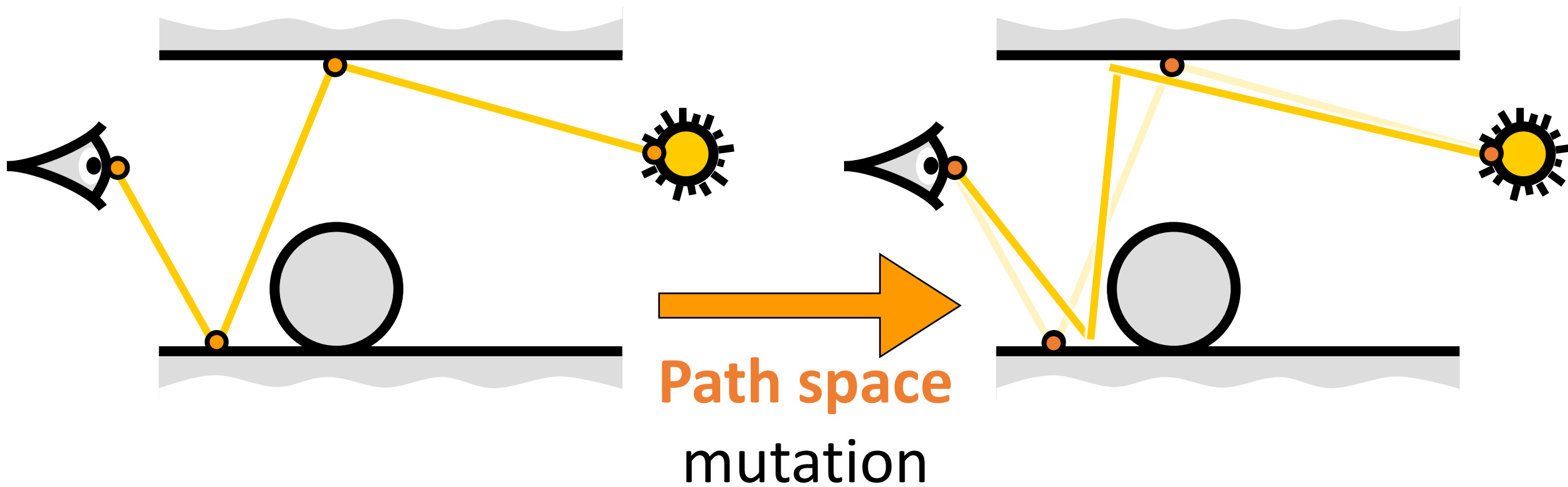
Map to
path space



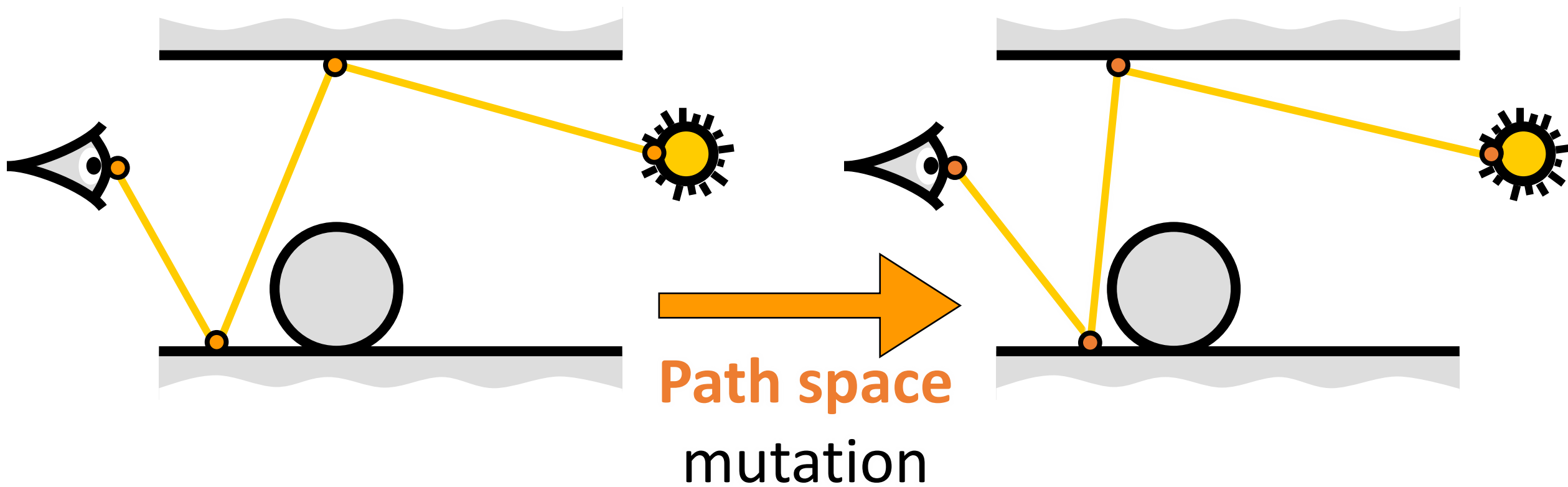
Path space mutations



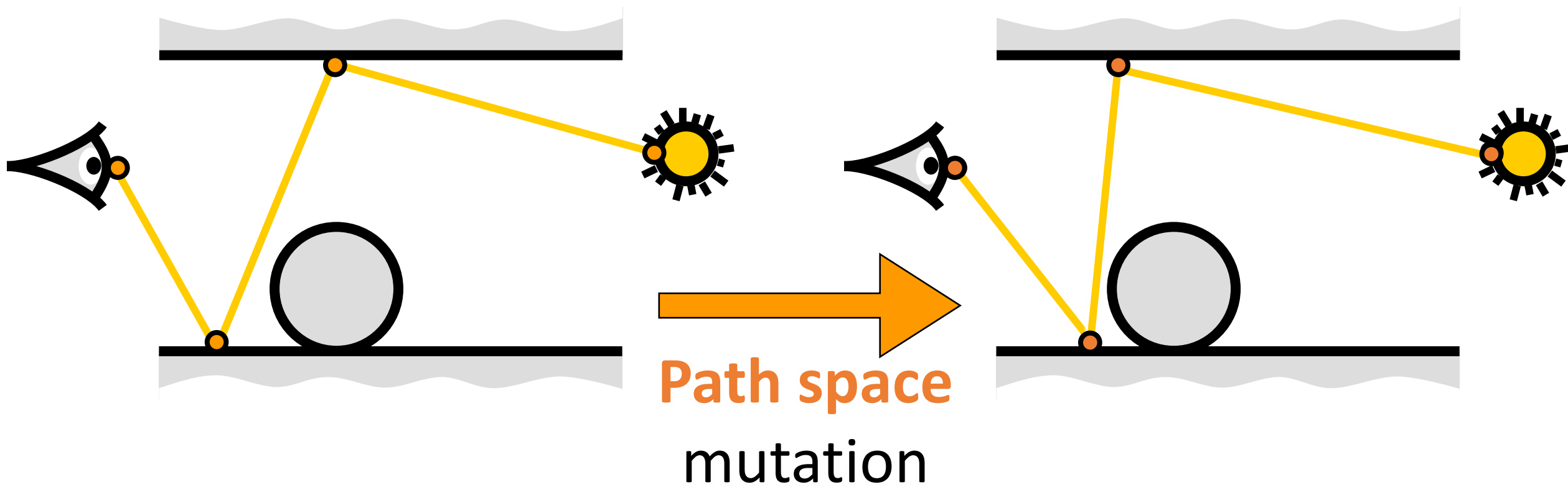
Path space mutations



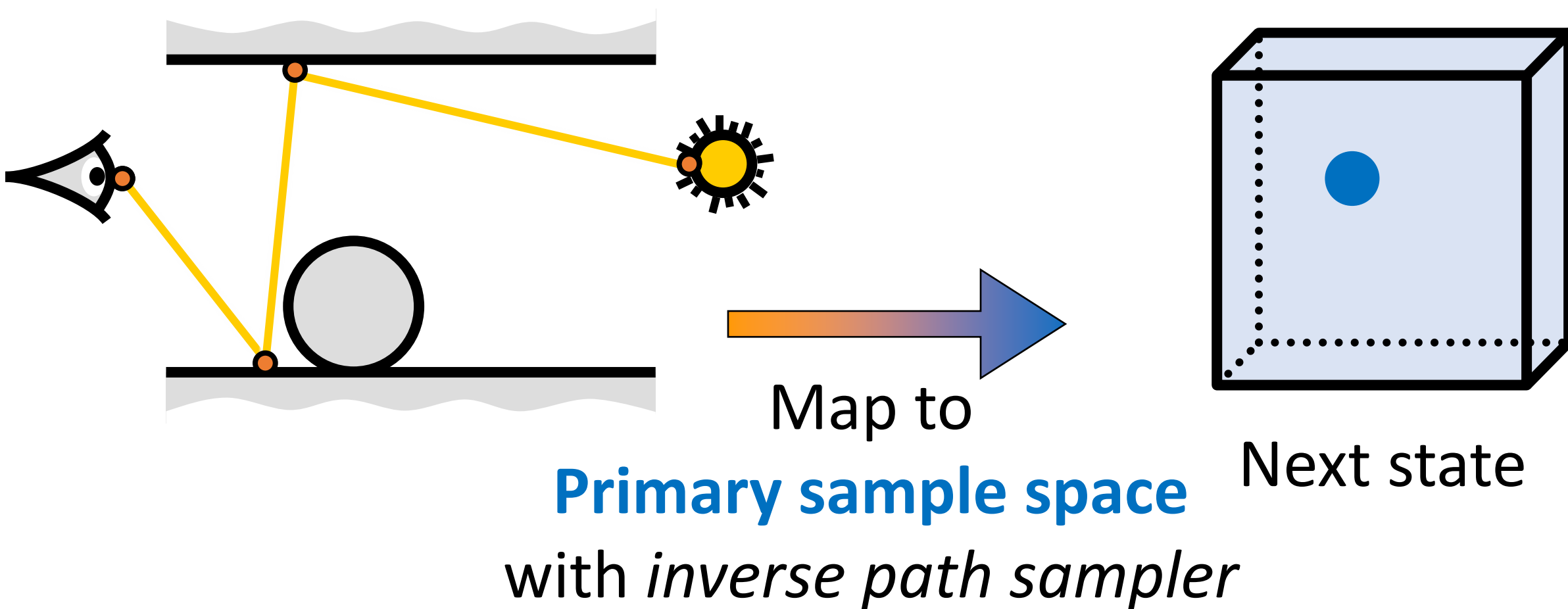
Path space mutations

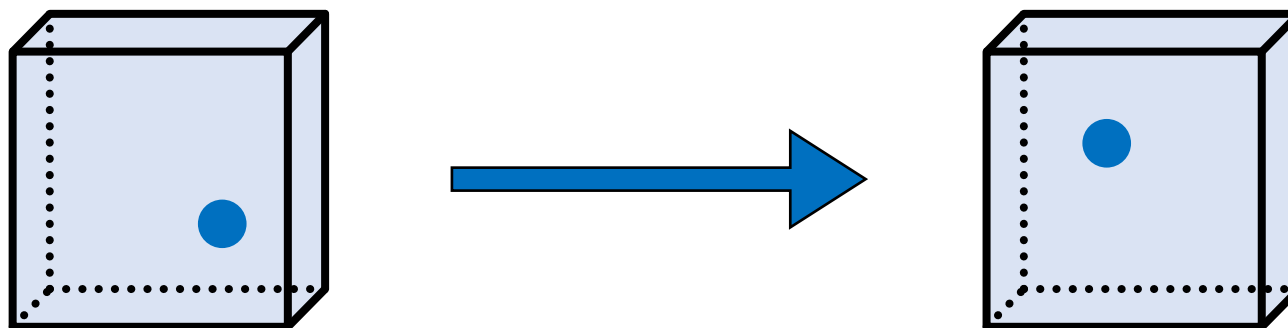


Path space mutations

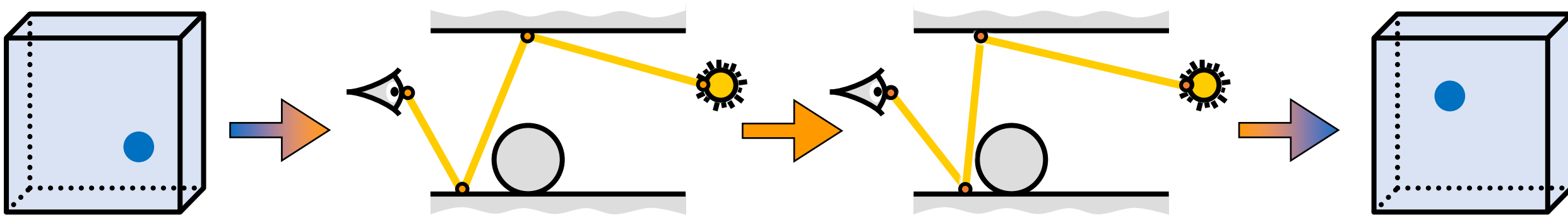


Path space mutations

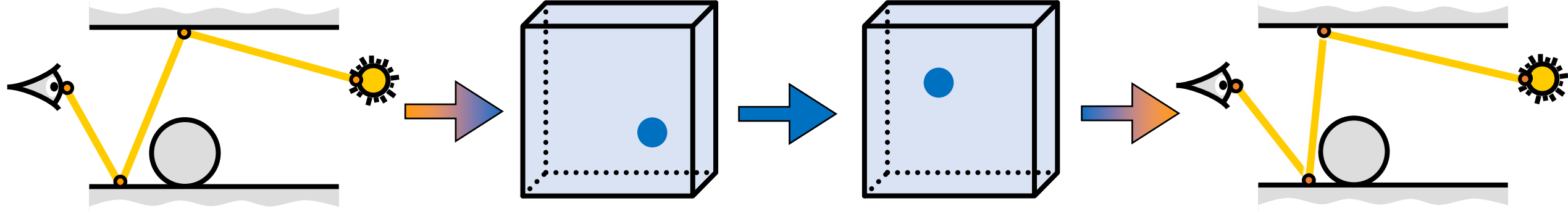




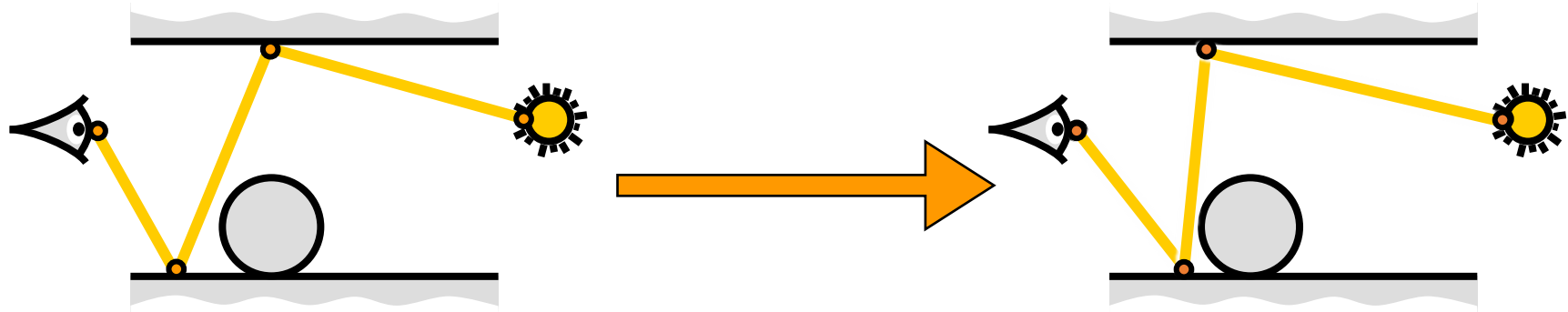
Primary sample space mutation



Path space mutation



Primary sample space mutation



Path space mutation

Reference

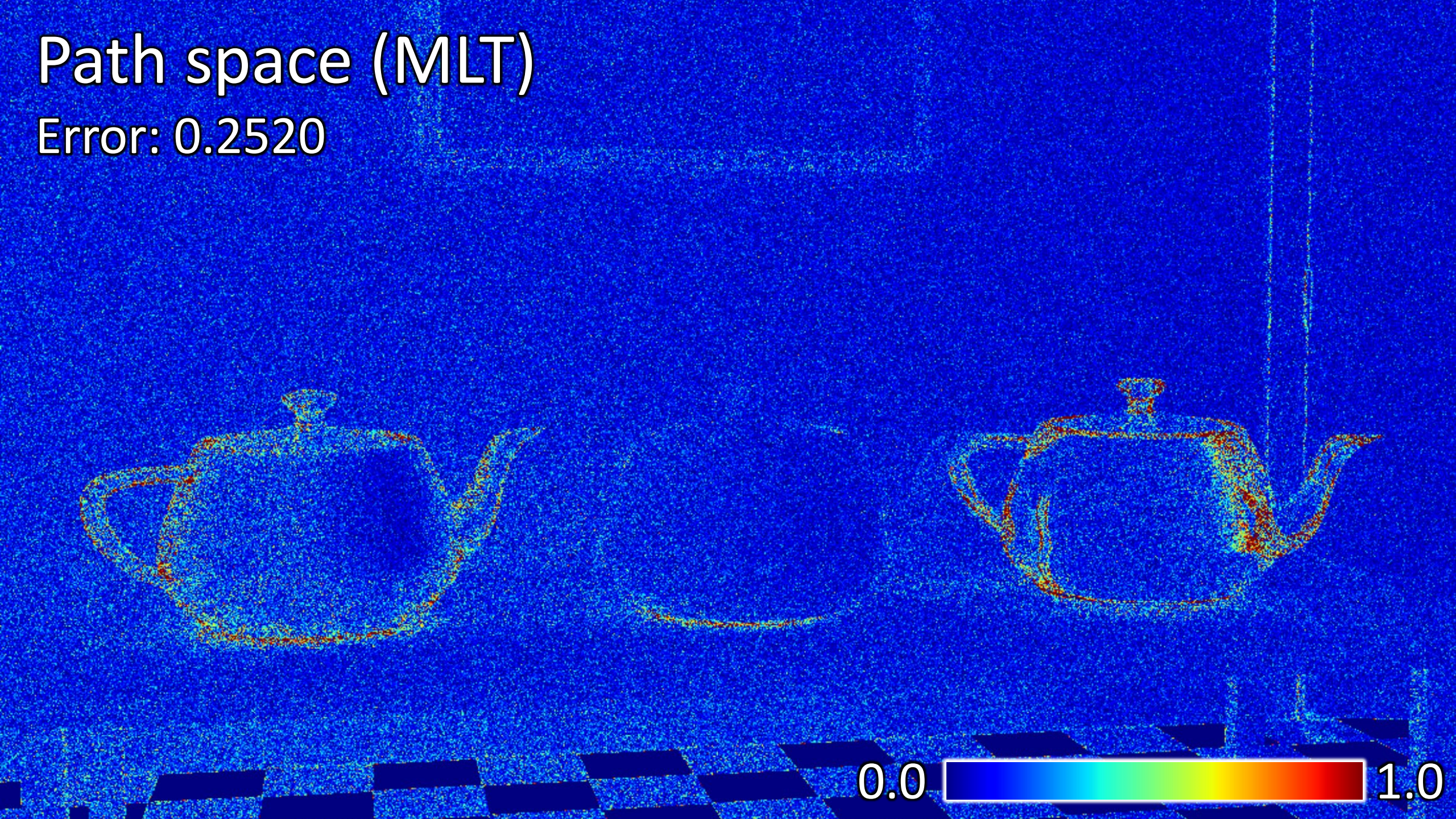


Path space (MLT)

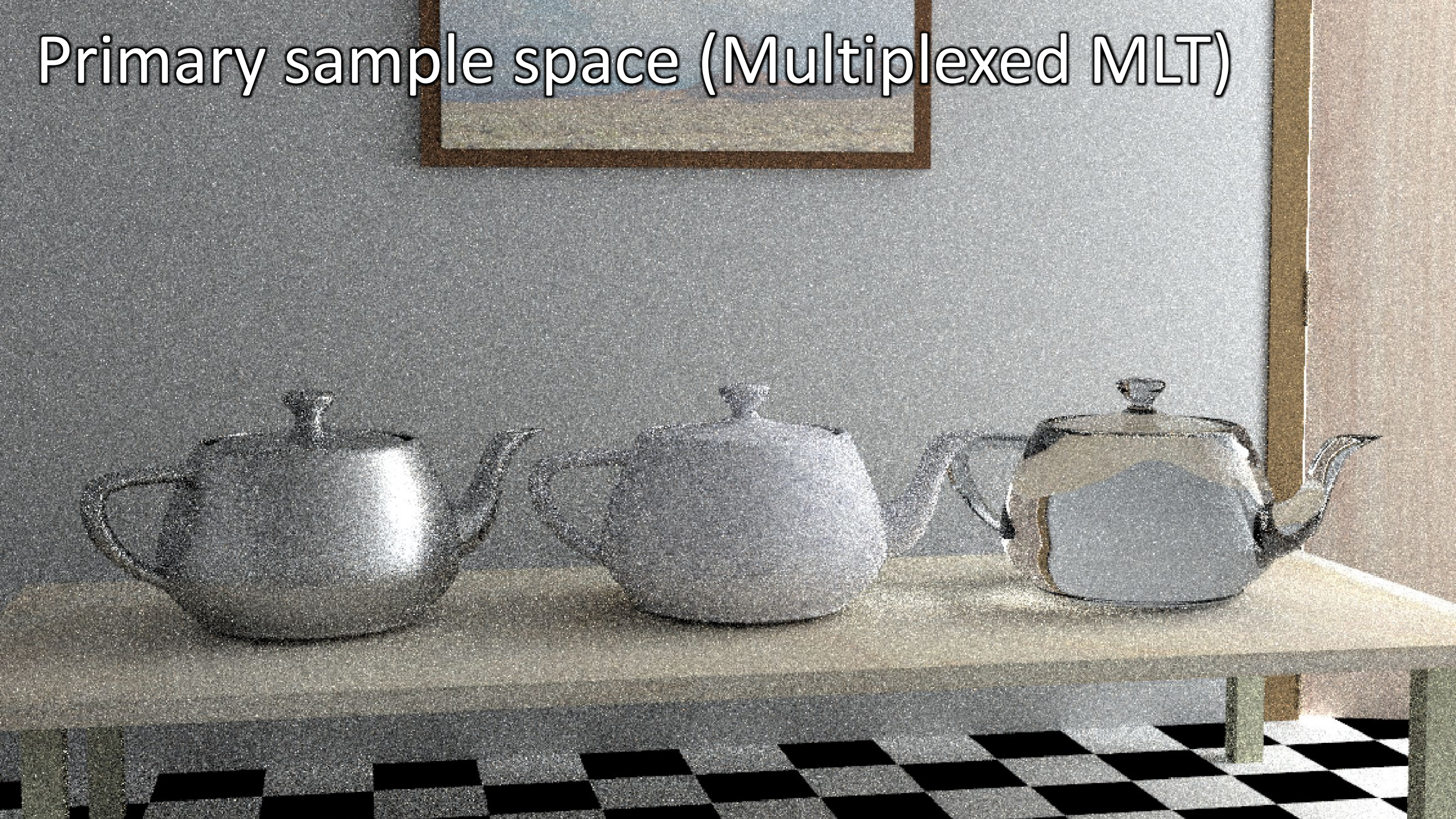


Path space (MLT)

Error: 0.2520

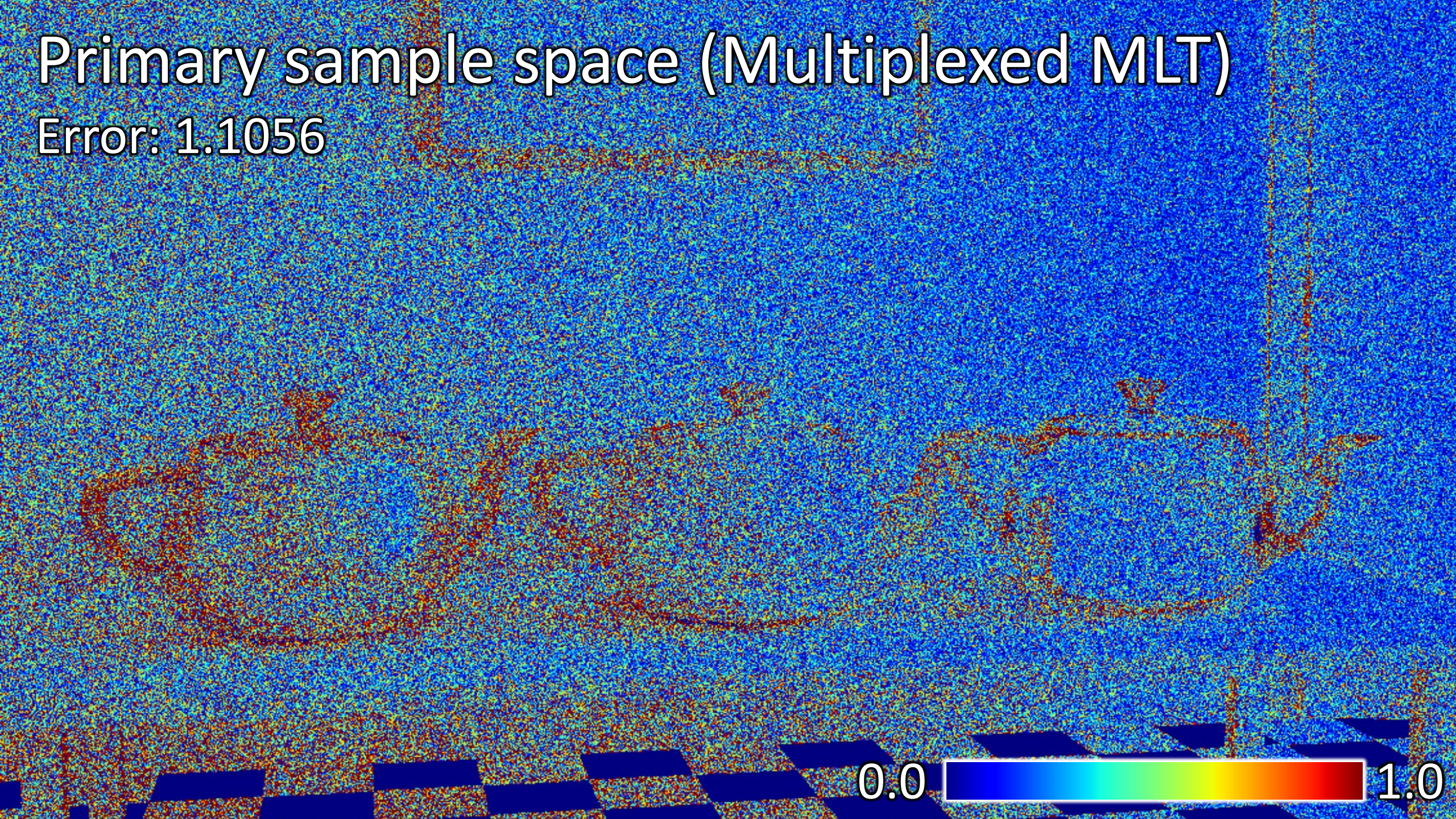


Primary sample space (Multiplexed MLT)



Primary sample space (Multiplexed MLT)

Error: 1.1056

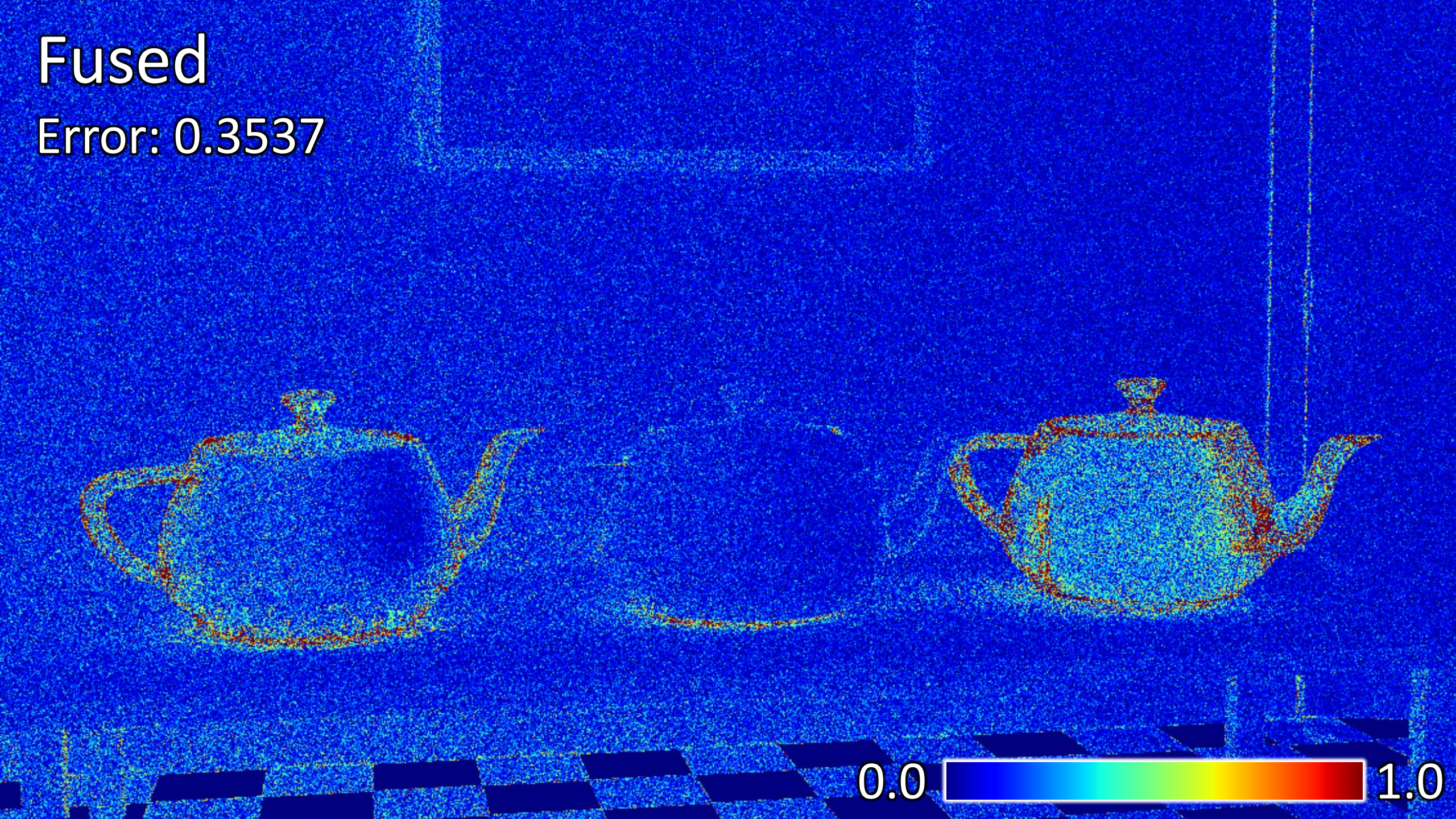


Fused



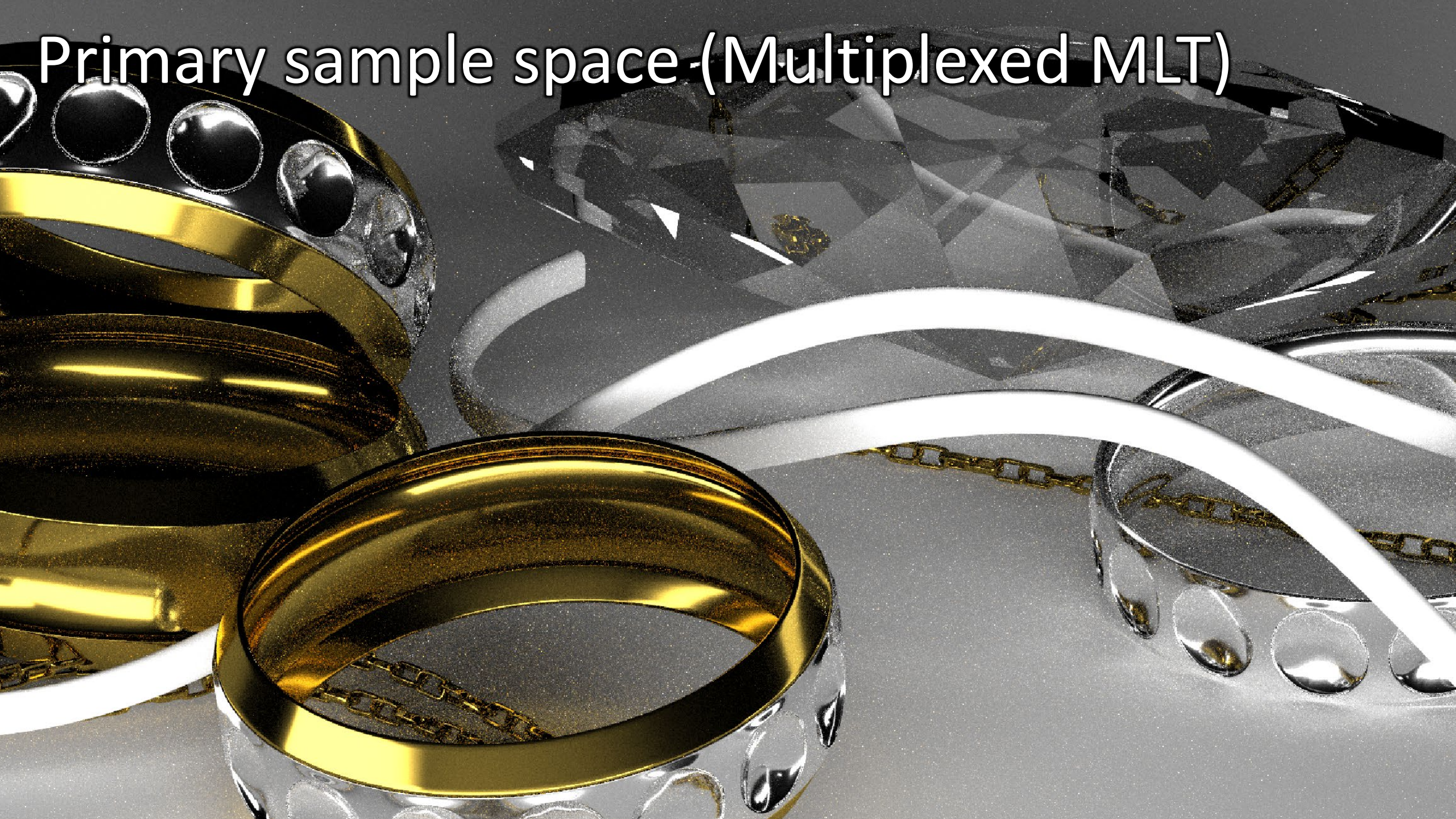
Fused

Error: 0.3537





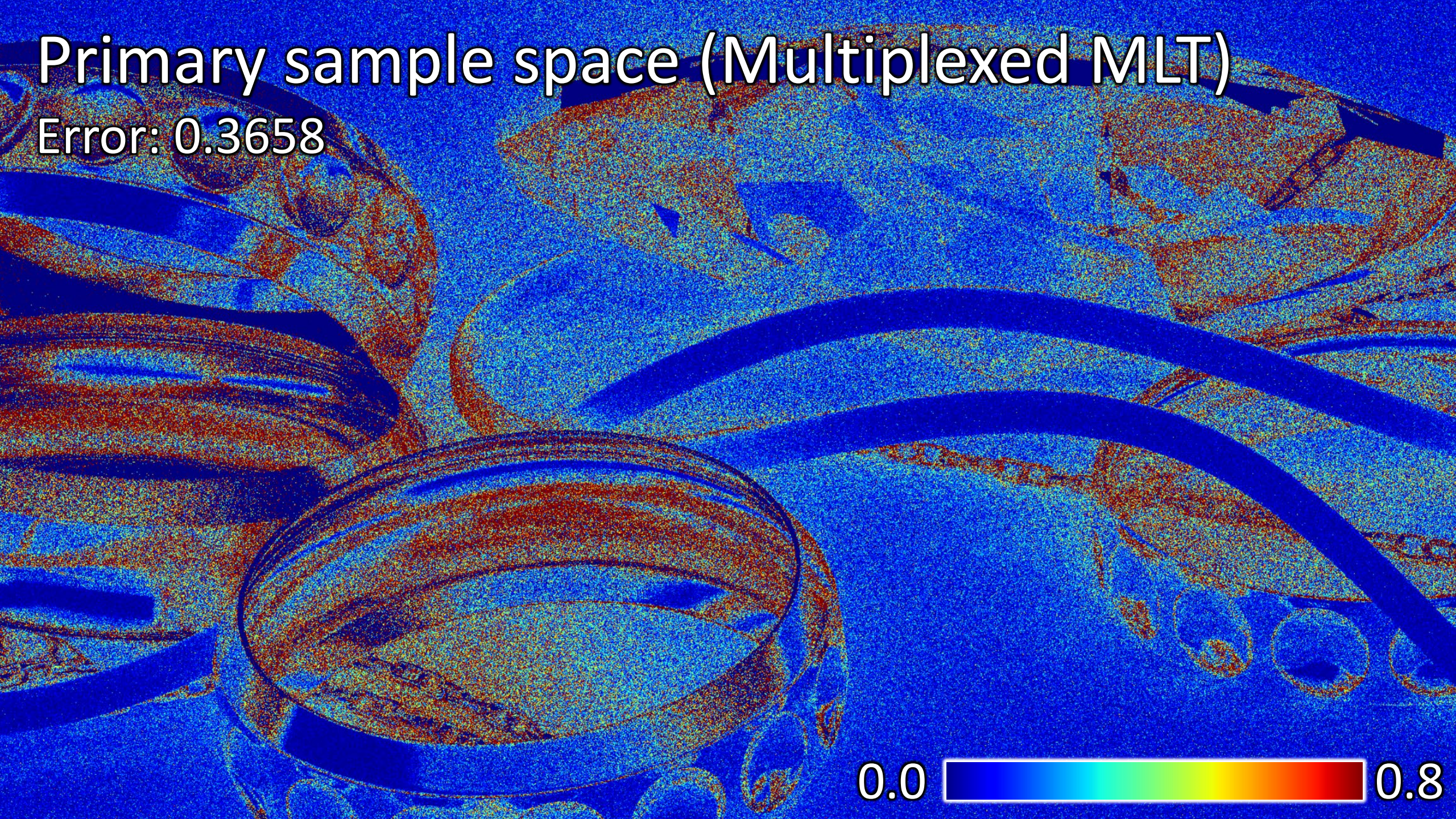
Reference

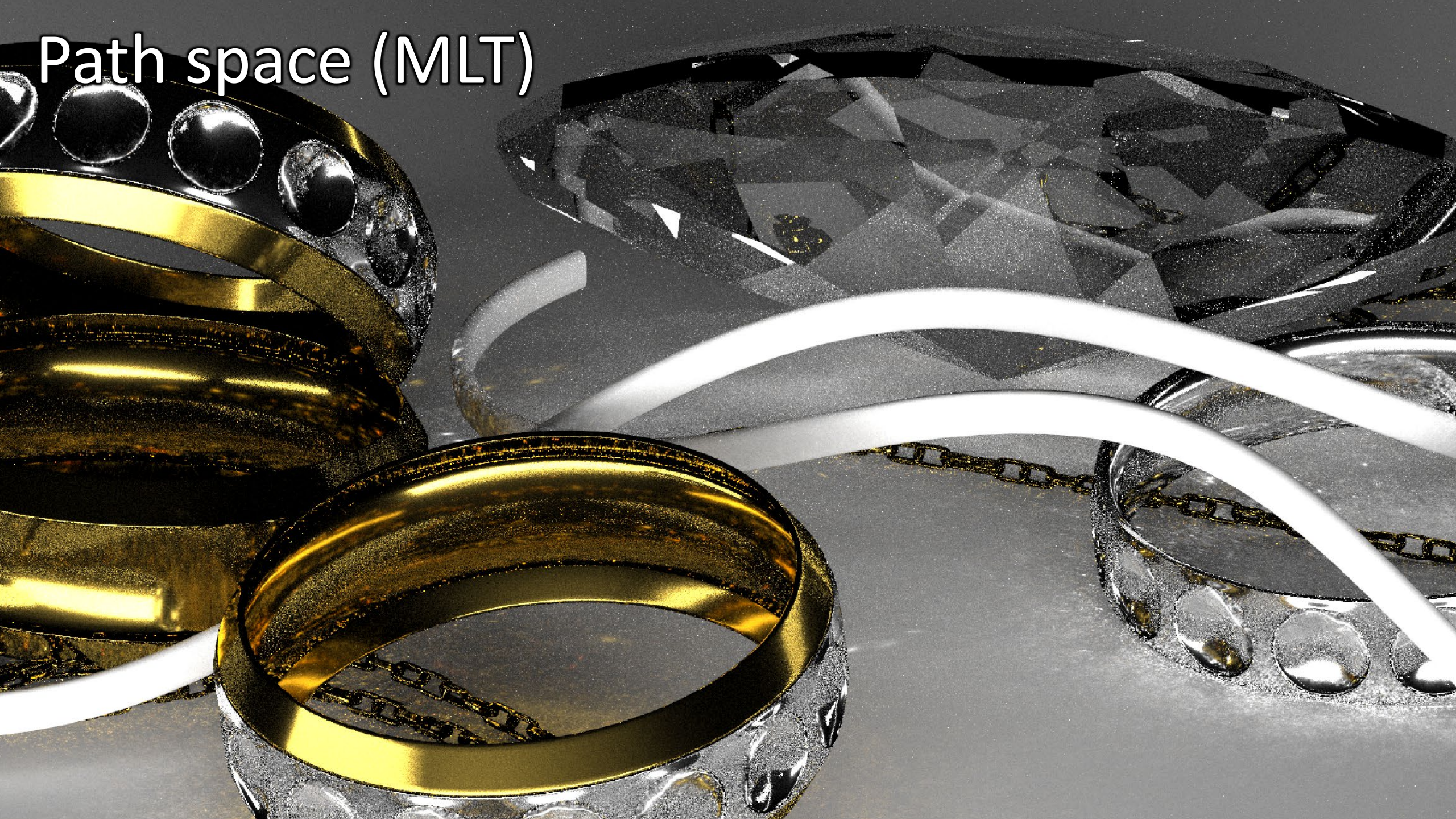


Primary sample space - (Multiplexed MLT)

Primary sample space (Multiplexed MLT)

Error: 0.3658

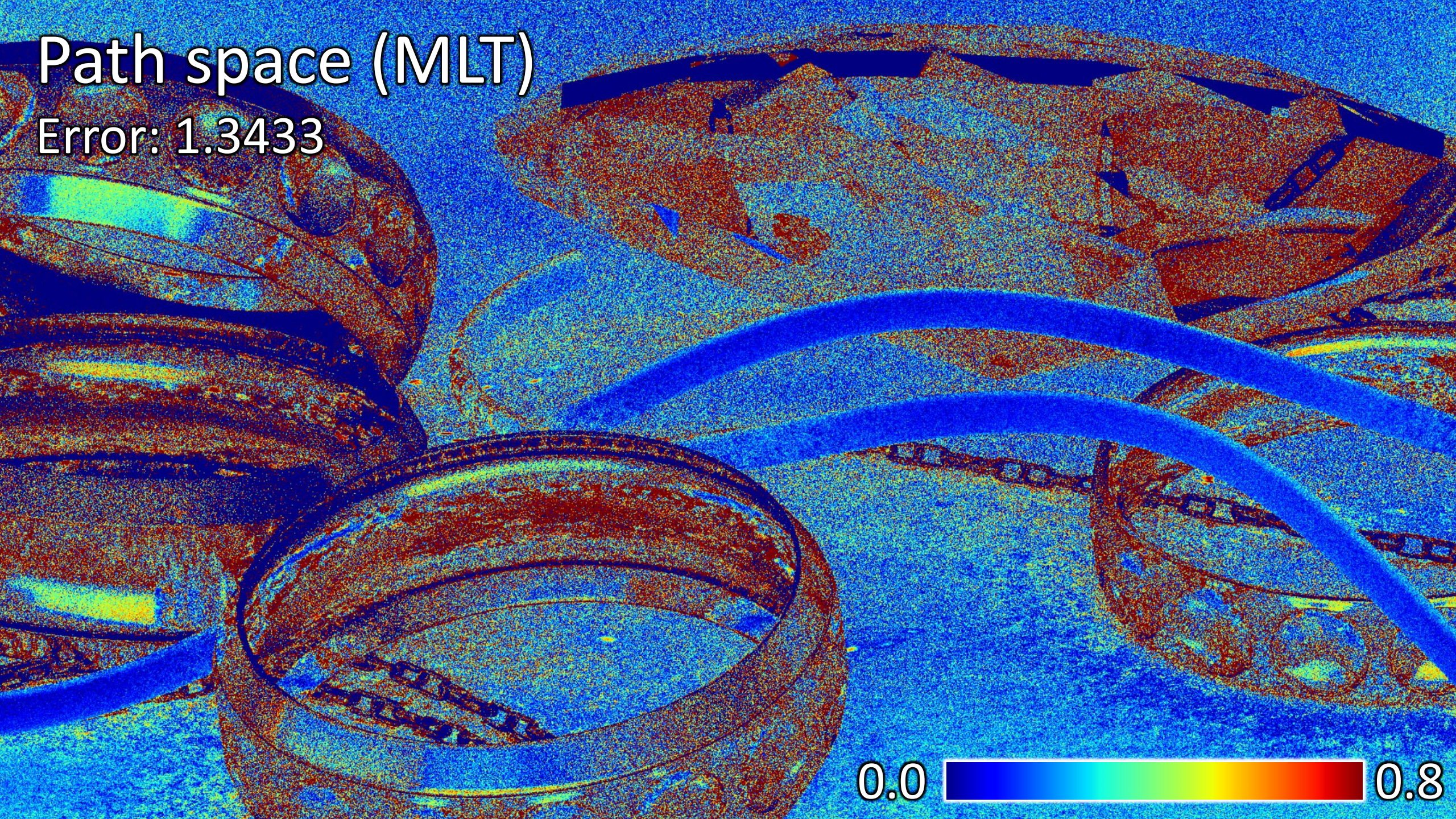




Path space (MLT)

Path space (MLT)

Error: 1.3433



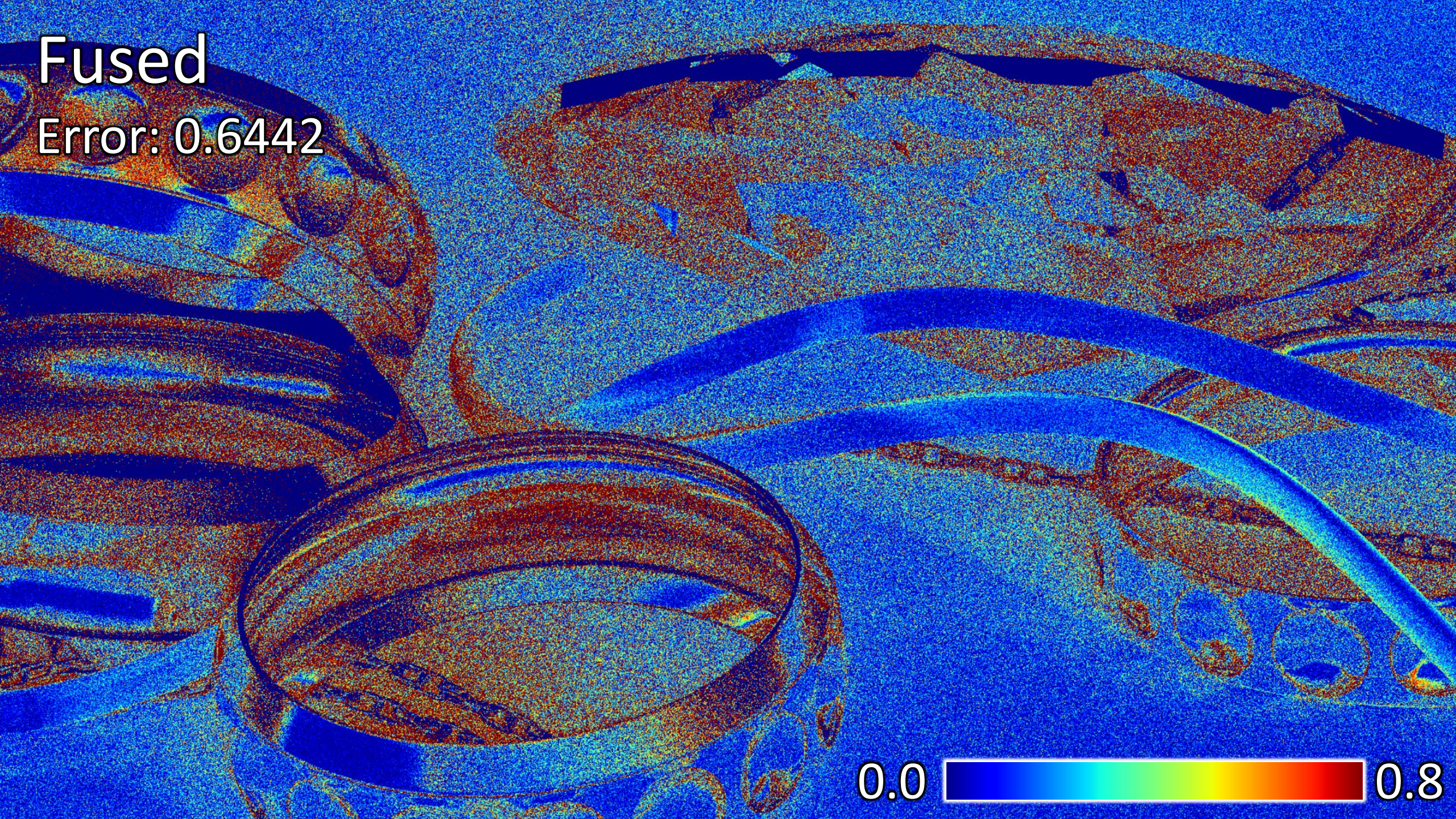
0.0



0.8



Fused



Fused

Error: 0.6442

0.0



0.8

Reference





MLT



Multiplexed MLT

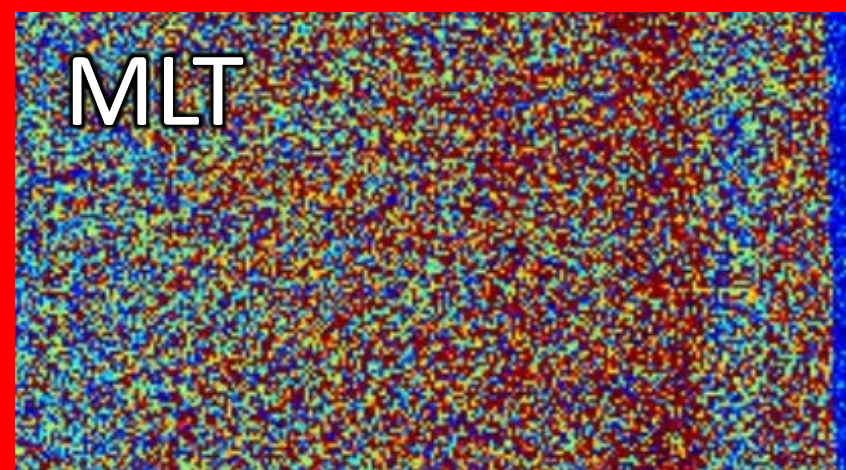


Fused

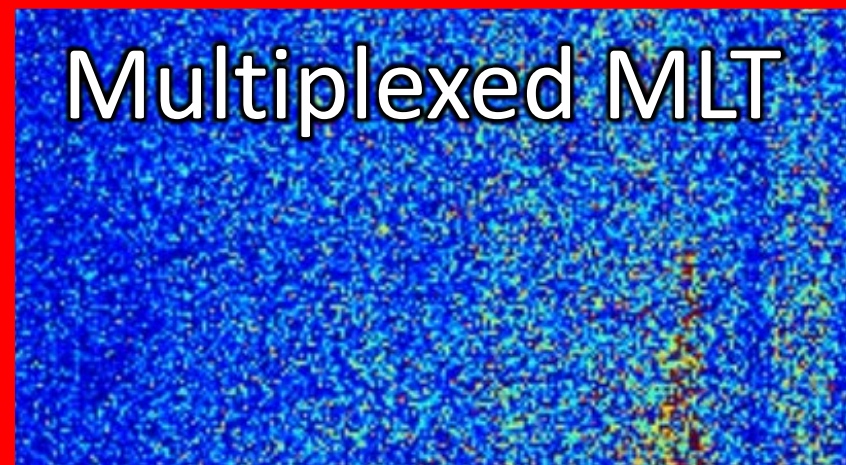




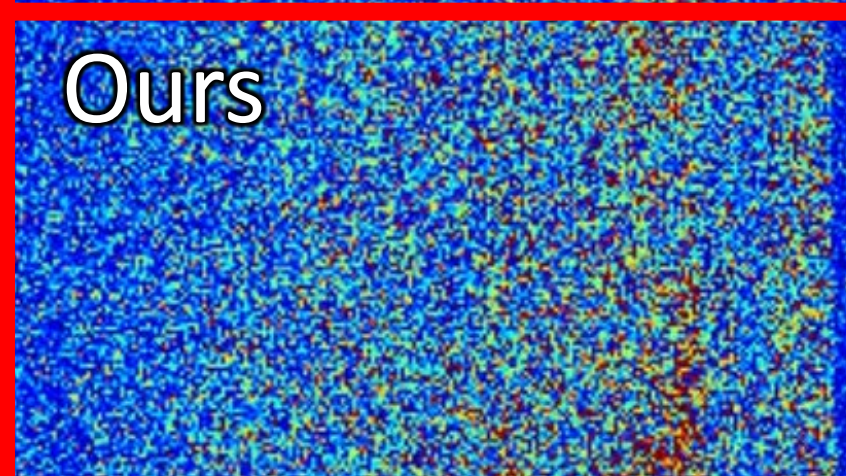
MLT



Multiplexed MLT

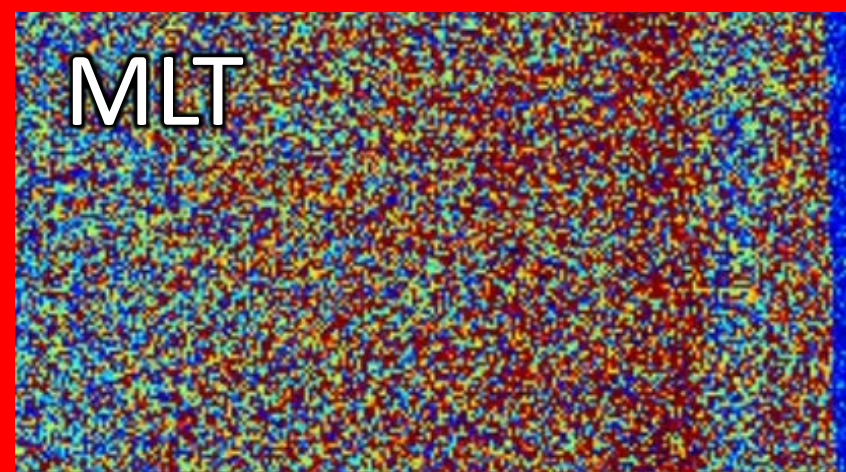


Ours

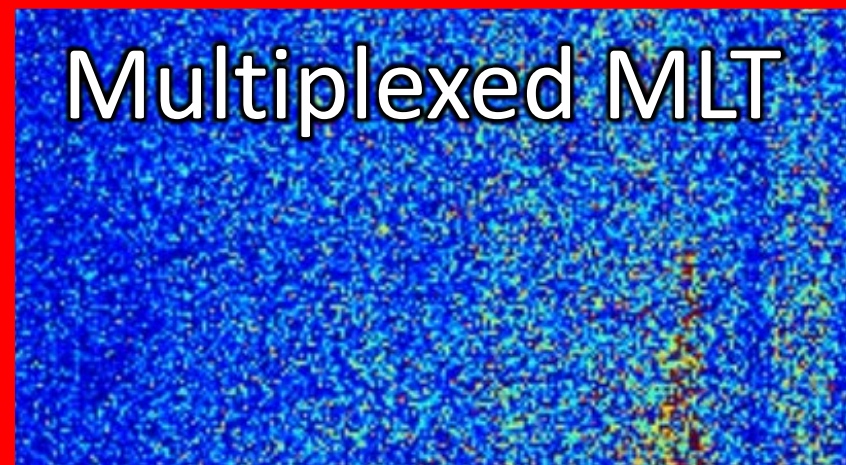




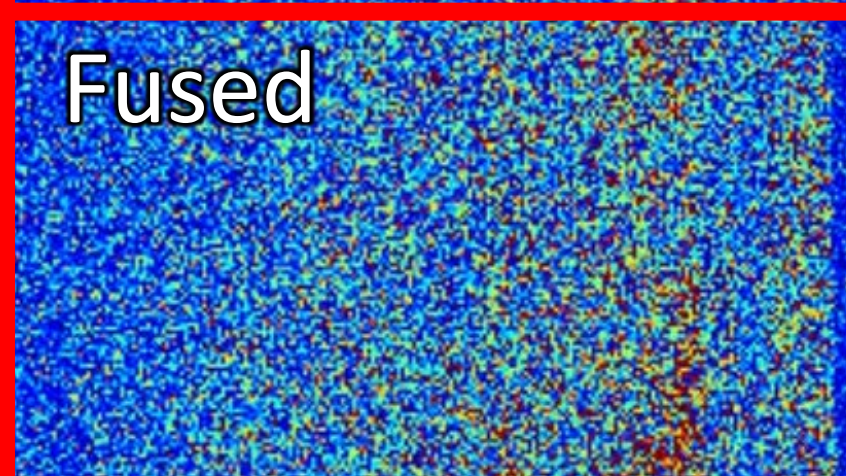
MLT



Multiplexed MLT

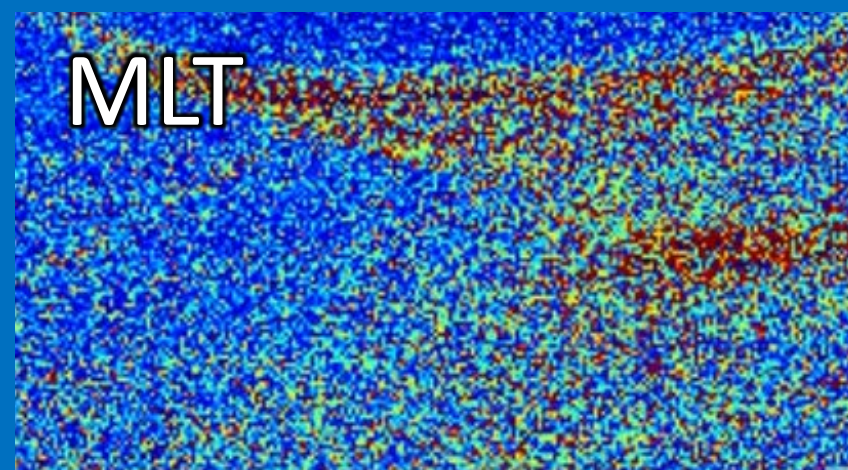


Fused

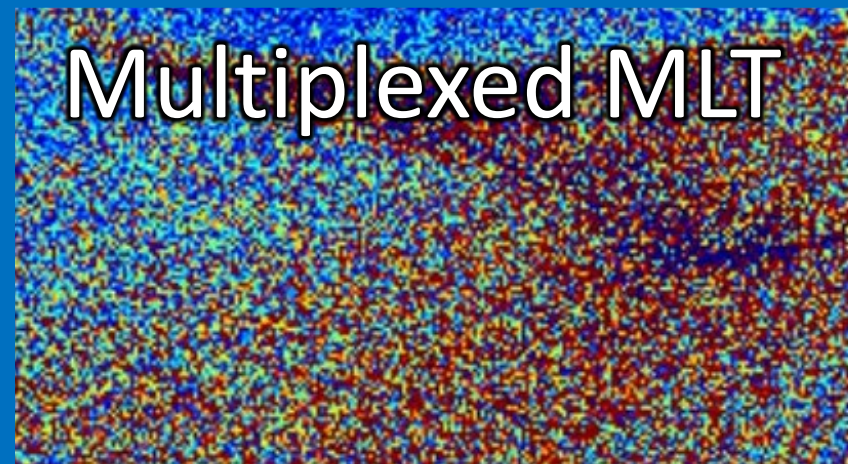




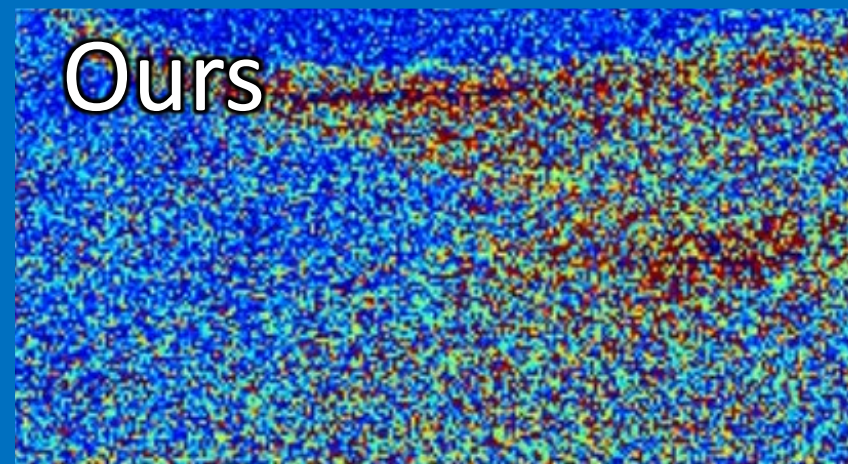
MLT



Multiplexed MLT



Ours



GRADIENTS ARE AWESOME



Deep learning

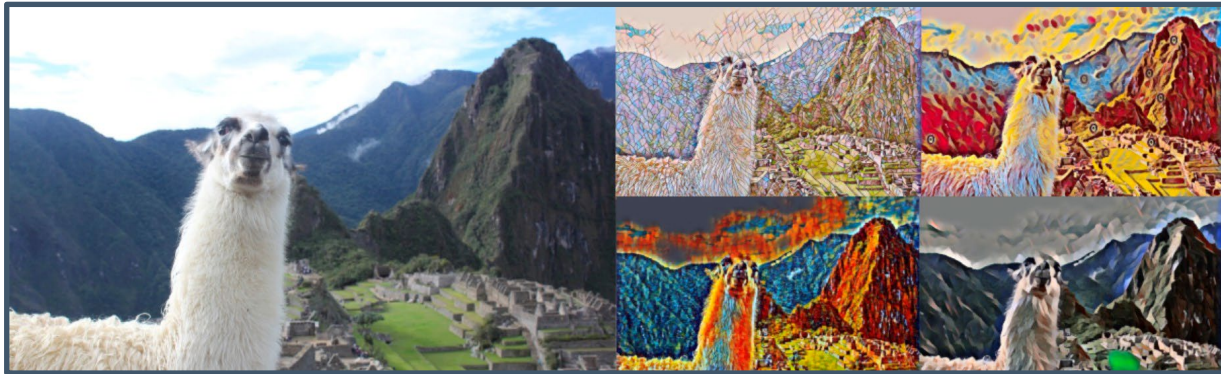
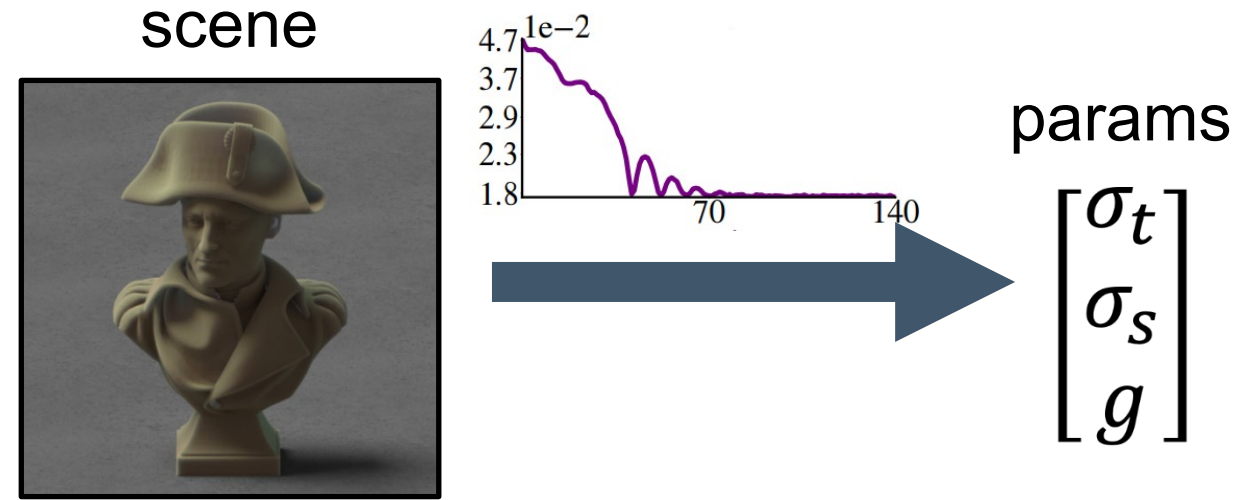


Image processing



Inverse problems



Physics simulation



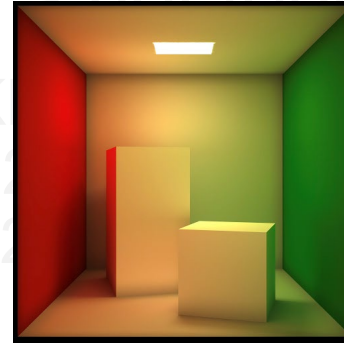
GRADIENTS IN RENDERING

Differentiable rendering is a hot topic:

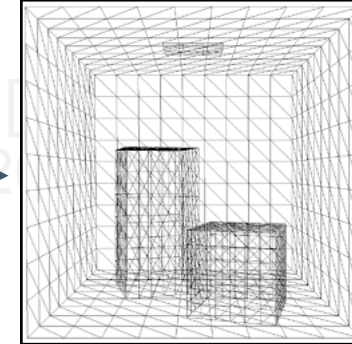
- [Gkioulekas et al. 2013, 2016], [Khungurn et al. 2015], [Zhao et al. 2016], [Che et al. 2018], [Li et al. 2018], [Tsai et al. 2019], [Loubet et al. 2019], [Zhang et al. 2019, 2020], [Nimier-David et al. 2019, 2020]...

GRADIENTS IN RENDERING

Mainly used for
inverse problem:

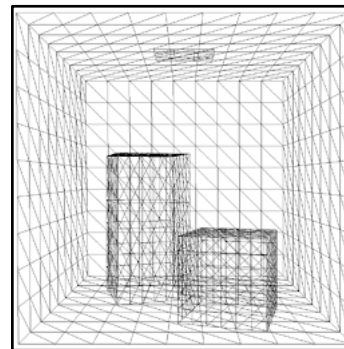


rendering

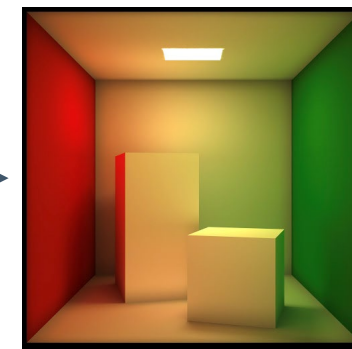


params

We focus on
forward rendering:



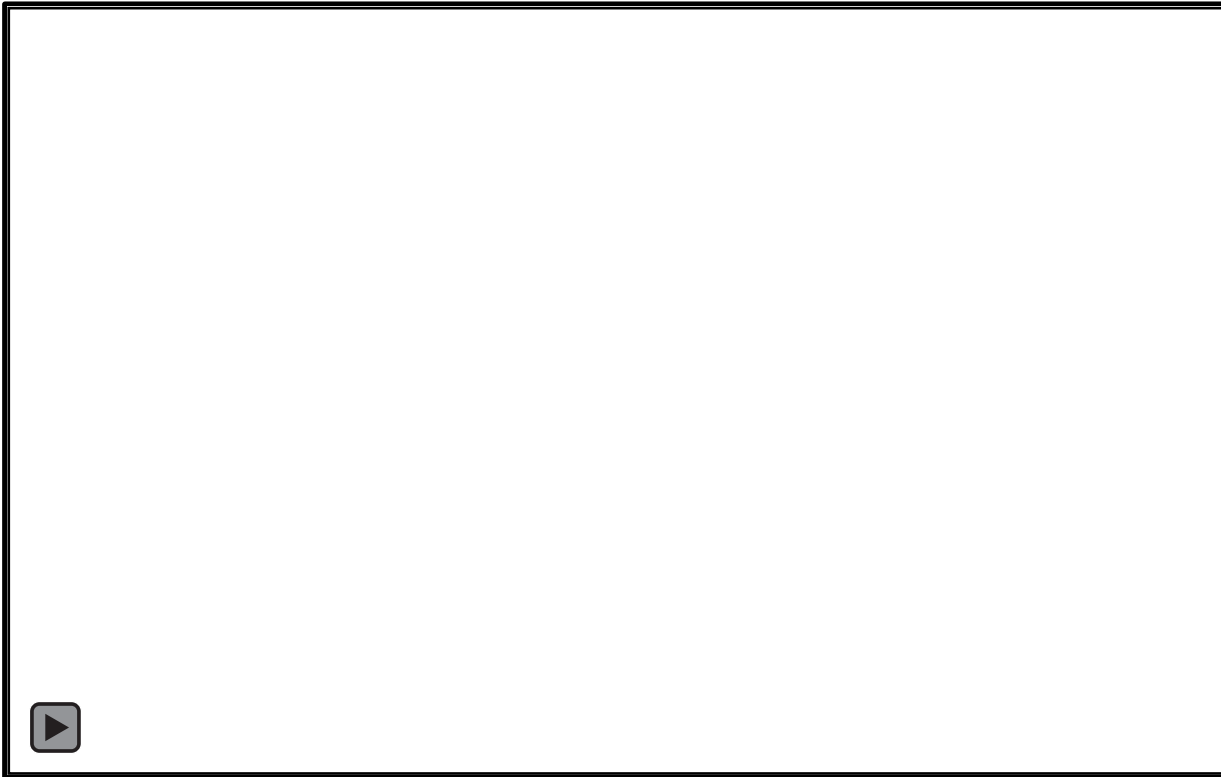
params



rendering

Optimization:

- Stochastic Gradient Descent (SGD)

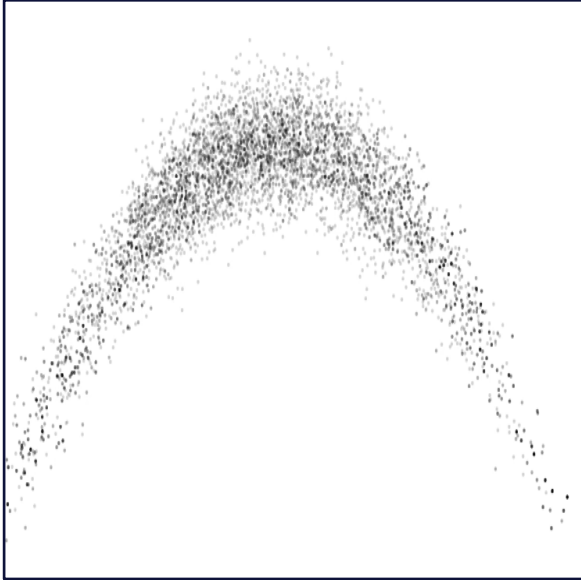


MCMC sampling:

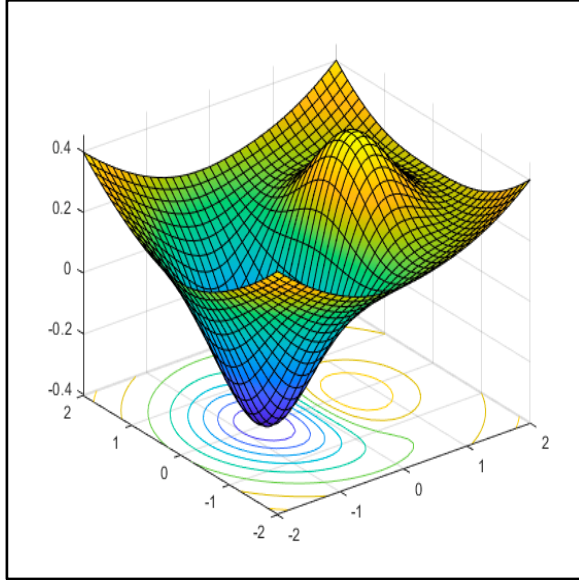
- Langevin Monte Carlo (LMC)



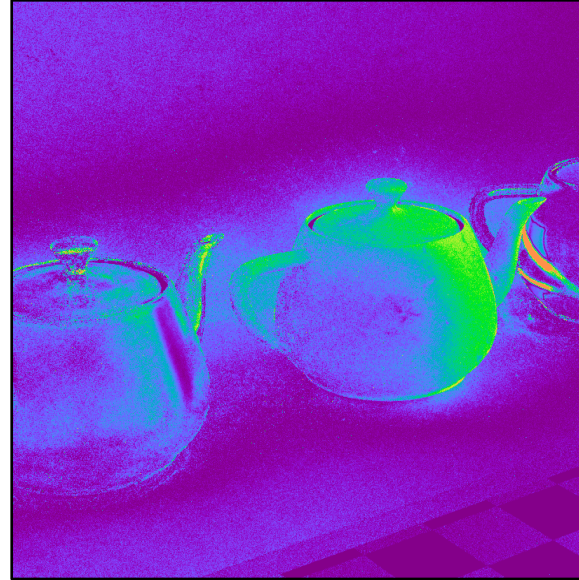
OUTLINE



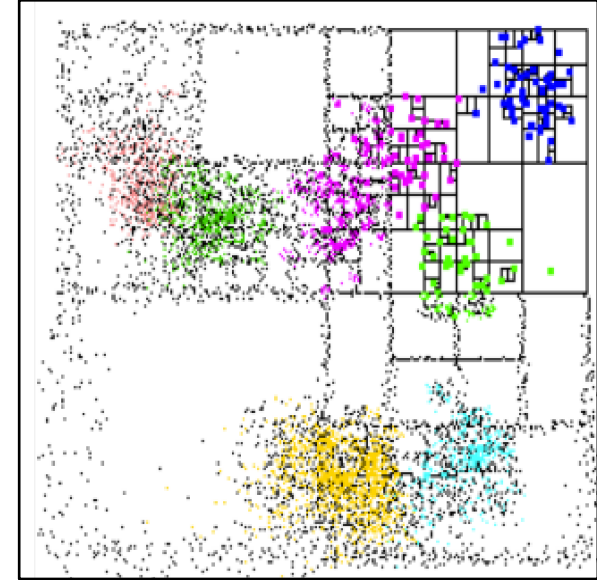
Introduction to
Langevin Monte Carlo (LMC)



Optimization-inspired
acceleration



Ensuring unbiasedness

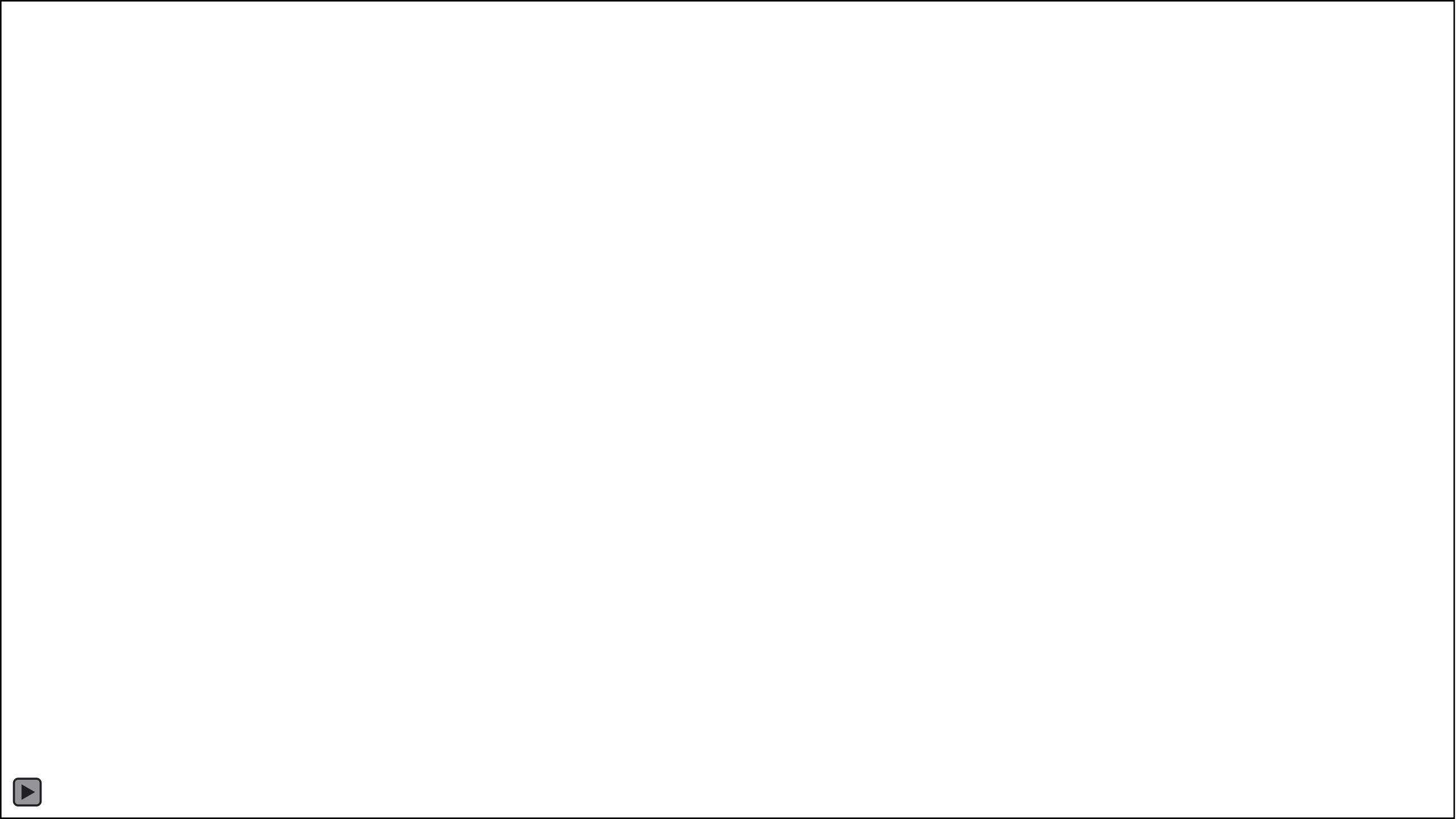


Gradient caching

H2MC [L



Ours



SAMPLING IN RENDERING

$$I = \int f(x) dx$$

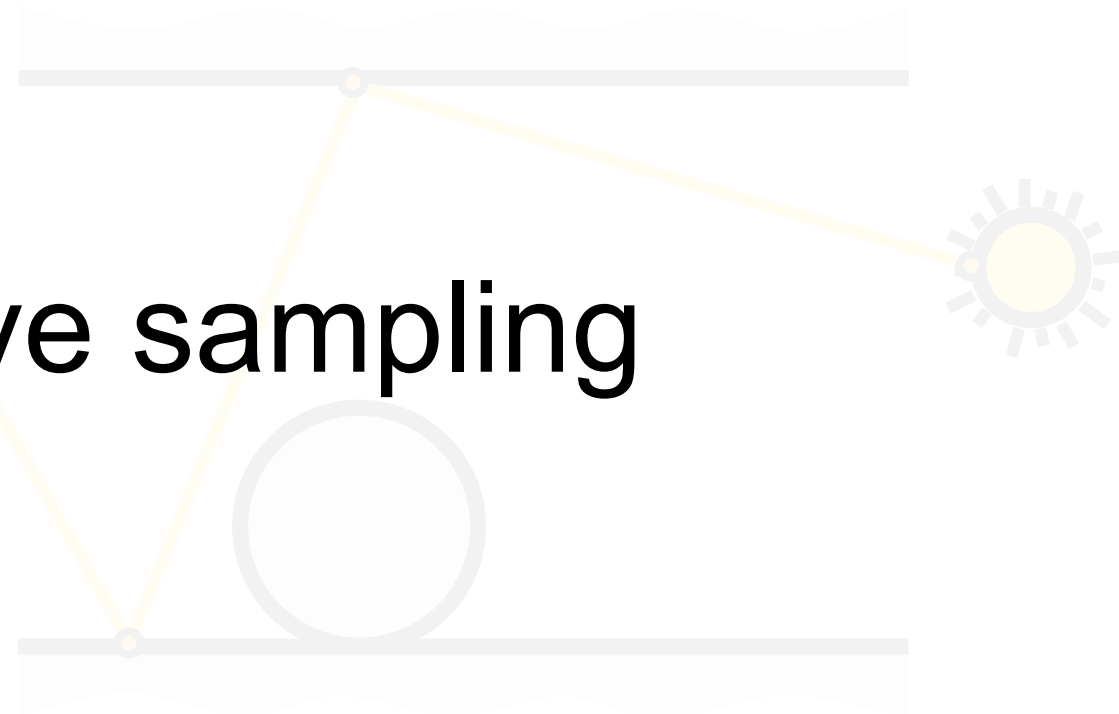
- Estimated with Monte Carlo
- Requires $x \sim f$ for efficiency

Use ∇f to improve sampling

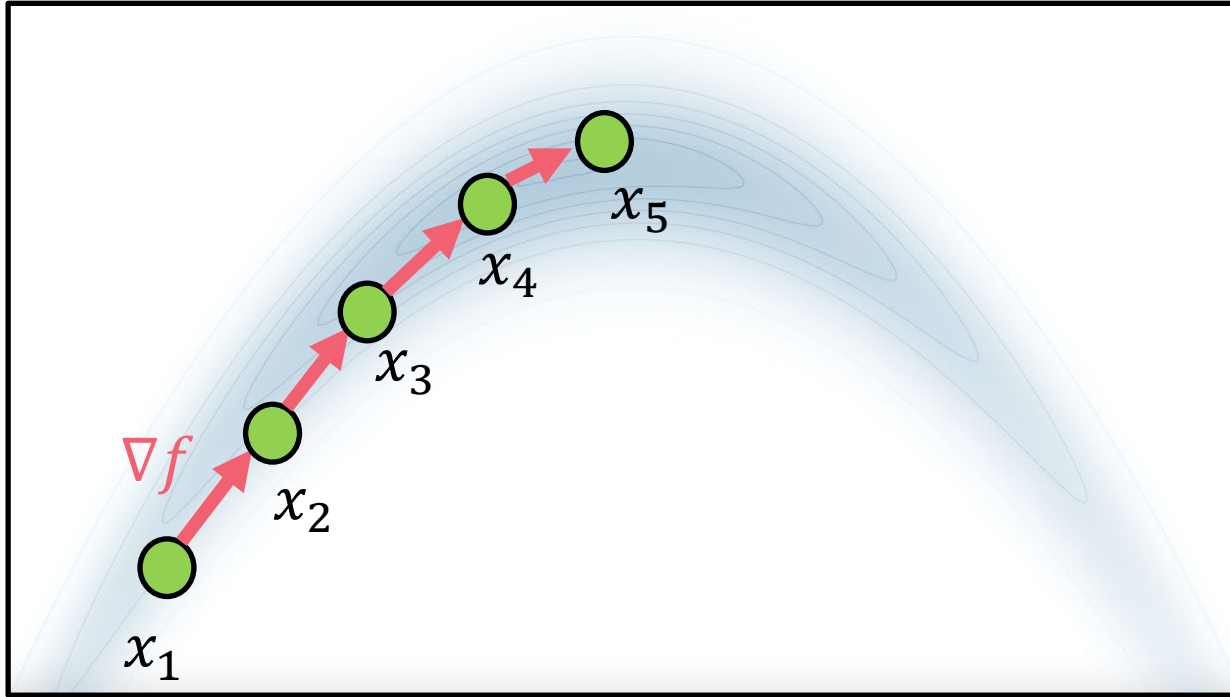
Light path
construction

Primary sample space

$$x \in [0, 1]^N$$



SGD OVERVIEW



Optimization problem:
$$\max_x f(\mathbf{x})$$

Stochastic gradient descent/ascent:

$$\mathbf{x}_t = \mathbf{x}_{t-1} + s_{t-1} \nabla f(\mathbf{x}_{t-1})$$

↑
scalar step size

KELEMEN 2002



Sampling problem:

$$\mathbf{x}_t \sim f$$

Kelemen [2002]:

$$\mathbf{x}_t = \mathbf{x}_{t-1} + N(0, \sigma^2 \mathbf{I})$$

↙ Gaussian noise

Apply Metropolis Hastings to accept/reject

LMC OVERVIEW



Sampling problem:

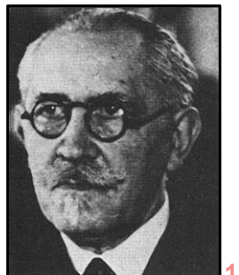
$$\mathbf{x}_t \sim f$$

Langevin MC:

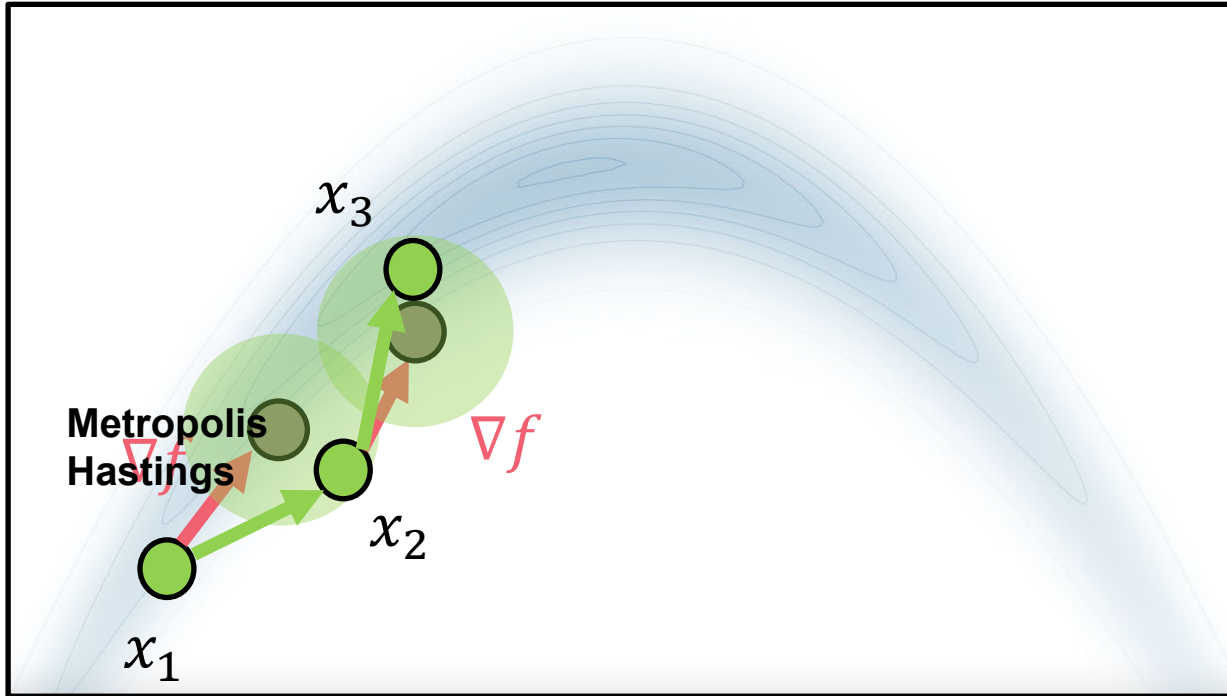
$$\mathbf{x}_t = \mathbf{x}_{t-1} + s_{t-1} \nabla f(\mathbf{x}_{t-1}) + \frac{1}{s_{t-1}} N(0, \sigma^2 I)$$

Apply Metropolis Hastings to accept/reject

Paul Langevin



LMC OVERVIEW



Sampling problem:

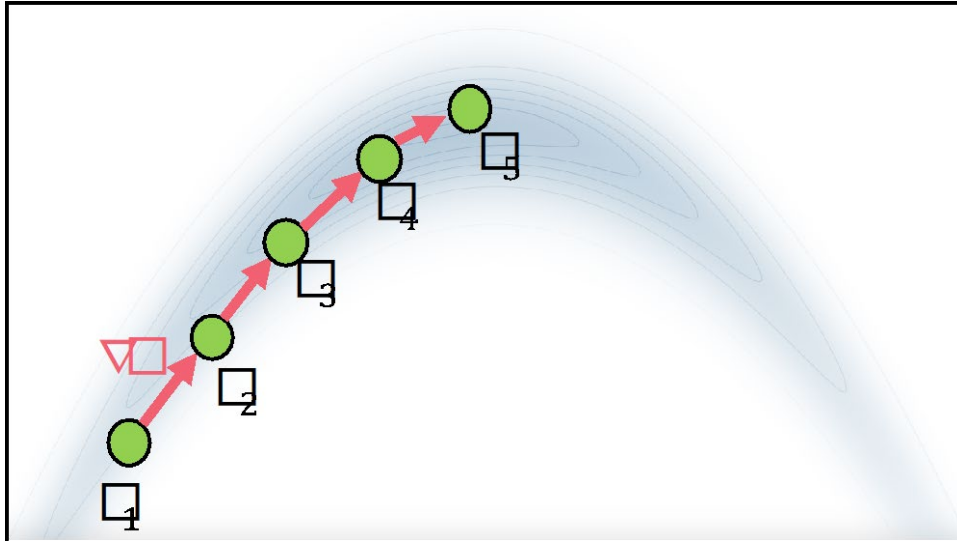
$$\mathbf{x}_t \sim f$$

Langevin MC:

$$\mathbf{x}_t = \mathbf{x}_{t-1} + s_{t-1} \nabla f(\mathbf{x}_{t-1}) + \frac{1}{s_{t-1}} N(0, \sigma^2 I)$$

Apply Metropolis Hastings to accept/reject

SGD VS. LMC

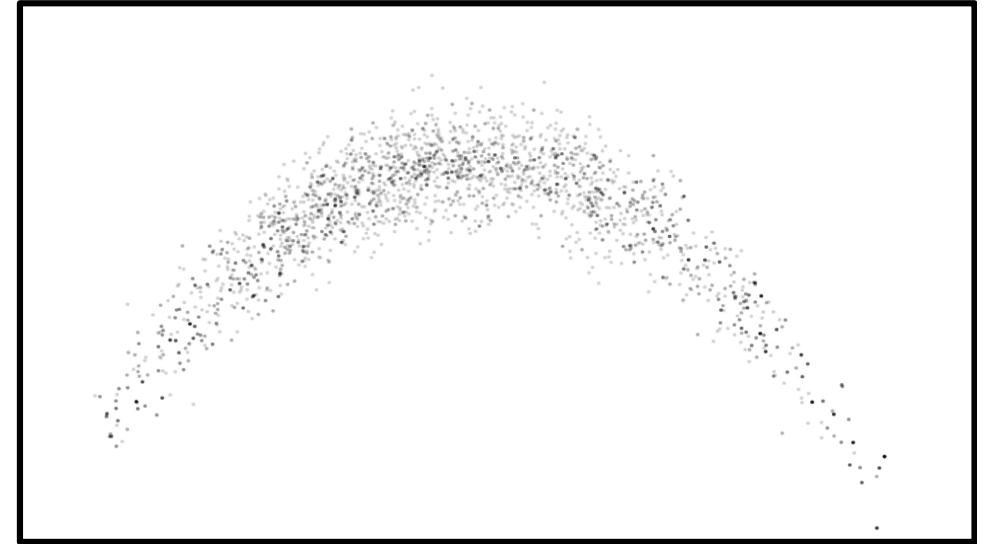


Optimization:

$$\max_x f(x)$$

SGD:

$$\mathbf{x}_t = \mathbf{x}_{t-1} + s_{t-1} \nabla f(\mathbf{x}_{t-1})$$



Sampling:

$$\mathbf{x}_t \sim f$$

LMC:

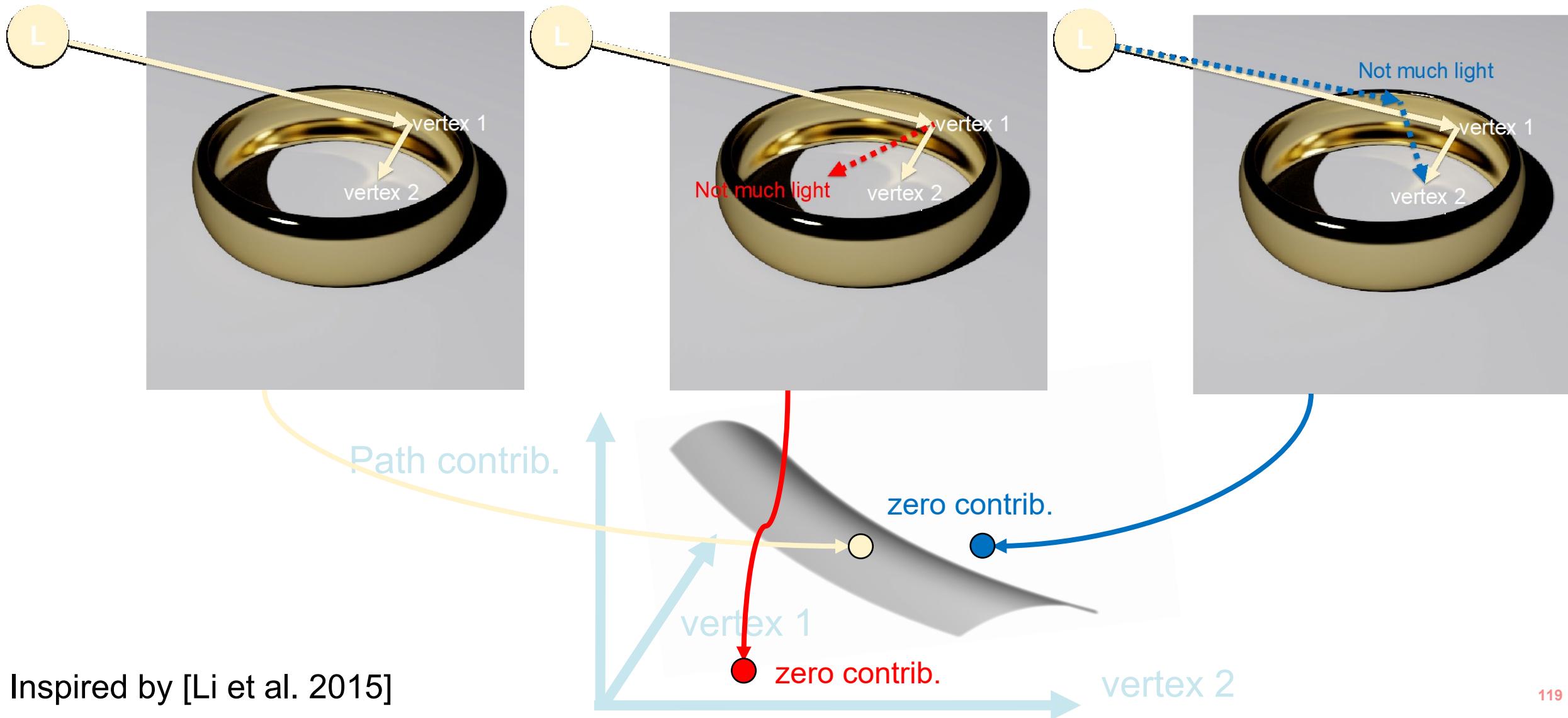
$$\mathbf{x}_t = \mathbf{x}_{t-1} + s_{t-1} \nabla f(\mathbf{x}_{t-1}) + \frac{1}{s_{t-1}} N(0, \sigma^2 I)$$

Original LMC

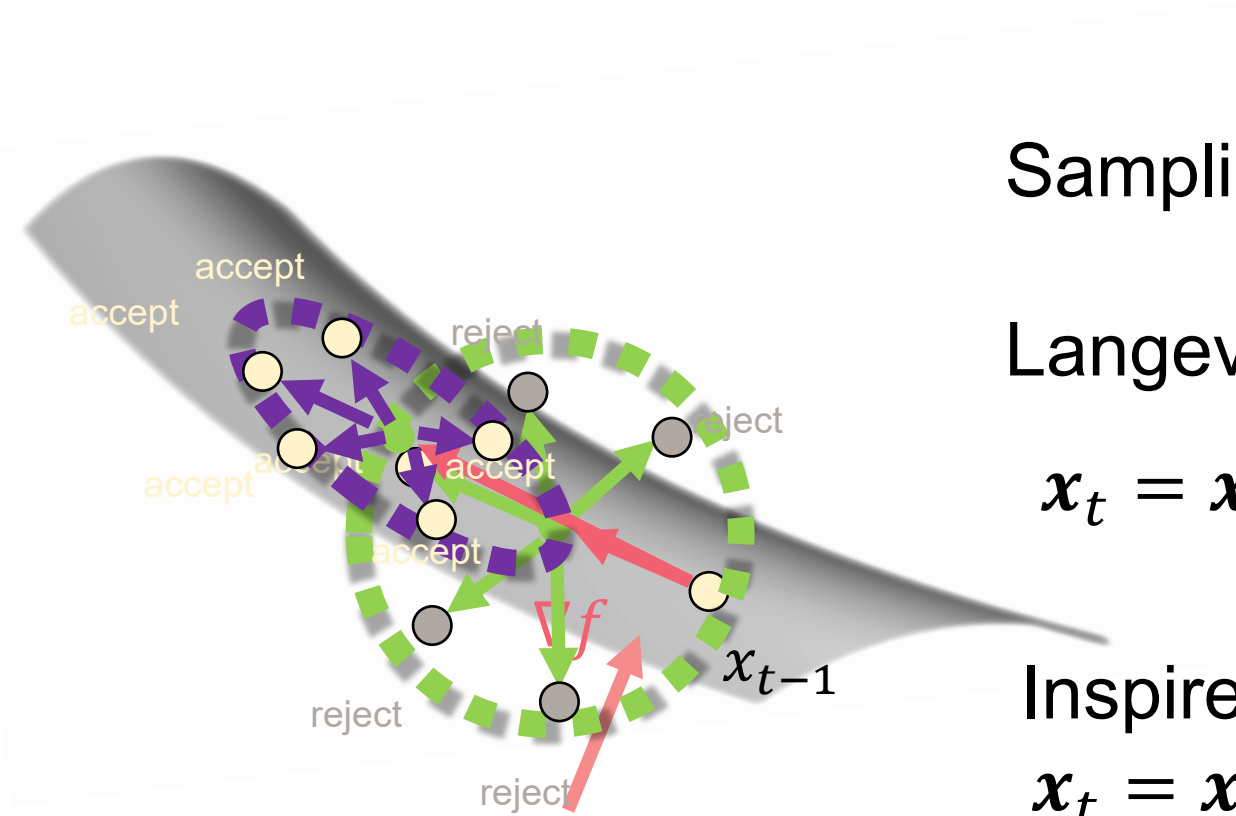
MSE: 0.2465



RING CAUSTICS



RING CAUSTICS



Gradient vanishes on flat surface

Sampling problem: $\mathbf{x}_t \sim f$

Langevin MC:
 ☒ Slow exploration
 ☒ Isotropic noise

$$\mathbf{x}_t = \mathbf{x}_{t-1} + s_{t-1} \nabla f(\mathbf{x}_{t-1}) + \frac{1}{s_{t-1}} N(0, \sigma^2 I)$$

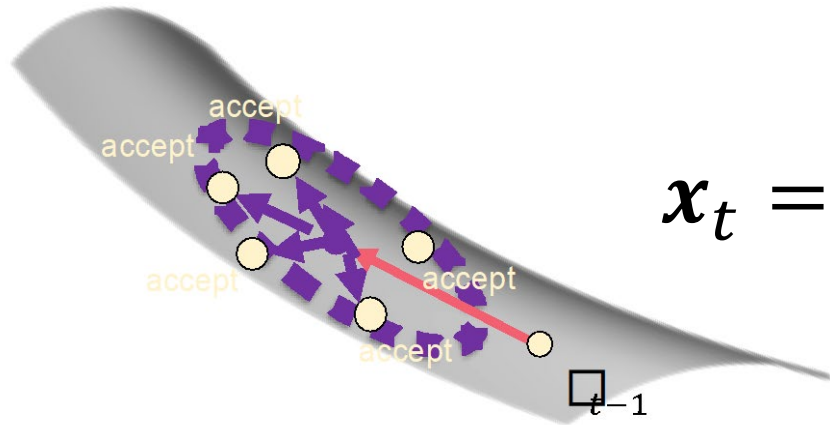
Inspired by optimization:

- ☒ Fast exploration
- ☒ Adapt to local geometry

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \mathbf{K}_{t-1} \nabla f(\mathbf{x}_{t-1}) + \mathbf{K}_{t-1}^{-1} N(0, \sigma^2 I)$$

preconditioning matrix

SELECTING THE PRECONDITIONING MATRIX



$$\mathbf{x}_t = \mathbf{x}_{t-1} + \underbrace{K_{t-1}}_{\text{preconditioning matrix}} \nabla f(\mathbf{x}_{t-1}) + \underbrace{K_{t-1}^{-1}}_{\text{preconditioning matrix}} N(0, \sigma^2 I)$$

preconditioning matrix

Exact **Hessian** of f [Li et al. 2015]

- ✗ requires **2nd-order** gradients
- ✗ full matrix
- ✗ expensive matrix operations

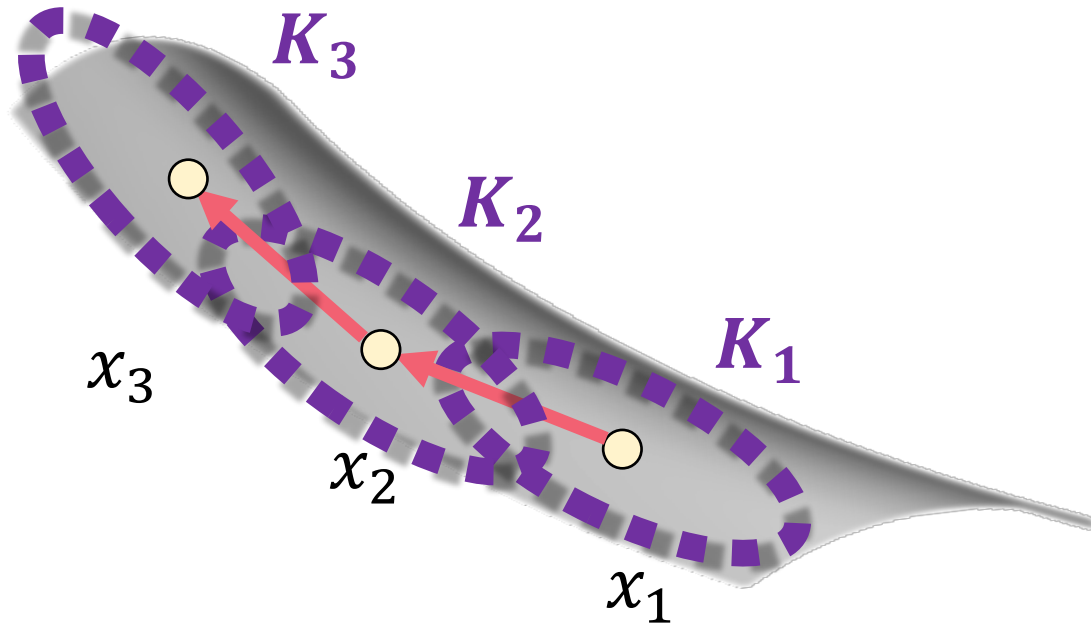
Approximate **Hessian** of f (ours)

- ✓ reuses **1st-order** gradients
- ✓ diagonal matrix
- ✓ efficient scalar operations

quasi-Newton methods: **Adam**, BFGS, ...

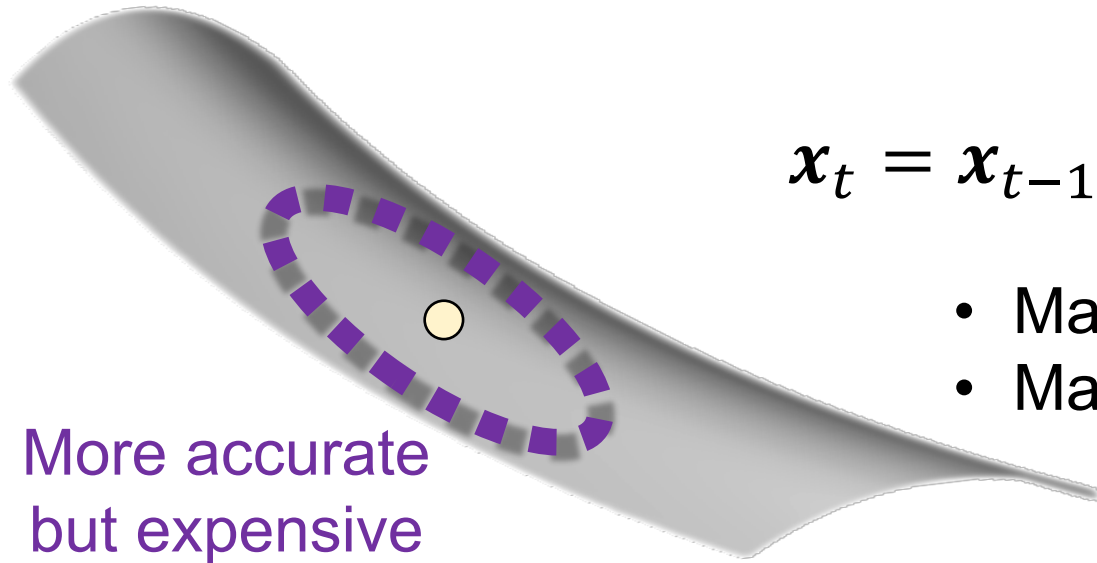
LMC + ADAM

Adam preconditioning matrix K_t is a function of **all** previous gradients



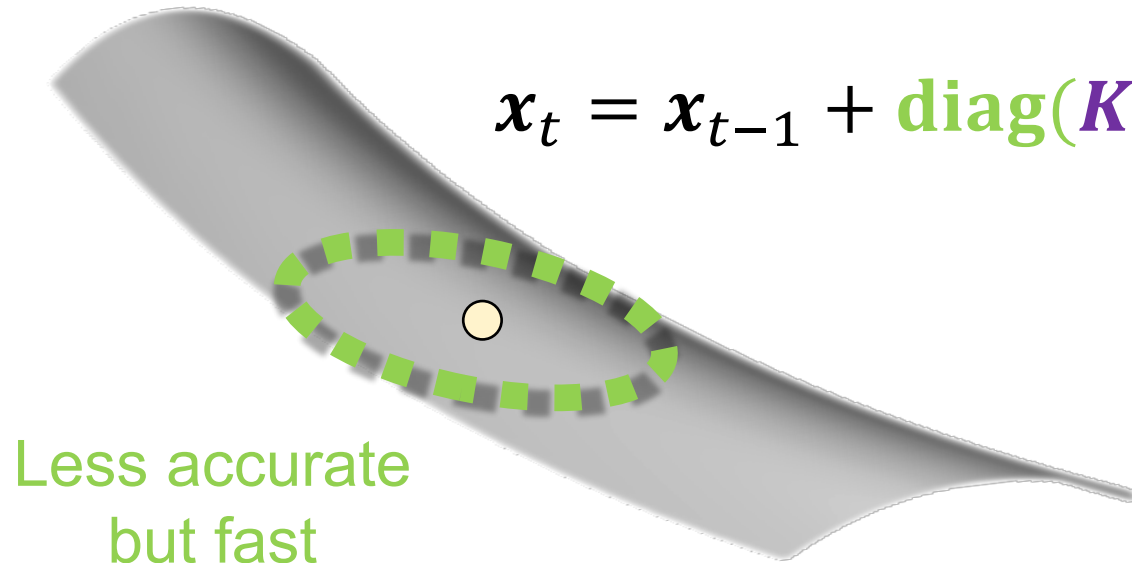
K_3 depends on
 $\nabla f(x_3), \nabla f(x_2), \nabla f(x_1)$

FULL VS. DIAGONAL PRECONDITIONING



$$\mathbf{x}_t = \mathbf{x}_{t-1} + \mathbf{K}_{t-1} \nabla f(\mathbf{x}_{t-1}) + \mathbf{K}_{t-1}^{-1} N(0, \sigma^2 \mathbf{I})$$

- Matrix-vector multiplication with gradient
- Matrix inversion for sampling



$$\mathbf{x}_t = \mathbf{x}_{t-1} + \text{diag}(\mathbf{K}_{t-1}) \nabla f(\mathbf{x}_{t-1}) + \text{diag}(\mathbf{K}_{t-1})^{-1} N(0, \sigma^2 \mathbf{I})$$

Better at equal time!

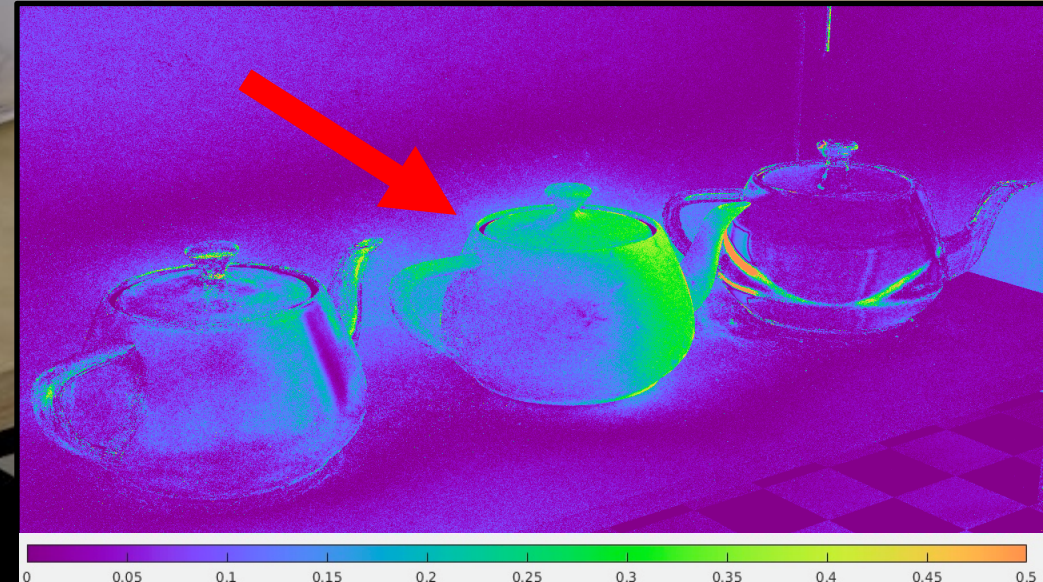
LMC + Adam, diagonal

MSE: 0.0779



Bias problem

Unfortunately, naïvely combining
LMC + Adam causes **Bias!**



WHY BIASED?

Unbiasedness requires **asymptotic time homogeneity**:
Preconditioning matrix $K_t \rightarrow \text{constant}$, when $t \rightarrow \infty$

Adam violates asymptotic time homogeneity:

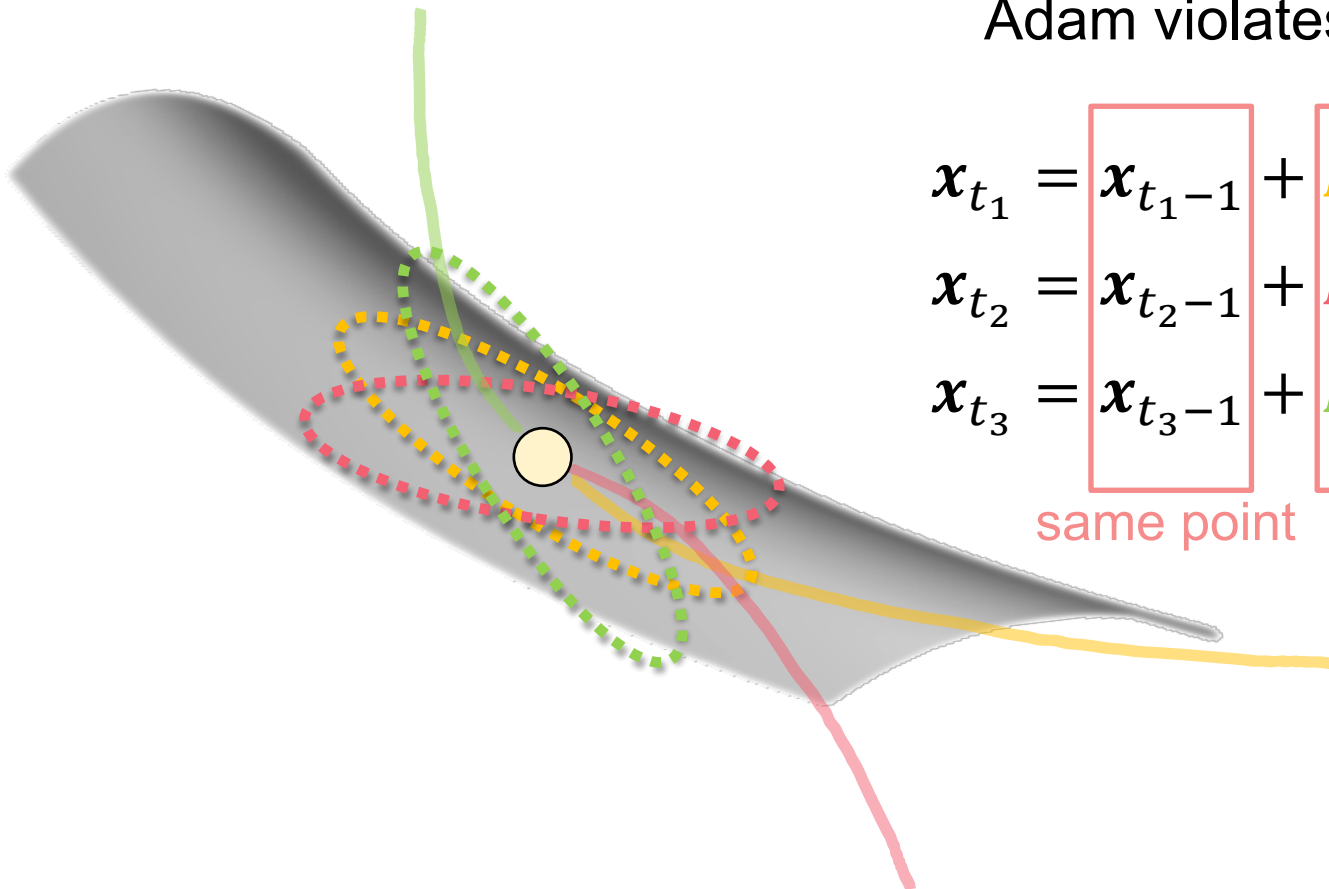
$$\mathbf{x}_{t_1} = \mathbf{x}_{t_1-1} + \mathbf{K}_{t_1-1} \nabla f(\mathbf{x}_{t_1-1}) + \mathbf{K}_{t_1-1}^{-1} N(0, \sigma^2 \mathbf{I})$$

$$\mathbf{x}_{t_2} = \mathbf{x}_{t_2-1} + \mathbf{K}_{t_2-1} \nabla f(\mathbf{x}_{t_2-1}) + \mathbf{K}_{t_2-1}^{-1} N(0, \sigma^2 \mathbf{I})$$

$$\mathbf{x}_{t_3} = \mathbf{x}_{t_3-1} + \mathbf{K}_{t_3-1} \nabla f(\mathbf{x}_{t_3-1}) + \mathbf{K}_{t_3-1}^{-1} N(0, \sigma^2 \mathbf{I})$$

same point

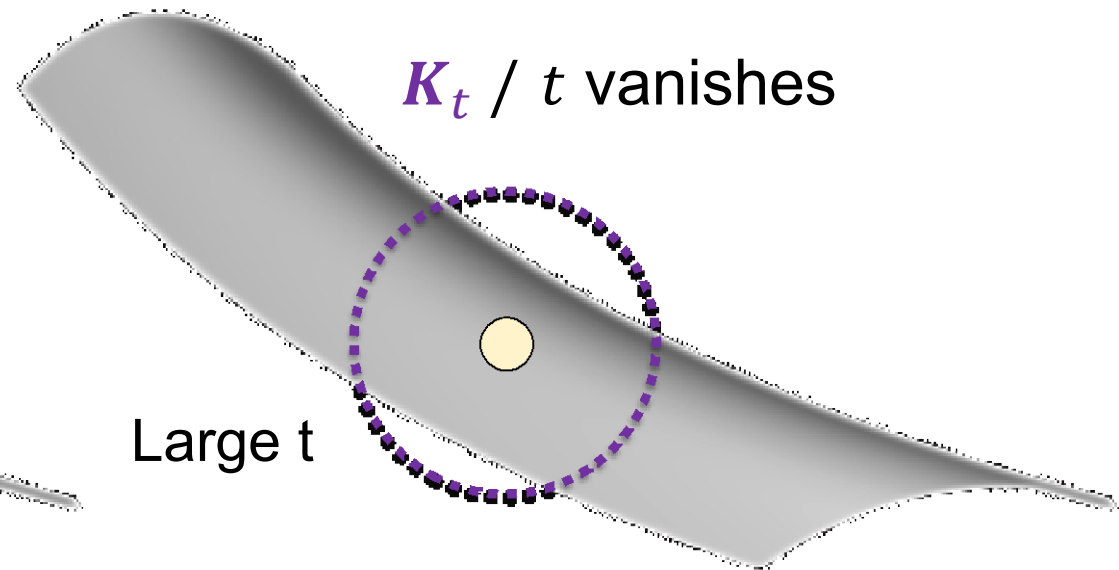
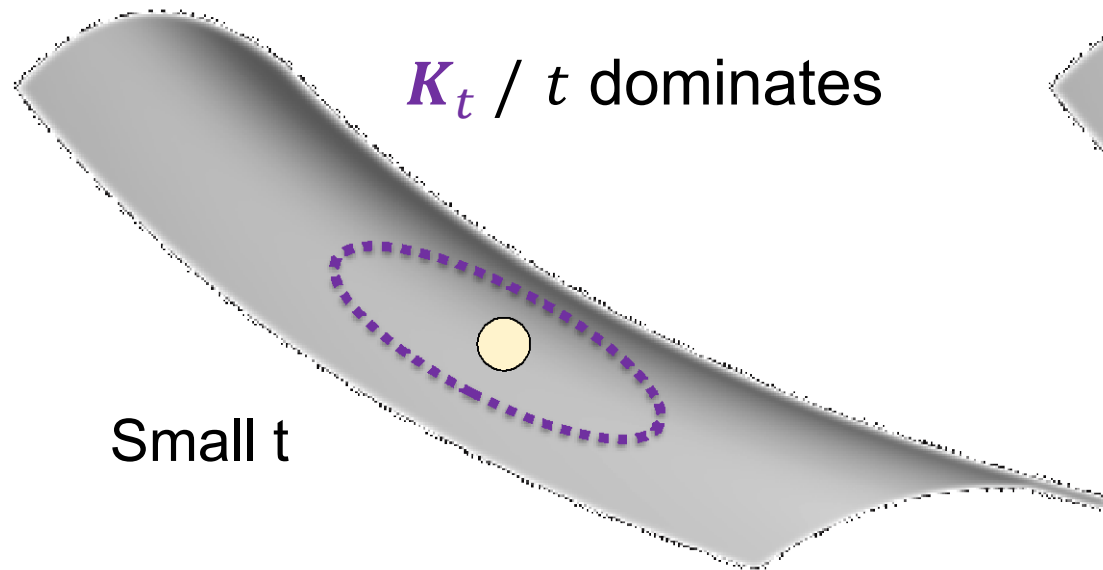
different
preconditioning matrices



SOLUTION 1: DIMINISHING ADAPTATION

Unbiasedness requires **asymptotic time homogeneity**:
Preconditioning matrix $K_t \rightarrow \text{constant}$, when $t \rightarrow \infty$

$$K'_t = \boxed{K_t / t} + I$$



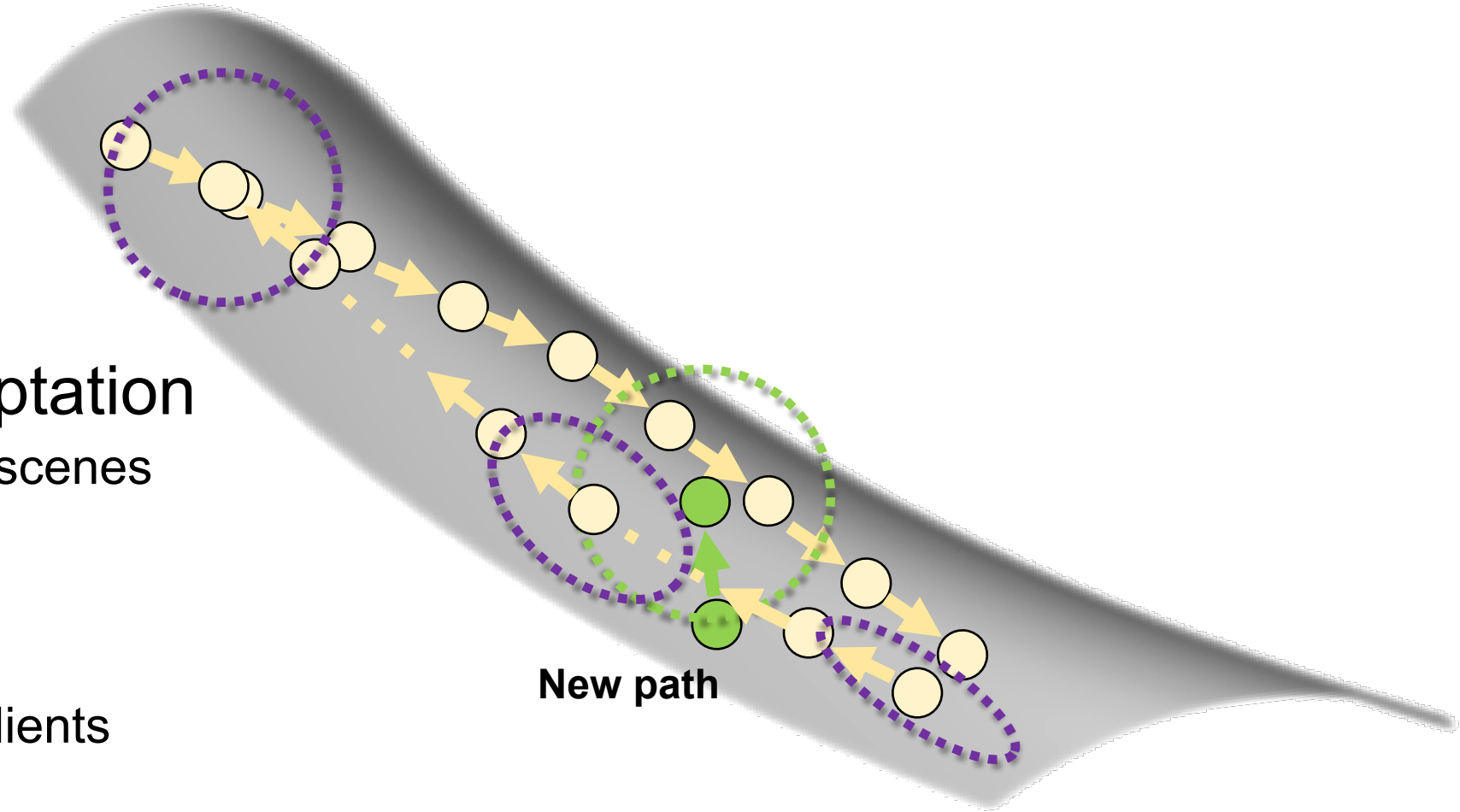
LMC + Adam, diagonal (w/ diminishing adaptation)



DRAWBACKS OF DIMINISHING ADAPTATION

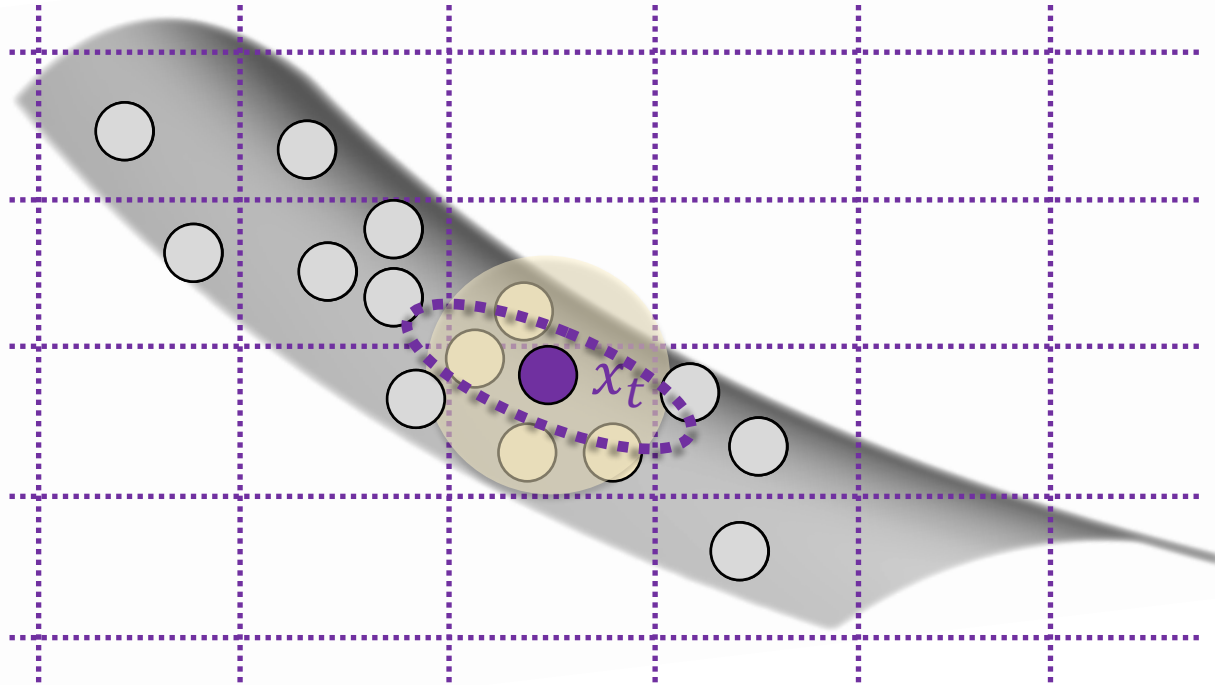
$$K'_t = K_t / t + I$$

- Gradual loss of adaptation
 - Problematic for complex scenes
- No gradient reuse
 - Need to re-calculate gradients



SOLUTION 2: CACHE-DRIVEN ADAPTATION

Unbiasedness requires **asymptotic time homogeneity**:
Preconditioning matrix $K_t \rightarrow \text{constant}$, when $t \rightarrow \infty$



- Stage 1
 - use Adam
 - cache gradients
- Stage 2
 - freeze cache
 - use cached gradients

$$K_t = \text{cacheQuery}(x_t)$$

Cache-driven adaptation

MSE: 0.0459



COMPARISONS WITH PRIOR WORK

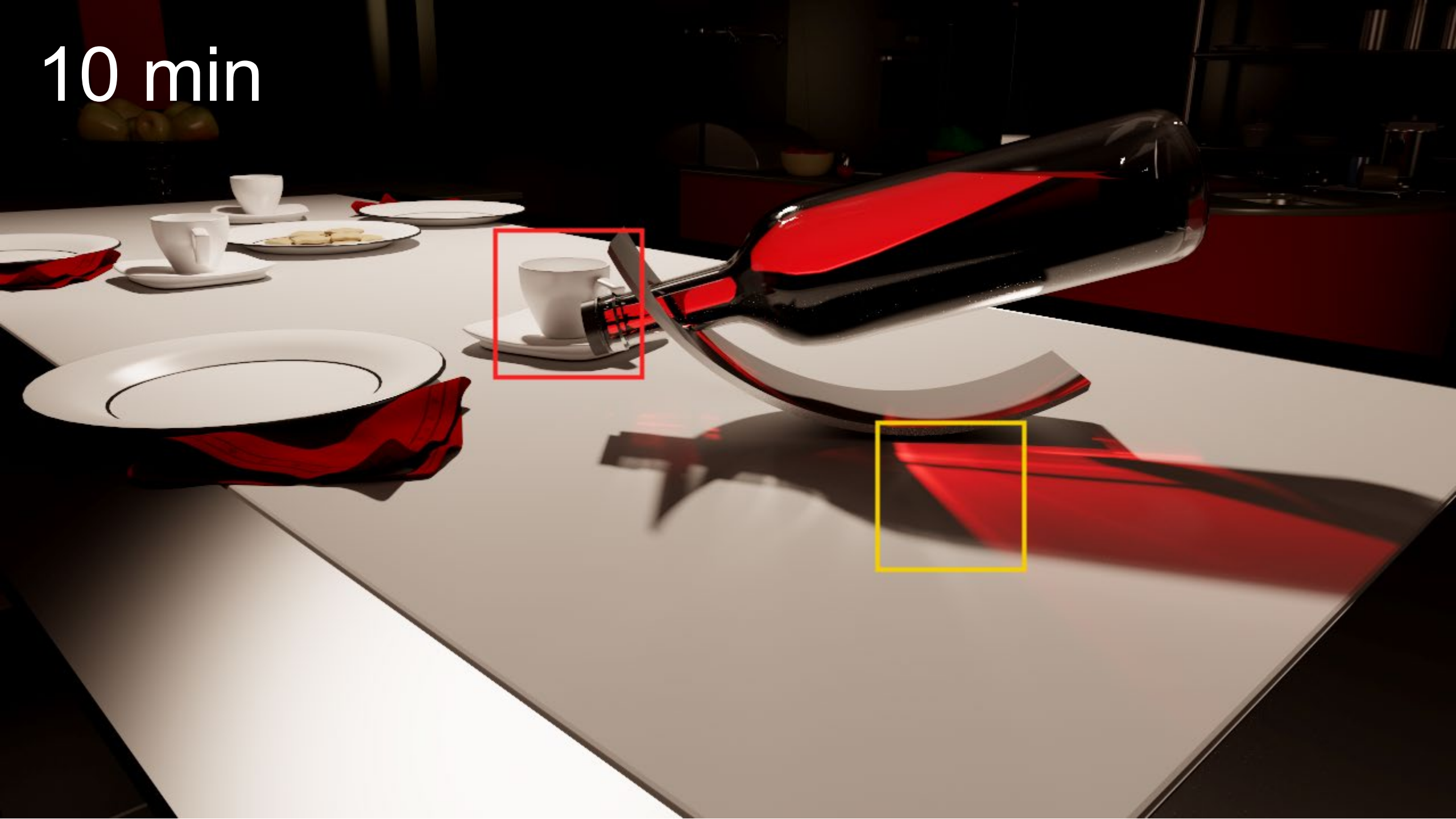








10 min



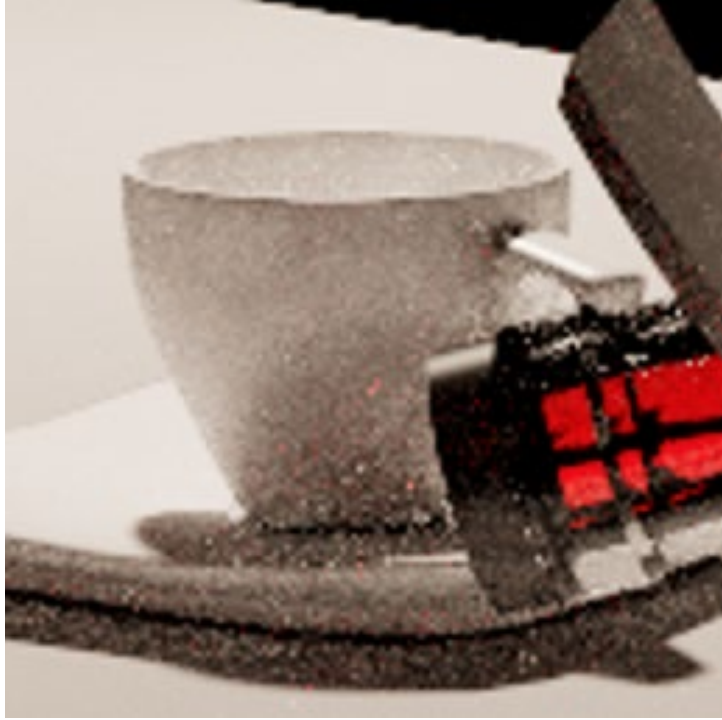
MEMLT [Jakob 2012]



H2MC [Li 2015]



Ours



MMLT [Hachisuka 2014]



Reference



RJMLT [Bitterli 2017]



MEMLT [Jakob 2012]



H2MC [Li 2015]



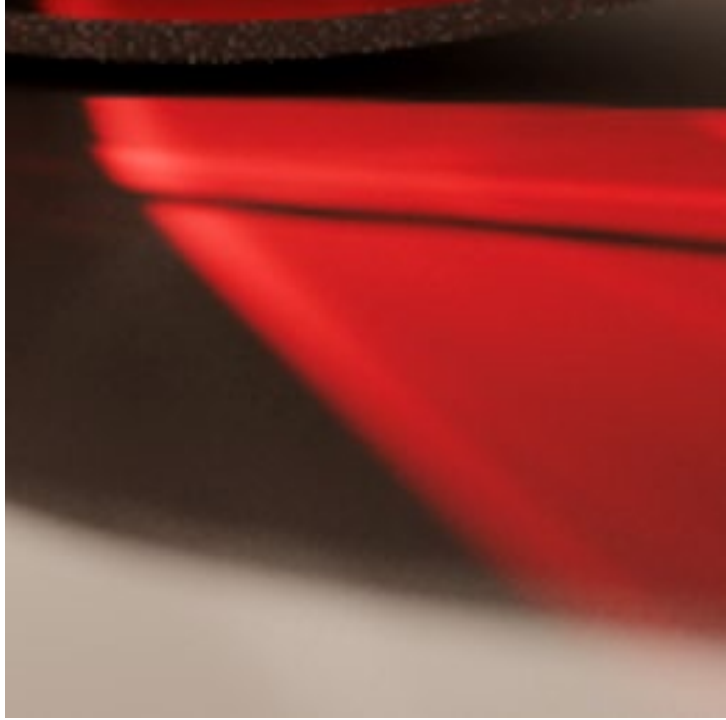
Ours

MMLT [Hachisuka 2014]



Reference

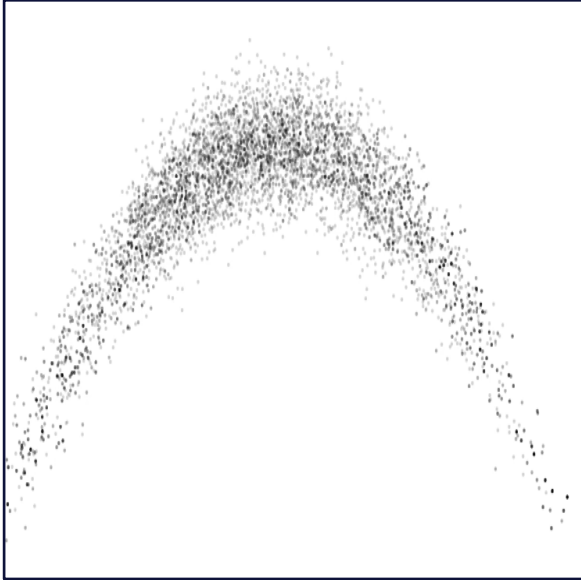
RJMLT [Bitterli 2017]



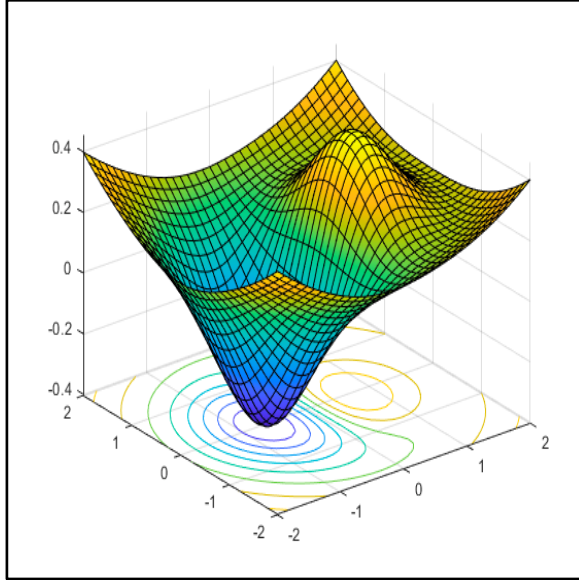
LIMITATIONS AND FUTURE WORK

- require gradients
 - becoming common in modern renderers
- global exploration
 - potentially use gradient cache for this purpose

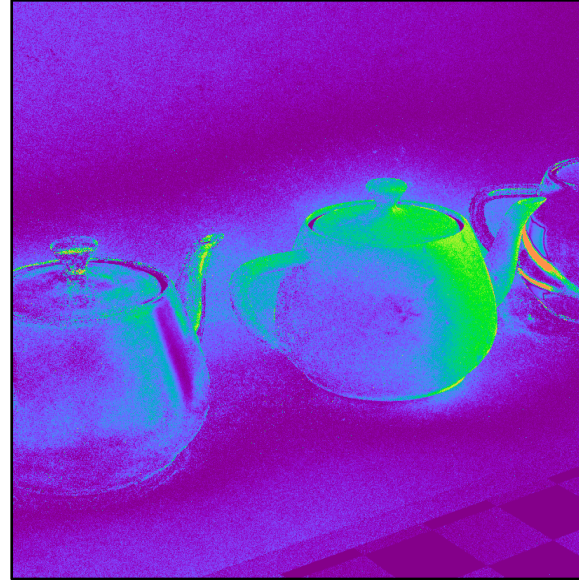
TAKE-HOME MESSAGE



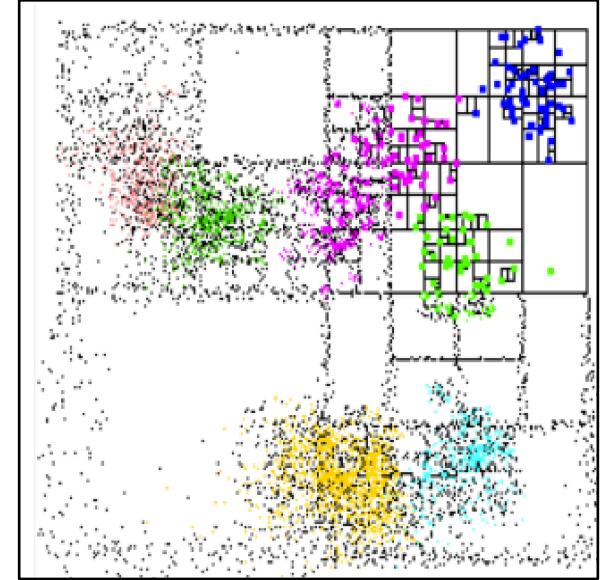
Introduction to
Langevin Monte Carlo (LMC)



Optimization-inspired
acceleration



Ensuring unbiasedness



Gradient caching