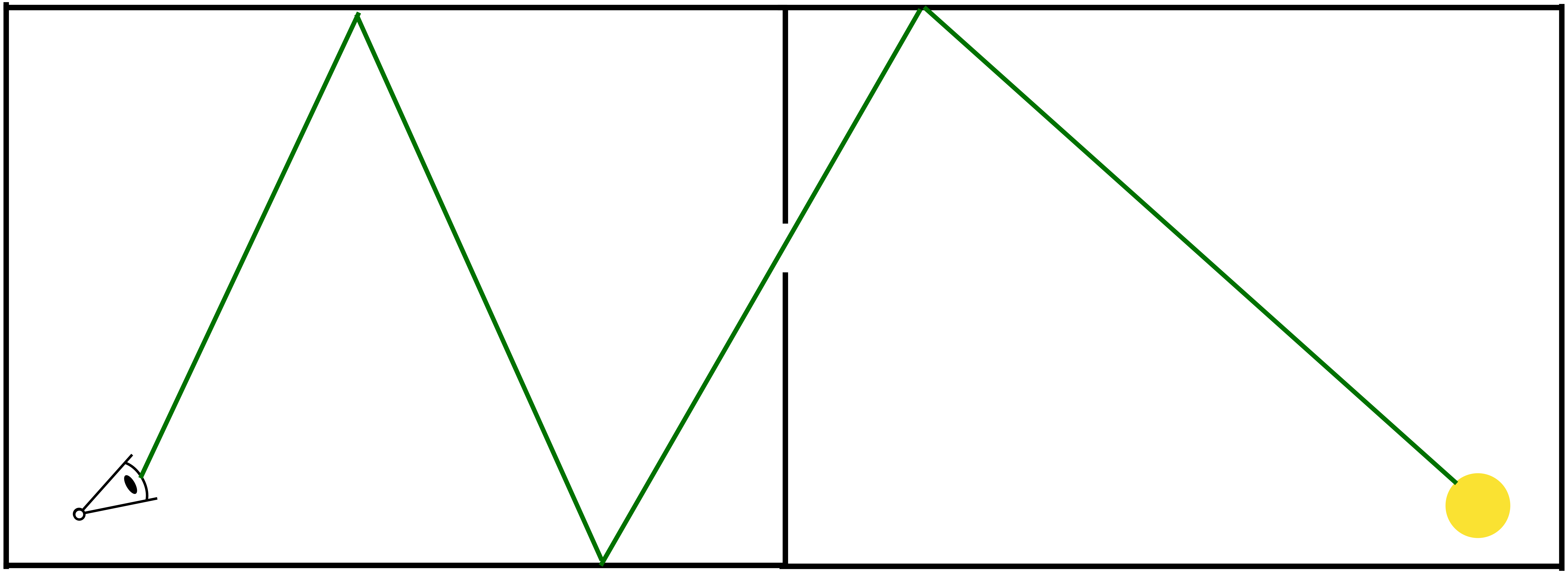
Metropolis Light Transport

UCSD CSE 272
Advanced Image Synthesis

Tzu-Mao Li

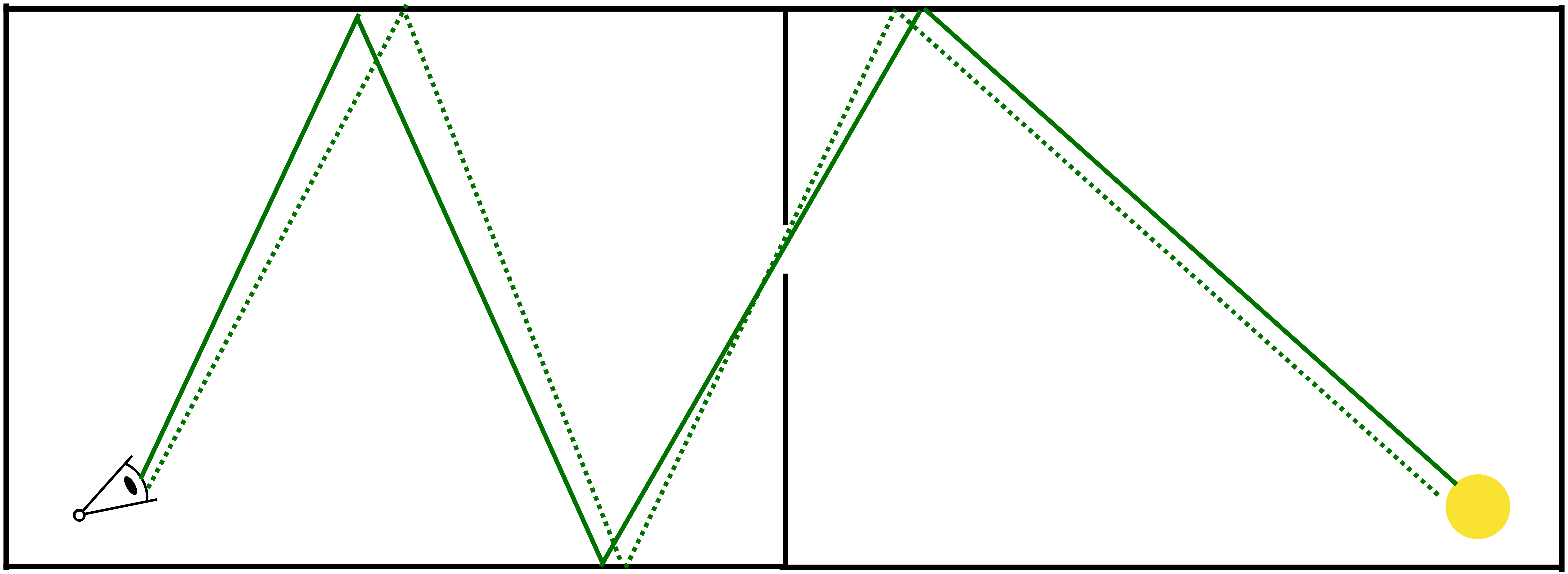
Light paths with difficult visibility

- bidirectional path tracing & photon mapping will both fail



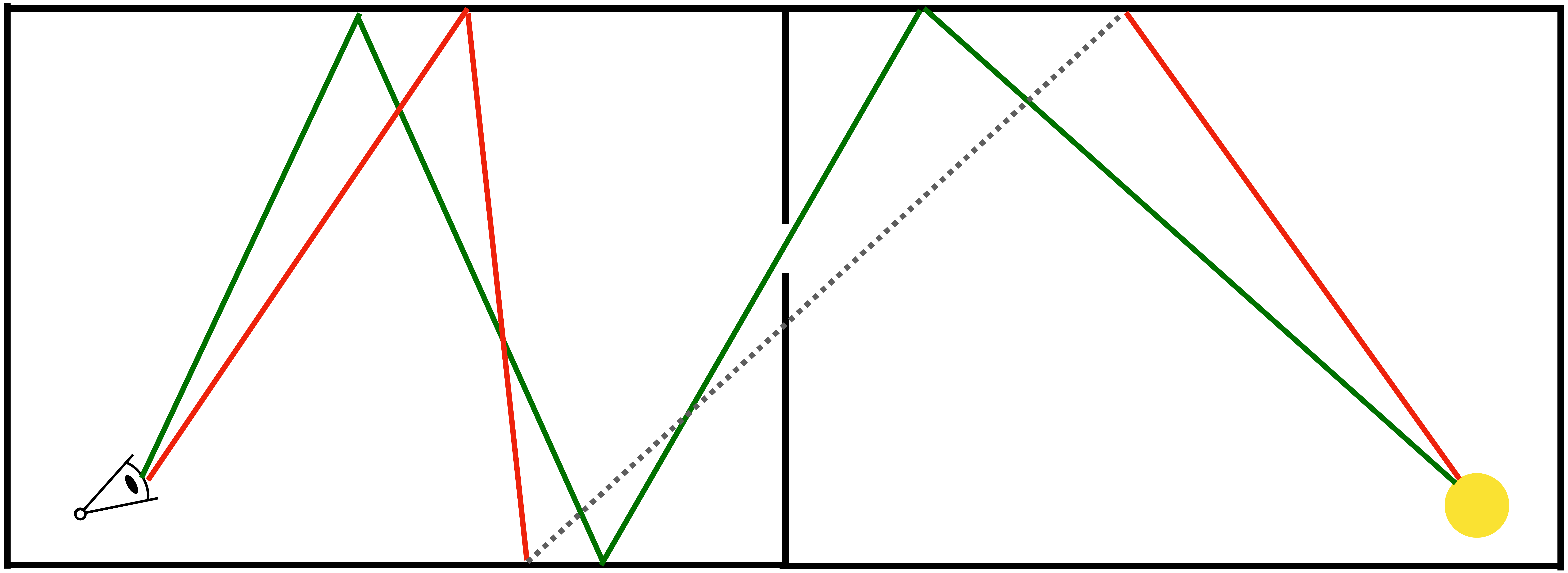
Idea: keep sampling in high-contribution regions by "mutating" light paths

aka Markov Chain Monte Carlo (MCMC) methods



Metropolis light transport [Veach 1997]

1.generate some "seed paths" using bidirectional path tracing, sample them based on their contribution



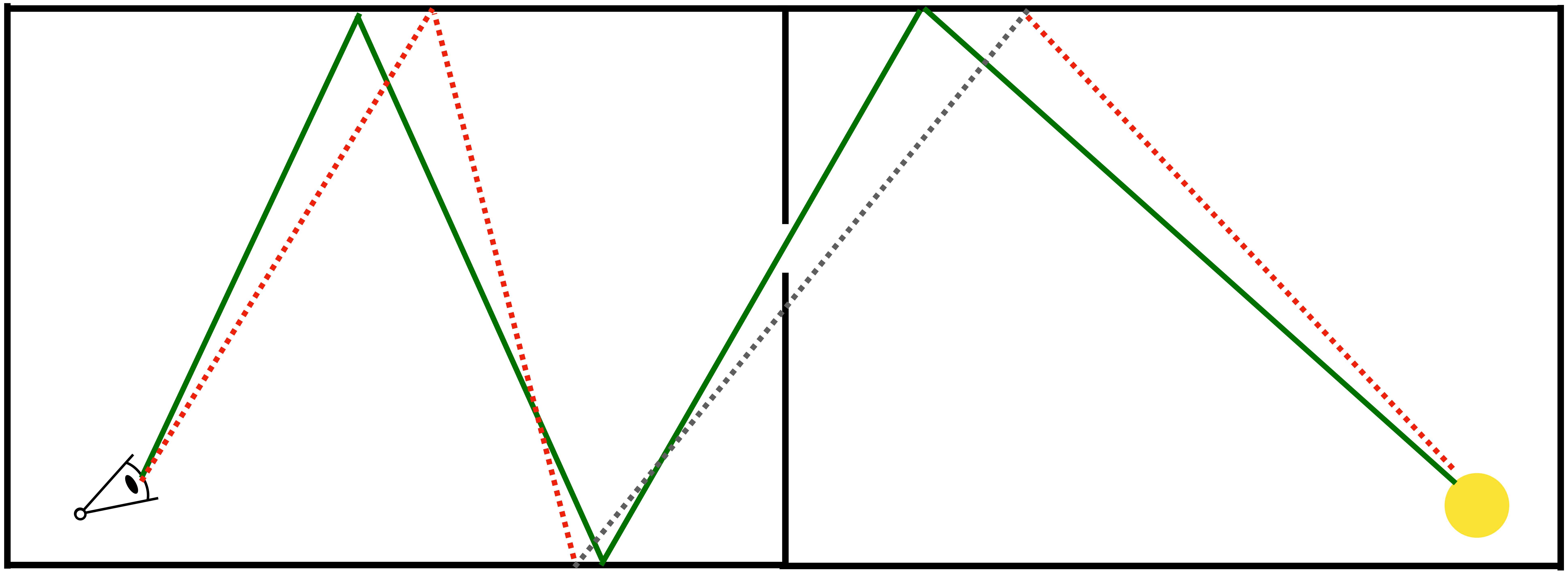
Metropolis Light Transport

Eric Veach *Leonidas J. Guibas*

Computer Science Department
Stanford University

Metropolis light transport [Veach 1997]

1. generate some “seed paths” using bidirectional path tracing, sample them based on their contribution
2. “mutate” the light path by changing it a little bit



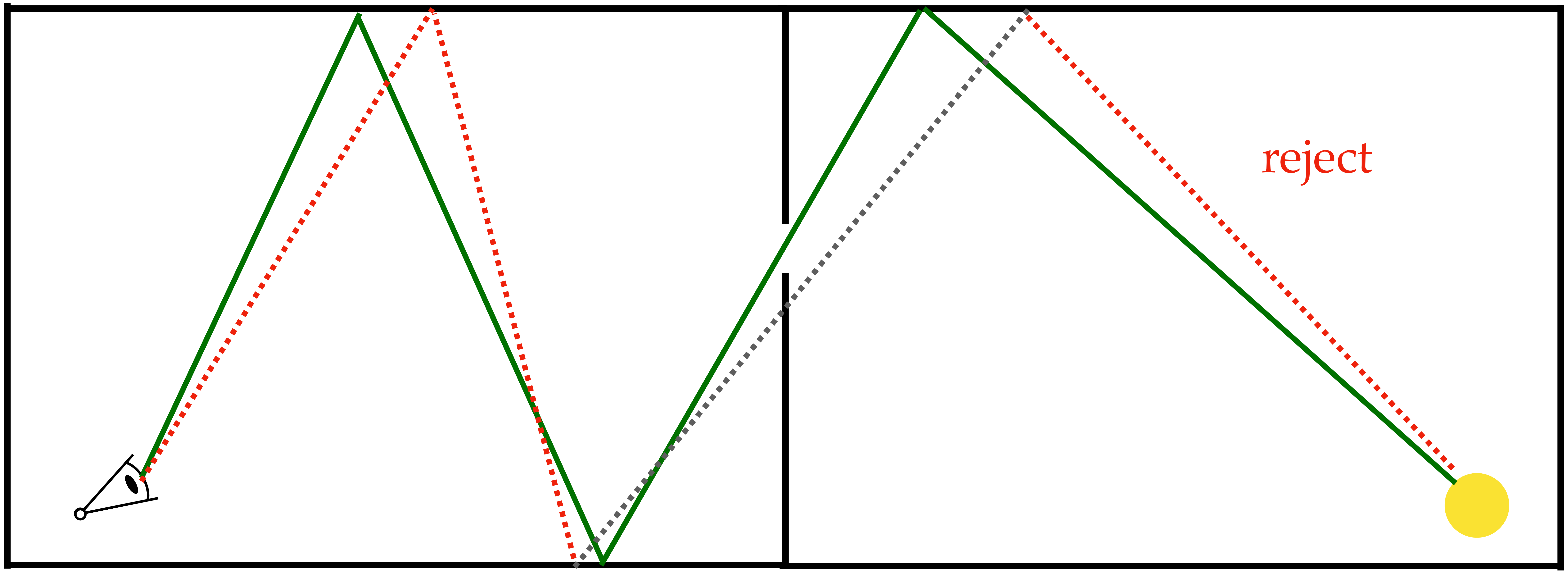
Metropolis Light Transport

Eric Veach *Leonidas J. Guibas*

Computer Science Department
Stanford University

Metropolis light transport [Veach 1997]

1. generate some “seed paths” using bidirectional path tracing, sample them based on their contribution
2. “mutate” the light path by changing it a little bit
3. probabilistically “accept” the new path based on its contribution
   if a path is accepted, make it the new seed path, else stay at the current path



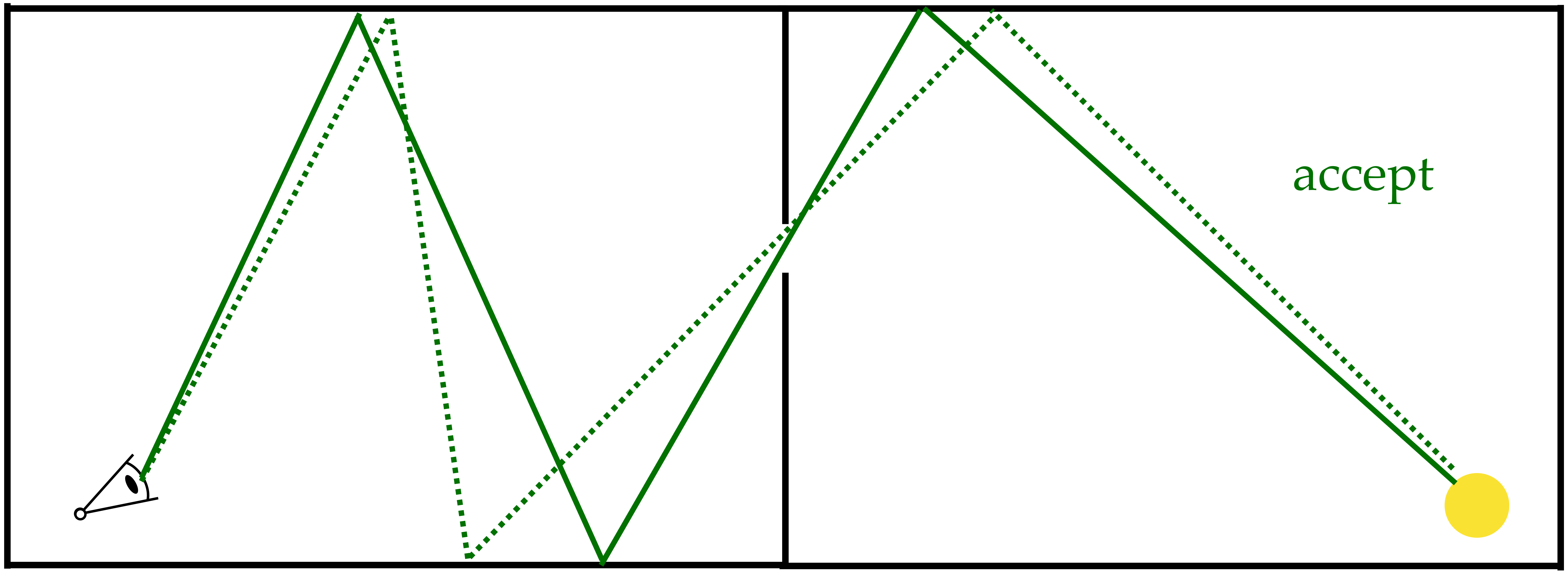
Metropolis Light Transport

Eric Veach *Leonidas J. Guibas*

Computer Science Department
Stanford University

Metropolis light transport [Veach 1997]

1.generate some "seed paths" using bidirectional path tracing, sample them based on their contribution
2."mutate" the light path by changing it a little bit
3. probabilistically "accept" the new path based on its contribution
   if a path is accepted, make it the new seed path, else stay at the current path

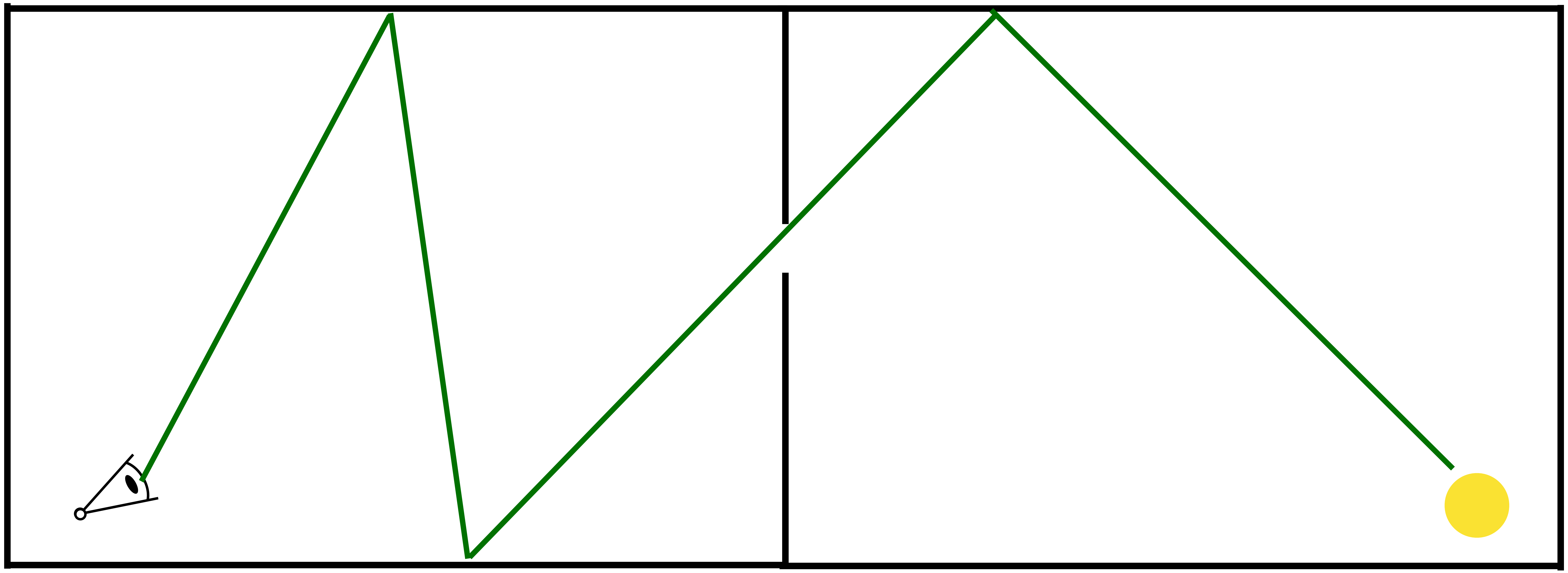


Metropolis Light Transport

Eric Veach *Leonidas J. Guibas*
Computer Science Department
Stanford University

Metropolis light transport [Veach 1997]

1.generate some “seed paths” using bidirectional path tracing, sample them based on their contribution
2.“mutate” the light path by changing it a little bit
3. probabilistically “accept” the new path based on its contribution
if a path is accepted, make it the new seed path, else stay at the current path



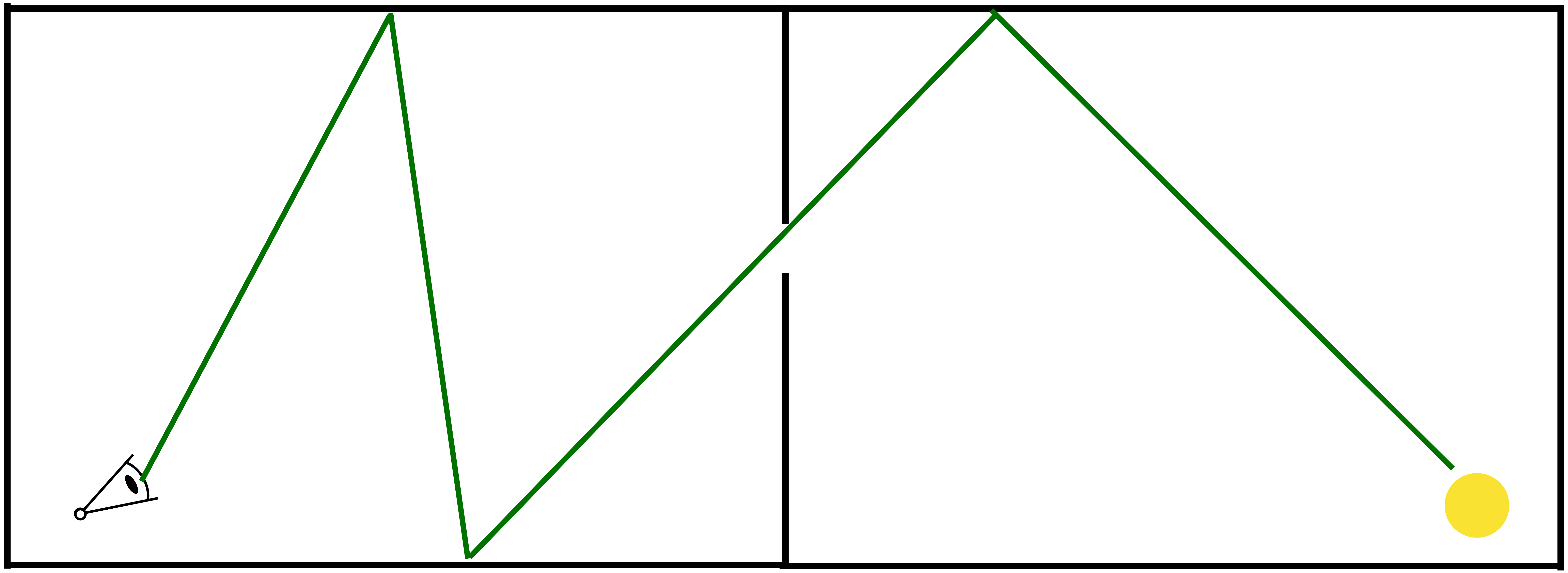
Metropolis Light Transport

Eric Veach *Leonidas J. Guibas*

Computer Science Department
Stanford University

Metropolis light transport [Veach 1997]

1.generate some "seed paths" using bidirectional path tracing, sample them based on their contribution
2."mutate" the light path by changing it a little bit
3. probabilistically "accept" the new path based on its contribution
   if a path is accepted, make it the new seed path, else stay at the current path
4. "+1" to the pixel correspond to the path (even if rejected), go to 2 until budget is met



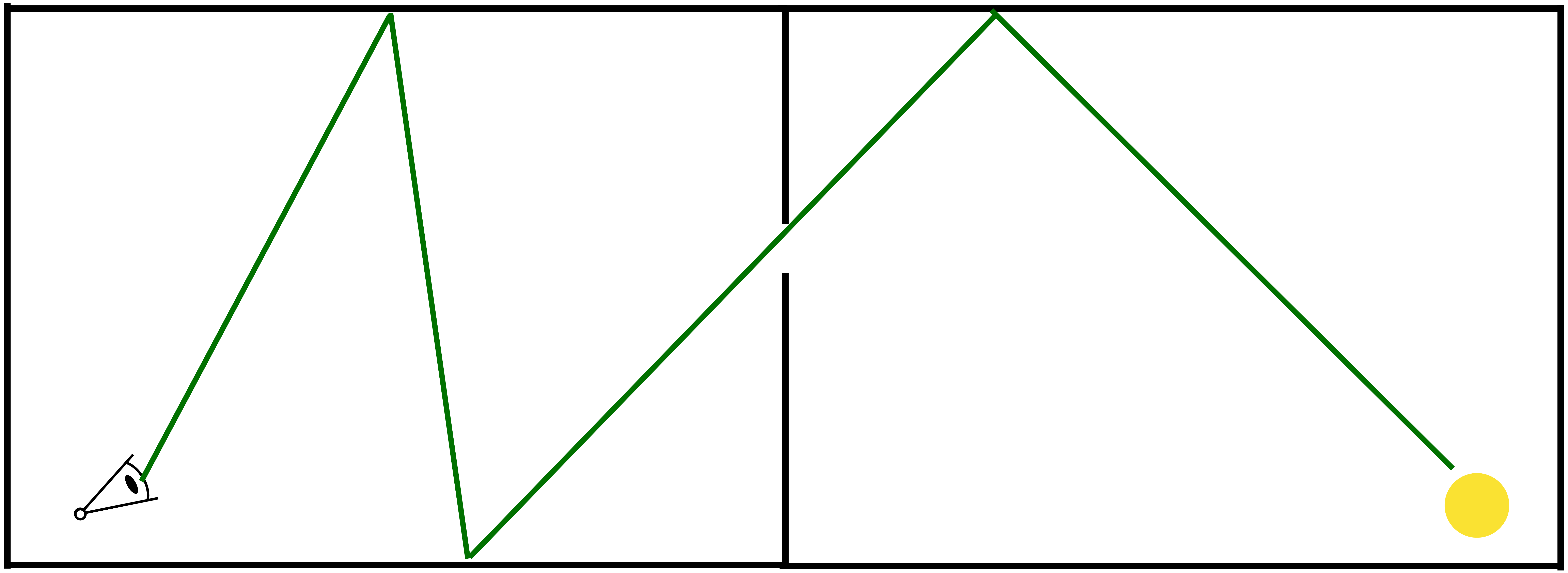
Metropolis Light Transport

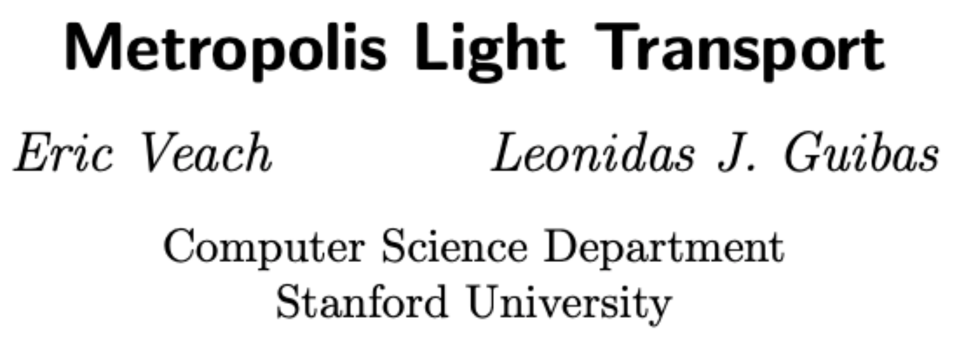
Eric Veach *Leonidas J. Guibas*

Computer Science Department
Stanford University

Metropolis light transport [Veach 1997]

1.generate some "seed paths" using bidirectional path tracing, sample them based on their contribution
2."mutate" the light path by changing it a little bit
3. probabilistically "accept" the new path based on its contribution
    if a path is accepted, make it the new seed path, else stay at the current path
4. "+1" to the pixel correspond to the path (even if rejected), go to 2 until budget is met
5. normalize the whole image by the average brightness estimated by bidirectional path tracing



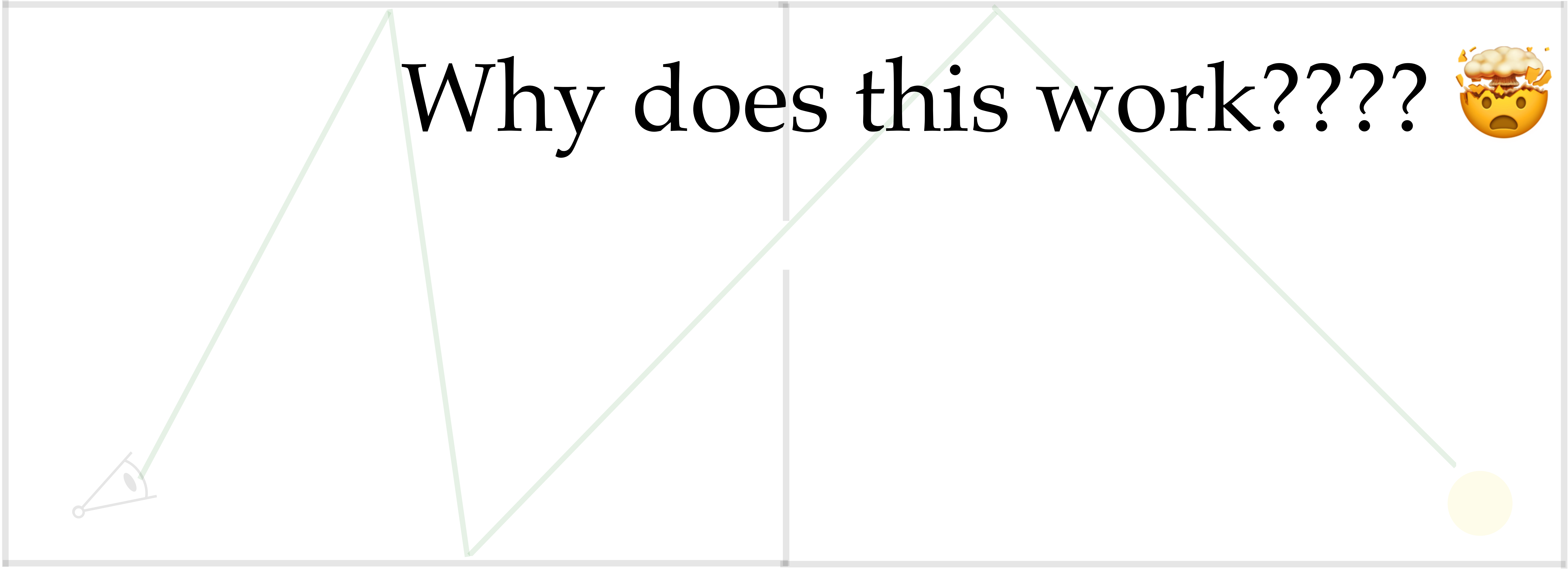

Metropolis Light Transport

Eric Veach *Leonidas J. Guibas*

Computer Science Department
Stanford University

Metropolis light transport [Veach 1997]

1. generate some "seed paths" using bidirectional path tracing, sample them based on their contribution
2. "mutate" the light path by changing it a little bit
3. probabilistically "accept" the new path based on its contribution
   if a path is accepted, make it the new seed path, else stay at the current path
4. "+1" to the pixel correspond to the path (even if rejected), go to 2 until budget is met
5. normalize the whole image by the average brightness estimated by bidirectional path tracing



Why does this work???? 🤯

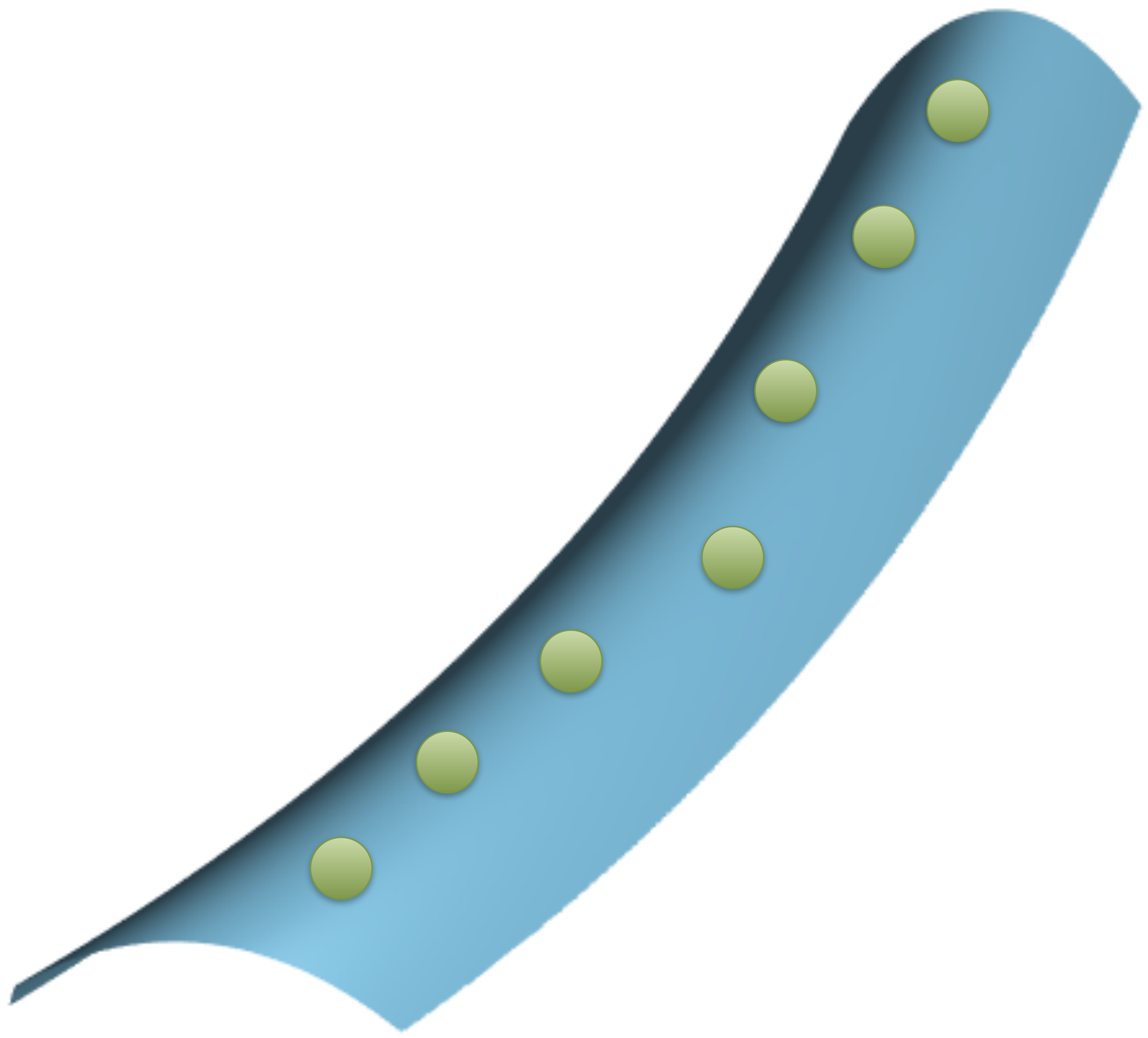
Metropolis Light Transport

Eric Veach *Leonidas J. Guibas*

Computer Science Department
Stanford University

Mathematical formulation

given the luminance of path contribution $f(\bar{x}) \in \mathbb{R}$ (the path $\bar{x}$ can land on any pixel),
want to sample $\bar{x}$ s.t. $p(\bar{x}) \propto f(\bar{x})$



Metropolis Light Transport

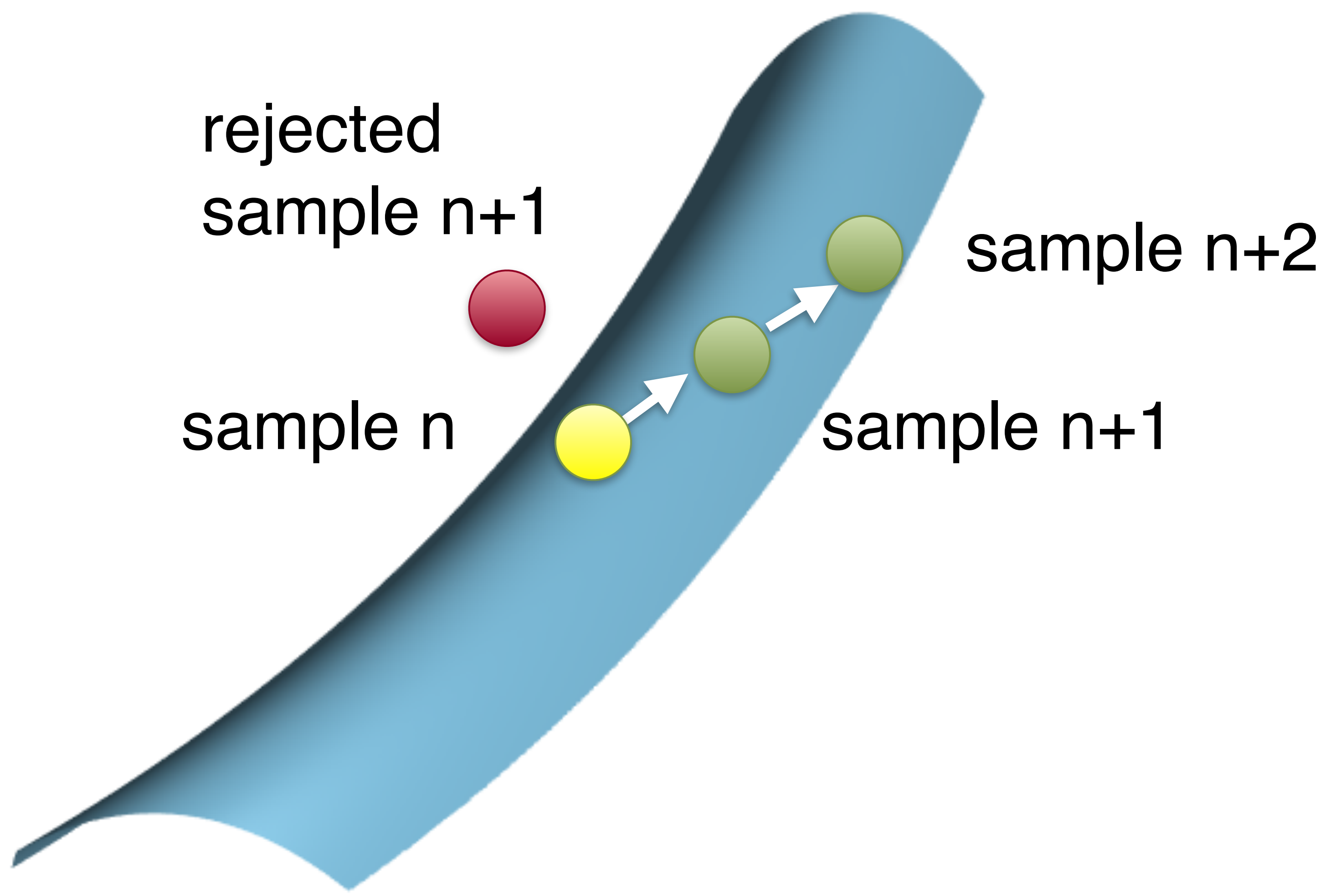
Eric Veach *Leonidas J. Guibas*

Computer Science Department
Stanford University

Metropolis-Hastings algorithm

given the luminance of path contribution $f(\bar{x}) \in \mathbb{R}$ (the path $\bar{x}$ can land on any pixel),
want to sample $\bar{x}$ s.t. $p(\bar{x}) \propto f(\bar{x})$

```
x = x0 // bidirectional path tracing
for i in range(n):
  x’ = mutate(x)
  a = min((f(x’)/f(x)) *
          (p_m(x’->x)/p_m(x->x’)), 1)
  if random() < a:
    x = x’
  record(image, x)
```



Equation of State Calculations by Fast Computing Machines

Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, and Augusta H. Teller,
Los Alamos Scientific Laboratory, Los Alamos, New Mexico

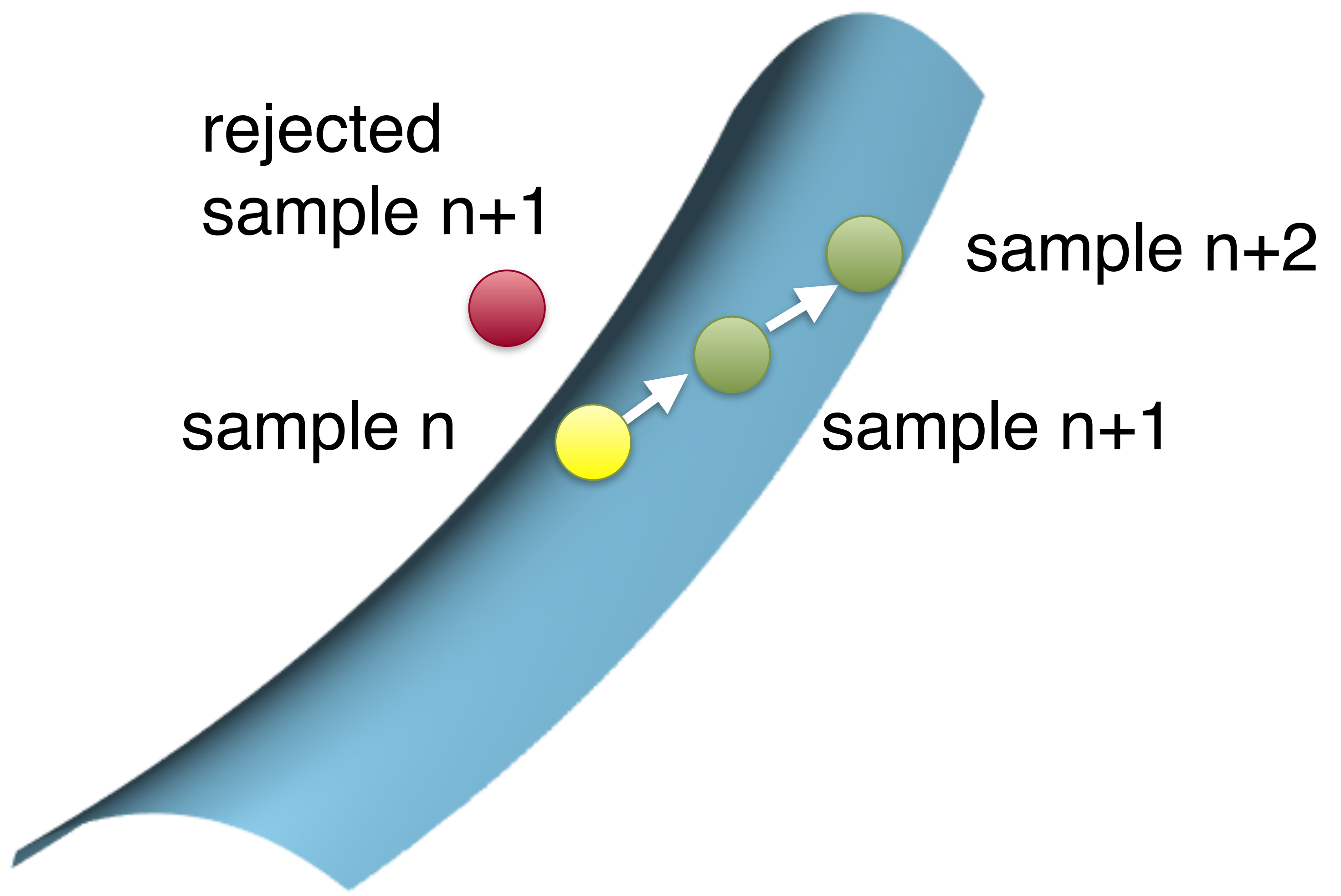
and

Edward Teller,* *Department of Physics, University of Chicago, Chicago, Illinois*
(Received March 6, 1953)

Metropolis-Hastings algorithm

given the luminance of path contribution $f(\bar{x}) \in \mathbb{R}$ (the path $\bar{x}$ can land on any pixel),
want to sample $\bar{x}$ s.t. $p(\bar{x}) \propto f(\bar{x})$

```
x = x0 // bidirectional path tracing
for i in range(n):
  x’ = mutate(x)
  a = min((f(x’)/f(x)) *
          (p_m(x’->x)/p_m(x->x’)), 1)
  if random() < a:
    x = x’
  record(image, x)
```



Metropolis: lab director
A. Rosenbluth: junior researcher
M. Rosenbluth: junior researcher’s husband
A. Teller: advisor’s wife
E. Teller: advisor

Equation of State Calculations by Fast Computing Machines

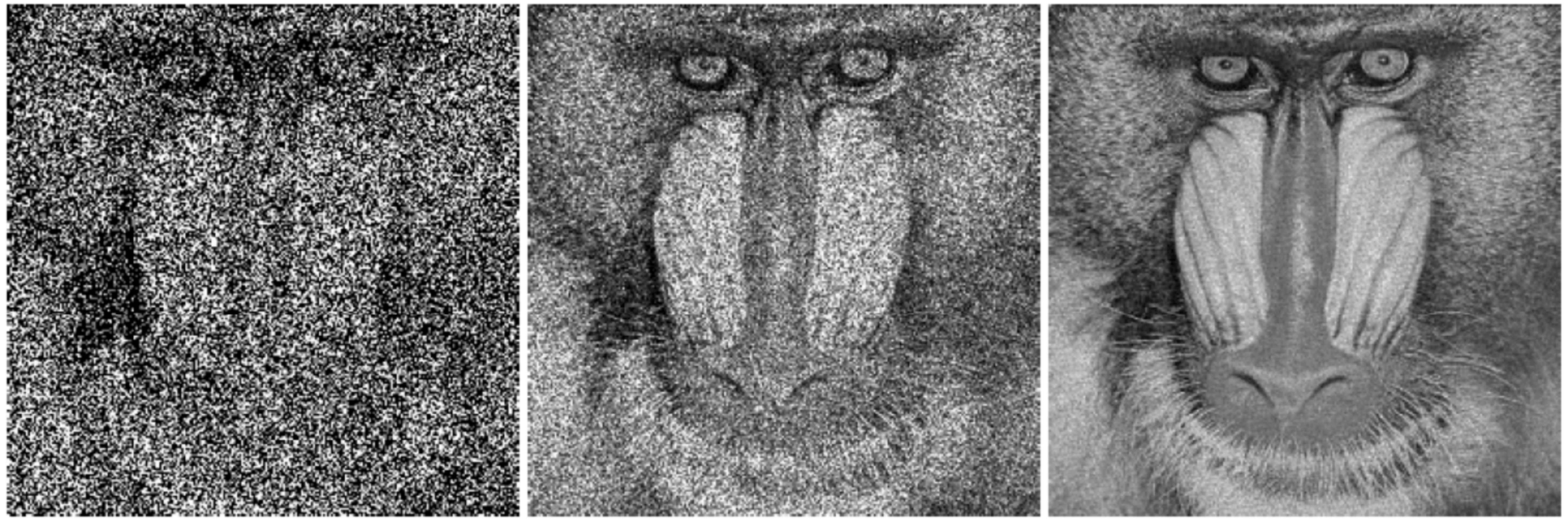
Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, and Augusta H. Teller,
Los Alamos Scientific Laboratory, Los Alamos, New Mexico

and

Edward Teller,* *Department of Physics, University of Chicago, Chicago, Illinois*
(Received March 6, 1953)

2D image copy example

```
x = x0
for i in range(n):
  x’ = mutate(x)
  a = min((f(x’)/f(x)) *
          (p_m(x’->x)/p_m(x->x’)), 1)
  if random() < a:
    x = x’
  record(image, x’)
```



1 sample per pixel | 8 samples per pixel | 256 samples per pixel

https://www.csie.ntu.edu.tw/~cyy/courses/rendering/16fall/lectures/handouts/chap13_mc.pdf

Why does Metropolis algorithm work?

- easier to think in the discrete state space: assume our path space lives on an integer domain
  - a “path” x is, for now, an integer
- we start with some (discrete) PDF $\pi^0(x)$, defined by bidirectional path tracing

```
x = x0
for i in range(n):
  x’ = mutate(x)
  a = min((f(x’)/f(x)) *
          (p_m(x’->x)/p_m(x->x’)), 1)
  if random() < a:
    x = x’
  record(image, x’)
```

Why does Metropolis algorithm work?

- easier to think in the discrete state space: assume our path space lives on an integer domain
  - a “path” x is, for now, an integer
- we start with some (discrete) PDF $\pi^0(x)$, defined by bidirectional path tracing
- each mutation/acceptance changes the PDF:
  - $K\pi^t = \pi^{t+1}$, K_{ij} = probability to go from i to j
- want to prove that $\lim_{t\to\infty} \pi^t \propto f$

```
x = x0
for i in range(n):
  x’ = mutate(x)
  a = min((f(x’)/f(x)) *
          (p_m(x’->x)/p_m(x->x’)), 1)
  if random() < a:
    x = x’
  record(image, x’)
```

Why does Metropolis algorithm work?

- when t goes to infinity, the mutation update $K\pi^t = \pi^{t+1}$ reaches a fixed point $K\pi = \pi$
  - with assumption that the mutation is “ergodic” — it should have non-zero probability to visit all states

```
x = x0
for i in range(n):
  x’ = mutate(x)
  a = min((f(x’)/f(x)) *
          (p_m(x’->x)/p_m(x->x’)), 1)
  if random() < a:
    x = x’
  record(image, x’)
```

Why does Metropolis algorithm work?

- when t goes to infinity, the mutation update $K\pi^t = \pi^{t+1}$ reaches a fixed point $K\pi = \pi$
  - with assumption that the mutation is "ergodic" — it should have non-zero probability to visit all states
- Theorem: if a kernel K satisfies the **detailed balance condition**:
  - $K_{ij}\pi_i = K_{ji}\pi_j \forall i, j$
- then, starting from any distribution π^0, K has a unique fixed point π (usually called the stationary distribution)
  - exercise: prove it!

```
x = x0
for i in range(n):
  x' = mutate(x)
  a = min((f(x')/f(x)) *
          (p_m(x'->x)/p_m(x->x')), 1)
  if random() < a:
    x = x'
  record(image, x')
```

Why does Metropolis algorithm work?

- goal: design a such that $K_{ij} f_i = K_{ji} f_j$

$$K_{ij} = \begin{cases} p_{\mathrm{m}}(i \to j) a(i \to j) & \text{if } i \neq j \\ p_{\mathrm{m}}(i \to i) a(i \to i) + \sum_{j \neq i} p_{\mathrm{m}}(i \to j)(1 - a(i \to j)) & \text{if } i = j \end{cases}$$

```
x = x0
for i in range(n):
  x’ = mutate(x)
  a = min((f(x’)/f(x)) *
          (p_m(x’->x)/p_m(x->x’)), 1)
  if random() < a:
    x = x’
  record(image, x’)
```

Why does Metropolis algorithm work?

- goal: design a such that $K_{ij} f_i = K_{ji} f_j$

$$K_{ij} = \begin{cases} p_{\mathrm{m}}(i \to j) a(i \to j) & \text{if } i \neq j \\ p_{\mathrm{m}}(i \to i) a(i \to i) + \sum_{j \neq i} p_{\mathrm{m}}(i \to j)(1 - a(i \to j)) & \text{if } i = j \end{cases}$$

$$\text{if } a\left(i \to j\right) = \min\left(\frac{f_j}{f_i}\frac{p_m(j \to i)}{p_m(i \to j)}, 1\right),$$

K satisfies detailed balance

```
x = x0
for i in range(n):
  x' = mutate(x)
  a = min((f(x')/f(x)) *
          (p_m(x'->x)/p_m(x->x')), 1)
  if random() < a:
    x = x'
  record(image, x')
```

Why does Metropolis algorithm work?

- goal: design a such that $K_{ij} f_i = K_{ji} f_j$

$$K_{ij} = \begin{cases} p_{\mathrm{m}}(i \to j) a(i \to j) & \text{if } i \neq j \\ p_{\mathrm{m}}(i \to i) a(i \to i) + \sum_{j \neq i} p_{\mathrm{m}}(i \to j)(1 - a(i \to j)) & \text{if } i = j \end{cases}$$

$$\text{if } a(i \to j) = \min\left(\frac{f_j}{f_i} \frac{p_m(j \to i)}{p_m(i \to j)}, 1\right),$$

K satisfies detailed balance

```
x = x0
for i in range(n):
  x' = mutate(x)
  a = min((f(x')/f(x)) *
          (p_m(x'->x)/p_m(x->x')), 1)
  if random() < a:
    x = x'
  record(image, x')
```

What should the record function do?

```
x = x0
for i in range(n):
  x’ = mutate(x)
  a = min((f(x’)/f(x)) *
          (p_m(x’->x)/p_m(x->x’)), 1)
  if random() < a:
    x = x’
  record(image, x)
```

- since $\pi(x) \propto f$ in the limit, $\dfrac{f(x)}{\pi(x)} = \text{constant}$
- estimate the constant across image using bidirectional path tracing (average brightness of the image)
- add the constant divided by the number of samples to the corresponding pixel
- Metropolis light transport is recording image histogram!

Making MLT unbiased

```
x = x0
for i in range(n):
  x’ = mutate(x)
  a = min((f(x’)/f(x)) *
          (p_m(x’->x)/p_m(x->x’)), 1)
  if random() < a:
    x = x’
  record(image, x)
```

the sampling distribution π^t
only converges to f in the limit,
so naive MLT is biased

Making MLT unbiased

```
x = x0
for i in range(n):
  x’ = mutate(x)
  a = min((f(x’)/f(x)) *
          (p_m(x’->x)/p_m(x->x’)), 1)
  if random() < a:
    x = x’
  record(image, x)
```

the sampling distribution π^t
only converges to f in the limit,
so naive MLT is biased

solution: weigh all samples with $\frac{f(x_0)}{p(x_0)}$ where
p is BDPT sampling density

Making MLT unbiased

- intuition: bidirectional path tracing is unbiased, each mutation is preserving the unbiasedness using detailed balance

- see Veach's thesis for proof

Appendix 11.A Proof of Unbiased Initialization

In this appendix, we show that the estimate

$$I_j \;=\; E\left[\frac{1}{N}\sum_{i=1}^{N} W_i\, h_j(\bar{X}_i)\right]$$

is unbiased (see Section 11.3.1). To do this, we show that the following*weighted equilibrium condition* is satisfied at each step of the random walk:

$$\int_{\mathbb{R}} w\, p_i(w, \bar{x})\, dw \;=\; f(\bar{x})\,, \qquad (11.14)$$

where p_i is the joint density function of the$i$-th weighted sample $(W_i, \bar{X}_i)$. This is a sufficient condition for the above estimate to be unbiased, since

$$\begin{aligned} E\Big[W_i\, h_j(\bar{X}_i)\Big] &= \int_{\Omega}\int_{\mathbb{R}} w\, h_j(\bar{x})\, p_i(w, \bar{x})\, dw\, d\mu(\bar{x}) \\ &= \int_{\Omega} h_j(\bar{x})\, f(\bar{x})\, d\mu(\bar{x}) \\ &= I_j\,. \end{aligned}$$

Metropolis light transport with a single Markov chain is unbiased but **NOT** consistent

- in practice, just average over many Markov chains

Five Common Misconceptions about Bias in Light Transport Simulation

Toshiya Hachisuka

Aarhus University

3.5. Markov chain algorithms are unbiased and consistent

Misconception: Throughout the literature, it is well recognized that the original Markov chain Monte Carlo method is biased and consistent. The reason is that the distribution of samples converges to the target distribution for infinitely long Markov chains by definition. The difference between the initial distribution and the target distribution is called start-up bias. Veach proposed to eliminate start-up bias in order to make Metropolis light transport (MLT) unbiased. The misconception is that this technique makes MLT unbiased and consistent.

Metropolis light transport with a single Markov chain is unbiased but **NOT** consistent

- in practice, just average over many Markov chains

Five Common Misconceptions about Bias in Light Transport Simulation

Toshiya Hachisuka

Aarhus University

MLT with many short Markov chains

Energy Redistribution Path Tracing

David Cline Justin Talbot Parris Egbert *

Brigham Young University

3.5. Markov chain algorithms are unbiased and consistent

Misconception: Throughout the literature, it is well recognized that the original Markov chain Monte Carlo method is biased and consistent. The reason is that the distribution of samples converges to the target distribution for infinitely long Markov chains by definition. The difference between the initial distribution and the target distribution is called start-up bias. Veach proposed to eliminate start-up bias in order to make Metropolis light transport (MLT) unbiased. The misconception is that this technique makes MLT unbiased and consistent.

MLT is very different from path tracing

```
x = x0
for i in range(n):
  x' = mutate(x)
  a = min((f(x')/f(x)) *
          (p_m(x'->x)/p_m(x->x')), 1)
  if random() < a:
    x = x'
  record(image, x)
```

quiz: if we only have one pixel, would MLT be helpful?

MLT is very different from path tracing

```
x = x0
for i in range(n):
  x' = mutate(x)
  a = min((f(x')/f(x)) *
          (p_m(x'->x)/p_m(x->x')), 1)
  if random() < a:
    x = x'
  record(image, x)
```

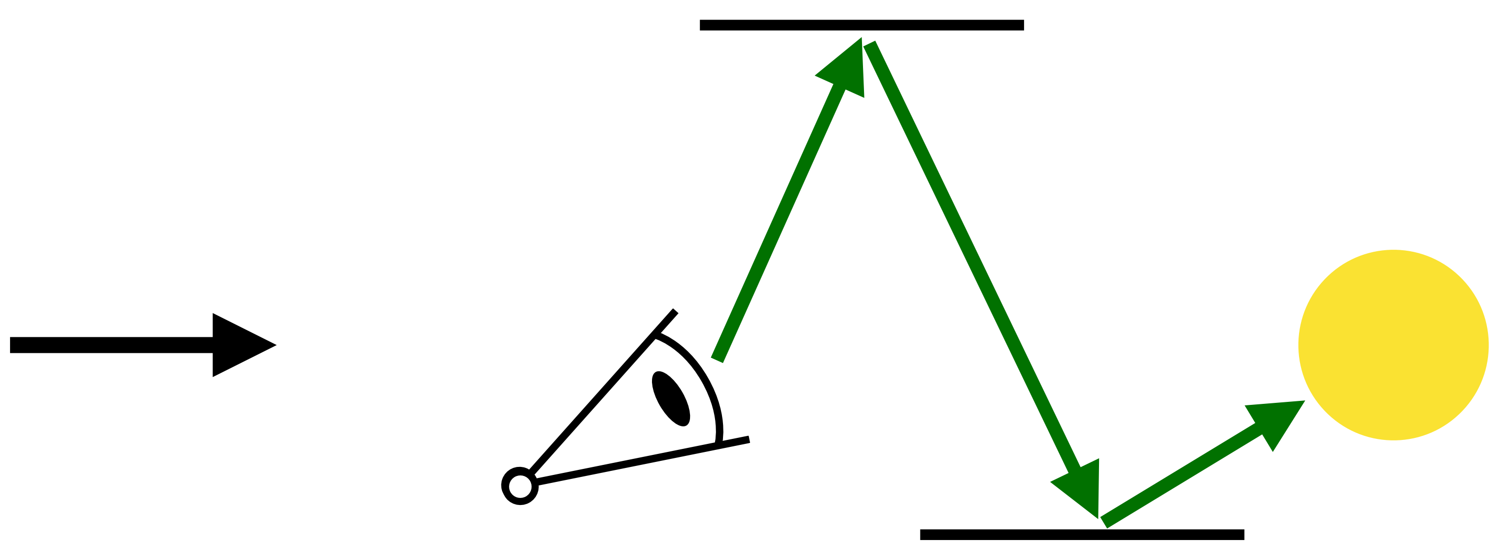
quiz: if we only have one pixel, would MLT be helpful?

- since $\pi(x) \propto f$ in the limit, $\frac{f(x)}{\pi(x)} = \text{constant}$
- but we have to estimate the constant, so MLT is not helpful!

Mutation: Kelemen-style

- simple to implement, less efficient than more sophisticated mutation
- idea: do the mutation in the **random number space**

$$u_0$$
$$u_1$$
$$u_2$$
$$\vdots$$



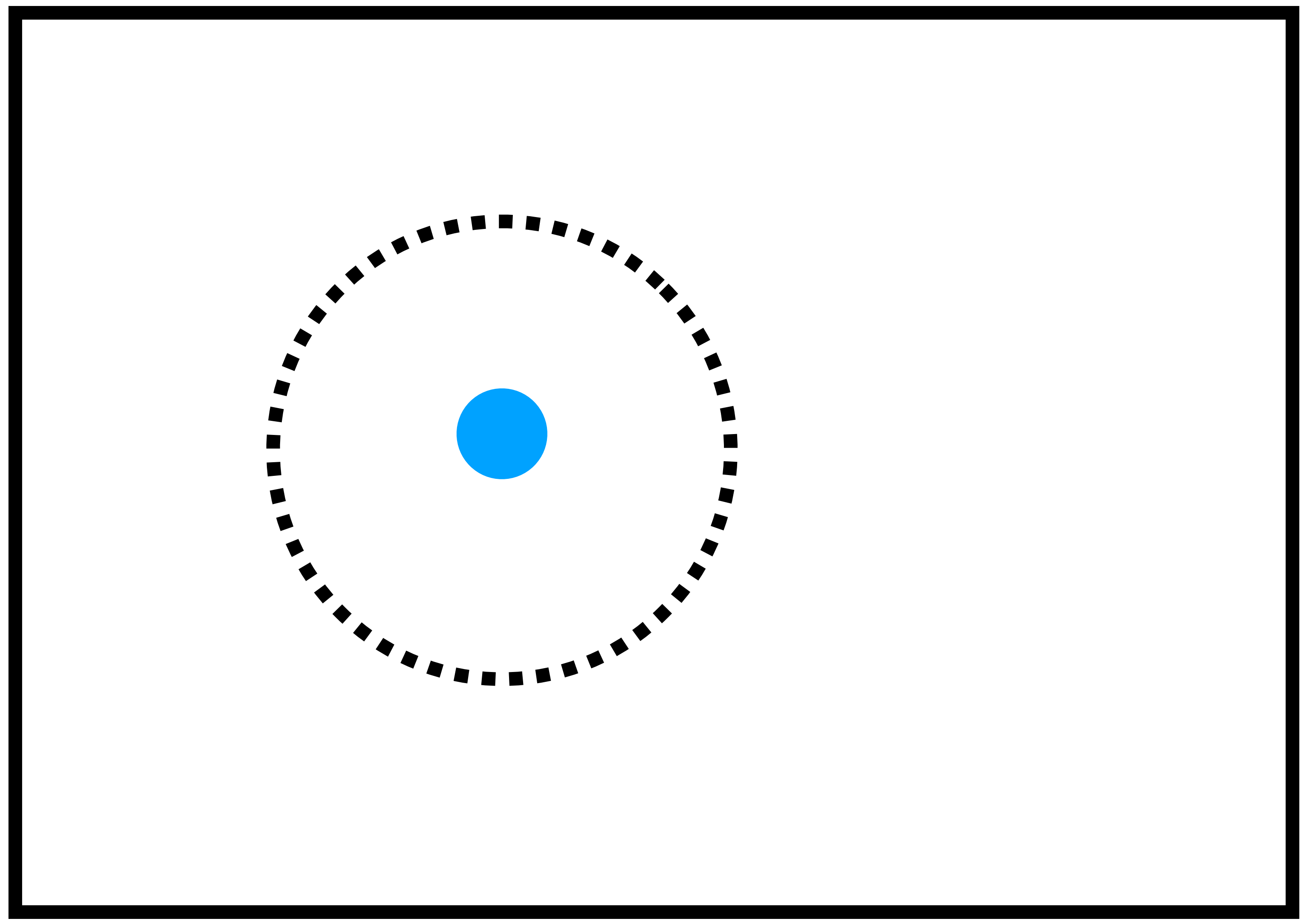
Simple and Robust Mutation Strategy for Metropolis Light Transport Algorithm

Csaba Kelemen and László Szirmay-Kalos

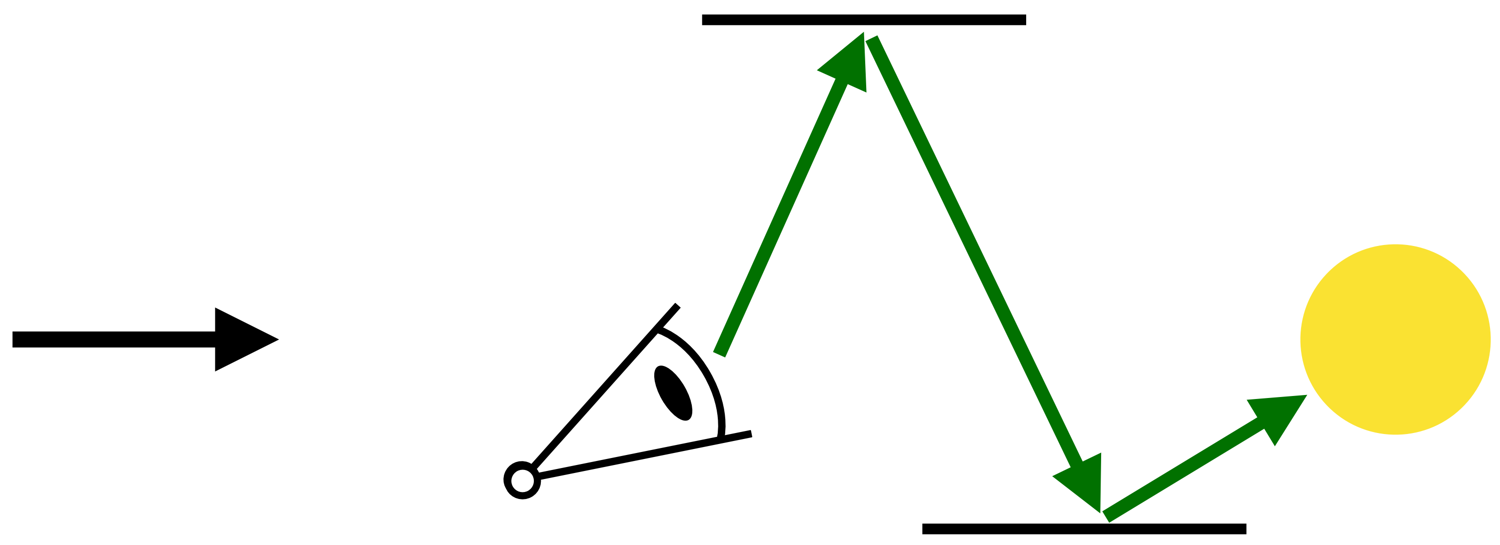
Department of Control Engineering and Information Technology, Technical University of Budapest
Budapest, Magyar Tudósok krt. 2, H-1117, HUNGARY
Email: szirmay@iit.bme.hu

Mutation: Kelemen-style

- randomly choose among two kinds of mutations:
  - large steps: forget about the current path, regenerate a path using bidirectional path tracing
  - small steps: a Gaussian-like distribution in the random number space



$$u_0$$
$$u_1$$
$$u_2$$
$$\vdots$$



Mutation: Kelemen-style

- code walkthrough
  - https://cs.uwaterloo.ca/~thachisu/smallpssmlt.cpp

Mutation size trade-off

- small mutation size: high accept rate, but introduce correlation between pixels
- large mutation size: better exploration and better noise, but low accept rate
- in practice: adapt mutation size to keep acceptance rate at a constant
  (aka adaptive MCMC)

(a) $\sigma^2 = 0.028$ *accept rate* 28.96%

(b) $\sigma^2 = 0.007$ *accept rate* 54.02%

(c) $\sigma^2 = 0.001$ *accept rate* 82.11%

Mutation: Veach-style

- randomly choose among 5 mutation strategies:
  - bidirectional mutation (similar to large steps but more complex)
  - lens perturbation
  - caustic perturbation
  - multi-chain perturbation
  - lens mutation (complex but not very useful)

Bidirectional mutation

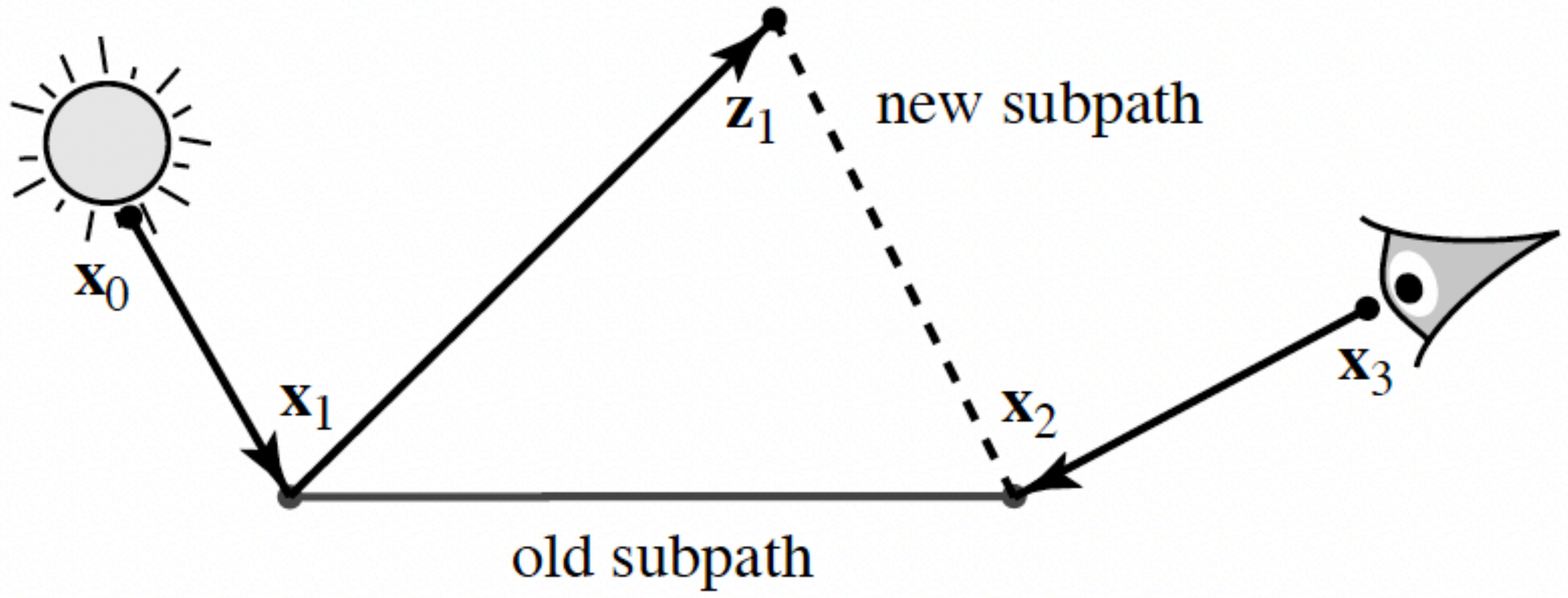
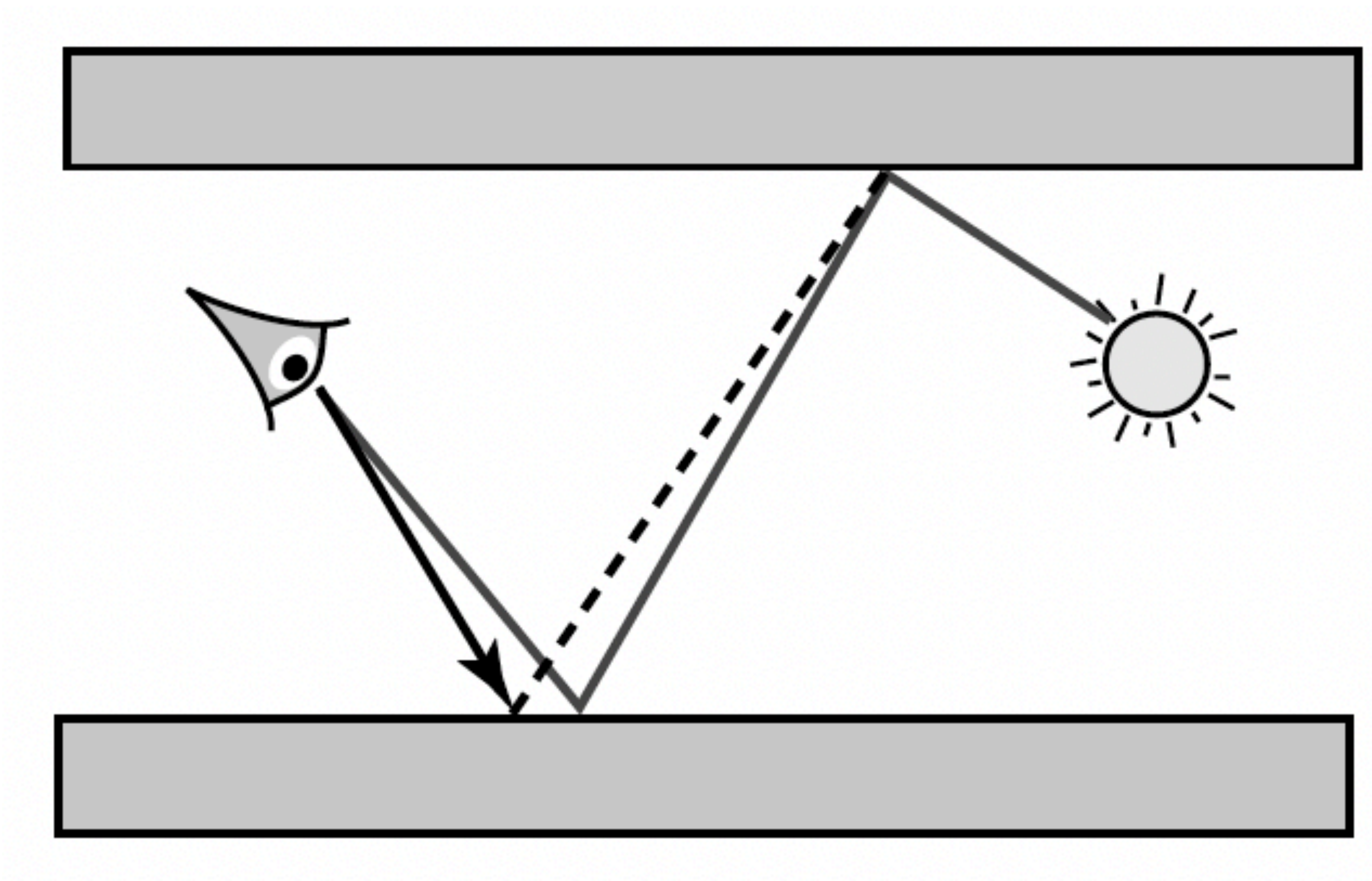


Figure 11.3: A simple example of a bidirectional mutation. The original path $\bar{x} = \mathbf{x}_0\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3$ is modified by deleting the edge $\mathbf{x}_1\mathbf{x}_2$ and replacing it with a new vertex $\mathbf{z}_1$. The new vertex is generated by sampling a direction at $\mathbf{x}_1$ (according to the BSDF) and casting a ray. This yields a mutated path $\bar{y} = \mathbf{x}_0\mathbf{x}_1\mathbf{z}_1\mathbf{x}_2\mathbf{x}_3$.

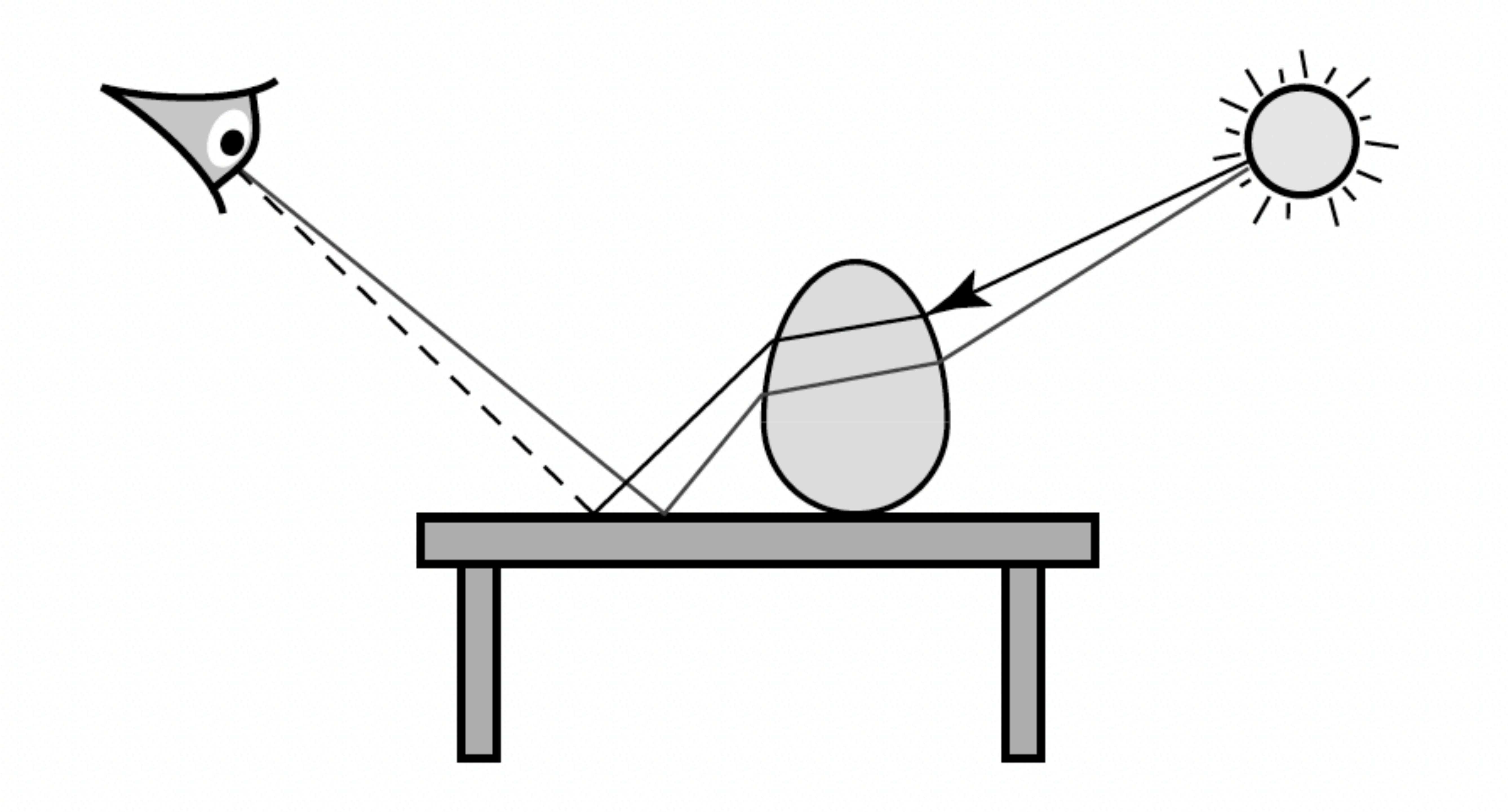
see my code here : >
https://github.com/aekul/yotsuba/blob/master/src/integrators/myintegrators/bidirmutation.cpp

Lens perturbation



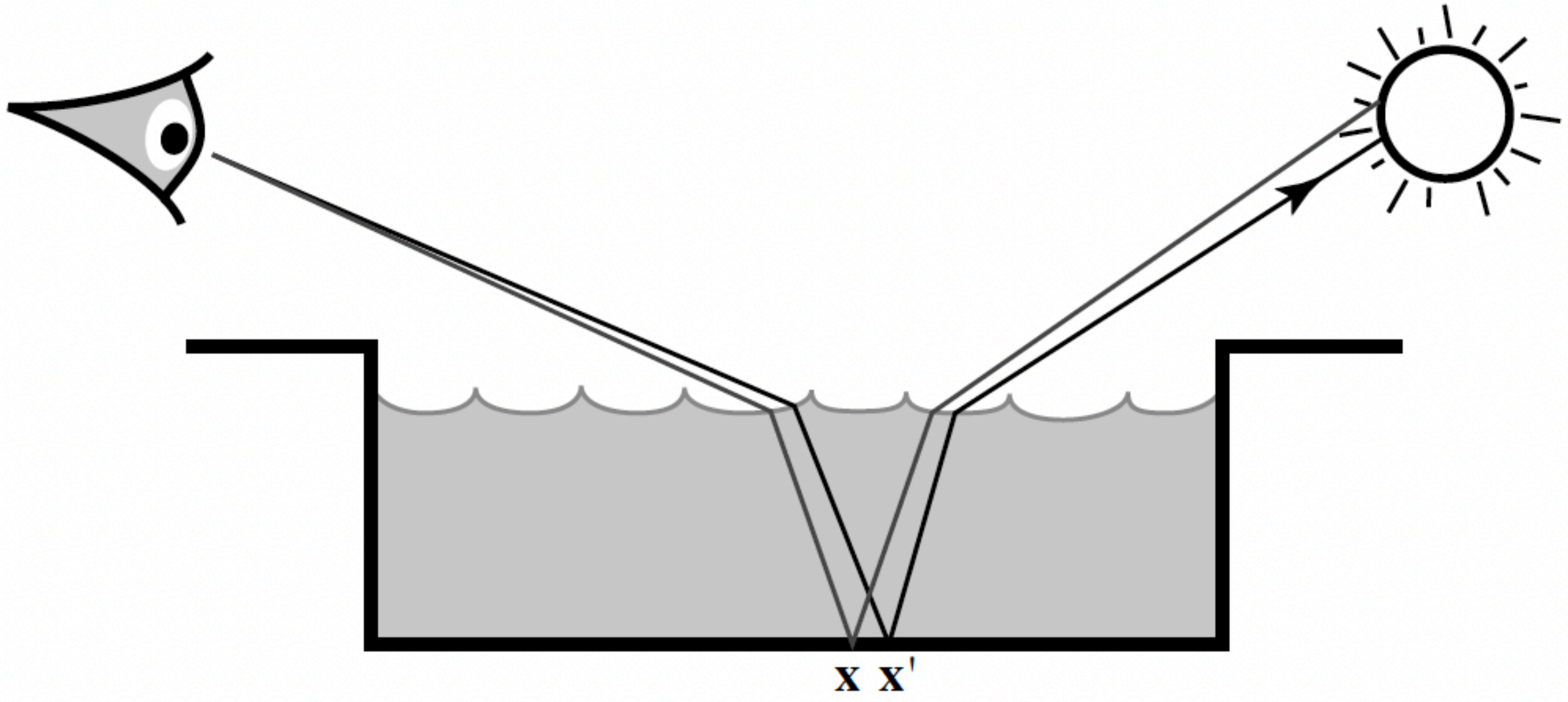
propagate the change through specular vertices

Caustics perturbation



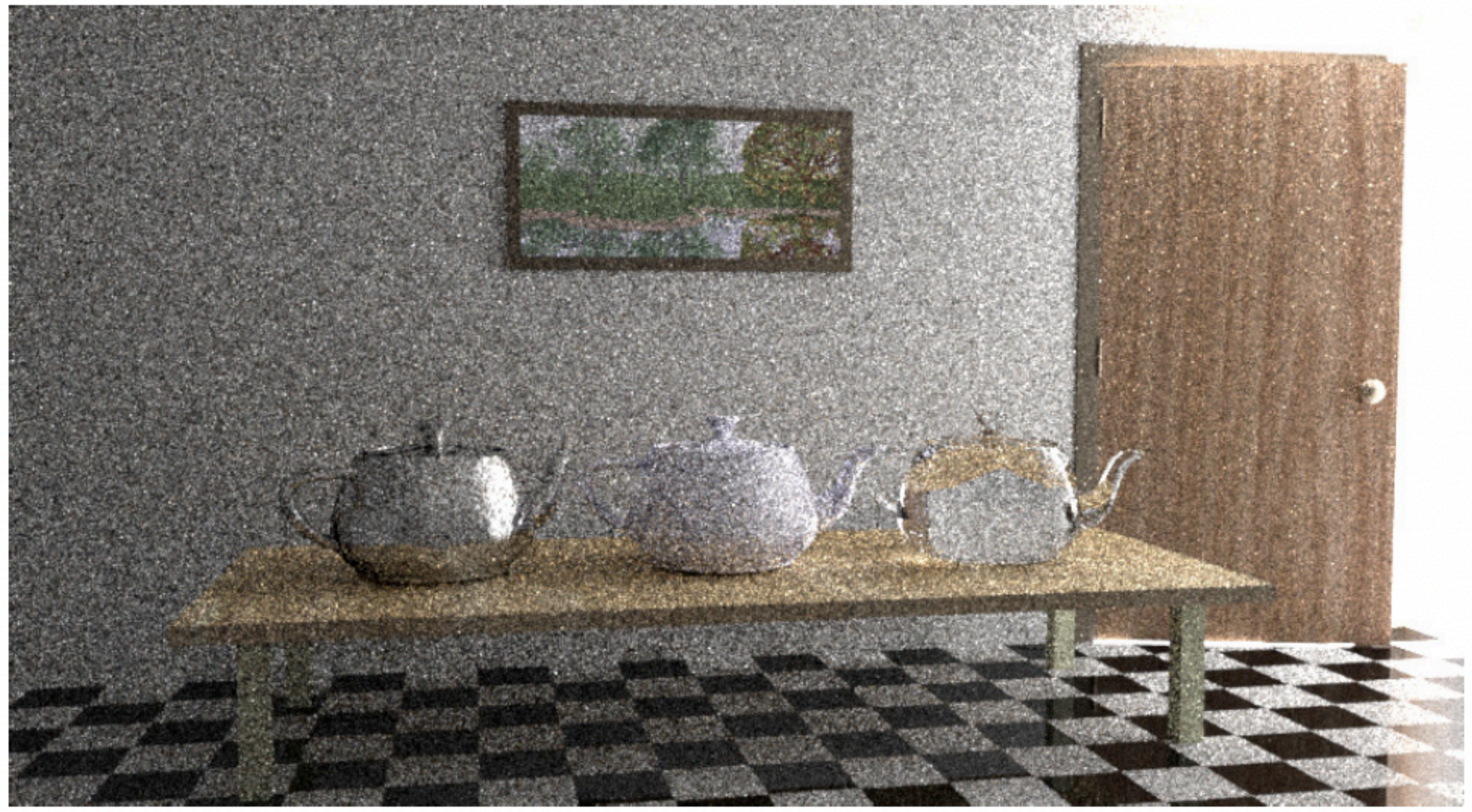
propagate the change through specular vertices

Multi-chain perturbation



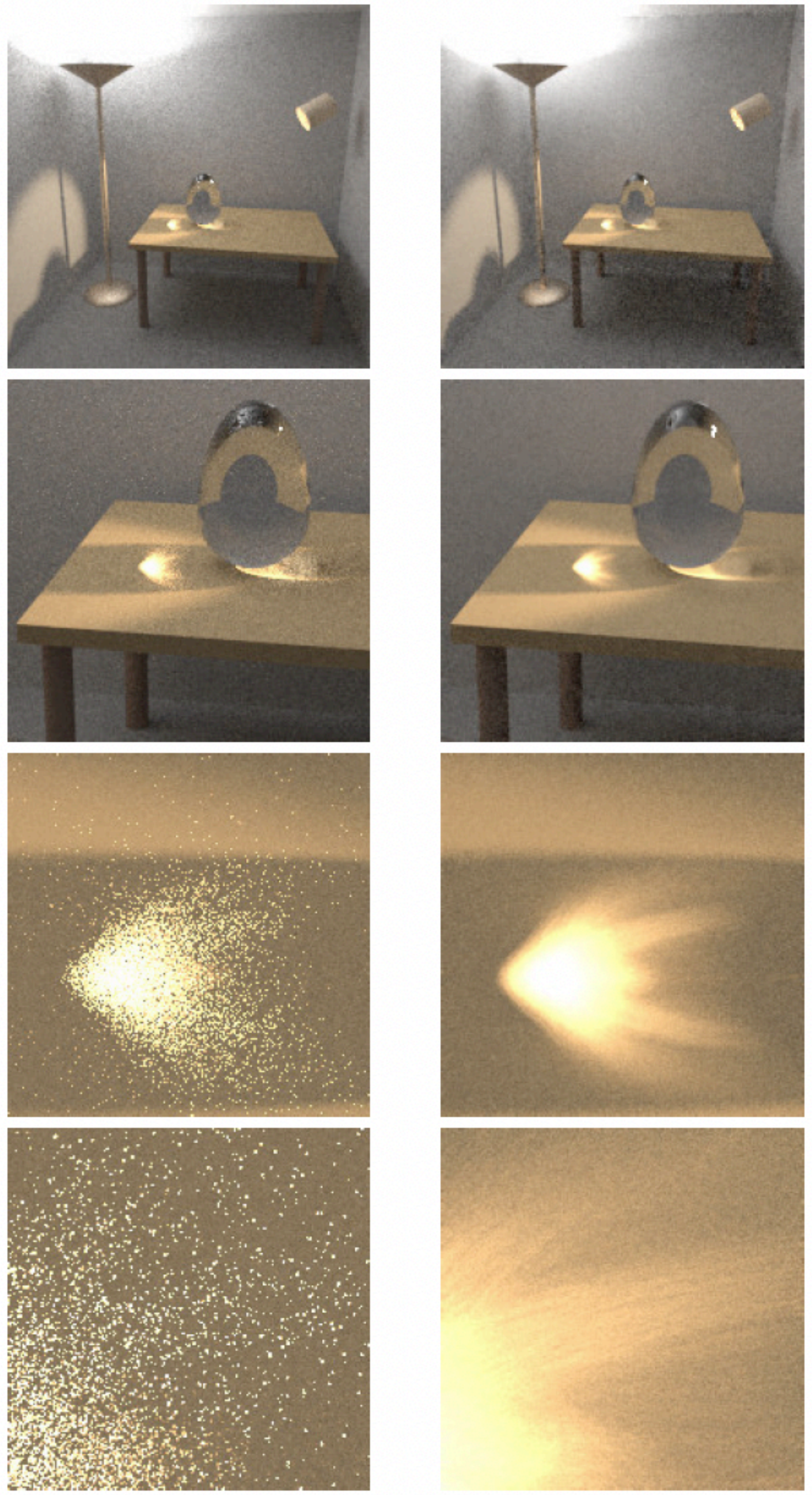
propagate the change through specular vertices
crucial for SDS paths

MLT is good at complex scenes

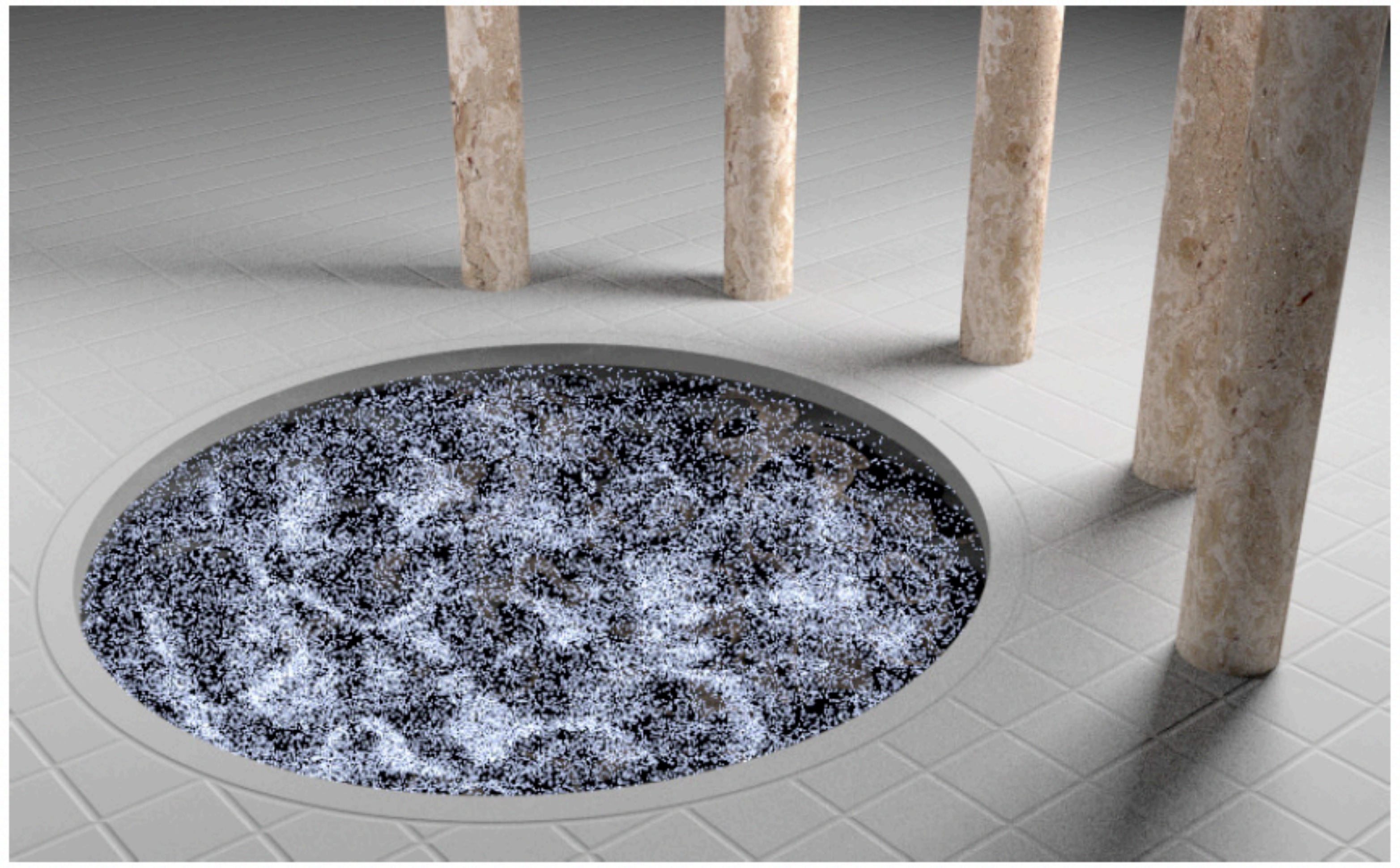


(a) Bidirectional path tracing with 40 samples per pixel.

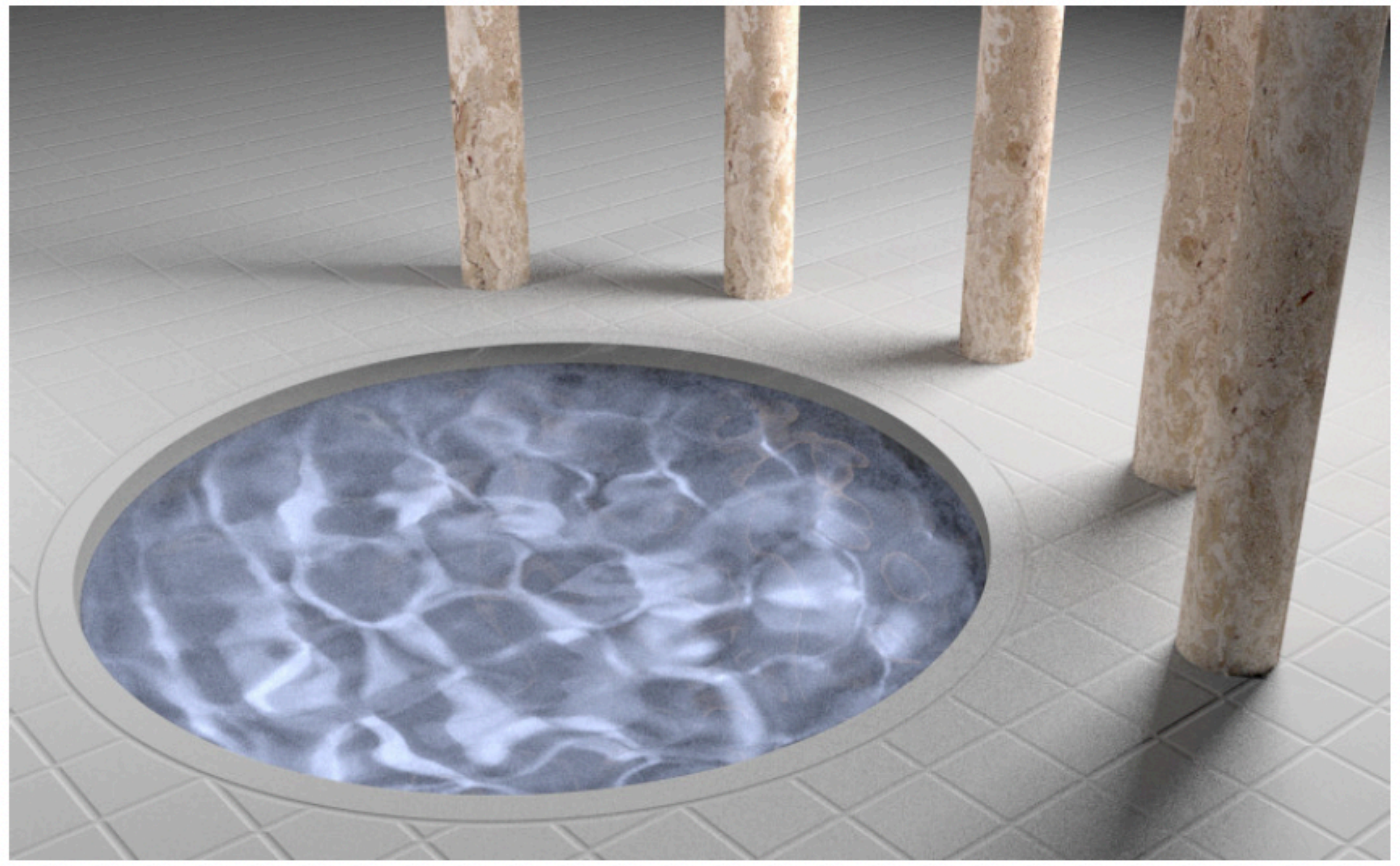
(b) Metropolis light transport with 250 mutations per pixel [the same computation time as (a)].



BDPT MLT



(a) Path tracing with 210 samples per pixel.

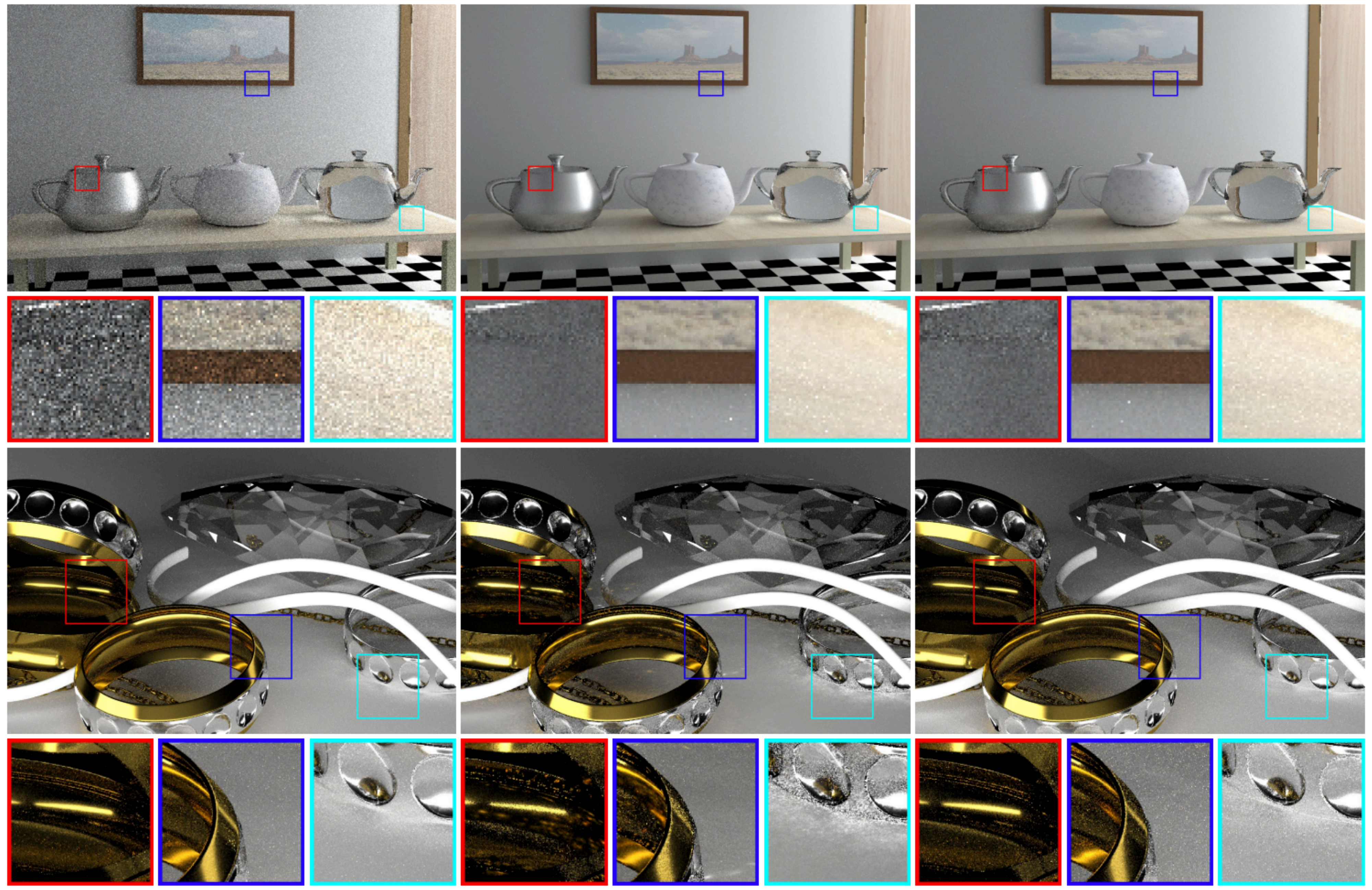


(b) Metropolis light transport with 100 mutations per pixel [the same computation time as (a)].

Combination of Veach & Kelemen

Fusing State Spaces for Markov Chain Monte Carlo Rendering

HISANARI OTSU, The University of Tokyo
ANTON S. KAPLANYAN, NVIDIA
JOHANNES HANIKA, Karlsruhe Institute of Technology
CARSTEN DACHSBACHER, Karlsruhe Institute of Technology
TOSHIYA HACHISUKA, The University of Tokyo



MMLT MLT ~~Ours~~
Hisanari's

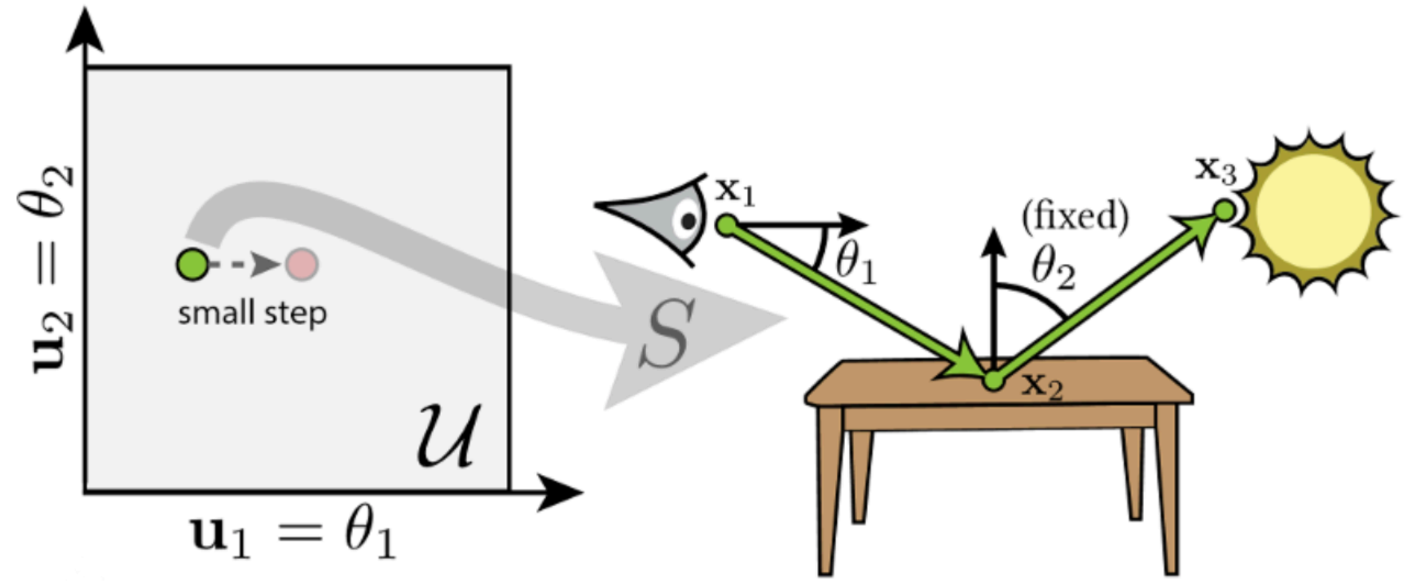
Charted Metropolis Light Transport

Jacopo Pantaleoni*
NVIDIA

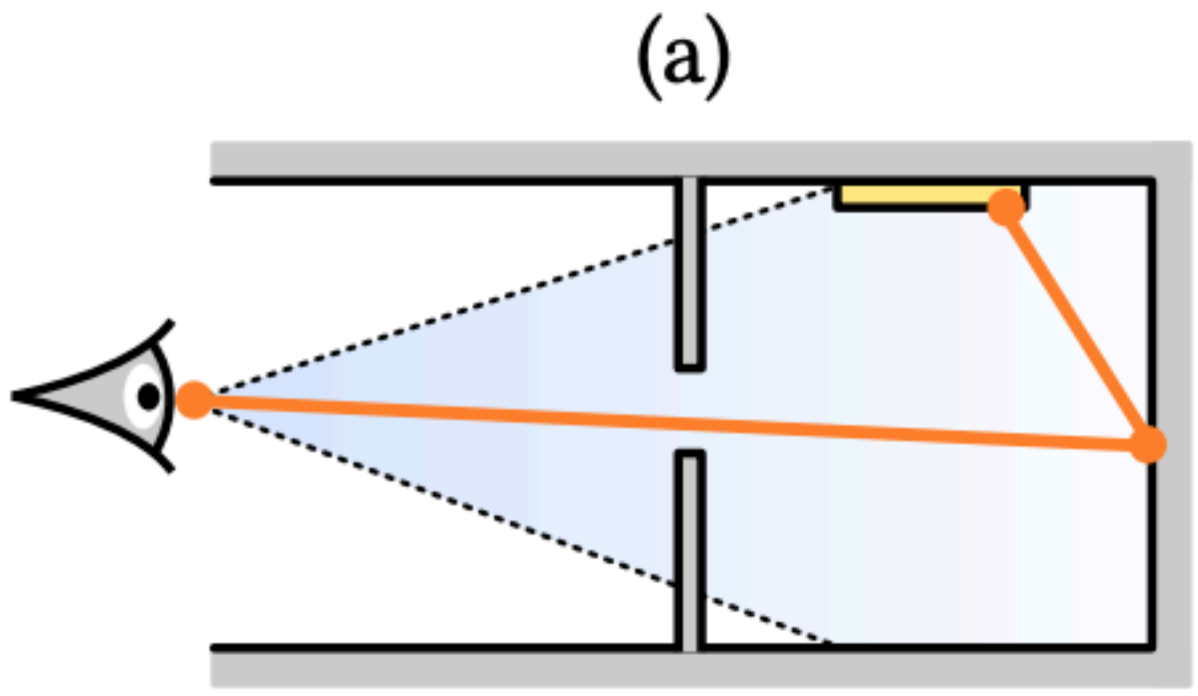
Reversible Jump Metropolis Light Transport using Inverse Mappings

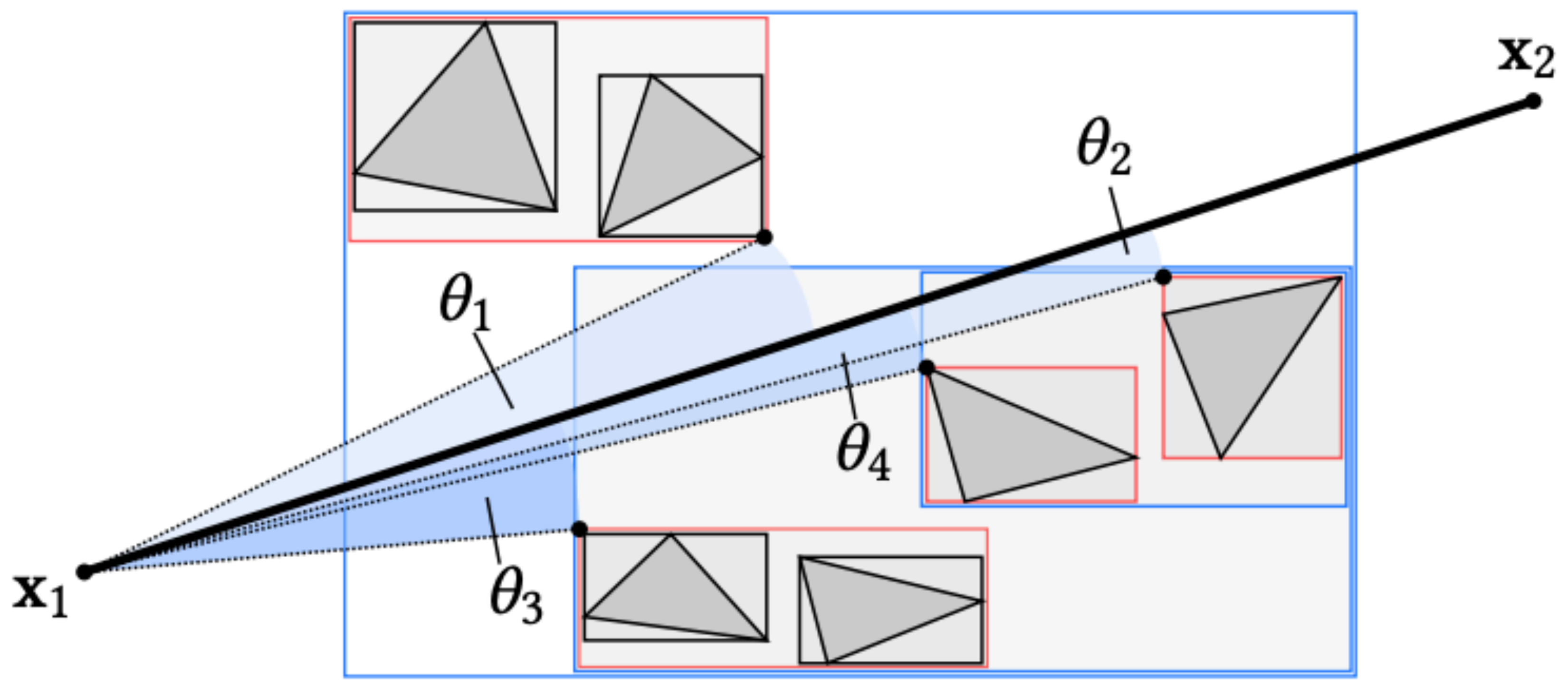
Benedikt Bitterli Wenzel Jakob Jan Novák Wojciech Jarosz

ACM Transactions on Graphics (TOG), 37(1), October 2017



Better lens / caustics perturbation with cone fitting

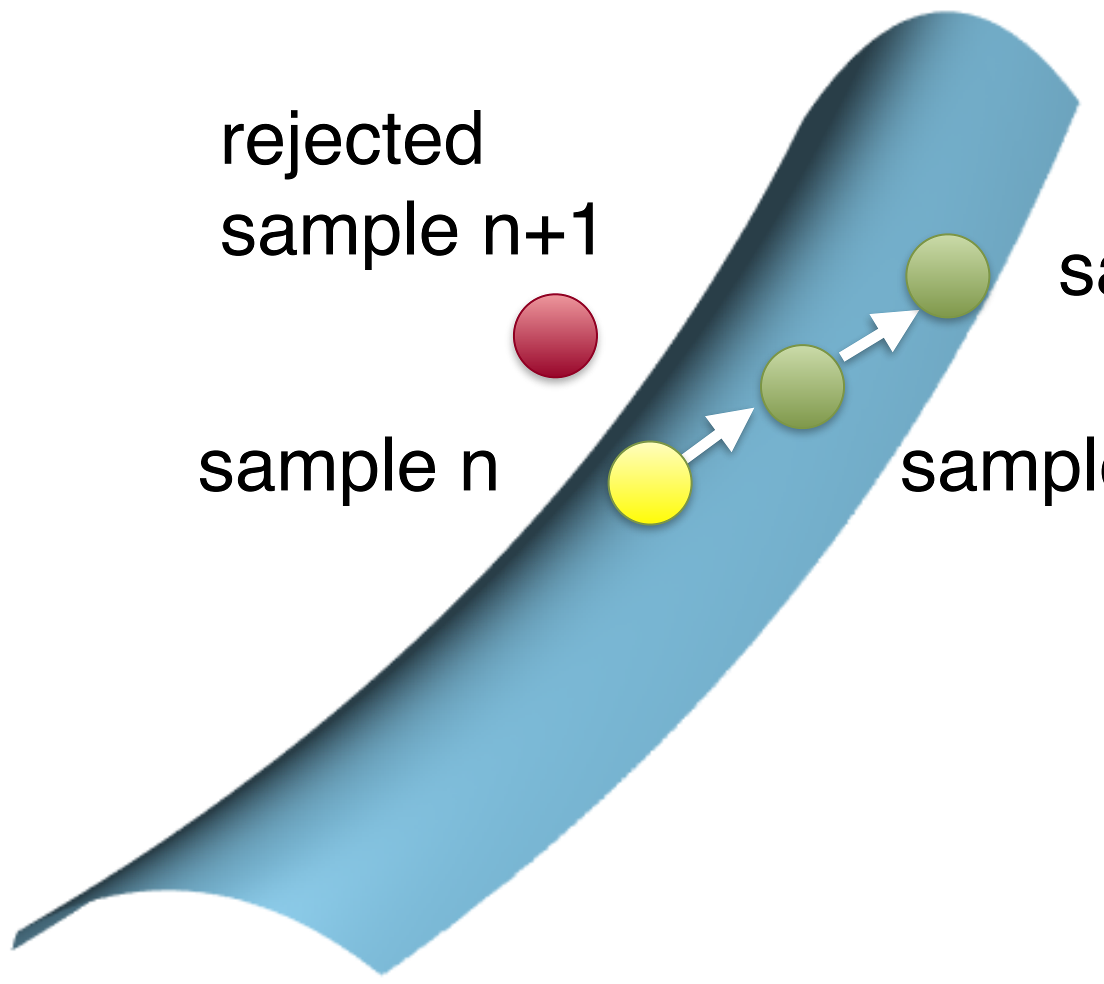




Geometry-Aware Metropolis Light Transport

HISANARI OTSU, Karlsruhe Institute of Technology and The University of Tokyo
JOHANNES HANIKA, Karlsruhe Institute of Technology
TOSHIYA HACHISUKA, The University of Tokyo
CARSTEN DACHSBACHER, Karlsruhe Institute of Technology

Can we use differentiable rendering to help MLT?



this looks like gradient ascent/Newton's method!

Anisotropic Gaussian Mutations for Metropolis Light Transport through Hessian-Hamiltonian Dynamics

Tzu-Mao Li
MIT CSAIL

Jaakko Lehtinen
Aalto University
NVIDIA

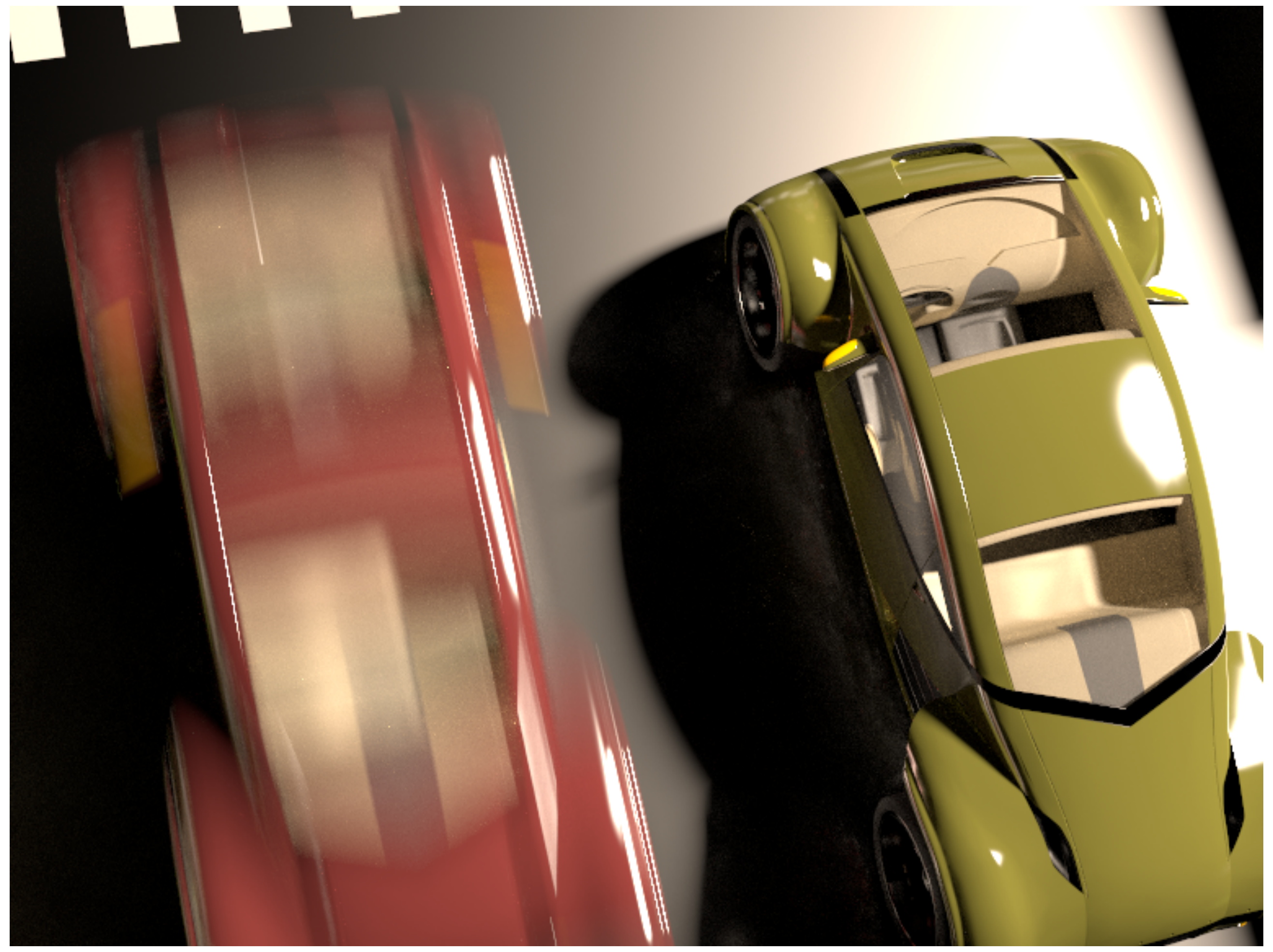
Ravi Ramamoorthi
University of California, San Diego

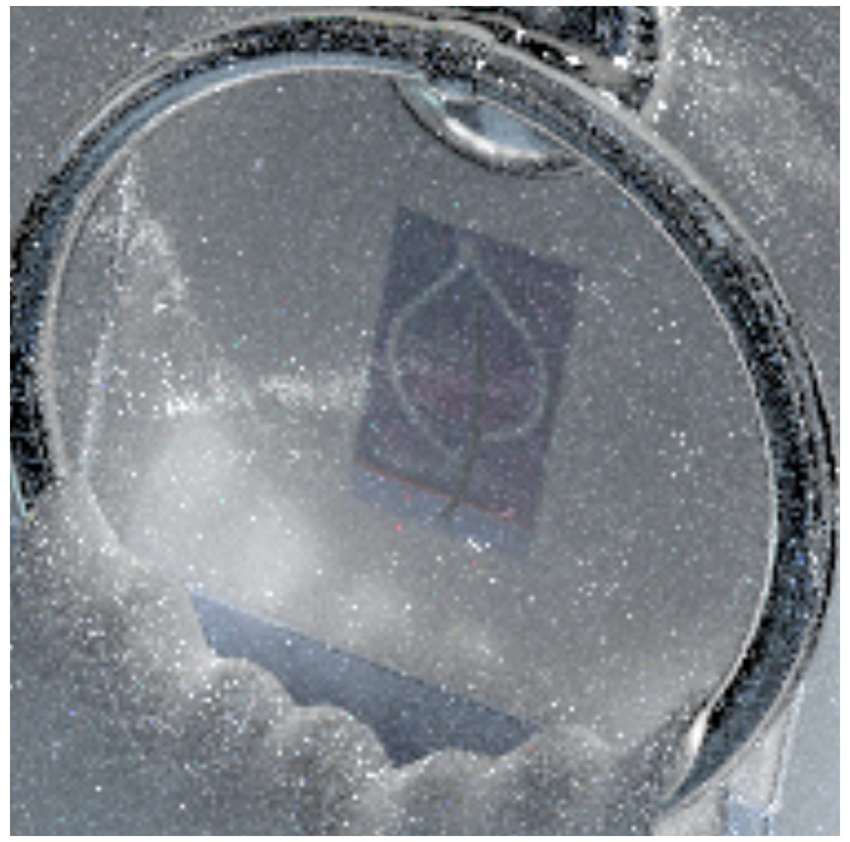
Wenzel Jakob
ETH Zürich

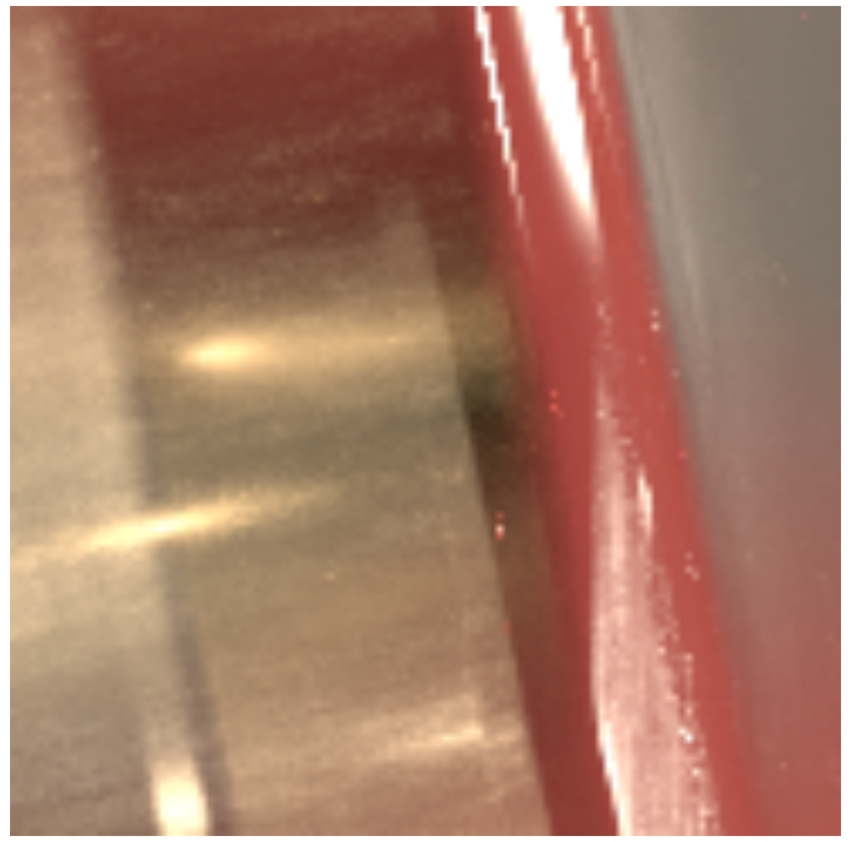
Frédo Durand
MIT CSAIL

Motivation: rendering difficult light paths

e.g. multi-bounce glossy light paths combined with motion blur

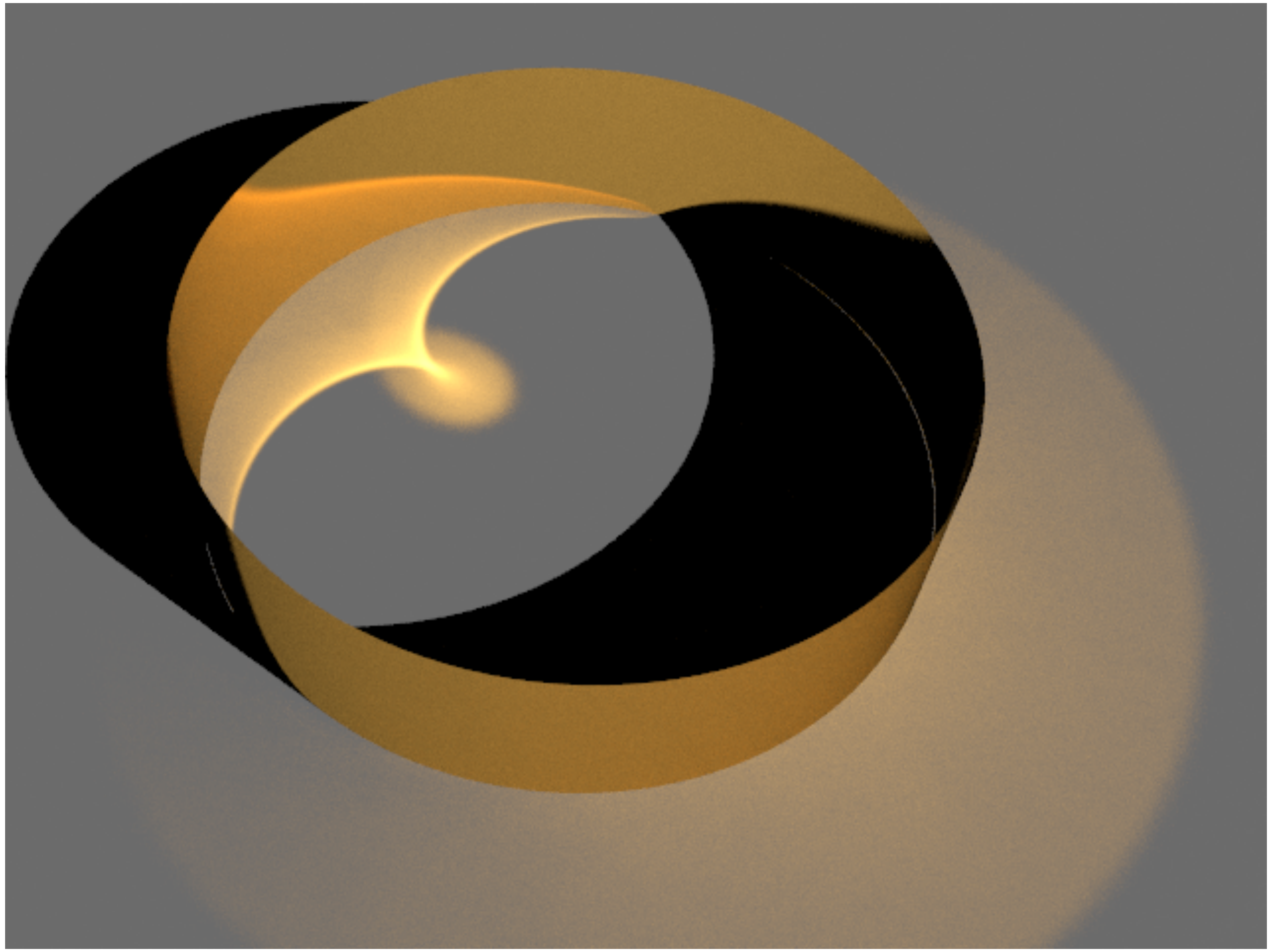


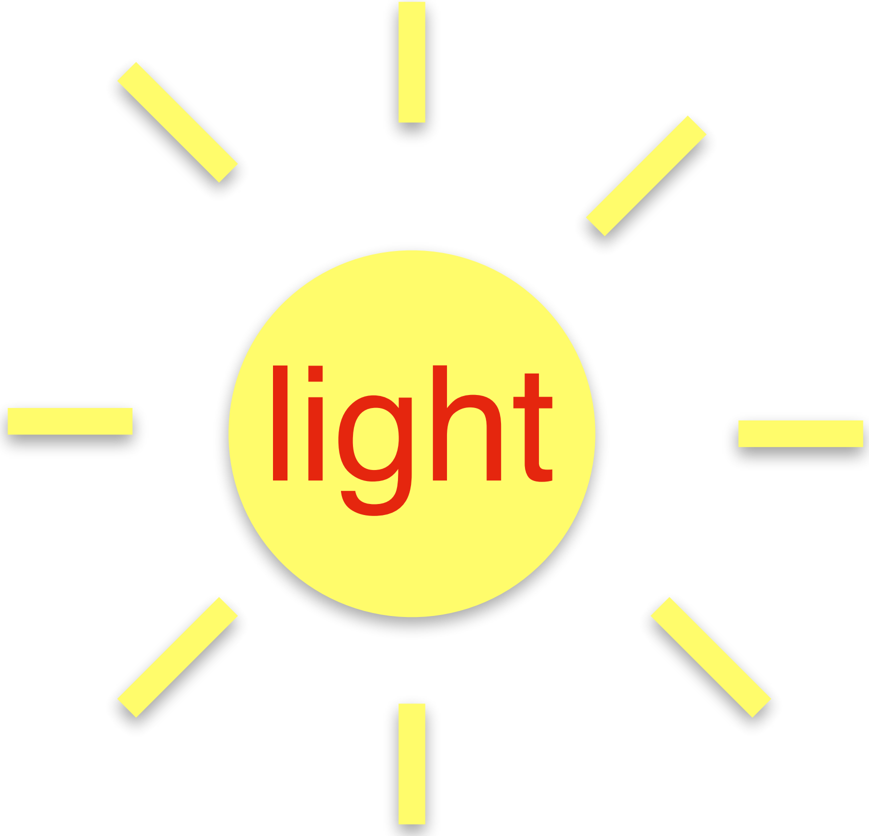




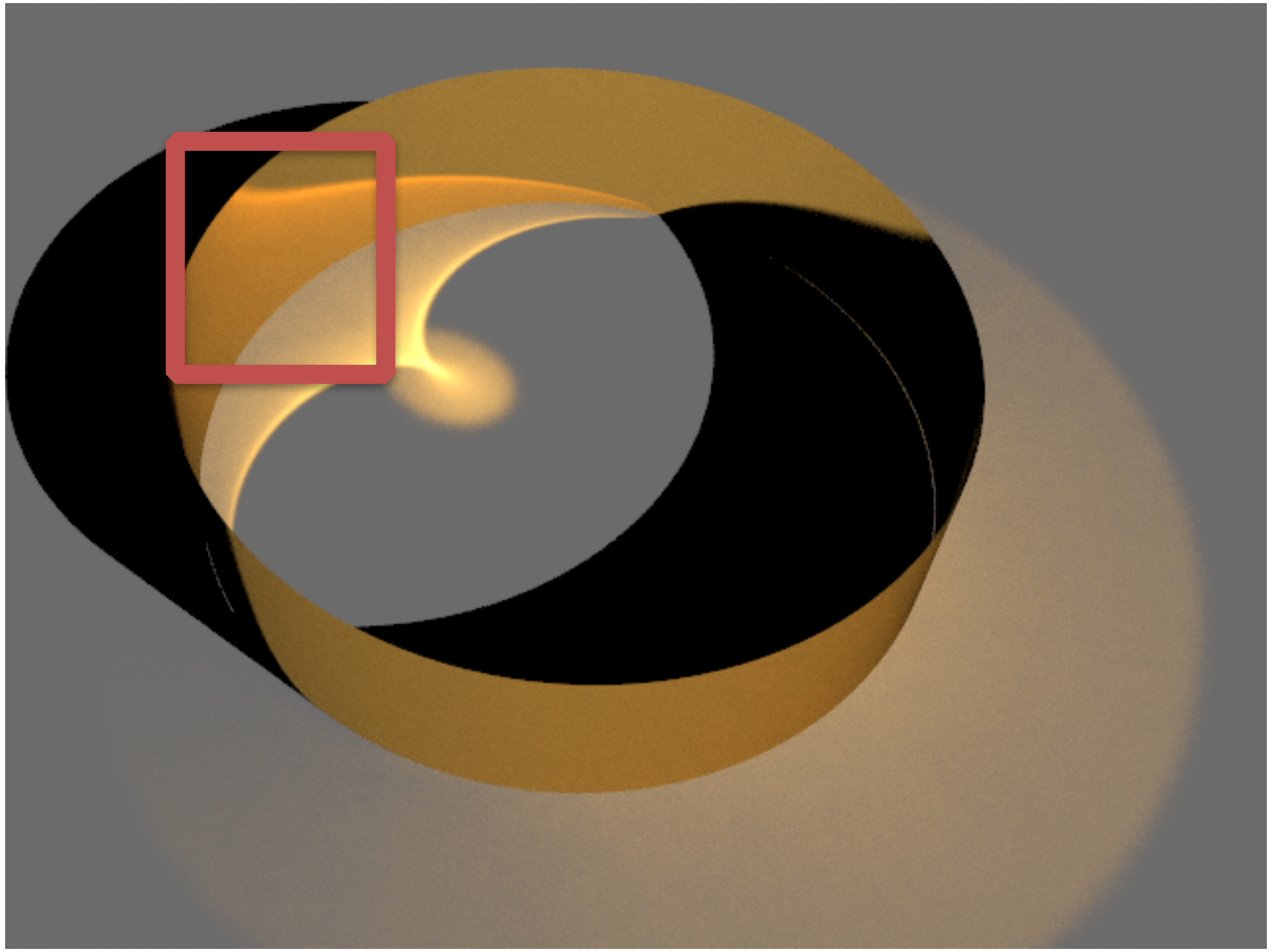
narrow contribution regions can lead to noisy images

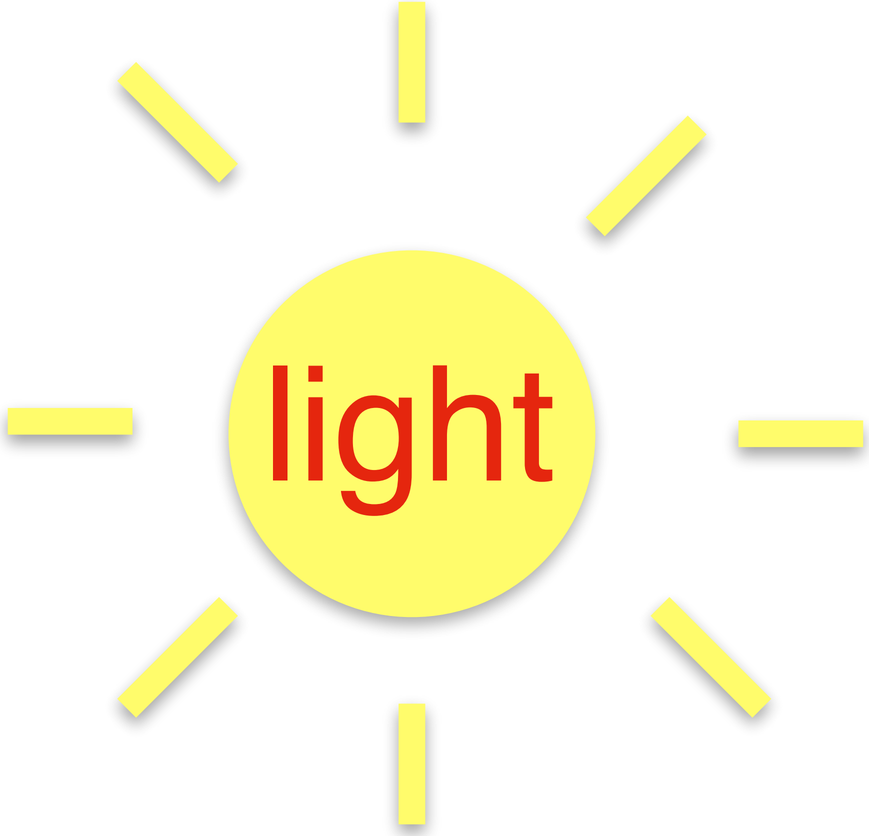
The ring example



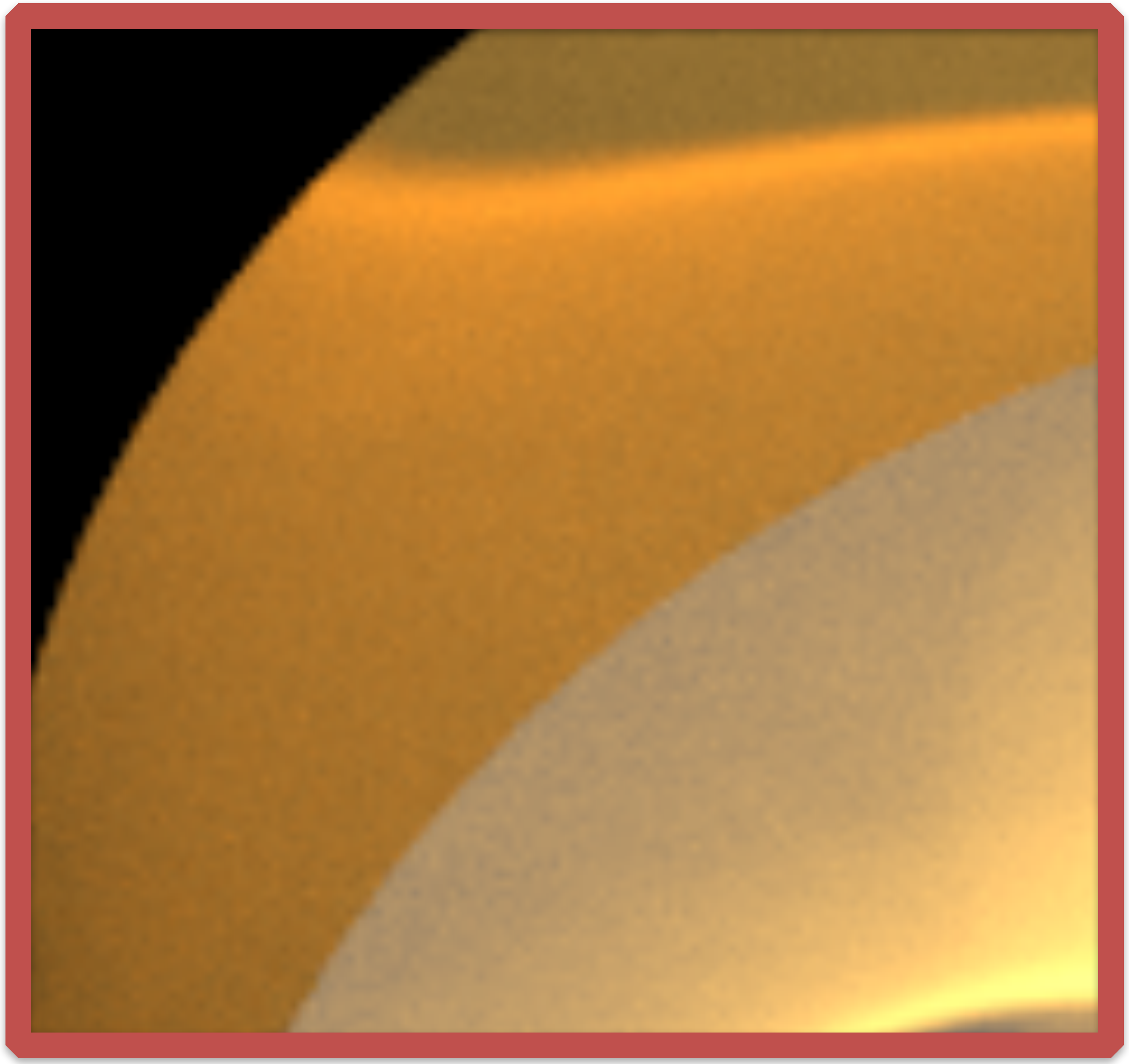


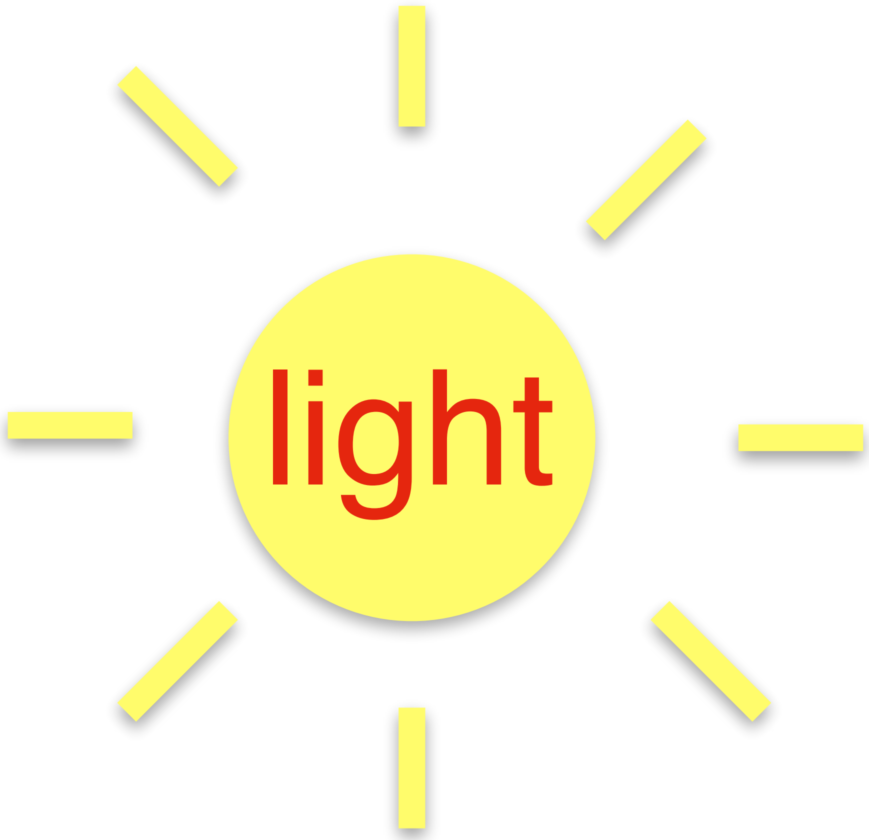
The ring example





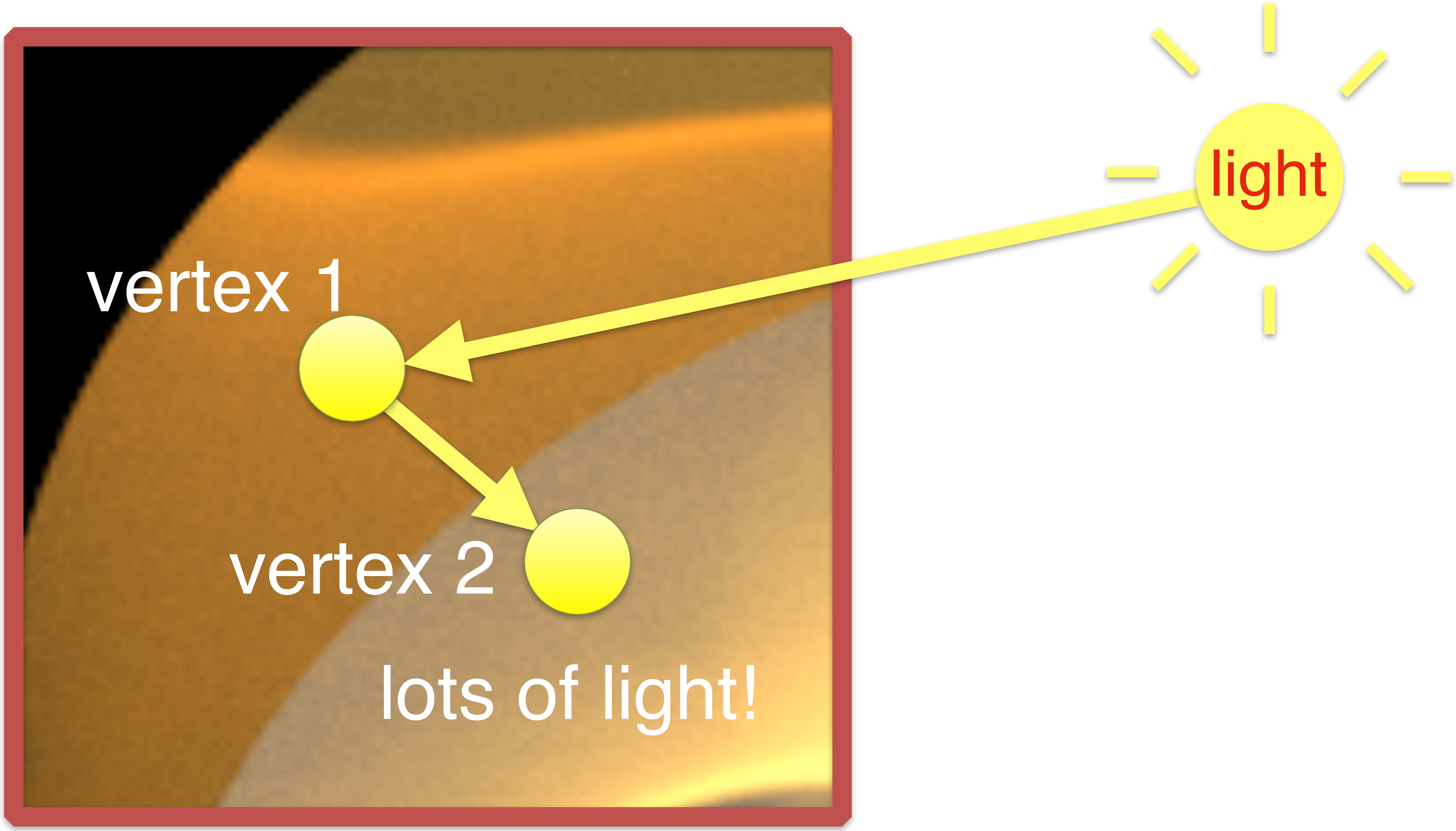
The ring example





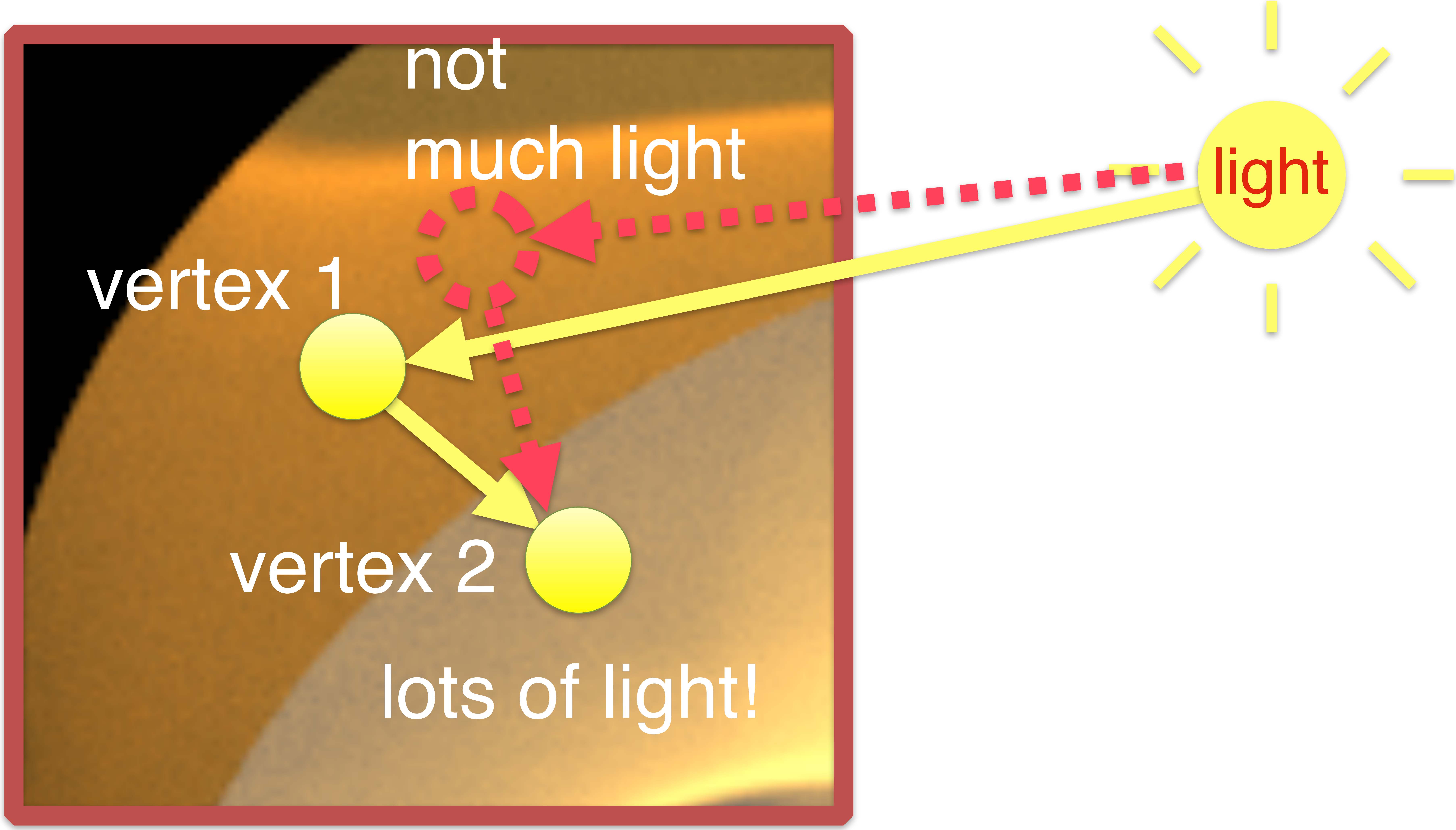
Path contribution varies

depends on geometry, BRDF, light, etc



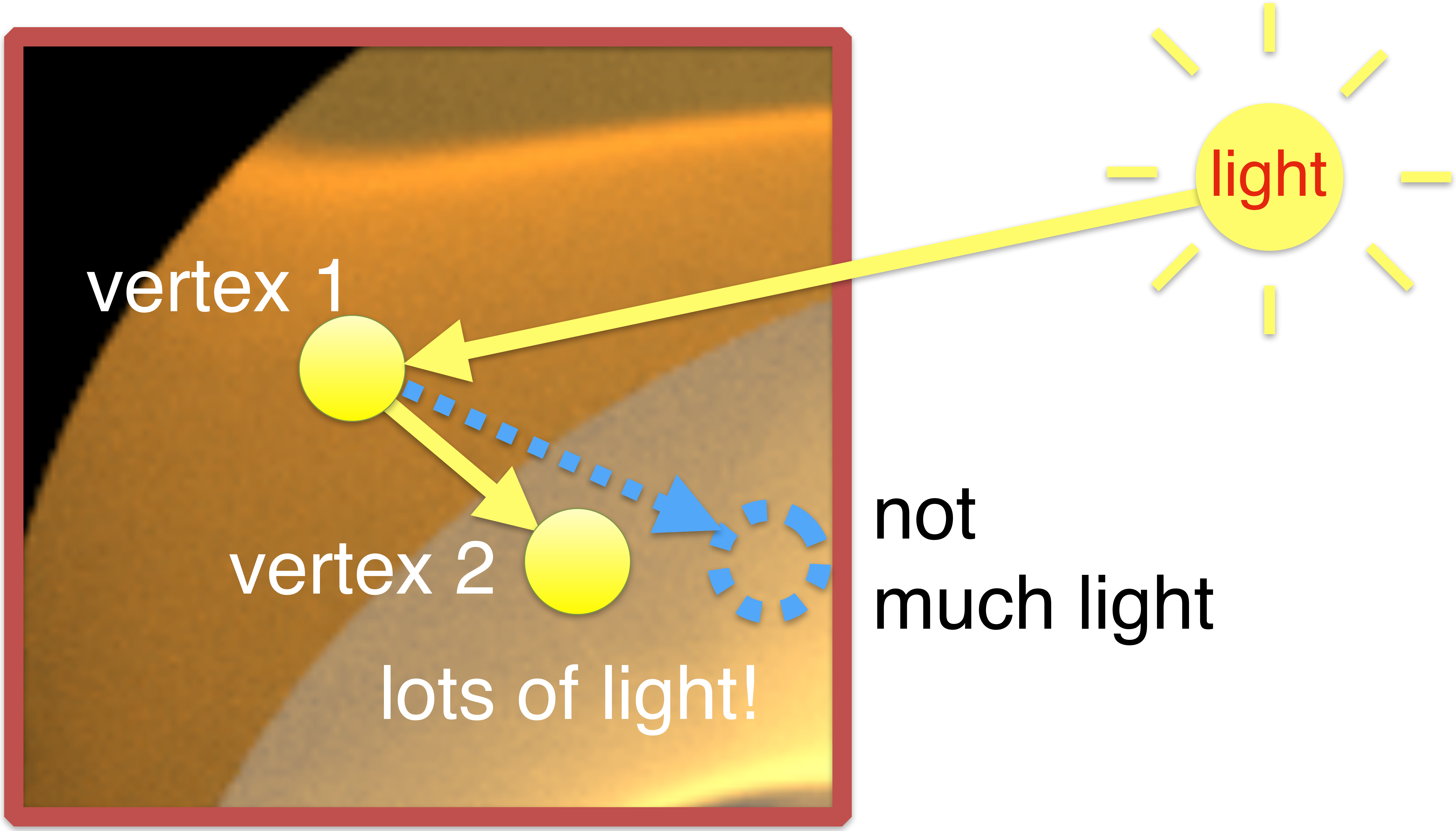
Path contribution varies

depends on geometry, BRDF, light, etc



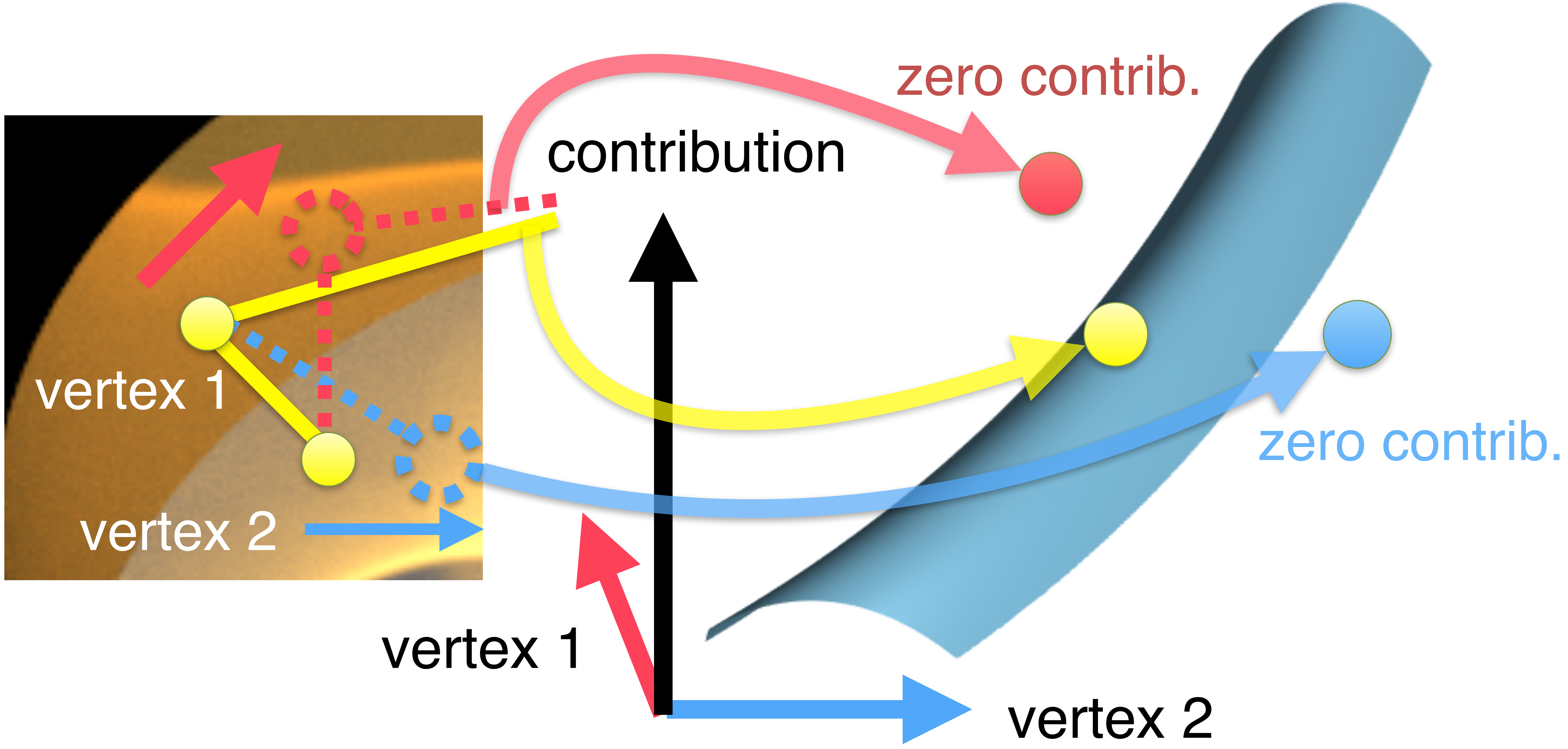
Path contribution varies

depends on geometry, BRDF, light, etc



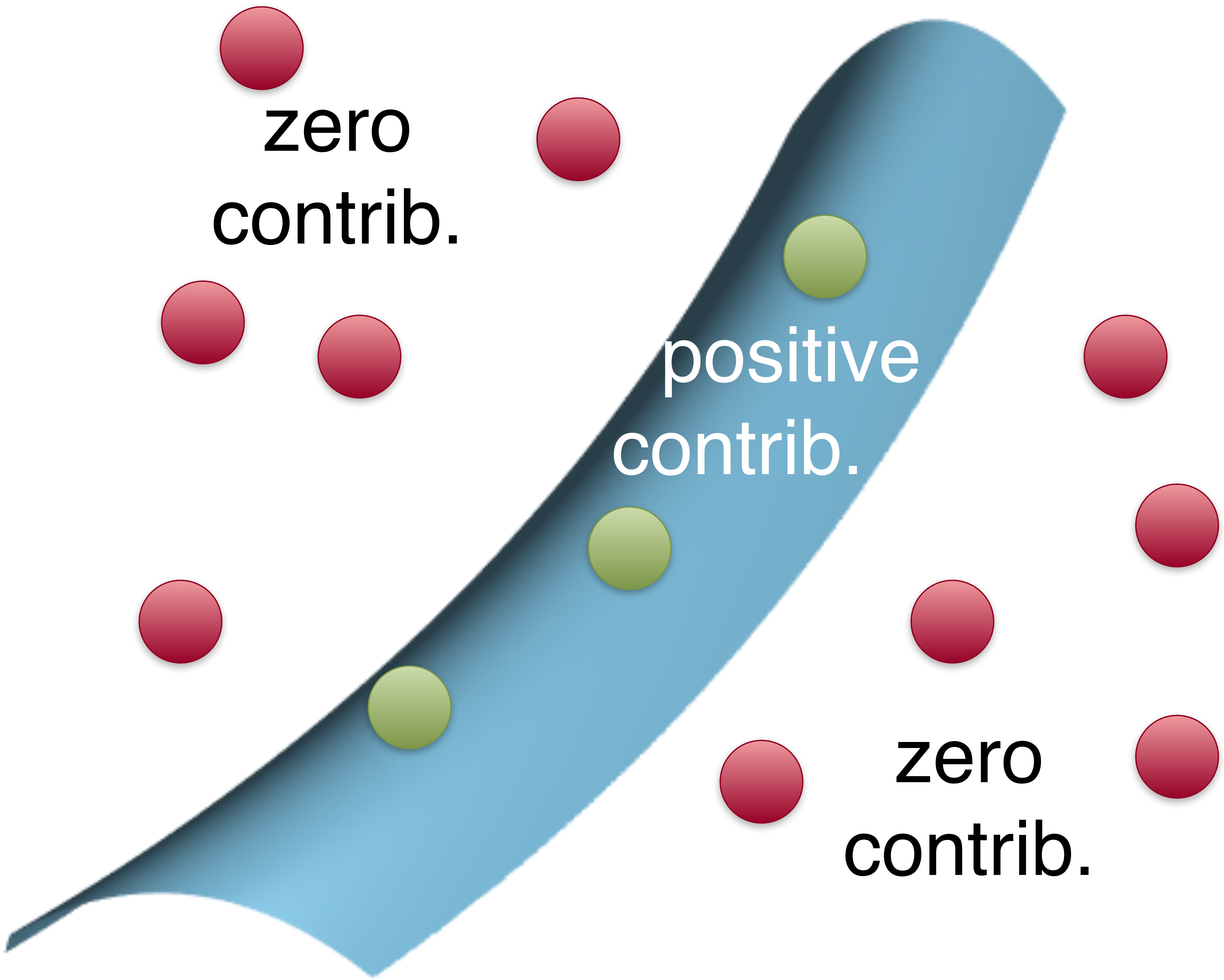
Visualization of path space contribution

- paths → 2D horizontal locations
- contribution → up direction
- narrow & anisotropic



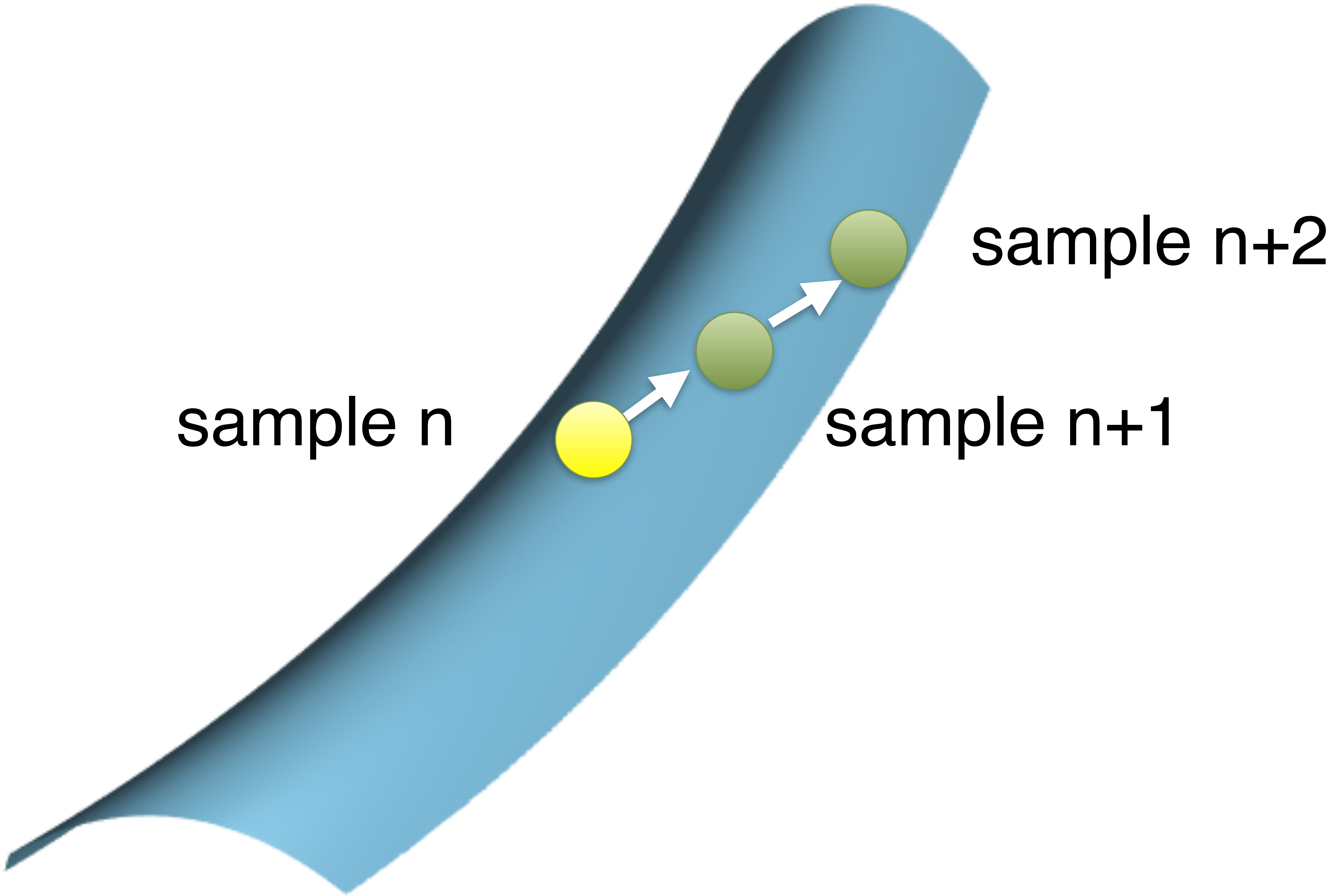
Monte Carlo: inefficient!

- don’t know contribution function, can only sample it
- few samples in high contribution region



Metropolis Light Transport [Veach 1997]

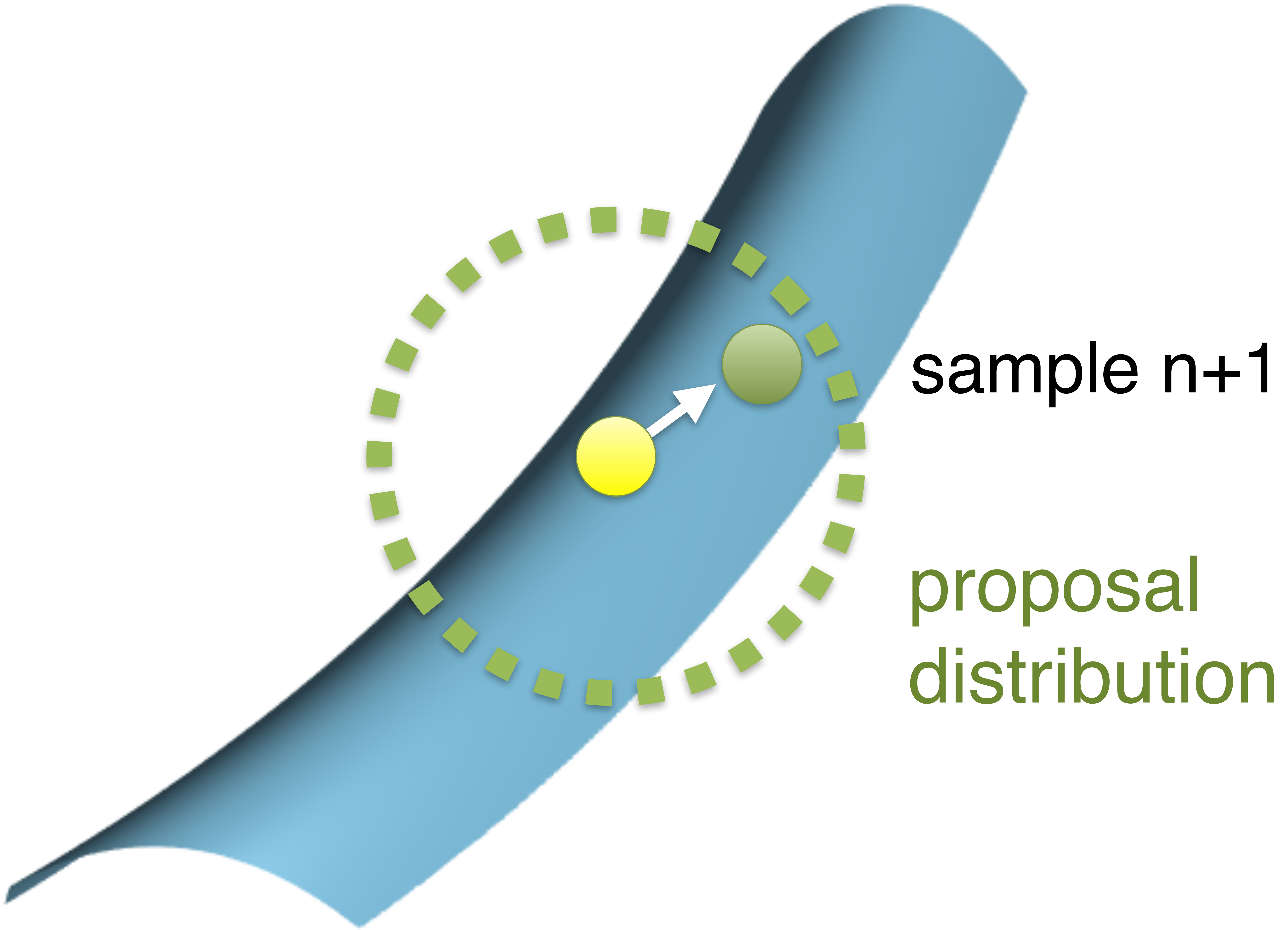
idea: stays in high contribution region with Markov chain



Metropolis Light Transport [Veach 1997]

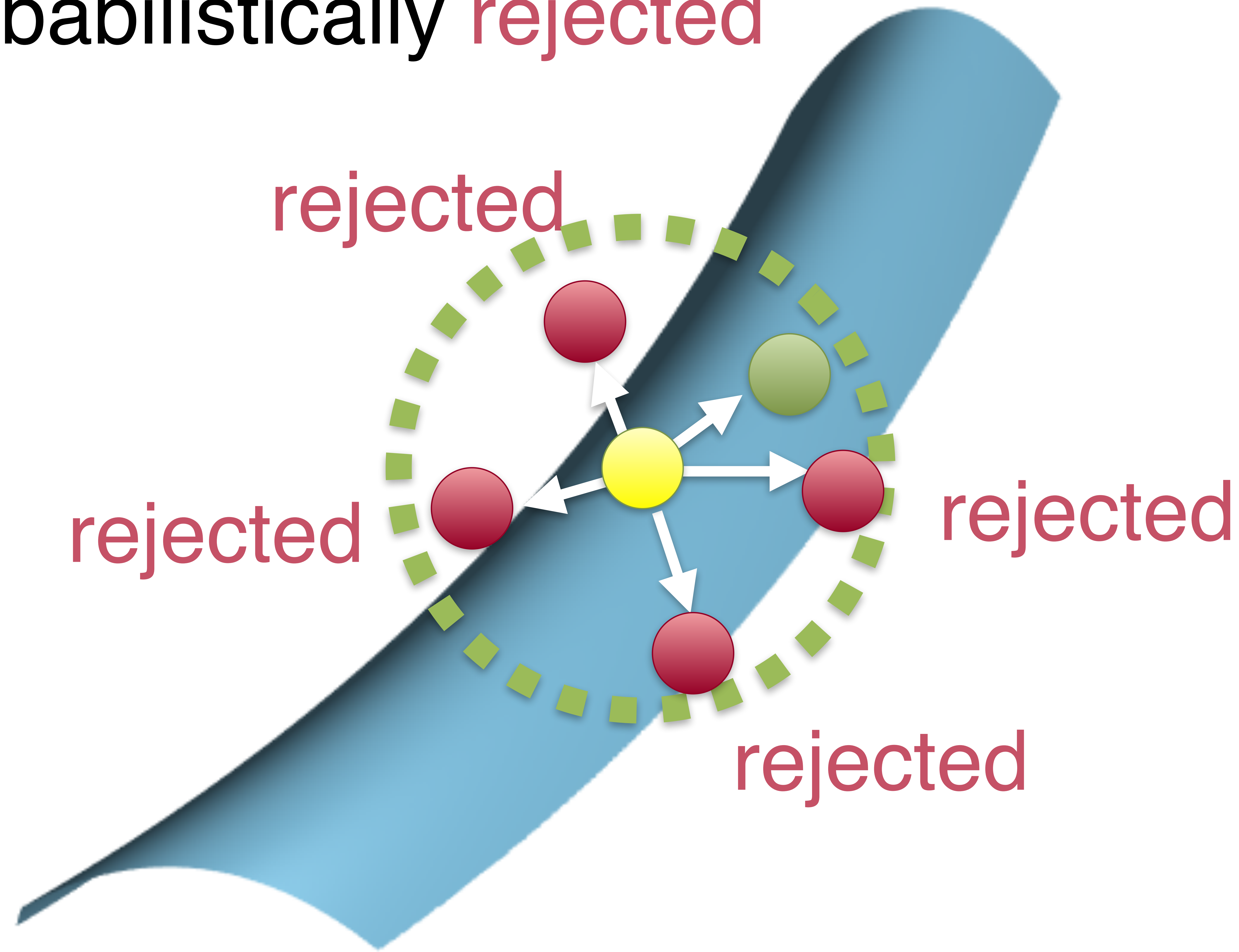
idea: stays in high contribution region with Markov chain

sample n+1 drawn from proposal distribution



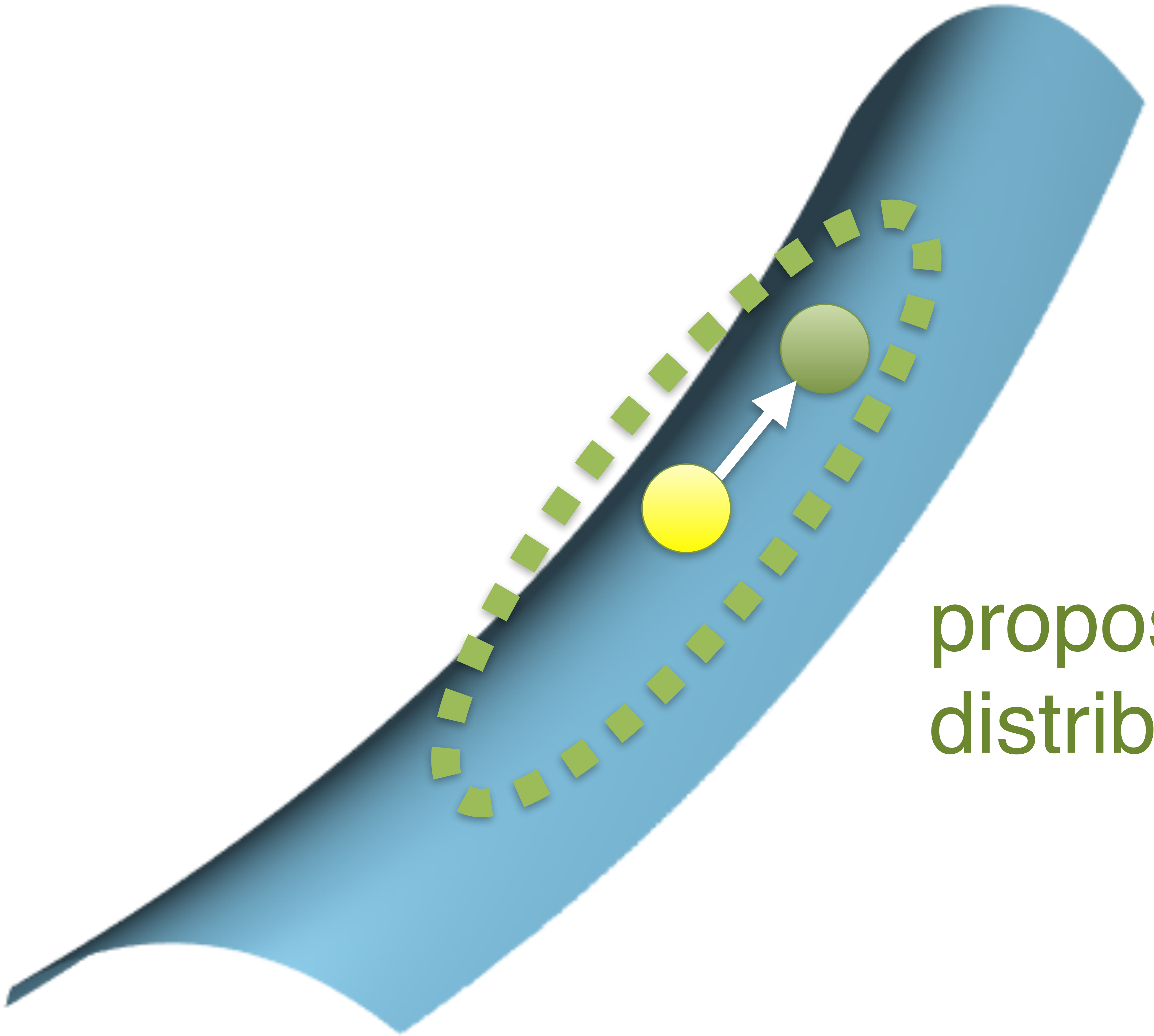
Metropolis Light Transport [Veach 1997]

problem:
proposals with low contribution are probabilistically rejected



Our goal: anisotropic proposal

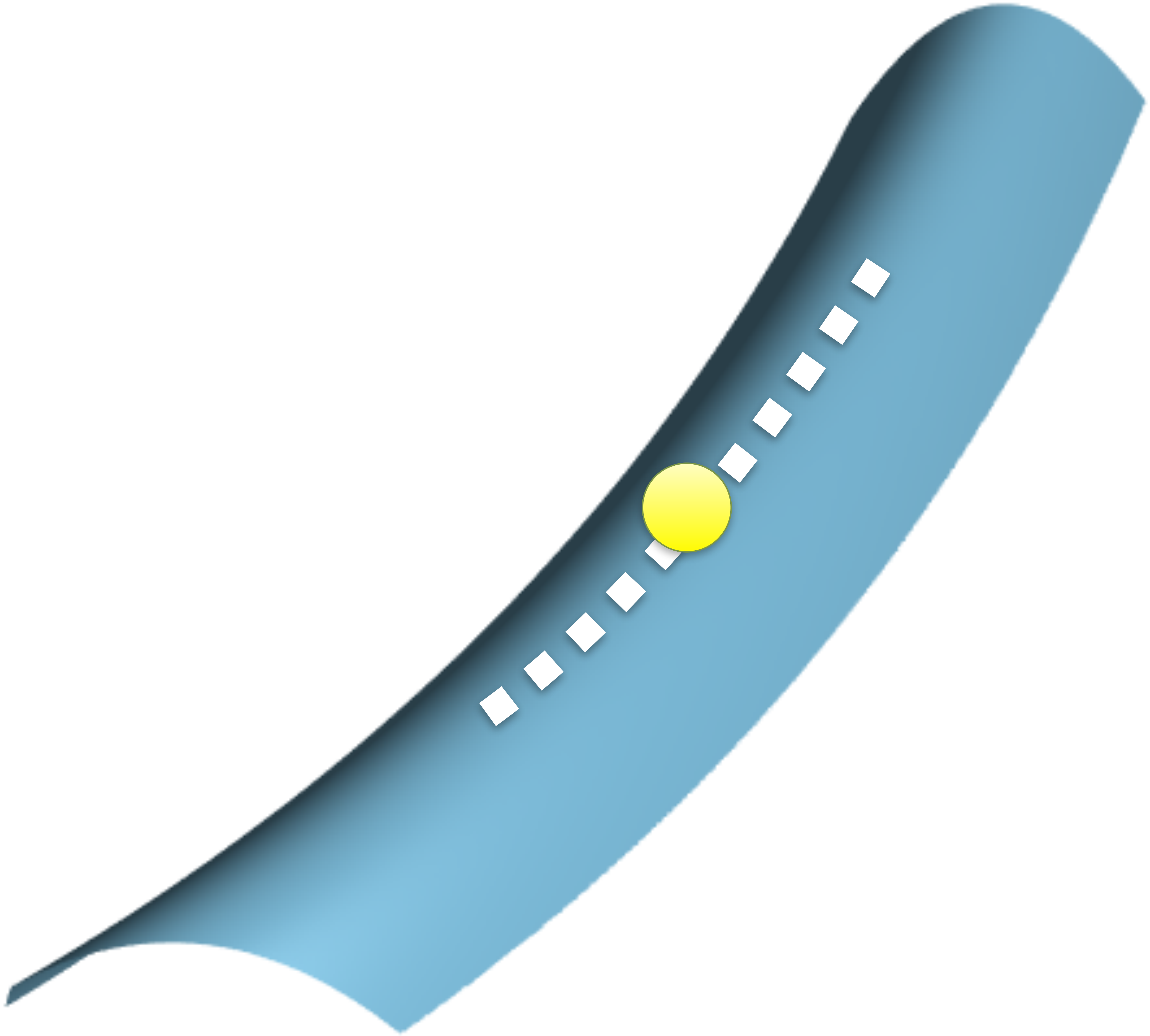
- proposal stays in high contribution region



proposal
distribution

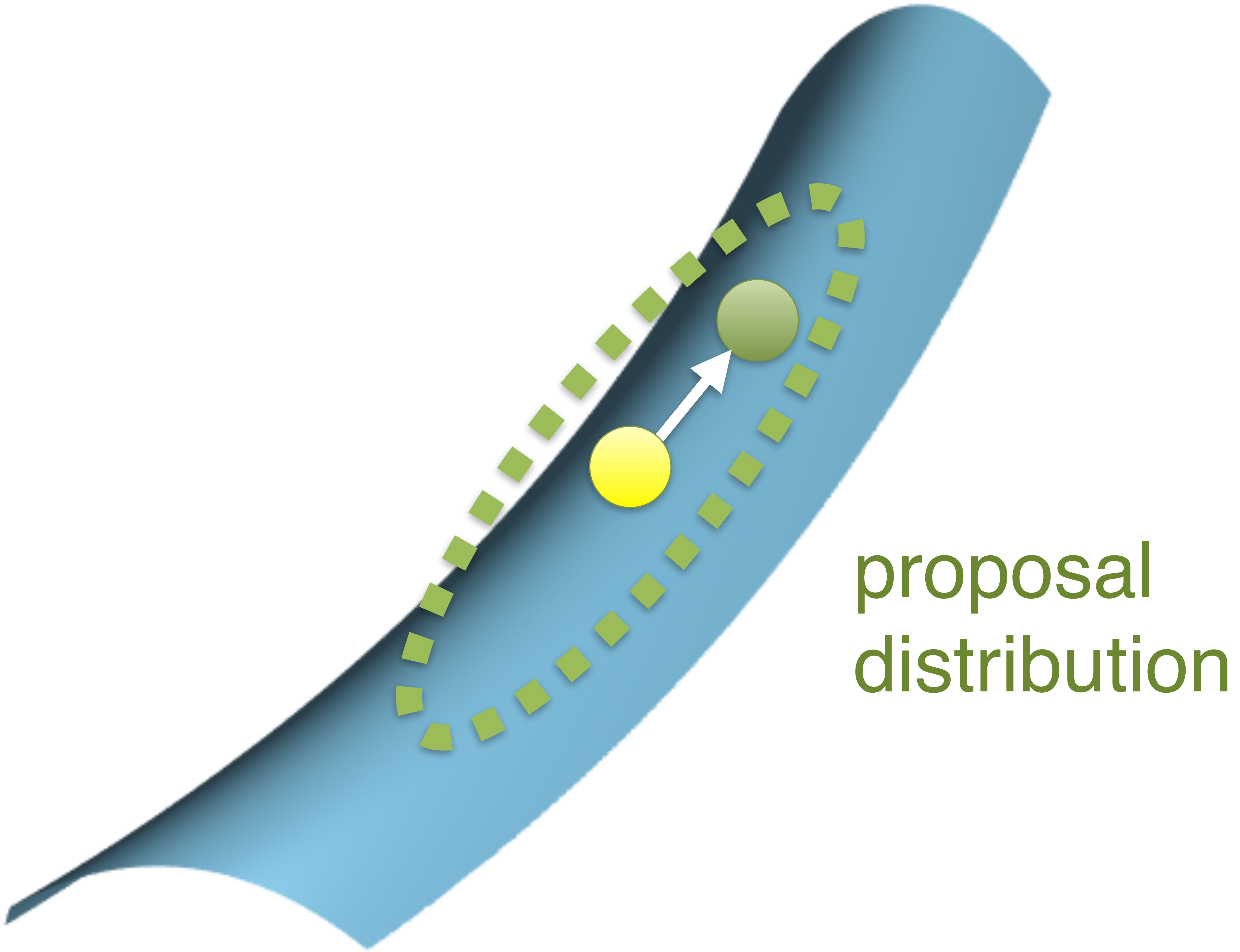
Previous work [Jakob 2012, Kaplanyan 2014]

- specialized for microfacet BRDF & mirror directions
- proposal in special directions

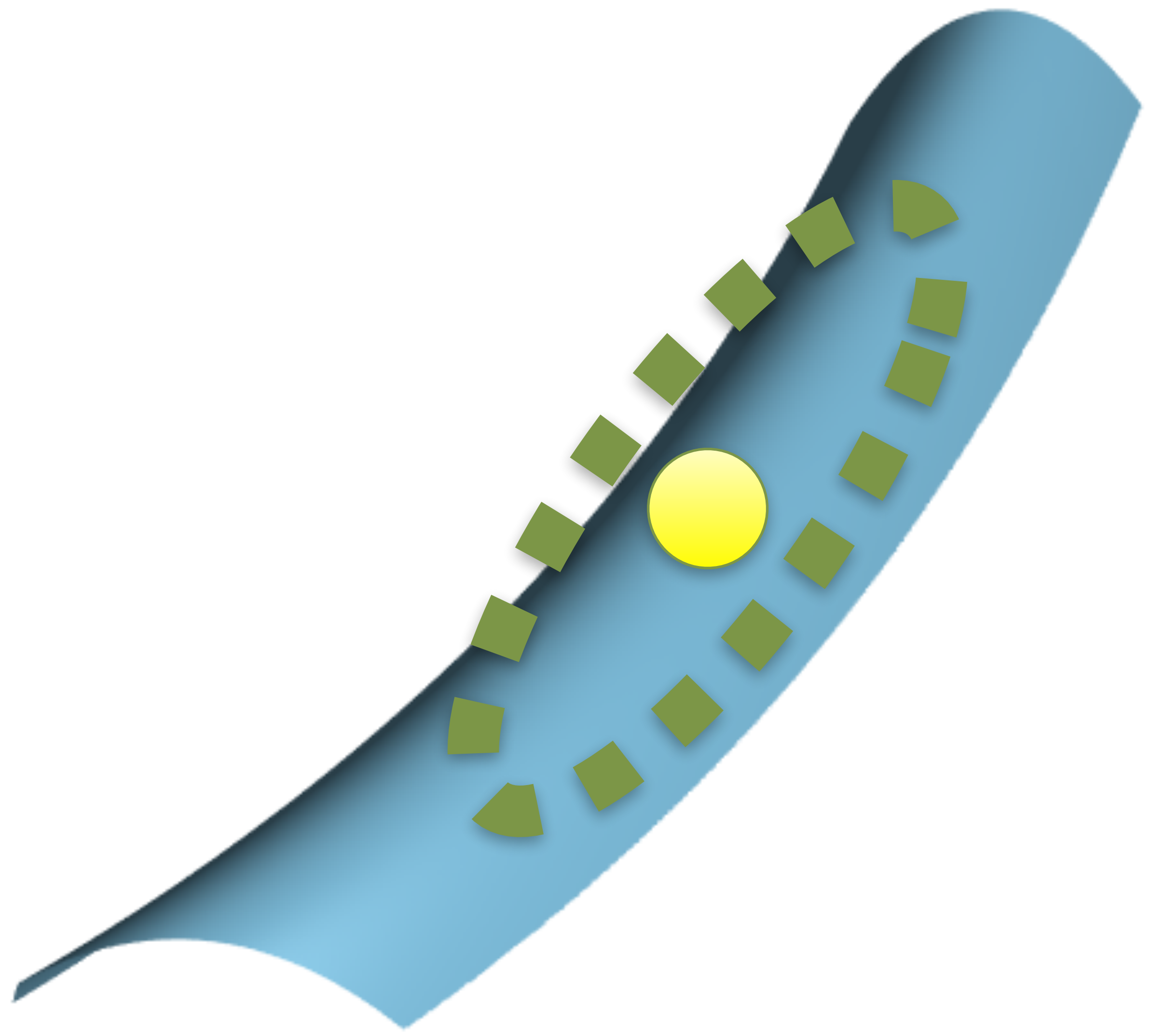


Our goal: anisotropic proposal

- proposal stays in high contribution region
- fully general approach

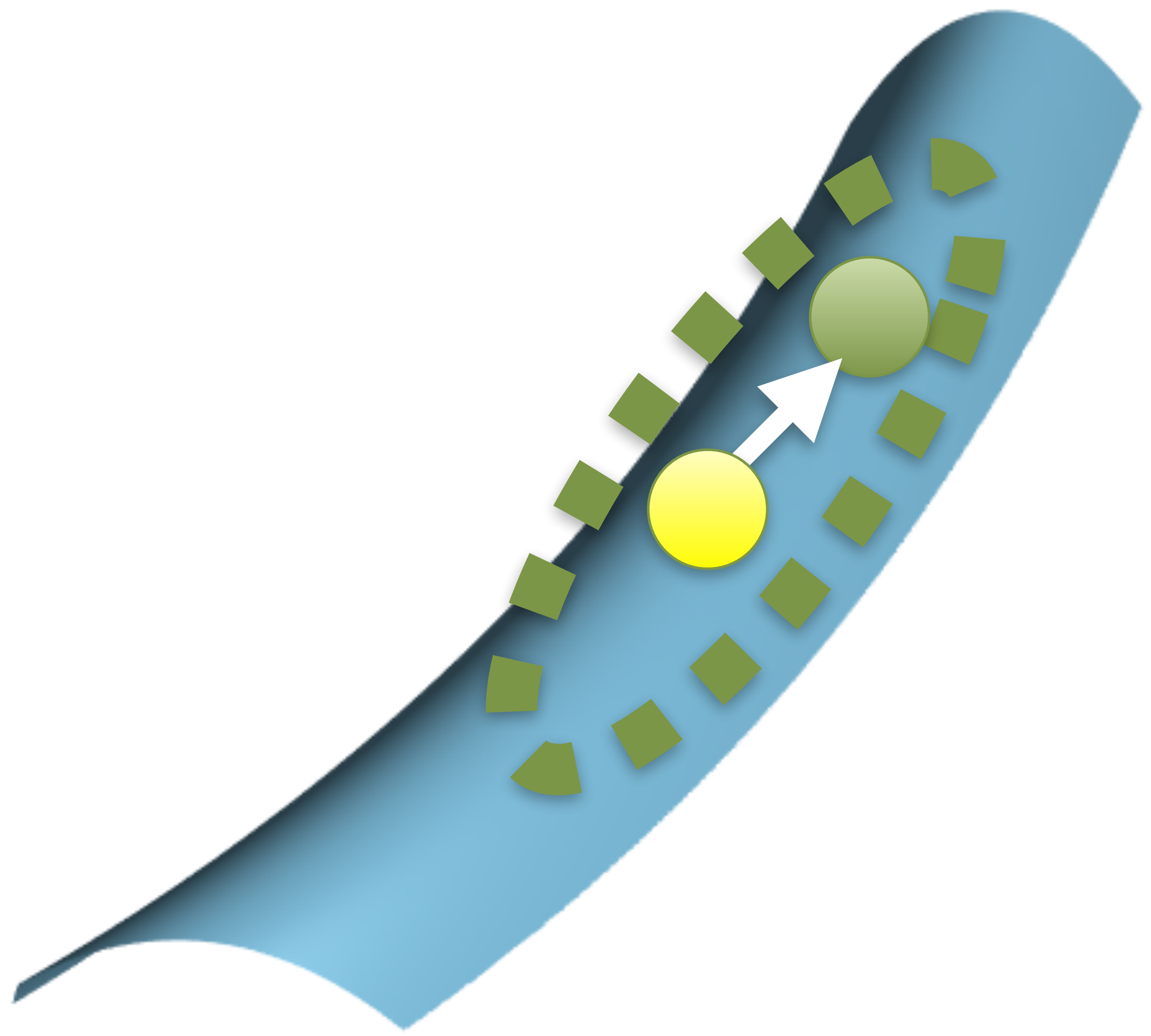


Challenges & our solutions



1: characterize anisotropy

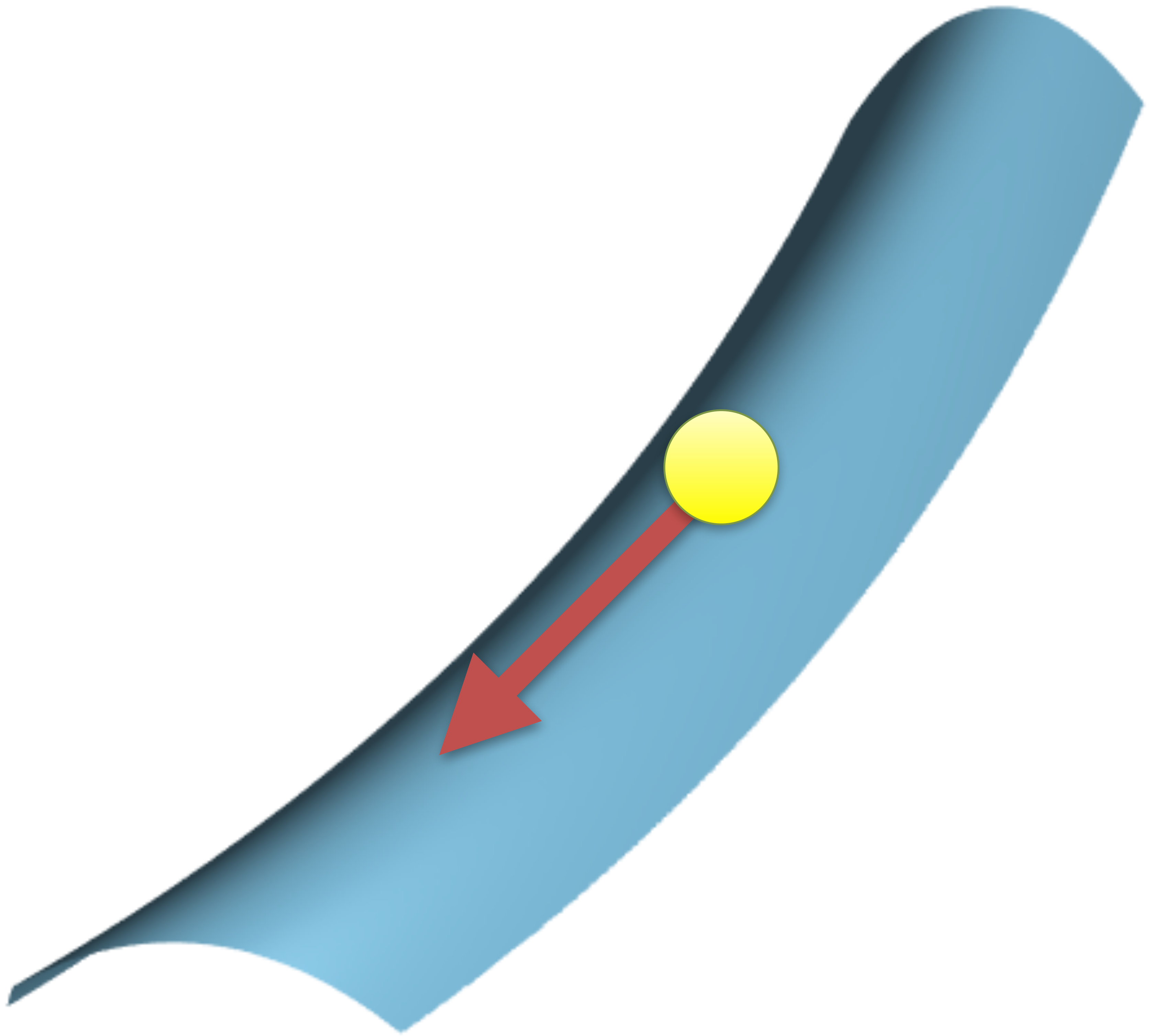
→ use 2nd derivatives (Hessian)
→quadratic approximation



2: sample quadratic
(not distributions!)

→ simulate Hamiltonian dynamics

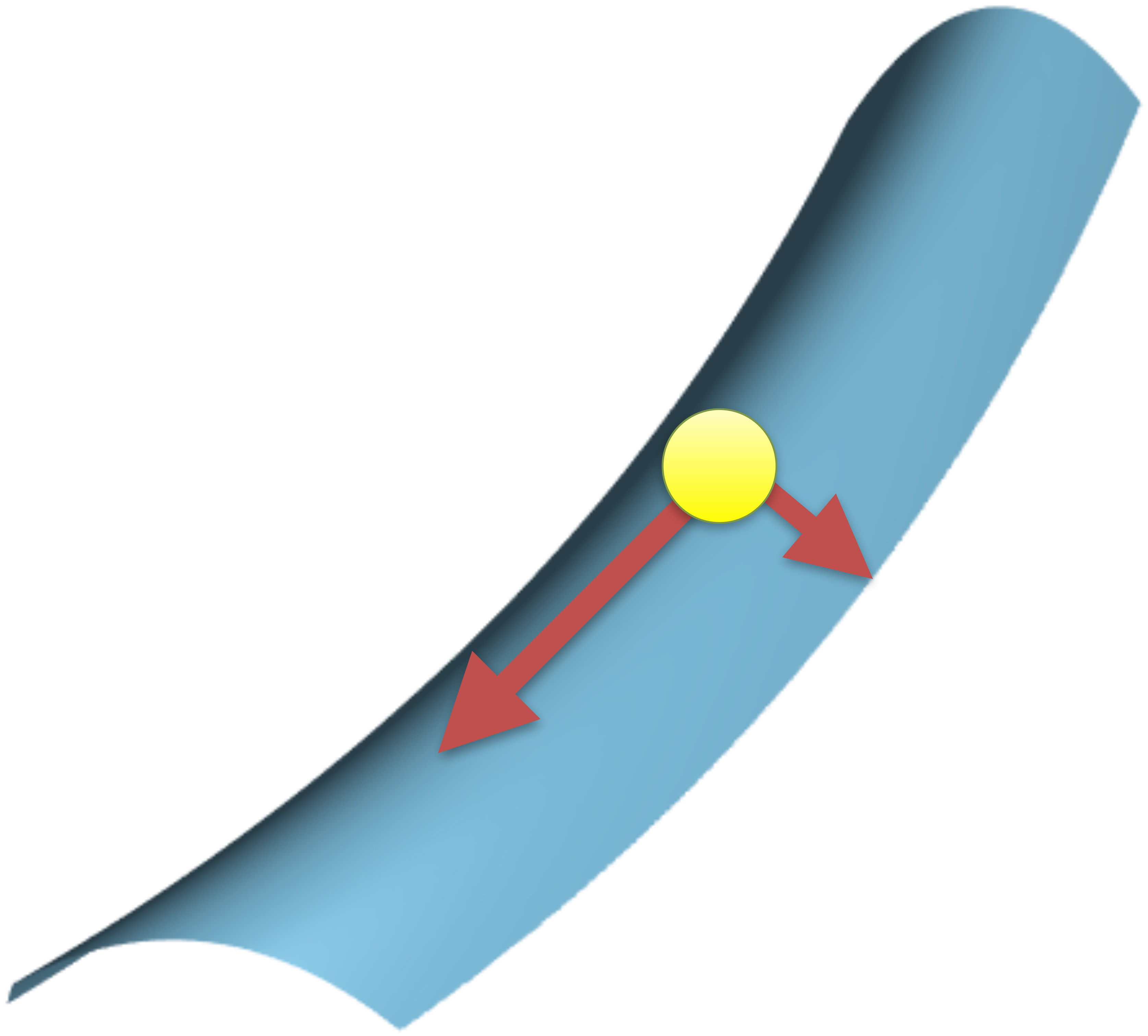
Gradient informs only one direction



$$\frac{\partial f}{\partial u_i}$$

Hessian provides correlation between coordinates

characterize anisotropy in all direction



$$\frac{\partial^2 f}{\partial u_i \partial u_j}$$

Automatic differentiation provides gradient + Hessian

- no hand derivation
- metaprogramming approach
  - chain rule applied automatically
- in practice, implement with special datatype

```
ADFloat f(const ADFloat x[2]) {
 ADFloat y = sin(x[0]);
 ADFloat z = cos(x[1]);
 return y * z;
}
```

e.g. [Griewank and Walther 2008]

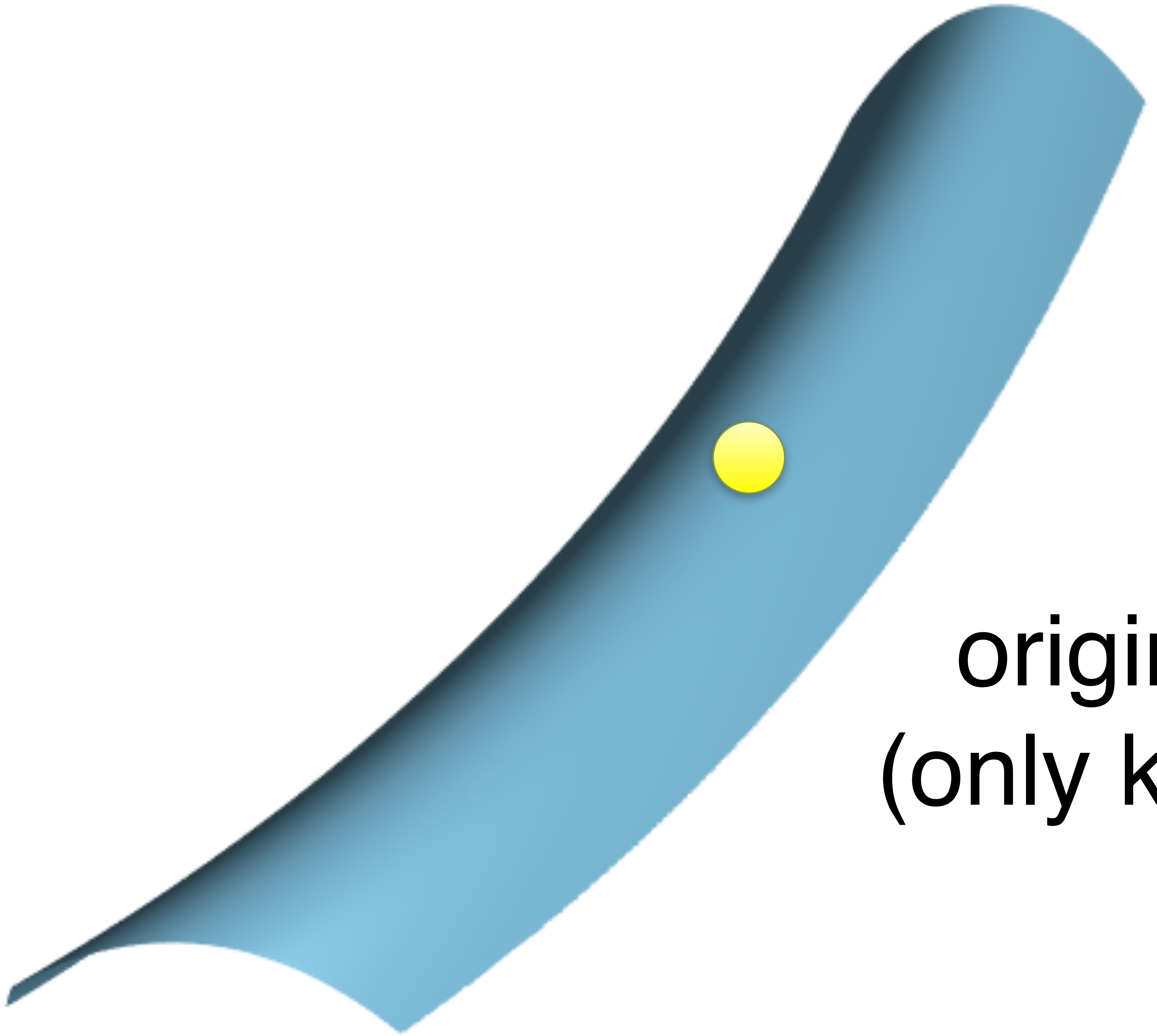
Automatic differentiation provides gradient + Hessian

- implement path contribution with automatic differentiation datatypes
  - normal, BRDF, light source
  - derivatives w.r.t path vertex coordinates

```
ADFloat f(const ADFloat x[2]) {
 ADFloat y = sin(x[0]);
 ADFloat z = cos(x[1]);
 return y * z;
}
```

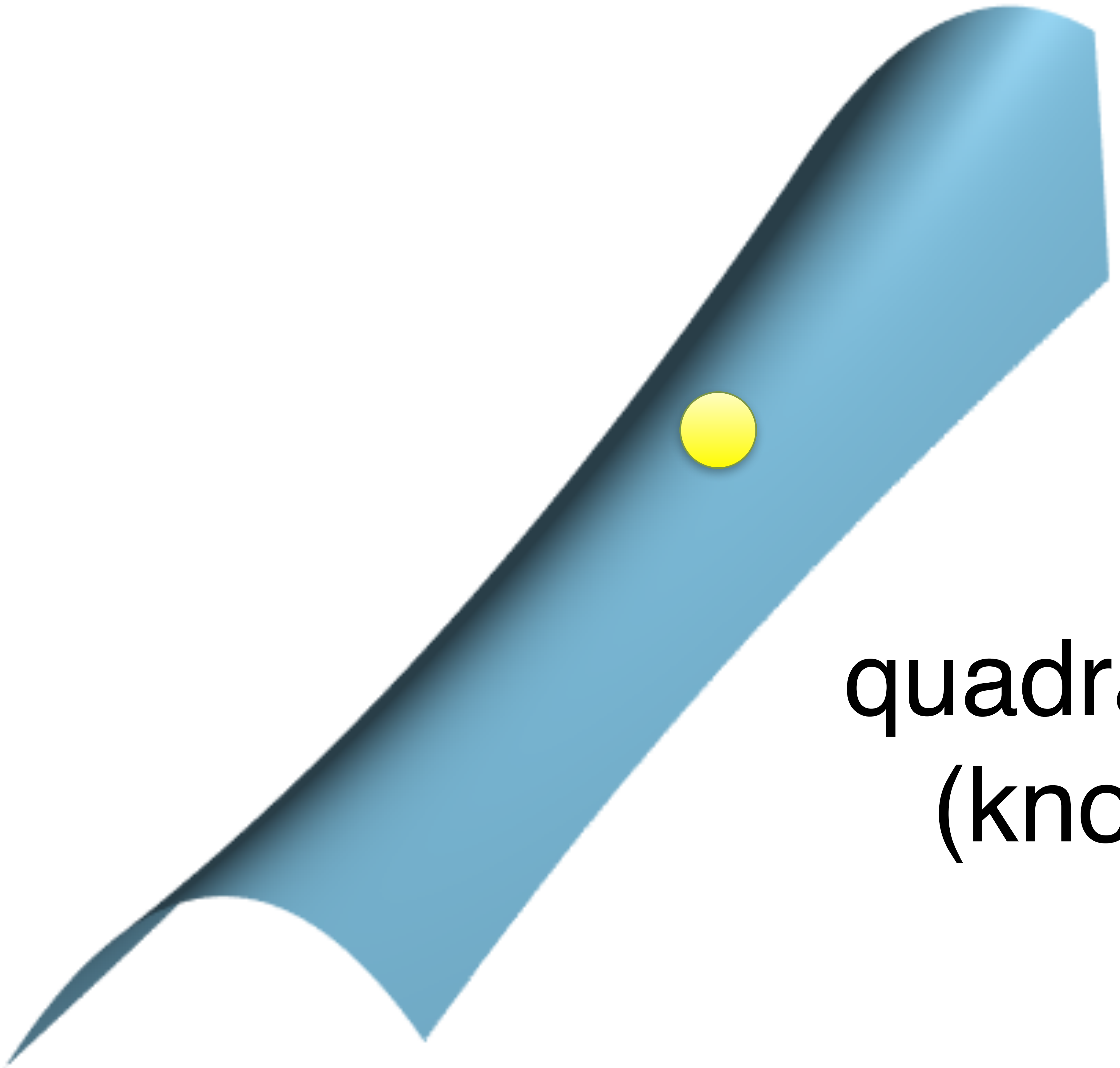
e.g. [Griewank and Walther 2008]

Quadratic approximation of contribution



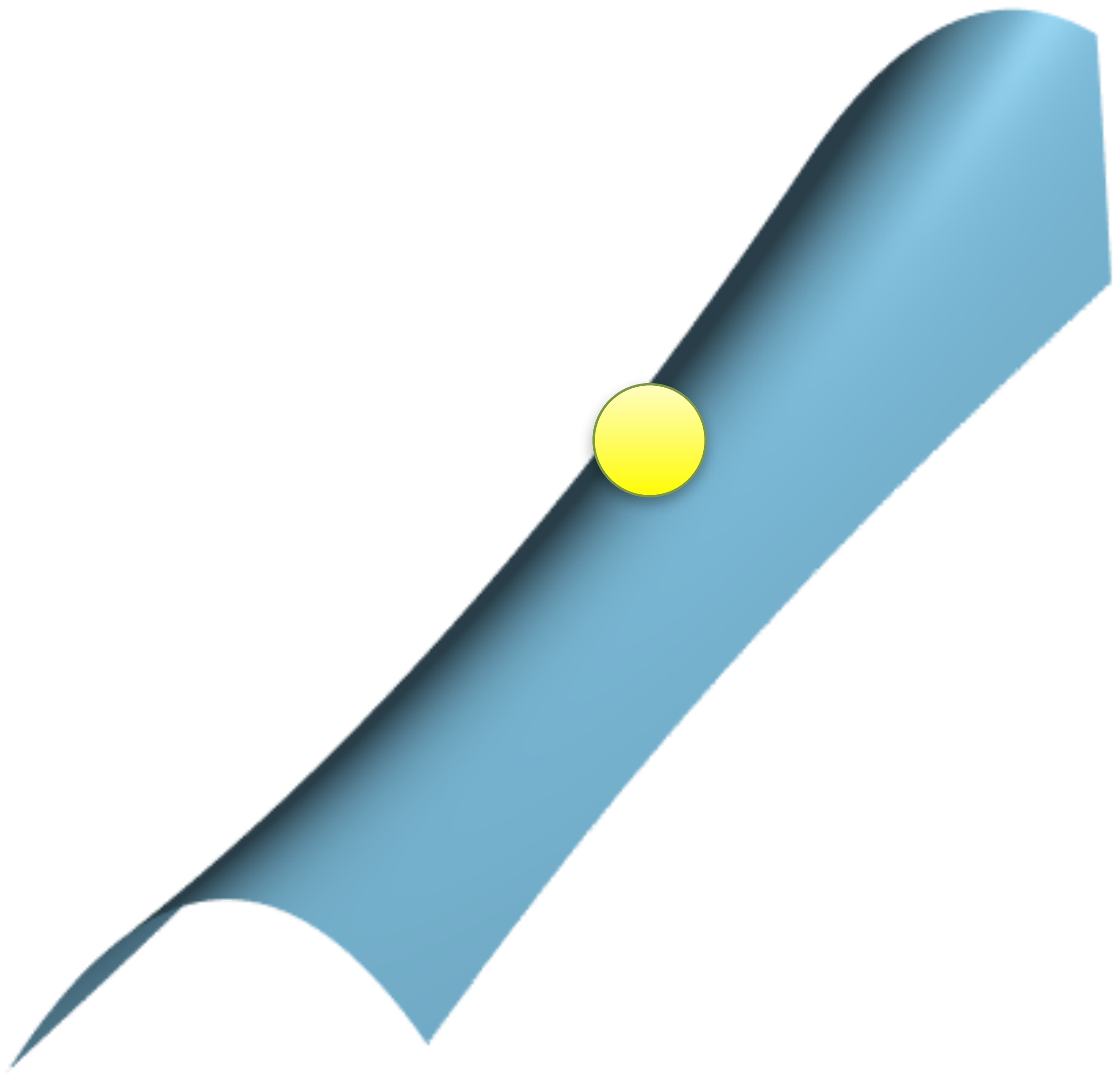
Quadratic approximation of contribution

gradient + Hessian (2nd-order Taylor)
around current sample

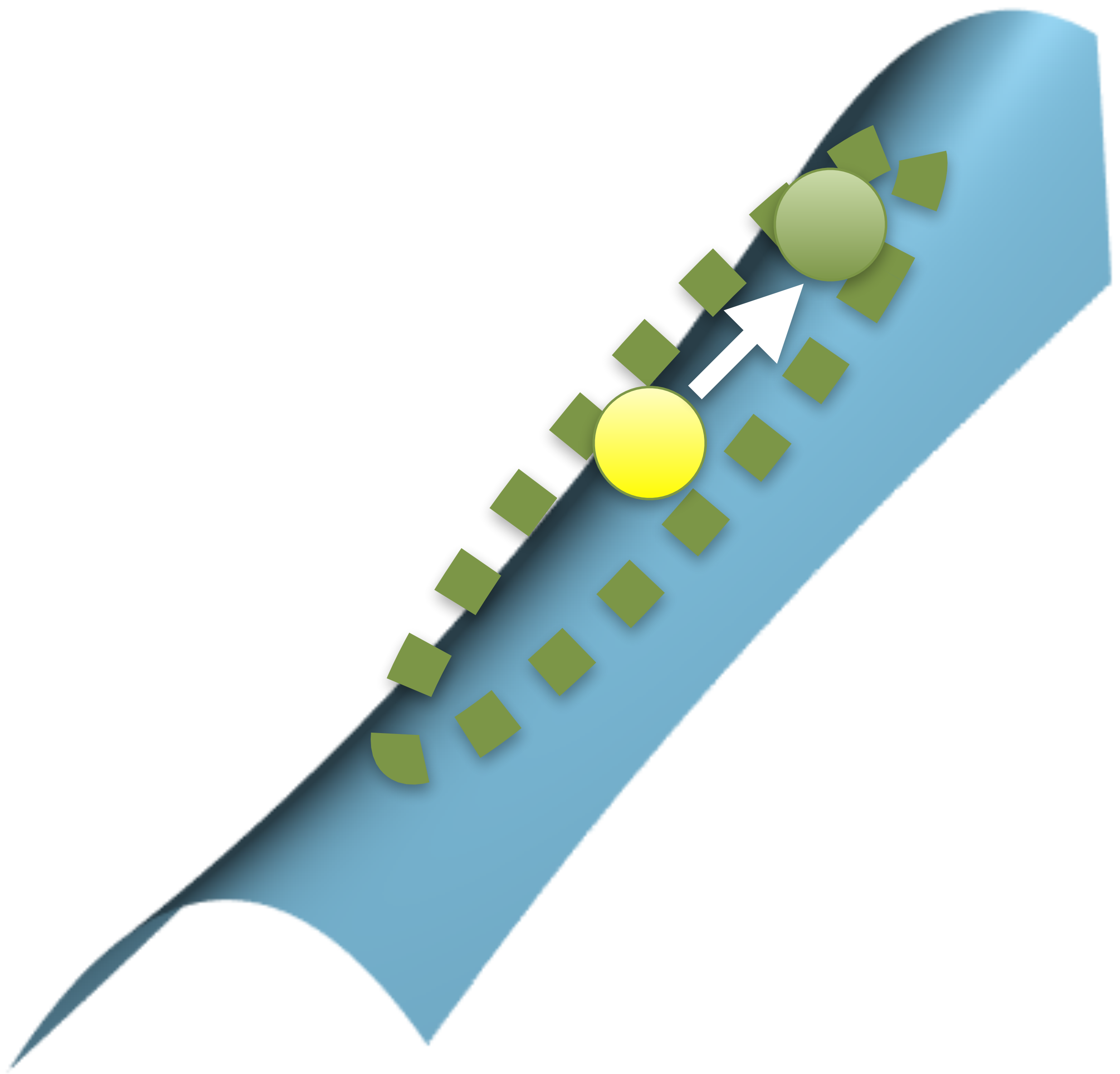


quadratic approximation
(known everywhere)

Recap

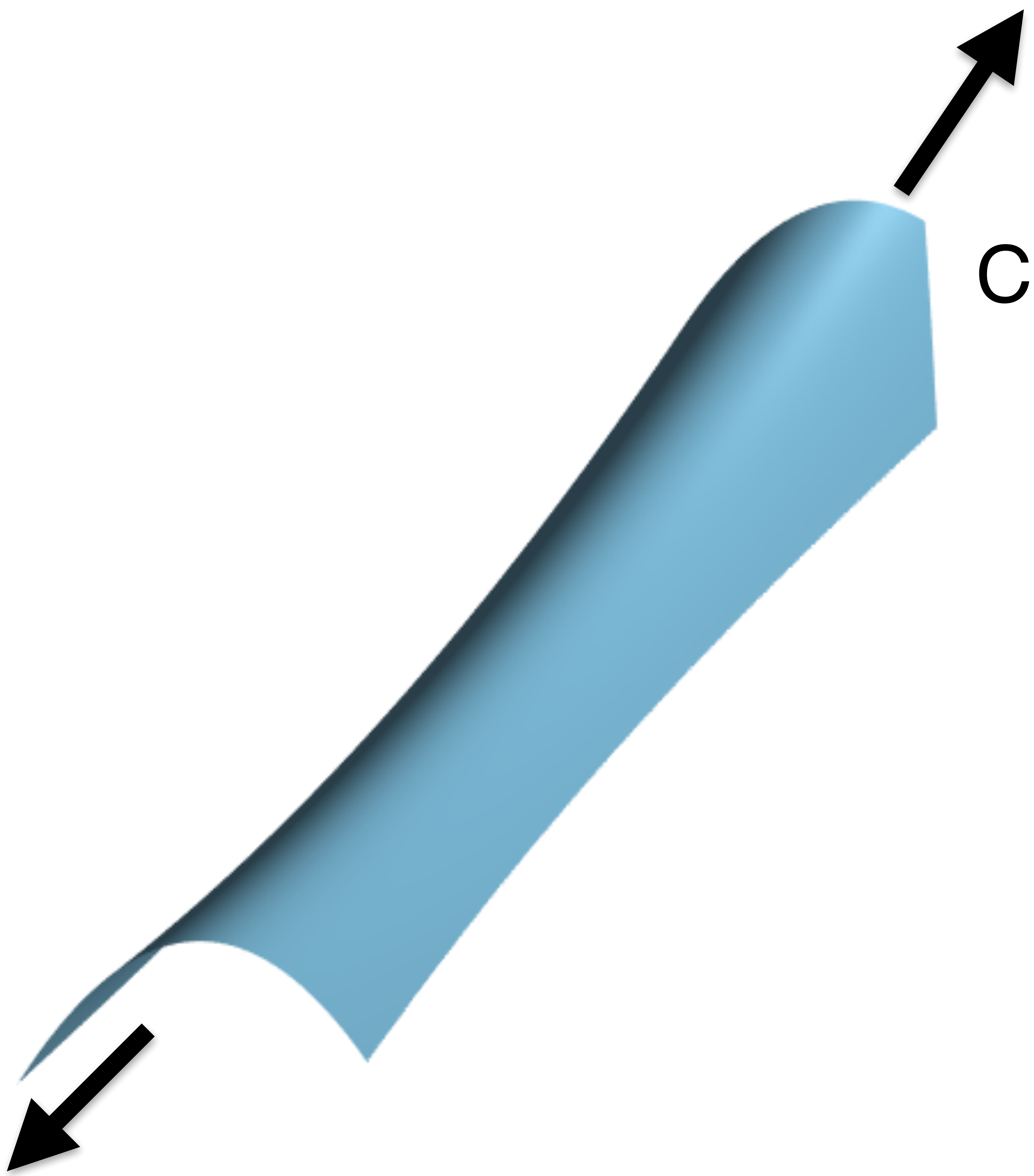


quadratic approximation
at current sample



challenge:
sample quadratic

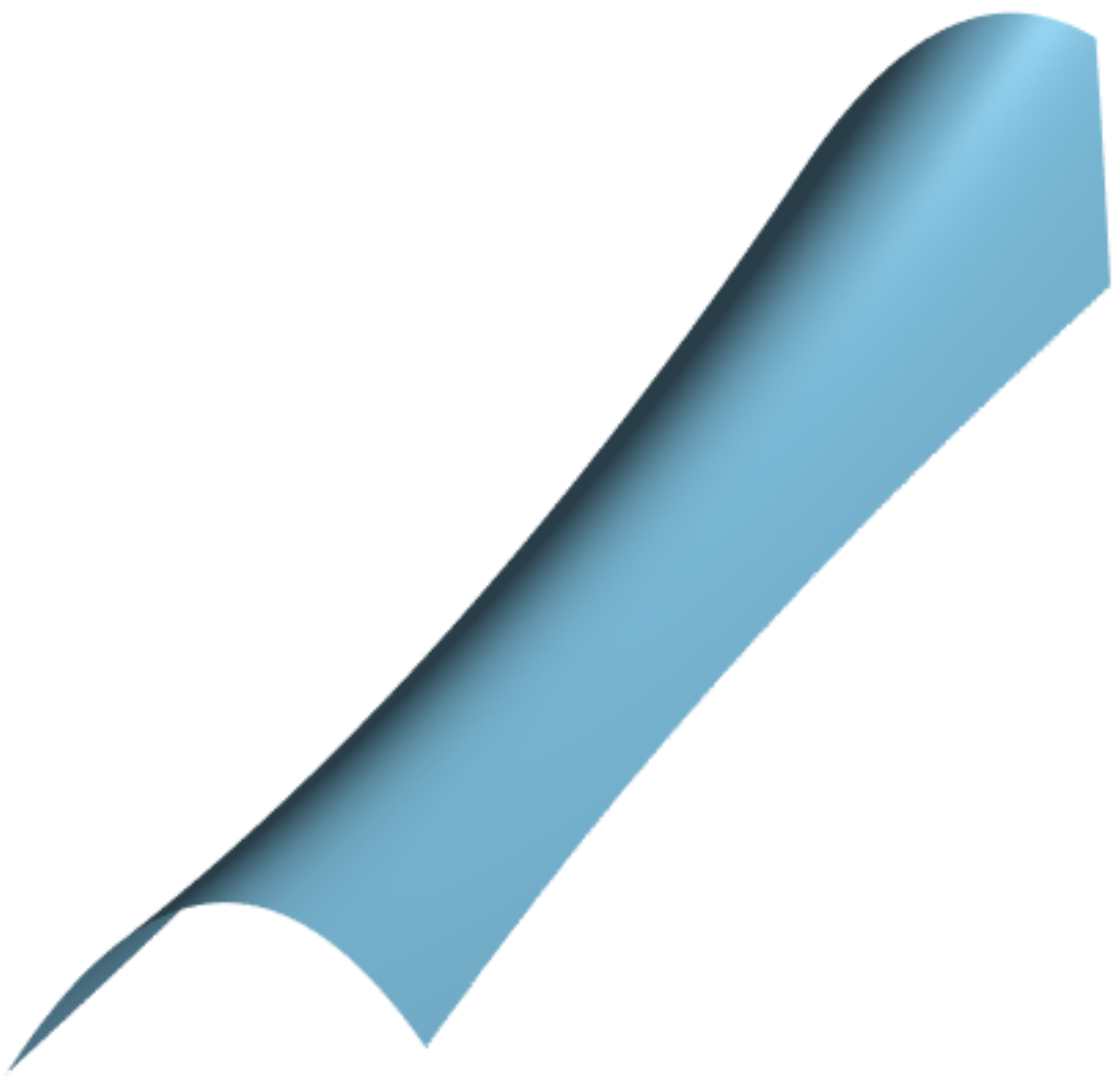
Quadratics are not distributions!



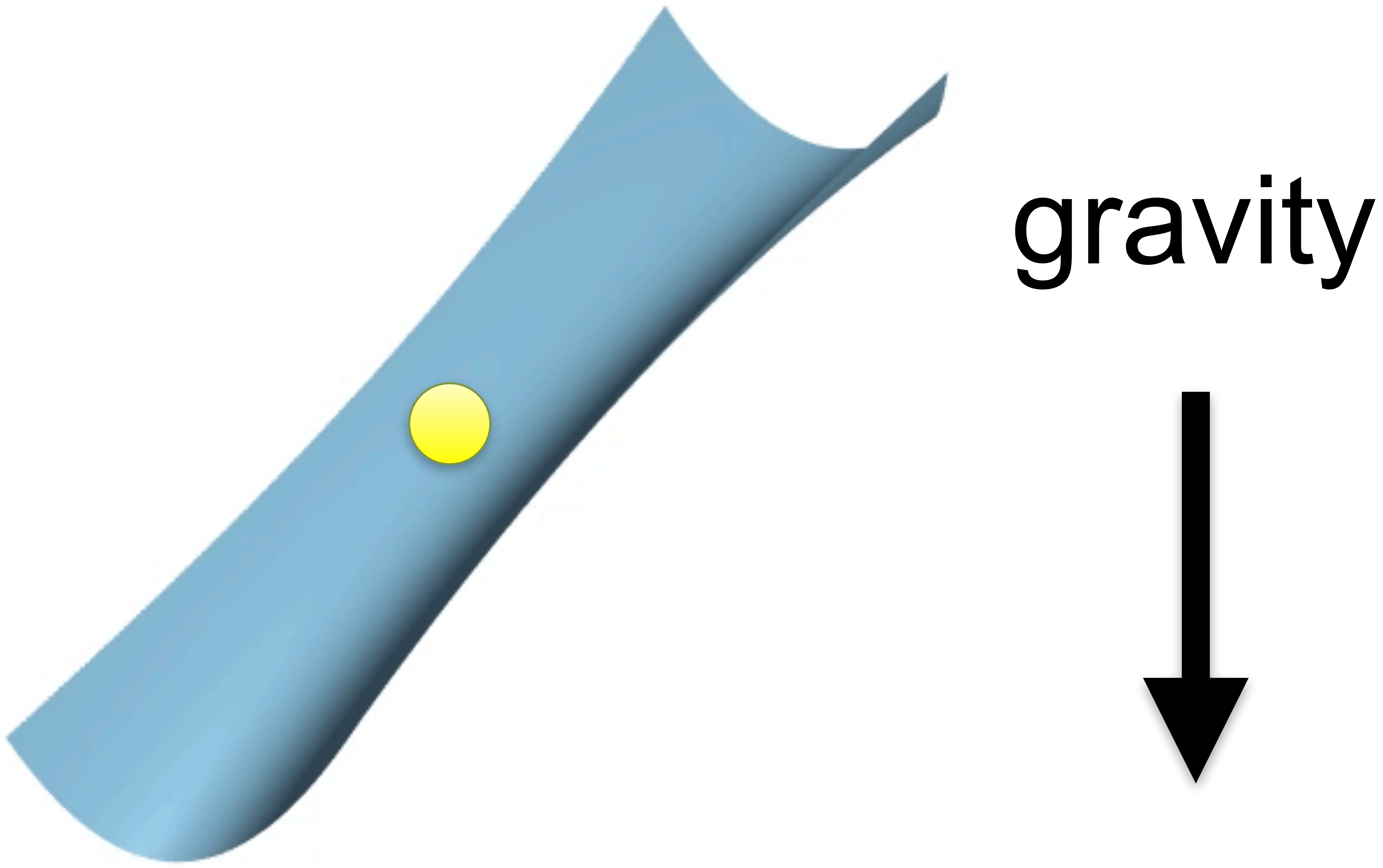
Can go to +/- infinity

Goal: attract samples to high contribution regions

- idea: flip landscape and simulate gravity
  - *Hamiltonian Monte Carlo [Duane et al. 1987]*



quadratic landscape

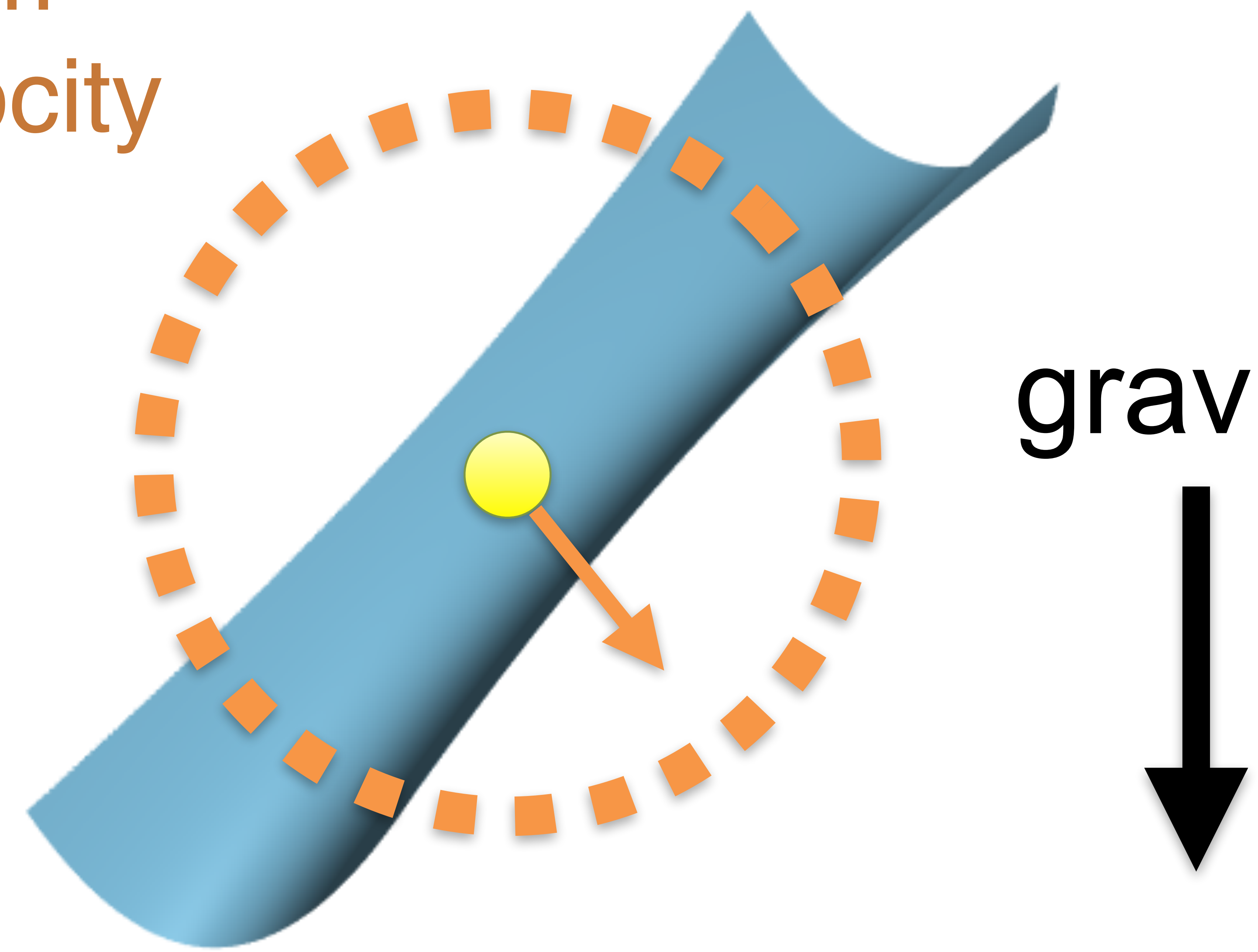


flipped quadratic landscape

Hamiltonian Monte Carlo simulates physics

- flip contribution landscape
- start from current sample with random velocity

Random
initial velocity

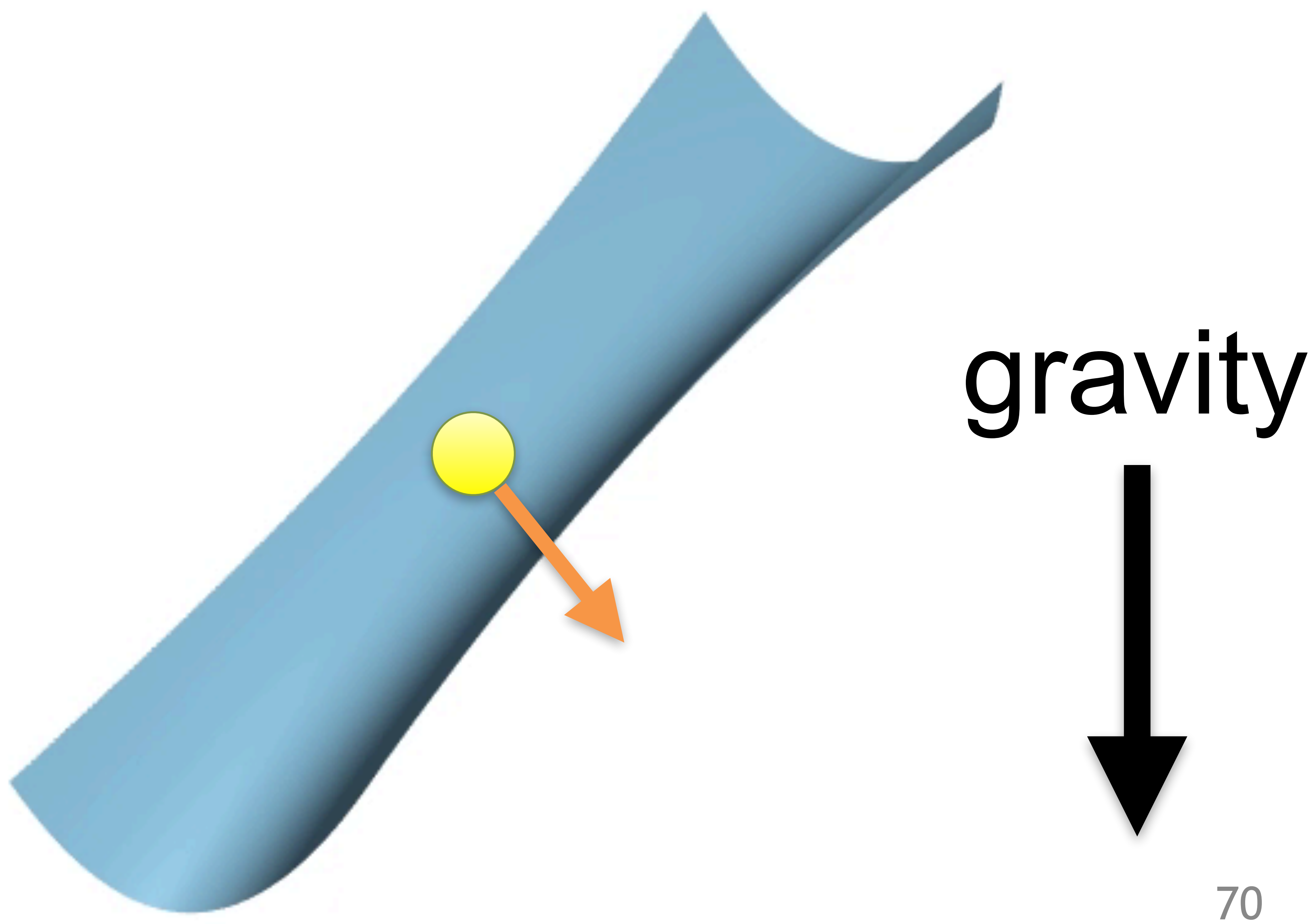


gravity

flipped quadratic
landscape

Hamiltonian Monte Carlo simulates physics

- flip contribution landscape
- start from current sample with random velocity
- simulate physics under gravity
  - particle is pulled to low ground (high contribution)
- proposal is final position

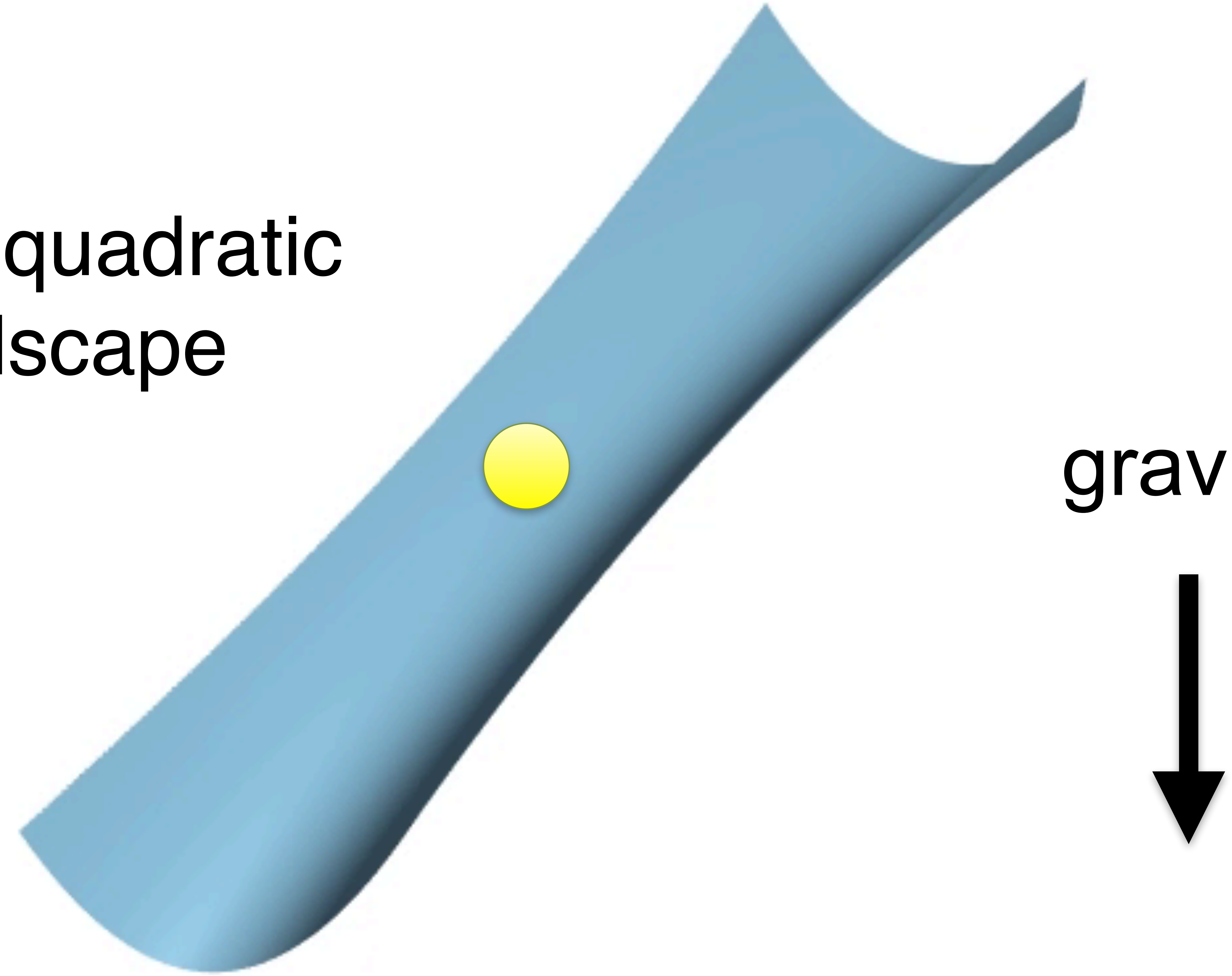


flipped quadratic
landscape

Challenge with traditional Hamiltonian Monte Carlo

expensive numerical simulation!

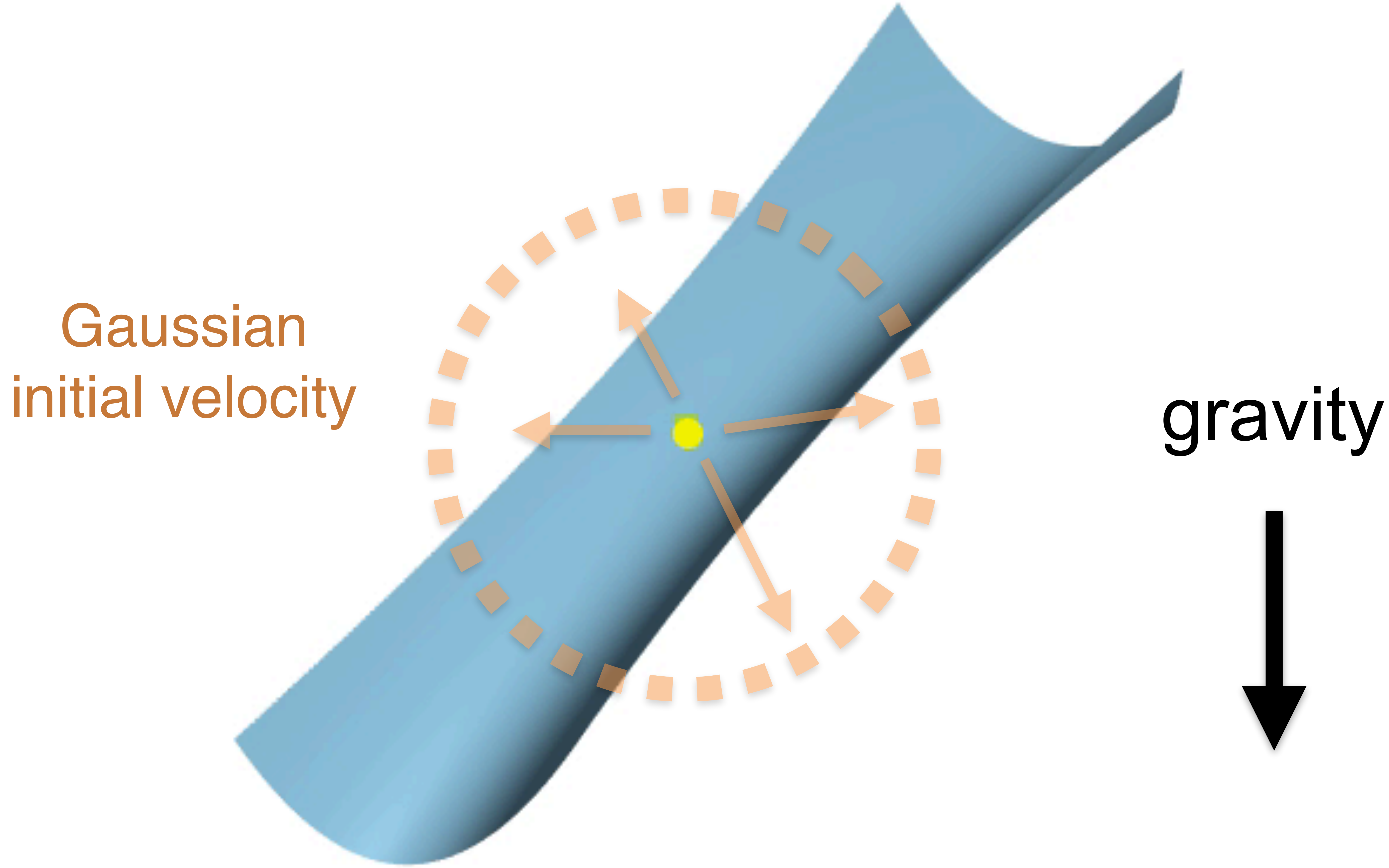
flipped quadratic landscape



gravity

HMC + quadratic has a closed form

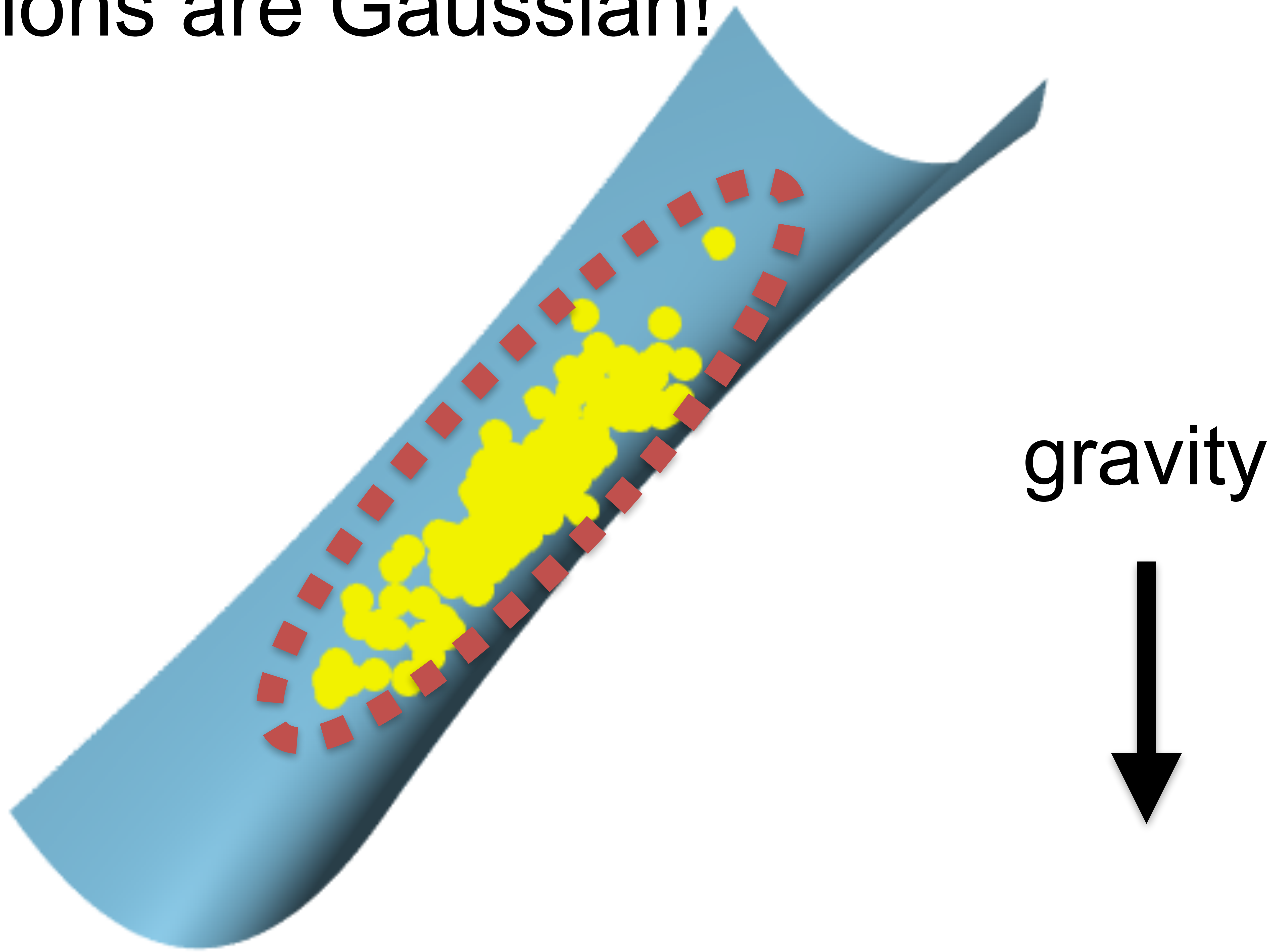
for Gaussian initial velocity



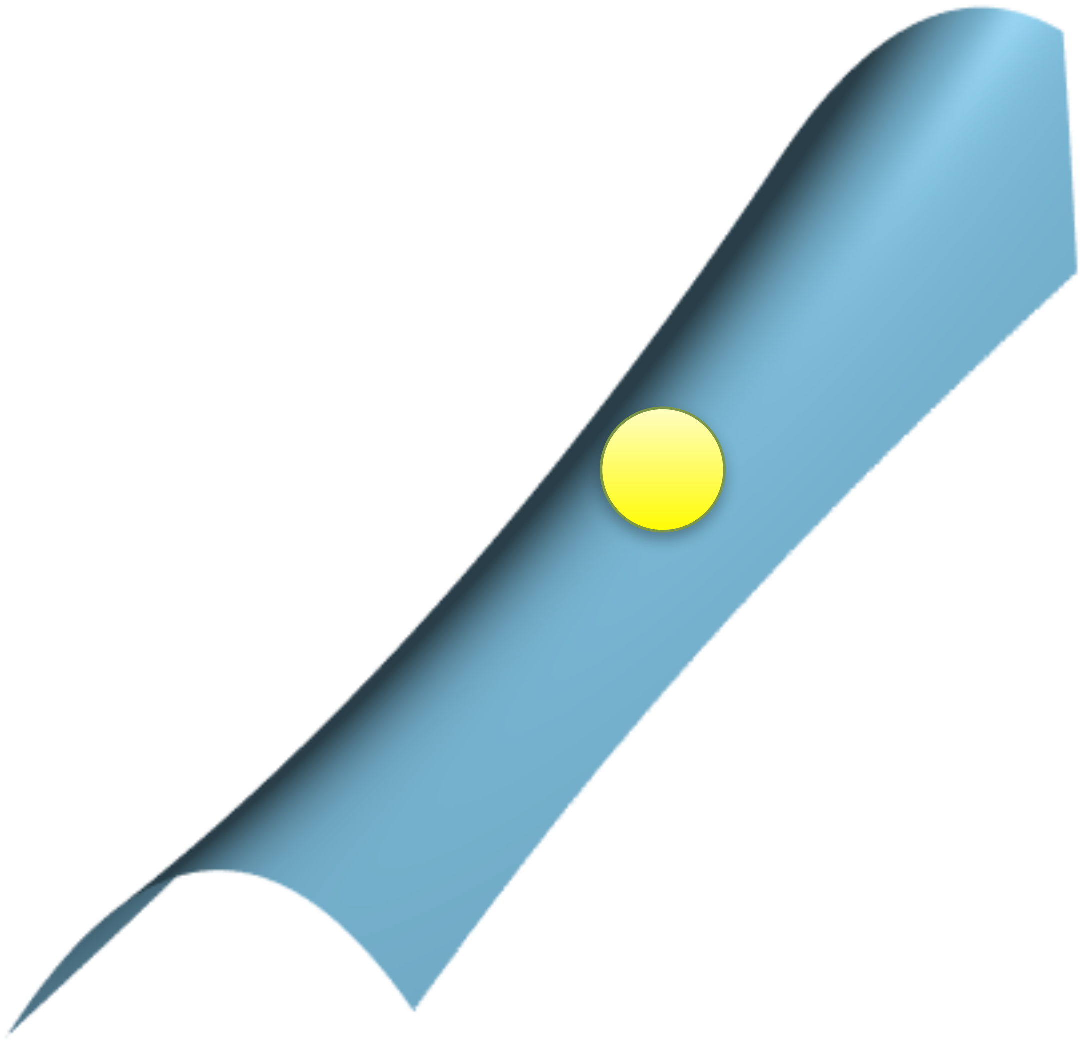
HMC + quadratic has a closed form

for Gaussian initial velocity

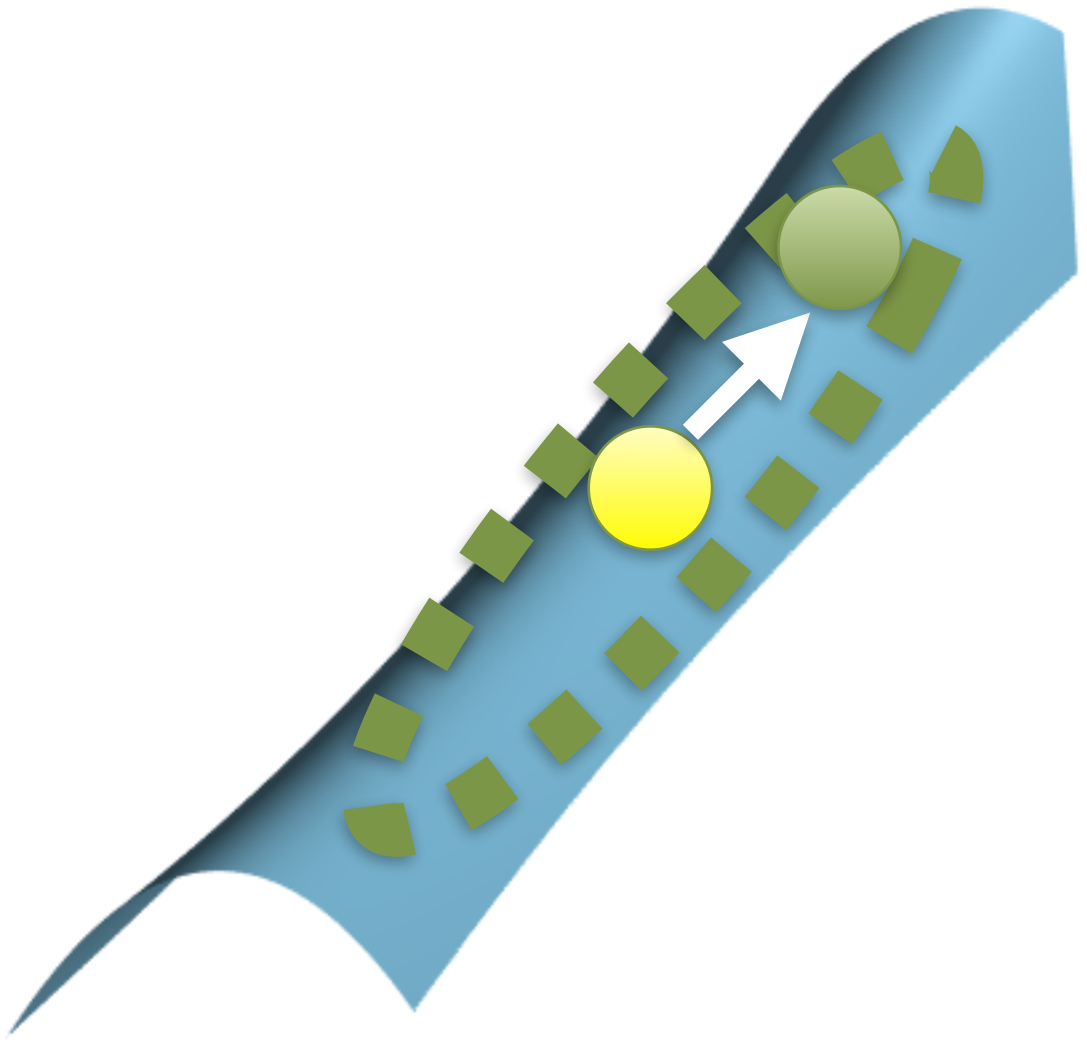
final positions are Gaussian!



Recap



use 2nd derivatives (Hessian)
to characterize anisotropy
→quadratic approximation

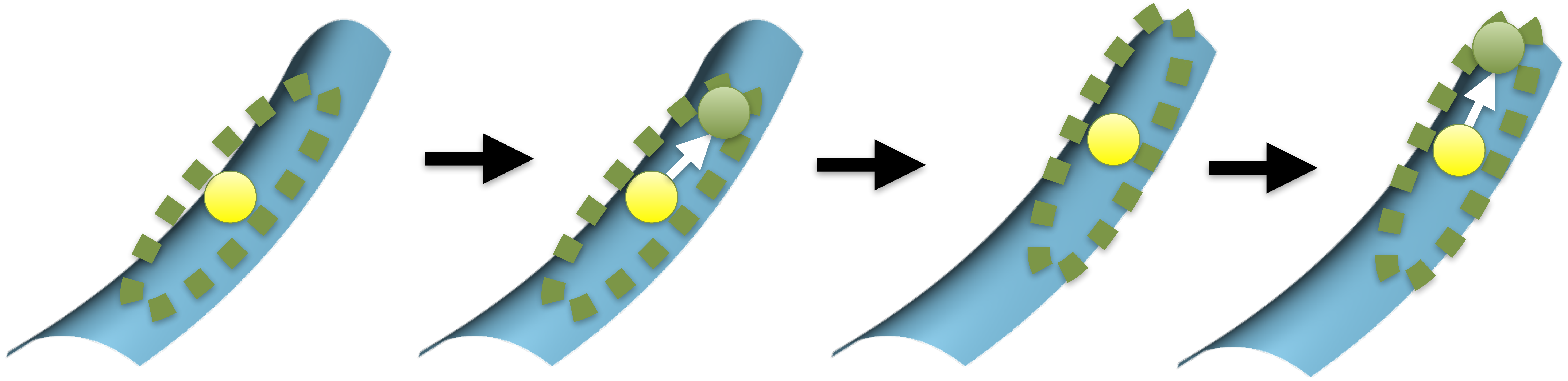


simulate Hamiltonian dynamics
to sample from quadratics

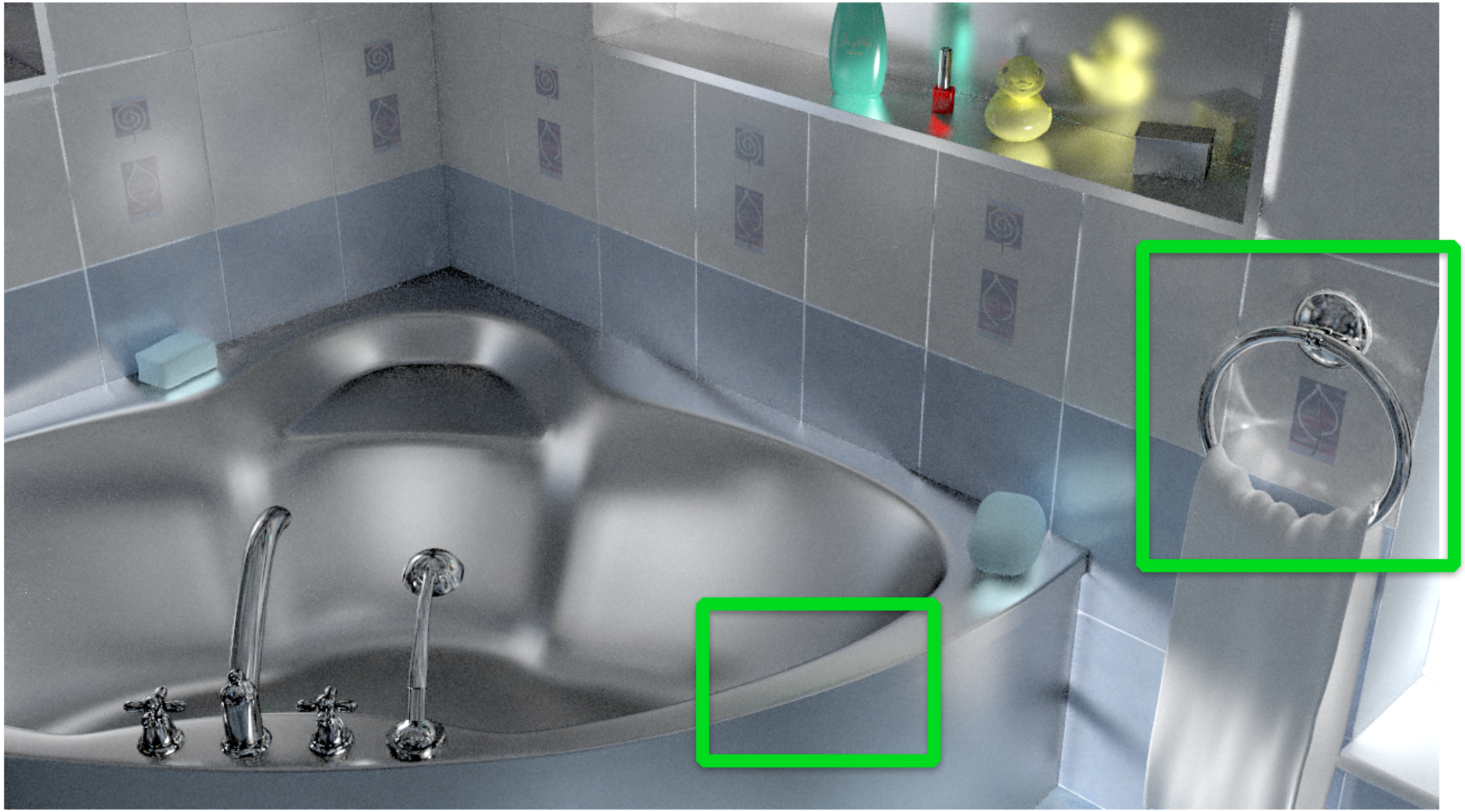
results in closed-form Gaussian

Recap

```
Given current sample
 compute gradient and Hessian
 compute anisotropic Gaussian
 draw proposal
 probabilistically accept
 repeat
```

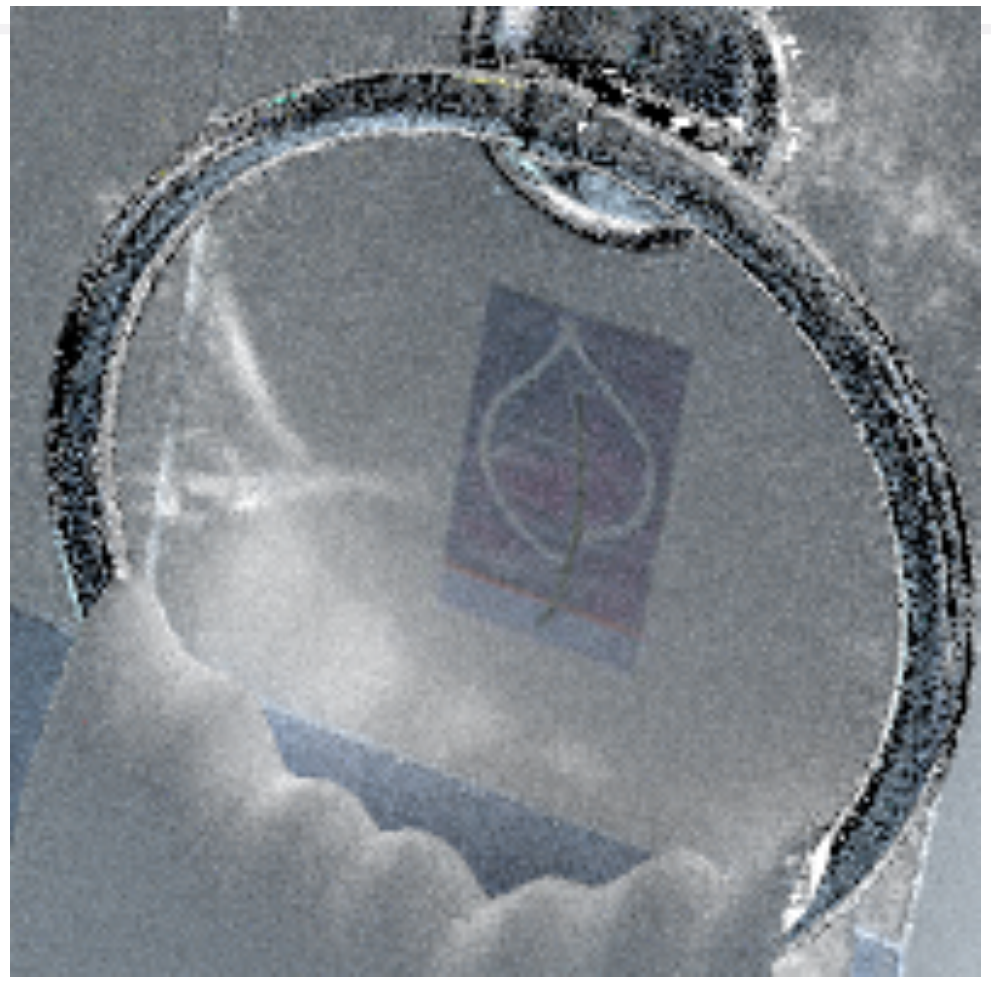


Results: Bathroom

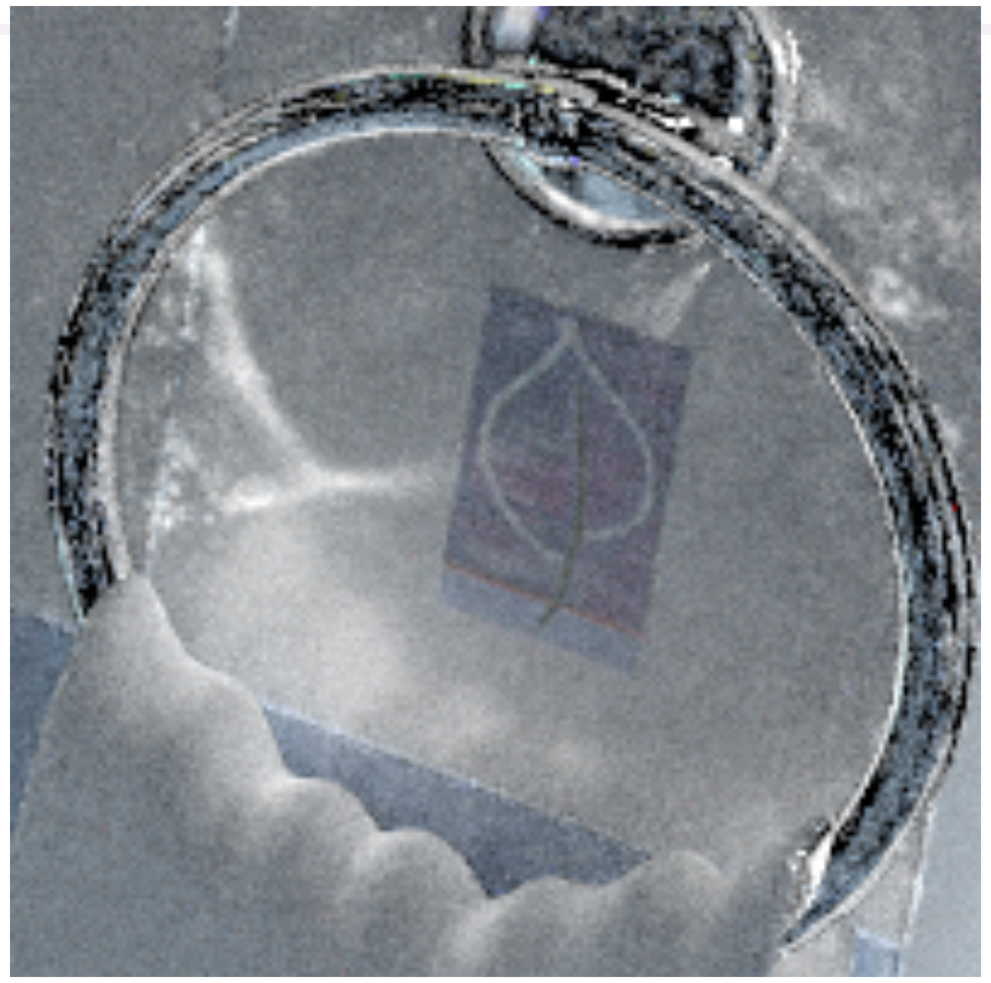


Bathroom: equal-time (10 mins) comparisons

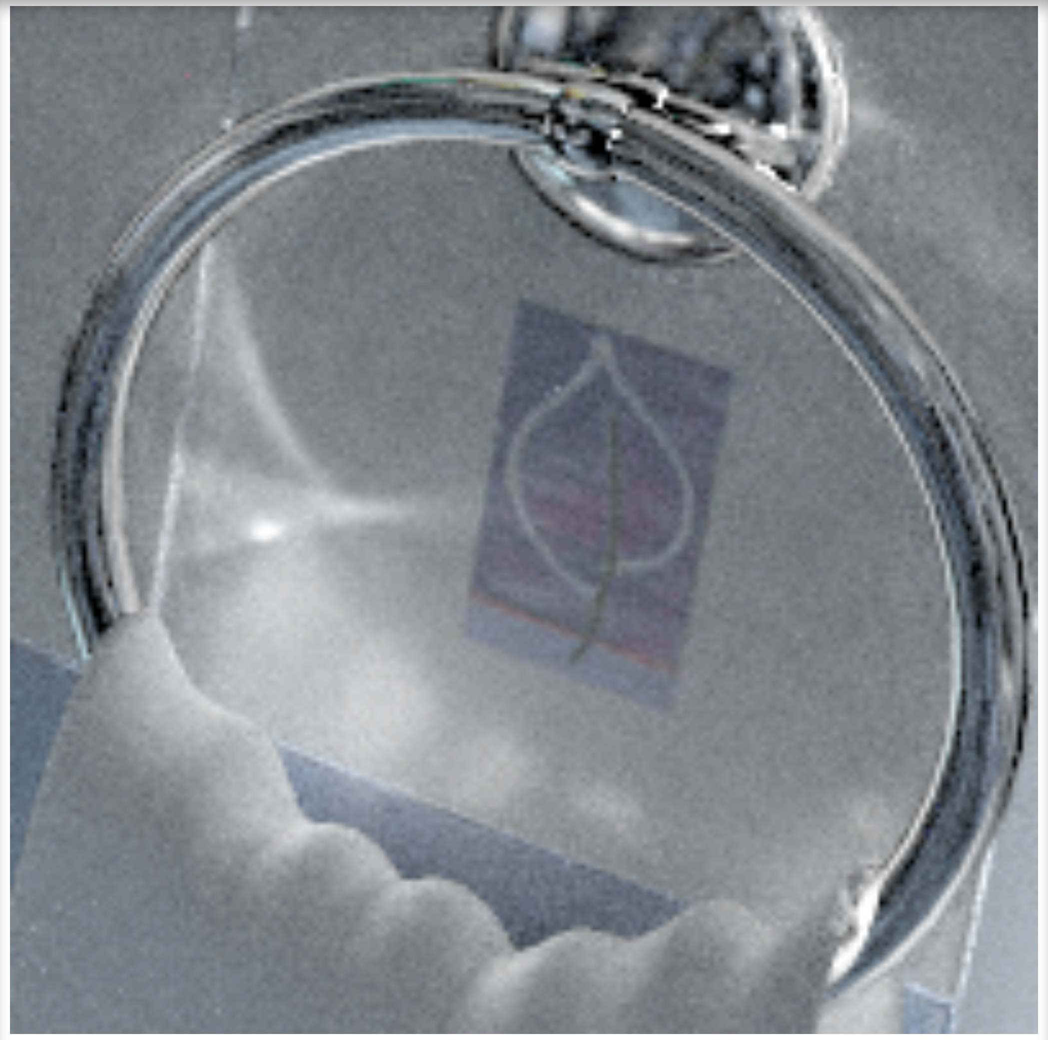
MMLT
[Hachisuka 2014]



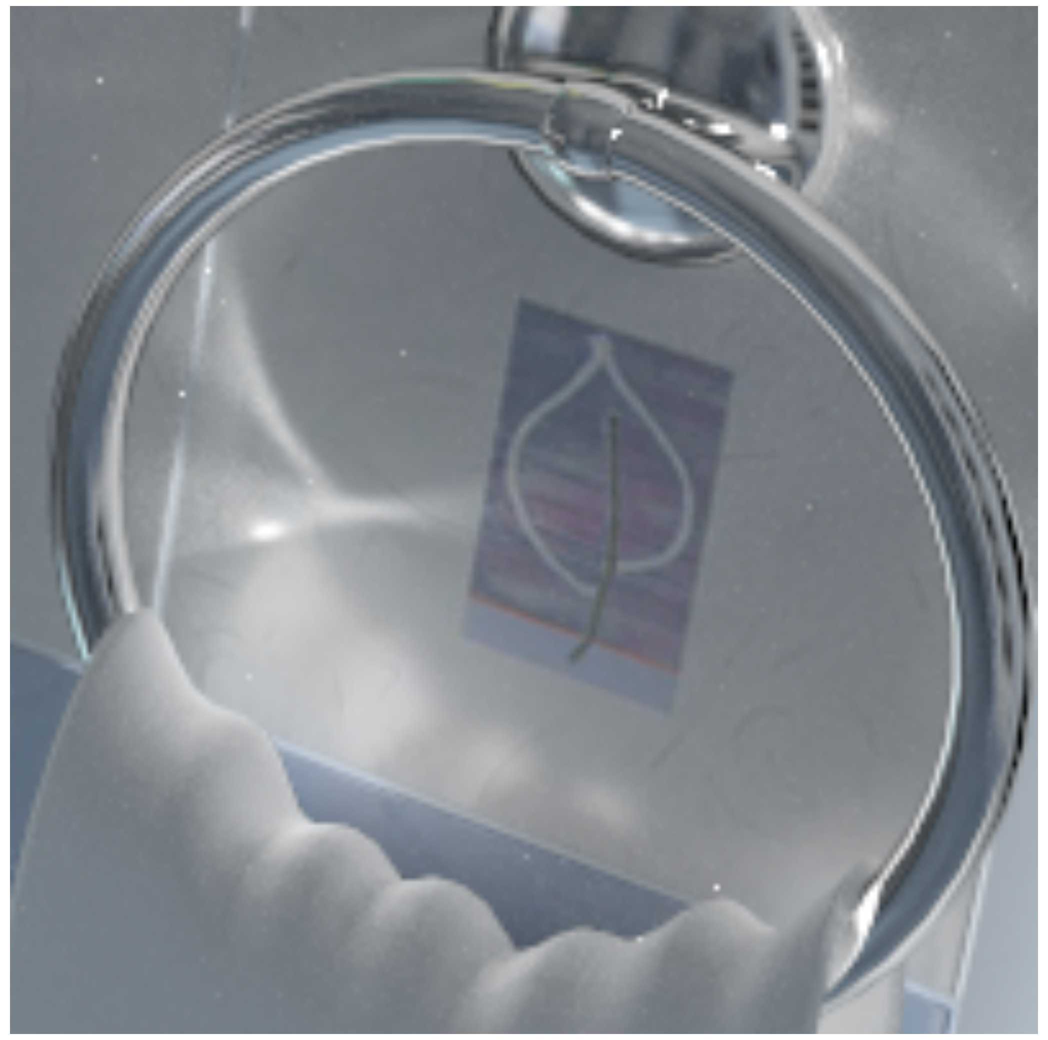
MEMLT
[Jakob 2012]



HSLT
[Kaplanyan 2014, Hanika 2015]



OURS



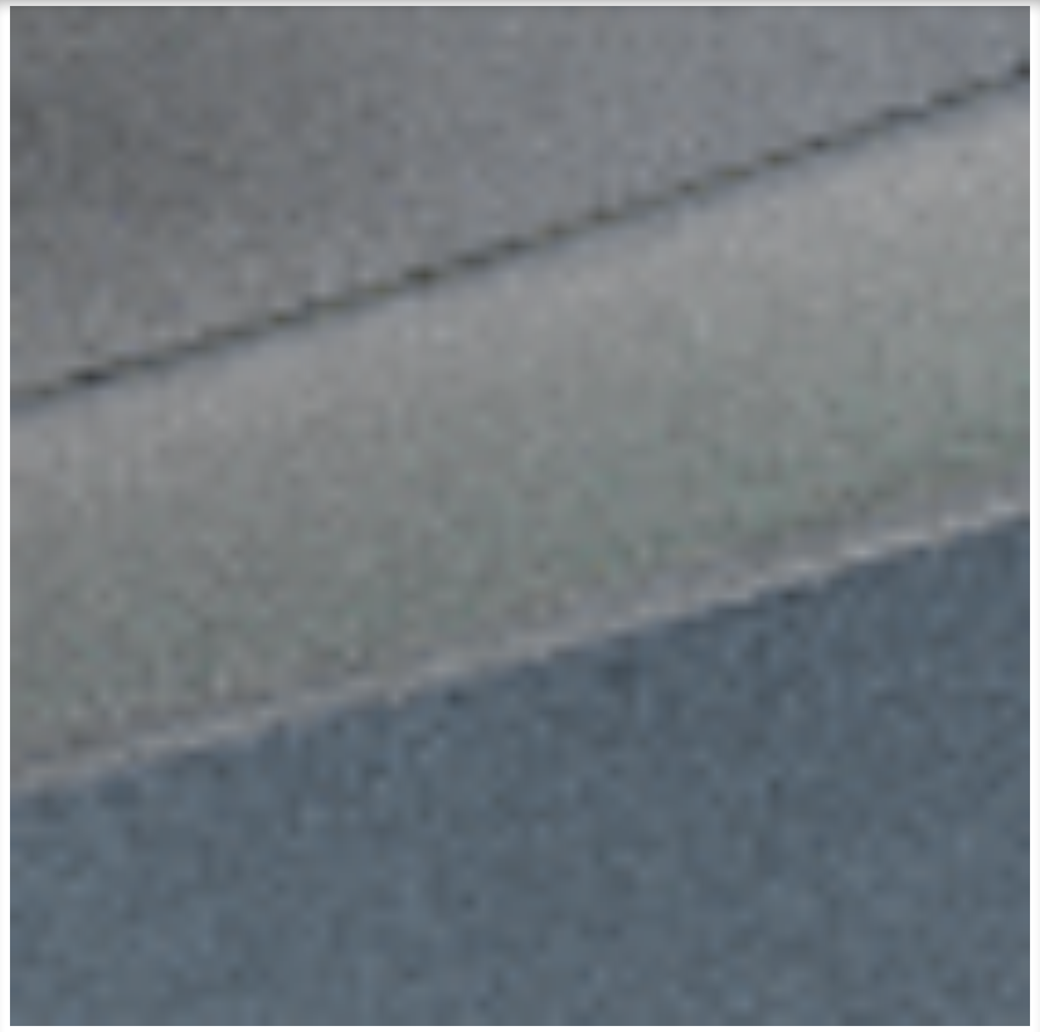
Reference (2 days)

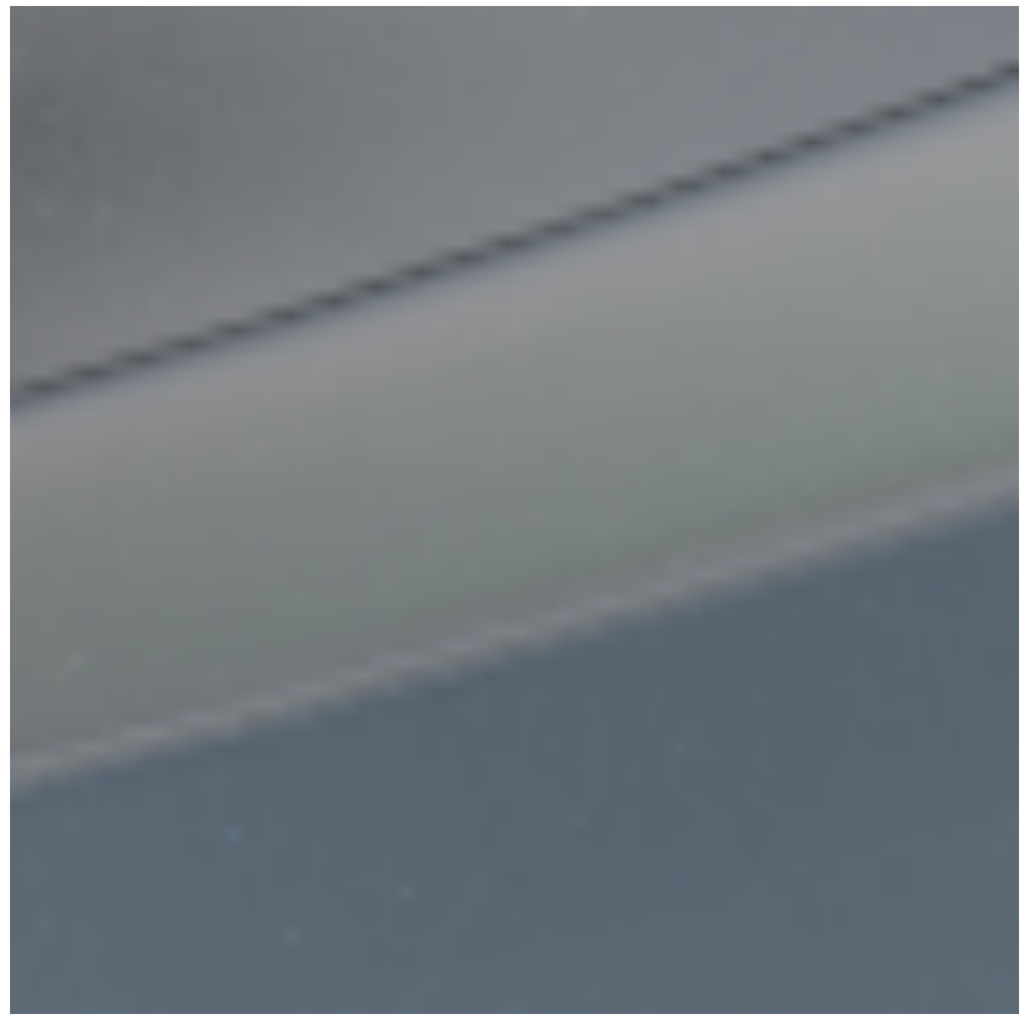
Bathroom: equal-time (10 mins) comparisons


MMLT
[Hachisuka 2014]


MEMLT
[Jakob 2012]

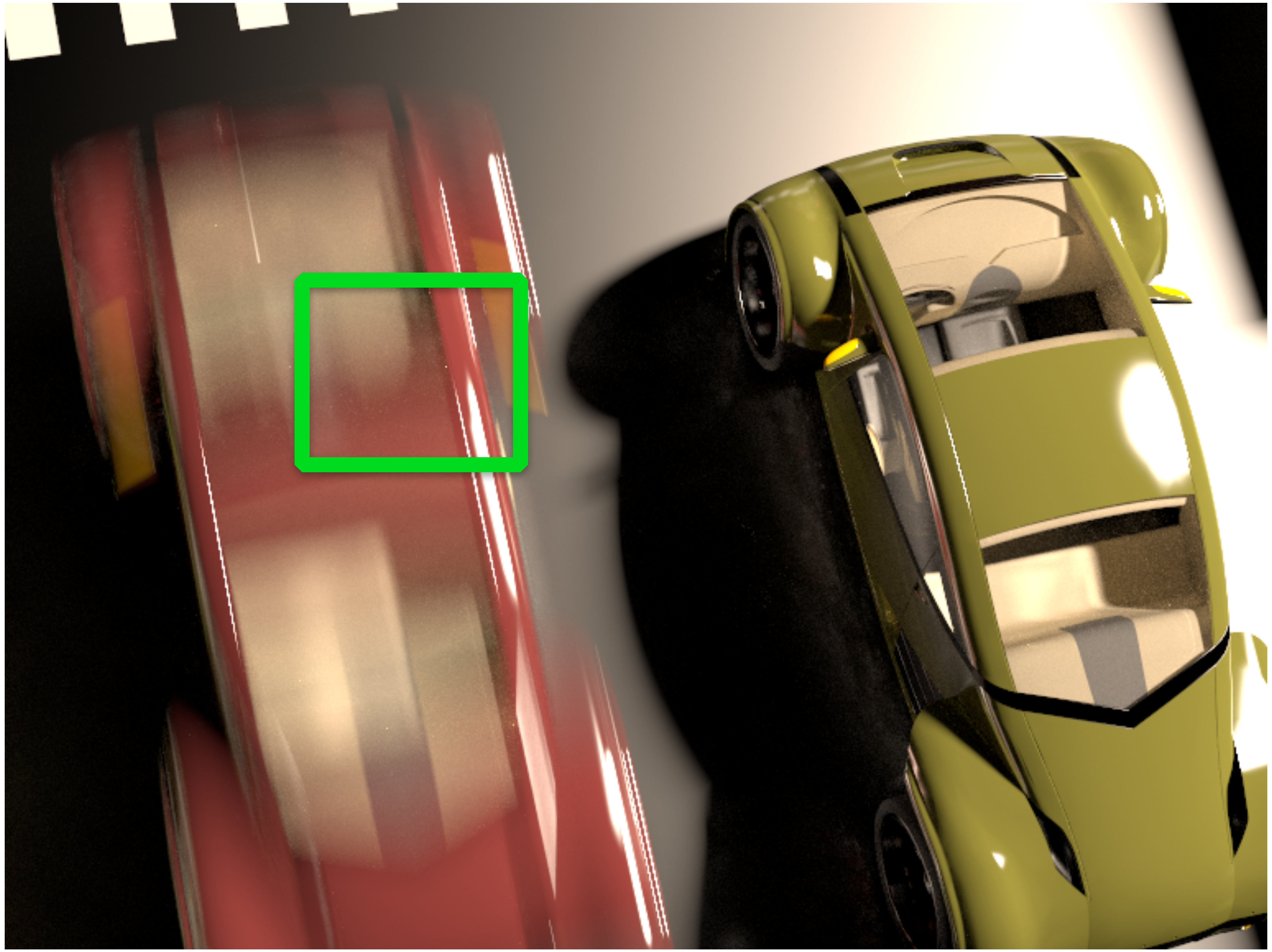

HSLT
[Kaplanyan 2014, Hanika 2015]


OURS

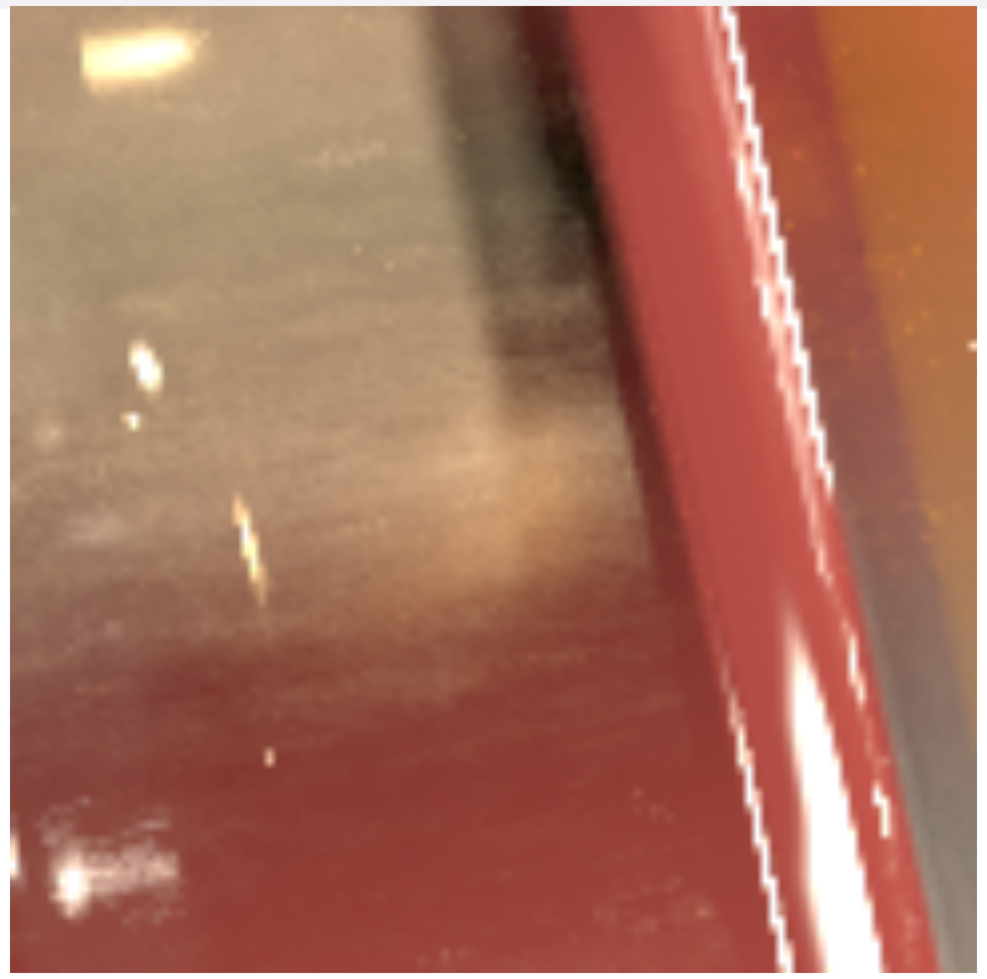

Reference (2 days)

Extension to time

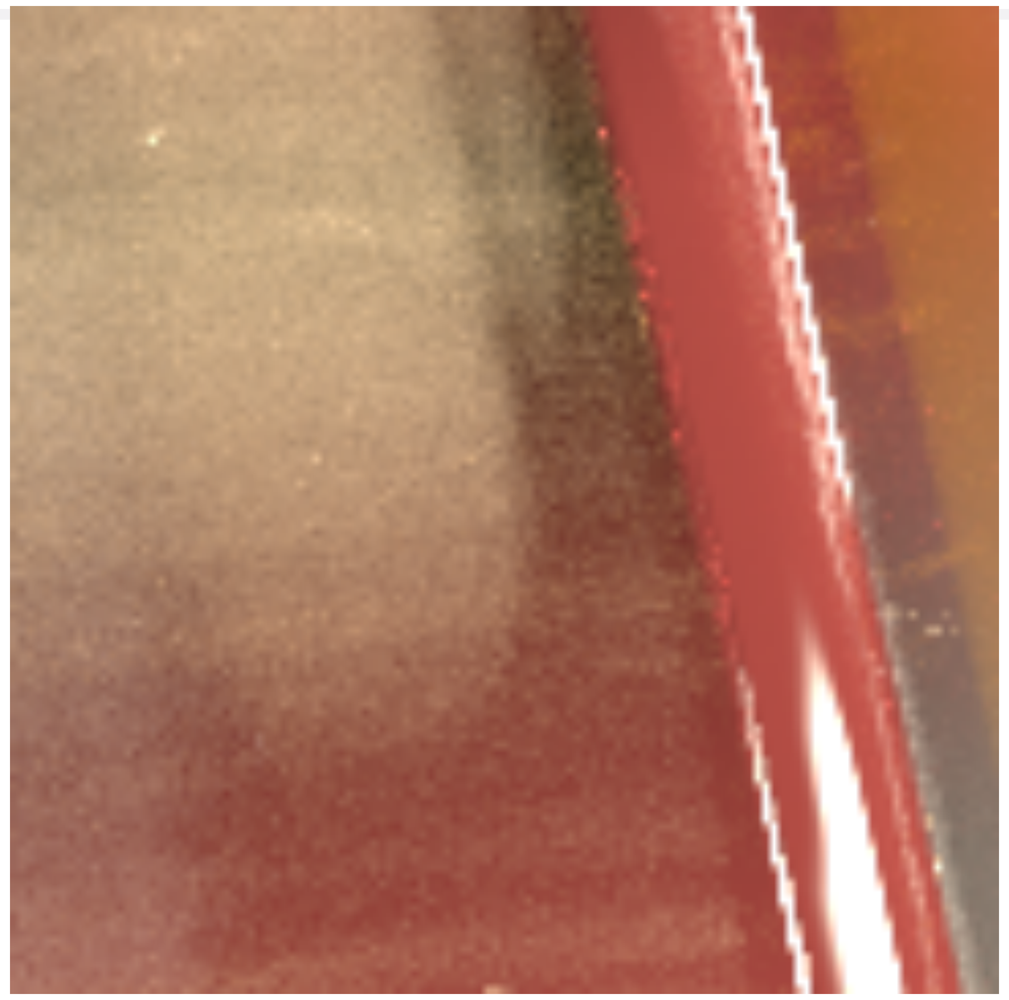
Our method is general thanks to automatic differentiation



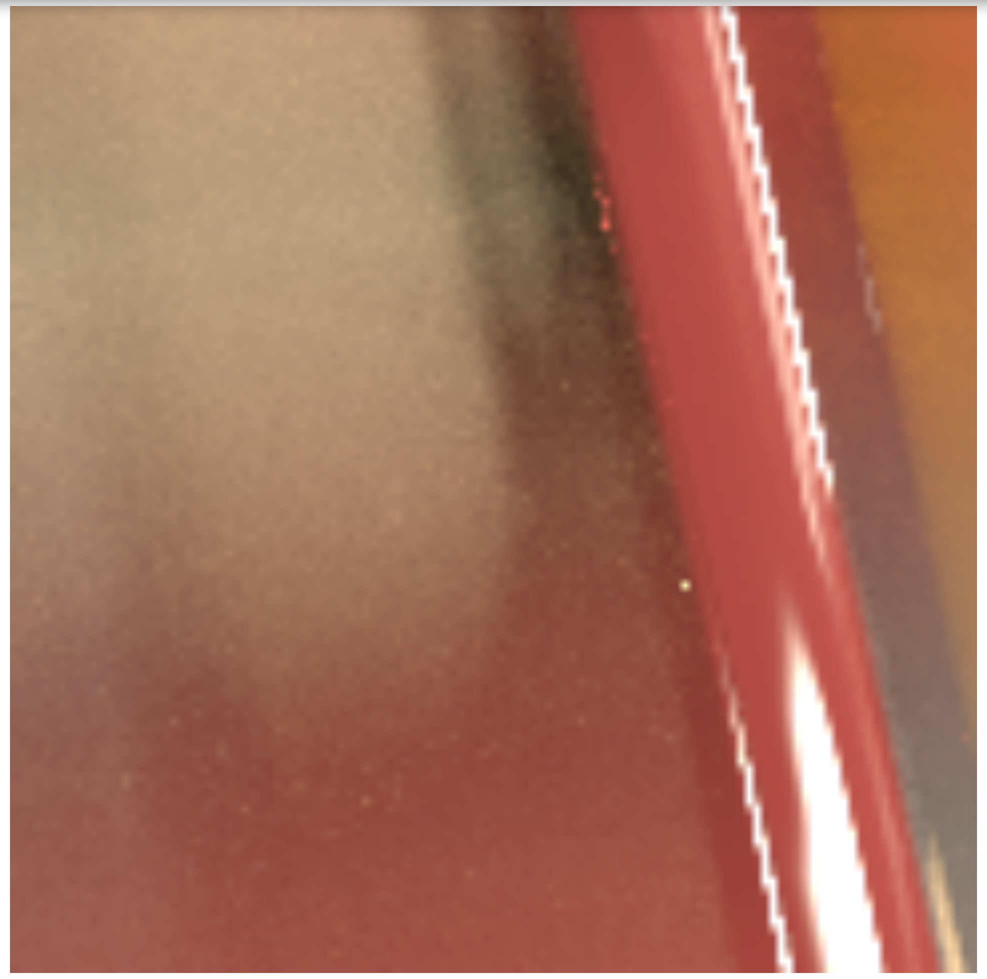
Cars: equal-time (20 mins) comparisons



MMLT
[Hachisuka 2014]

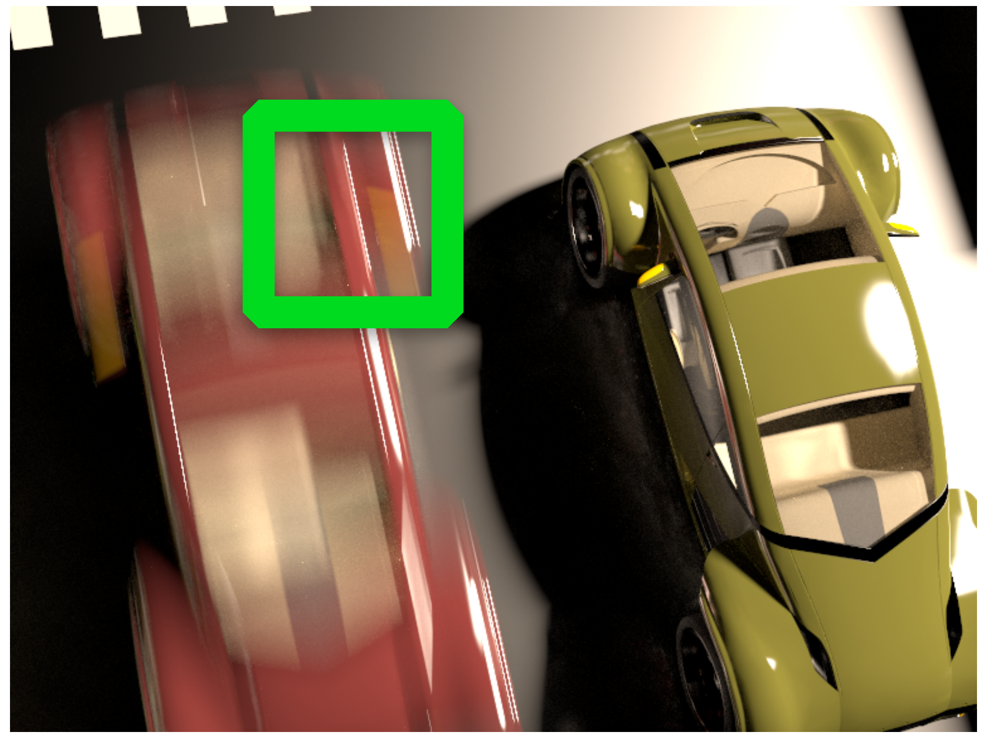


MEMLT
[Jakob 2012]



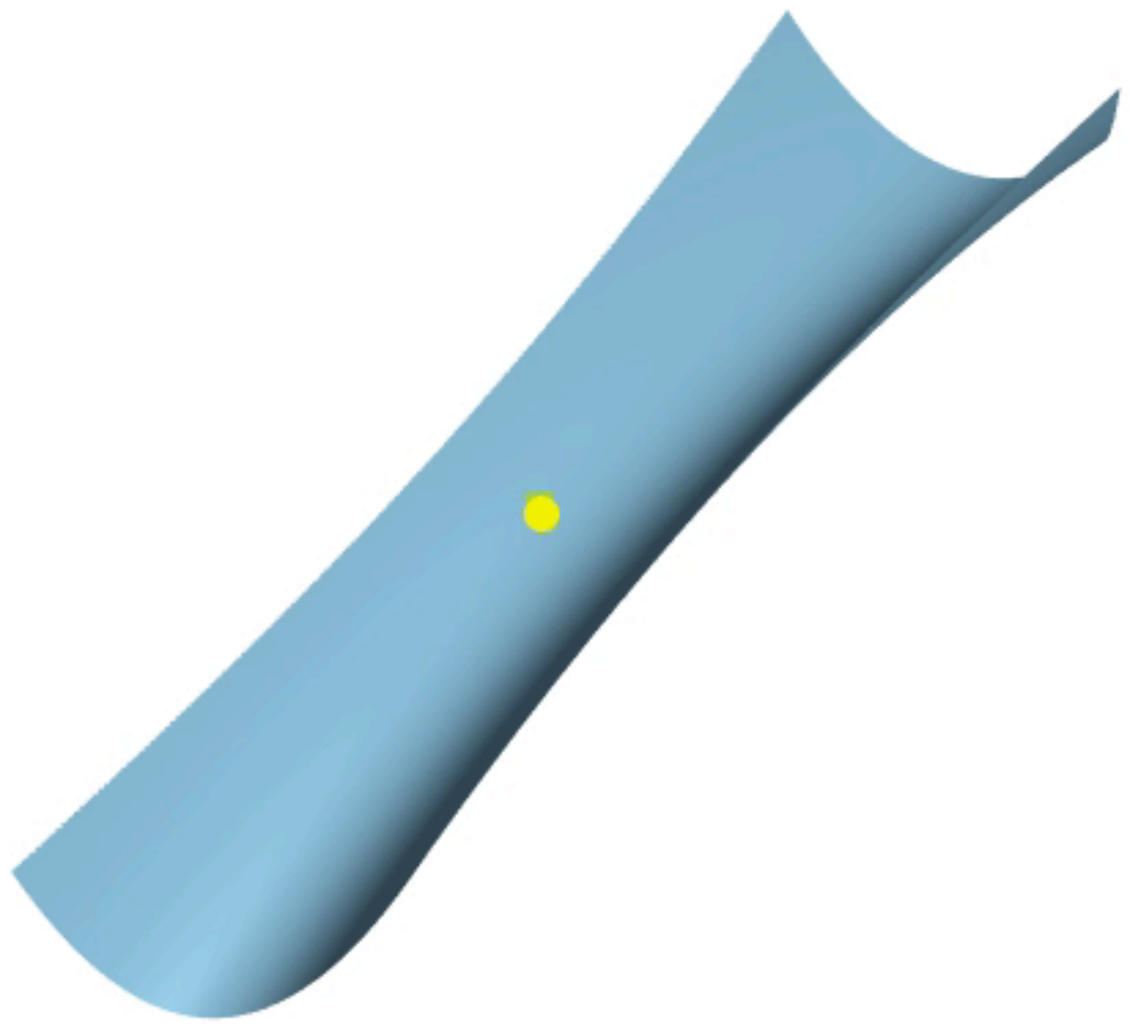
OURS

Reference (12 hours)



Conclusion

- Good anisotropic proposals for Metropolis
  - Hessian from automatic differentiation
  - Hamiltonian Monte Carlo
  - Closed-form Gaussian
  - General, easily extended to time



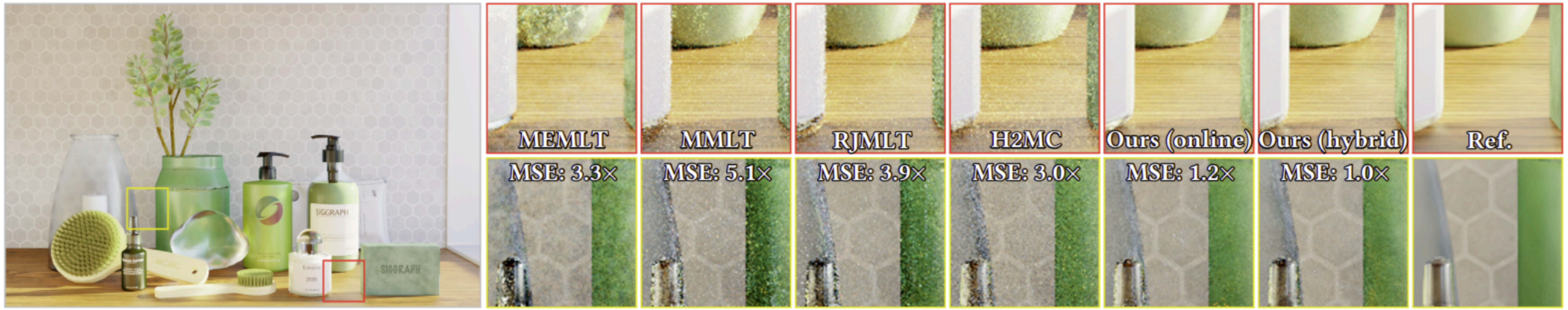
Hessian might not be necessary!

- use an Adam like algorithm to guide sampling

Langevin Monte Carlo Rendering with Gradient-based Adaptation

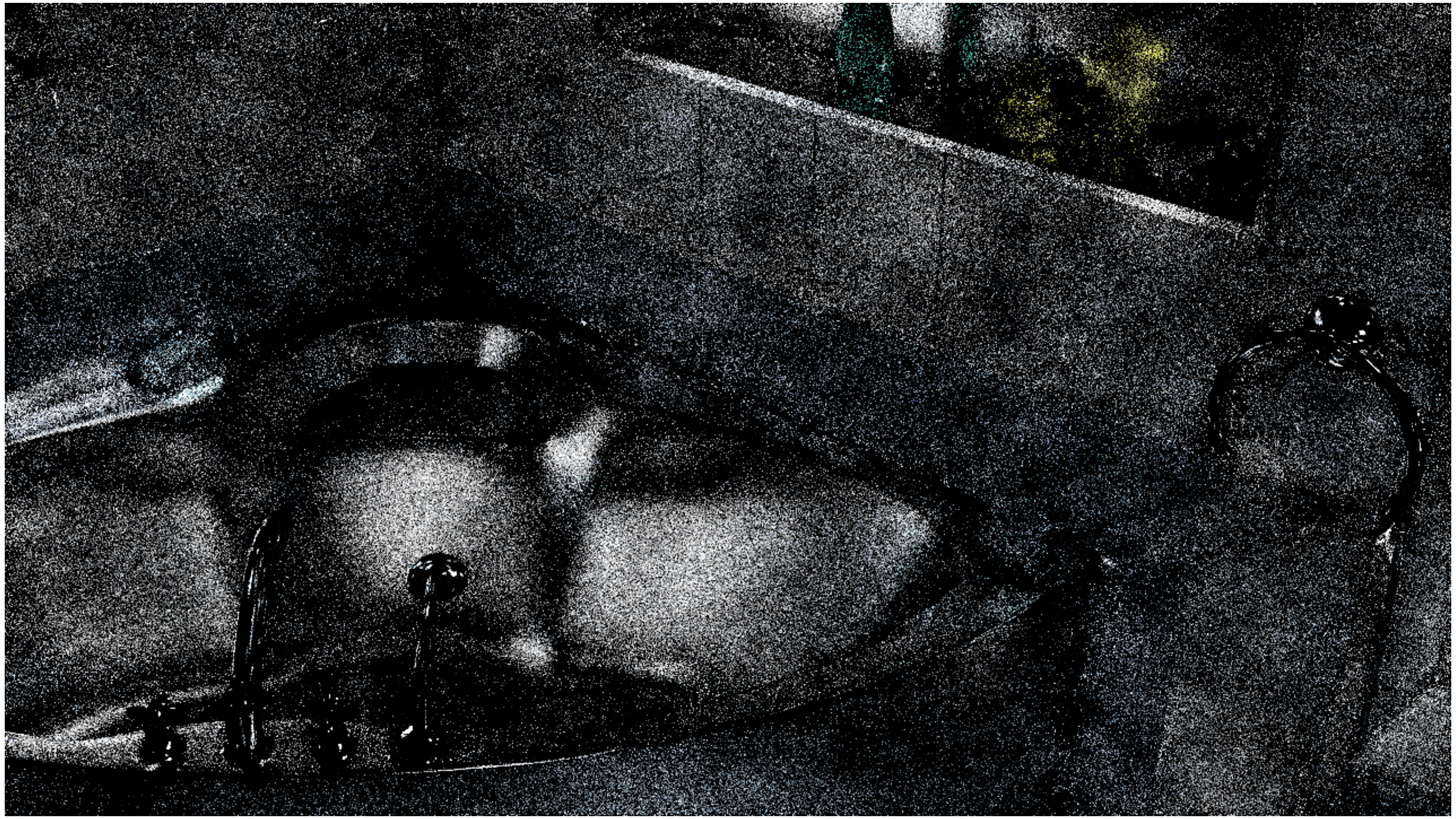
SIGGRAPH 2020

Fujun Luan — Cornell University
Shuang Zhao — University of California, Irvine
Kavita Bala — Cornell University
Ioannis Gkioulekas — Carnegie Mellon University



Open problem with MLT: global exploration

- large steps/bidirectional mutation usually have very low acceptance rate (1-2%)
- lead to uneven convergence & unstable results



Next: specular light path sampling