MCMC: Bridging rendering, optimization and generative AI

Gurprit Singh

MAX PLANCK INSTITUTE
FOR INFORMATICS

Wenzel Jakob

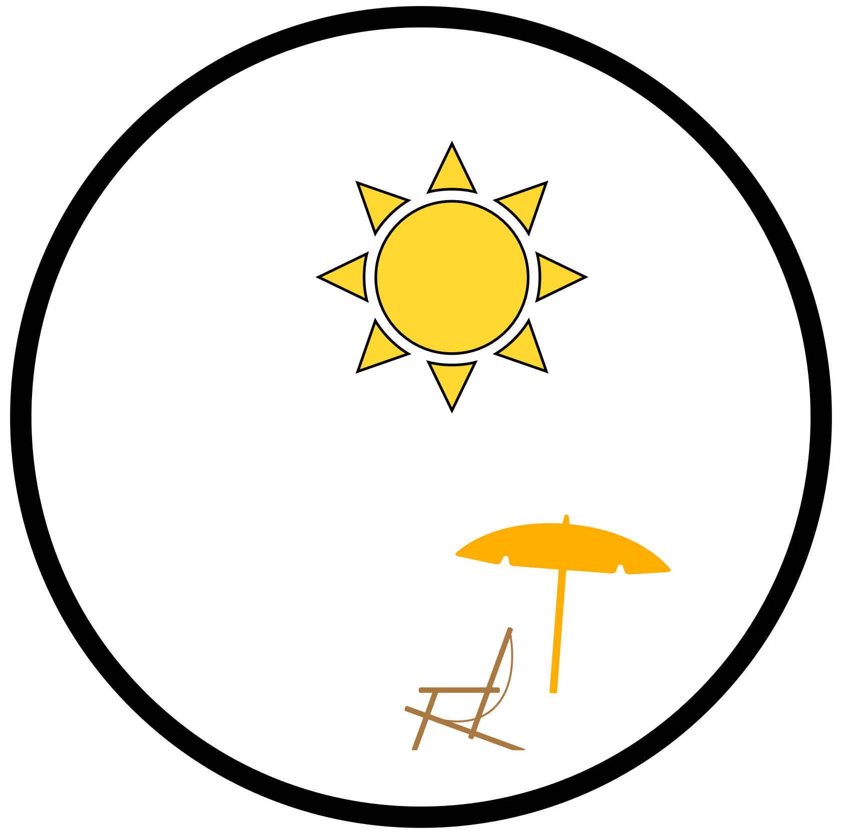
EPFL

MCMC: Bridging rendering, optimization and generative AI

MCMC stands for Markov chain Monte Carlo

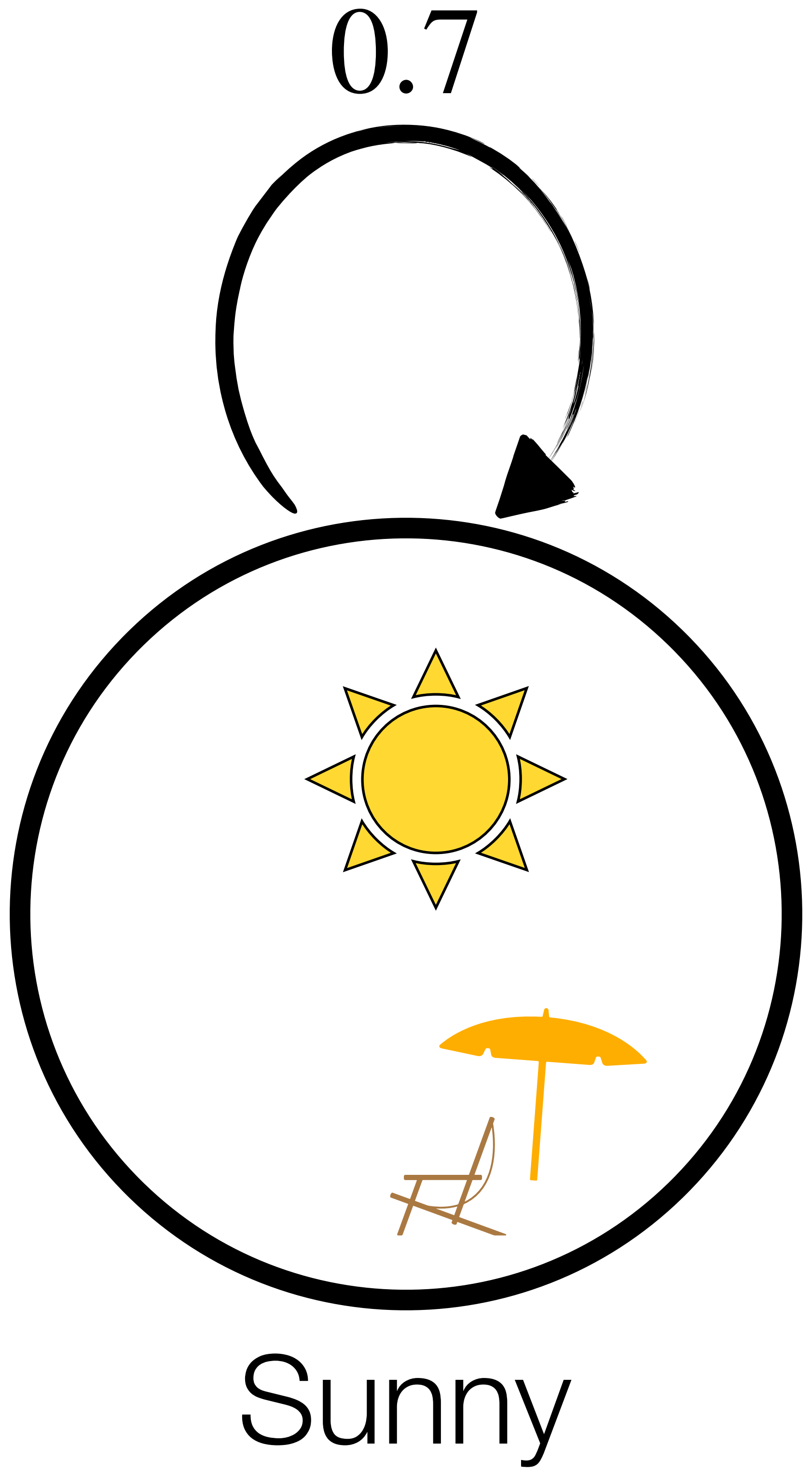
MCMC stands for Markov chain Monte Carlo

Markov chain: Weather forecast models

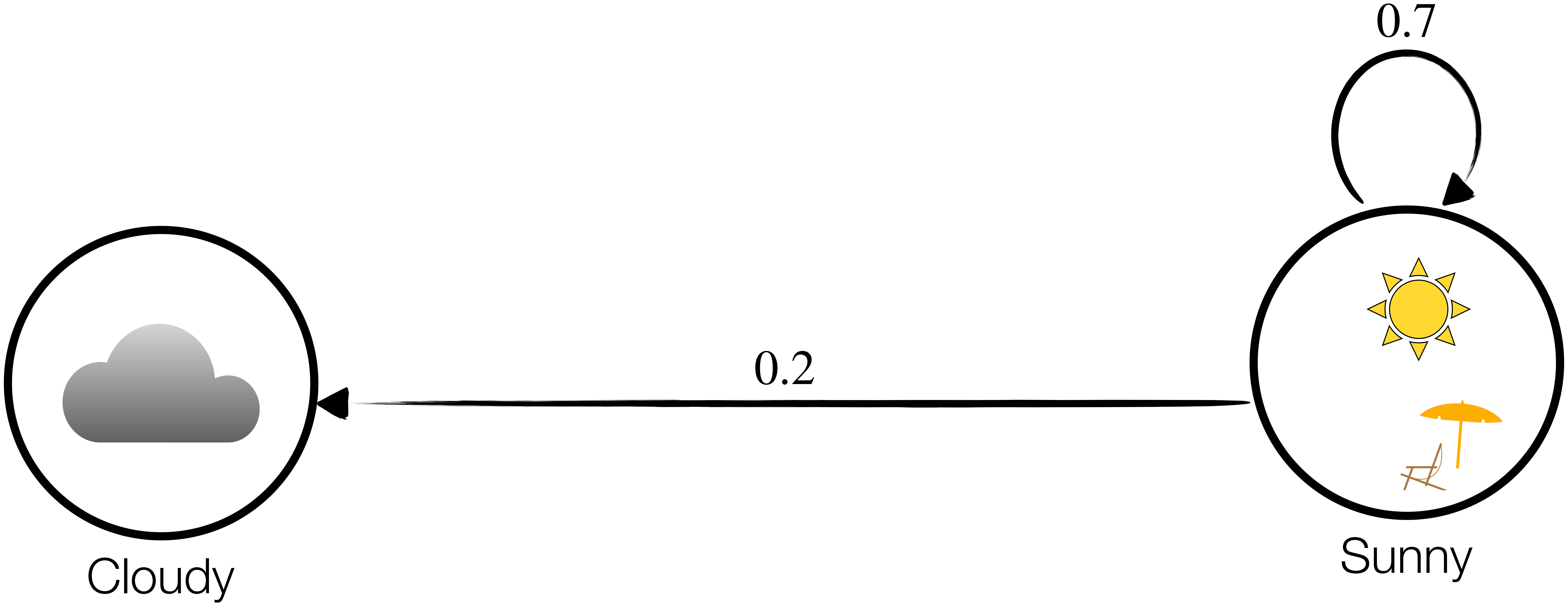


Sunny

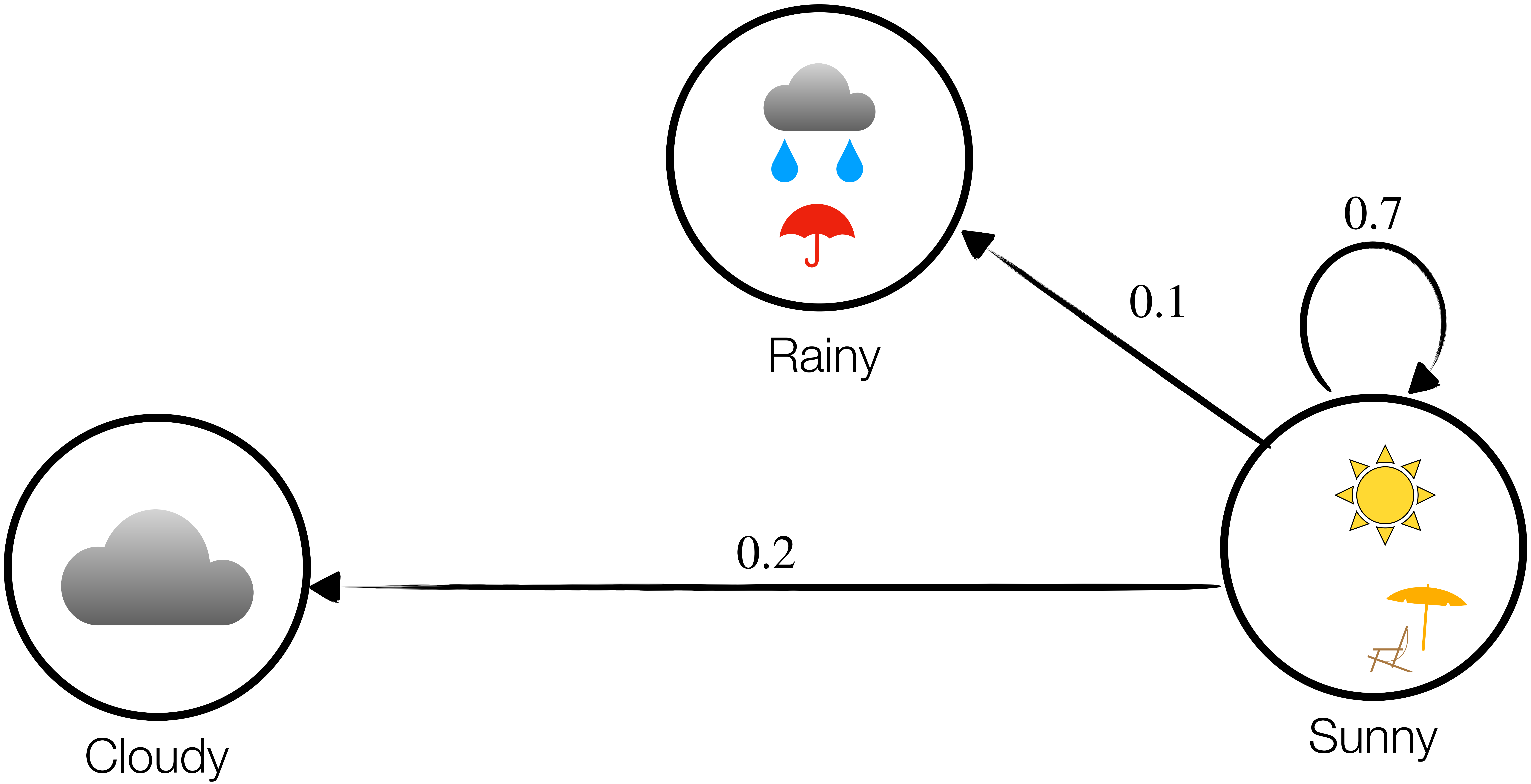
Markov chain: Weather forecast models



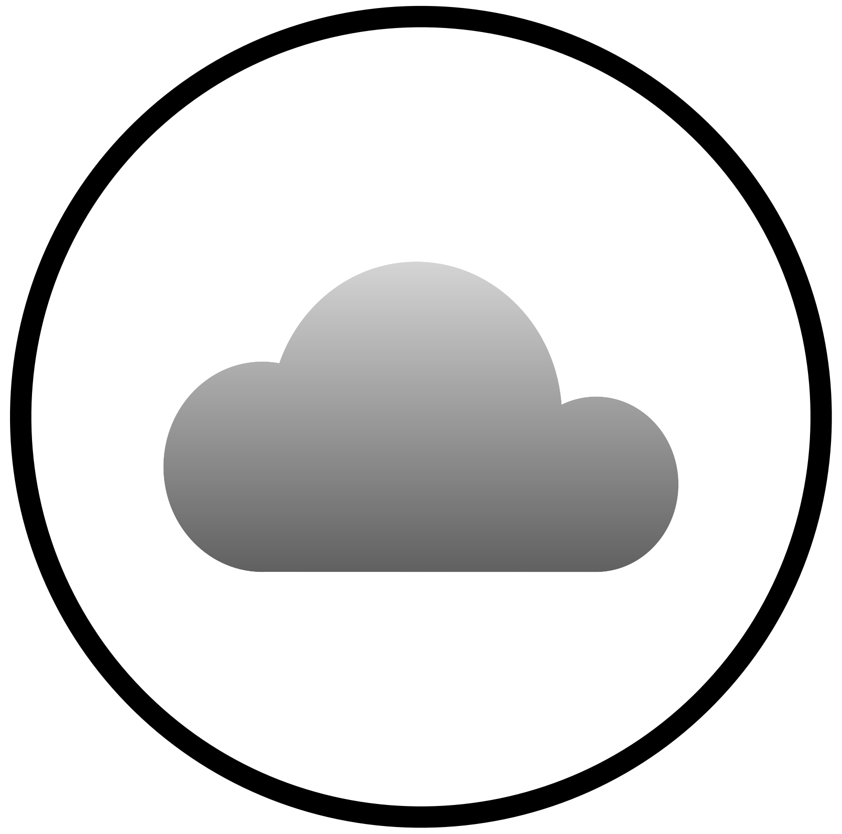
Markov chain: Weather forecast models



Markov chain: Weather forecast models

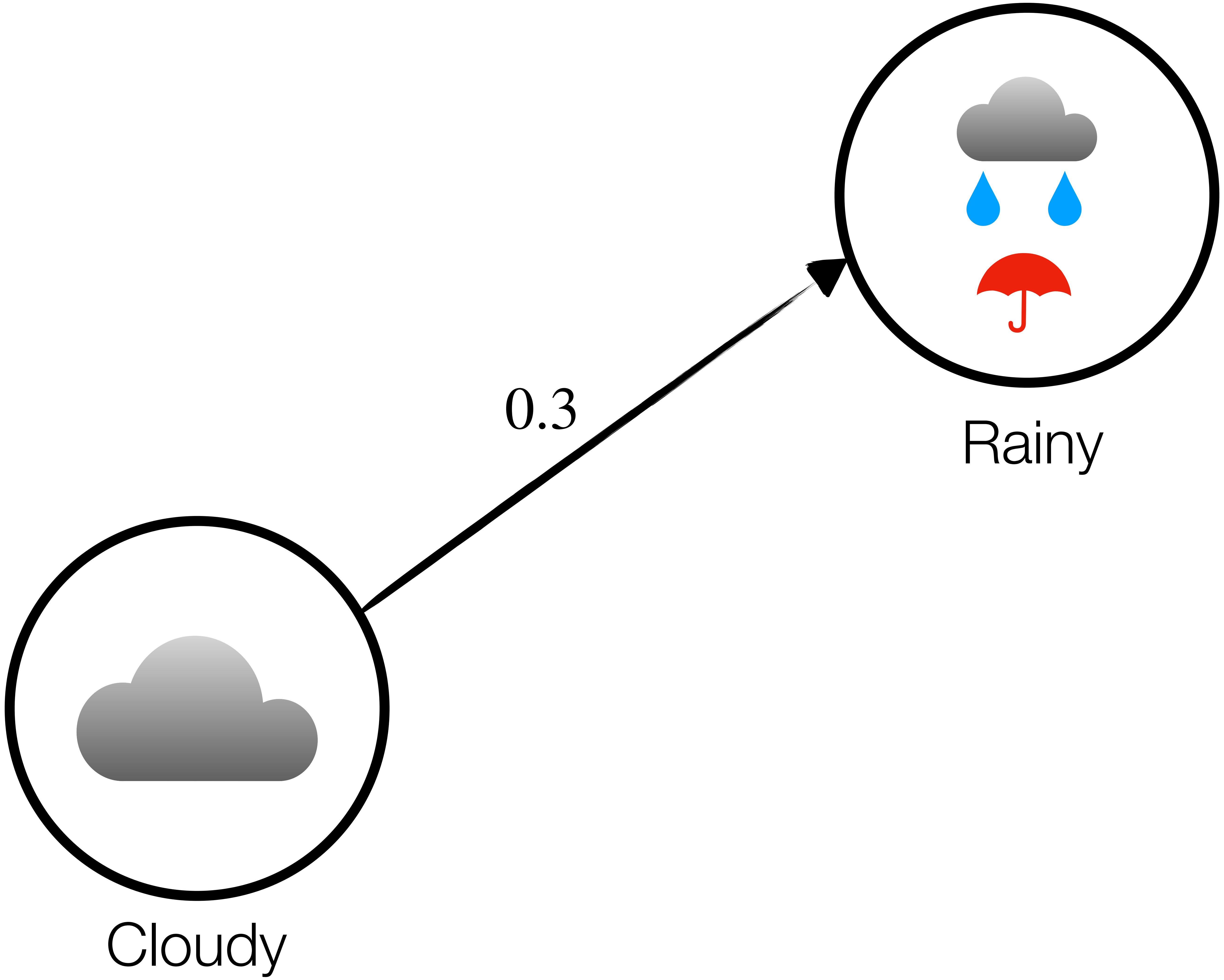


Markov chain: Weather forecast models

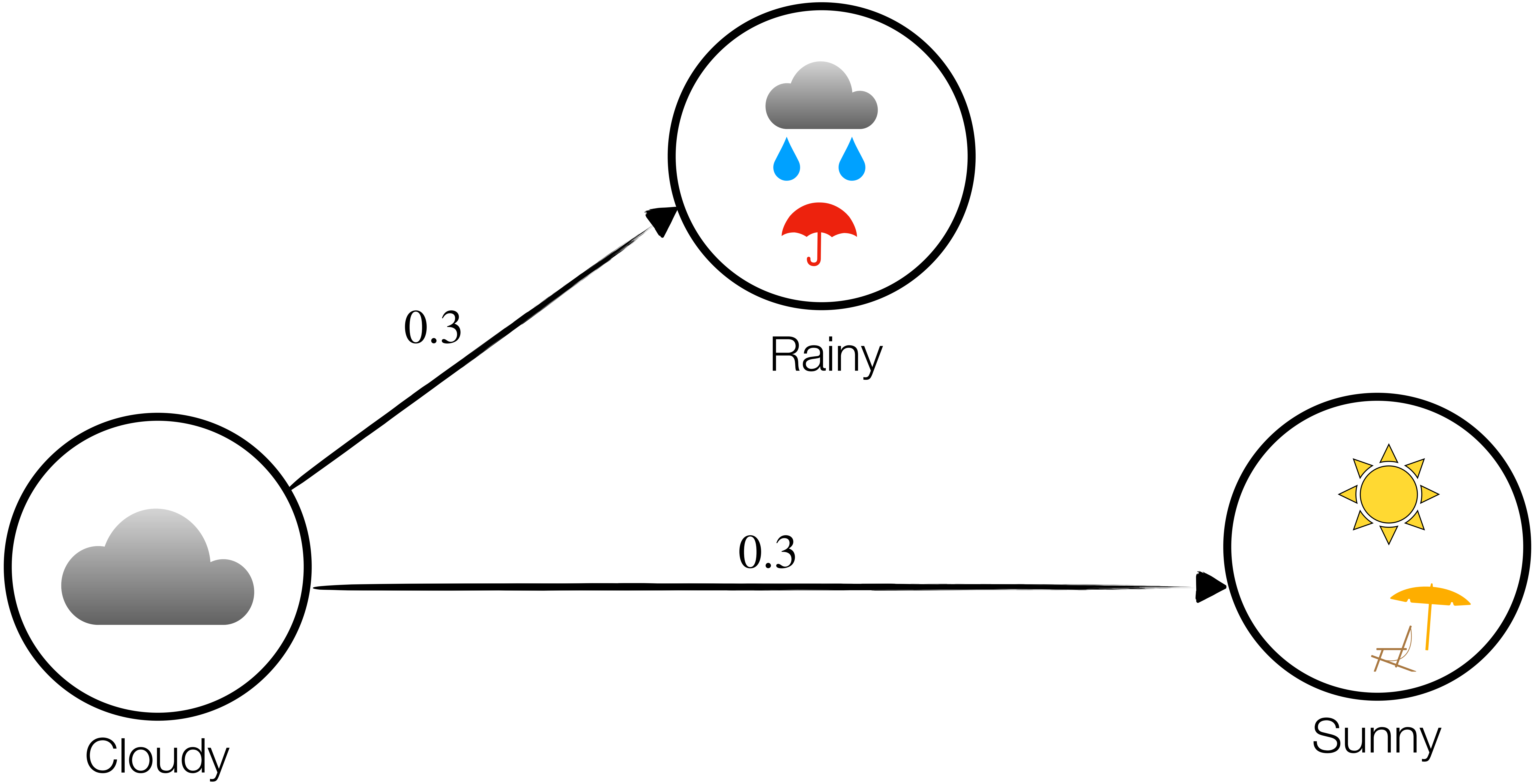


Cloudy

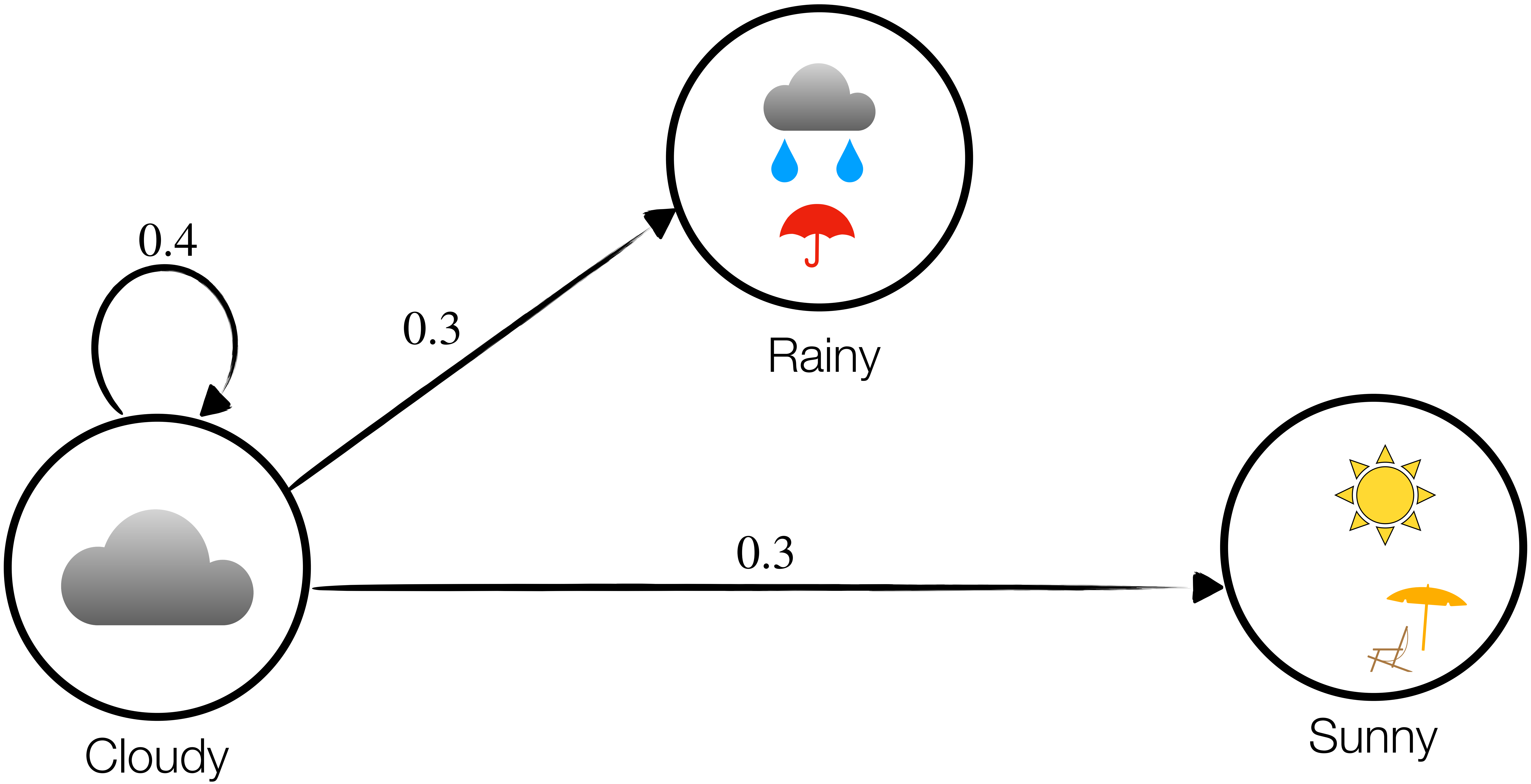
Markov chain: Weather forecast models



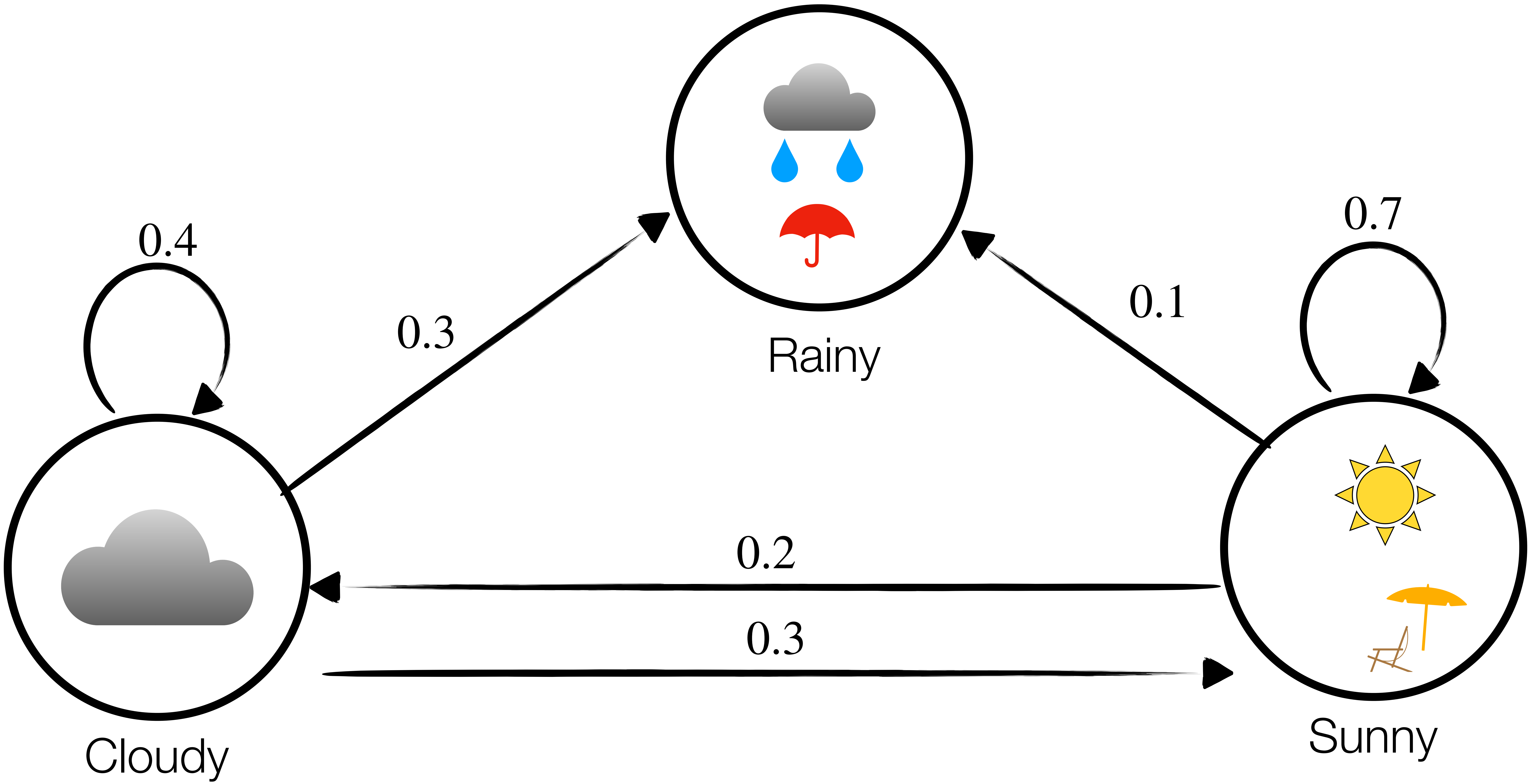
Markov chain: Weather forecast models



Markov chain: Weather forecast models



Markov chain: Weather forecast models



MCMC stands for Markov chain Monte Carlo

MCMC stands for Markov chain Monte Carlo

MCMC: Bridging rendering, optimization and generative AI

Image courtesy David Coeurjolly

Cornell Box
hitpoint
Image Plane

Image courtesy David Coeurjolly

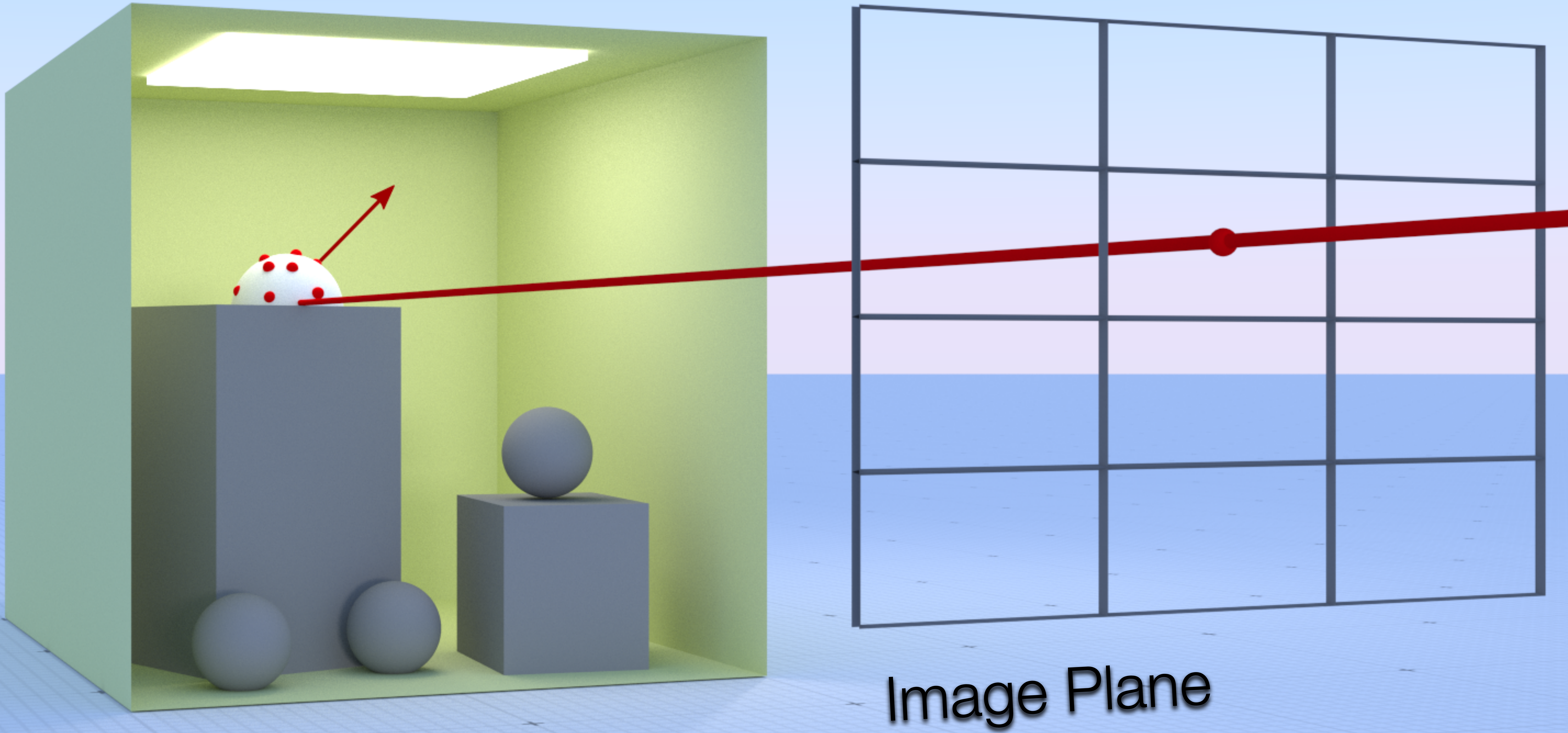


Image Plane

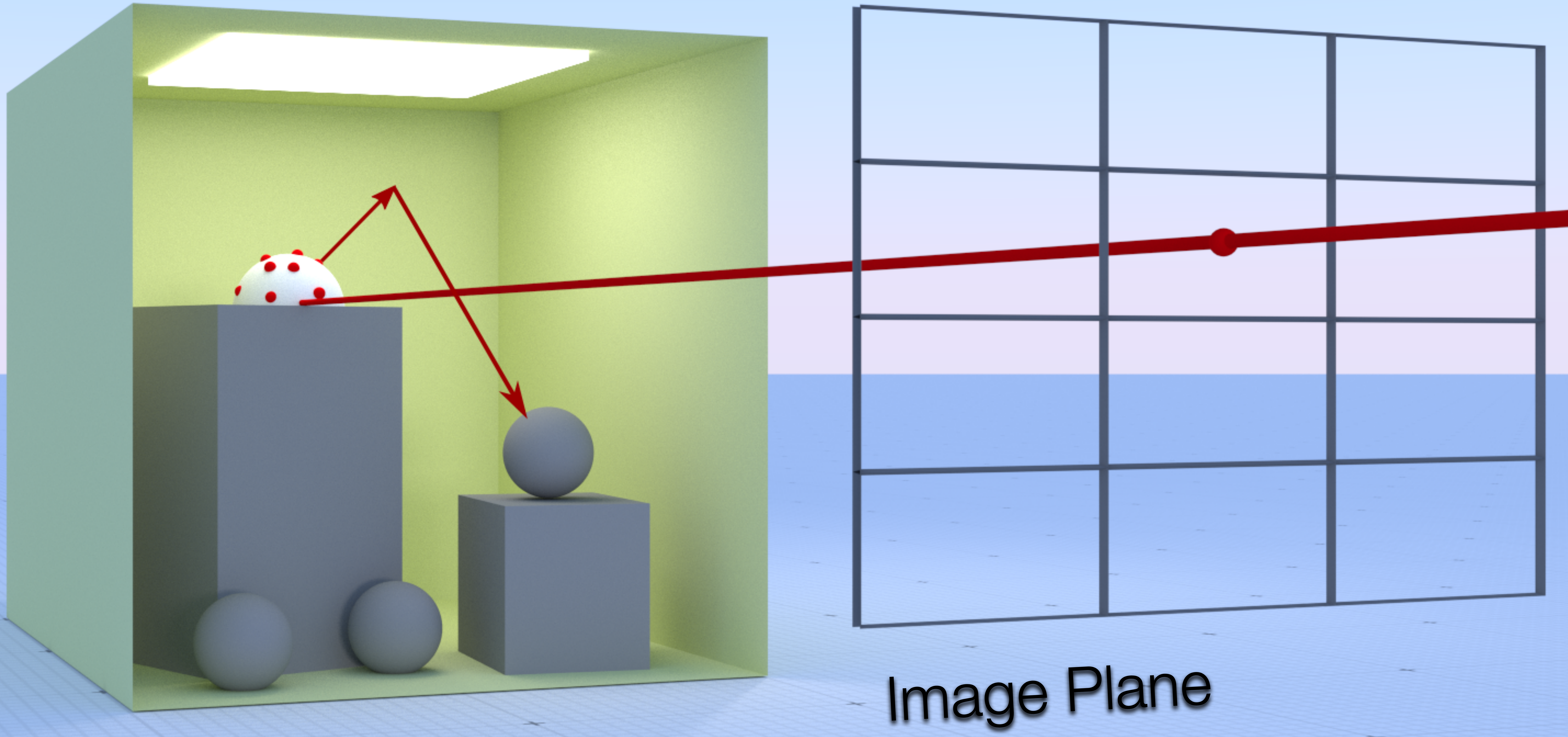


Image Plane

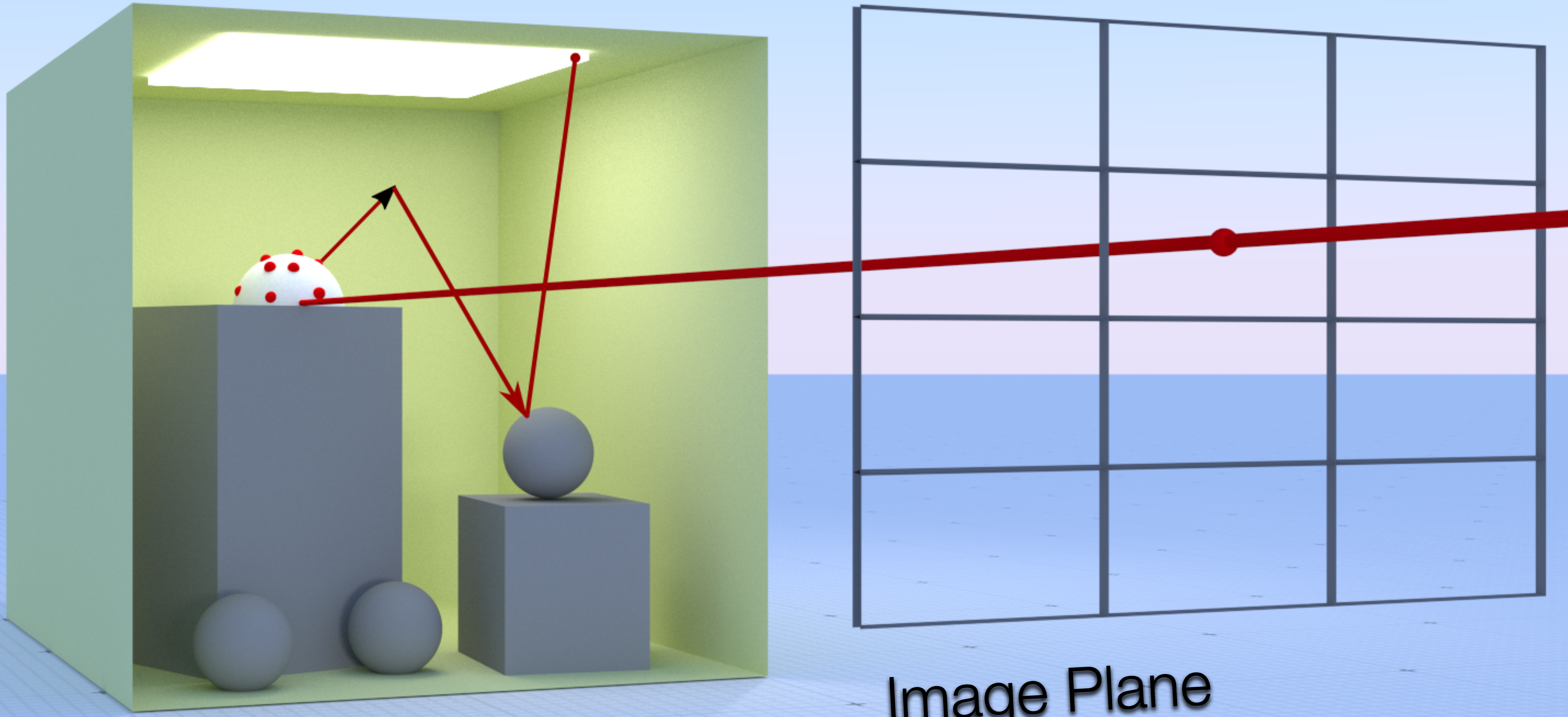


Image Plane

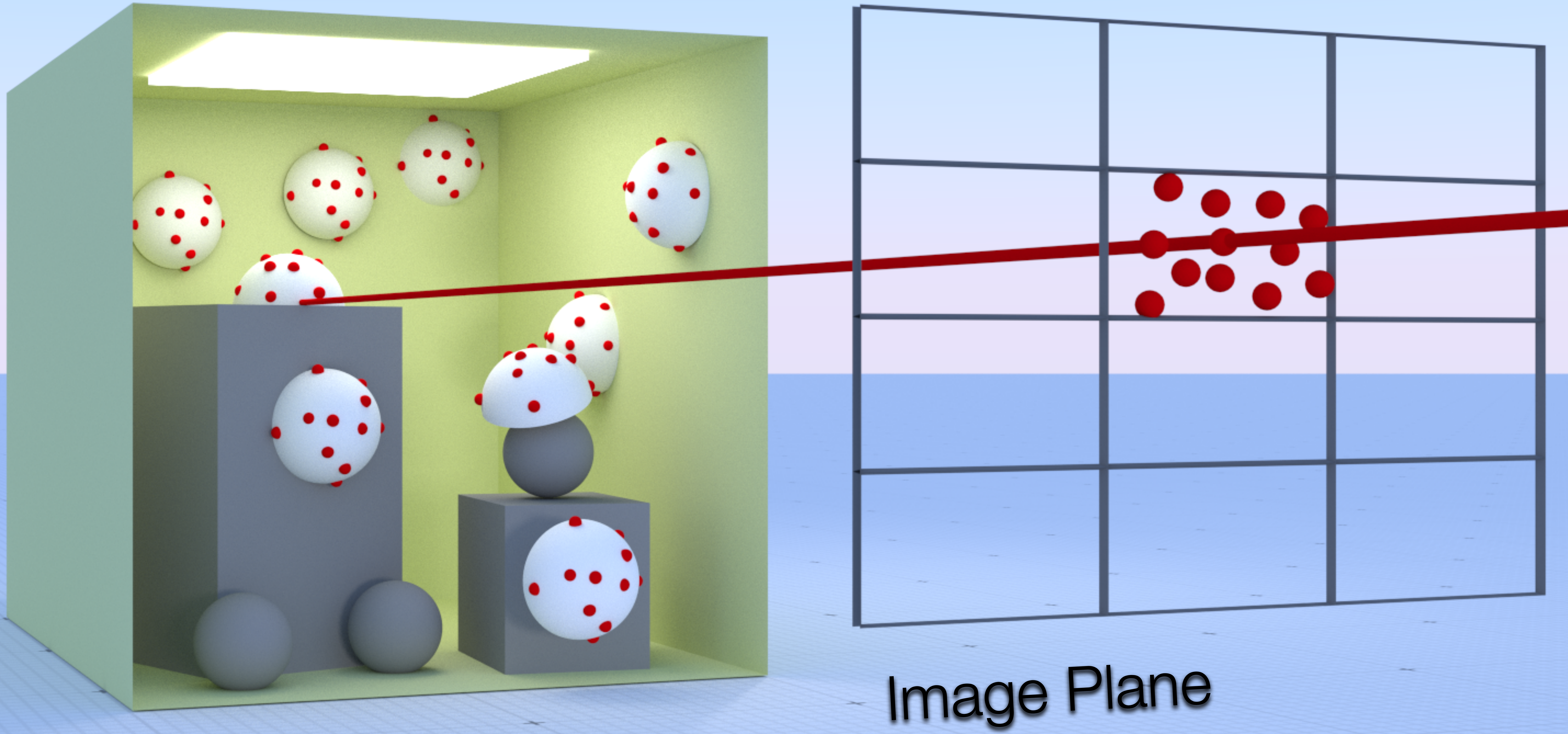
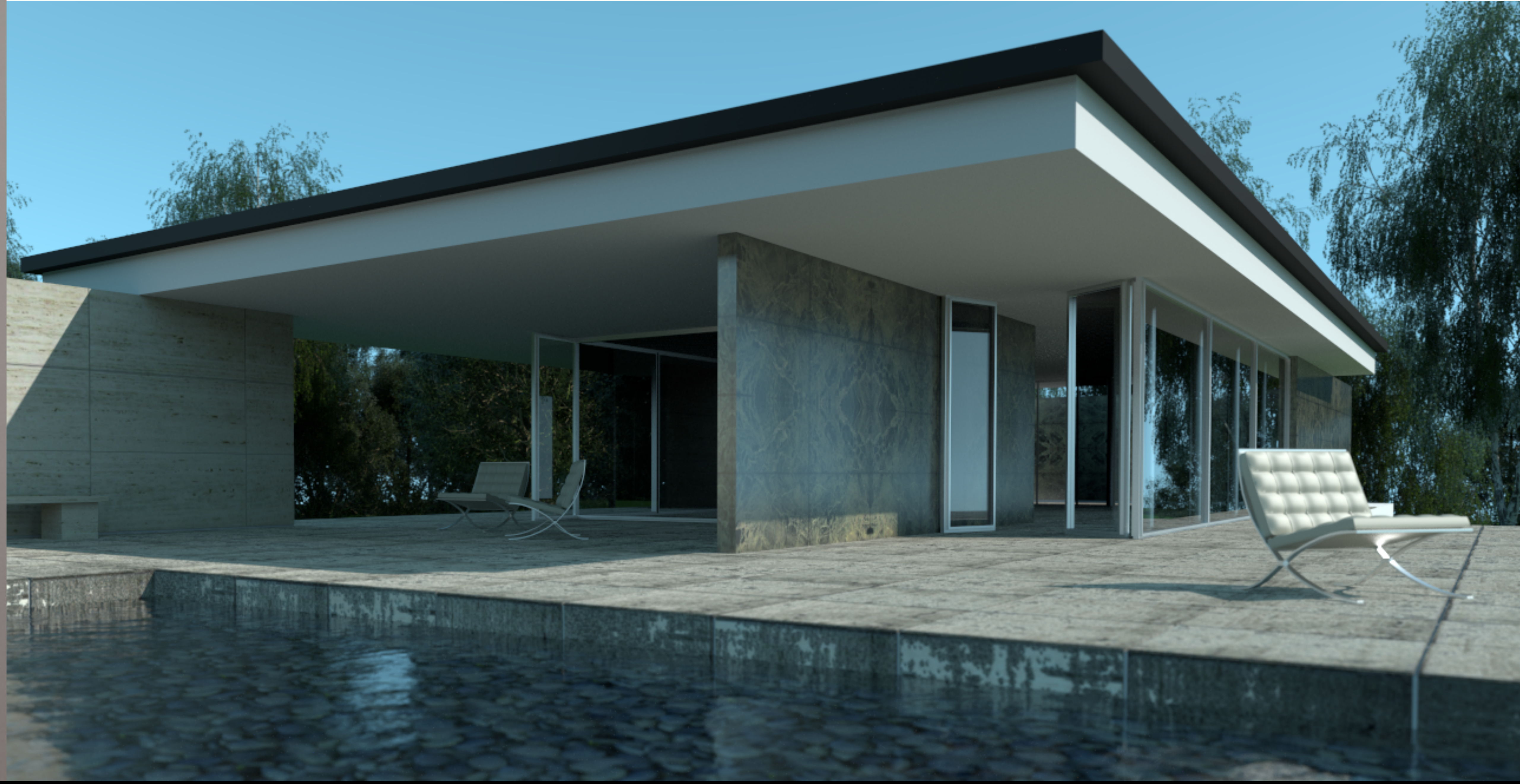
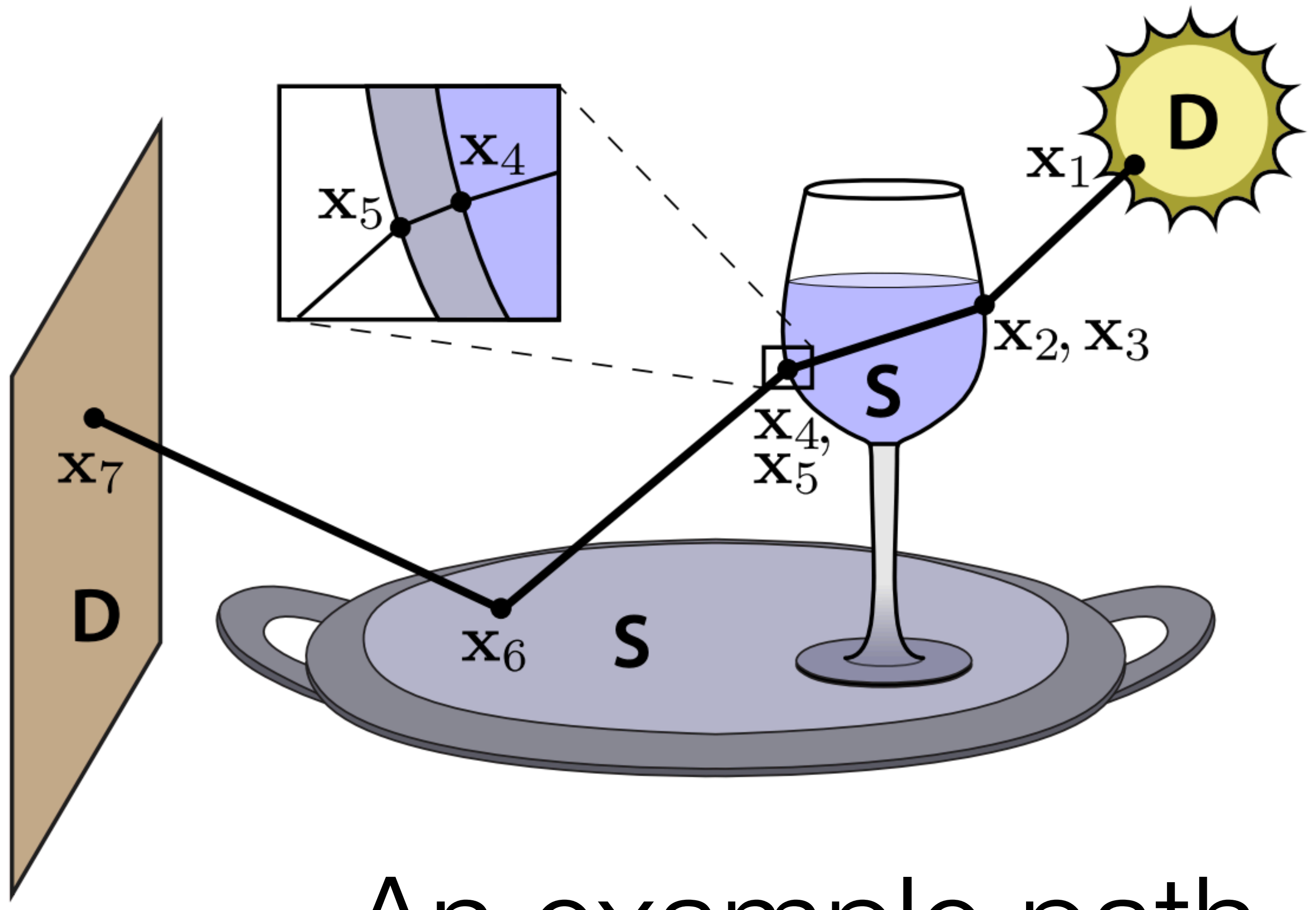


Image Plane



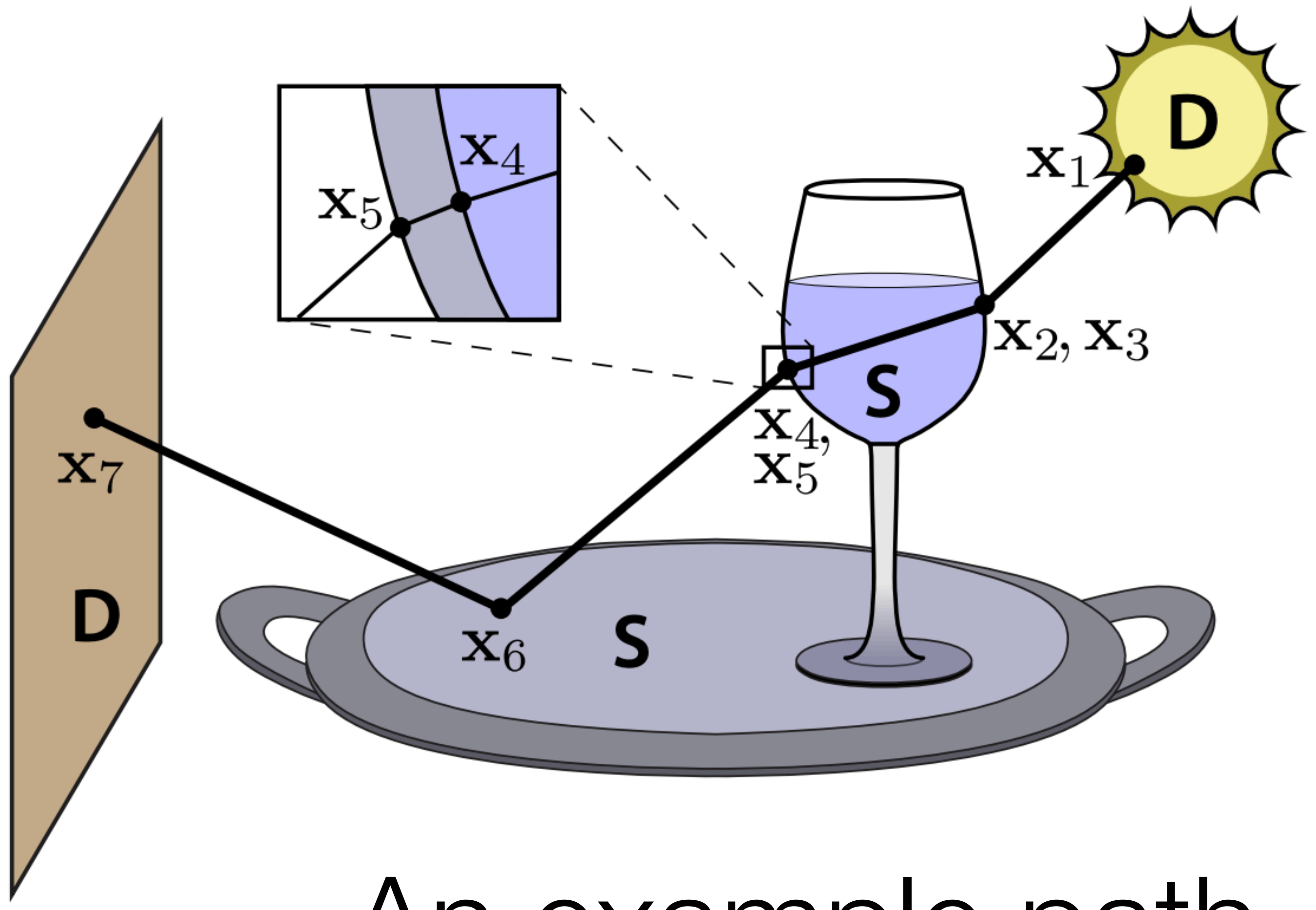
Source: PBRT & Bitterli Resources

MCMC Sampling for Light Transport

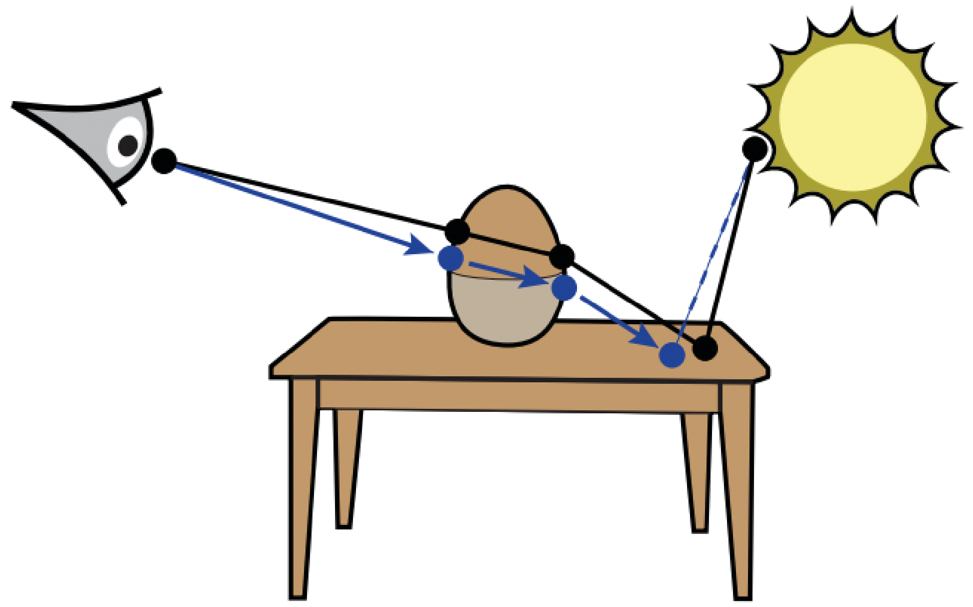


An example path

MCMC Sampling for Light Transport

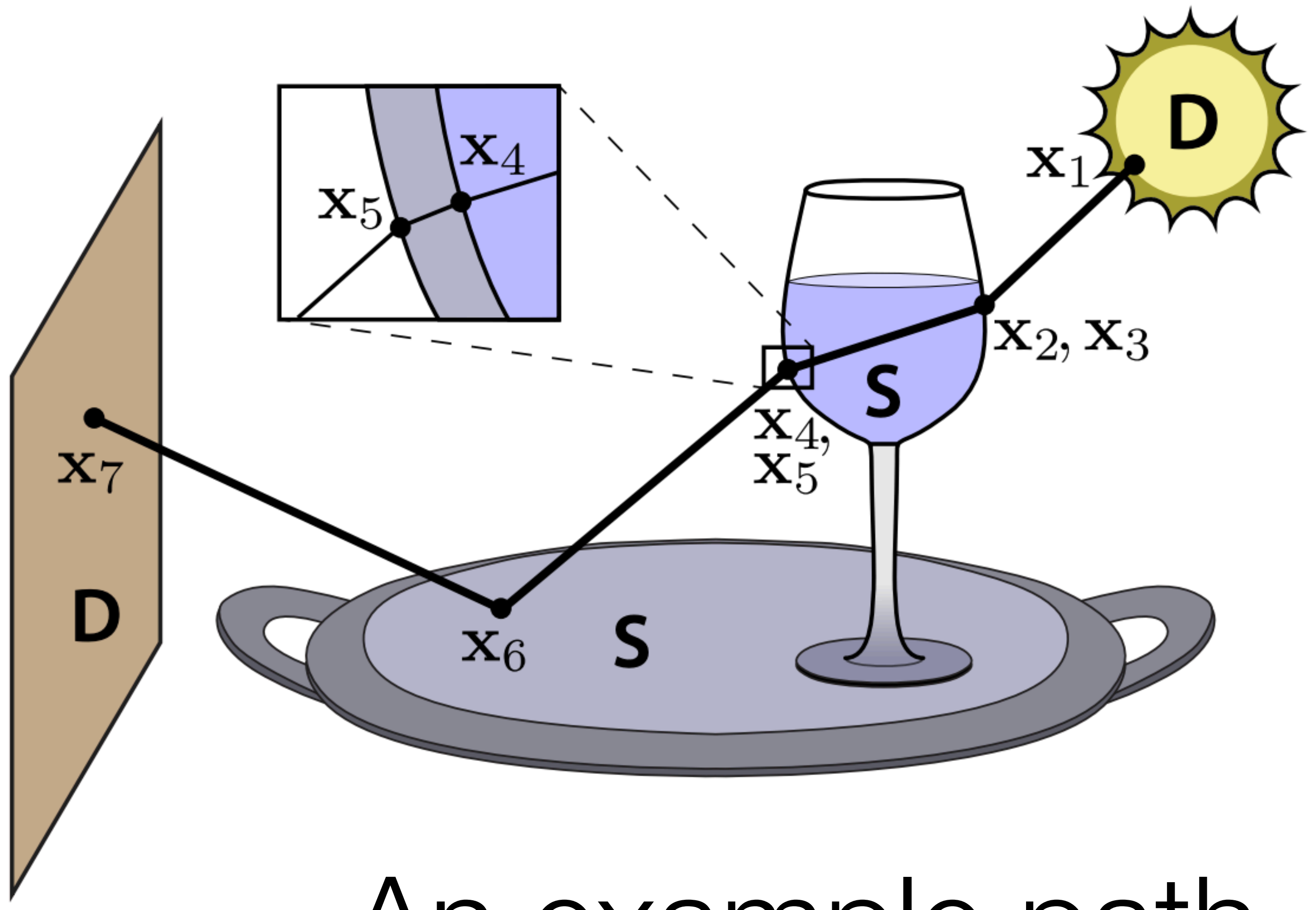


An example path

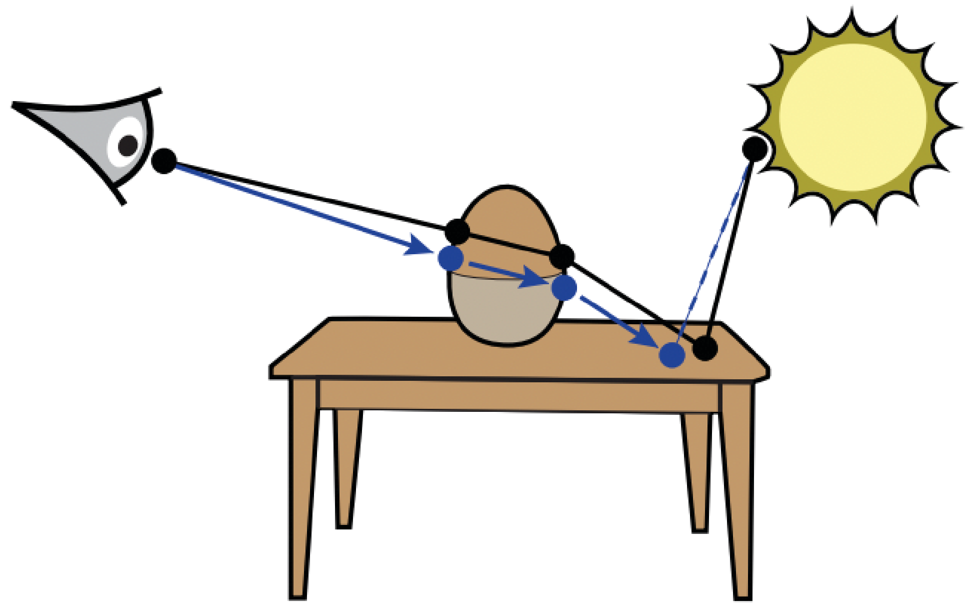


Lens perturbation

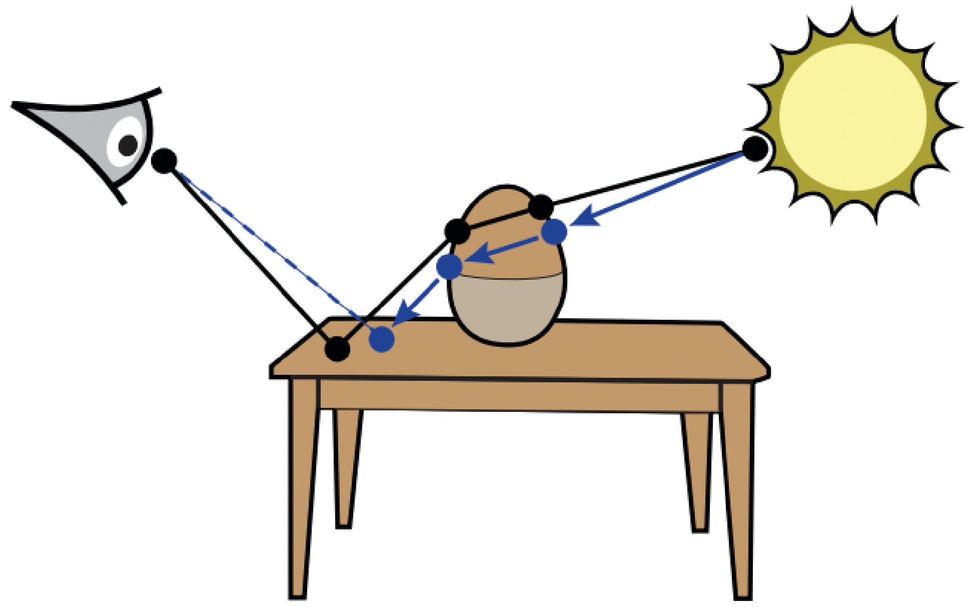
MCMC Sampling for Light Transport



An example path

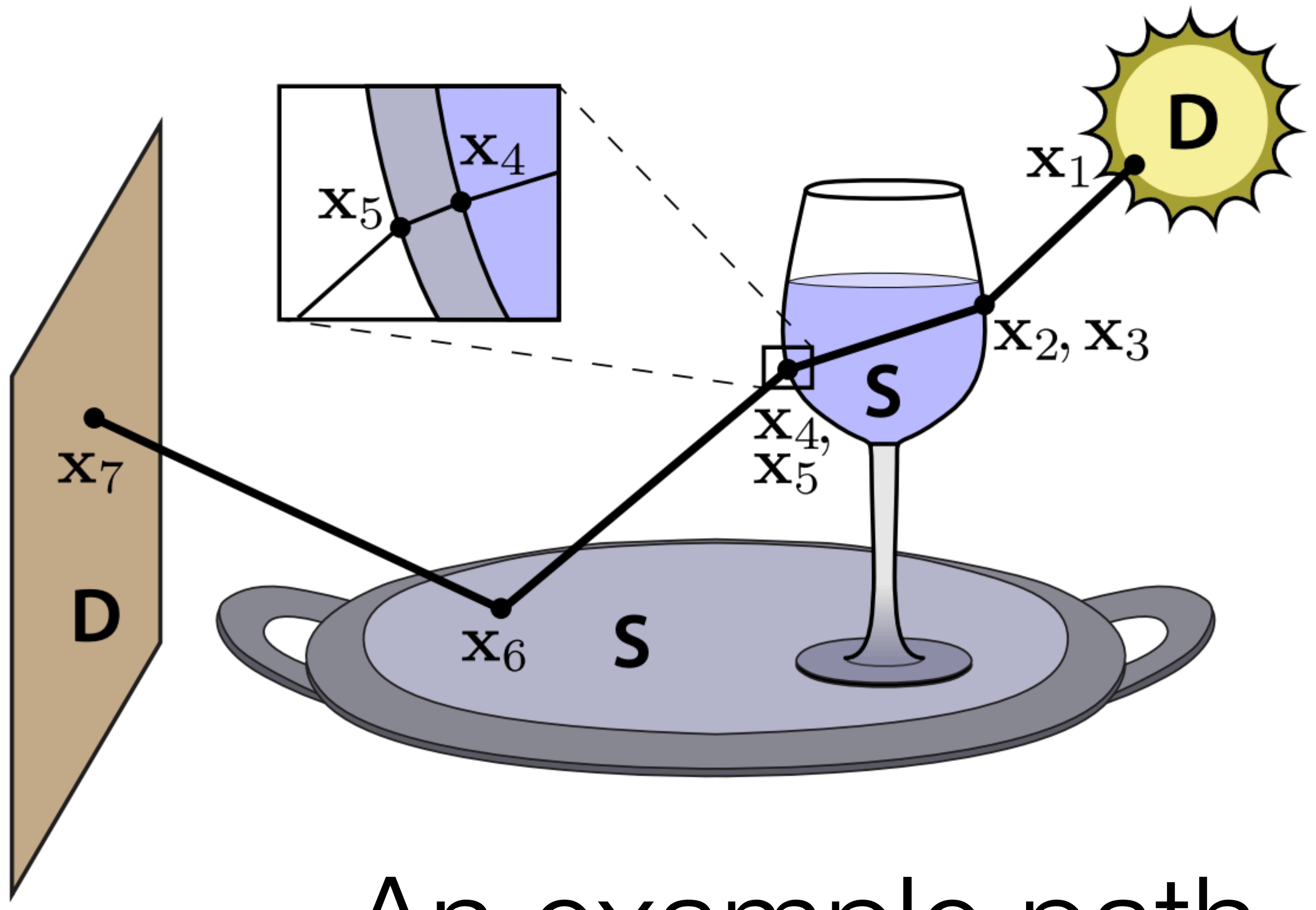


Lens perturbation

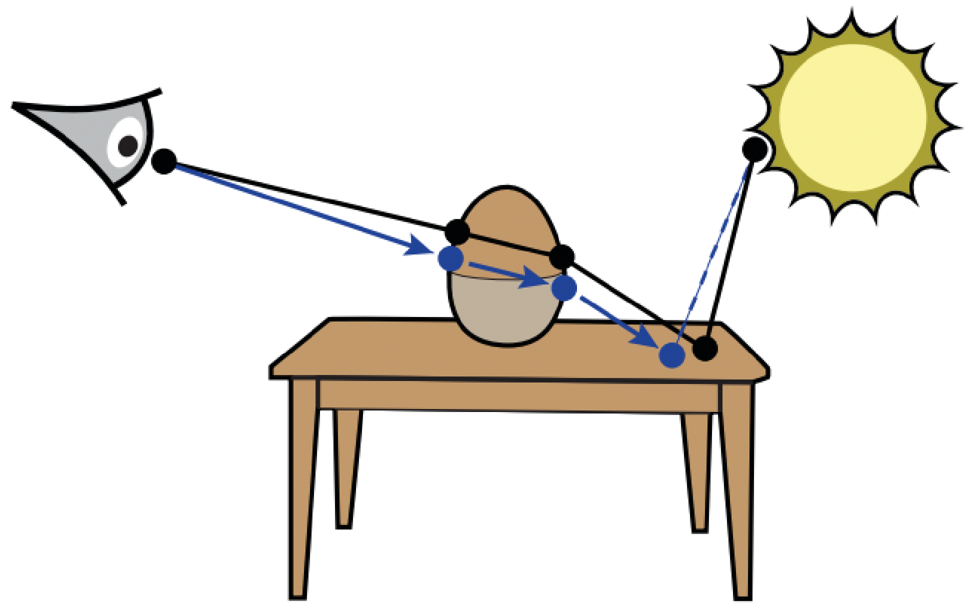


Caustic perturbation

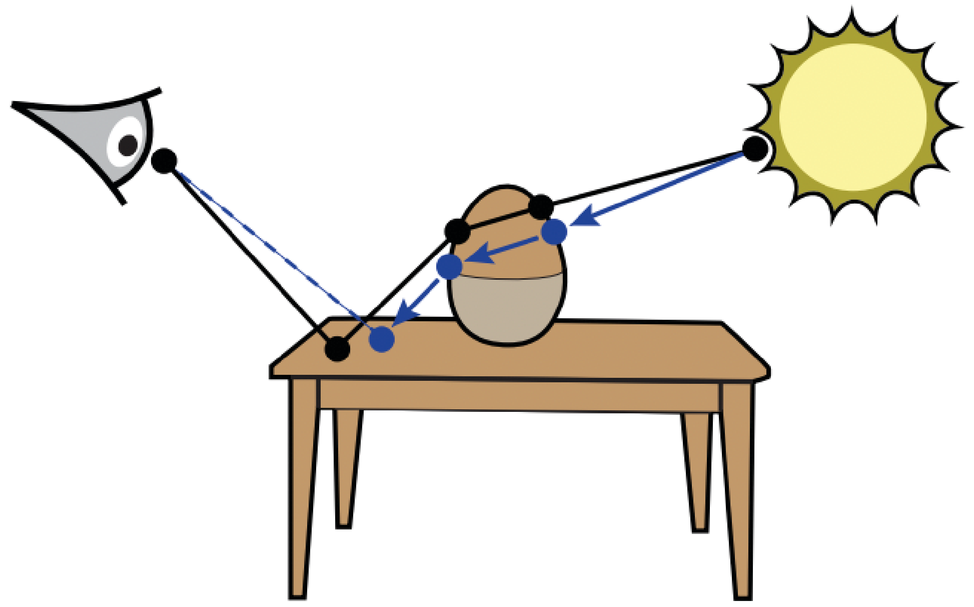
MCMC Sampling for Light Transport



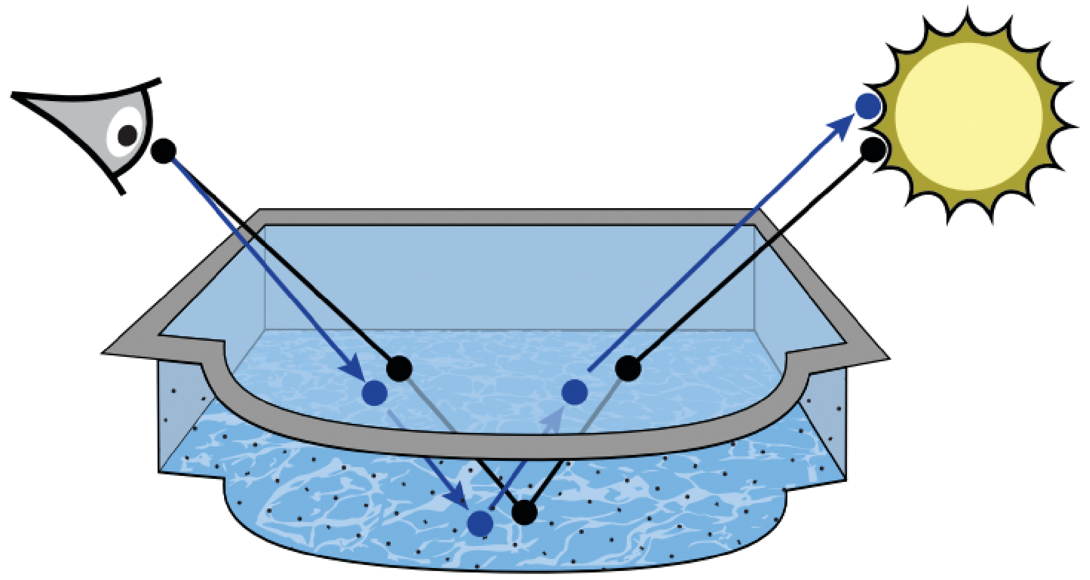
An example path



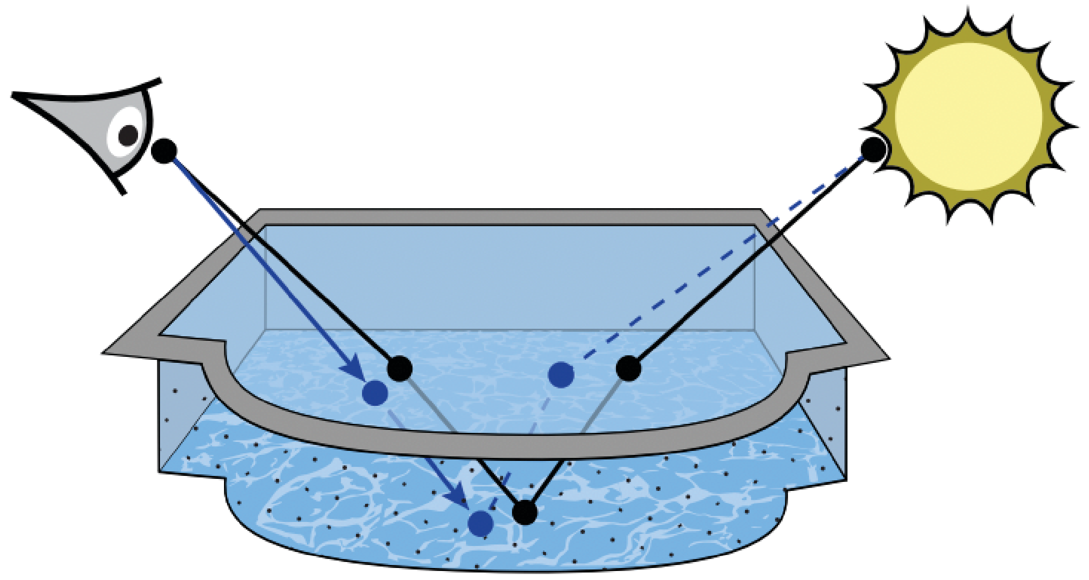
Lens perturbation



Caustic perturbation



Multi-chain perturbation



Manifold perturbation

MCMC Sampling for Light Transport

IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS 1

Survey of Markov Chain Monte Carlo Methods in Light Transport Simulation

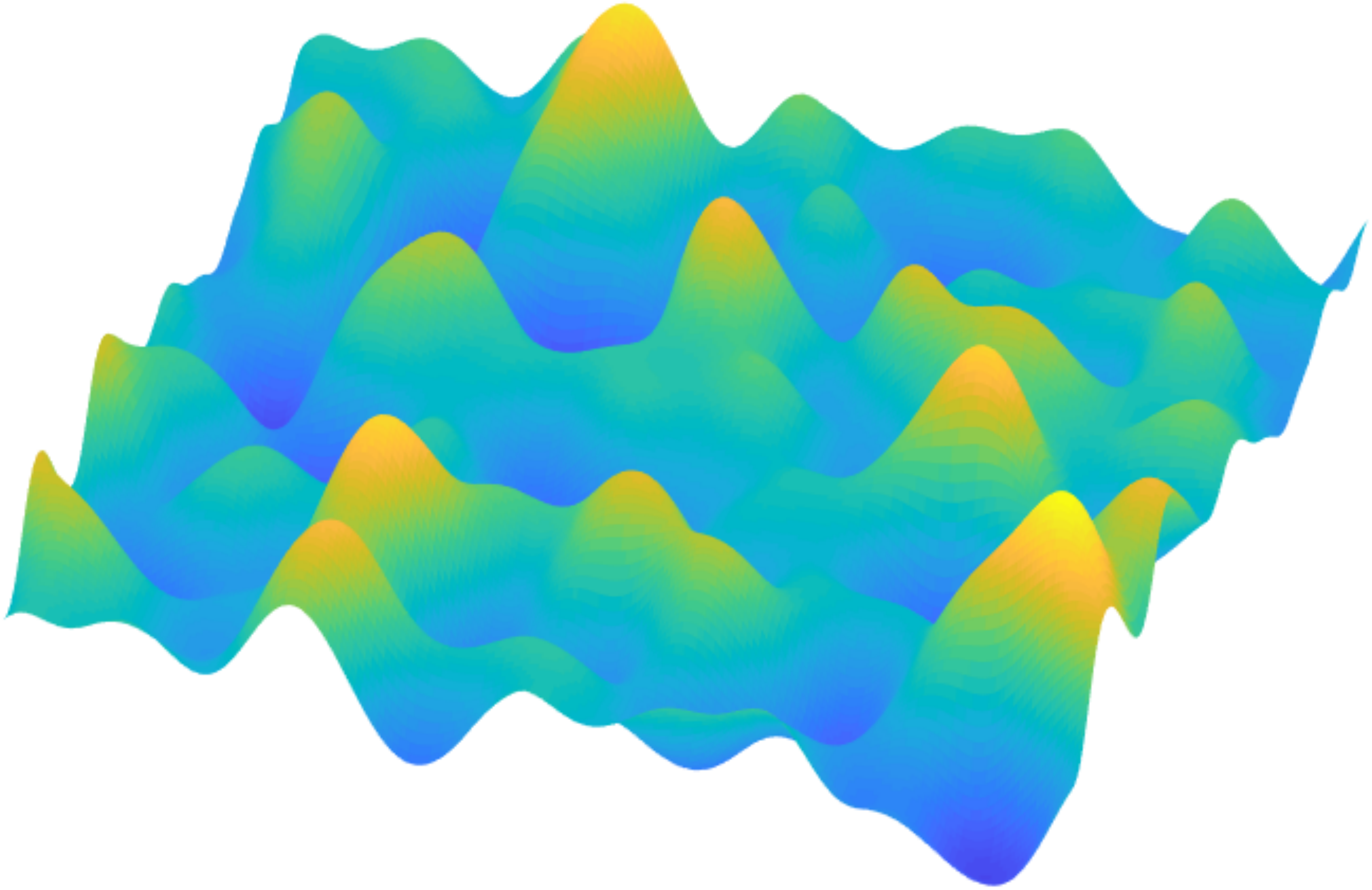
Martin Šik and Jaroslav Křivánek

Abstract—Two decades have passed since the introduction of Markov chain Monte Carlo (MCMC) into light transport simulation by Veach and Guibas, and numerous follow-up works have been published since then. However, up until now no survey has attempted to cover the majority of these methods. The aim of this paper is therefore to offer a first comprehensive survey of MCMC algorithms for light transport simulation. The methods presented in this paper are categorized by their objectives and properties, while we point out their strengths and weaknesses. We discuss how the methods handle the main issues of MCMC and how they could be combined or improved in the near future. To make the paper suitable for readers unacquainted with MCMC methods, we include an introduction to general MCMC and its demonstration on a simple example.

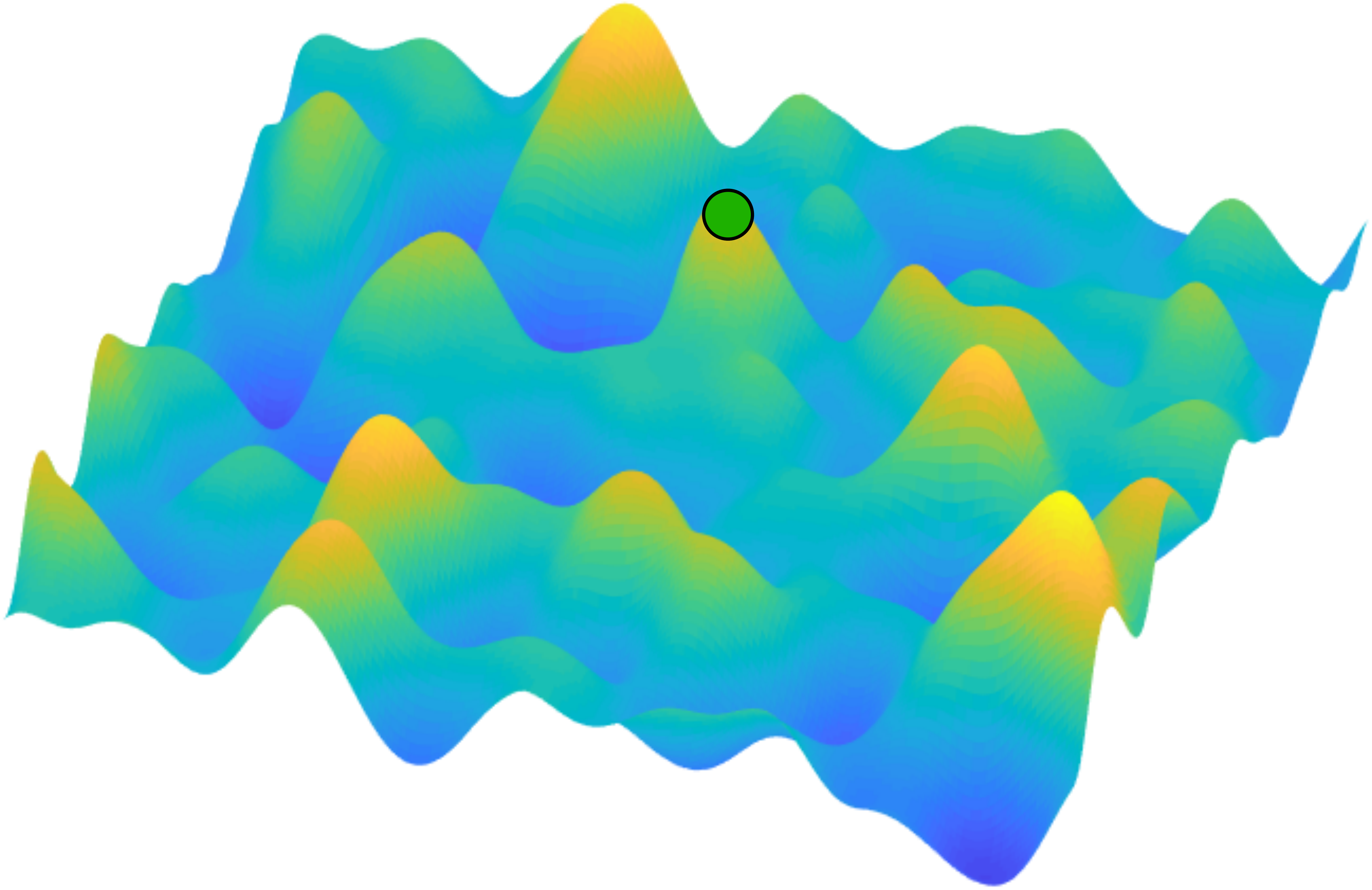
Index Terms—Markov Chain Monte Carlo, Metropolis-Hastings, Metropolis Light Transport, Light Transport Simulation, STAR.

MCMC: Bridging rendering, optimization and generative AI

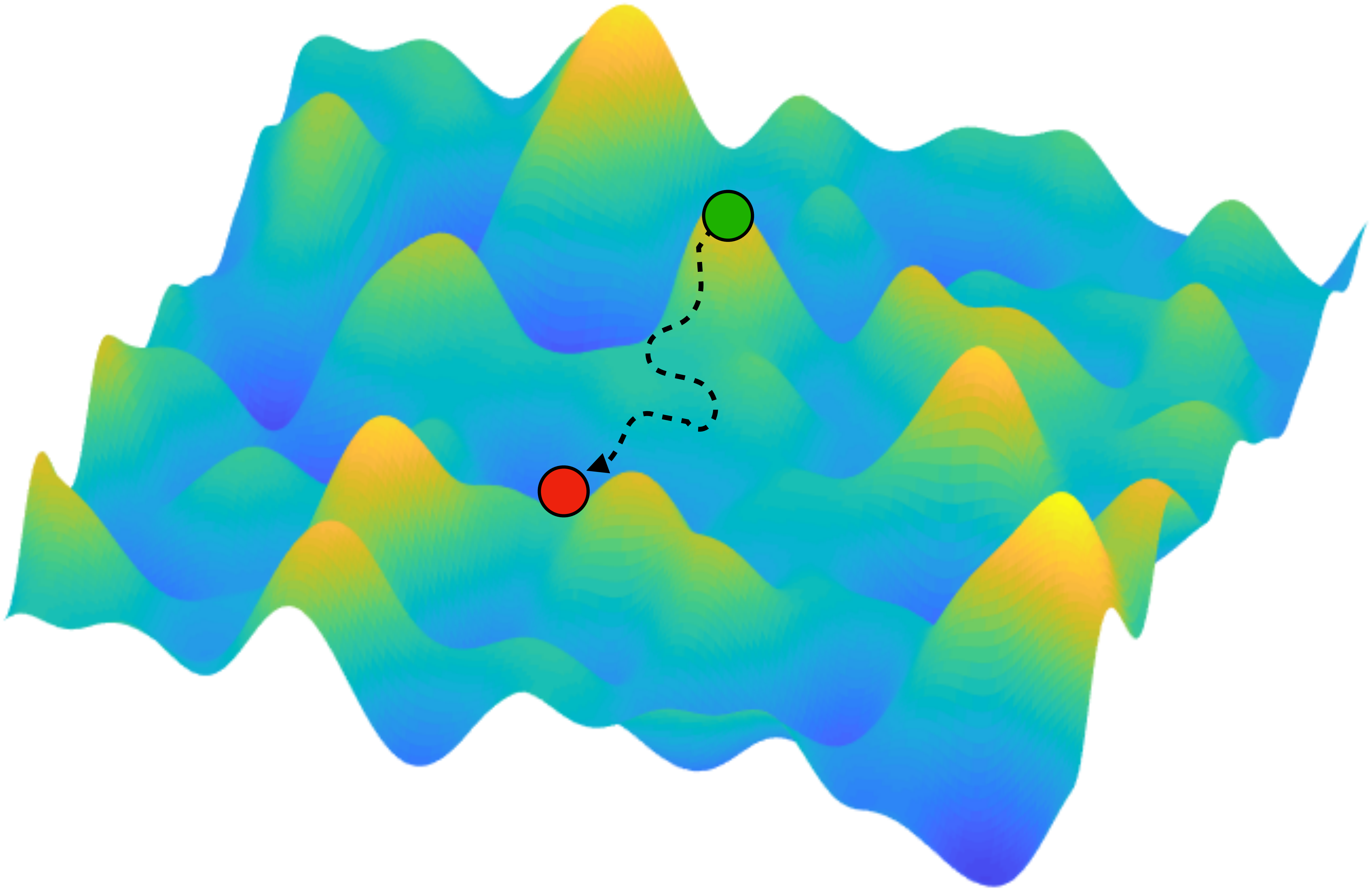
Optimization manifold



Optimization manifold

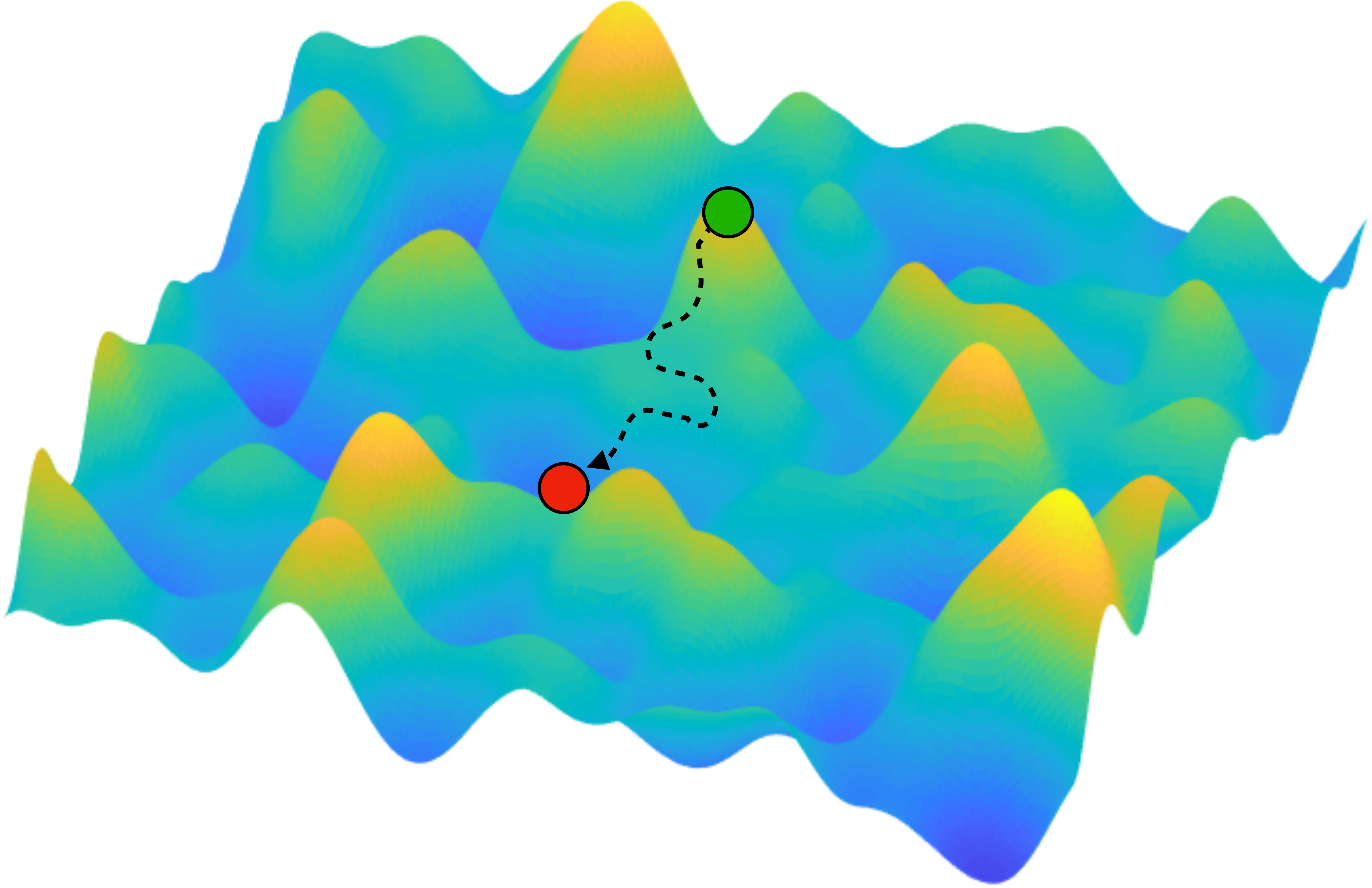


Optimization manifold



Optimization manifold

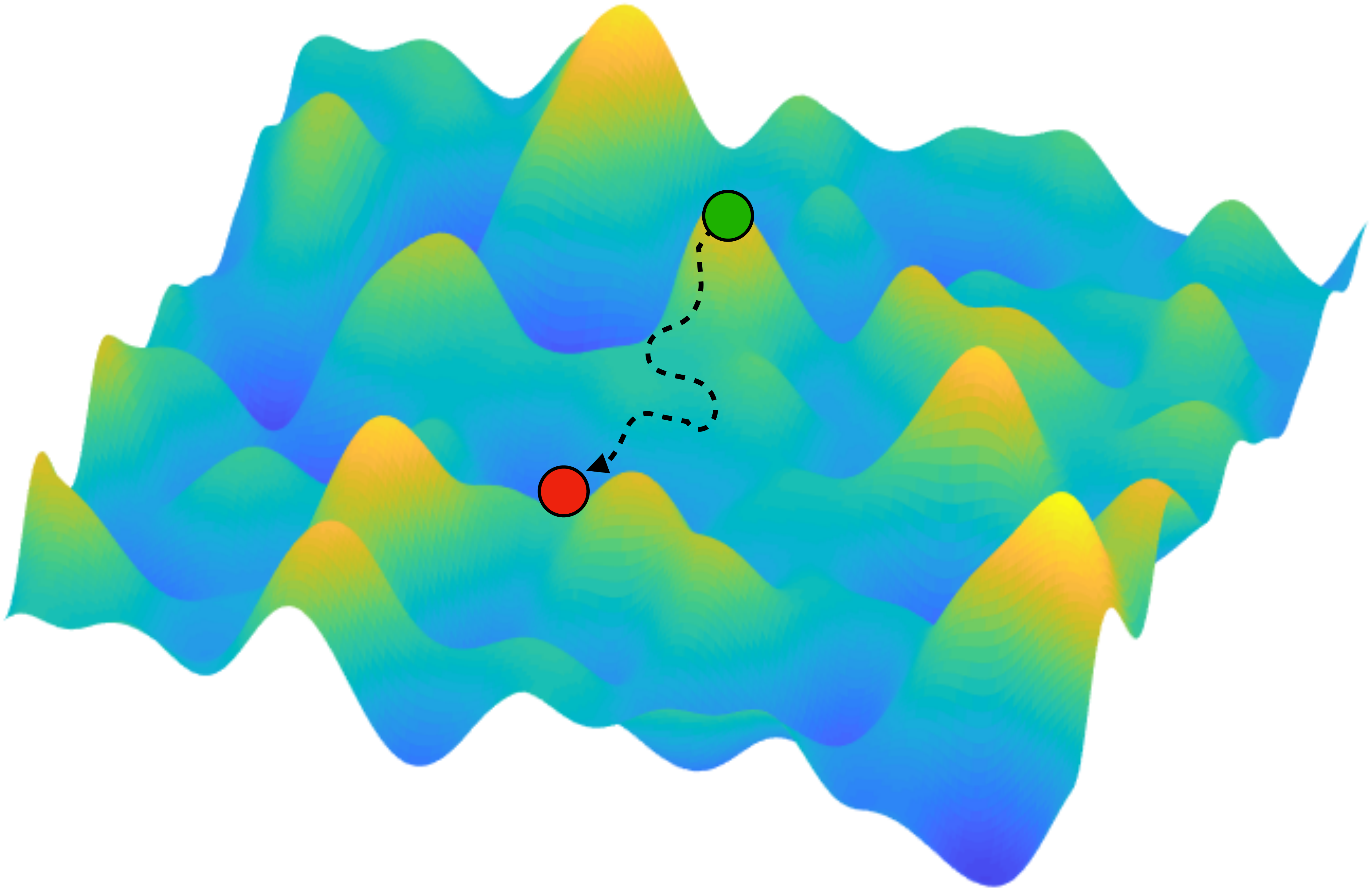
$\mathbf{x}_{t+1}$



Optimization manifold

future

$\mathbf{x}_{t+1}$

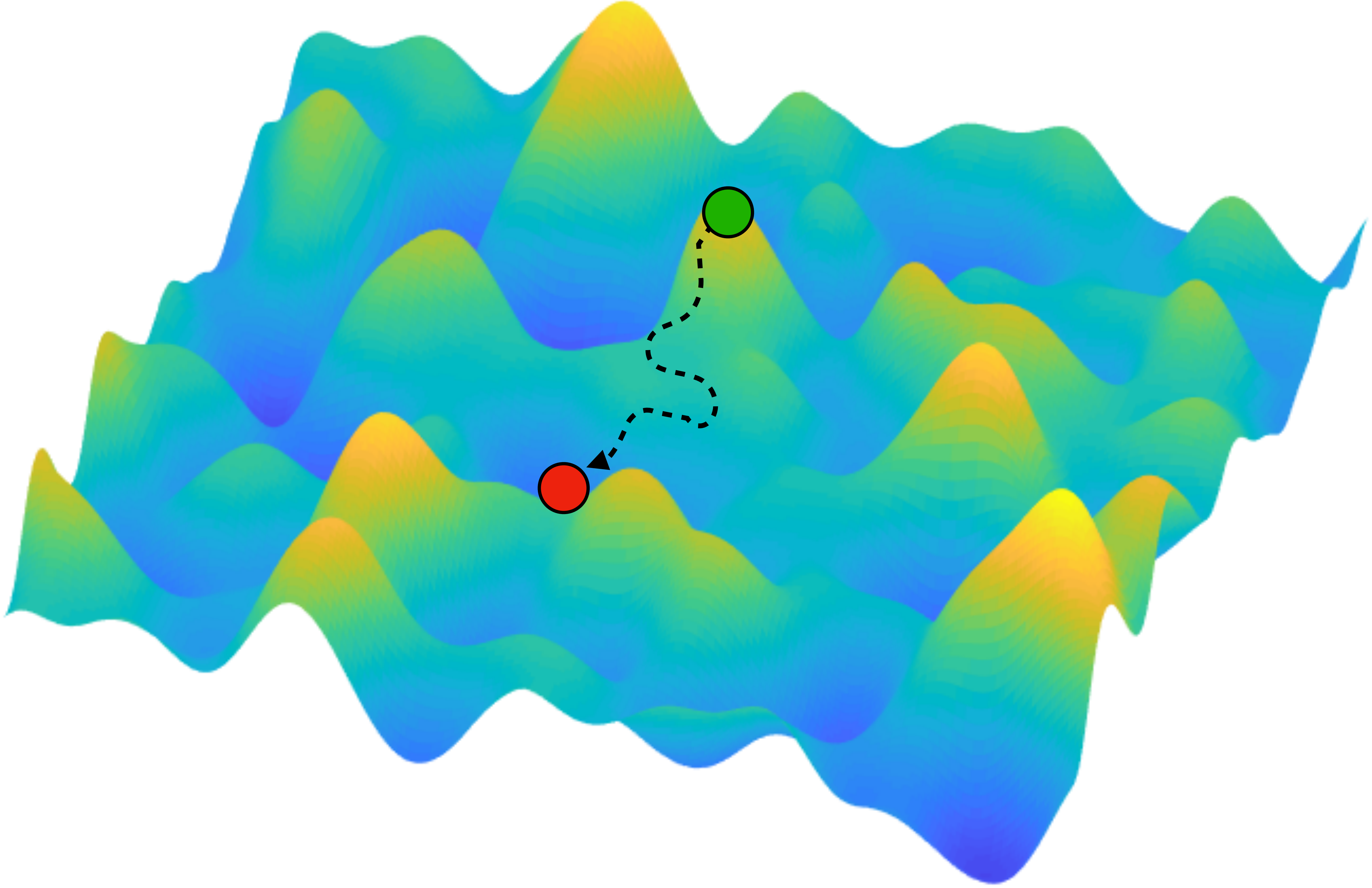


Optimization manifold

future

$$\mathbf{x}_{t+1} = \mathbf{x}_t$$

current

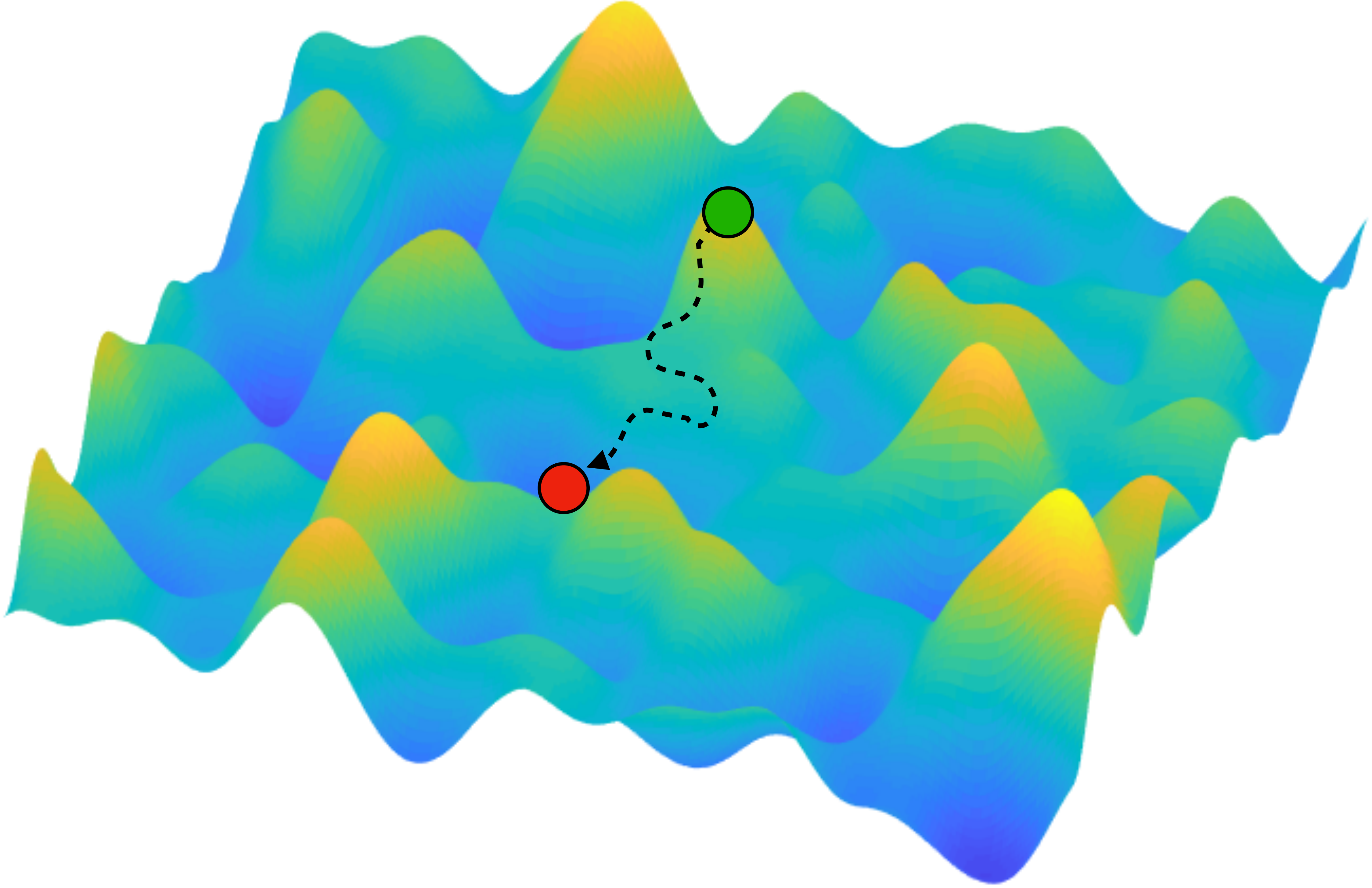


Optimization manifold

future

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \text{perturbation}$$

current



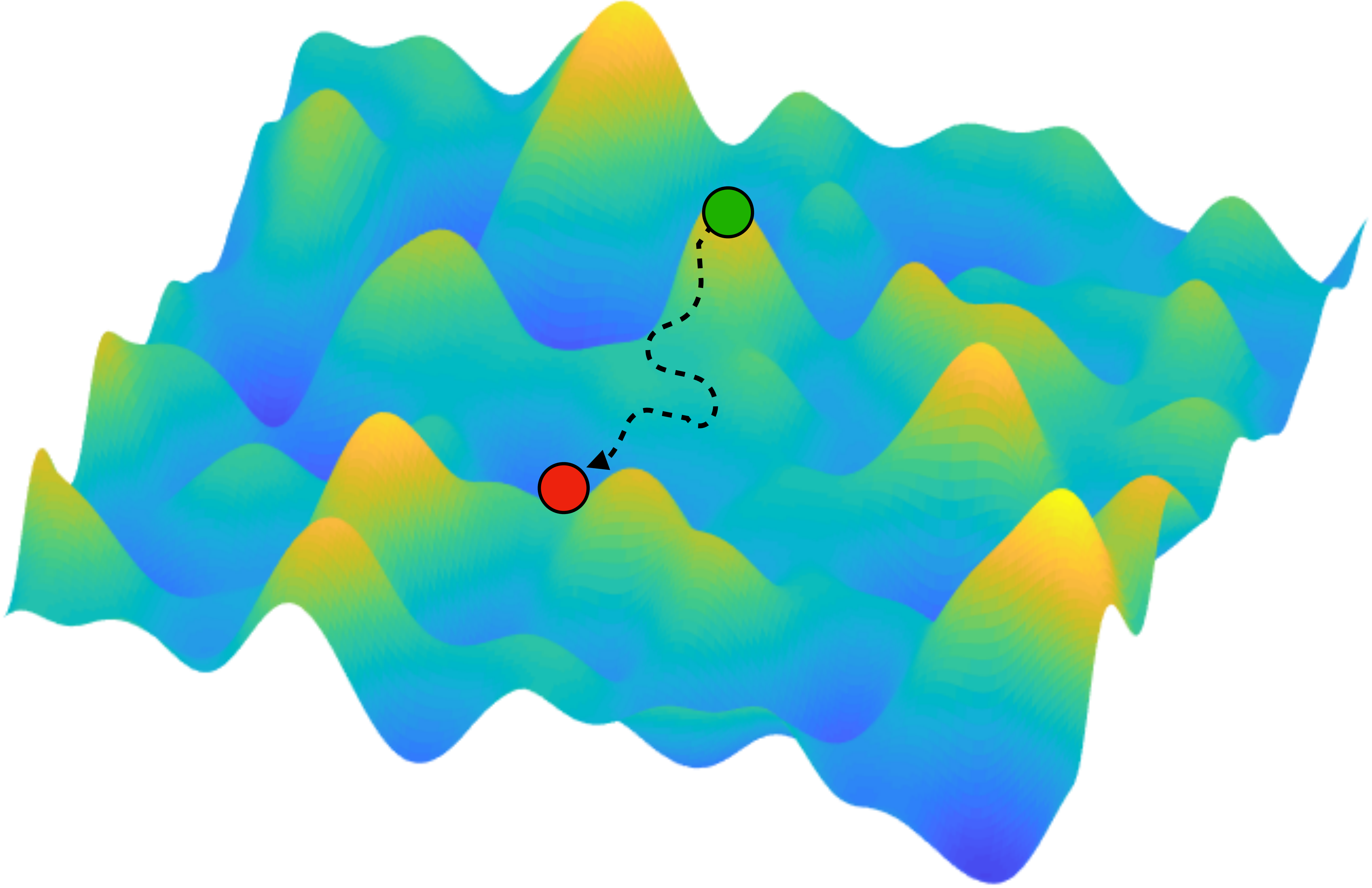
Optimization manifold

future

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \text{perturbation}$$

current

This is a Markov Chain!



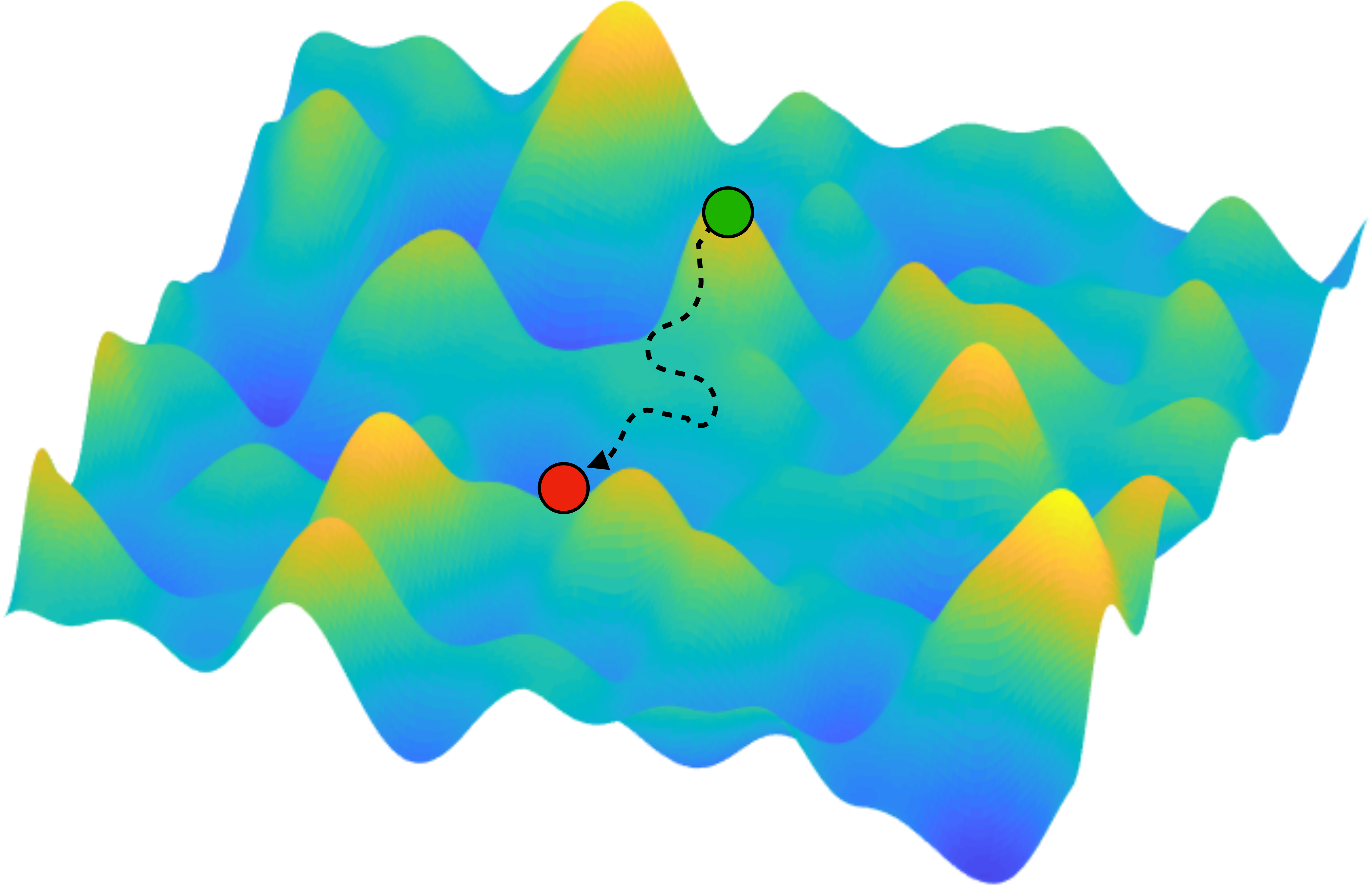
Optimization manifold

Stochastic Gradient Descent (SGD)

future

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \text{perturbation}$$

current



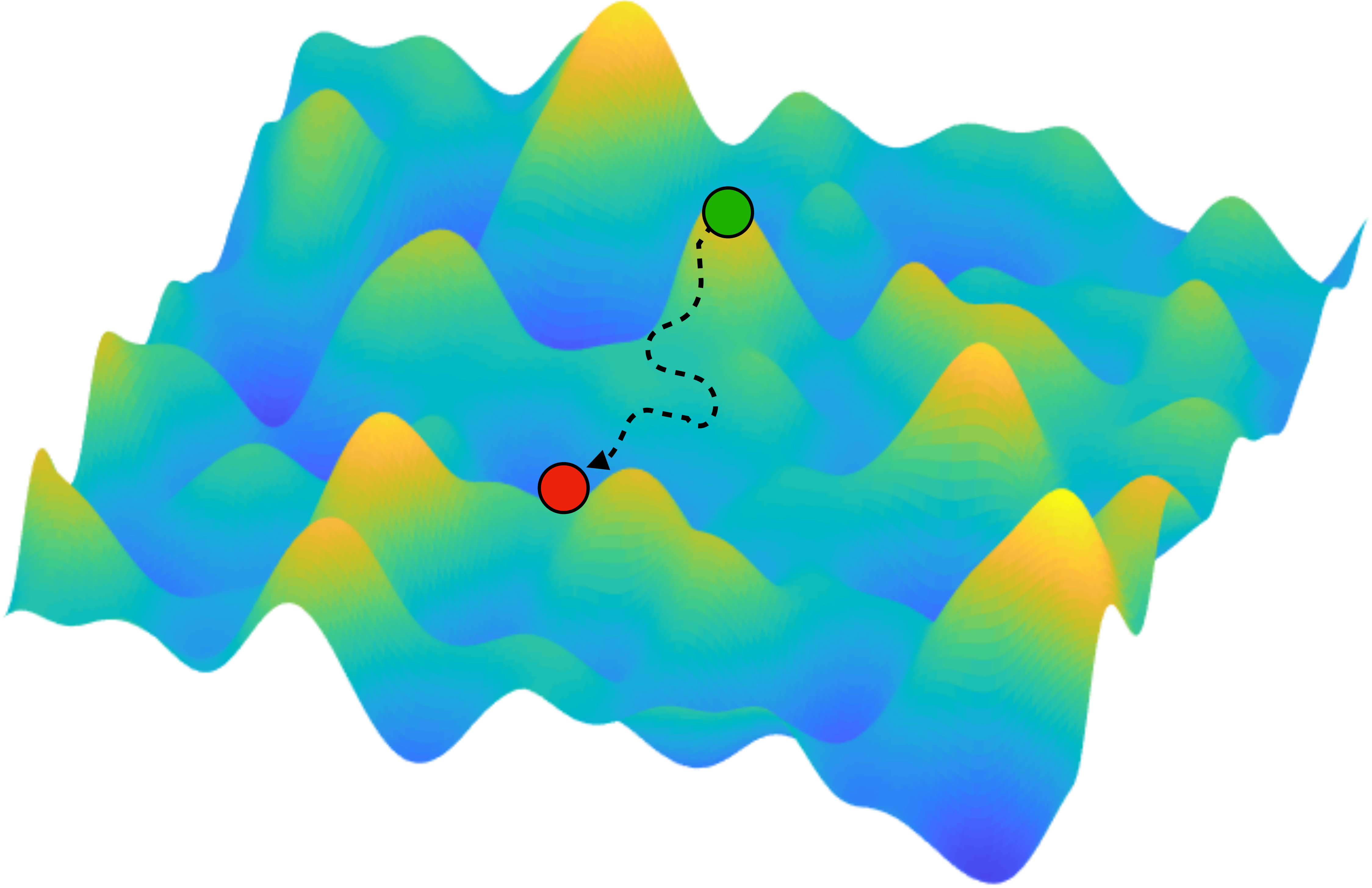
This is a Markov Chain!

Optimization manifold

Stochastic Gradient Descent (SGD)

Global discovery

Local exploration



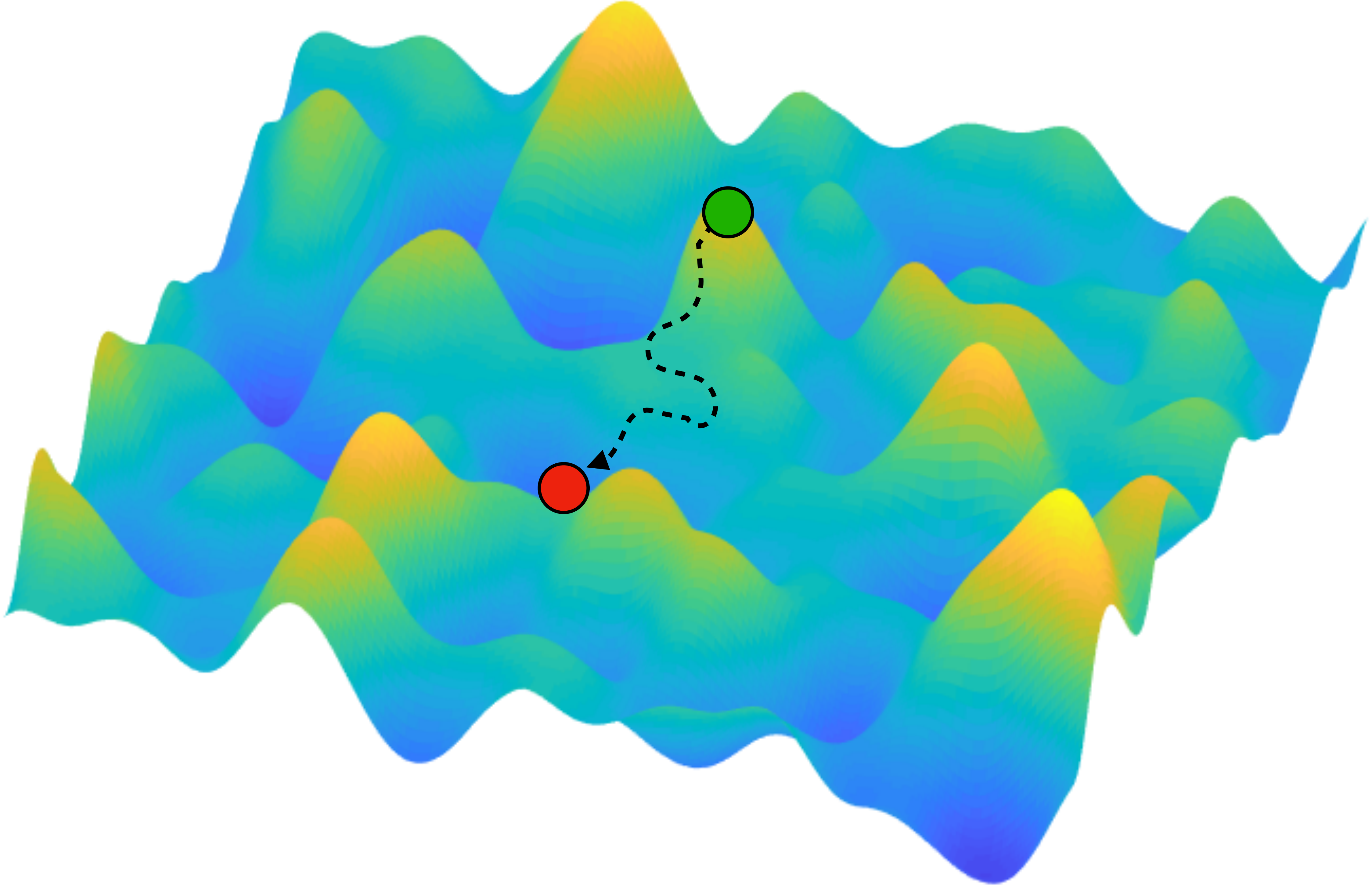
Optimization manifold

Stochastic Gradient Descent (SGD)

Global discovery

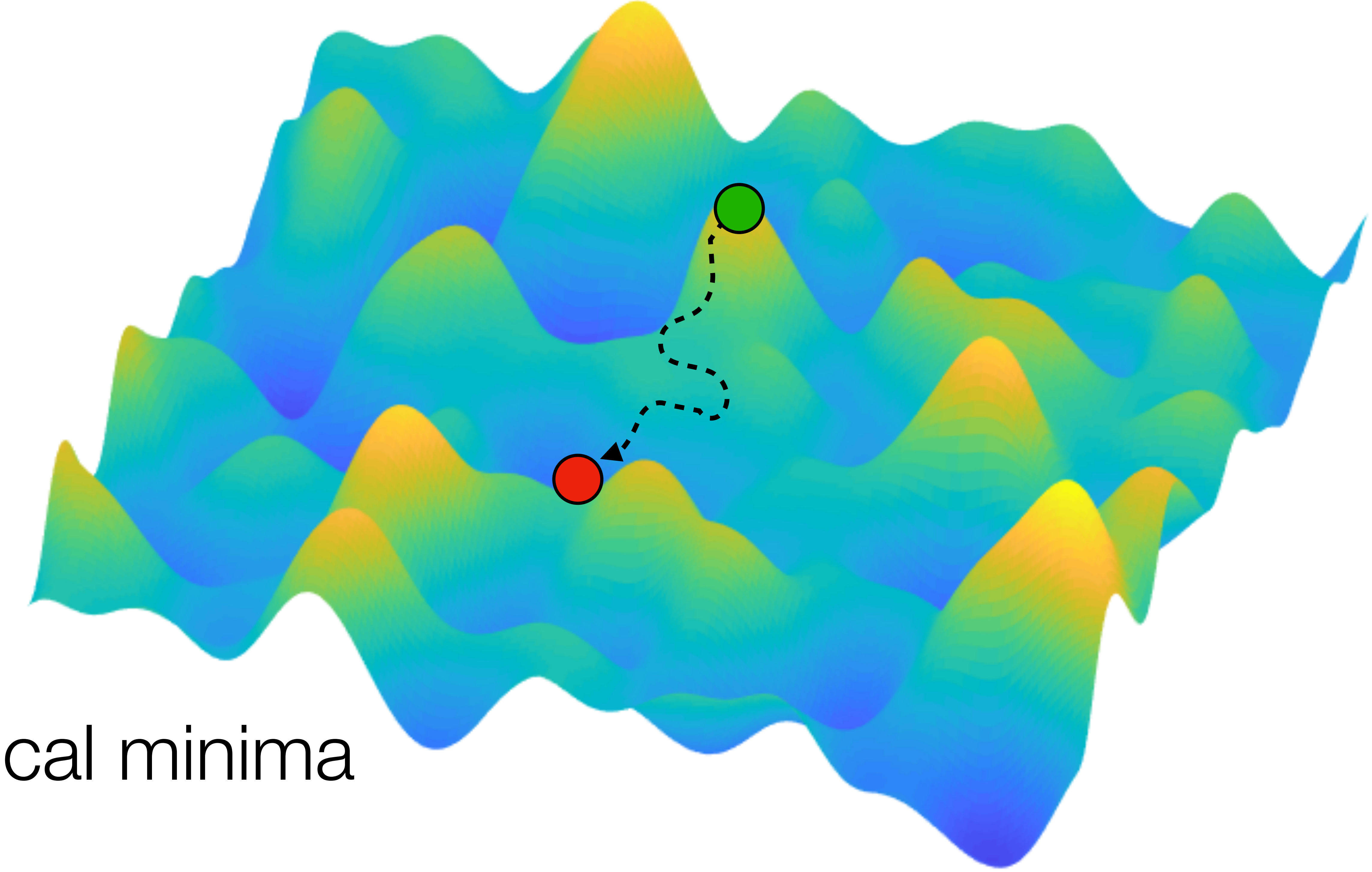
- Explore the whole manifold

Local exploration



Optimization manifold

Stochastic Gradient Descent (SGD)



Global discovery

- Explore the whole manifold

Local exploration

- Once the region is detected, reach the local minima

Stochastic Gradient Descent (SGD)

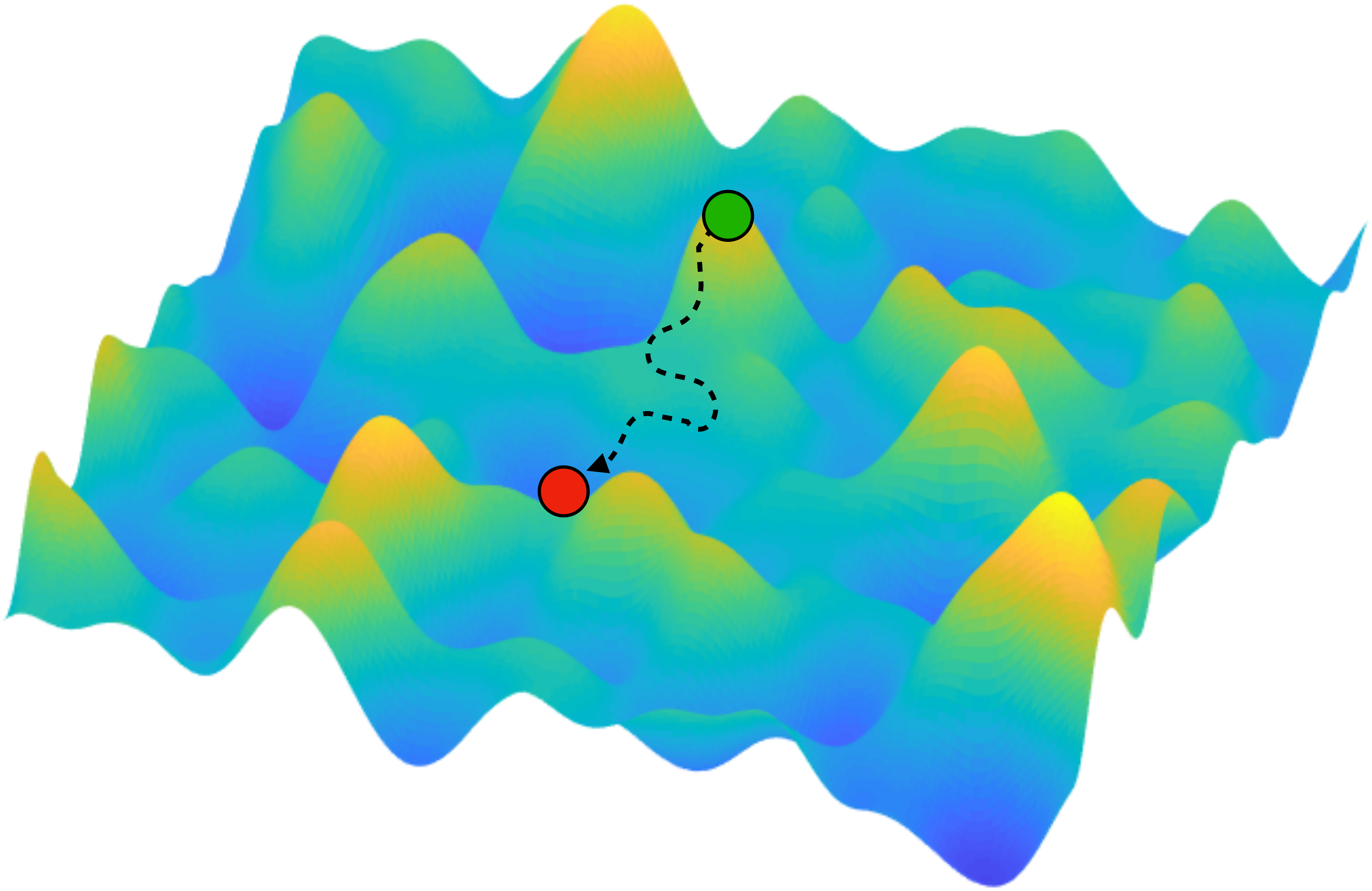
Global discovery

Local exploration

}

MCMC methods

Optimization manifold



MCMC: Bridging rendering, optimization and generative AI

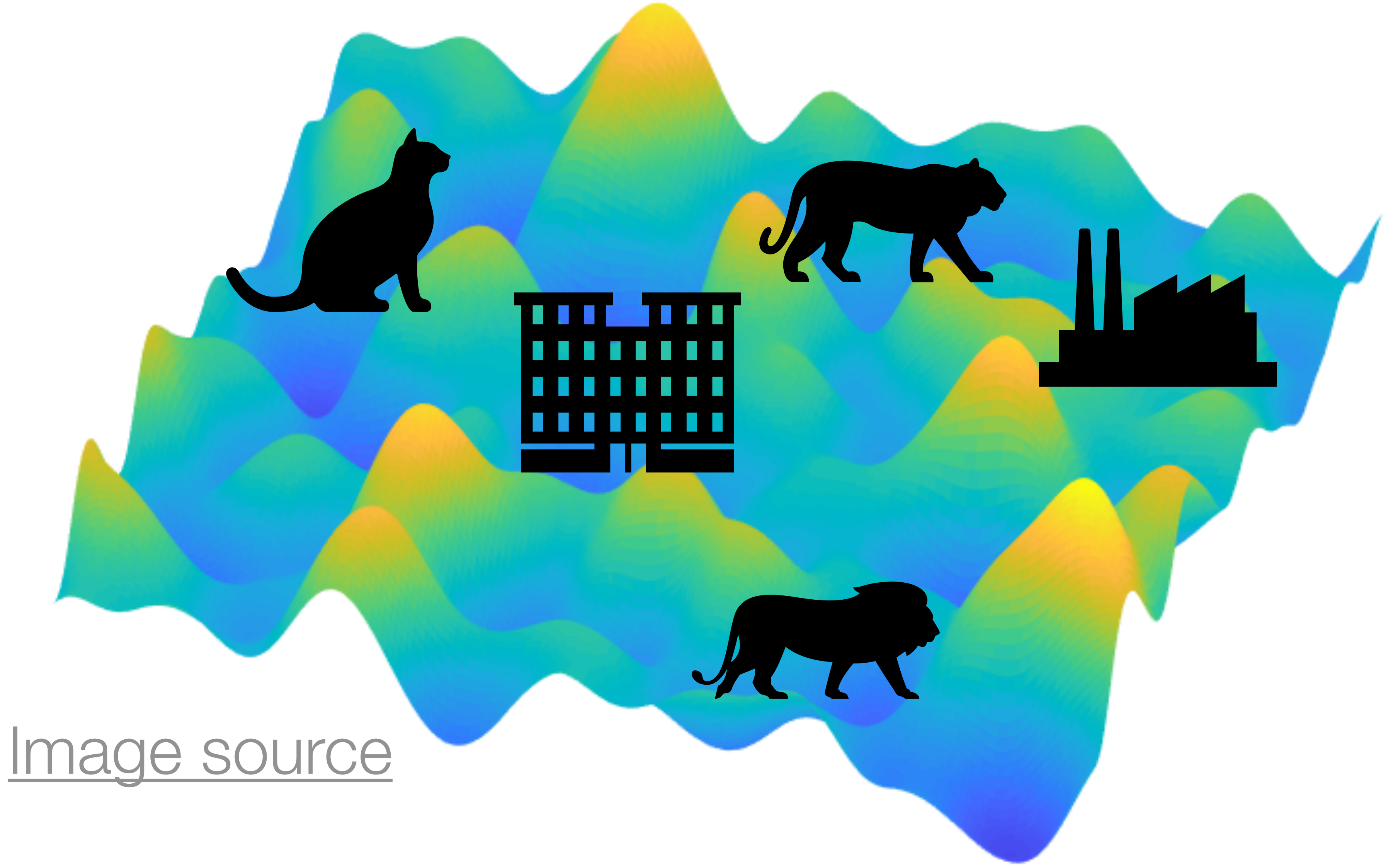


Image source

generative AI

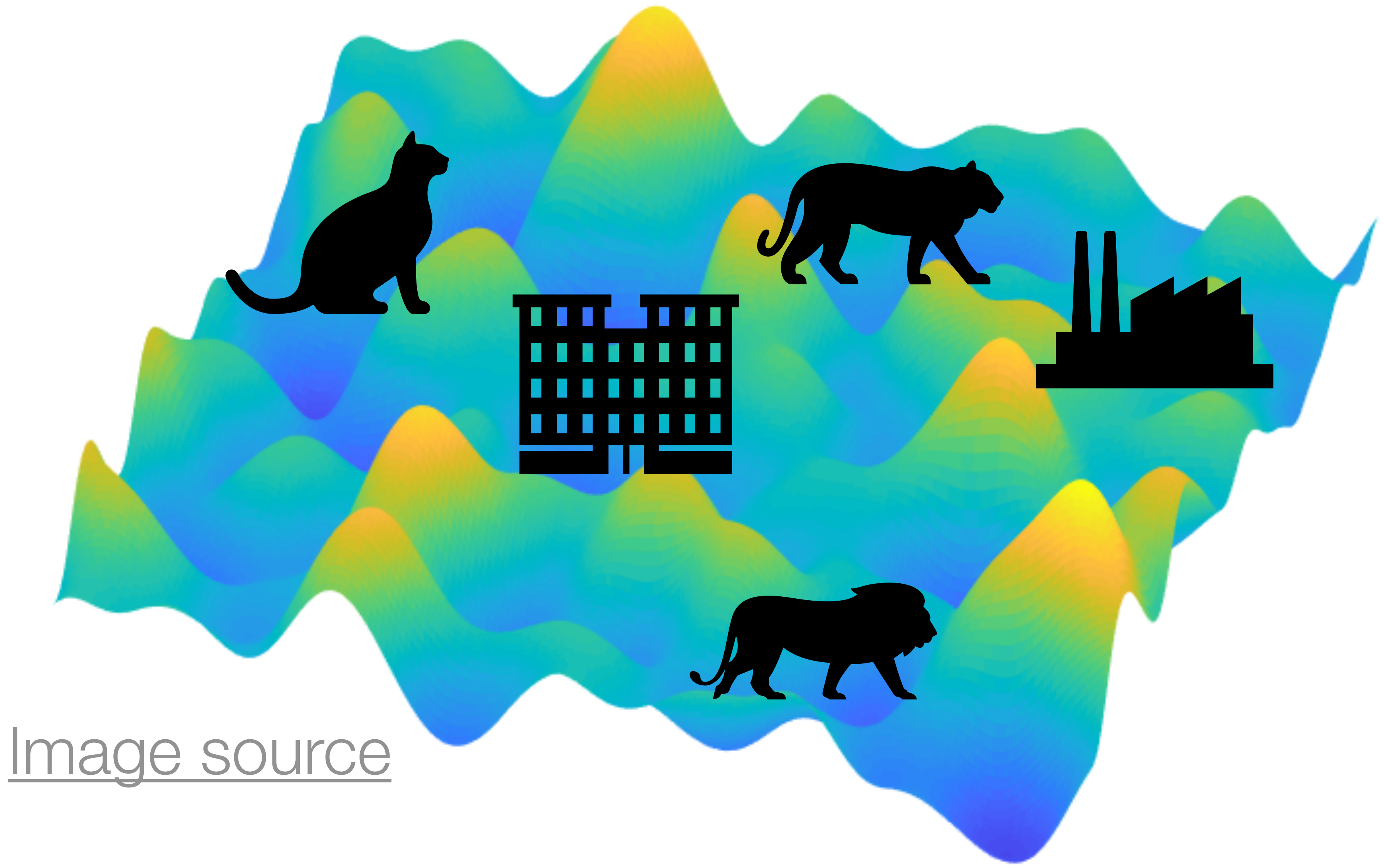


Image source

generative AI

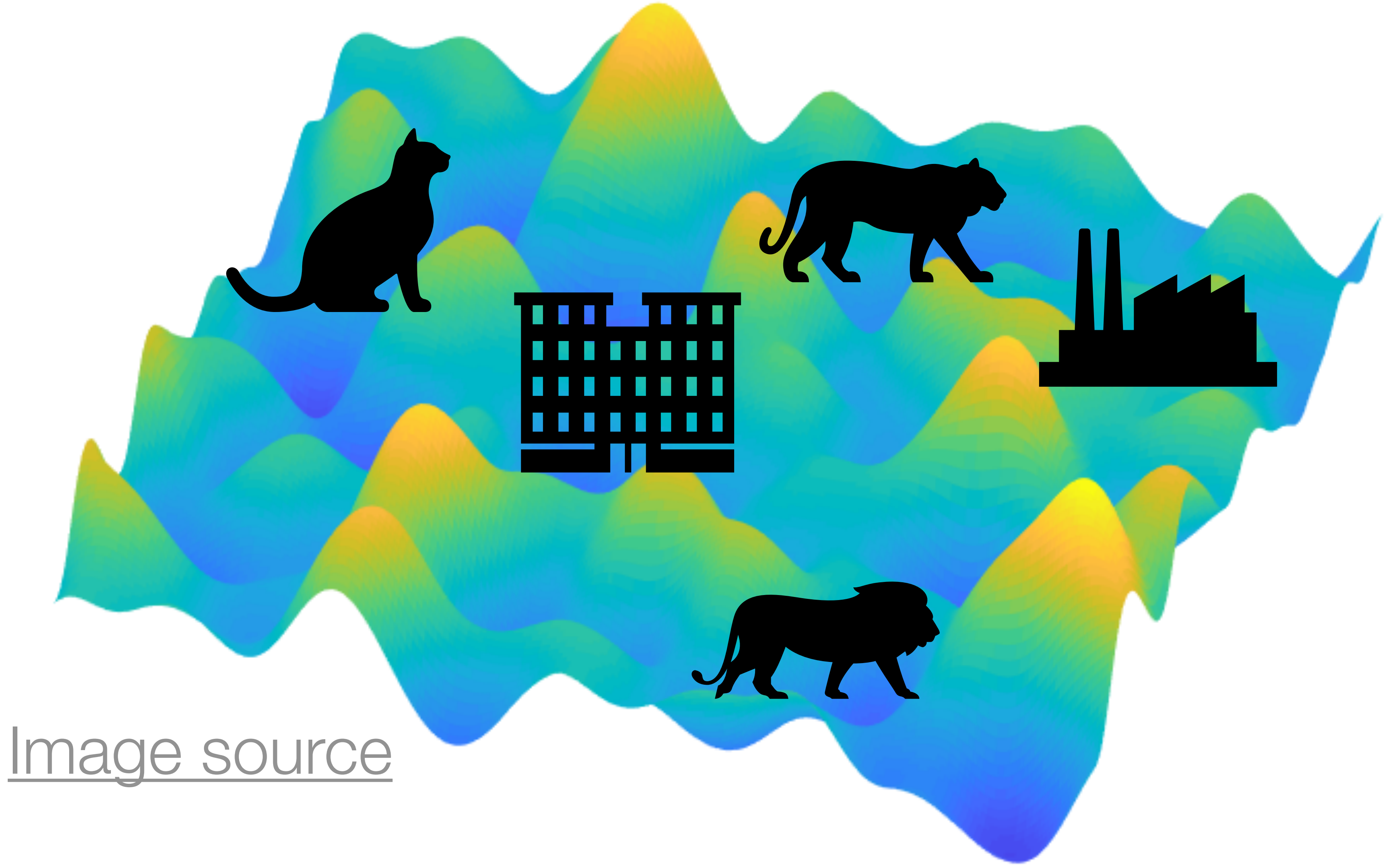


Image source

generative AI

Prompt: Create an image that represents the bridge between physically based rendering, optimization and generative AI

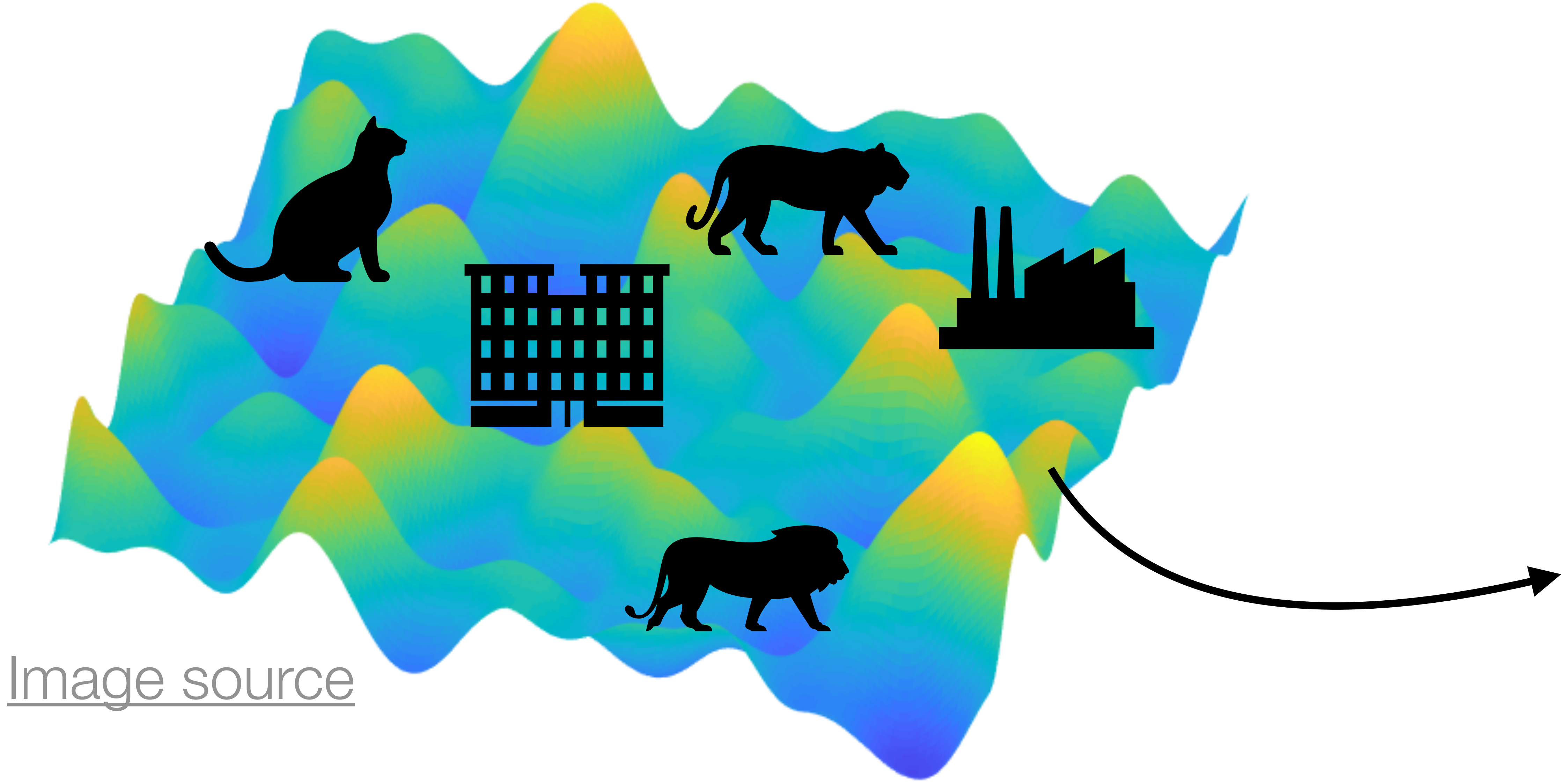


Image source

generative AI

Prompt: Create an image that represents the bridge between physically based rendering, optimization and generative AI

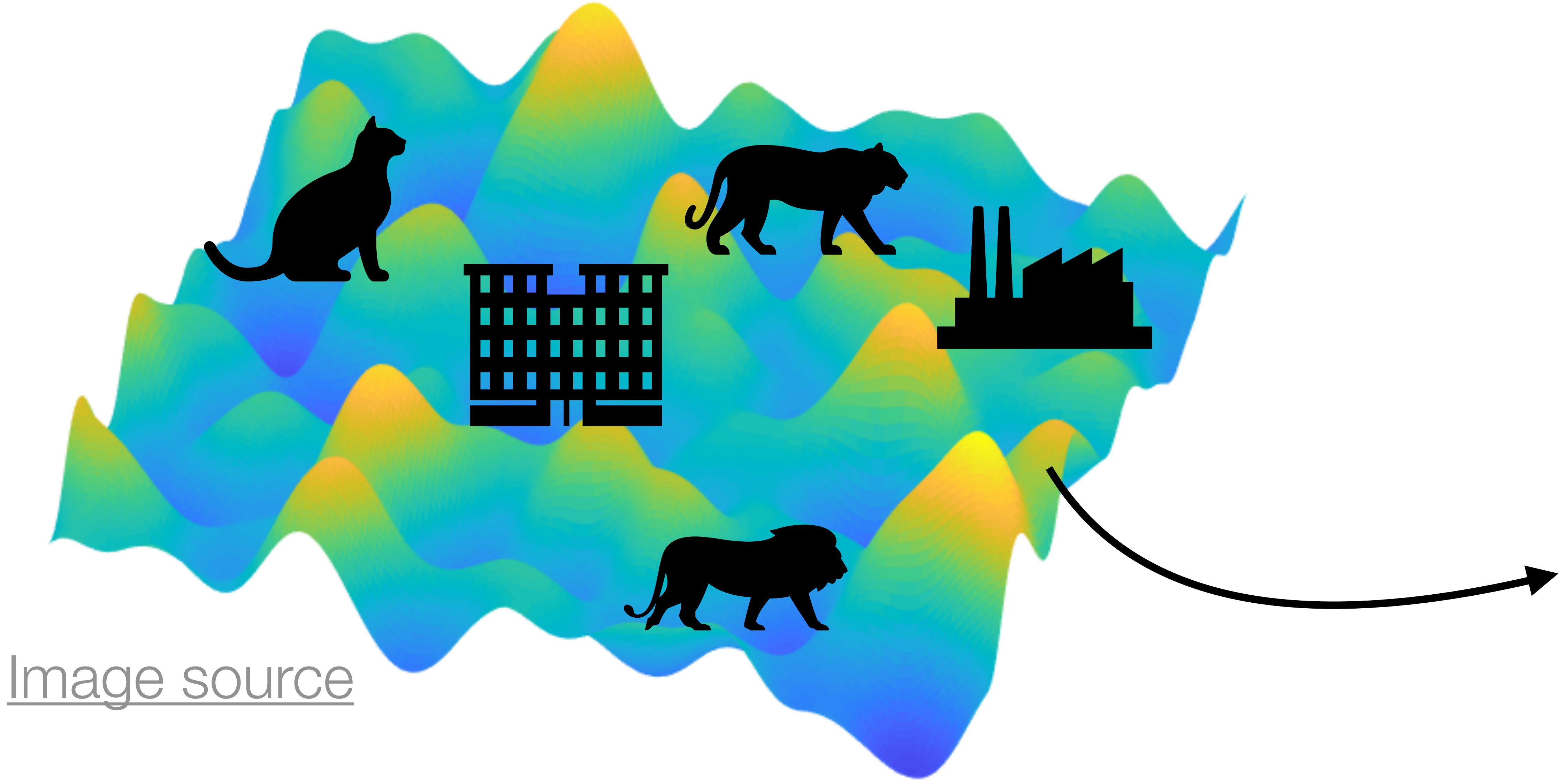


Image source

generative AI

Prompt: Create an image that represents the bridge between physically based rendering, optimization and generative AI

(using Microsoft Copilot)

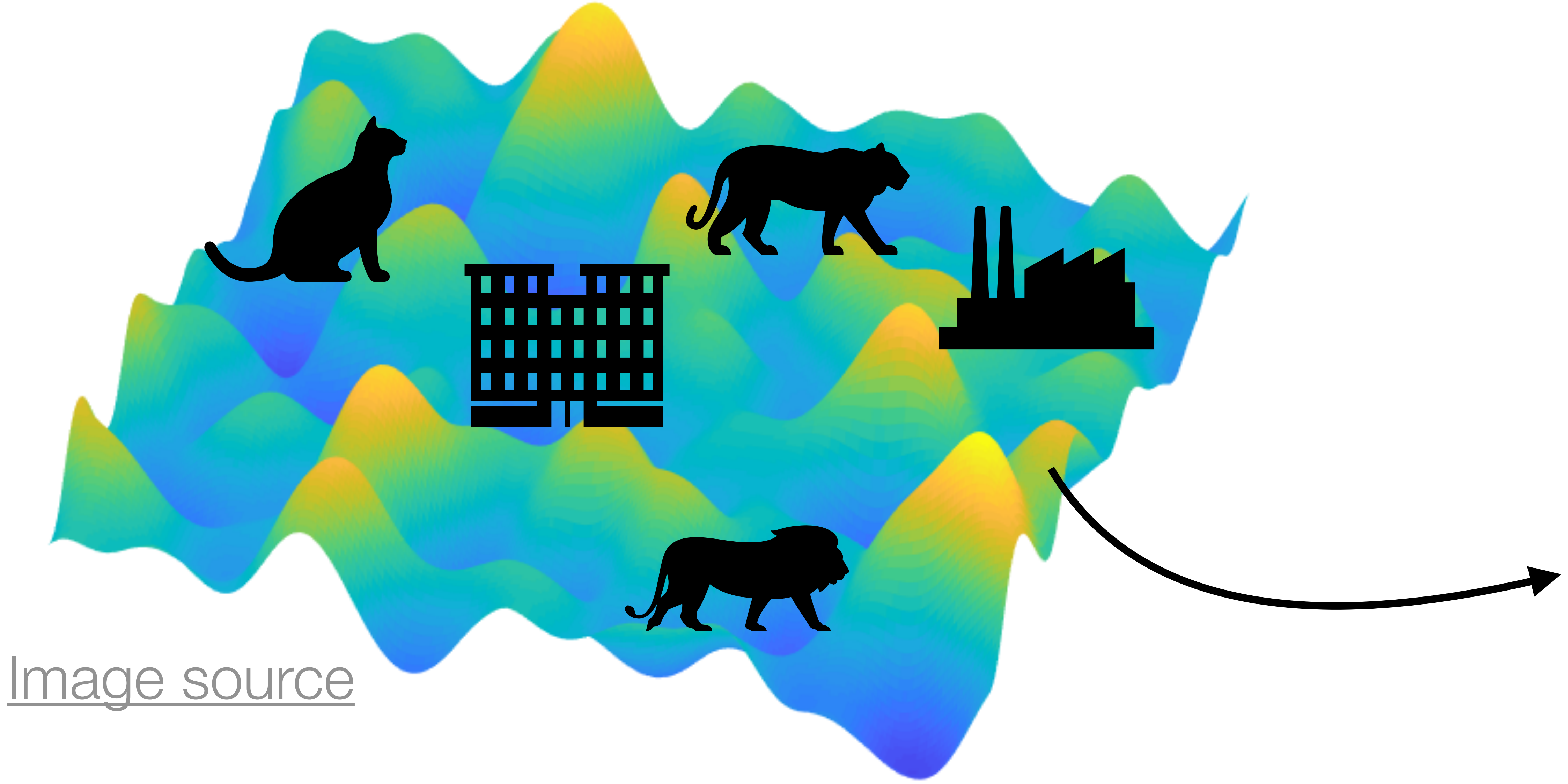


Image source

generative AI

Prompt: Create an image that represents the bridge between physically based rendering, optimization and generative AI

(using Microsoft Copilot)

What kind of generative models are available?

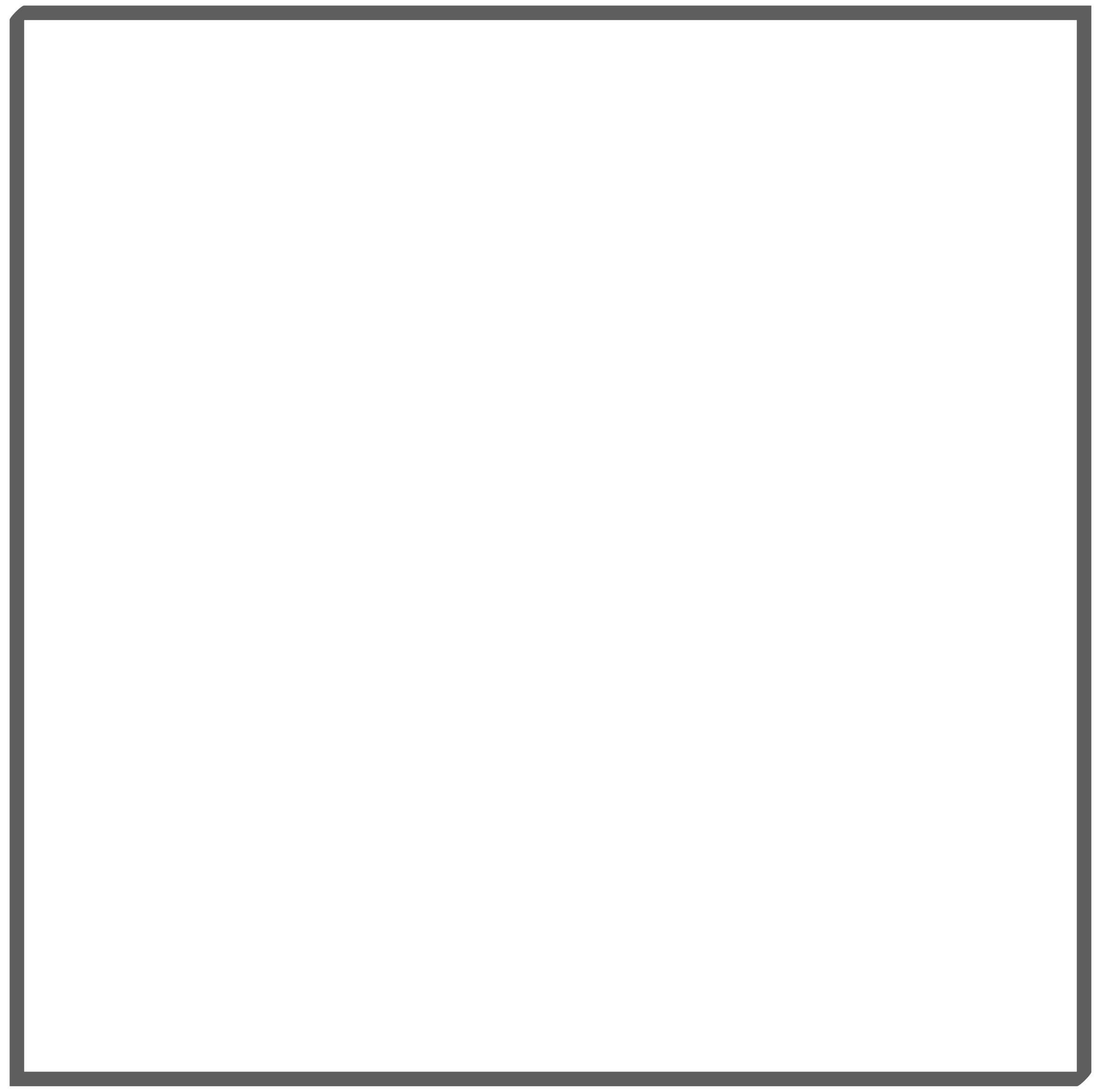
What kind of generative models are available?

- Energy-based models
- Score-based models
- Diffusion models

⋮

Which one to chose and why?

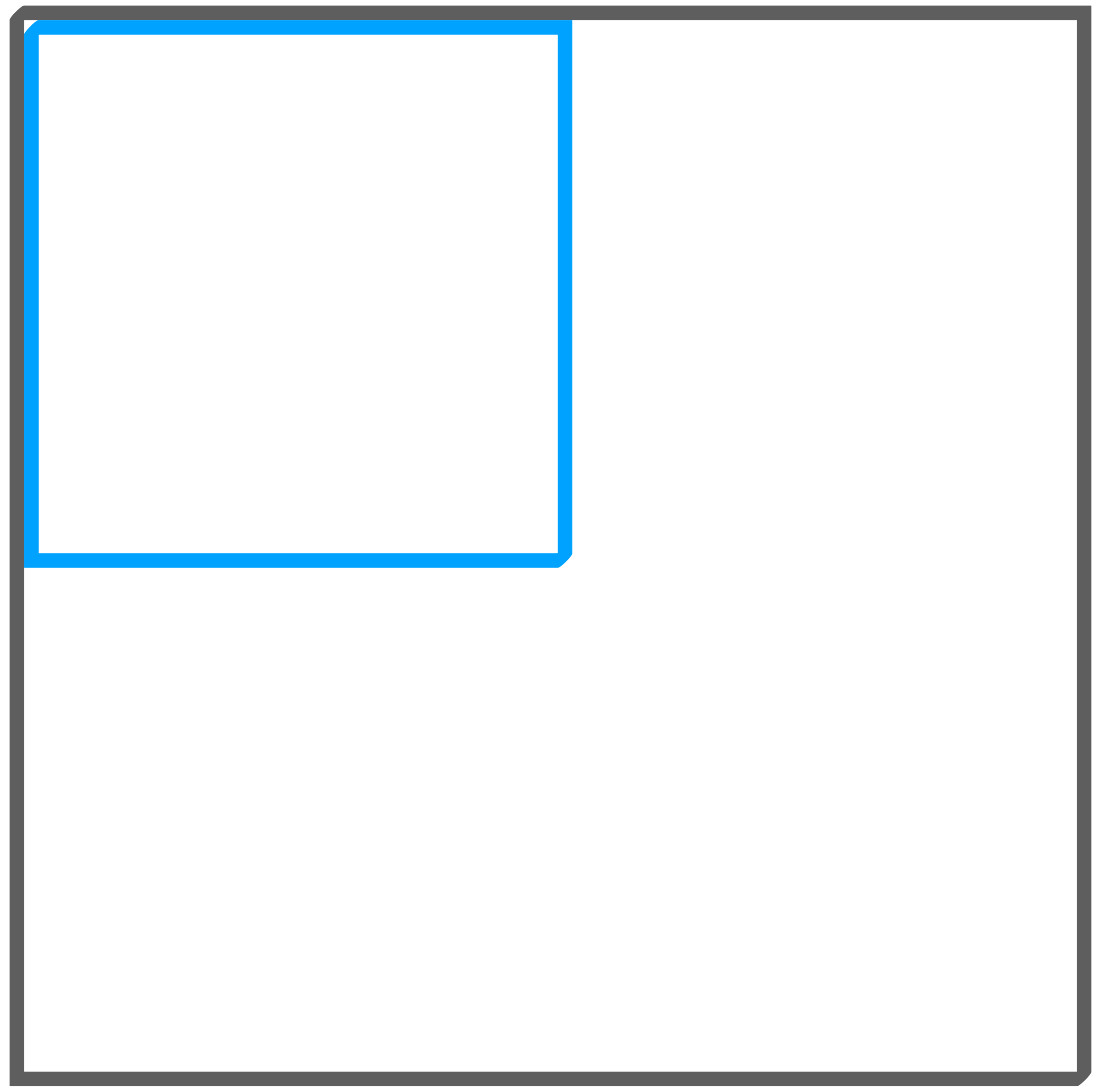
Task: Composing an image



Task: Composing an image

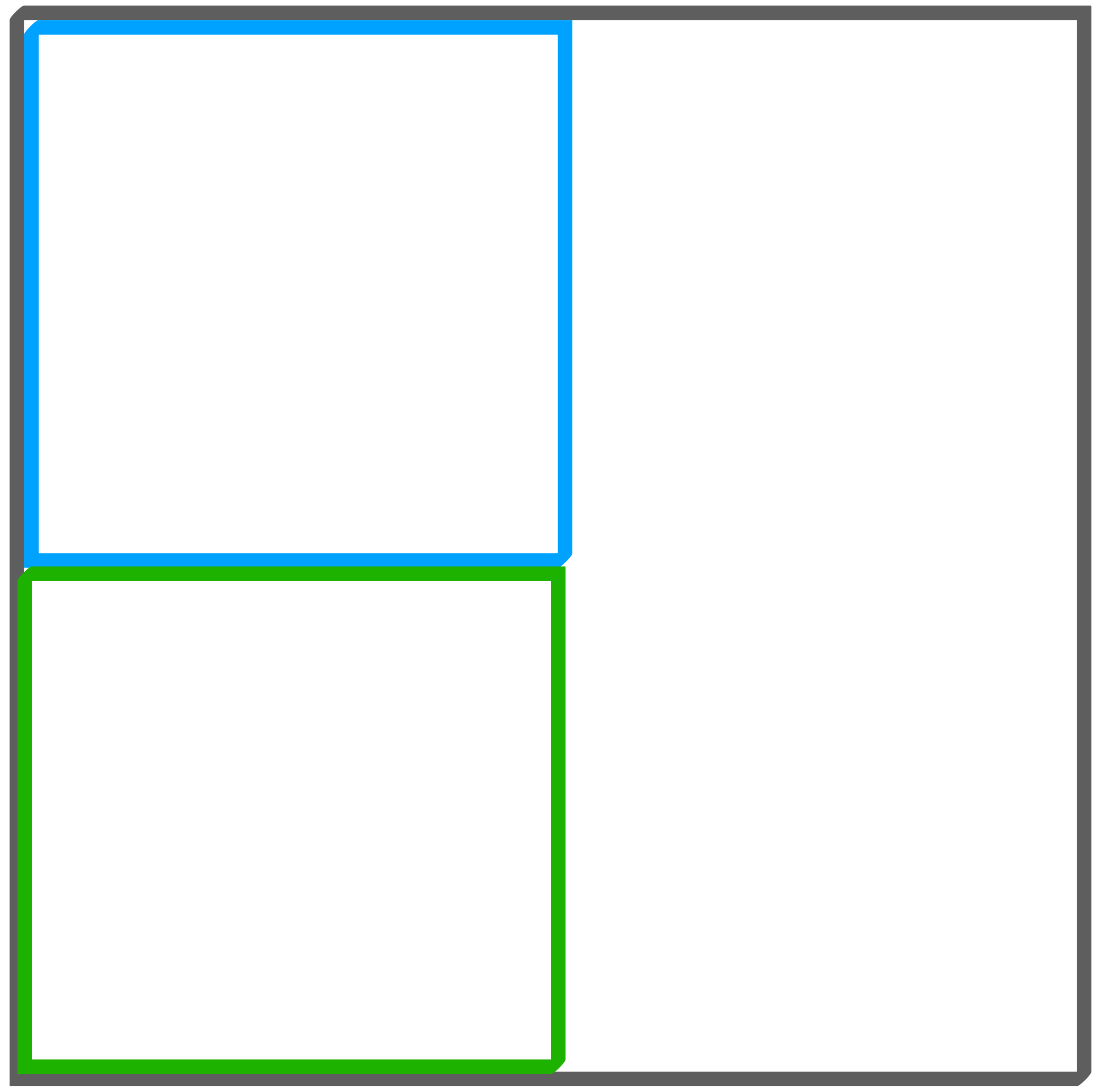
- ⬛ An oil painting of an ocean scene.
- 🟦
- 🟩
- 🟥
- 🟧
- 🟪

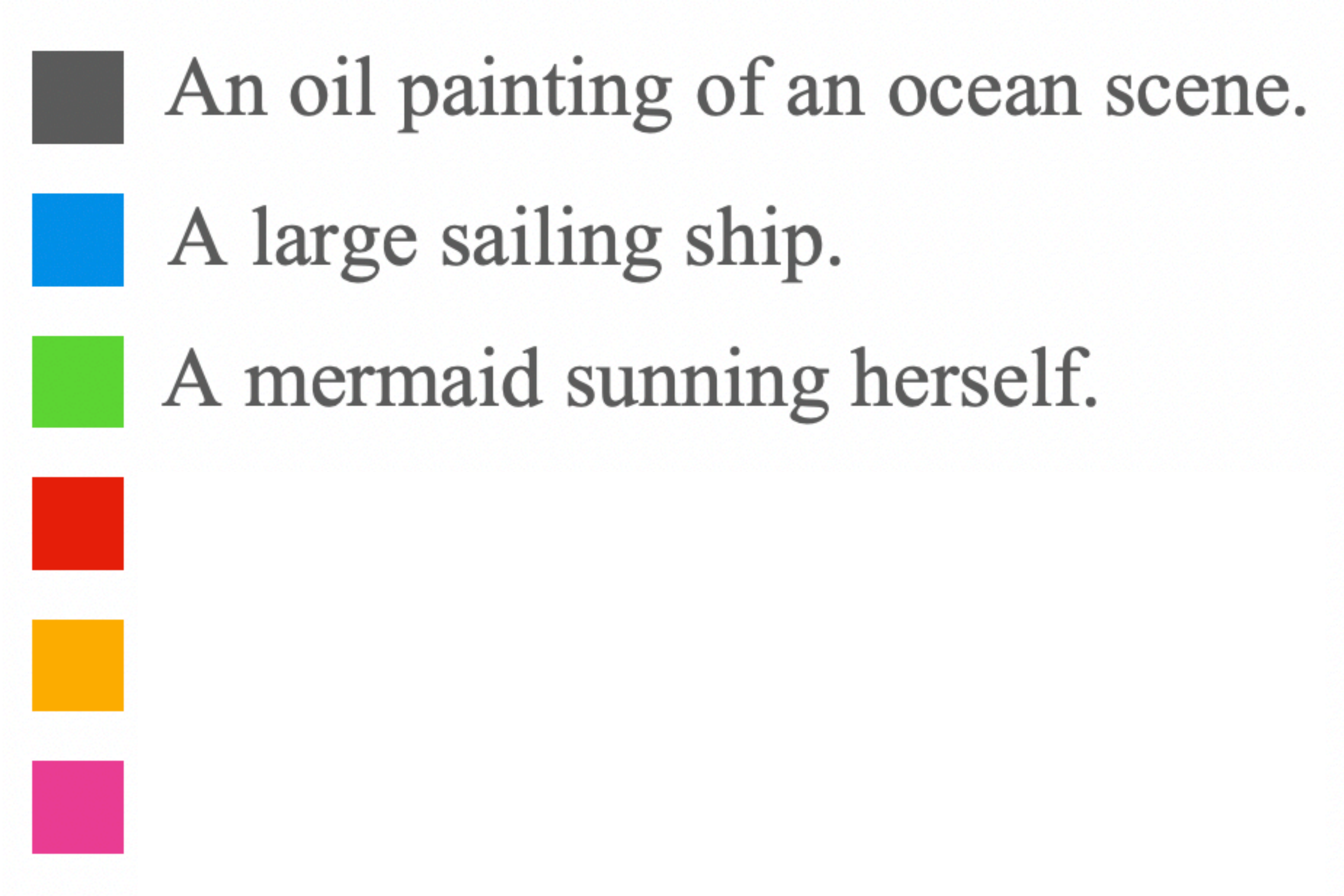
Task: Composing an image



- ⬛ An oil painting of an ocean scene.
- 🟦 A large sailing ship.
- 🟩
- 🟥
- 🟧
- 🟪

Task: Composing an image





⬛ An oil painting of an ocean scene.

🟦 A large sailing ship.

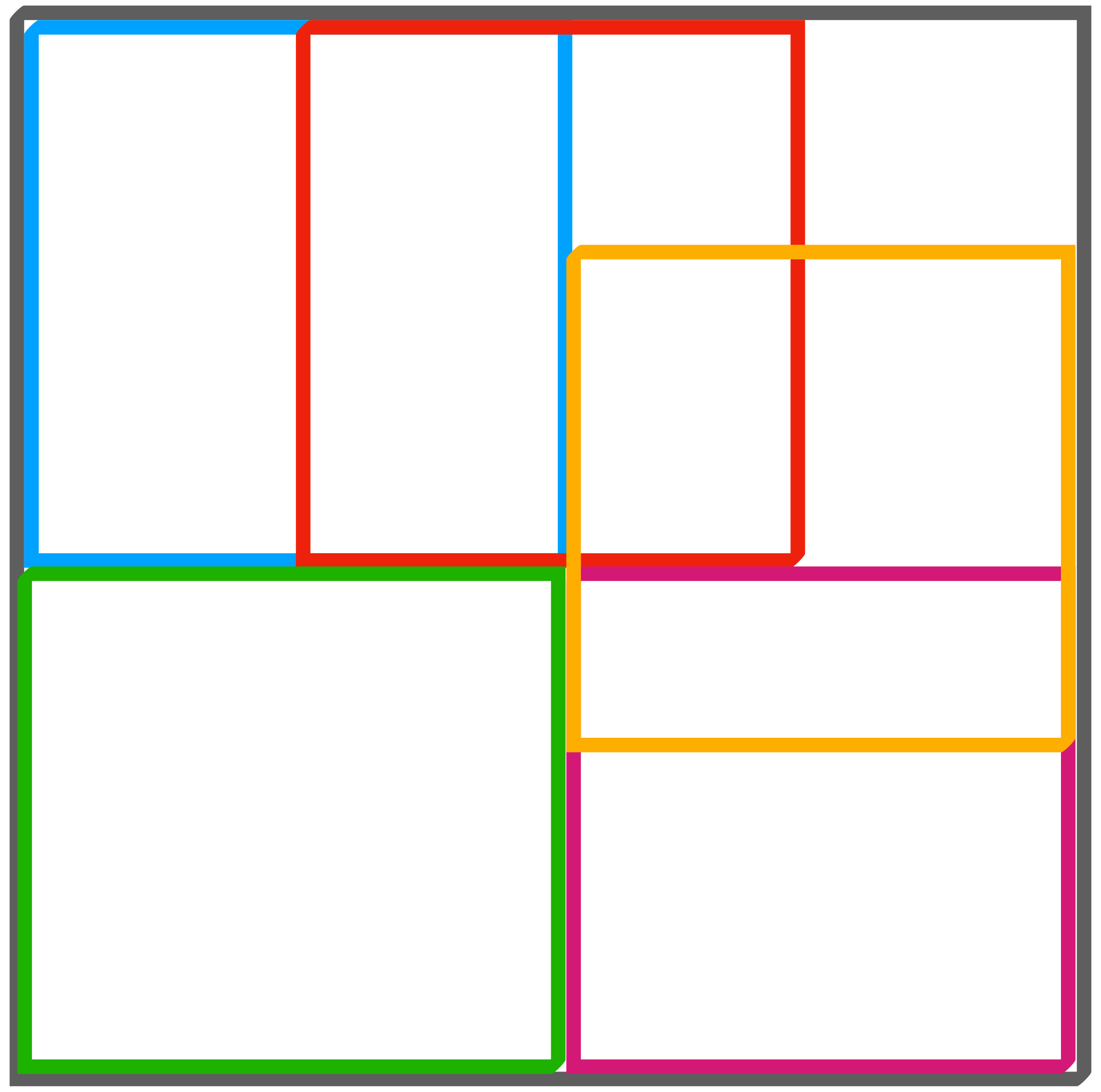
🟩 A mermaid sunning herself.

🟥

🟧

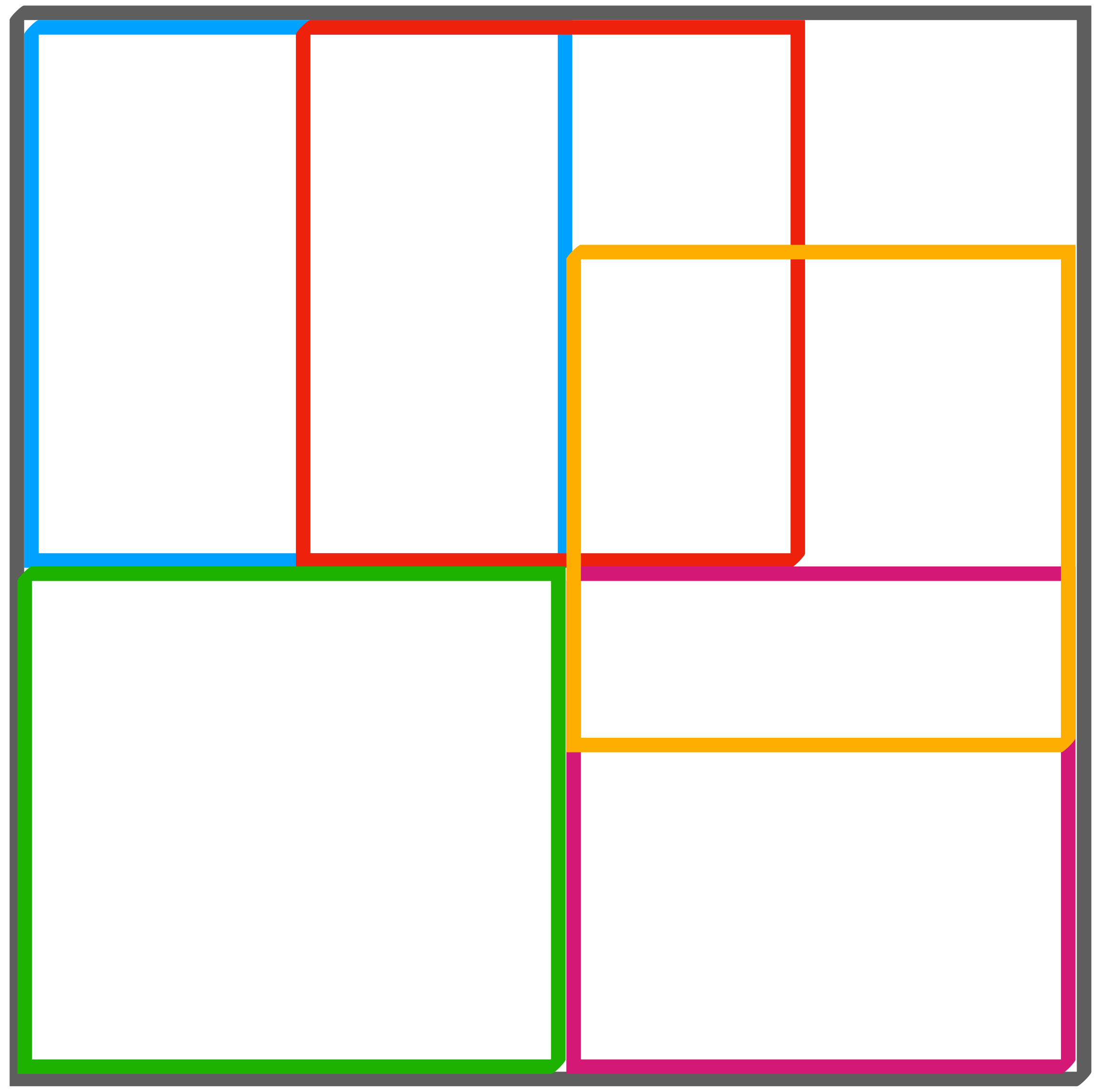
🟪

Task: Composing an image



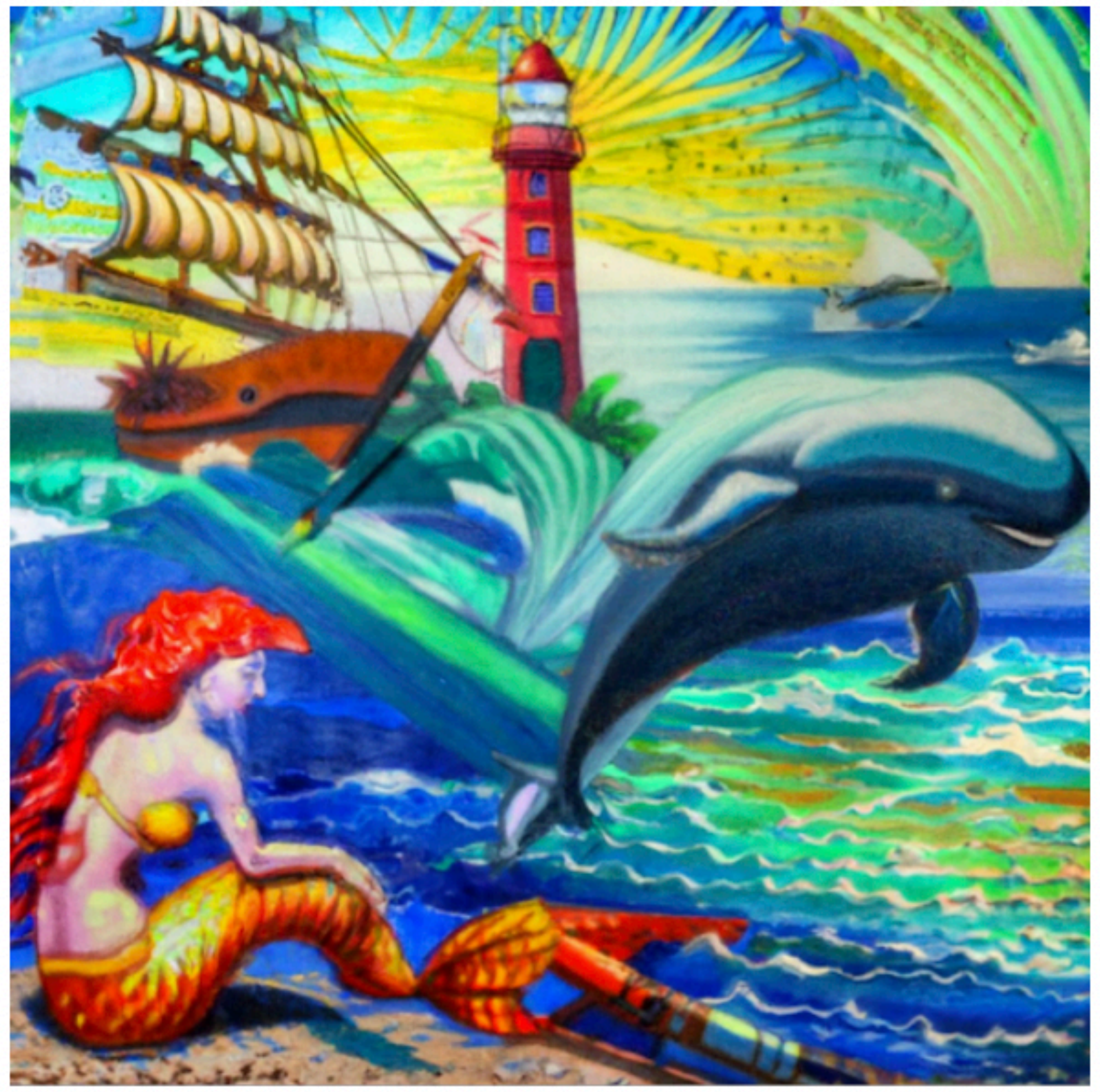
- ⬛ An oil painting of an ocean scene.
- 🟦 A large sailing ship.
- 🟩 A mermaid sunning herself.
- 🟥 A lighthouse.
- 🟧 A curious whale surfacing.
- 🟪 The ocean.

Task: Composing an image



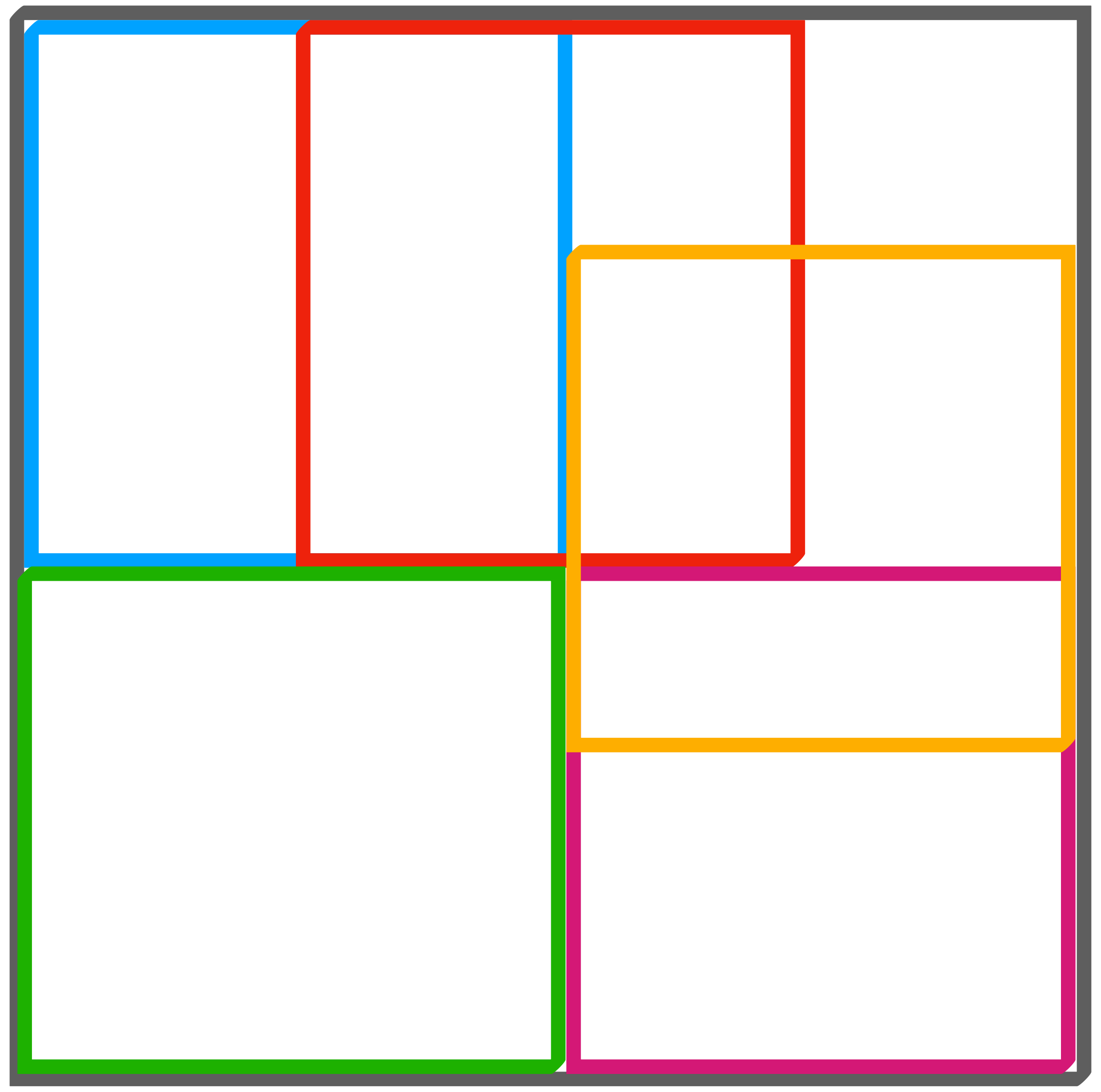
- ⬛ An oil painting of an ocean scene.
- 🟦 A large sailing ship.
- 🟩 A mermaid sunning herself.
- 🟥 A lighthouse.
- 🟧 A curious whale surfacing.
- 🟪 The ocean.

=



Task: Composing an image

What do we need for such a problem?



⬛ An oil painting of an ocean scene.

🟦 A large sailing ship.

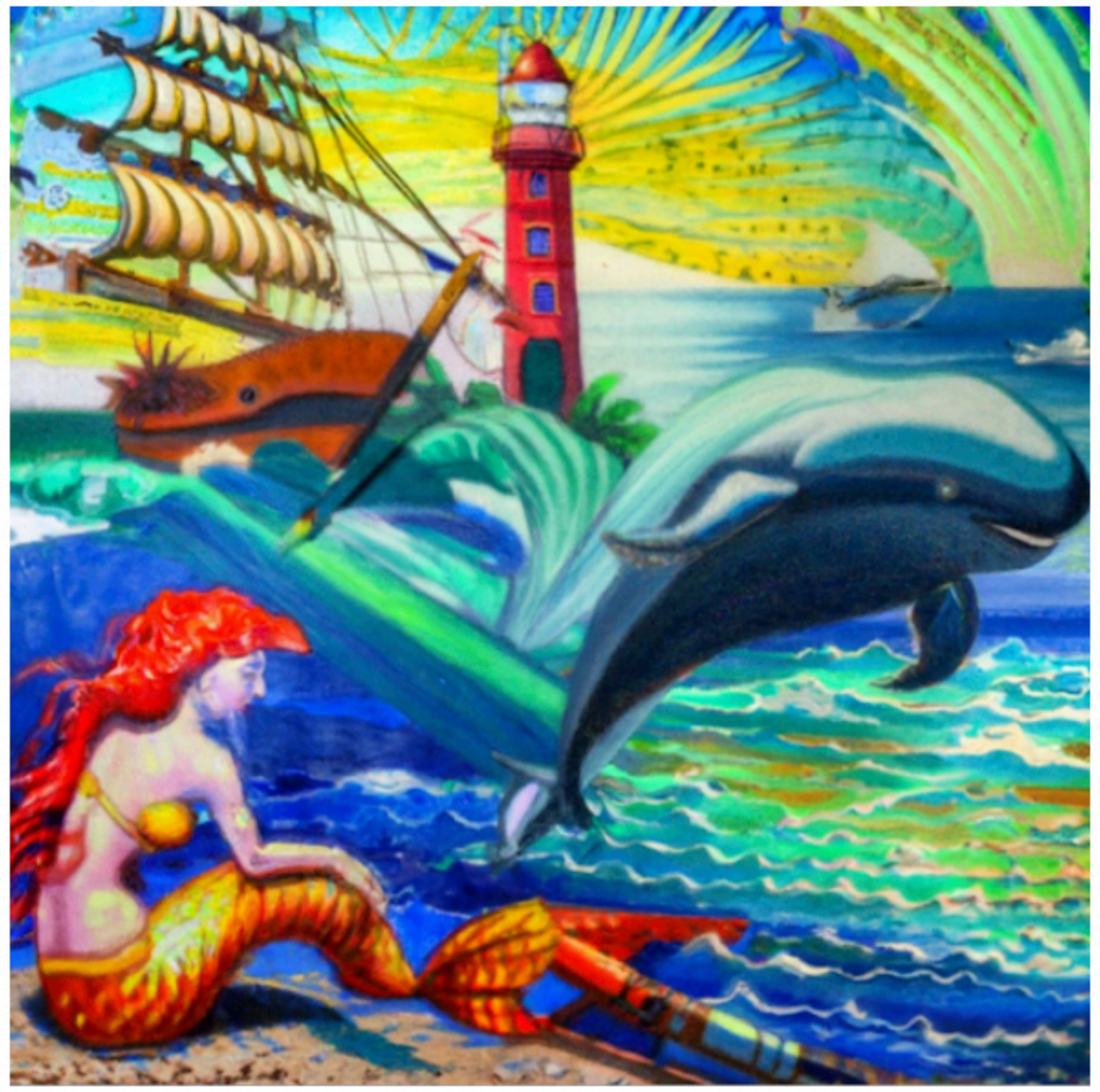
🟩 A mermaid sunning herself.

🟥 A lighthouse.

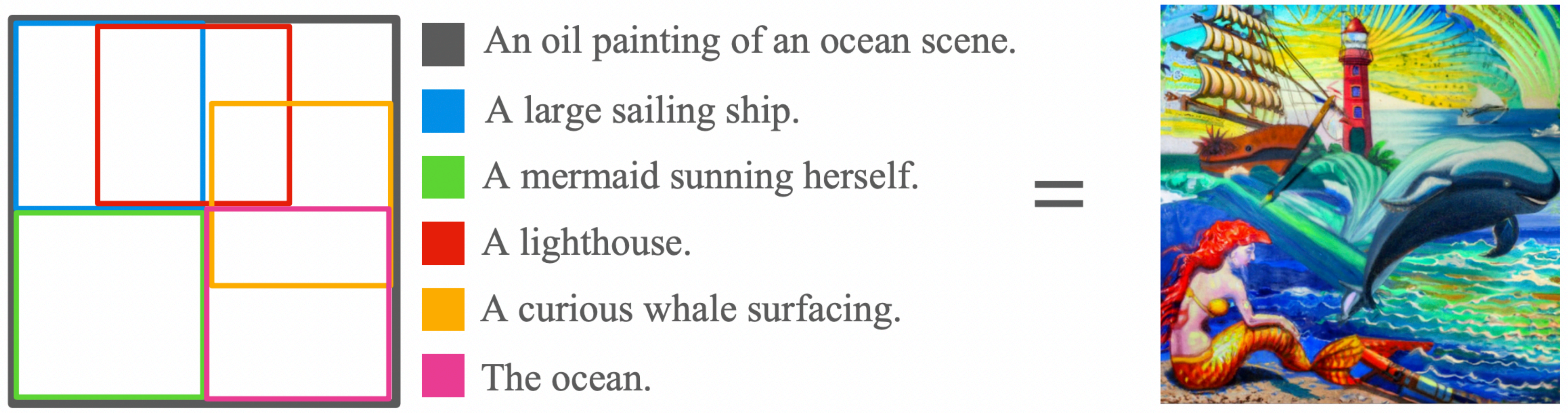
🟧 A curious whale surfacing.

🟪 The ocean.

=



Task: Composing an image

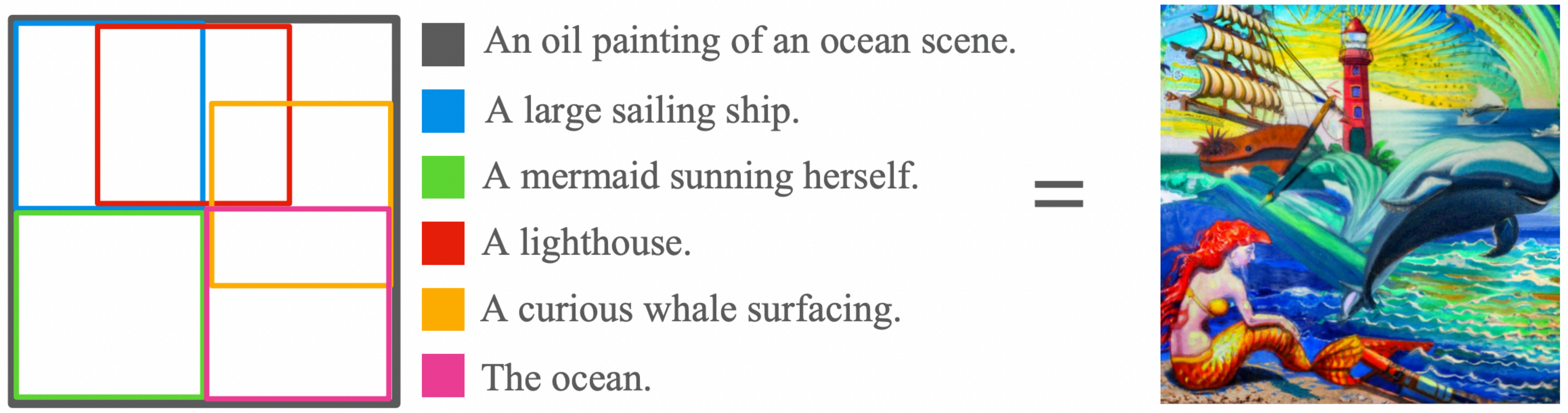


⬛ An oil painting of an ocean scene.
🟦 A large sailing ship.
🟩 A mermaid sunning herself.
🟥 A lighthouse.
🟧 A curious whale surfacing.
🟪 The ocean.

=

- Compositional generation with Energy-Based Diffusion Models and MCMC [Du et al. 2024]

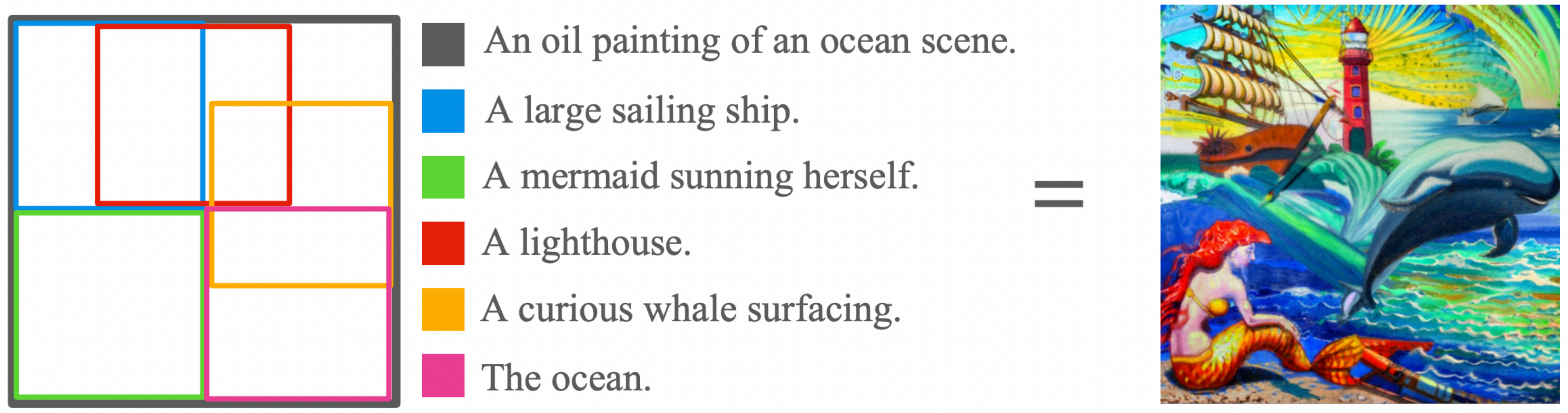
Task: Composing an image


An oil painting of an ocean scene.
A large sailing ship.
A mermaid sunning herself.
A lighthouse.
A curious whale surfacing.
The ocean.
=

- Compositional generation with Energy-Based Diffusion Models and MCMC [Du et al. 2024]

- It’s the sampler and not the architecture which needs to be changed!

Task: Composing an image

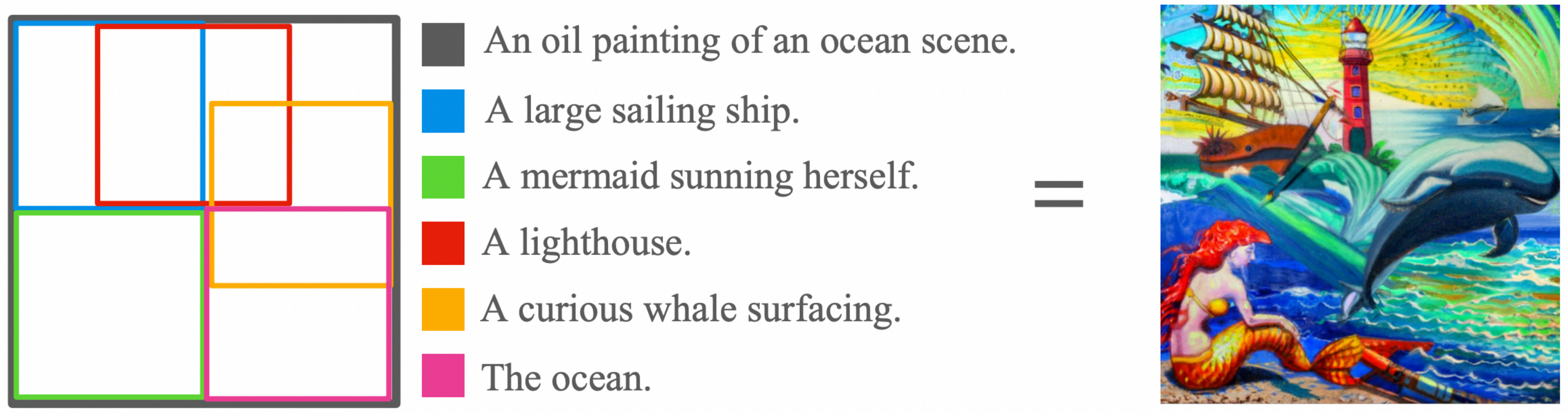

⬛ An oil painting of an ocean scene.
🟦 A large sailing ship.
🟩 A mermaid sunning herself.
🟥 A lighthouse.
🟧 A curious whale surfacing.
🟪 The ocean.
=

- Compositional generation with Energy-Based Diffusion Models and MCMC [Du et al. 2024]

- It’s the sampler and not the architecture which needs to be changed!

- Energy-based models are by construction very flexible

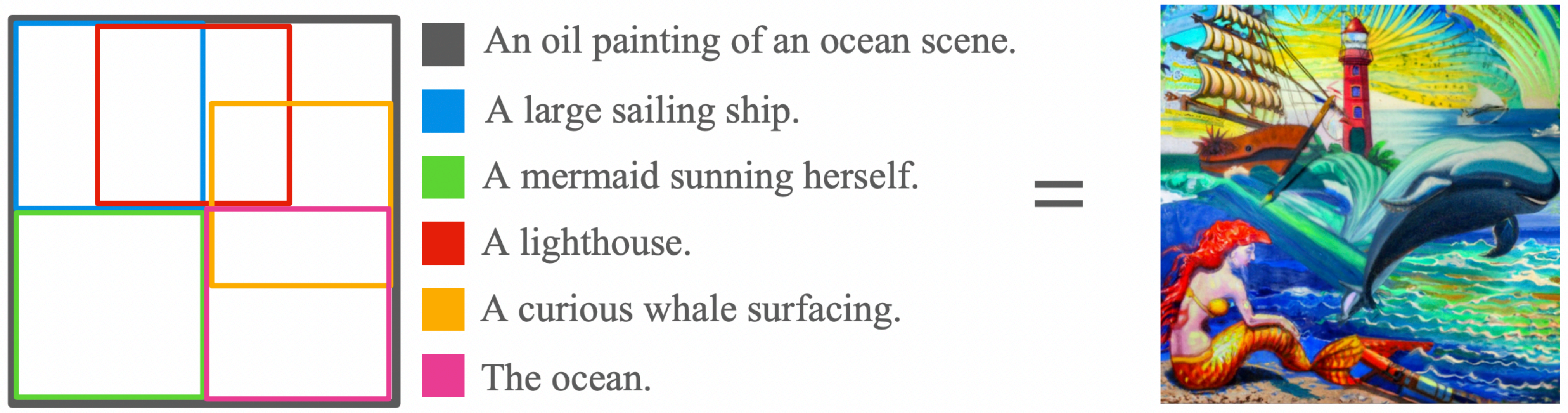
Task: Composing an image



⬛ An oil painting of an ocean scene.
🟦 A large sailing ship.
🟩 A mermaid sunning herself.
🟥 A lighthouse.
🟧 A curious whale surfacing.
🟪 The ocean.

- Compositional generation with Energy-Based Diffusion Models and MCMC [Du et al. 2024]
- It’s the sampler and not the architecture which needs to be changed!
- Energy-based models are by construction very flexible
  - They rely on MCMC sampling

Task: Composing an image


⬛ An oil painting of an ocean scene.
🟦 A large sailing ship.
🟩 A mermaid sunning herself.
🟥 A lighthouse.
🟧 A curious whale surfacing.
🟪 The ocean.
=

- Compositional generation with Energy-Based Diffusion Models and MCMC [Du et al. 2024]

- It’s the sampler and not the architecture which needs to be changed!

- Energy-based models are by construction very flexible
    - They rely on MCMC sampling

- We will go in details later on!

Theoretical background

Stochastic Differential Equations (SDEs)

Markov chain Monte Carlo (MCMC) Methods

Theoretical background

Stochastic Differential Equations (SDEs)

Markov chain Monte Carlo (MCMC) Methods

MCMC in Rendering

MC Integration / MIS / Limitations

Metropolis light Transport

Theoretical background

Stochastic Differential Equations (SDEs)

Markov chain Monte Carlo (MCMC) Methods

MCMC in Rendering

MC Integration / MIS / Limitations

Metropolis light Transport

MCMC in Optimization

Stochastic Gradient Descent (SGD)

Stochastic Gradient Langevin Dynamics

Bayesian inference using SGD

Theoretical background

Stochastic Differential Equations (SDEs)

Markov chain Monte Carlo (MCMC) Methods

MCMC in Rendering

MC Integration / MIS / Limitations

Metropolis light Transport

MCMC in Optimization

Stochastic Gradient Descent (SGD)

Stochastic Gradient Langevin Dynamics

Bayesian inference using SGD

MCMC in Generative AI

From VAEs to Diffusion models

Energy-based models (EBMs)

MCMC methods for EBMs

Score-based Generative models

Theoretical background

Stochastic Differential Equations (SDEs)

Markov chain Monte Carlo (MCMC) Methods

Deterministic motion

depends on history

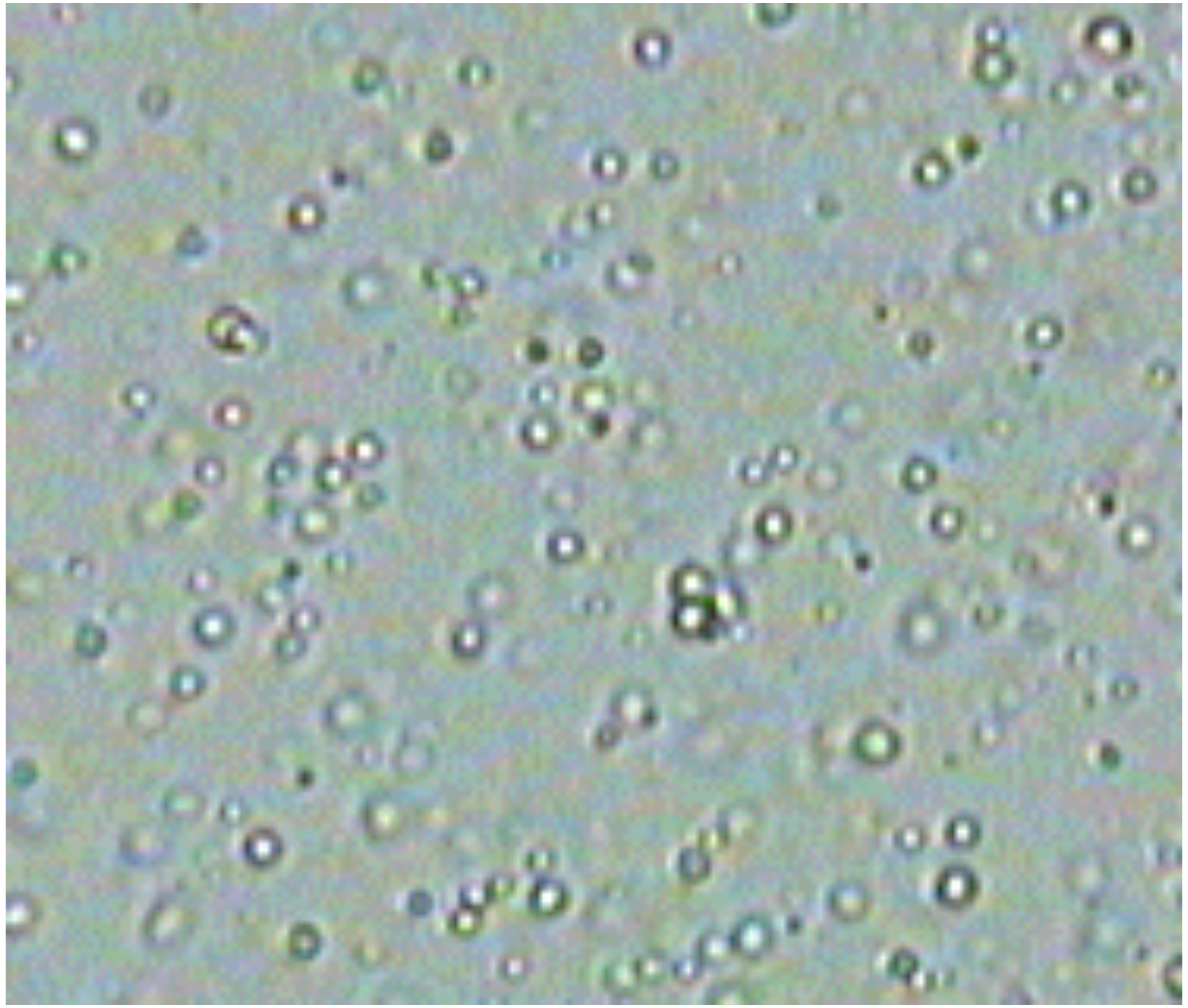
Deterministic motion

depends on history

Deterministic motion

depends on history

Random motion

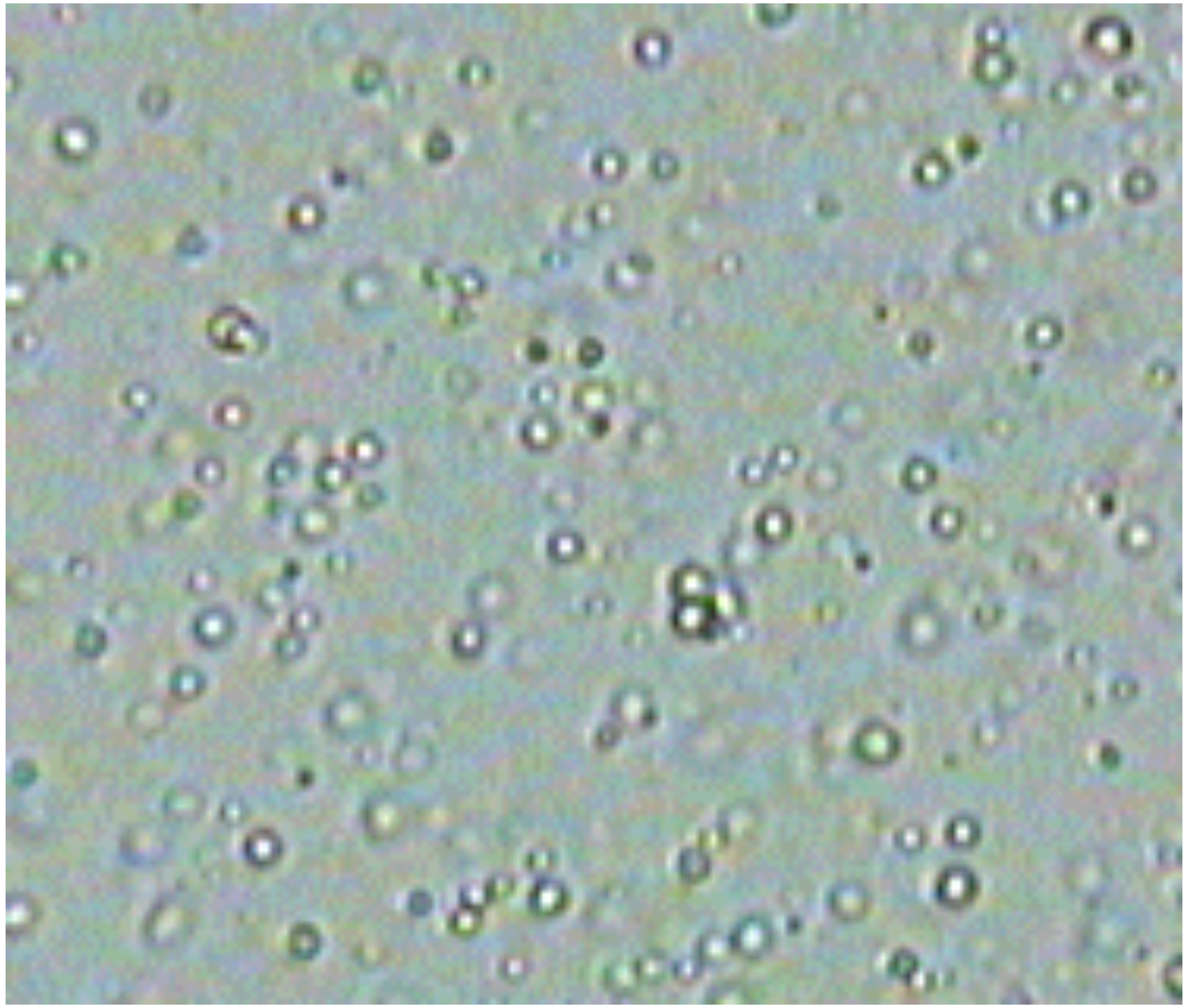


independent of history

Deterministic motion

depends on history

Random motion

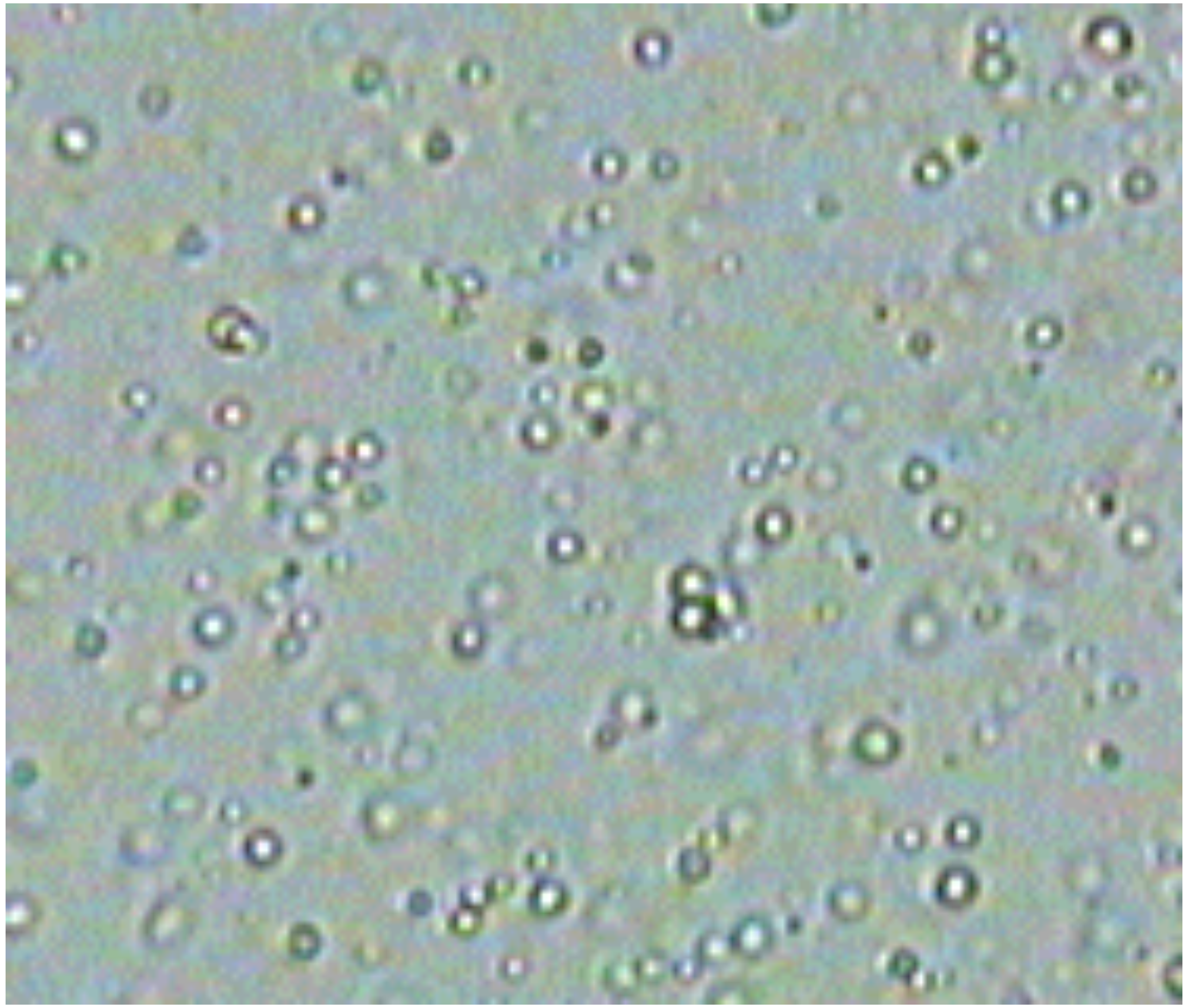


independent of history

Deterministic motion

depends on history

Random motion



independent of history

How we describe systems evolving over time?

How we describe systems evolving over time?

How do we incorporate randomness?

How we describe systems evolving over time?

How do we incorporate randomness?

How do we simulate motion numerically?

Stochastic Differential equations (SDEs)

How we describe systems evolving over time?

How do we incorporate randomness?

How do we simulate motion numerically?

Stochastic Differential equations (SDEs)

Stochastic Differential equations (SDEs)

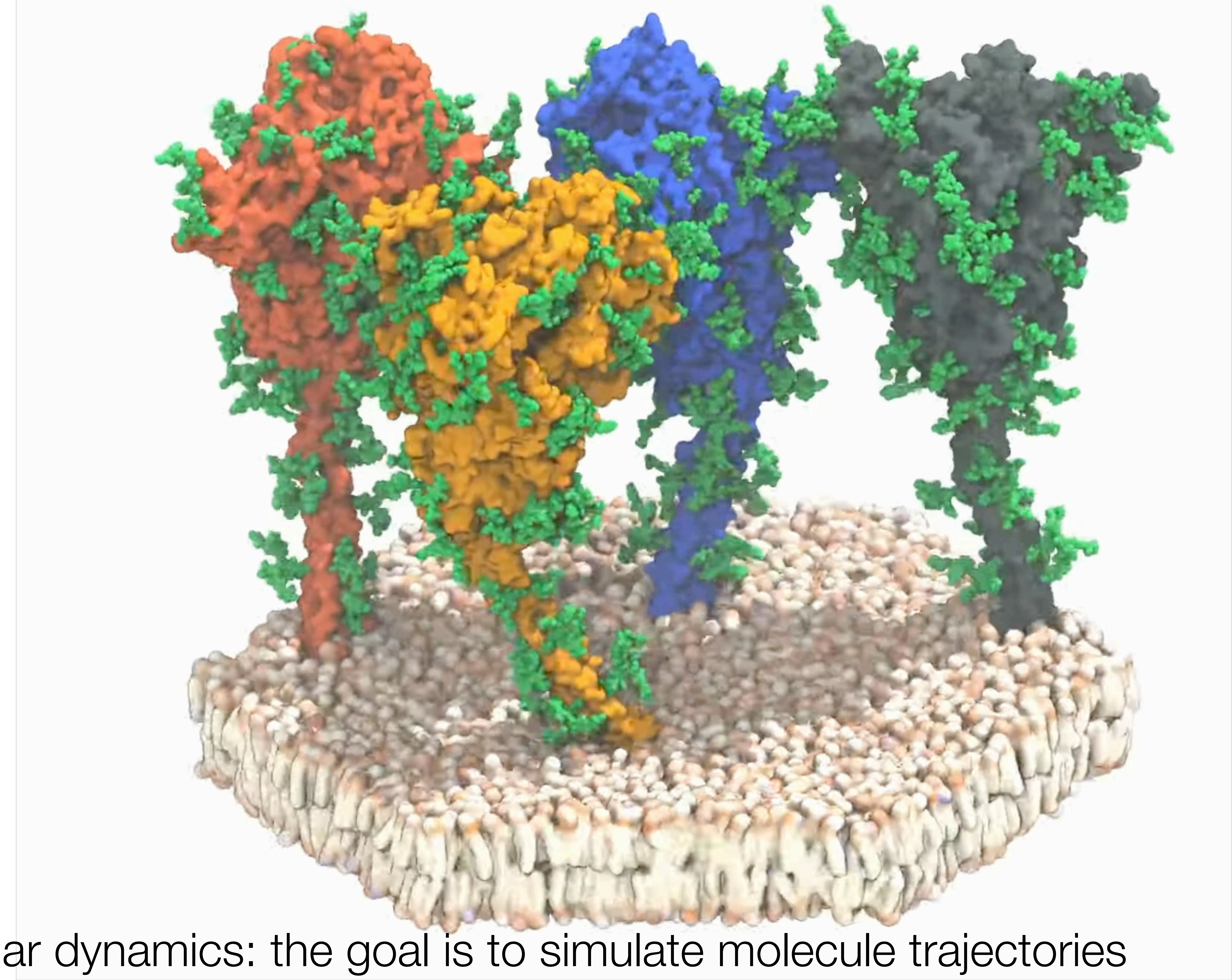
Stochastic Differential equations (SDEs)

Differential equations describe phenomena appearing throughout nature, technology & society

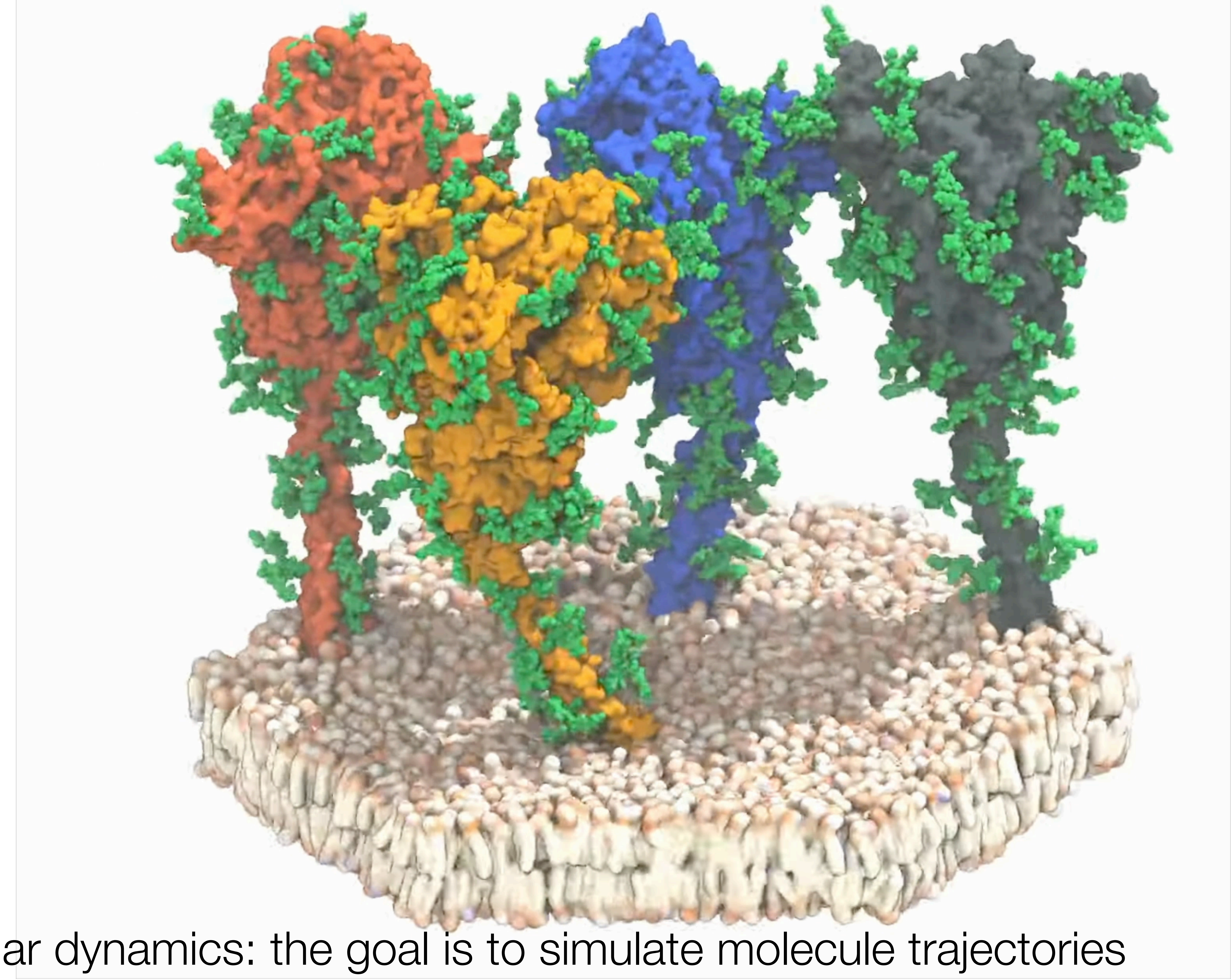
Stochastic Differential equations (SDEs)

Differential equations describe phenomena appearing throughout nature, technology & society

- Molecular dynamics: the goal is to simulate molecule trajectories



- Molecular dynamics: the goal is to simulate molecule trajectories

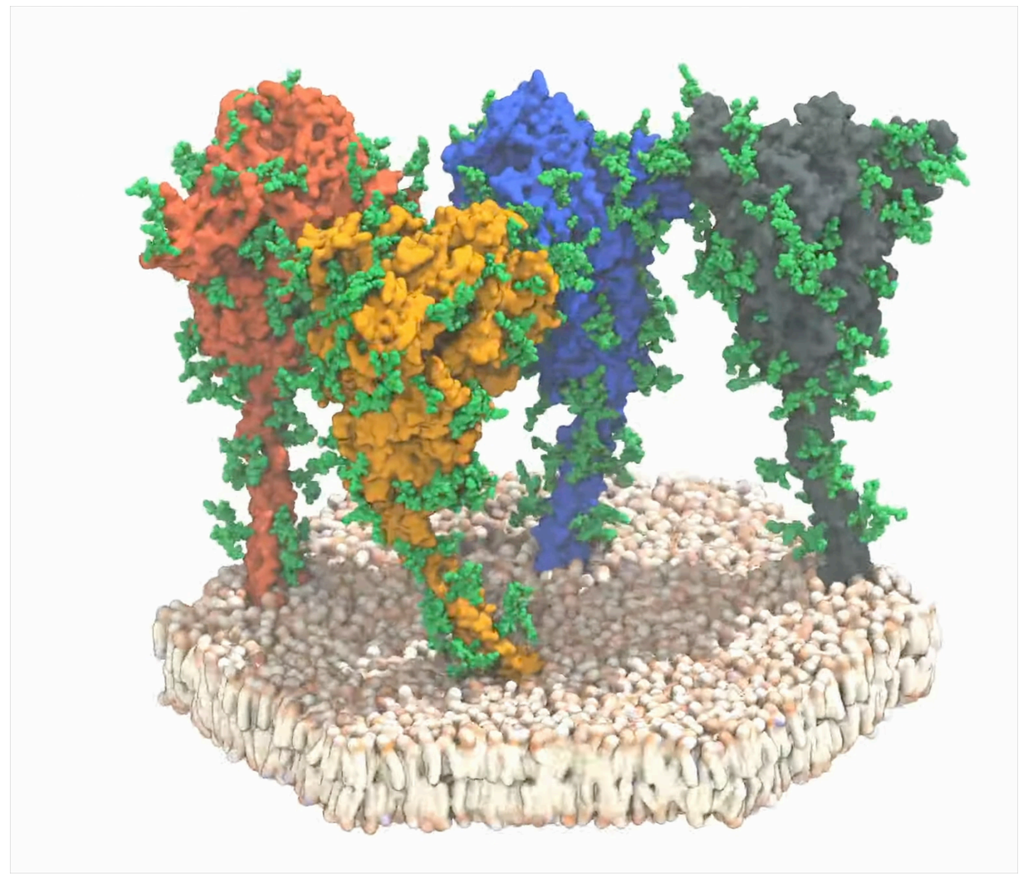


- Molecular dynamics: the goal is to simulate molecule trajectories

Stochastic Different equations (SDEs)

Differential equations describe phenomena appearing throughout nature, technology & society

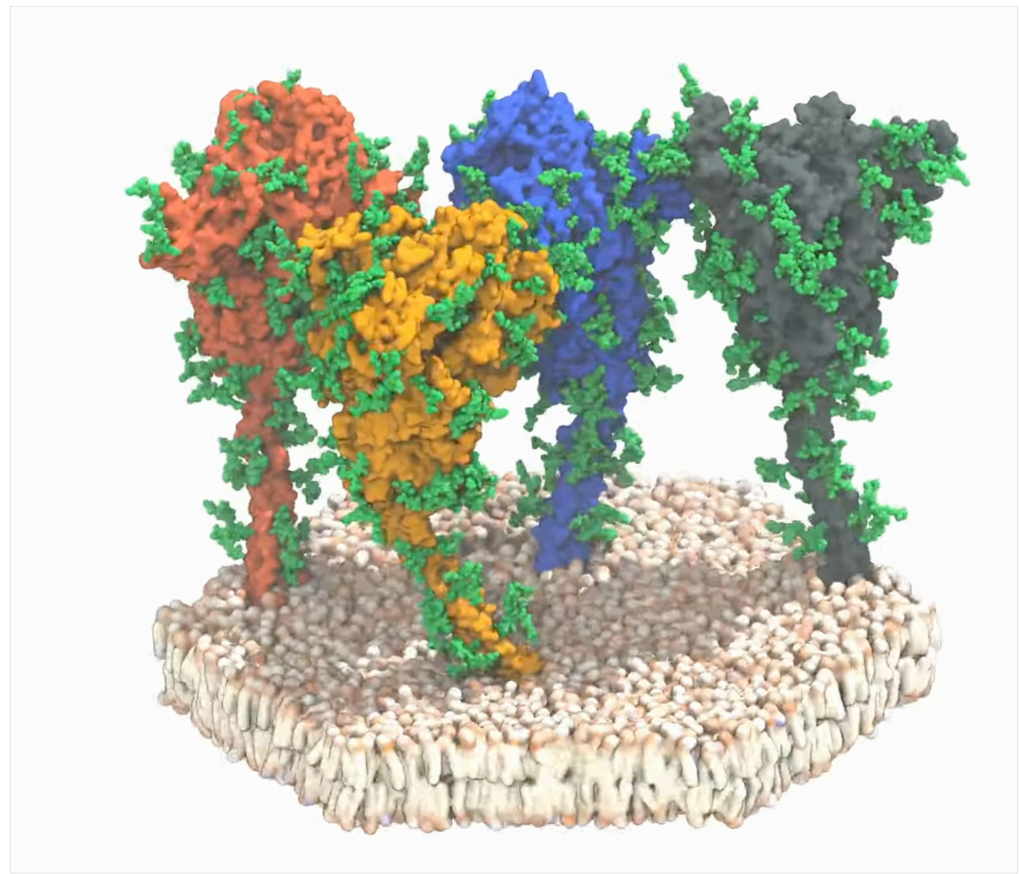
- Molecular dynamics: the goal is to simulate molecule trajectories
- Stock exchange



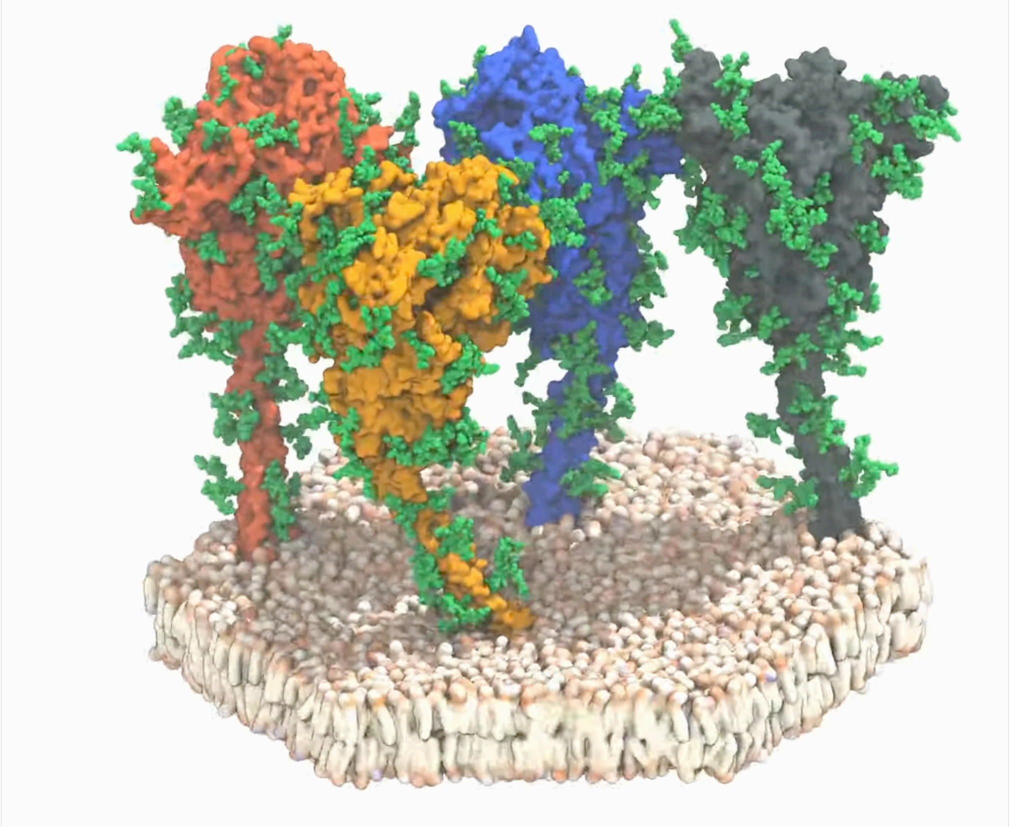
Stochastic Different equations (SDEs)

Differential equations describe phenomena appearing throughout nature, technology & society

- Molecular dynamics: the goal is to simulate molecule trajectories
- Stock exchange



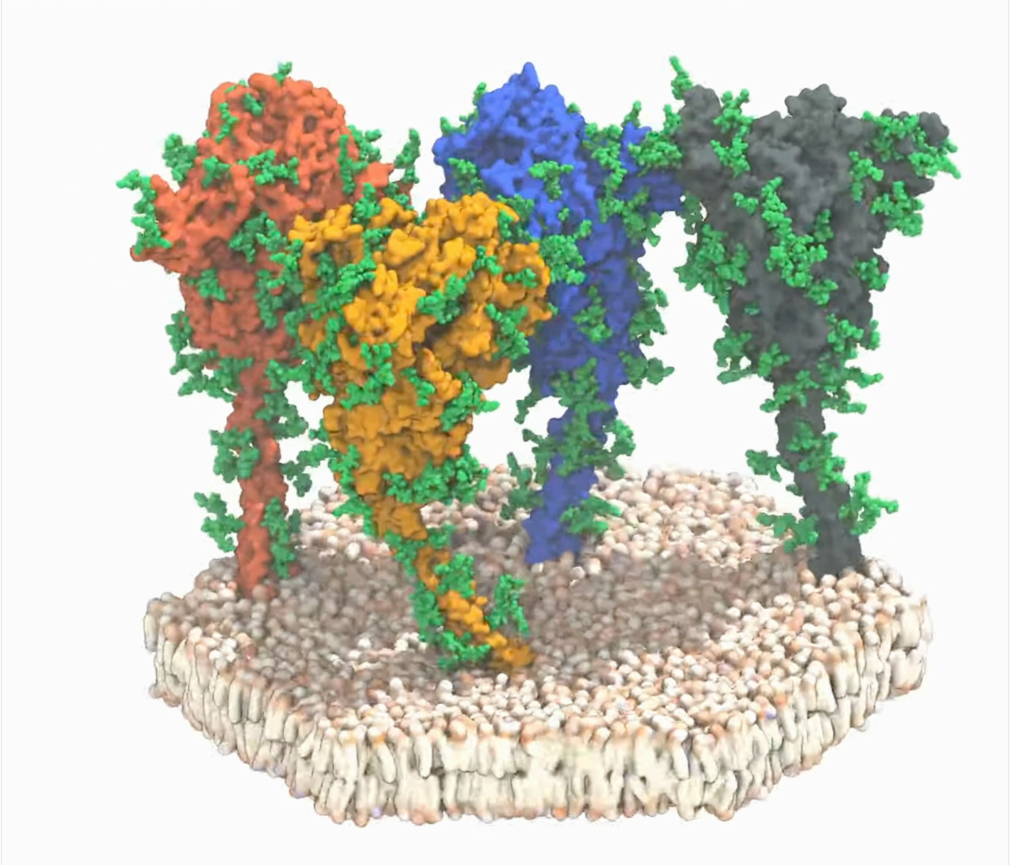
- Stock exchange



Stochastic Different equations (SDEs)

Differential equations describe phenomena appearing throughout nature, technology & society

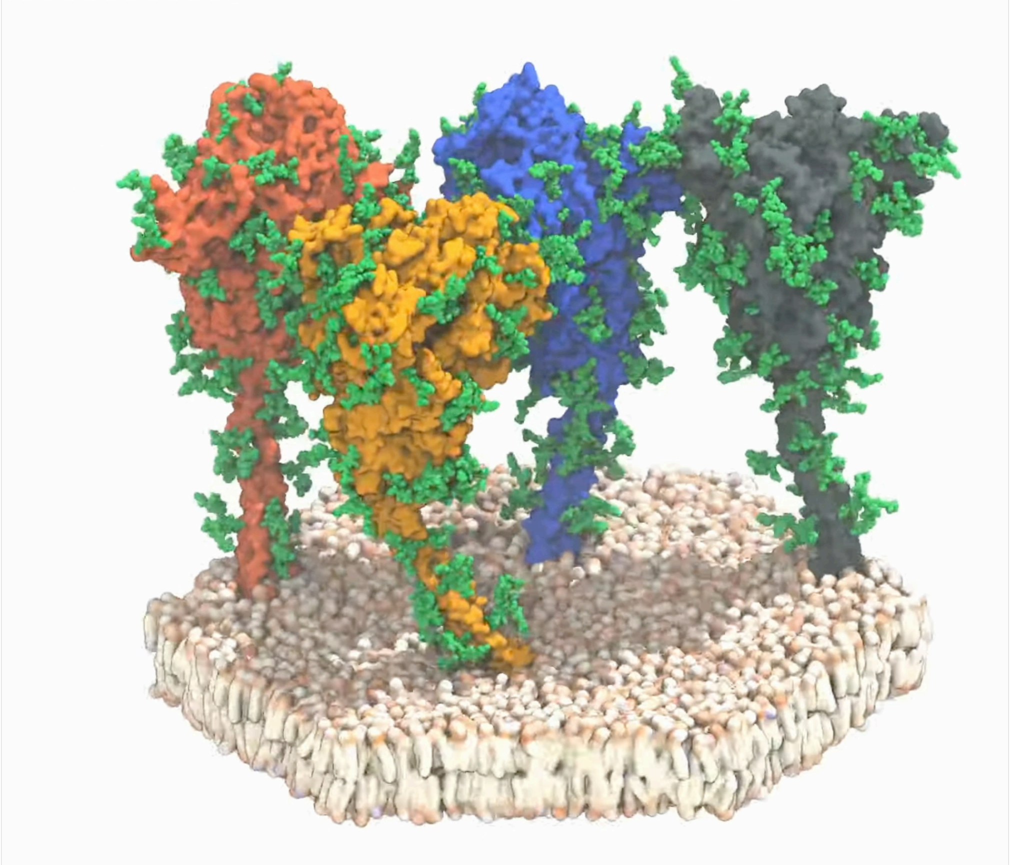
- Molecular dynamics: the goal is to simulate molecule trajectories
- Stock exchange
- Weather forecast models



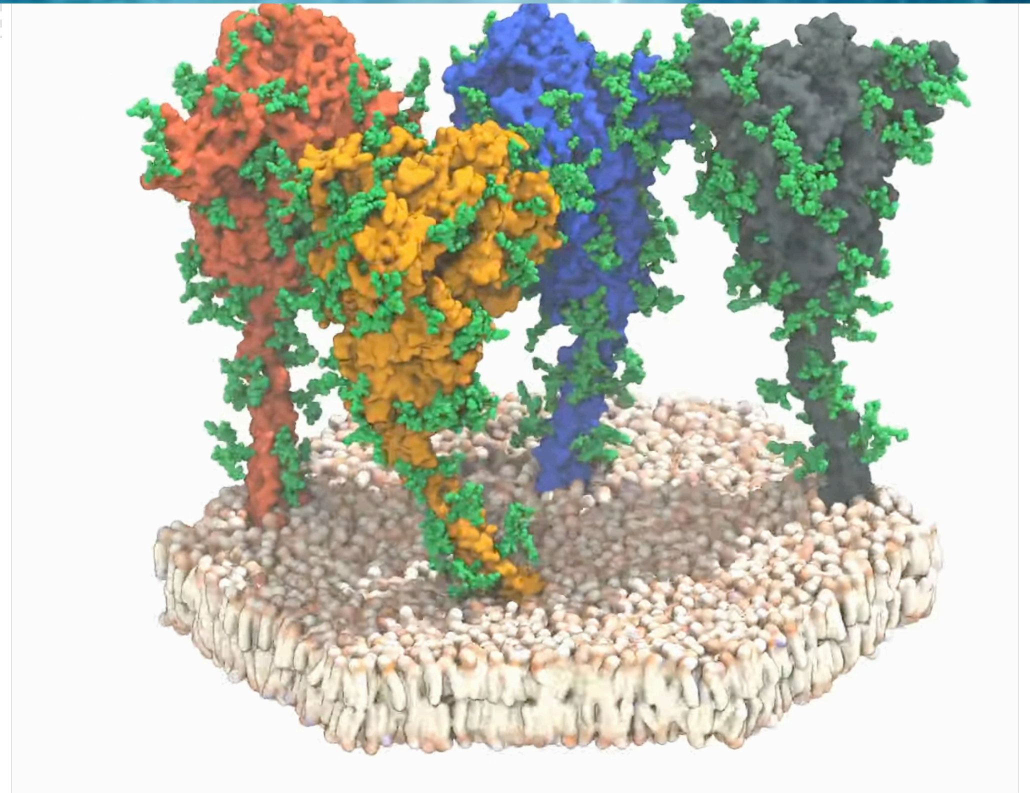
Stochastic Different equations (SDEs)

Differential equations describe phenomena appearing throughout nature, technology & society

- Molecular dynamics: the goal is to simulate molecule trajectories
- Stock exchange
- Weather forecast models



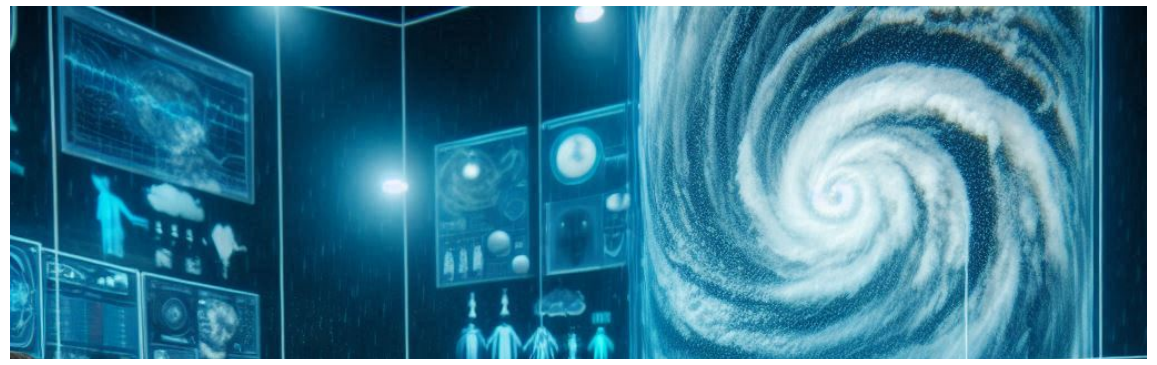
- Weather forecast models

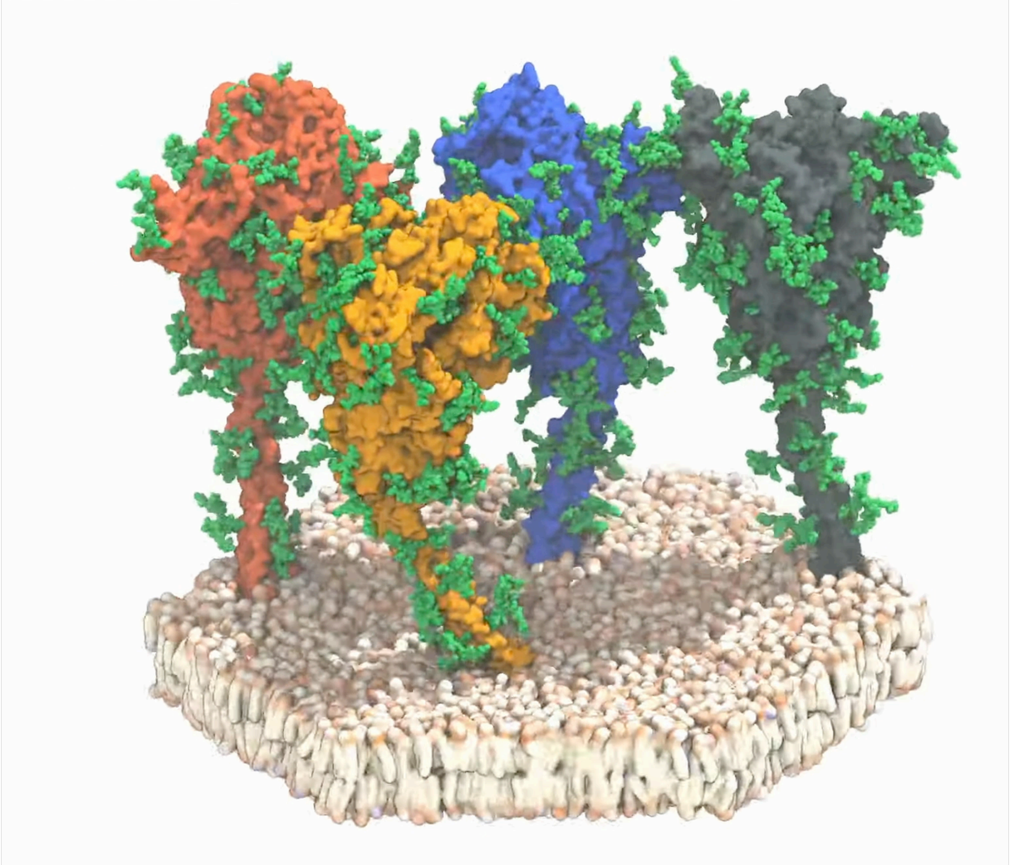


Stochastic Different equations (SDEs)

Differential equations describe phenomena appearing throughout nature, technology & society

- Molecular dynamics: the goal is to simulate molecule trajectories
- Stock exchange
- Weather forecast

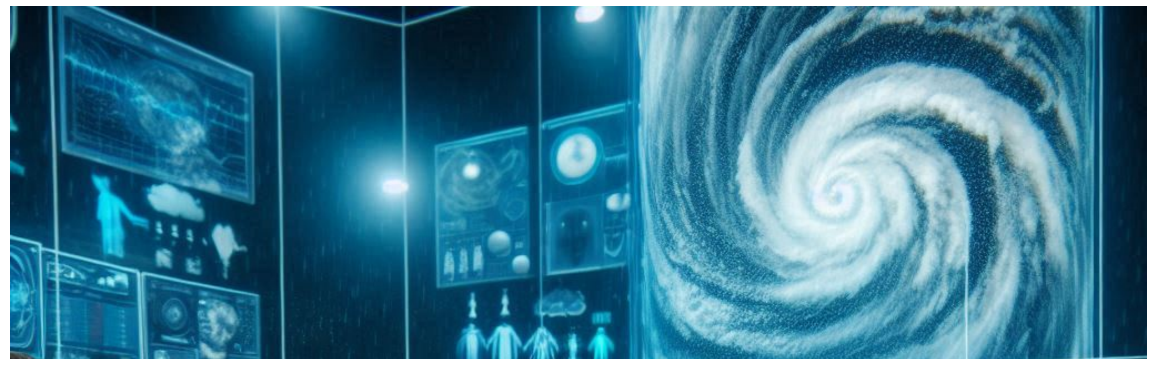


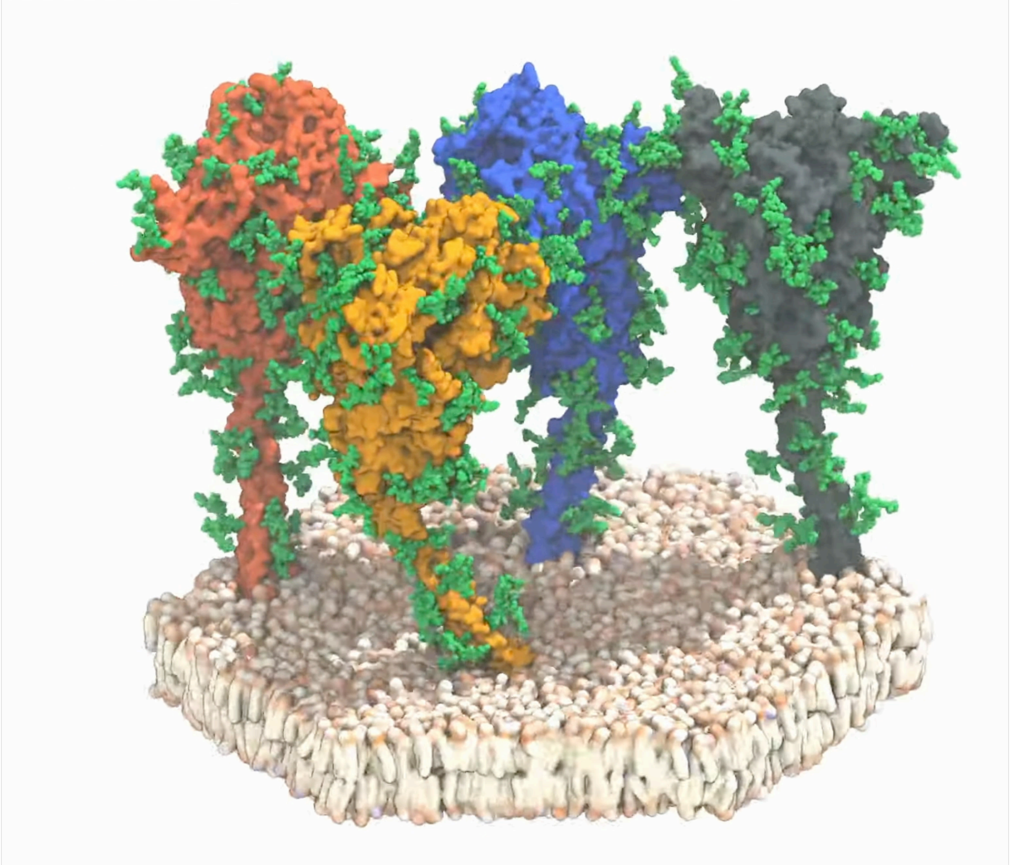


Stochastic Different equations (SDEs)

Differential equations describe phenomena appearing throughout nature, technology & society

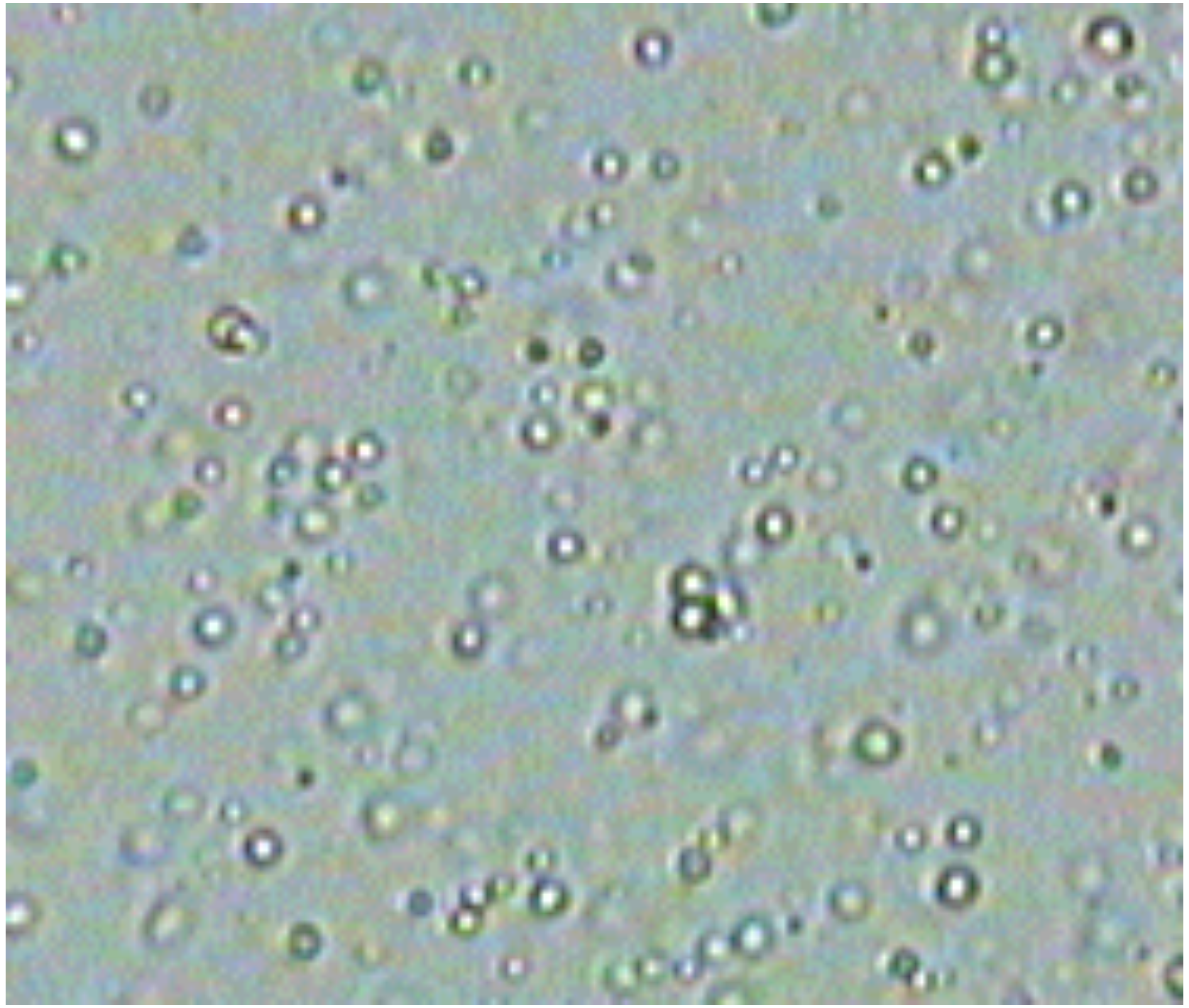
- Molecular dynamics: the goal is to simulate molecule trajectories
- Stock exchange
- Weather forecast





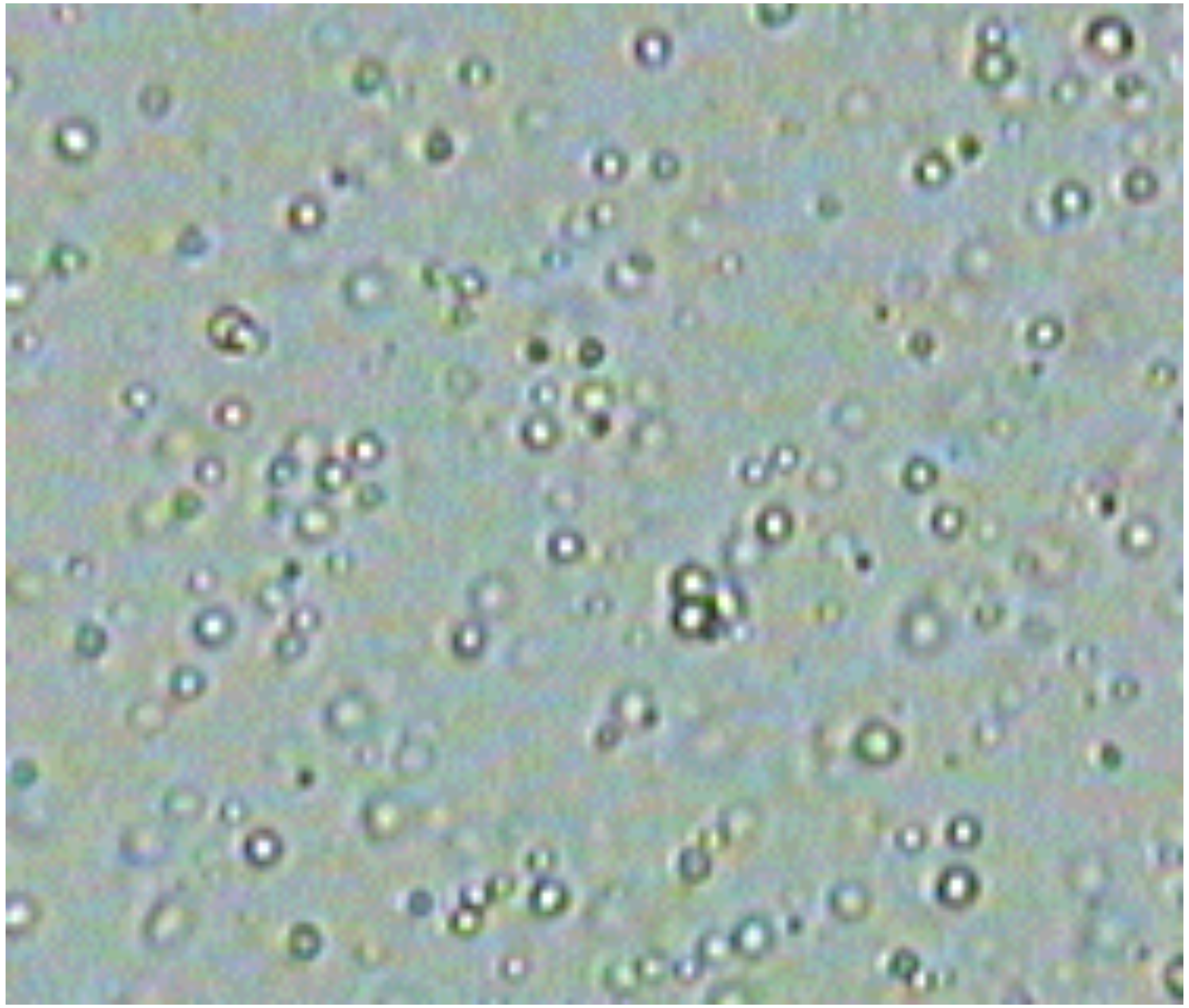
Stochastic Different equations (SDEs)

SDEs are powerful mathematical tools to formulate such motion from macroscopic to microscopic level



Stochastic Different equations (SDEs)

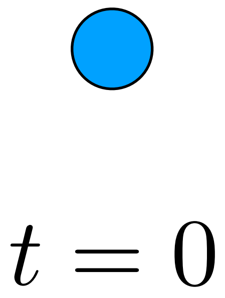
SDEs are powerful mathematical tools to formulate such motion from macroscopic to microscopic level



Markov chain Monte Carlo

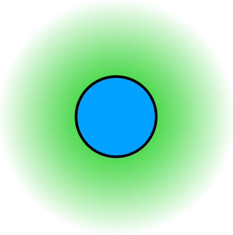
Markov chain Monte Carlo

Markov chain Monte Carlo



A Markov chain is a sequence of events, where the future event/state only depends on the current state.

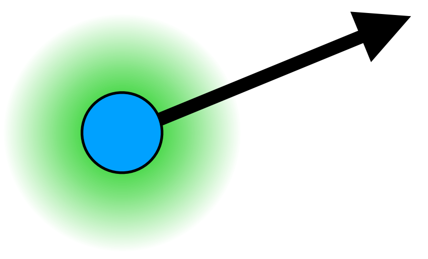
Markov chain Monte Carlo



$t = 0$

A Markov chain is a sequence of events, where the future event/state only depends on the current state.

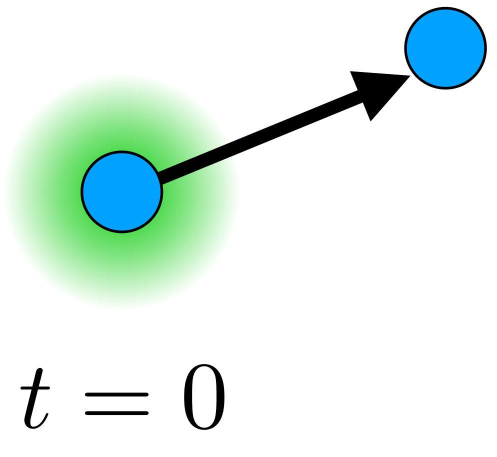
Markov chain Monte Carlo



$t = 0$

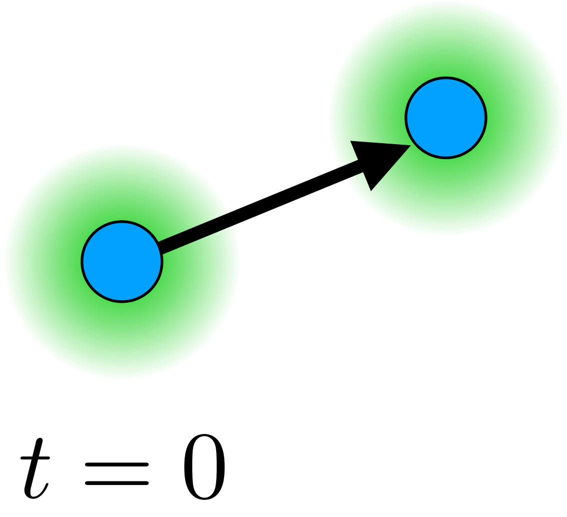
A Markov chain is a sequence of events, where the future event/state only depends on the current state.

Markov chain Monte Carlo



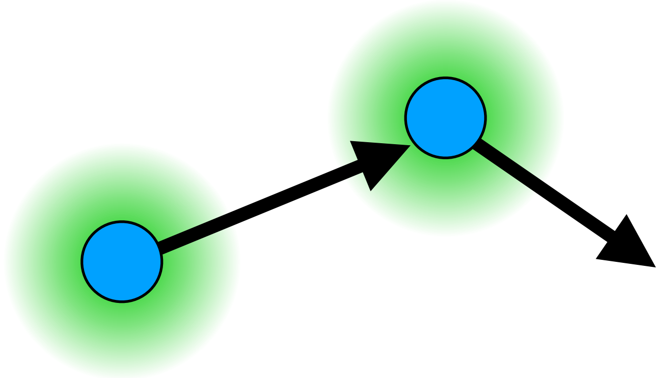
A Markov chain is a sequence of events, where the future event/state only depends on the current state.

Markov chain Monte Carlo



A Markov chain is a sequence of events, where the future event/state only depends on the current state.

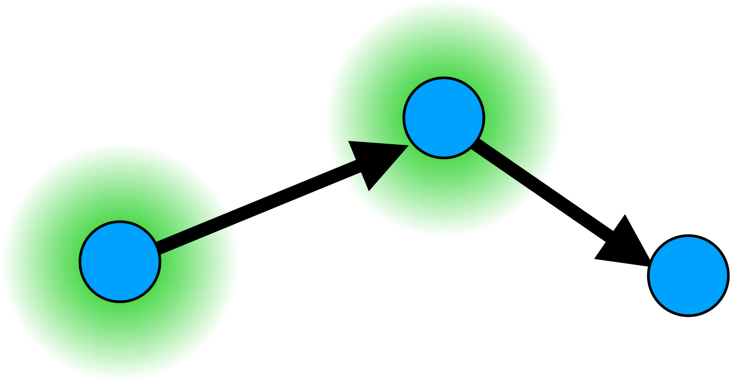
Markov chain Monte Carlo



$t = 0$

A Markov chain is a sequence of events, where the future event/state only depends on the current state.

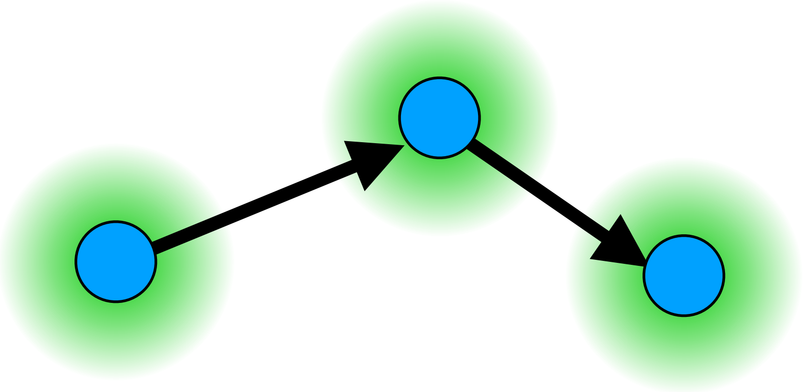
Markov chain Monte Carlo



$t = 0$

A Markov chain is a sequence of events, where the future event/state only depends on the current state.

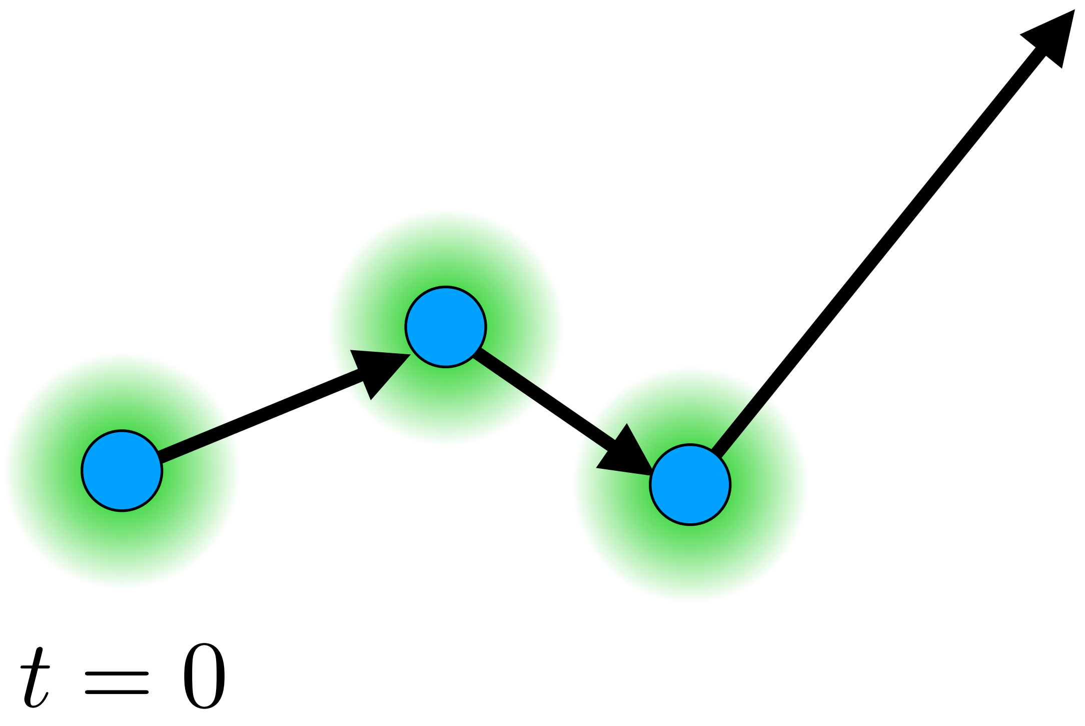
Markov chain Monte Carlo



$t = 0$

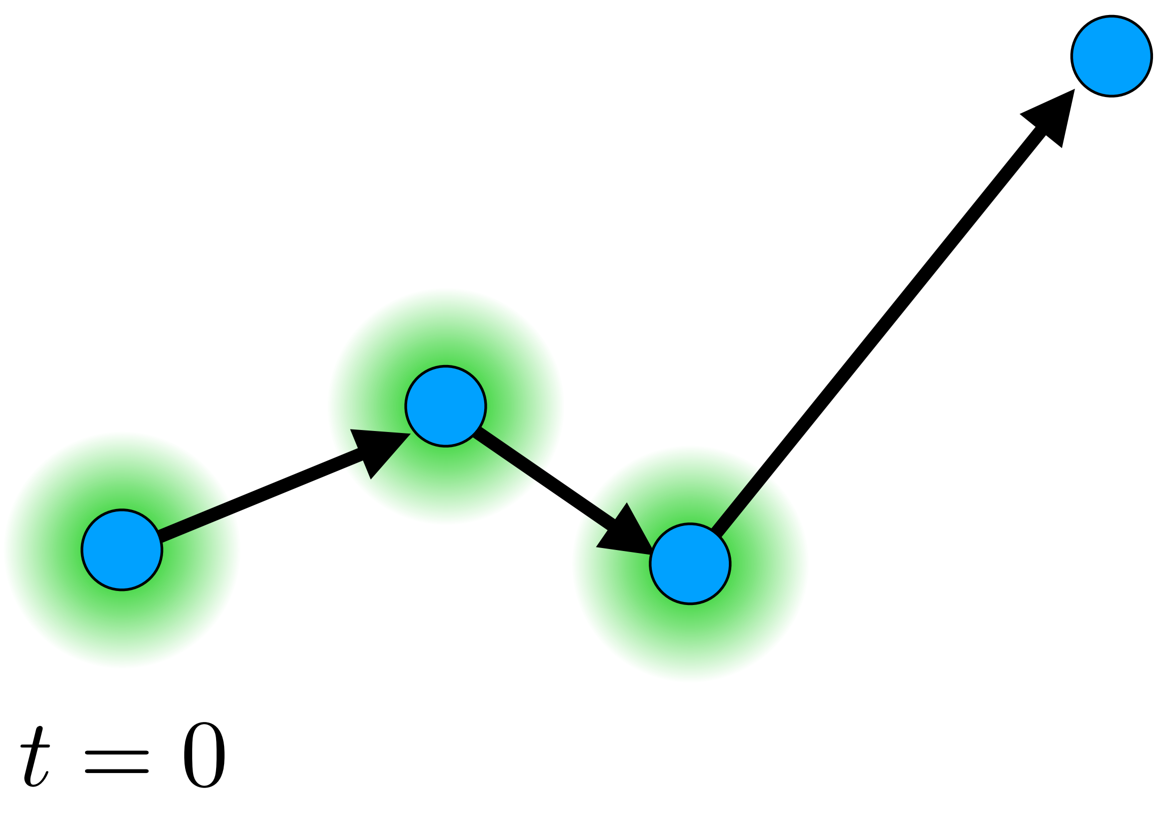
A Markov chain is a sequence of events, where the future event/state only depends on the current state.

Markov chain Monte Carlo



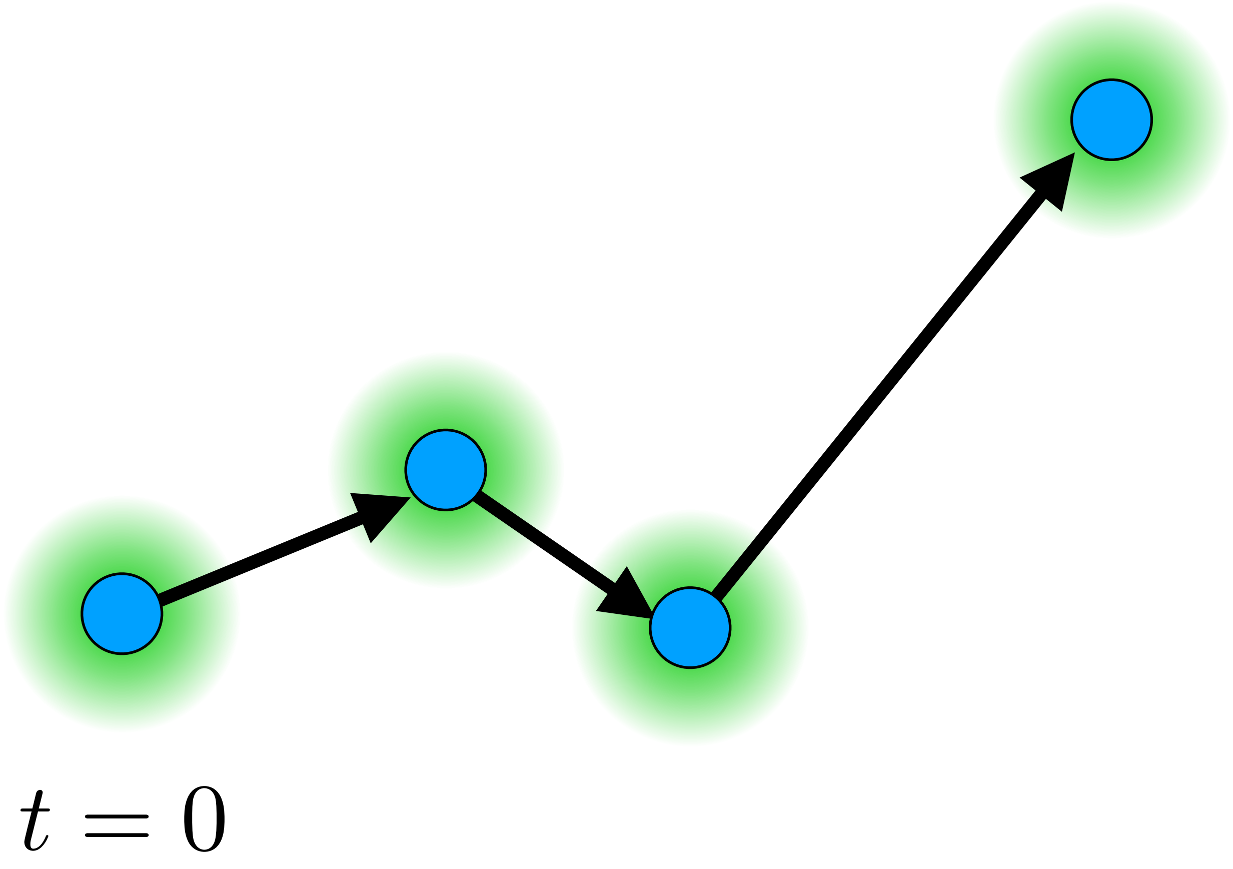
A Markov chain is a sequence of events, where the future event/state only depends on the current state.

Markov chain Monte Carlo



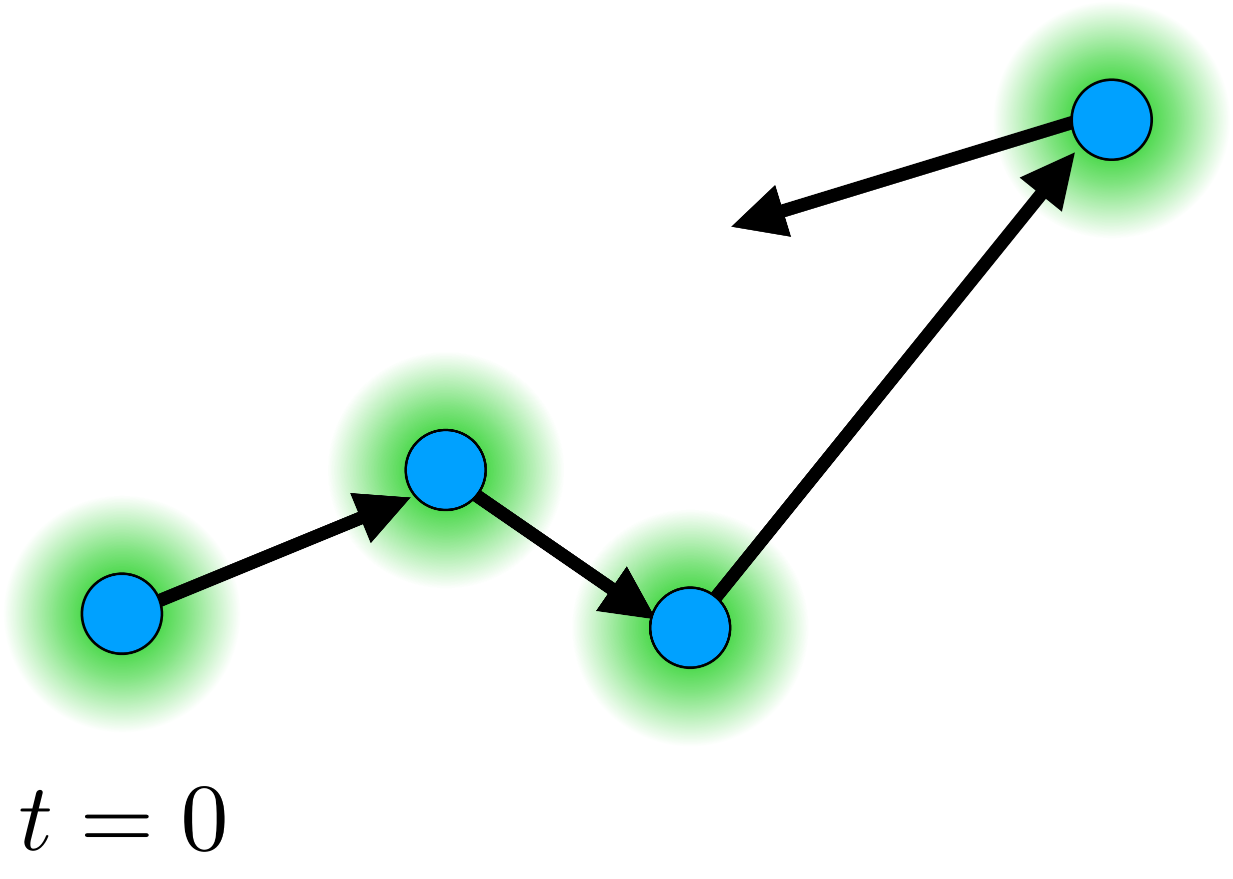
A Markov chain is a sequence of events, where the future event/state only depends on the current state.

Markov chain Monte Carlo



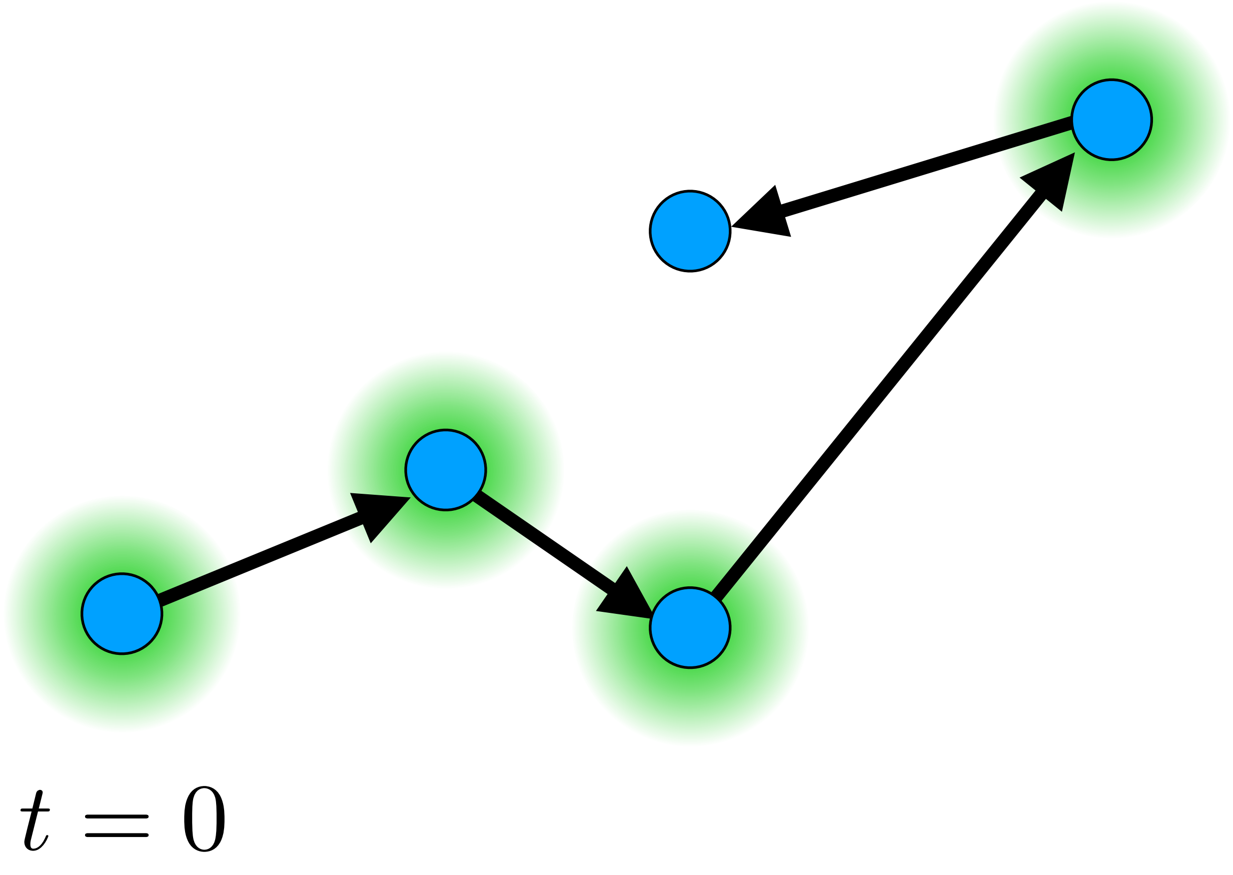
A Markov chain is a sequence of events, where the future event/state only depends on the current state.

Markov chain Monte Carlo



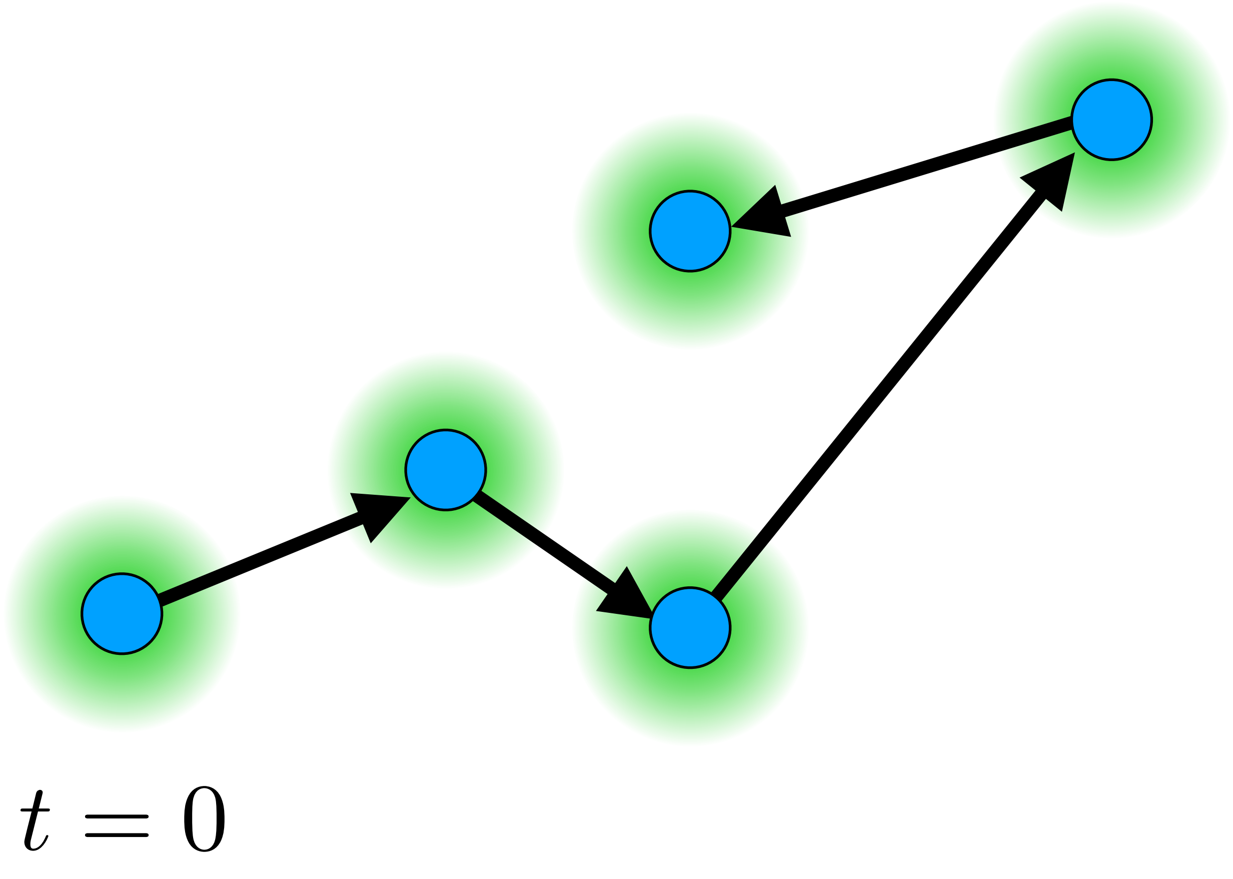
A Markov chain is a sequence of events, where the future event/state only depends on the current state.

Markov chain Monte Carlo



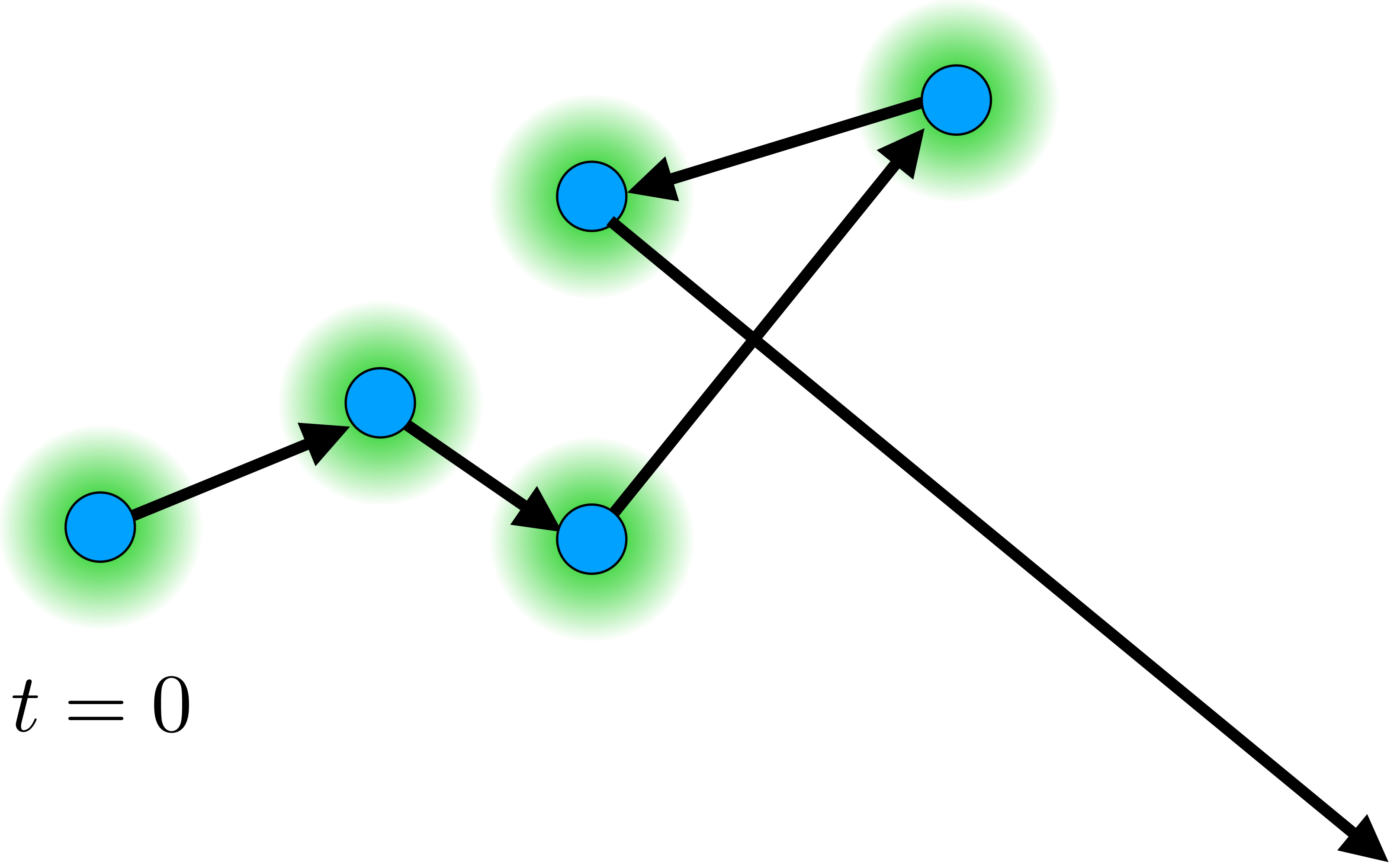
A Markov chain is a sequence of events, where the future event/state only depends on the current state.

Markov chain Monte Carlo



A Markov chain is a sequence of events, where the future event/state only depends on the current state.

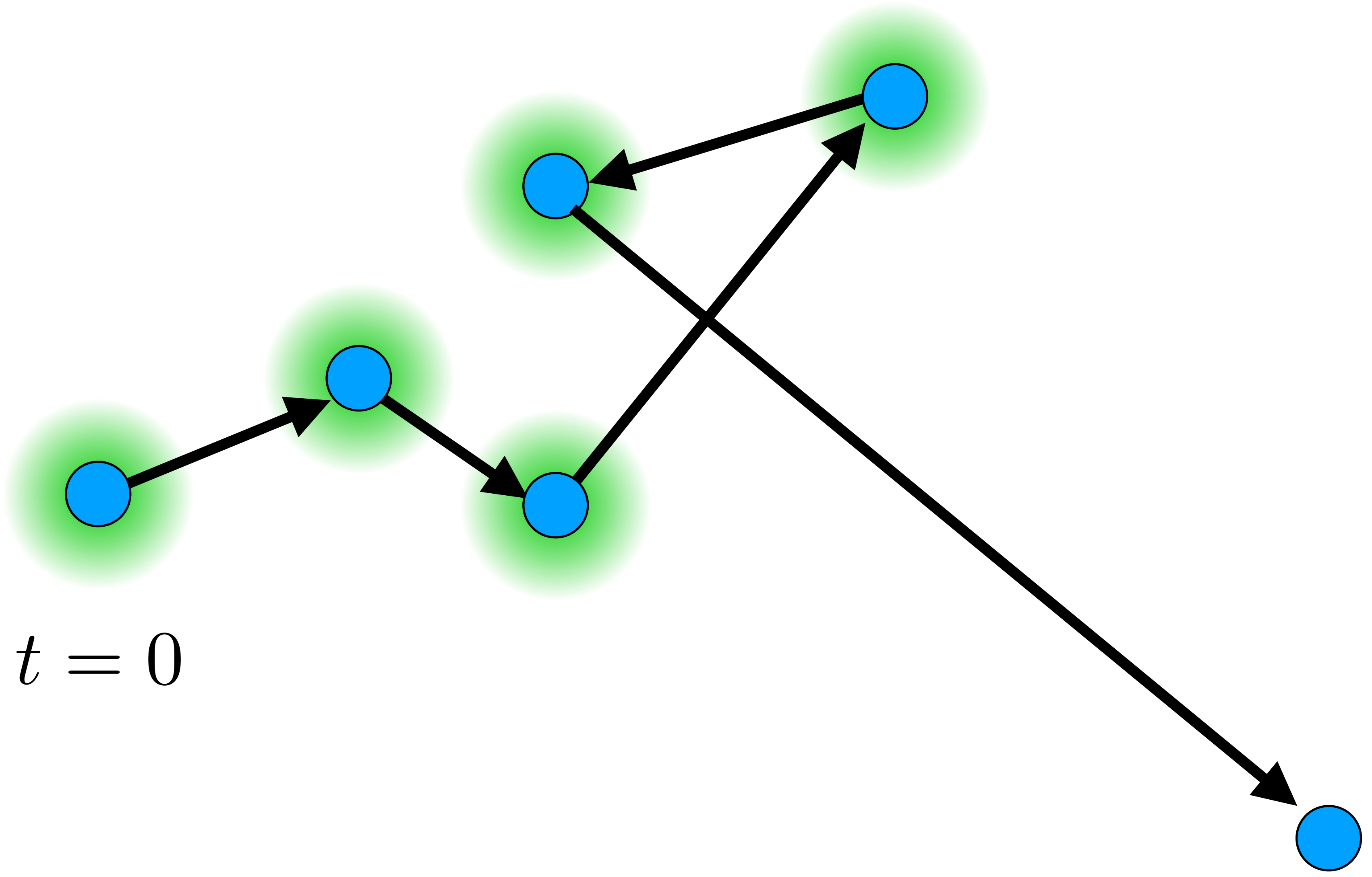
Markov chain Monte Carlo



$t = 0$

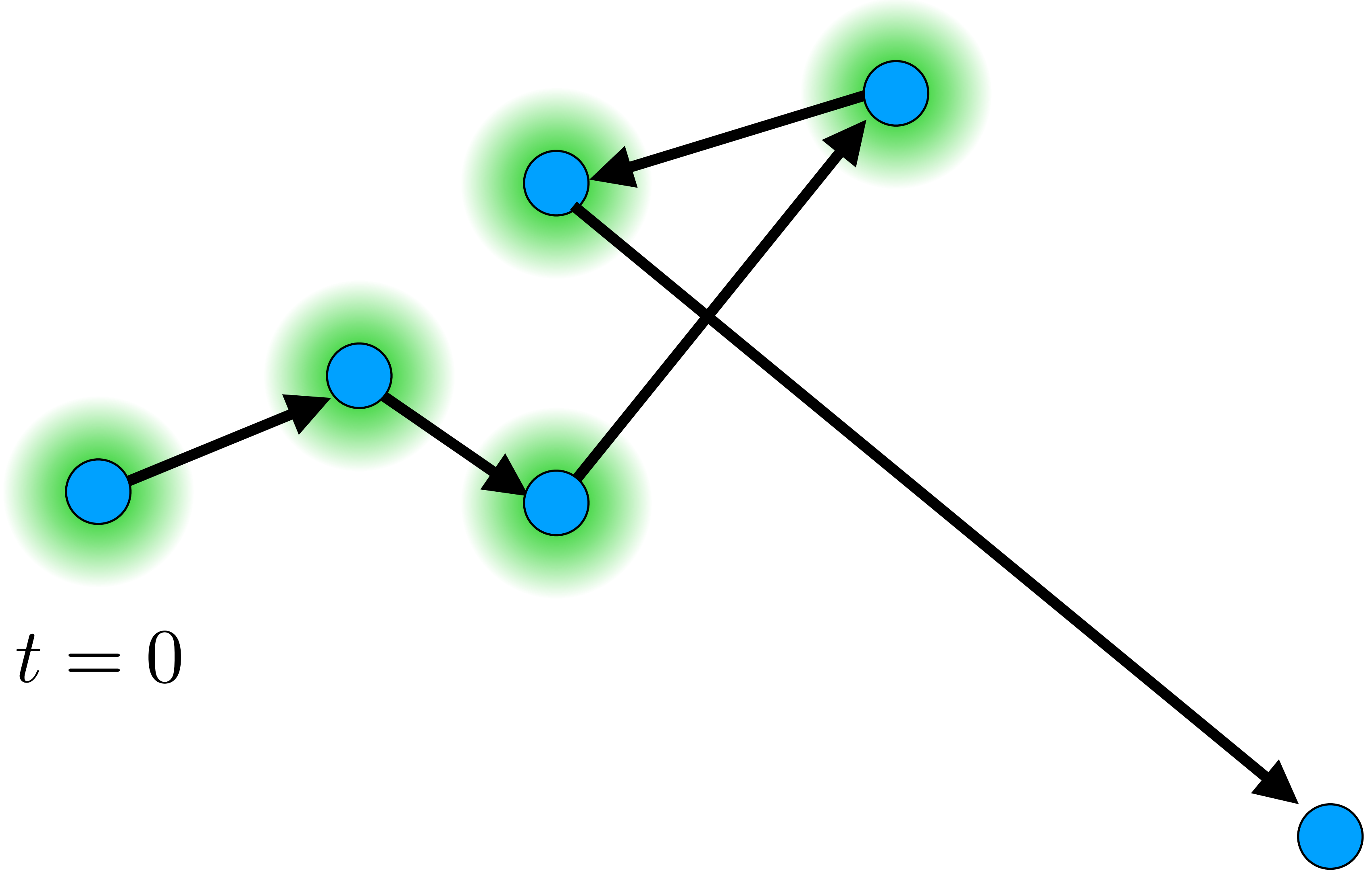
A Markov chain is a sequence of events, where the future event/state only depends on the current state.

Markov chain Monte Carlo

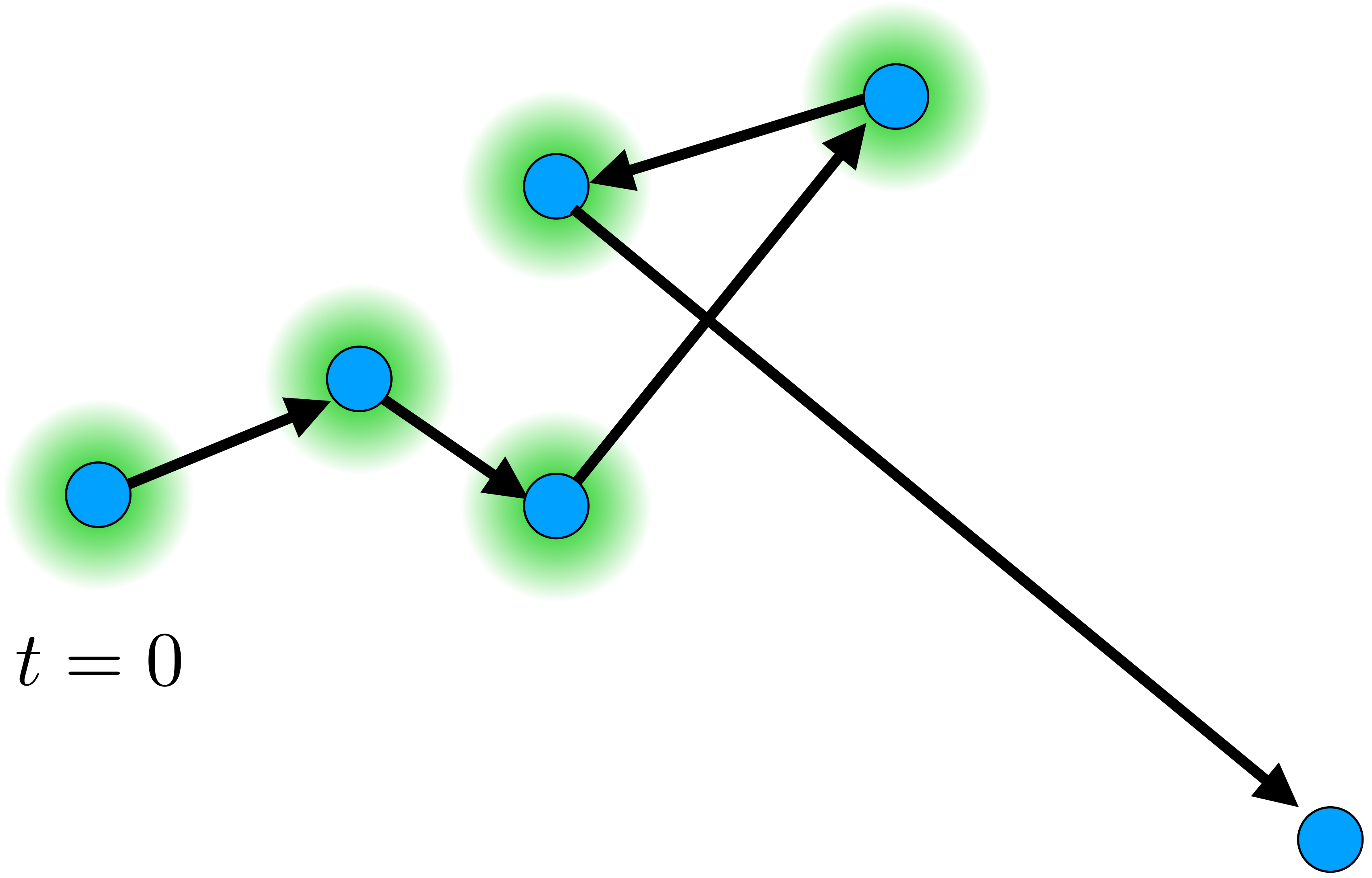


A Markov chain is a sequence of events, where the future event/state only depends on the current state.

Markov chain Monte Carlo

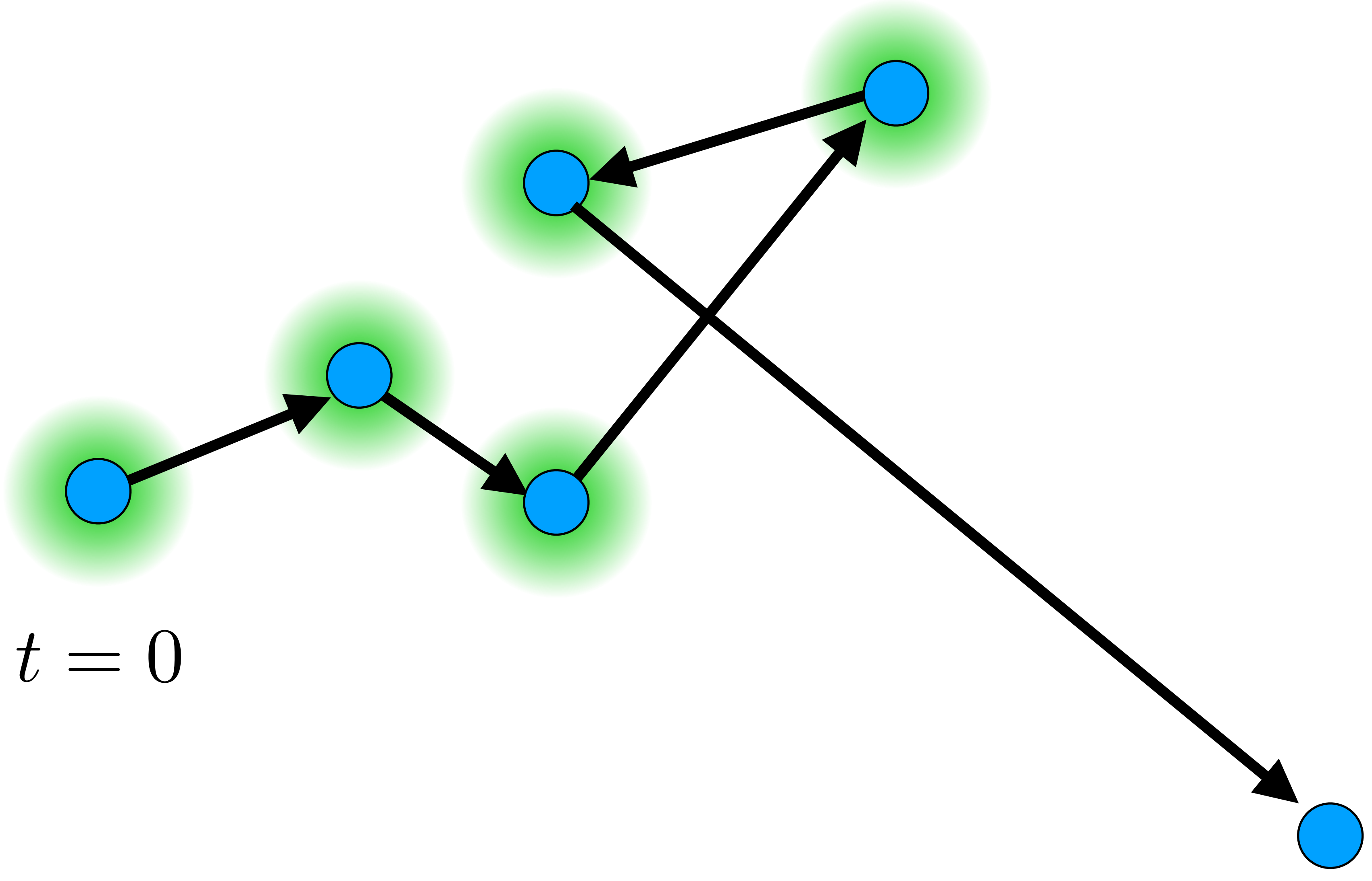


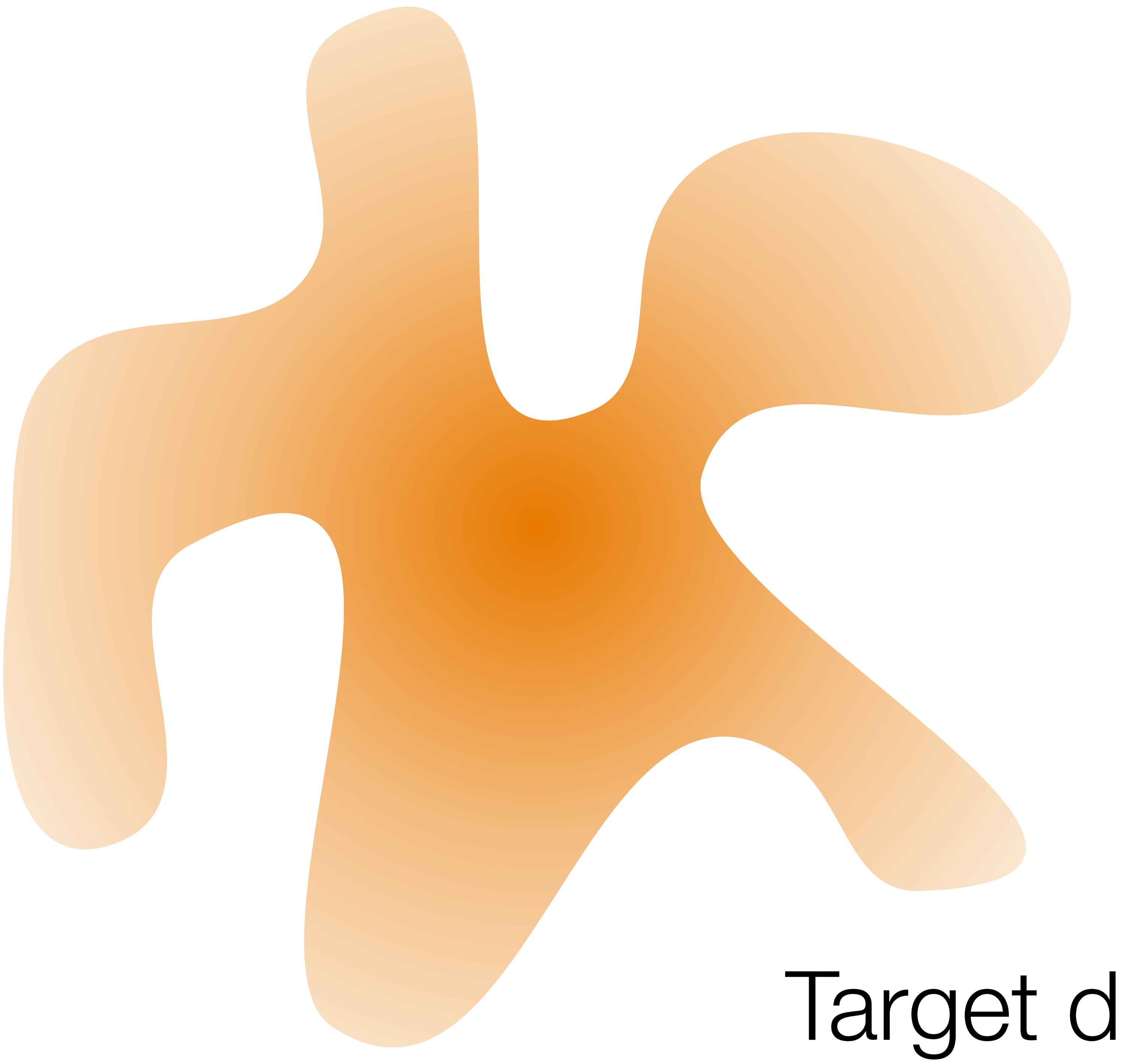
Markov chain Monte Carlo



An arbitrary Markov chain simply wanders in the space

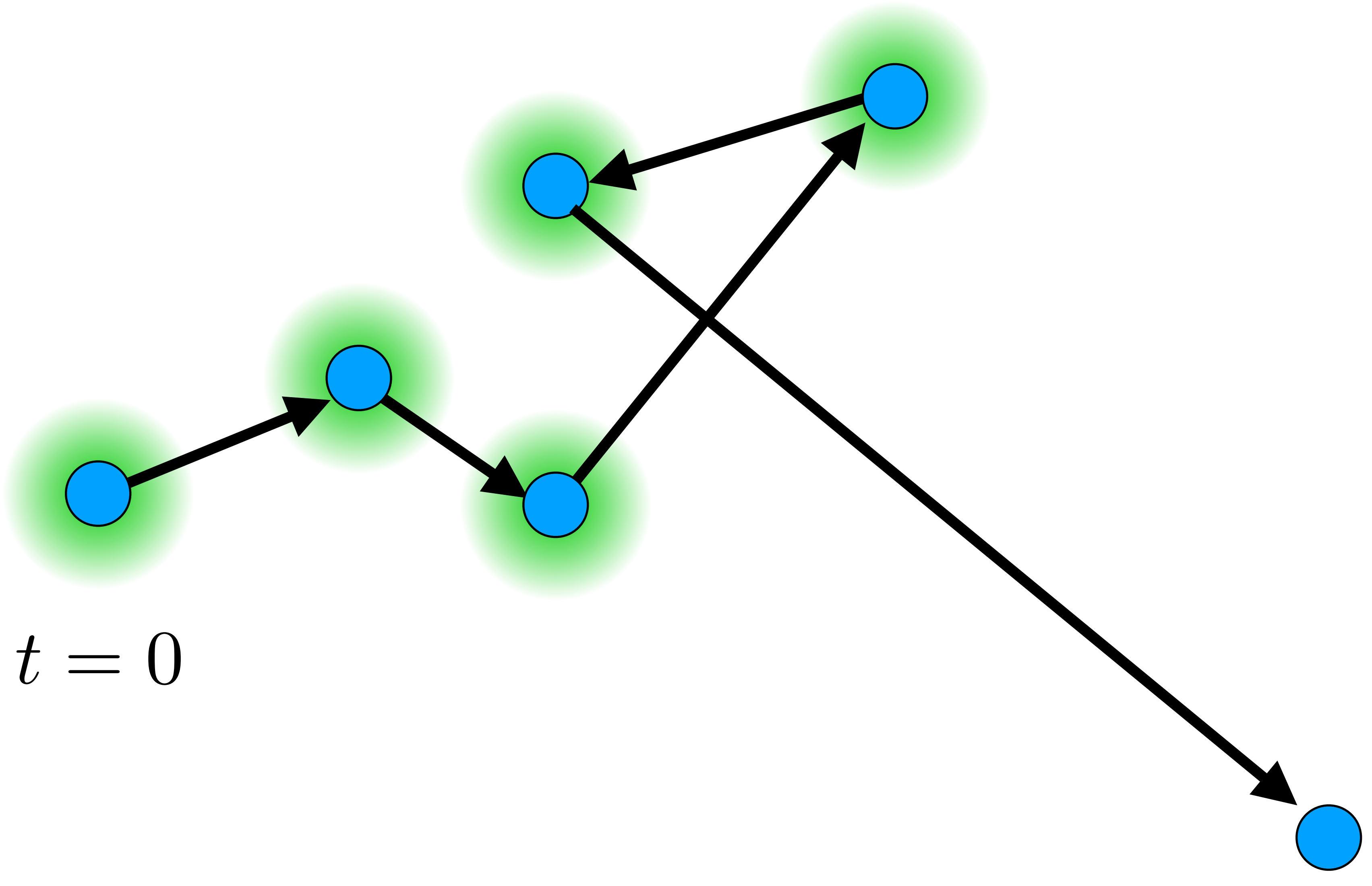
Markov chain Monte Carlo

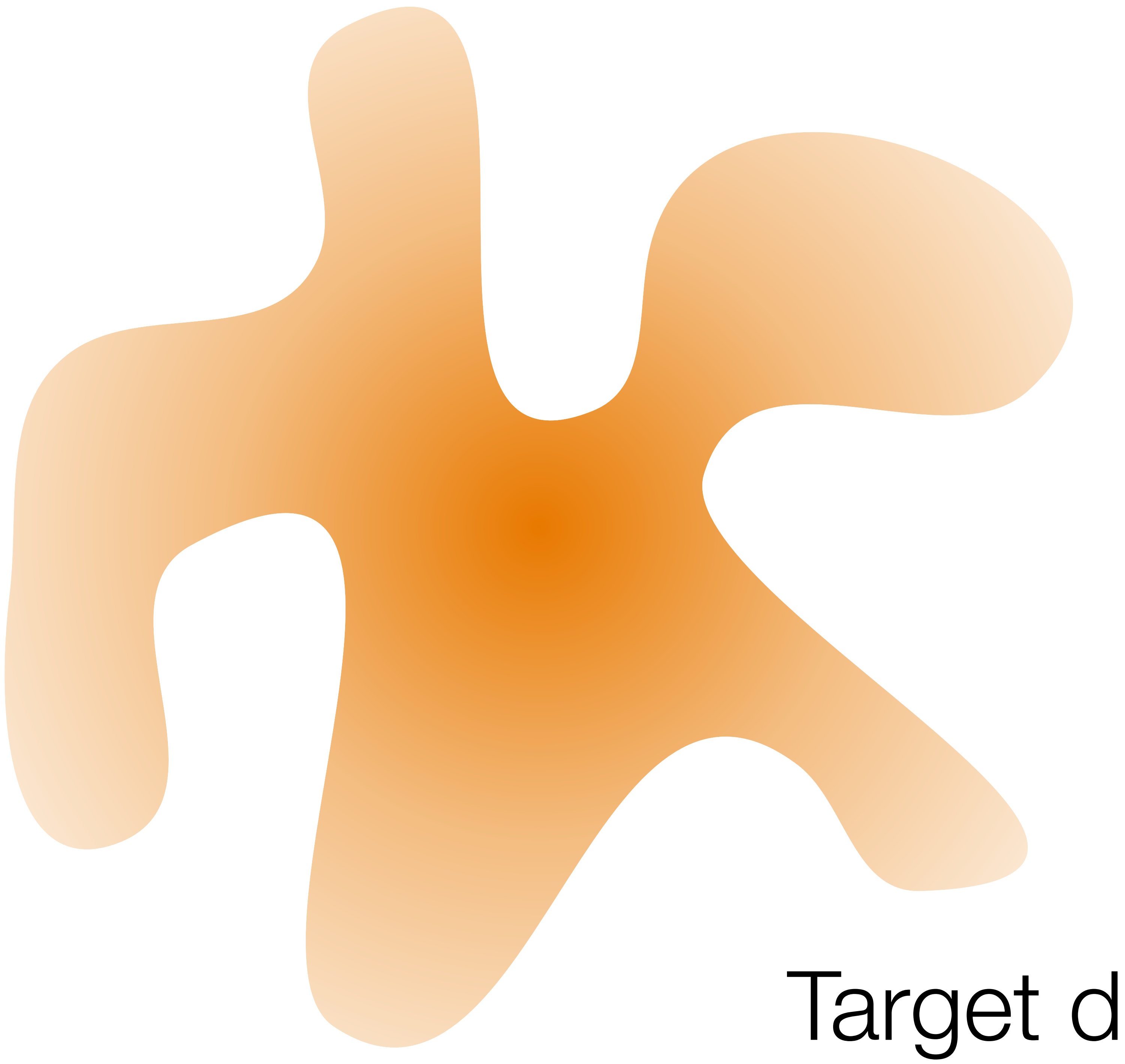




An arbitrary Markov chain simply wanders in the space

Markov chain Monte Carlo



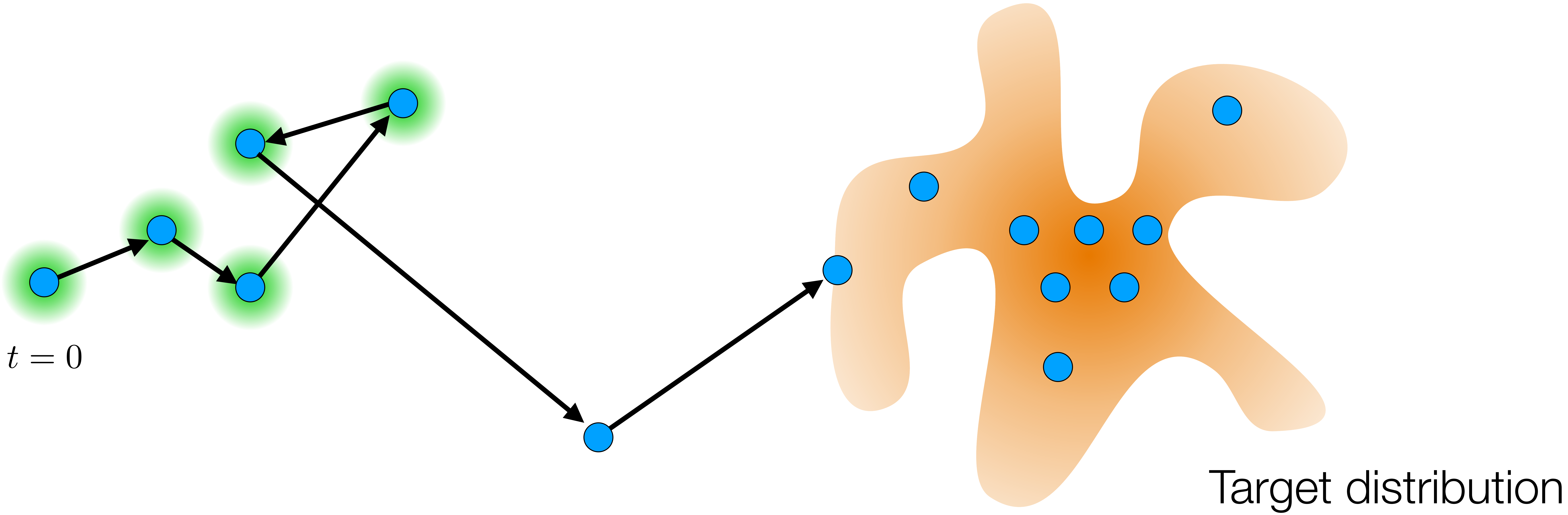


Target distribution

An arbitrary Markov chain simply wanders in the space

When run long enough, will preserve the underlying distribution (invariance property)

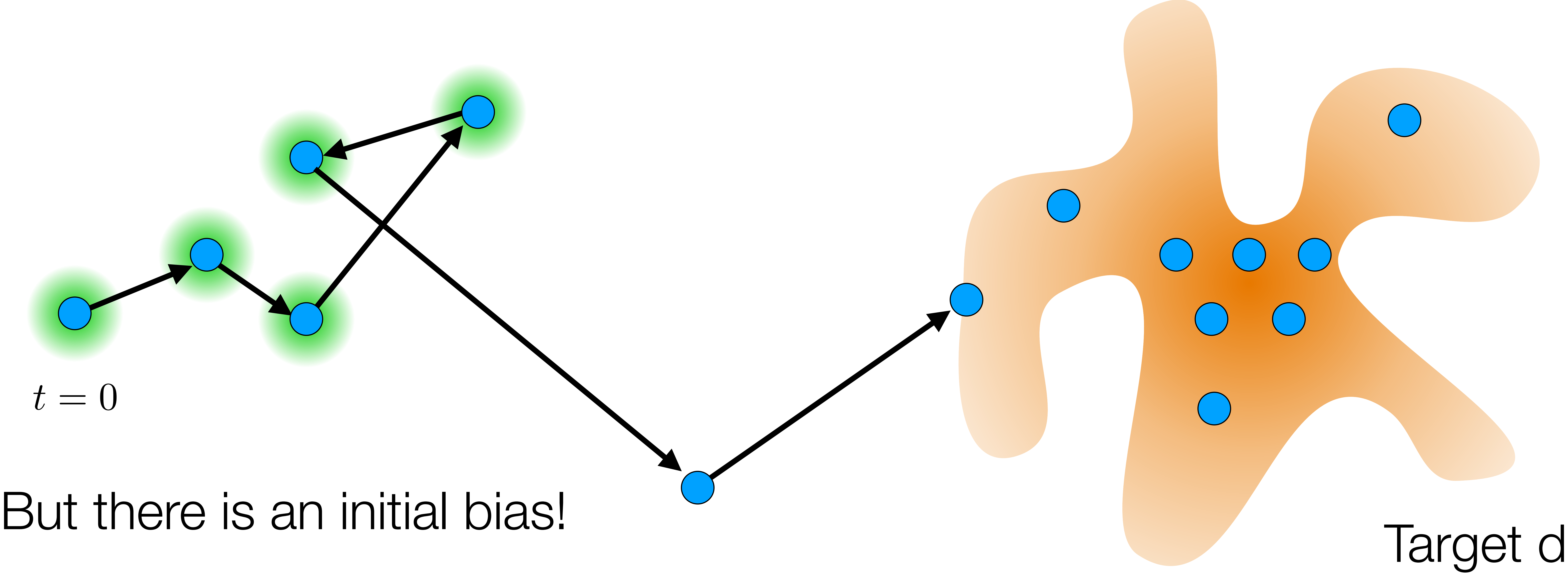
Markov chain Monte Carlo



An arbitrary Markov chain simply wanders in the space

When run long enough, will preserve the underlying distribution (invariance property)

Markov chain Monte Carlo

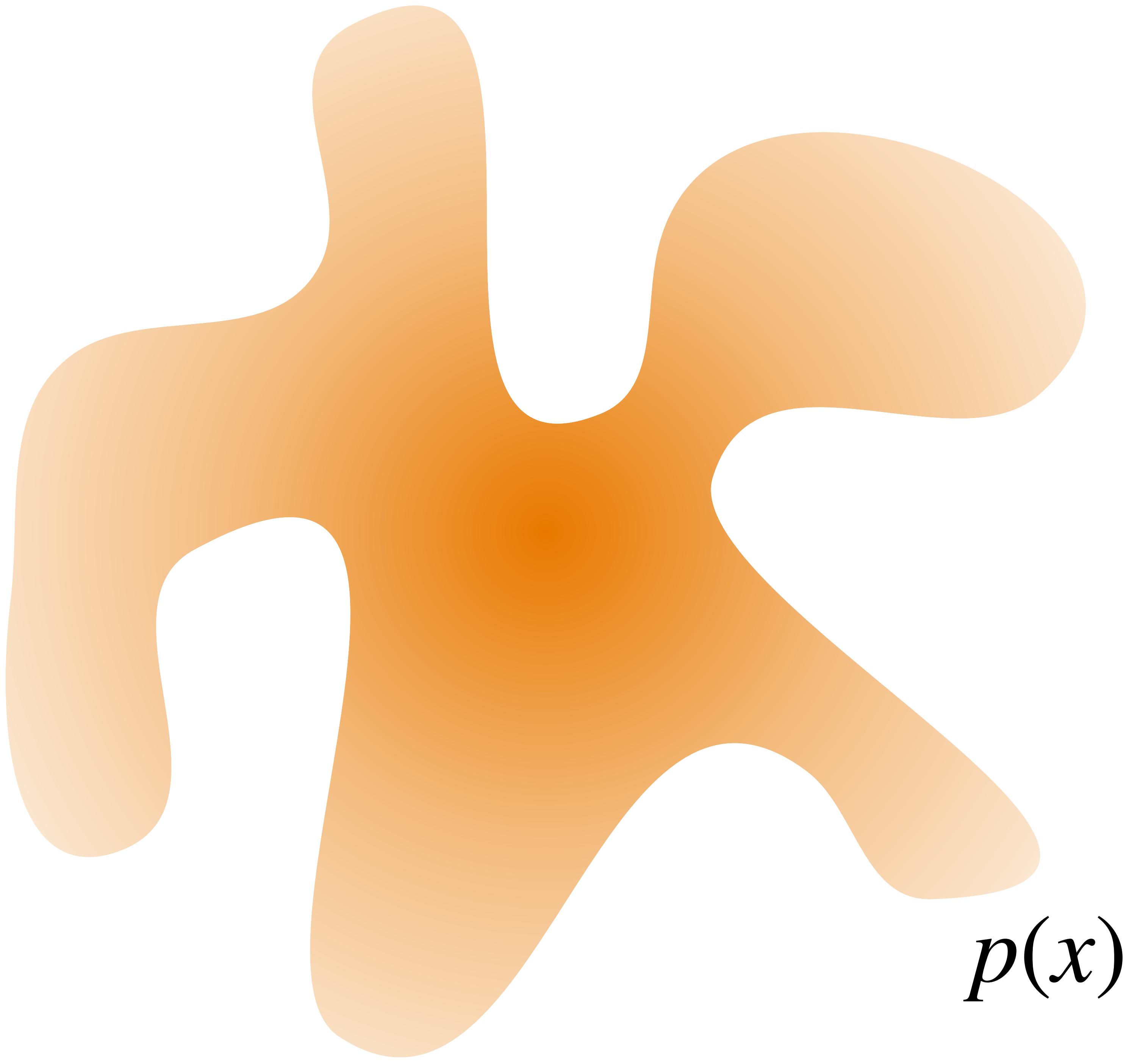


An arbitrary Markov chain simply wanders in the space

When run long enough, will preserve the underlying distribution (invariance property)

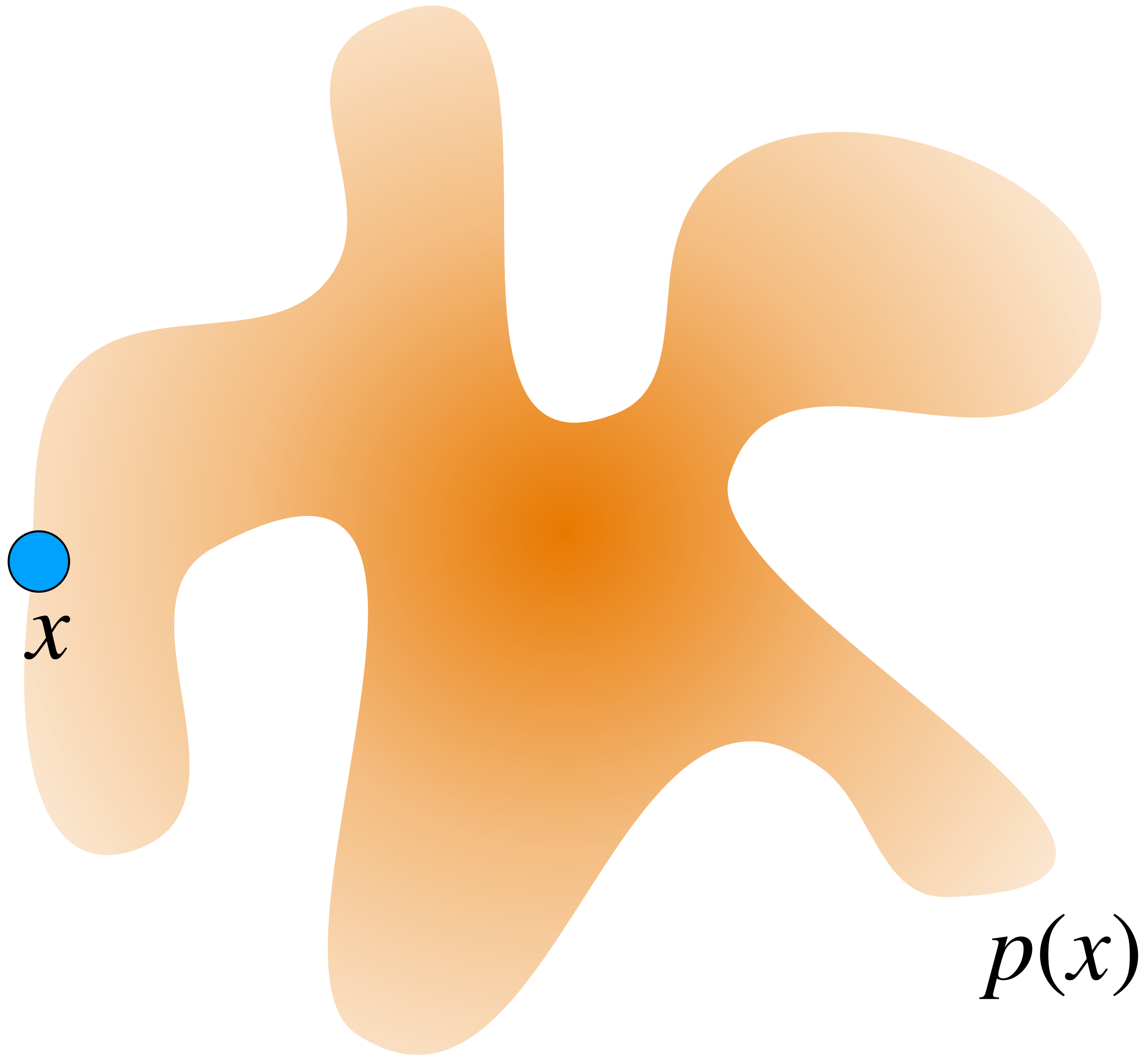
Metropolis-Hastings approach

Metropolis-Hastings approach



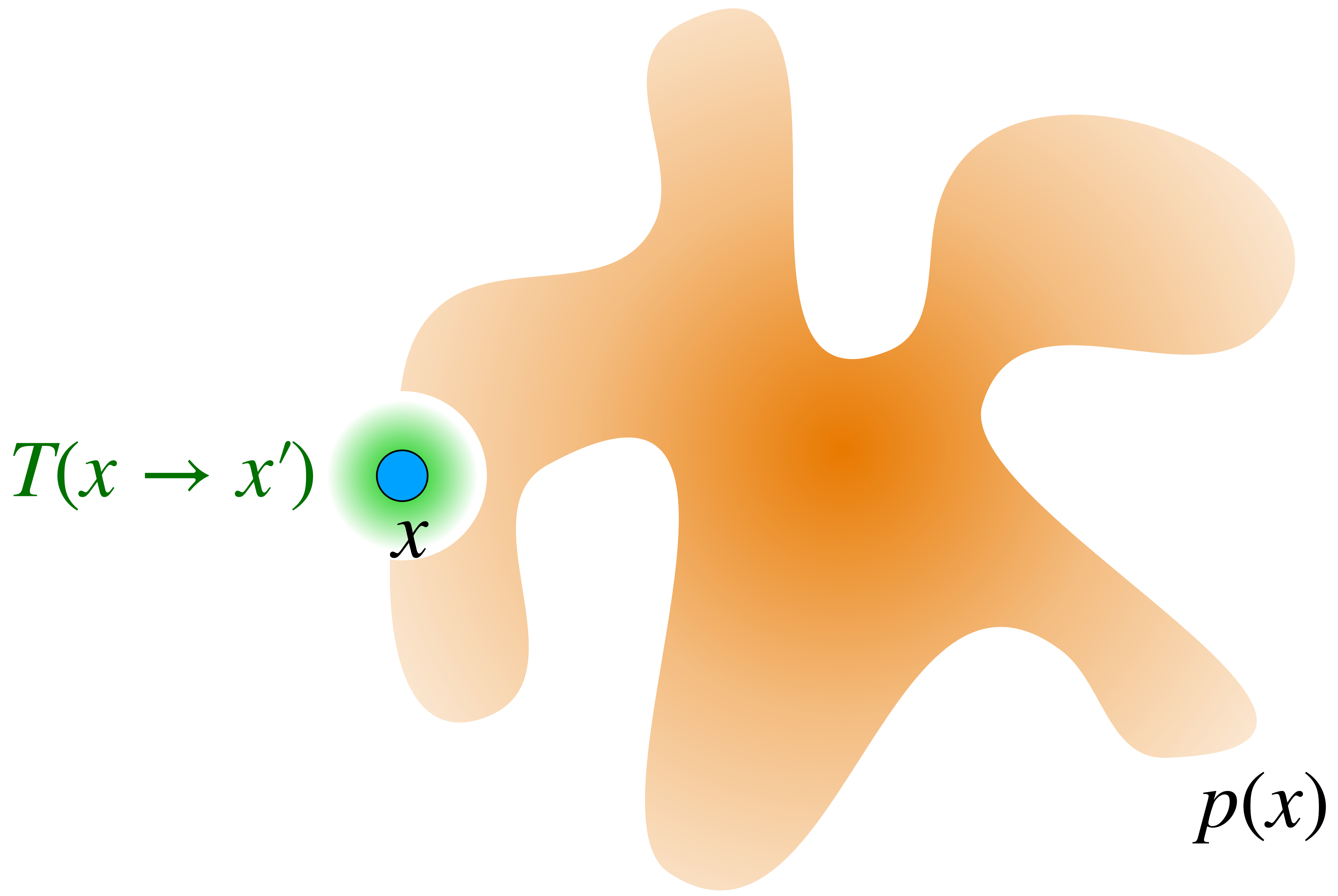
Target distribution

Metropolis-Hastings approach



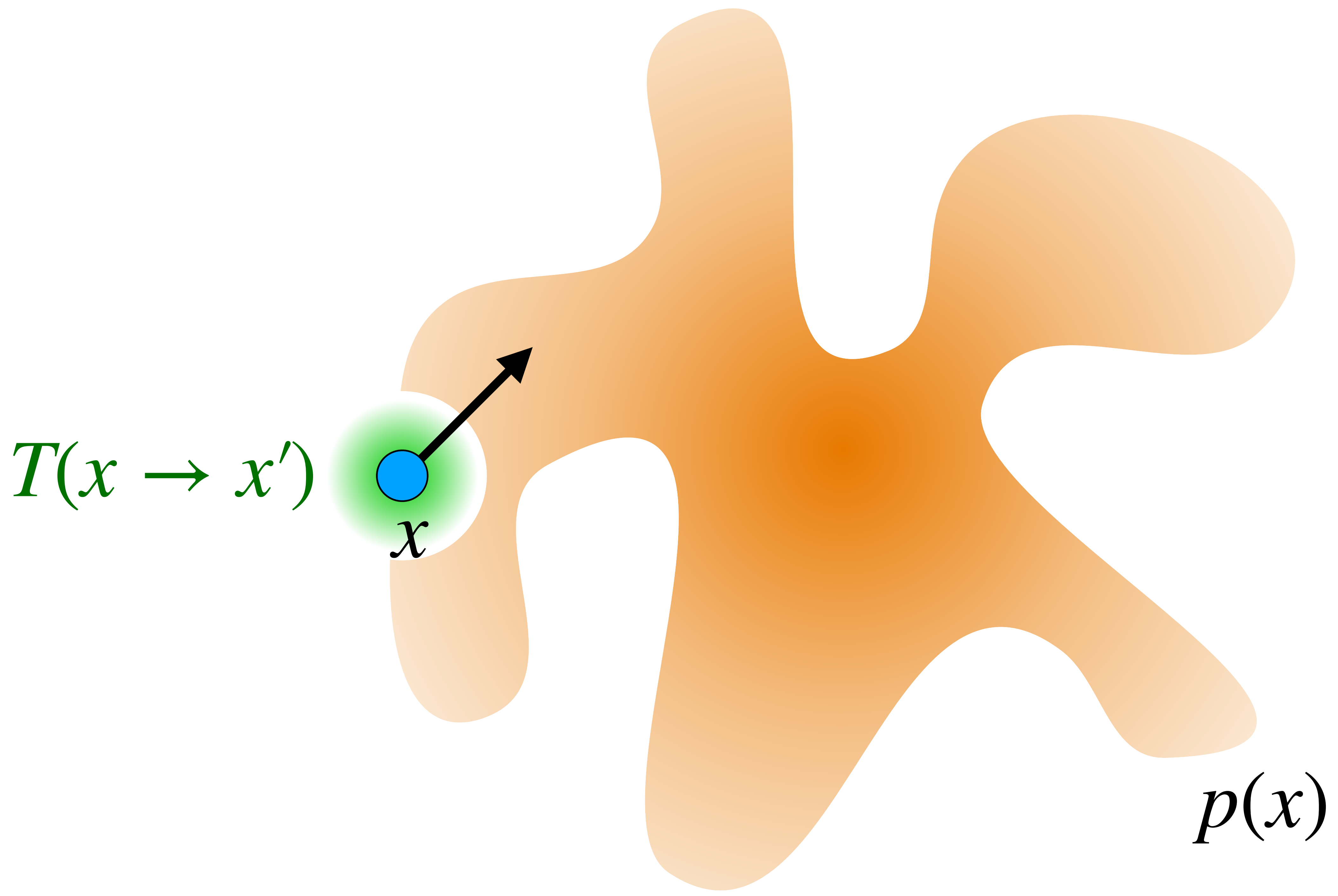
Target distribution

Metropolis-Hastings approach



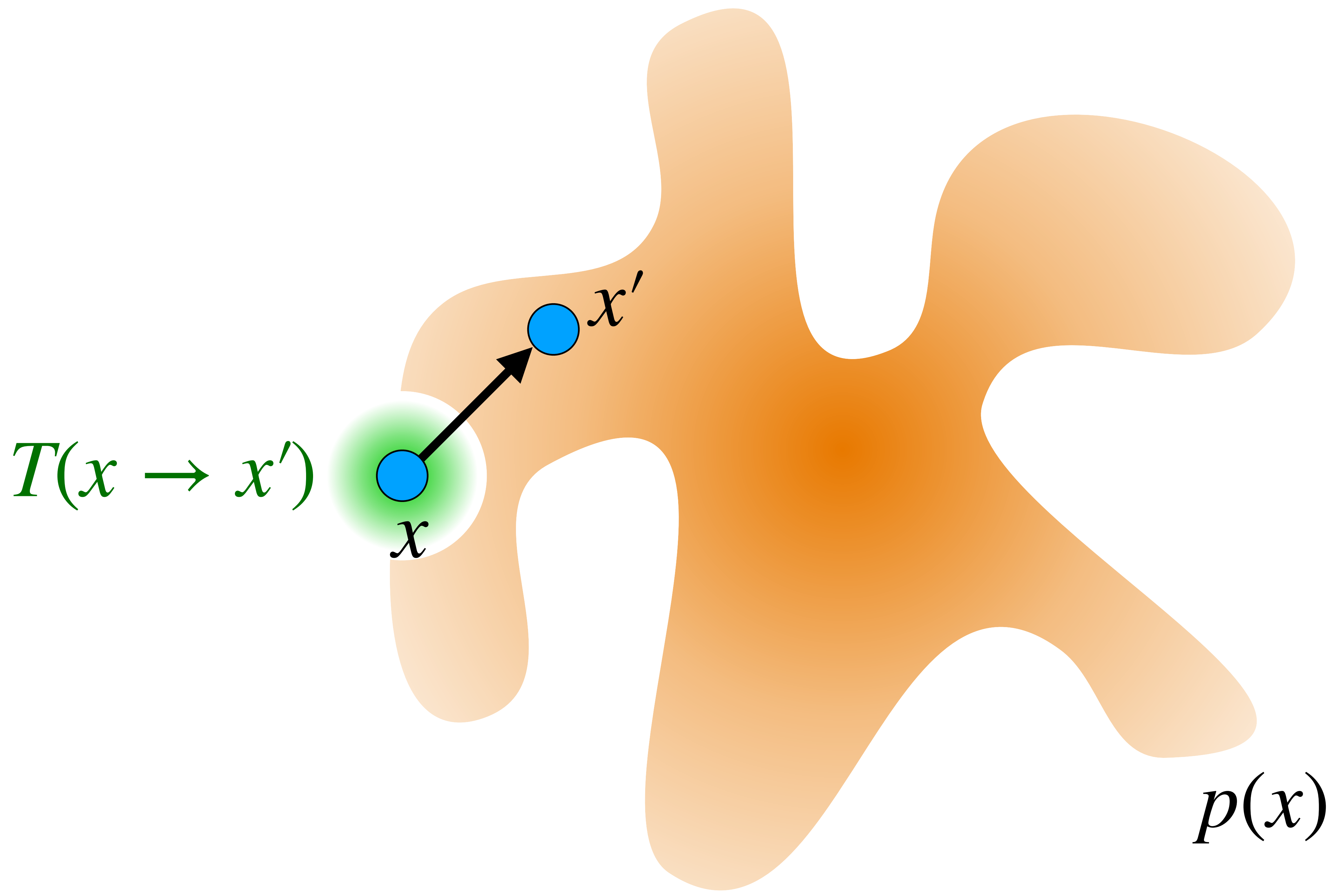
Target distribution

Metropolis-Hastings approach



Target distribution

Metropolis-Hastings approach

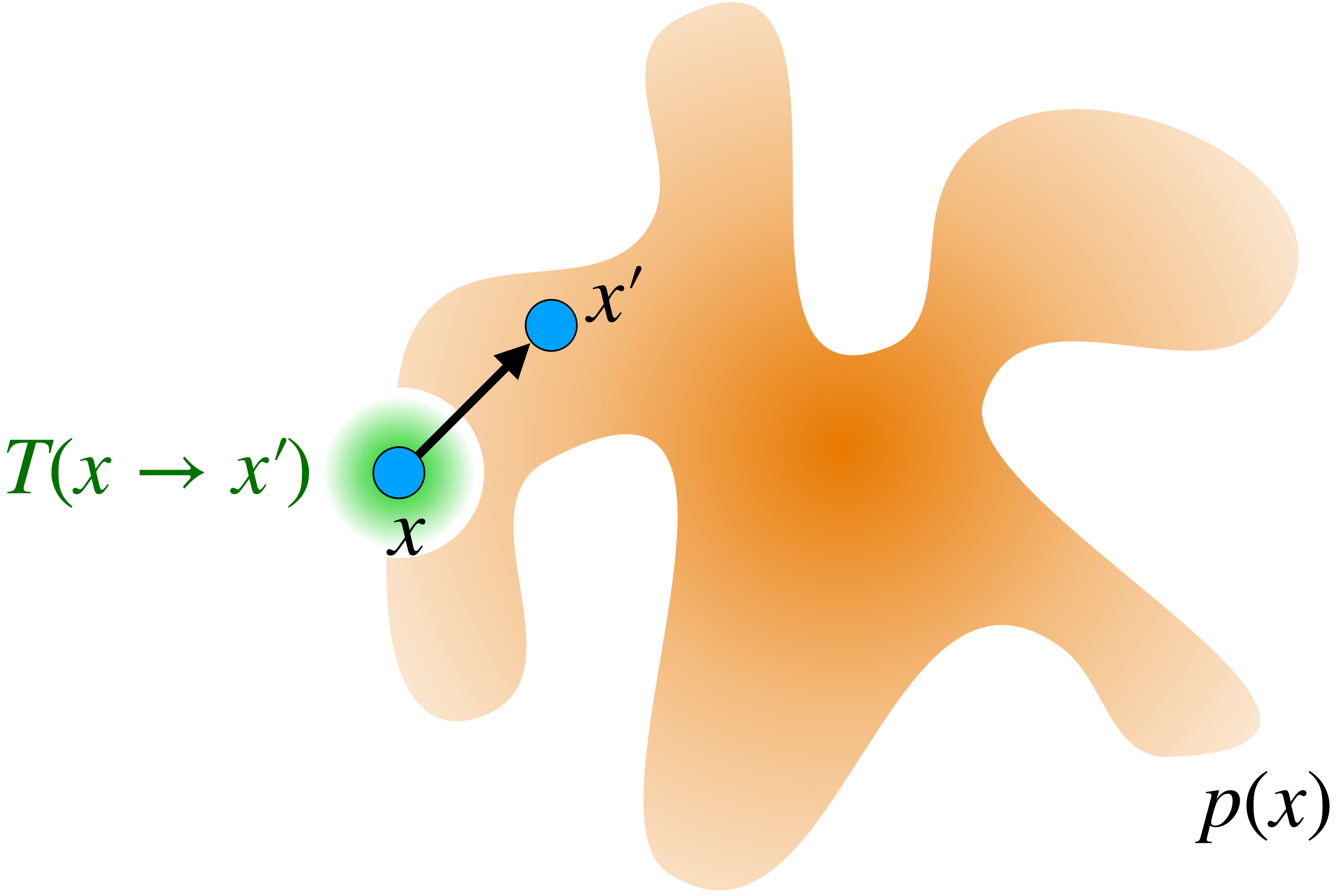


Target distribution

Metropolis-Hastings approach

$$\alpha = \frac{p(x')}{p(x)} \frac{T(x \to x')}{T(x' \to x)}$$

If $\alpha < \epsilon$ we accept, otherwise we reject

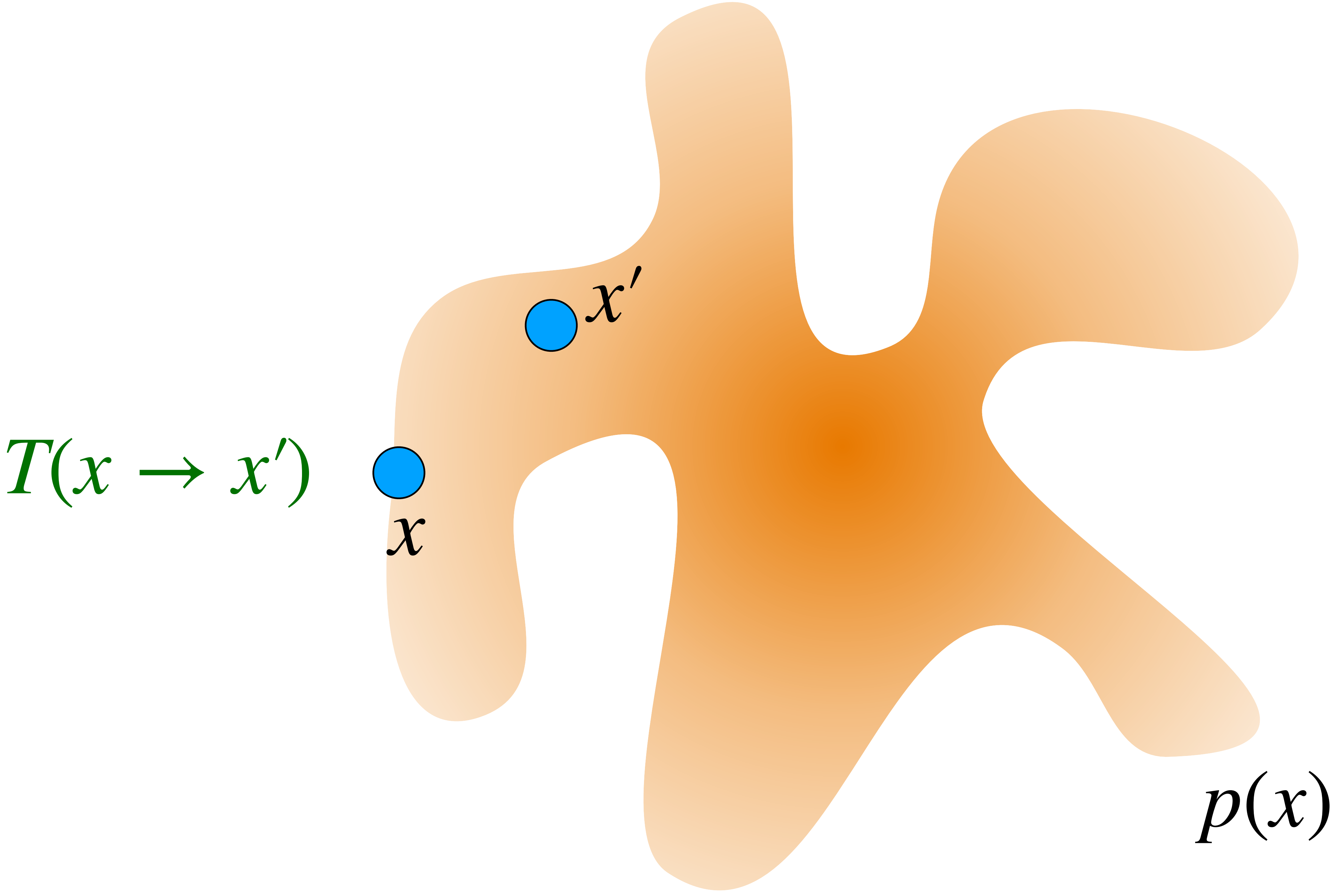


Target distribution

Metropolis-Hastings approach

$$\alpha = \frac{p(x')}{p(x)} \frac{T(x \to x')}{T(x' \to x)}$$

If $\alpha < \epsilon$ we accept, otherwise we reject

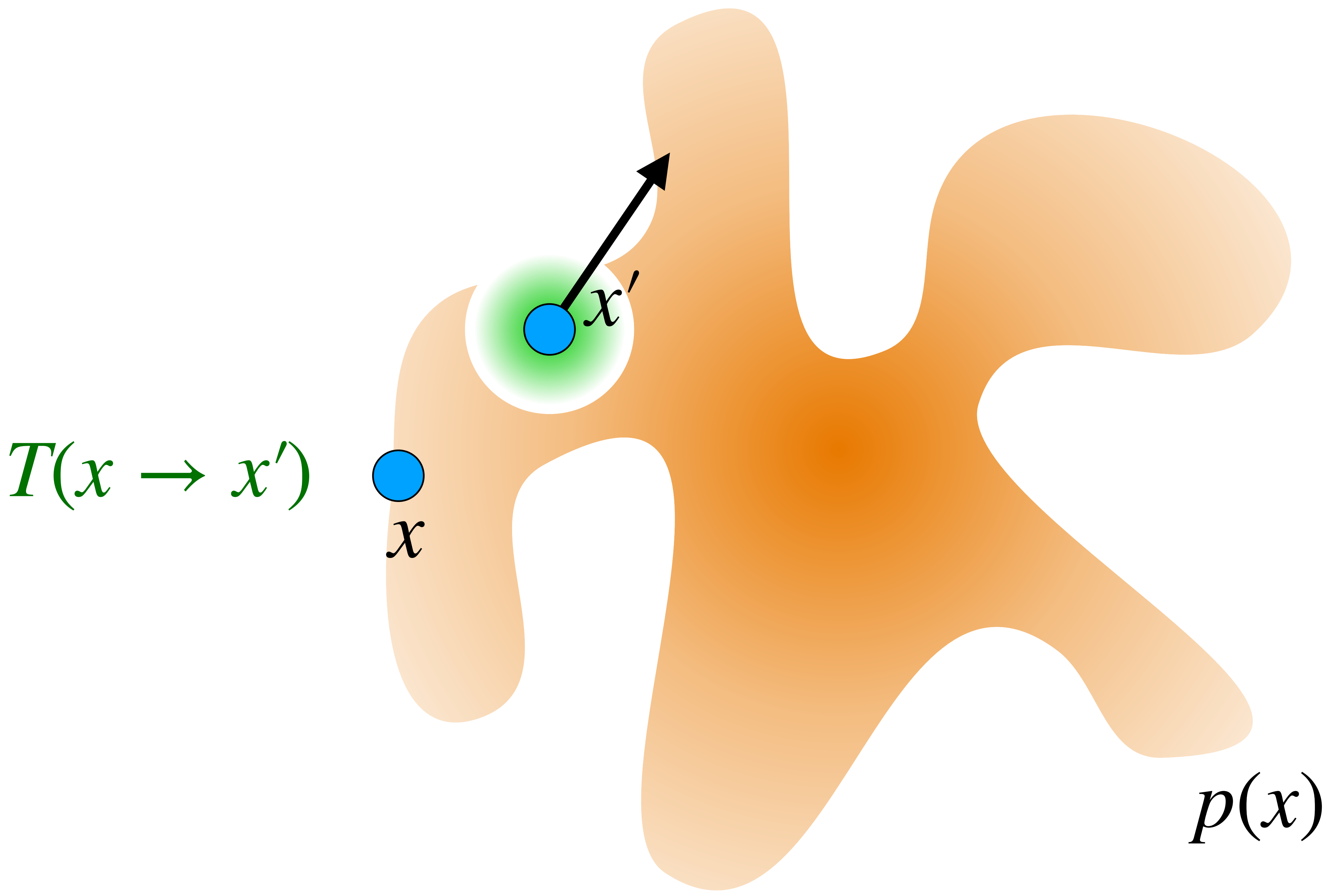


Target distribution

Metropolis-Hastings approach

$$\alpha = \frac{p(x')}{p(x)} \frac{T(x \to x')}{T(x' \to x)}$$

If $\alpha < \epsilon$ we accept, otherwise we reject

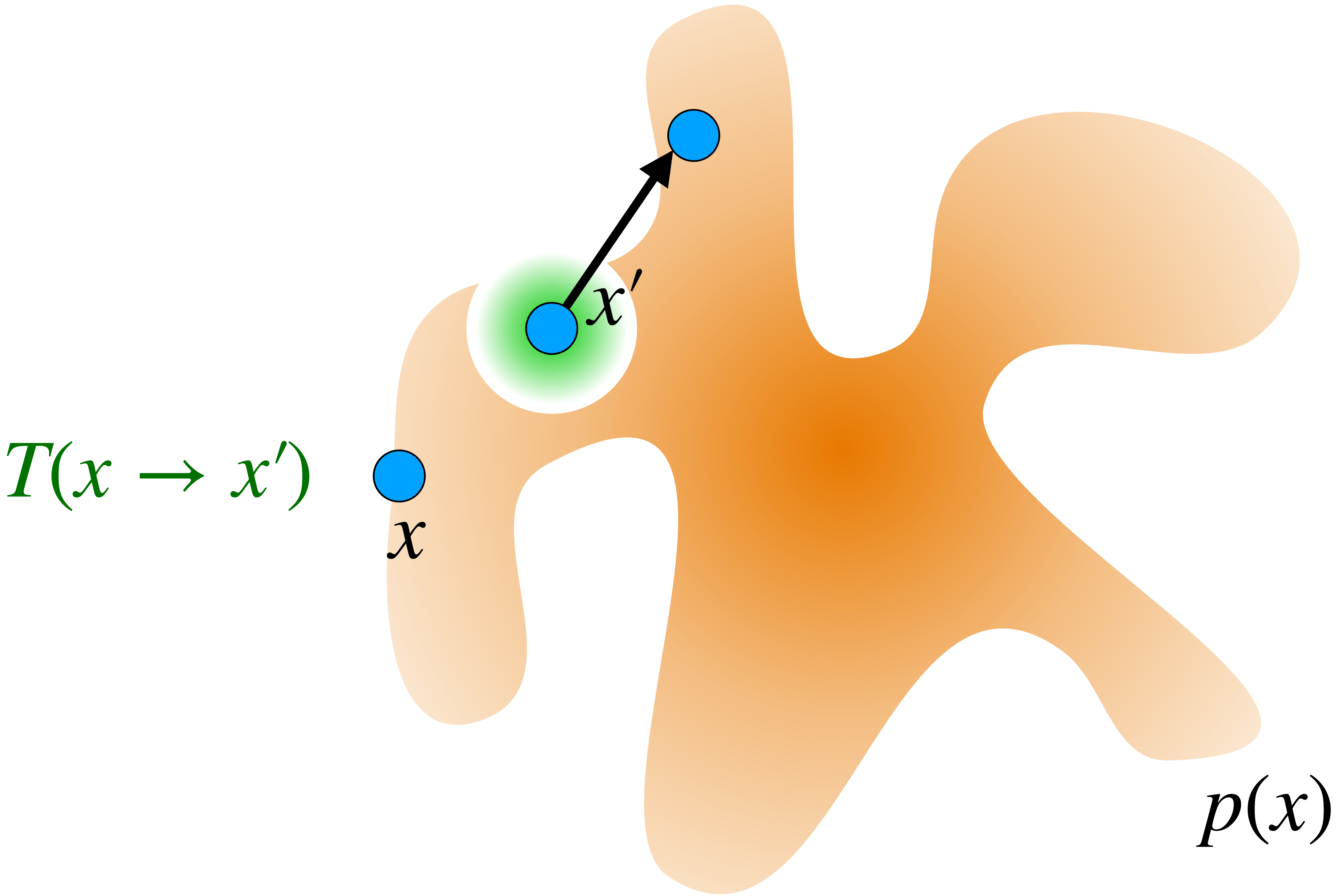


Target distribution

Metropolis-Hastings approach

$$\alpha = \frac{p(x')}{p(x)} \frac{T(x \to x')}{T(x' \to x)}$$

If $\alpha < \epsilon$ we accept, otherwise we reject

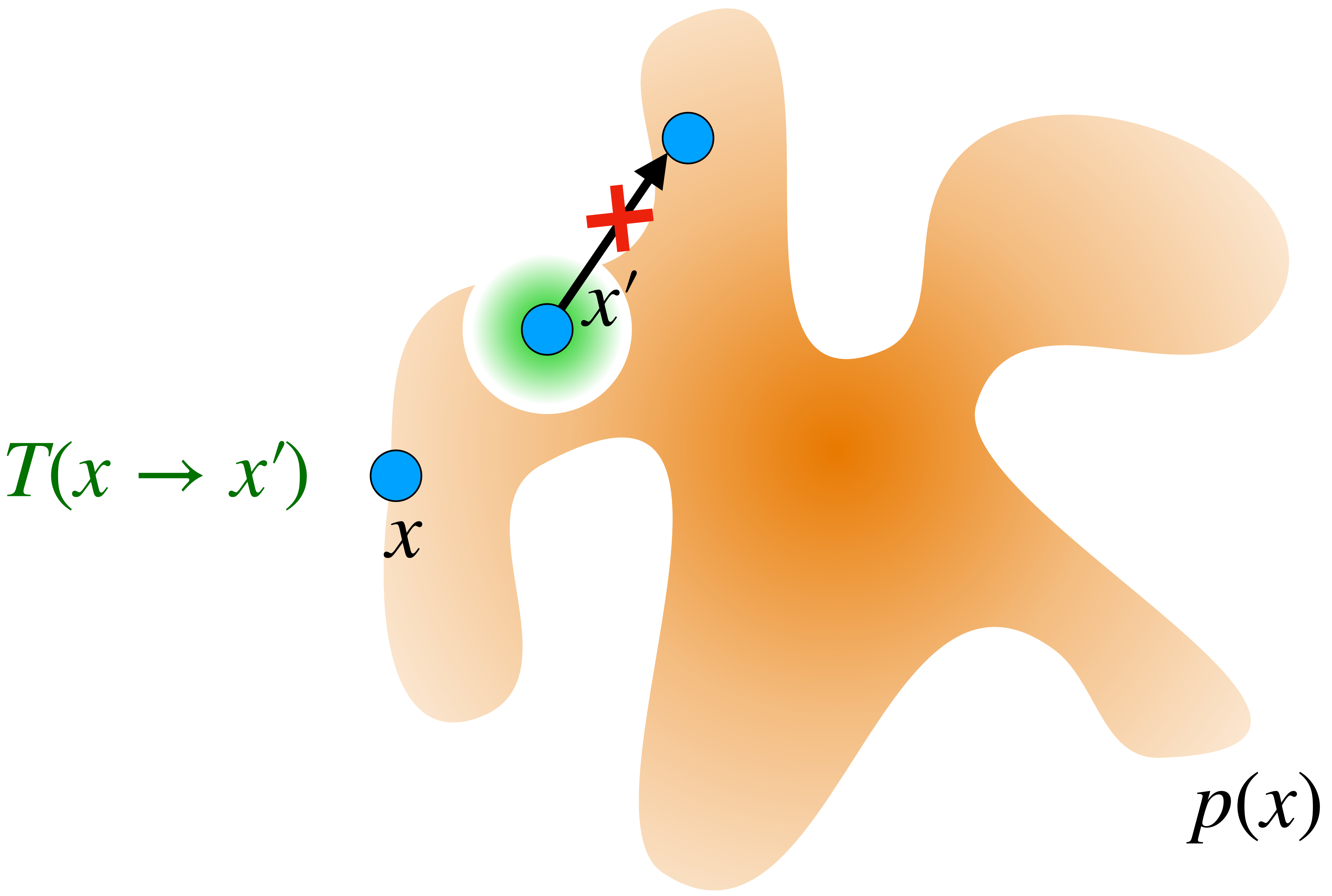


Target distribution

Metropolis-Hastings approach

$$\alpha = \frac{p(x')}{p(x)} \frac{T(x \to x')}{T(x' \to x)}$$

If $\alpha < \epsilon$ we accept, otherwise we reject

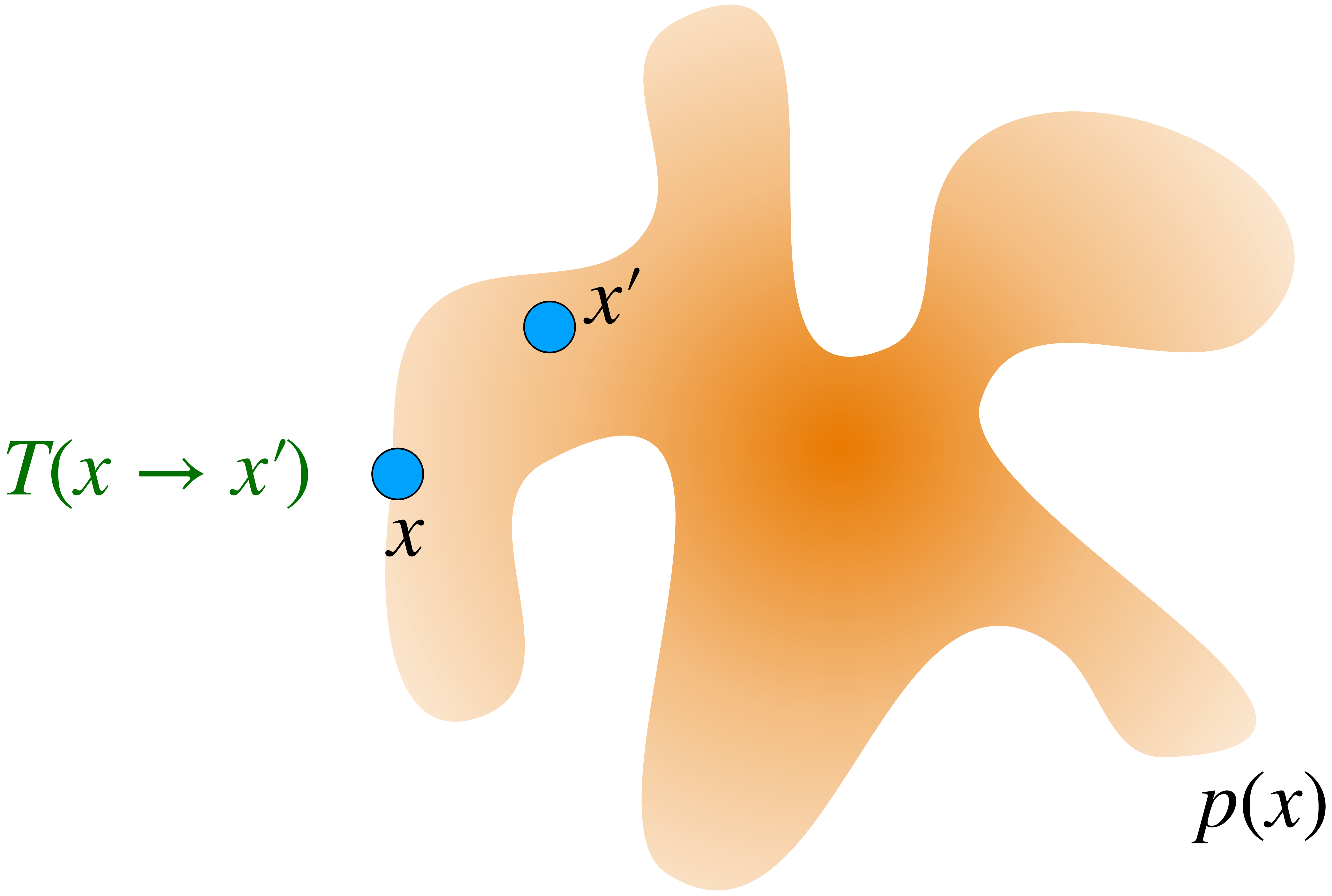


Target distribution

Metropolis-Hastings approach

$$\alpha = \frac{p(x')}{p(x)} \frac{T(x \to x')}{T(x' \to x)}$$

If $\alpha < \epsilon$ we accept, otherwise we reject

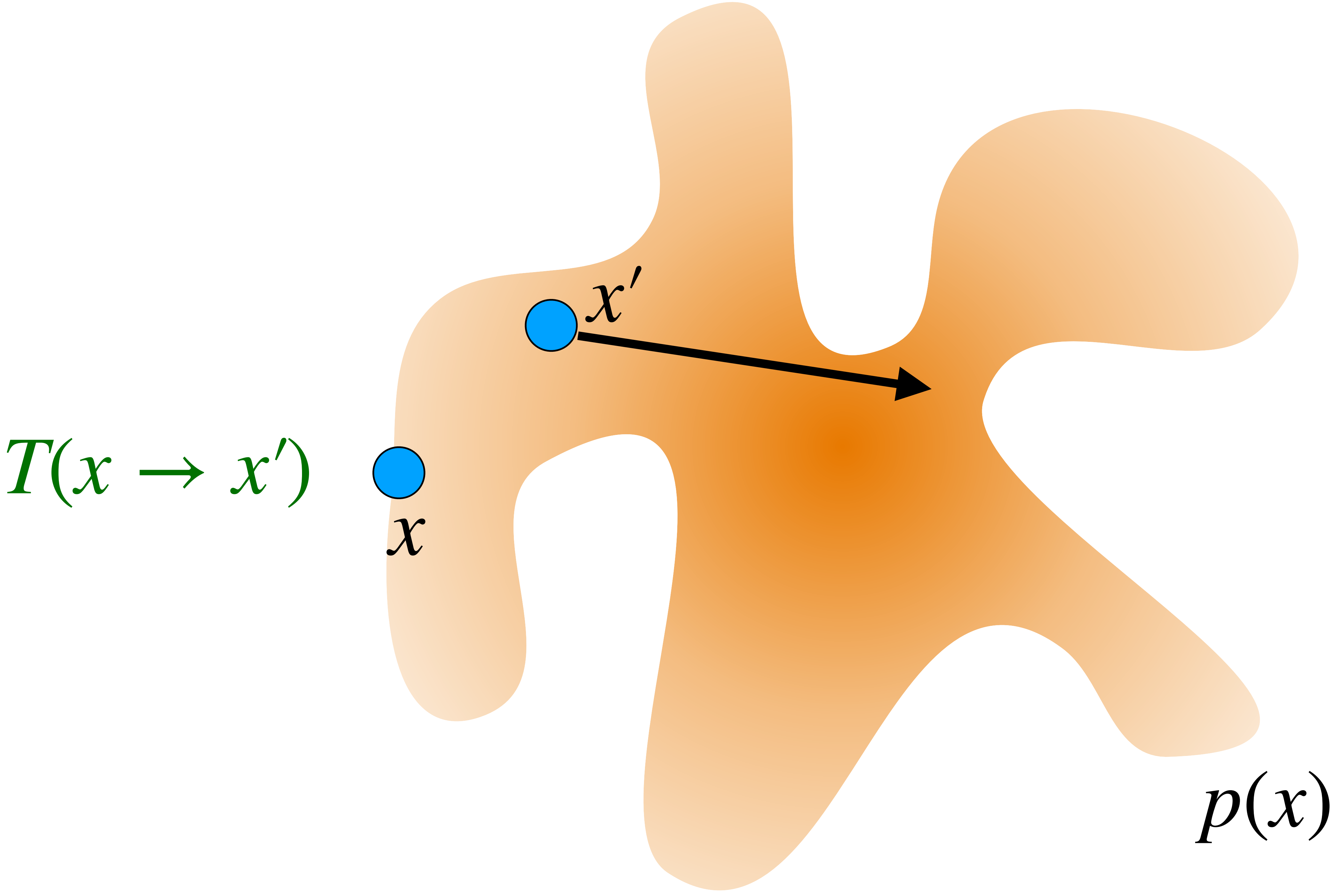


Target distribution

Metropolis-Hastings approach

$$\alpha = \frac{p(x')}{p(x)} \frac{T(x \to x')}{T(x' \to x)}$$

If $\alpha < \epsilon$ we accept, otherwise we reject

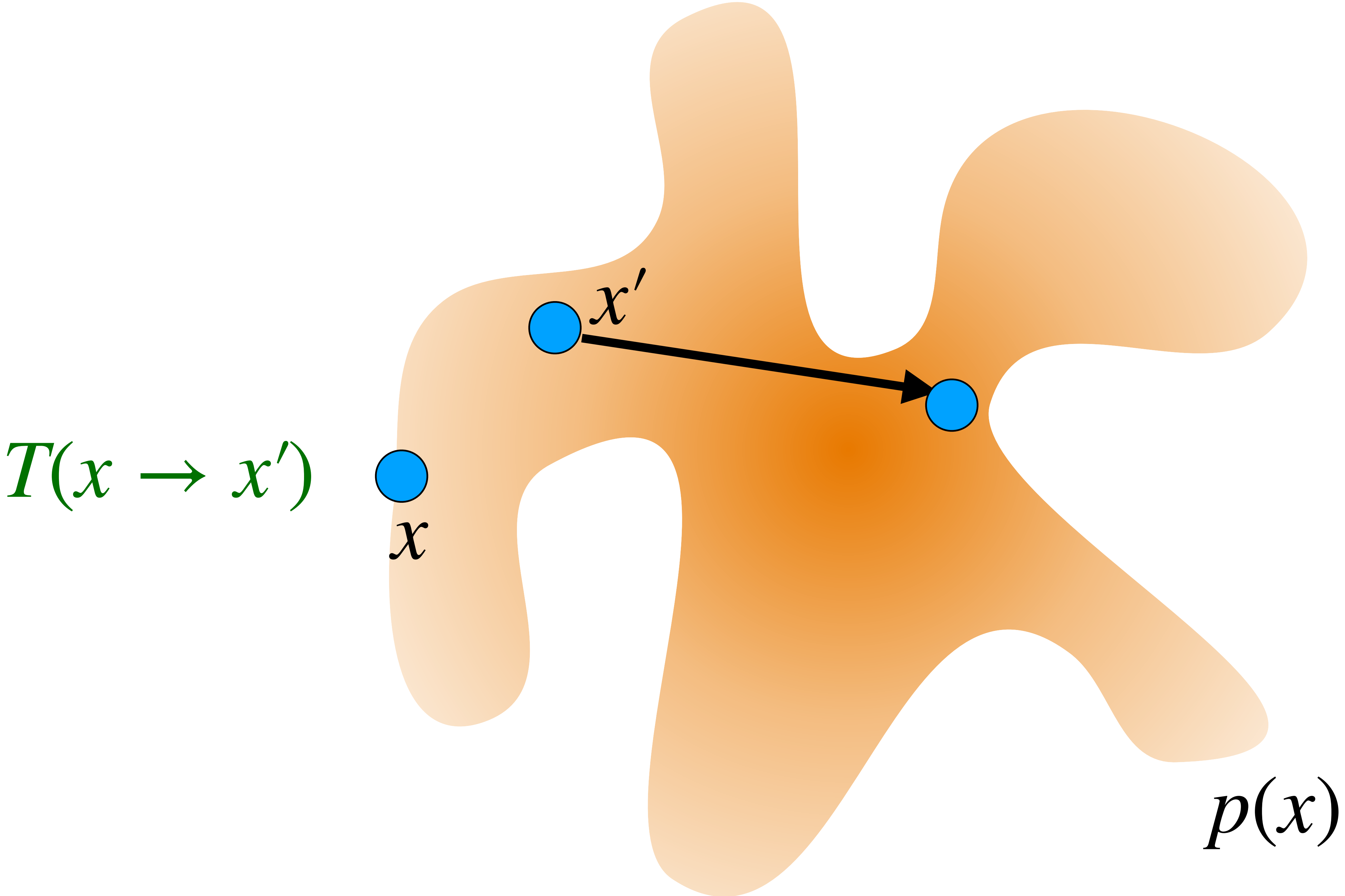


Target distribution

Metropolis-Hastings approach

$$\alpha = \frac{p(x')}{p(x)} \frac{T(x \to x')}{T(x' \to x)}$$

If $\alpha < \epsilon$ we accept, otherwise we reject

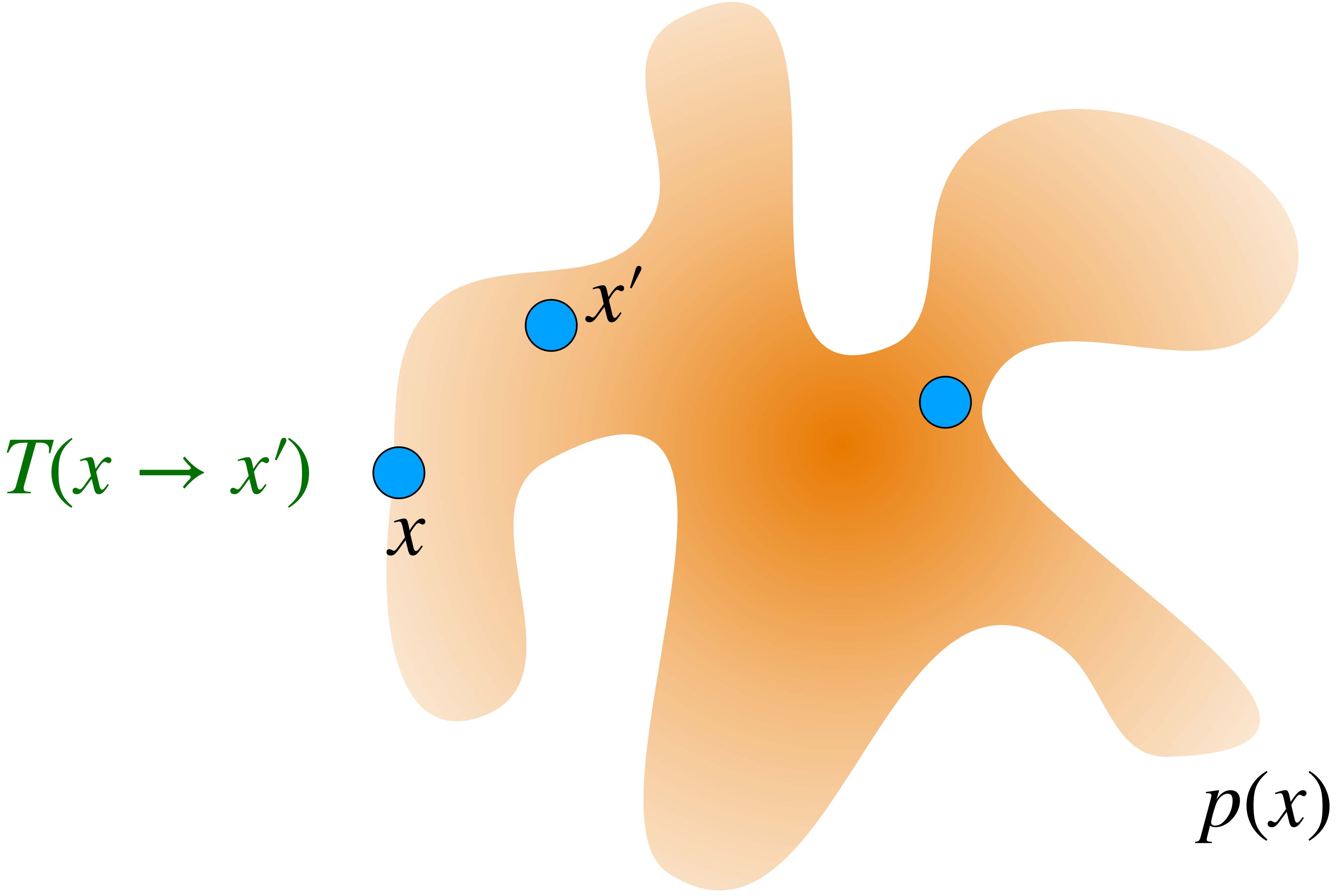


Target distribution

Metropolis-Hastings approach

$$\alpha = \frac{p(x')}{p(x)} \frac{T(x \to x')}{T(x' \to x)}$$

If $\alpha < \epsilon$ we accept, otherwise we reject

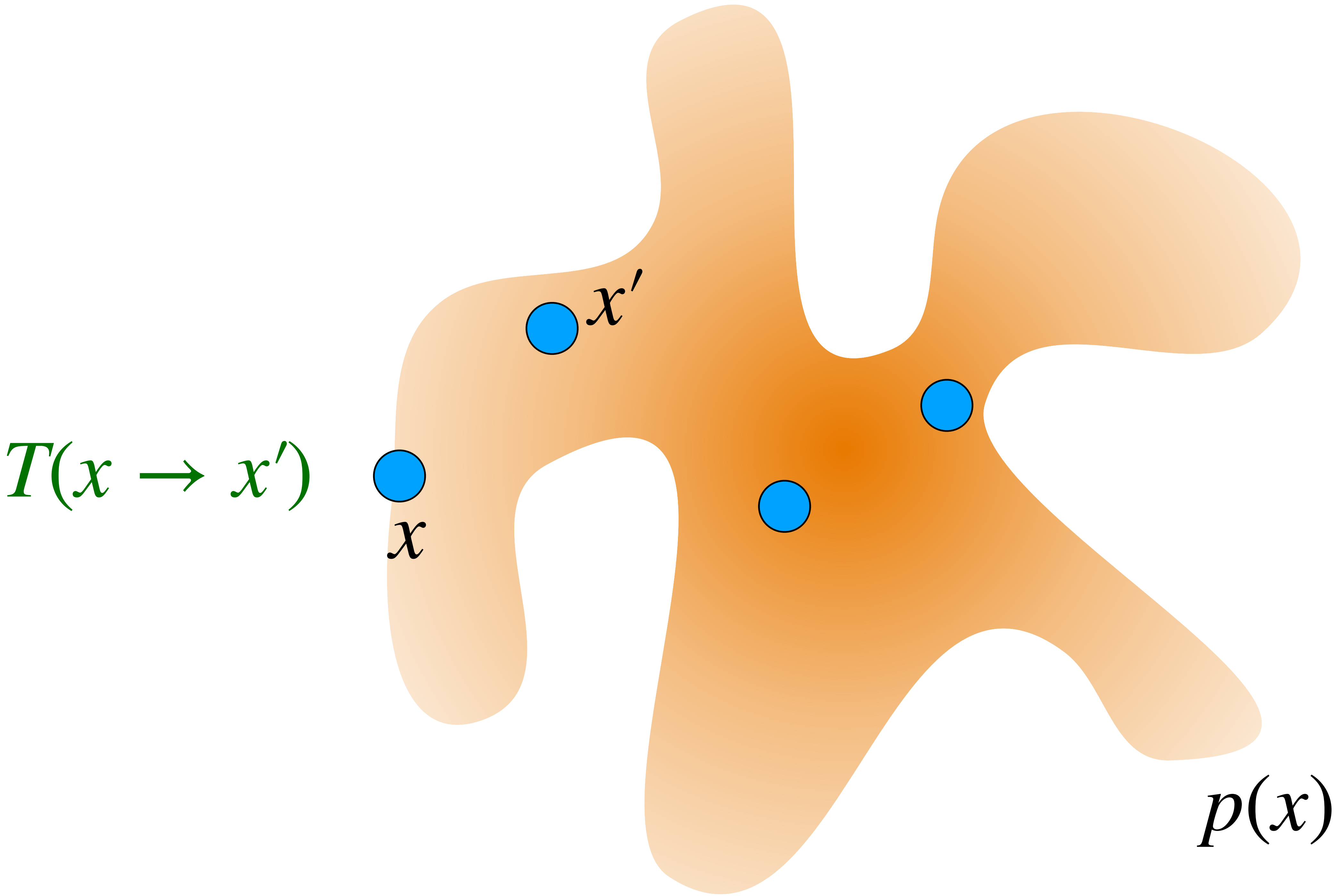


Target distribution

Metropolis-Hastings approach

$$\alpha = \frac{p(x')}{p(x)} \frac{T(x \to x')}{T(x' \to x)}$$

If $\alpha < \epsilon$ we accept, otherwise we reject

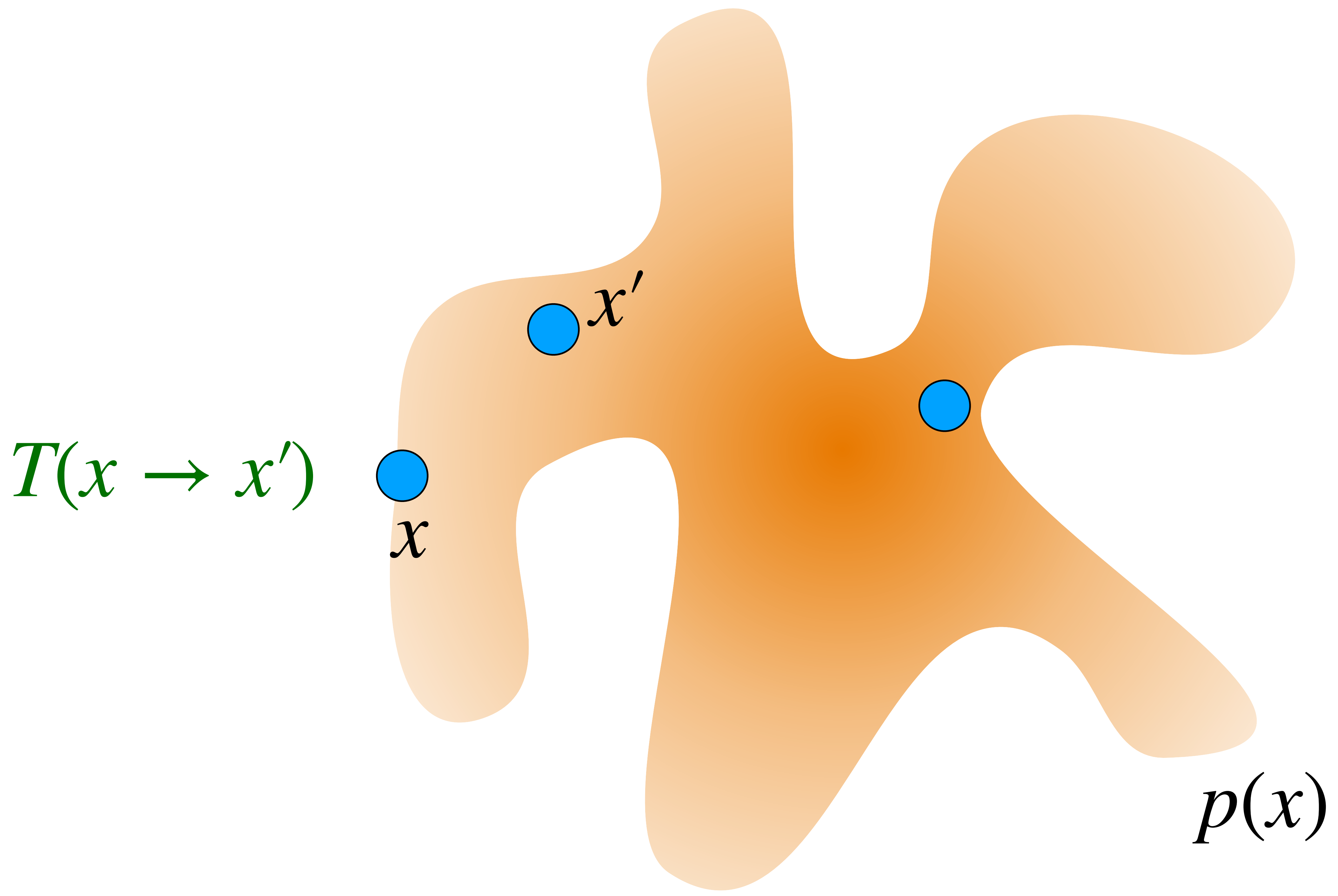


Target distribution

Metropolis-Hastings approach

$$\alpha = \frac{p(x')}{p(x)} \frac{T(x \rightarrow x')}{T(x' \rightarrow x)}$$

If $\alpha < \epsilon$ we accept, otherwise we reject

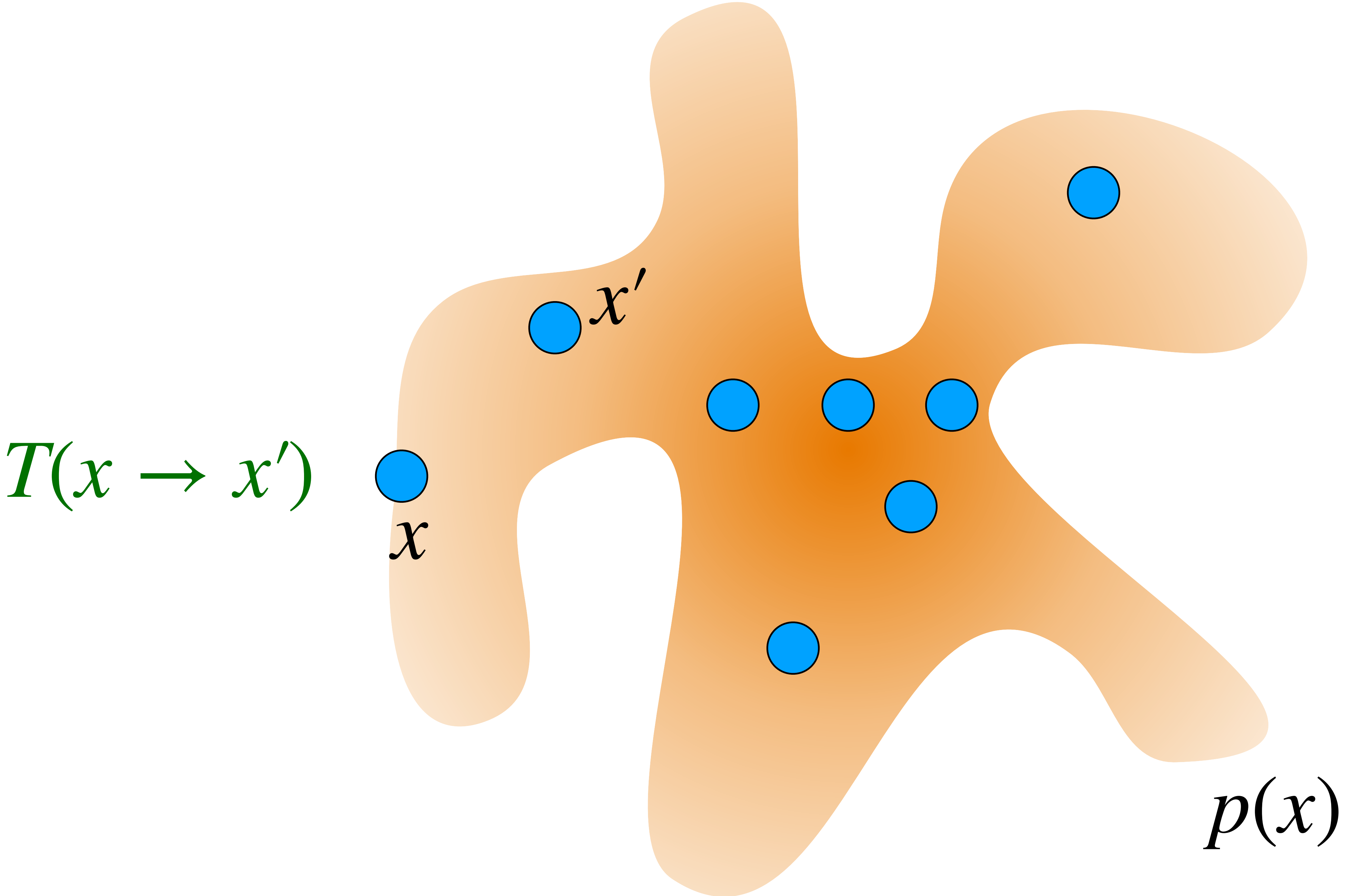


Target distribution

Metropolis-Hastings approach

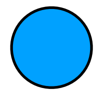
$$\alpha = \frac{p(x')}{p(x)} \frac{T(x \rightarrow x')}{T(x' \rightarrow x)}$$

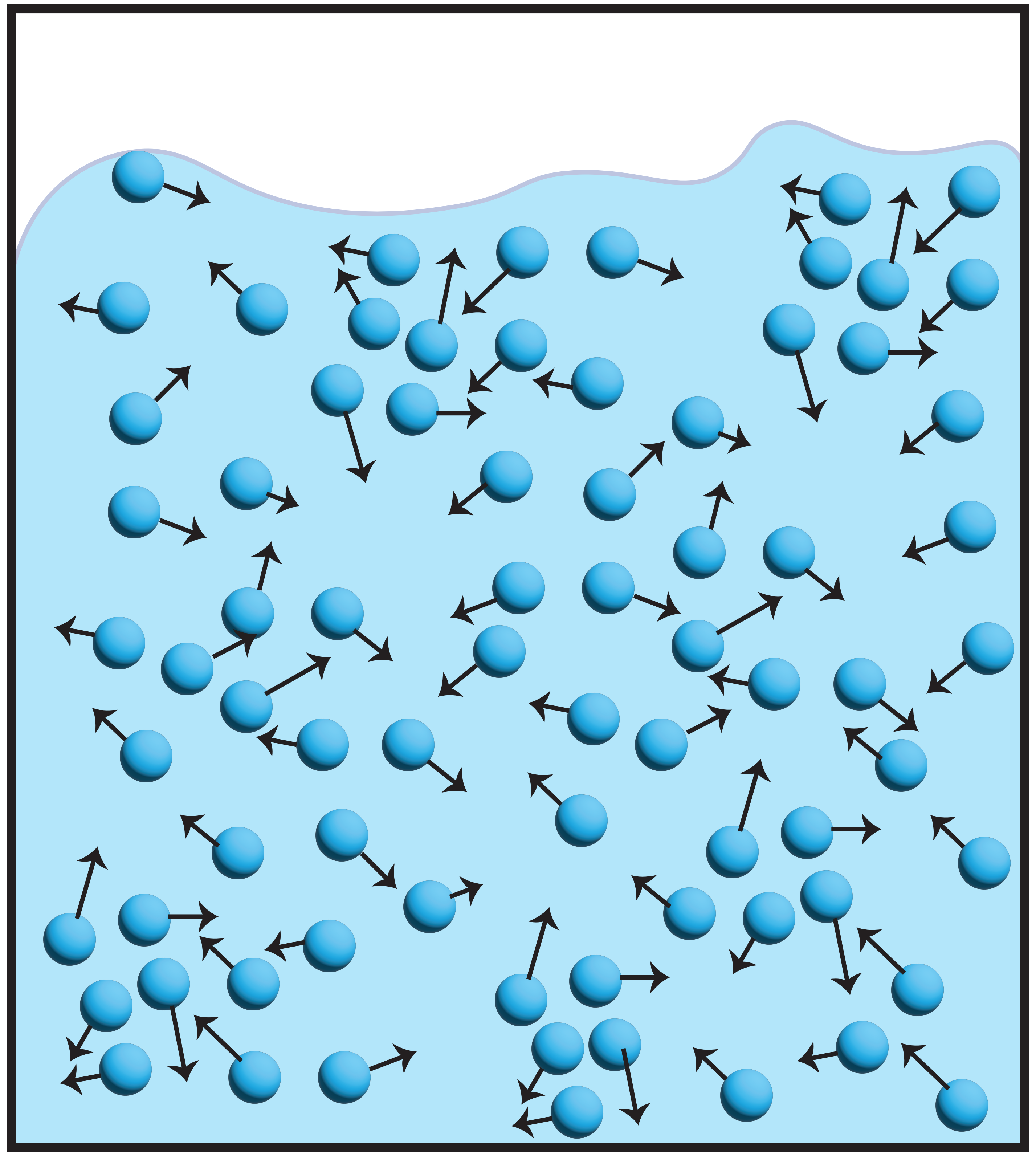
If $\alpha < \epsilon$ we accept, otherwise we reject



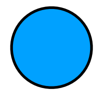
Target distribution

Mathematical formulation

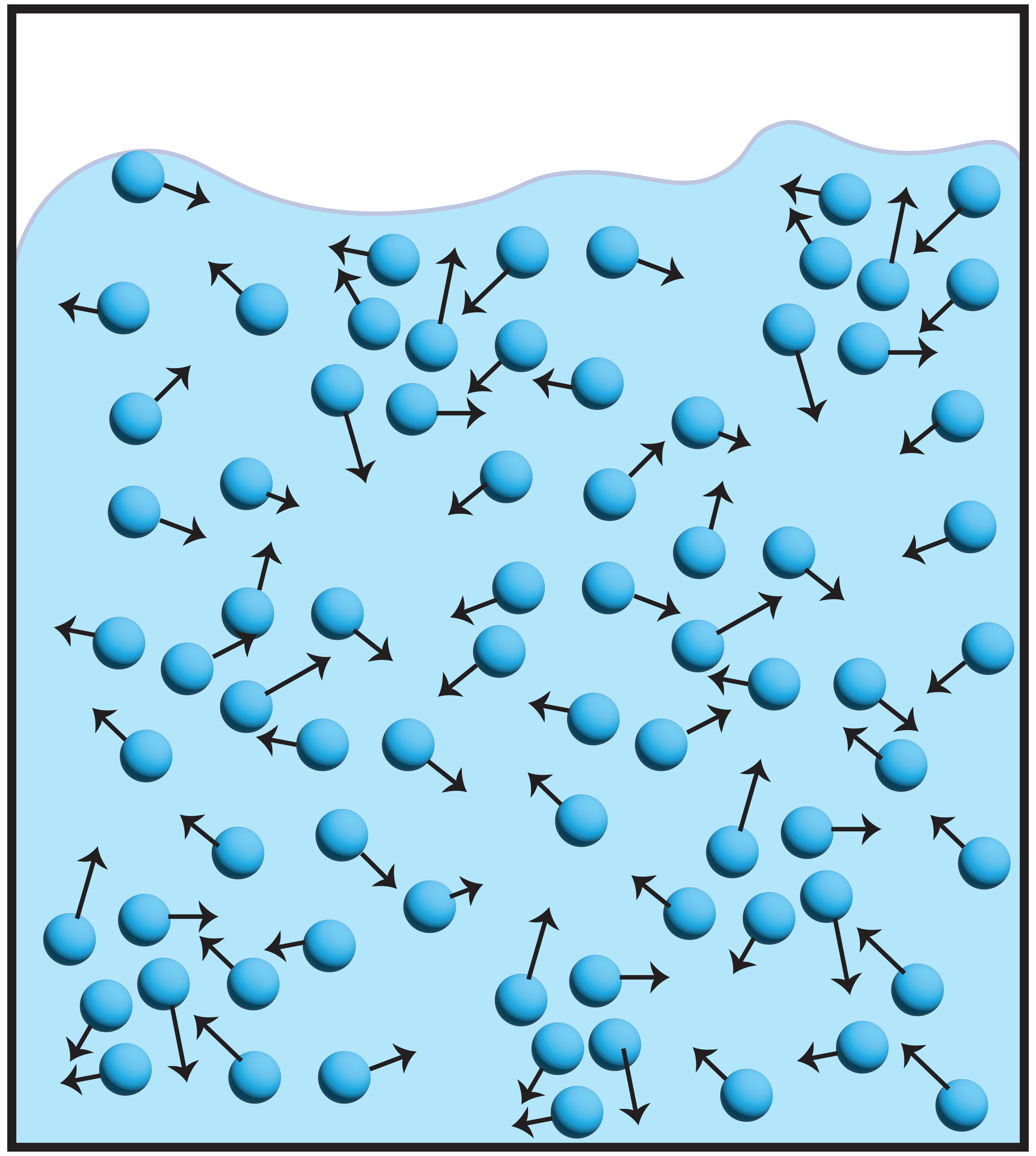




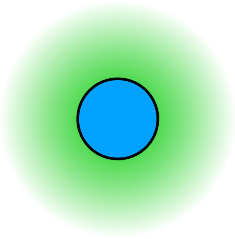
Mathematical formulation



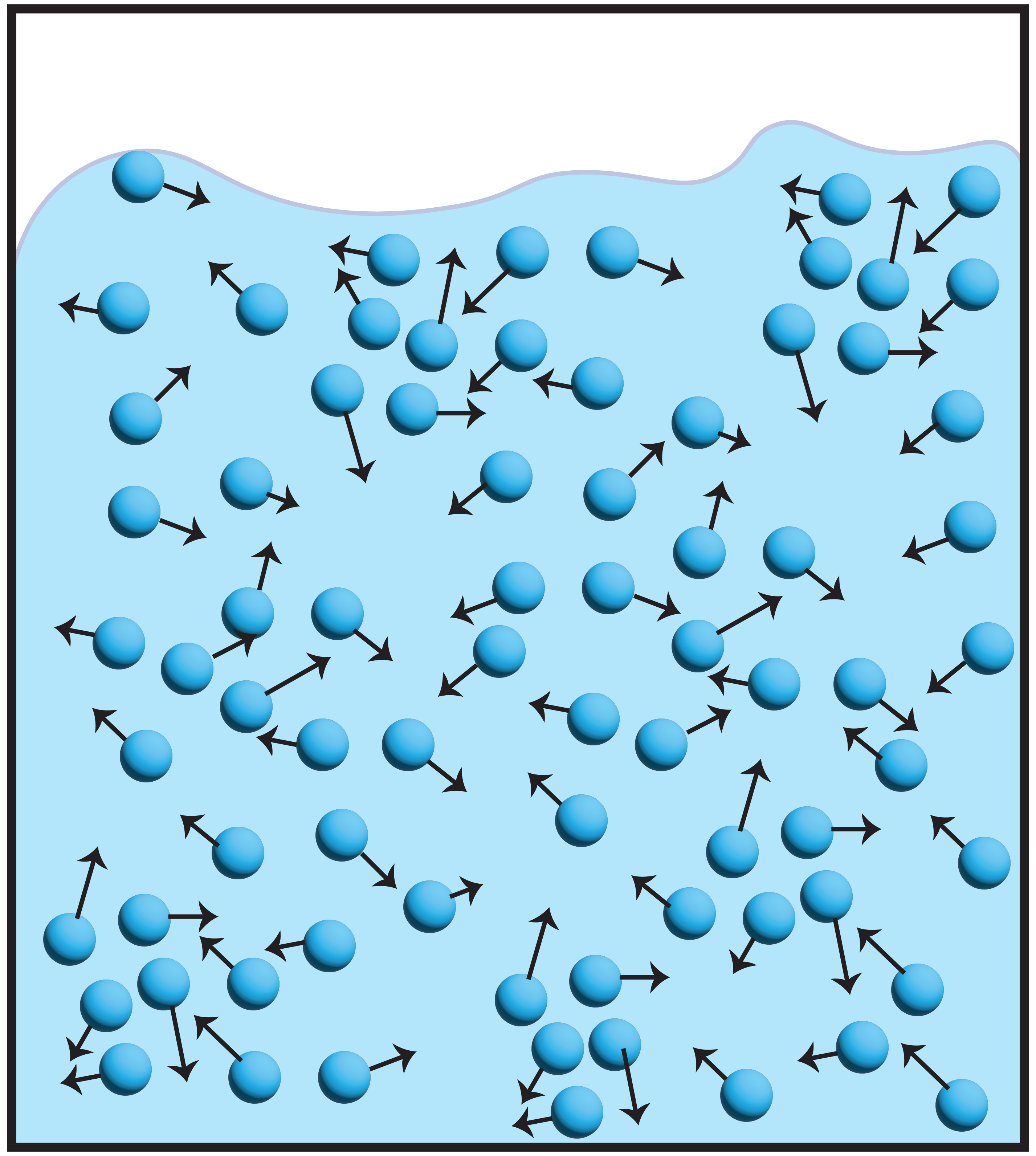
$\mathbf{x}_{t+1} = \mathbf{x}_t$ + drift



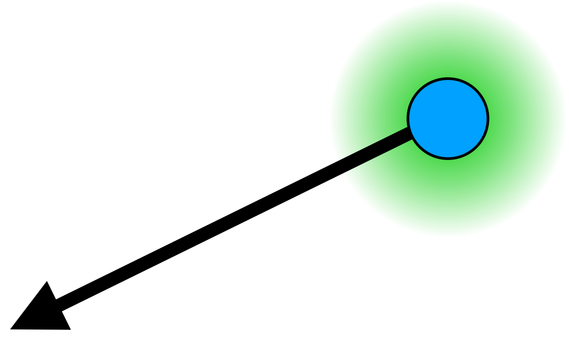
Mathematical formulation



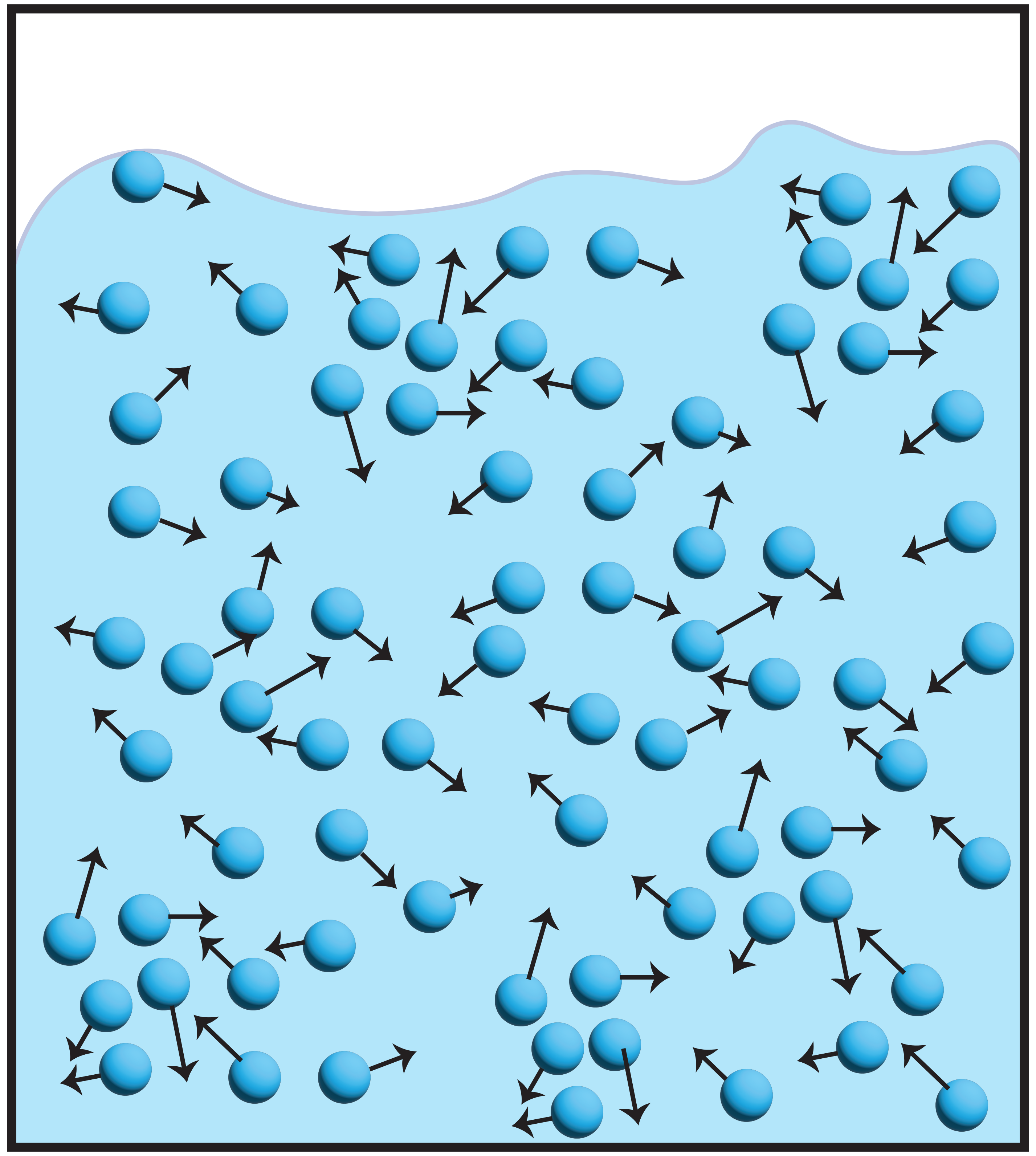
$\mathbf{x}_{t+1} = \mathbf{x}_t$ + drift



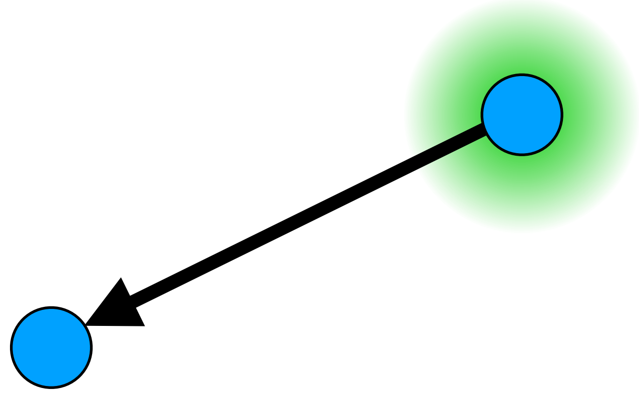
Mathematical formulation



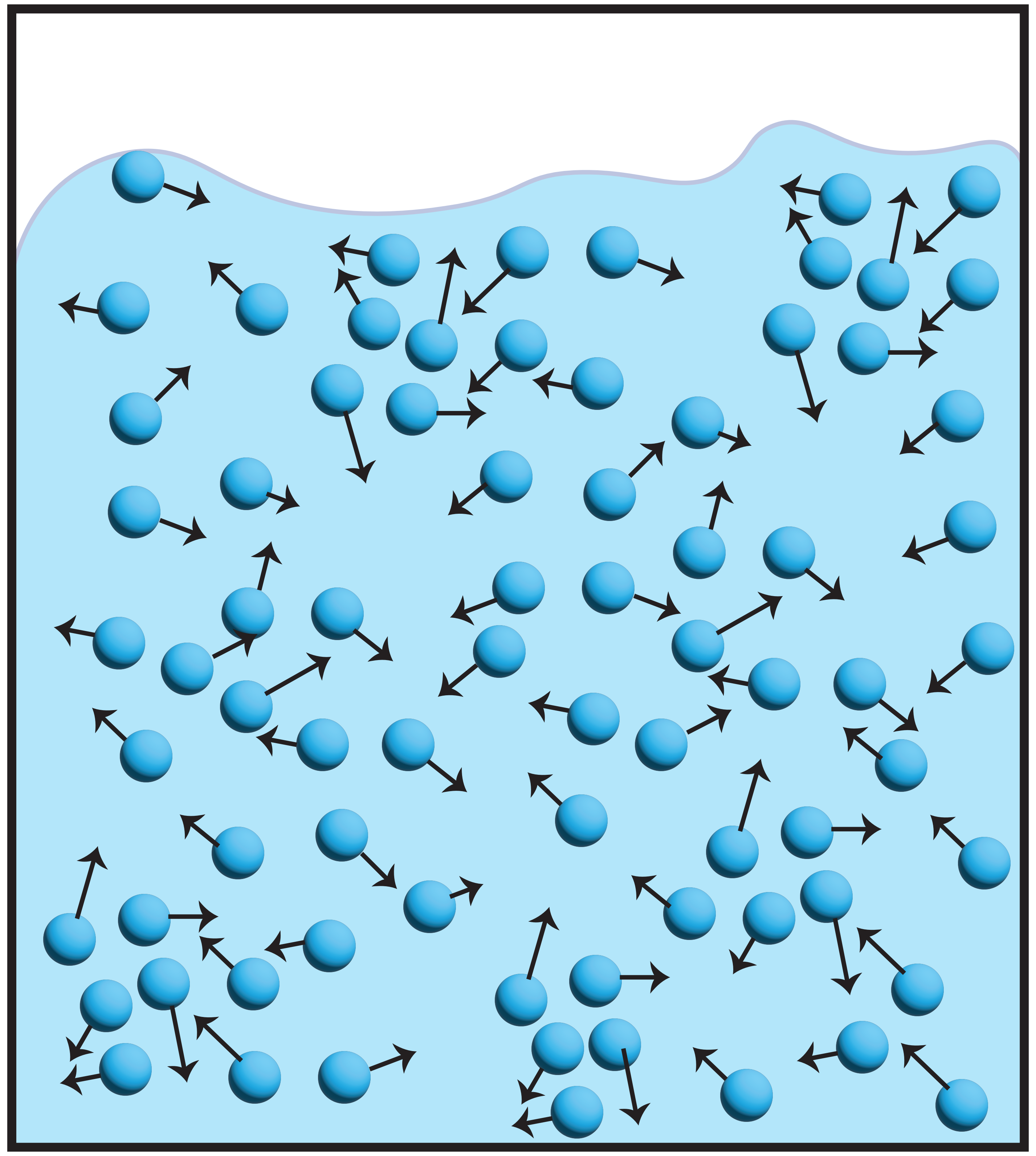
$\mathbf{x}_{t+1} = \mathbf{x}_t$ + drift



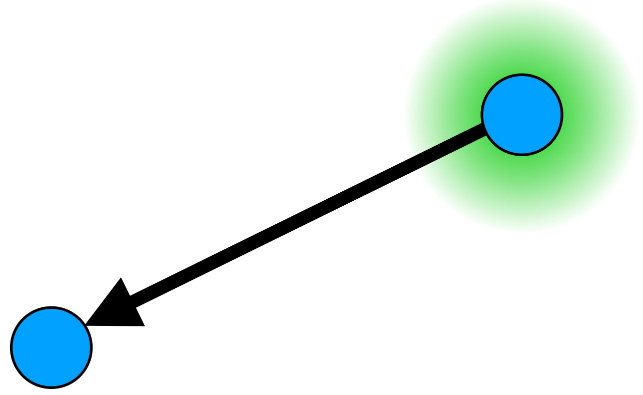
Mathematical formulation



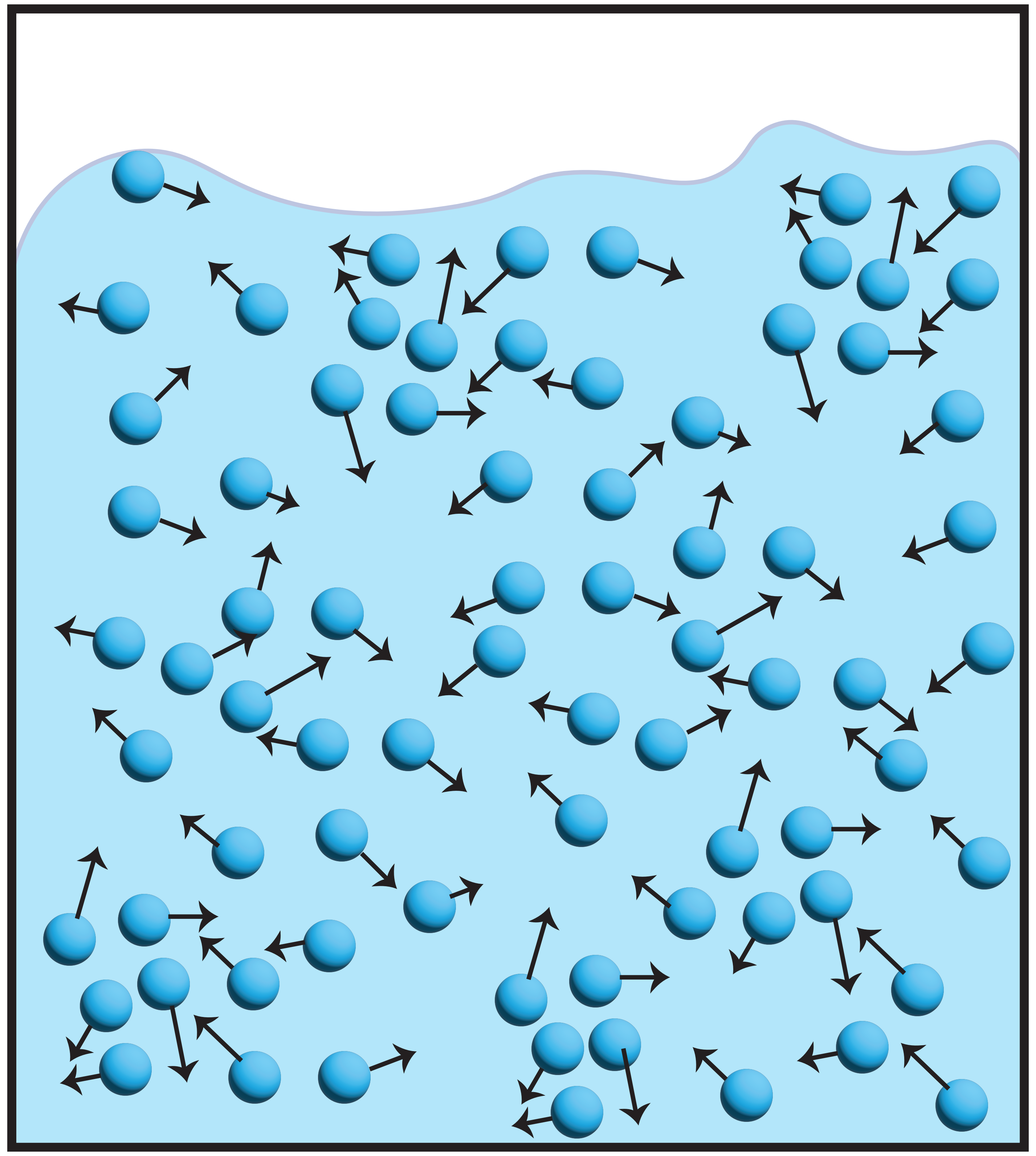
$$\mathbf{x}_{t+1} = \mathbf{x}_t \text{ + drift}$$



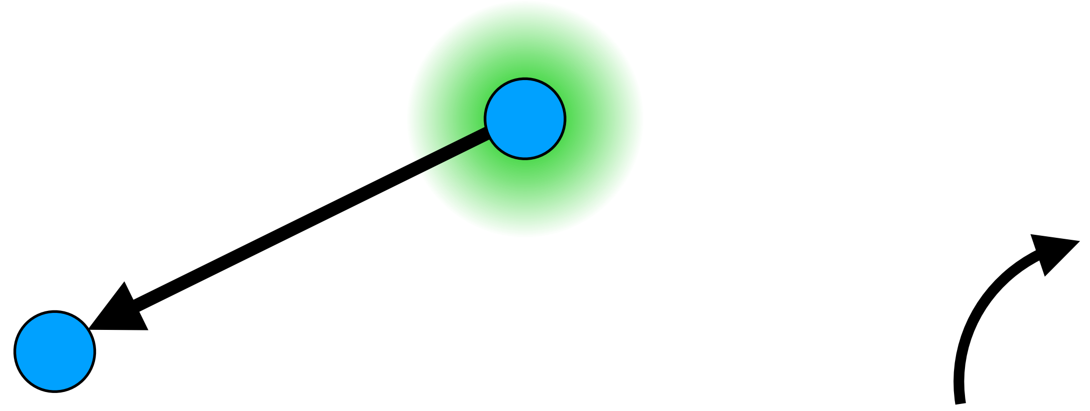
Mathematical formulation



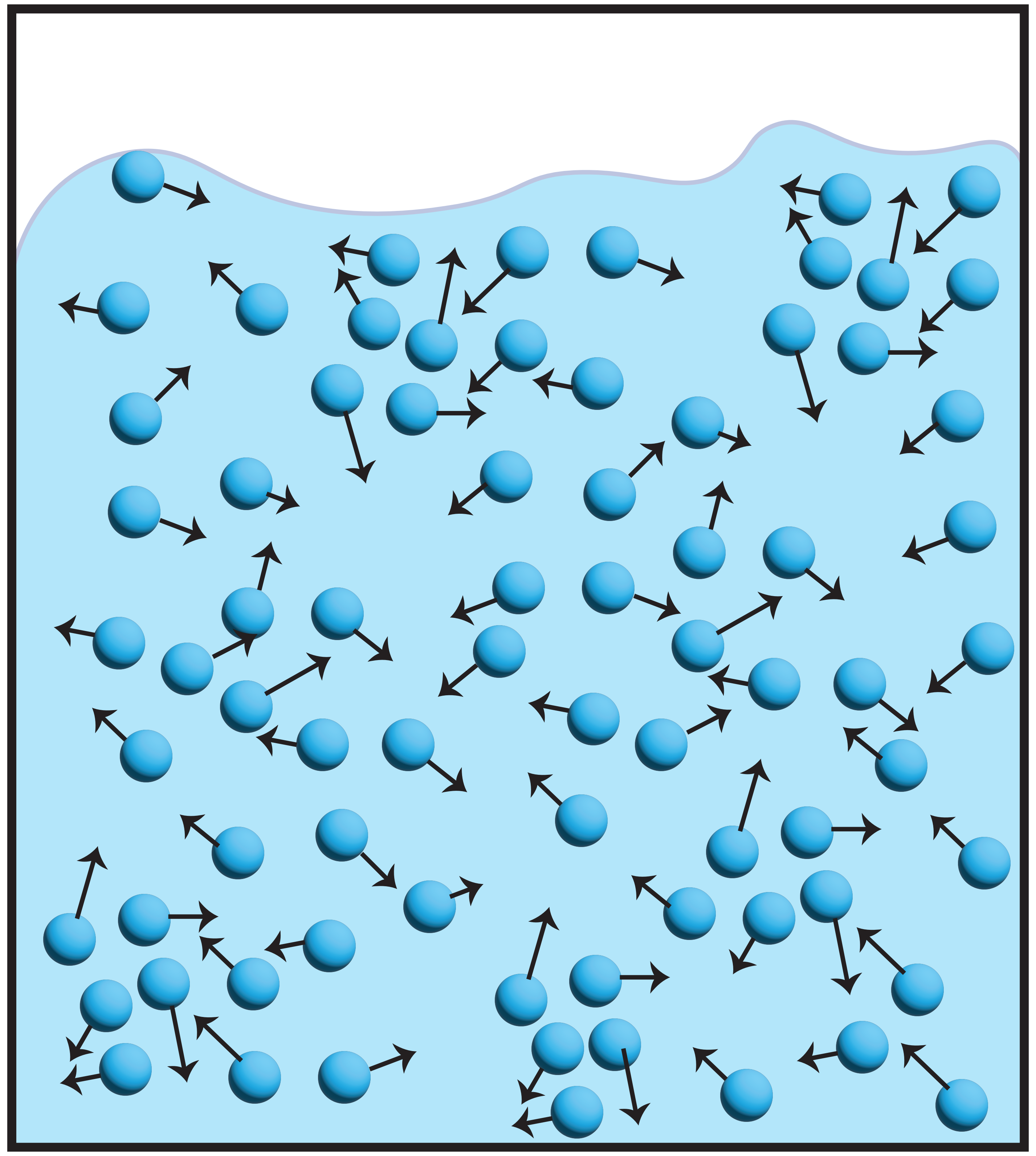
$\mathbf{x}_{t+1} = \mathbf{x}_t$ + drift + randomness



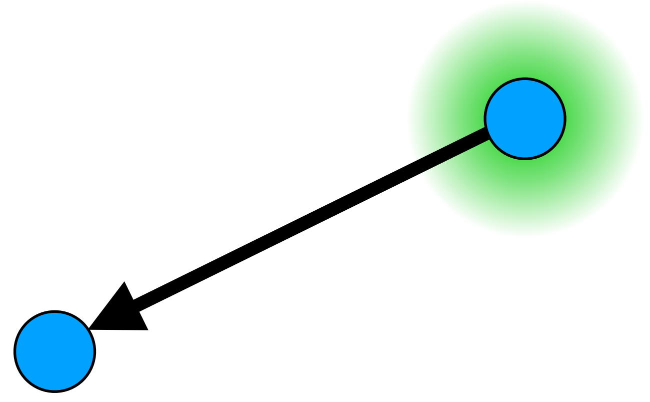
Mathematical formulation



$\mathbf{x}_{t+1} = \mathbf{x}_t$ + drift + randomness

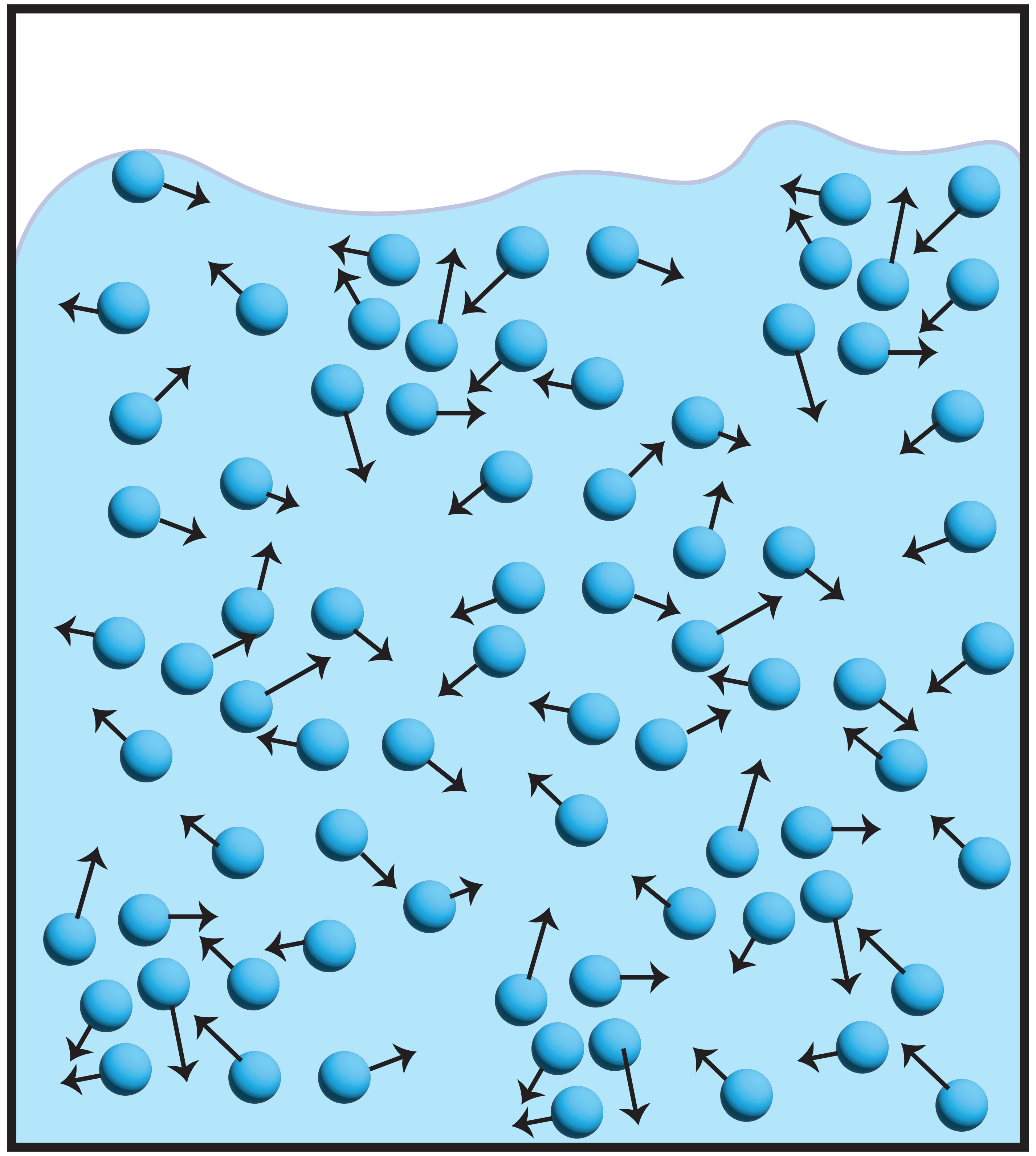


Mathematical formulation

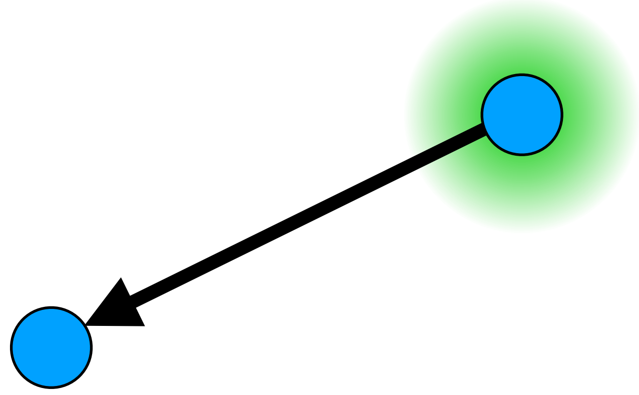


$$\mathbf{x}_{t+1} = \mathbf{x}_t + \text{drift} + \text{randomness}$$

average trajectory

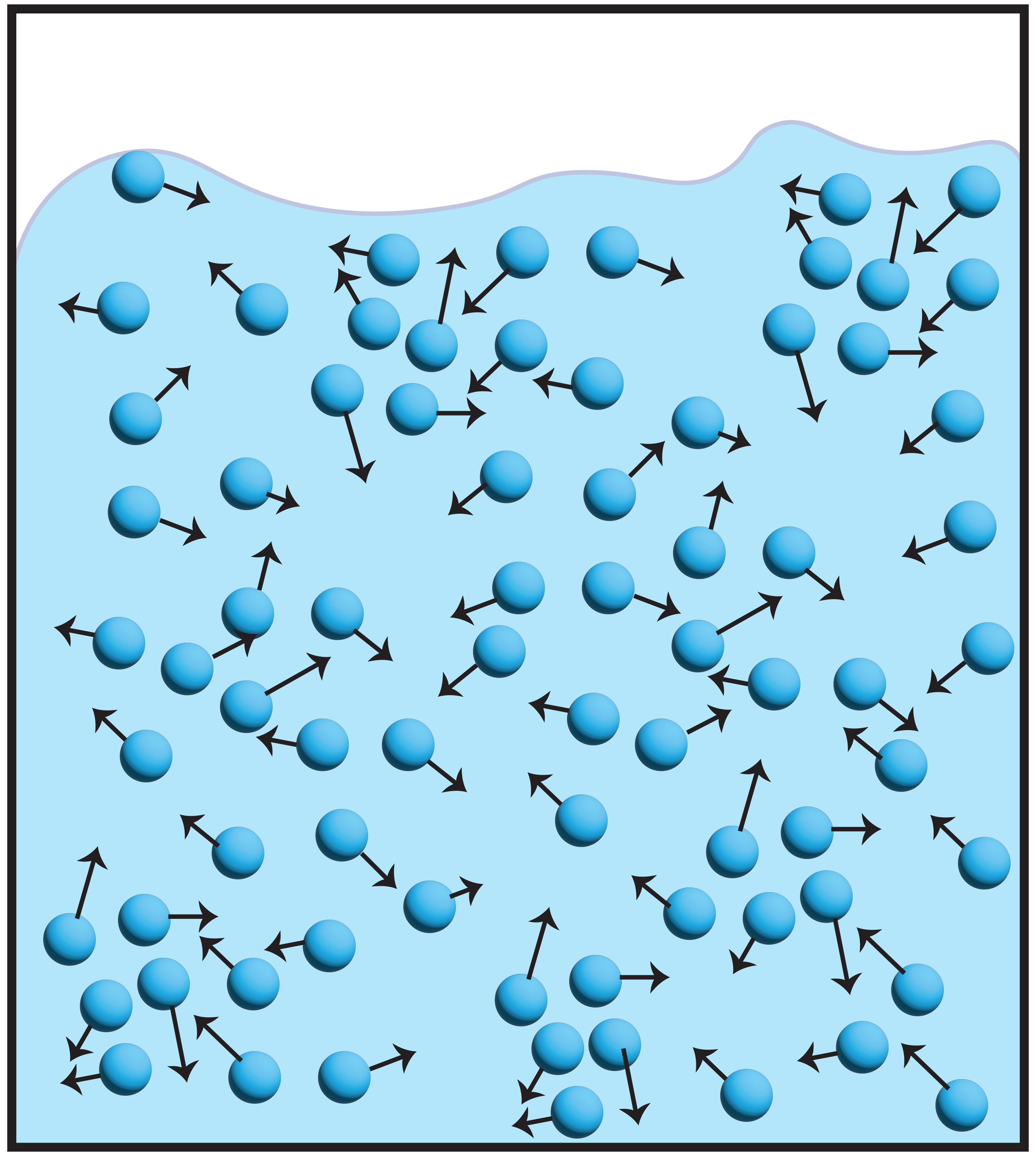


Mathematical formulation

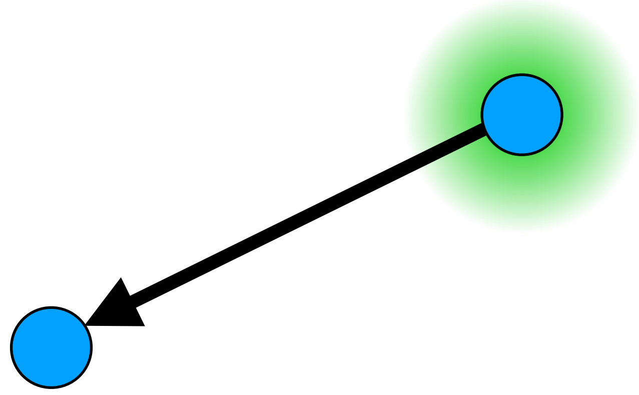


average trajectory

$\mathbf{x}_{t+1} = \mathbf{x}_t$ + drift + randomness



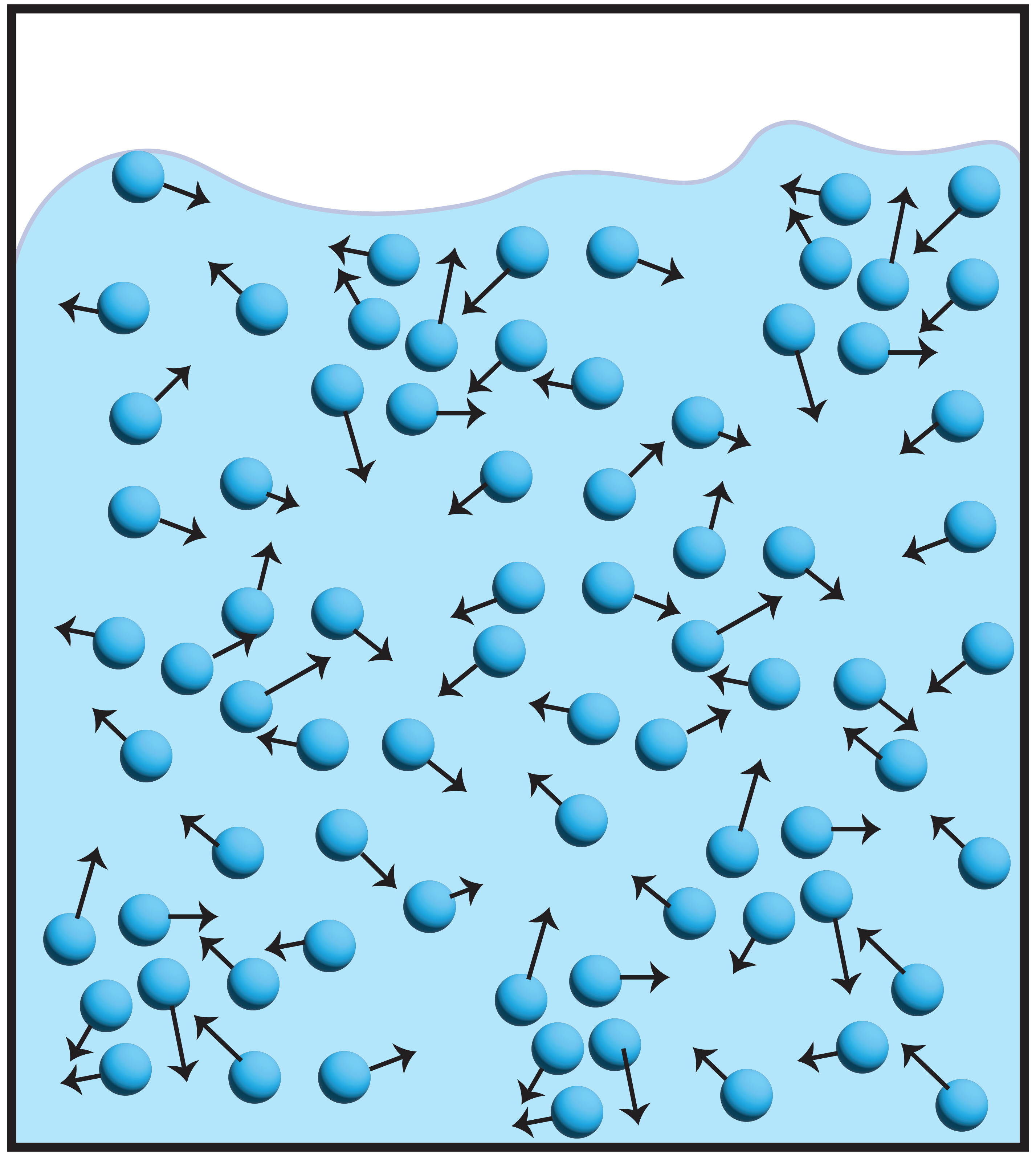
Mathematical formulation



$\mathbf{x}_{t+1} = \mathbf{x}_t$ + drift + randomness

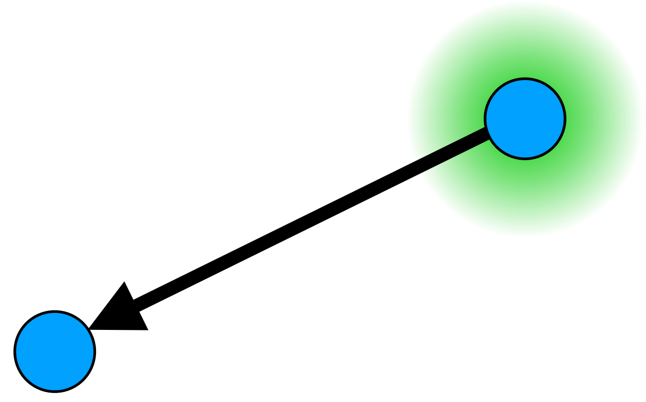
average trajectory

jiggle



Mathematical formulation

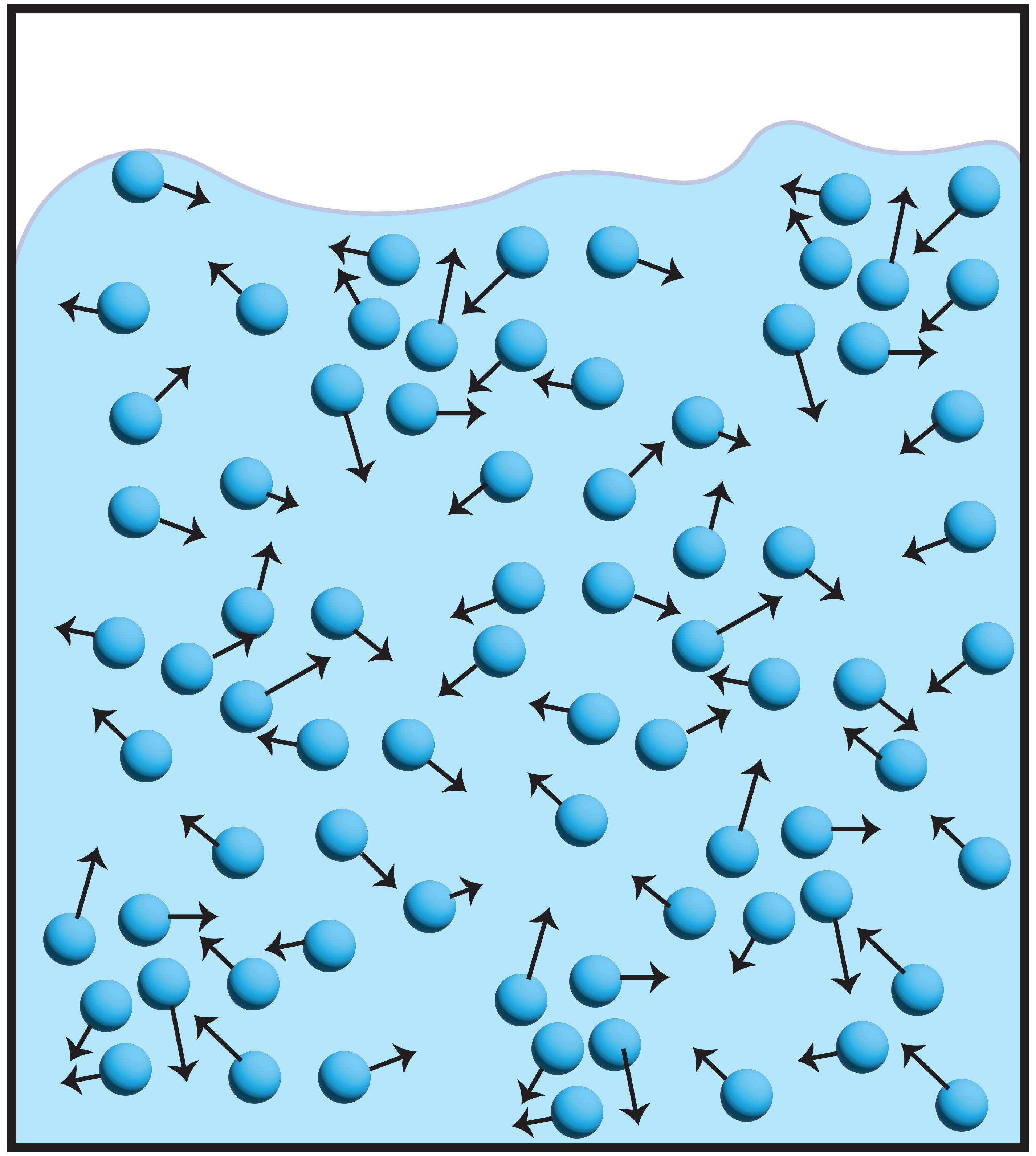
Stochastic Different equations (SDEs)



average trajectory

$\mathbf{x}_{t+1} = \mathbf{x}_t$ + drift + randomness

jiggle

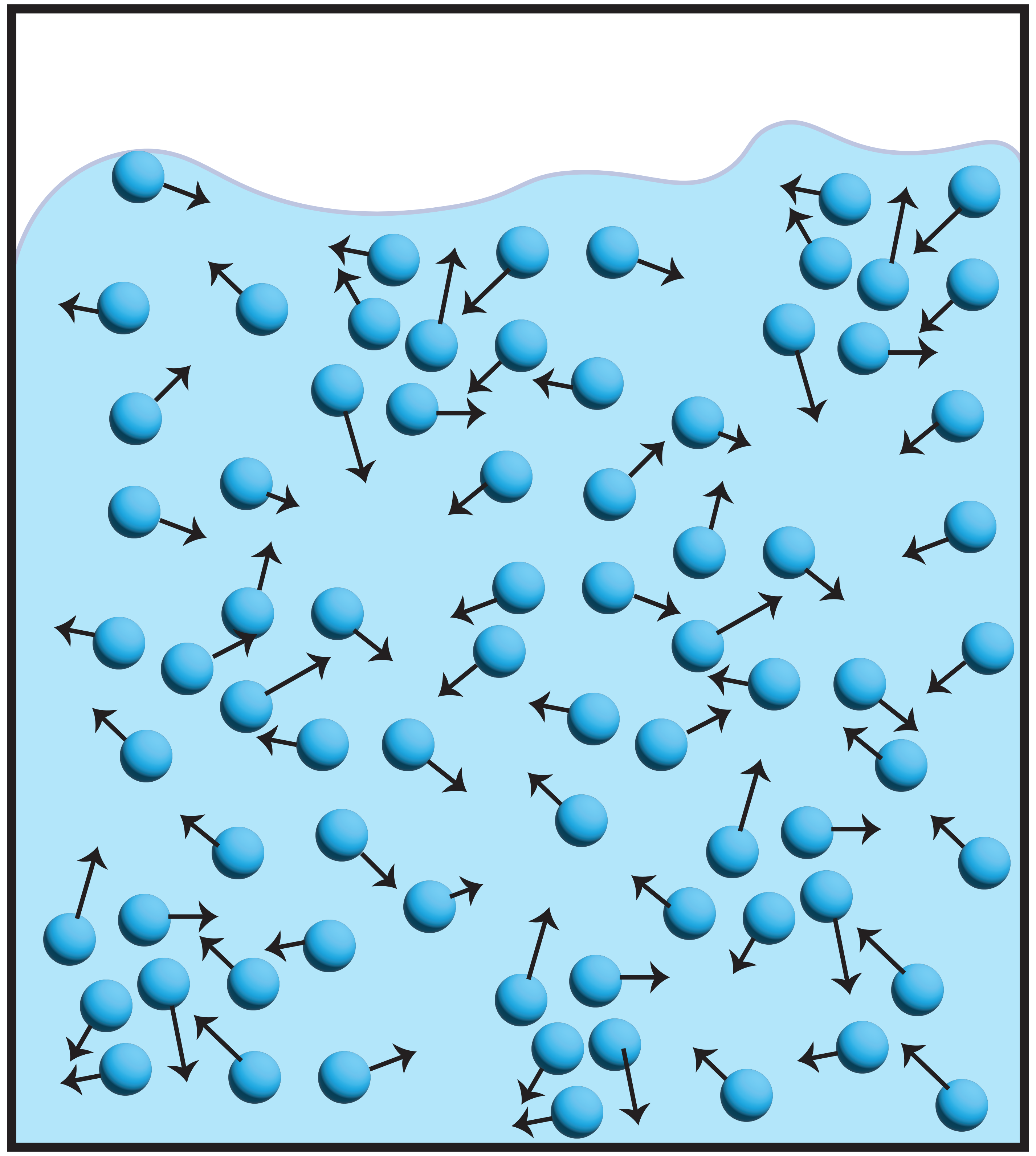


Stochastic Different equations (SDEs)

average trajectory

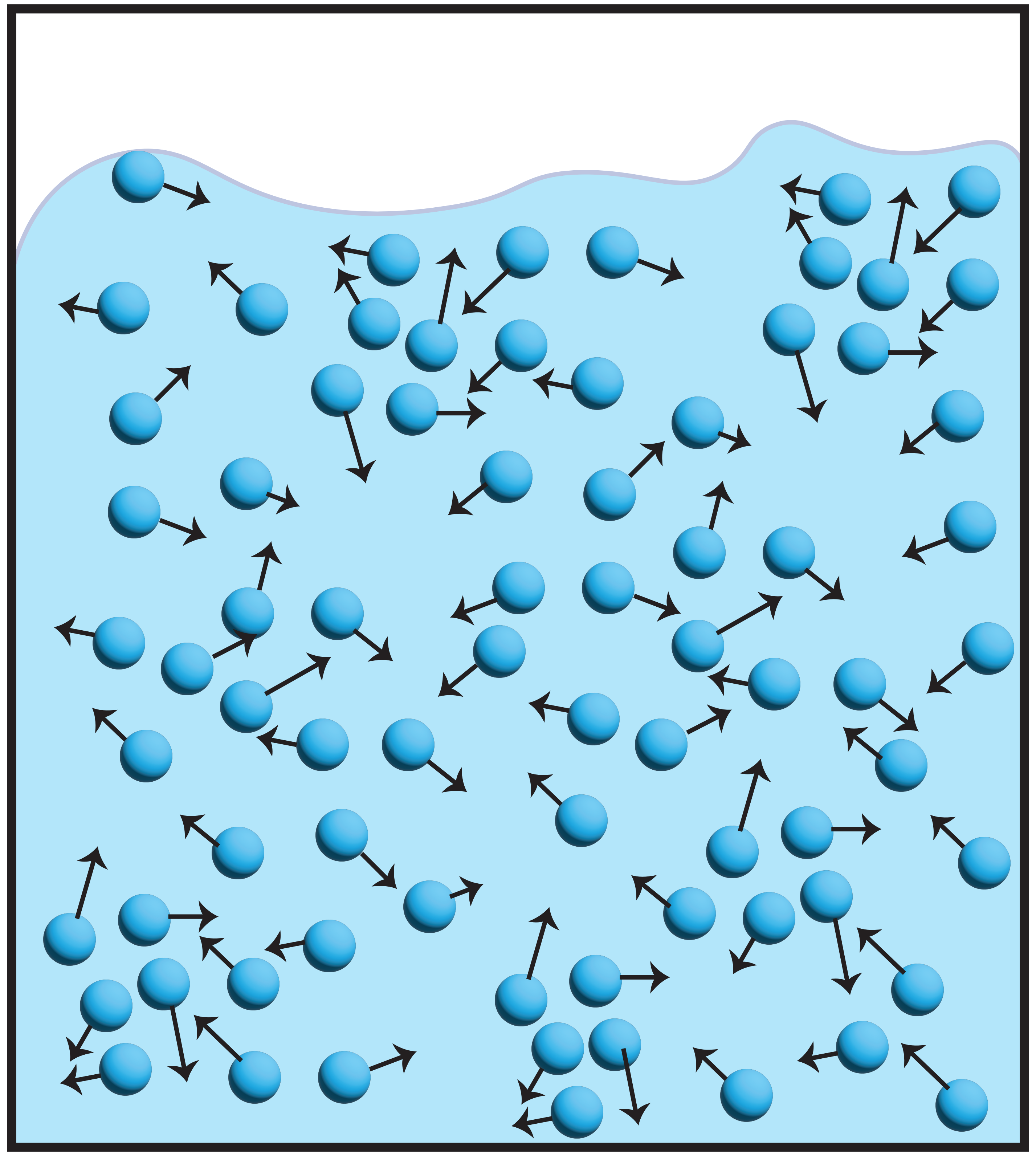
$\mathbf{x}_{t+1} = \mathbf{x}_t$ + drift + randomness

jiggle



Stochastic Different equations (SDEs)

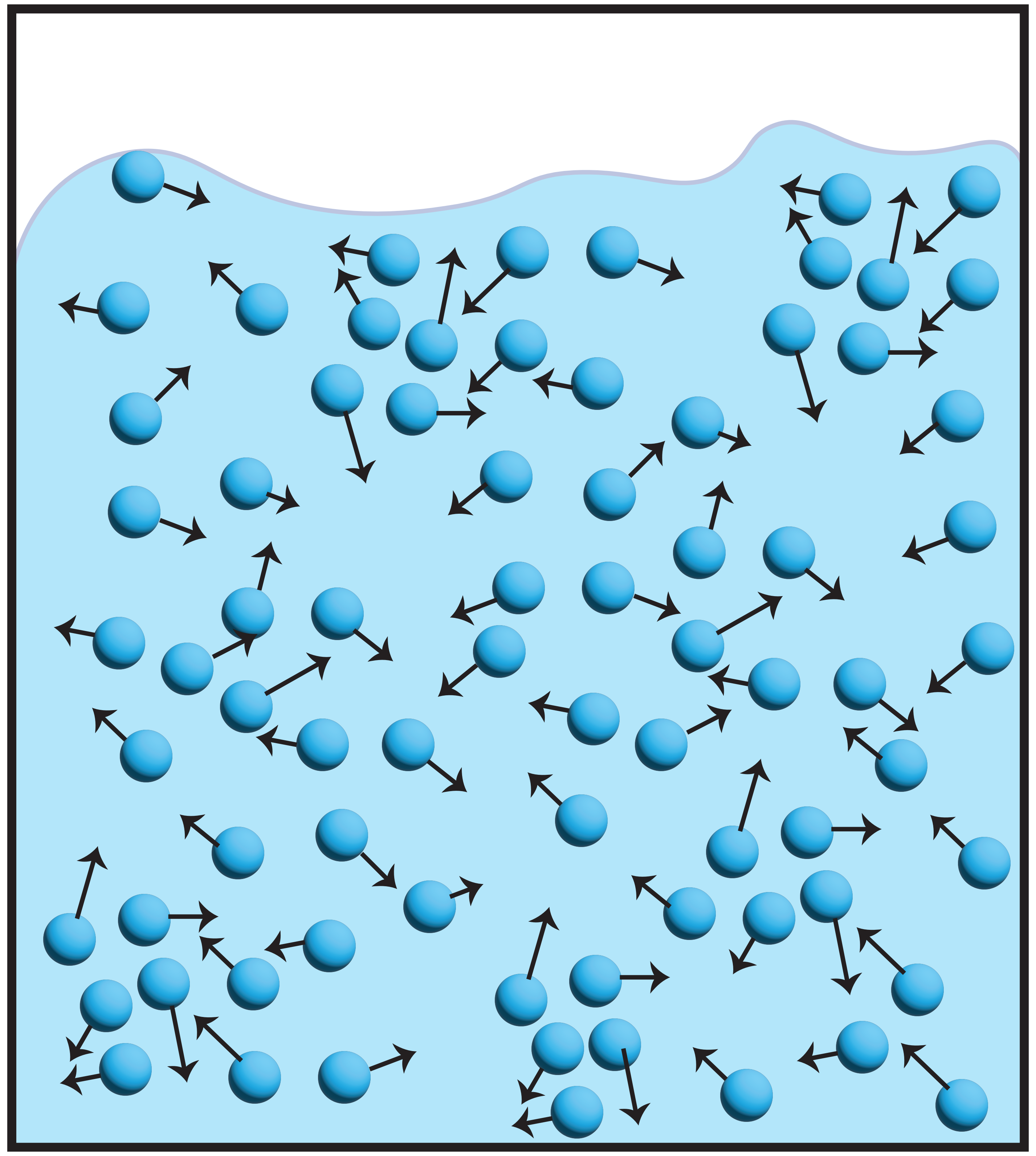
$\mathbf{x}_{t+1} = \mathbf{x}_t$ + drift + randomness



Stochastic Different equations (SDEs)

$$d\mathbf{x}_t = \mu(\mathbf{x}_t, t)dt + \sigma(\mathbf{x}_t, t)dW_t$$

$\mathbf{x}_{t+1} = \mathbf{x}_t$ + drift + randomness

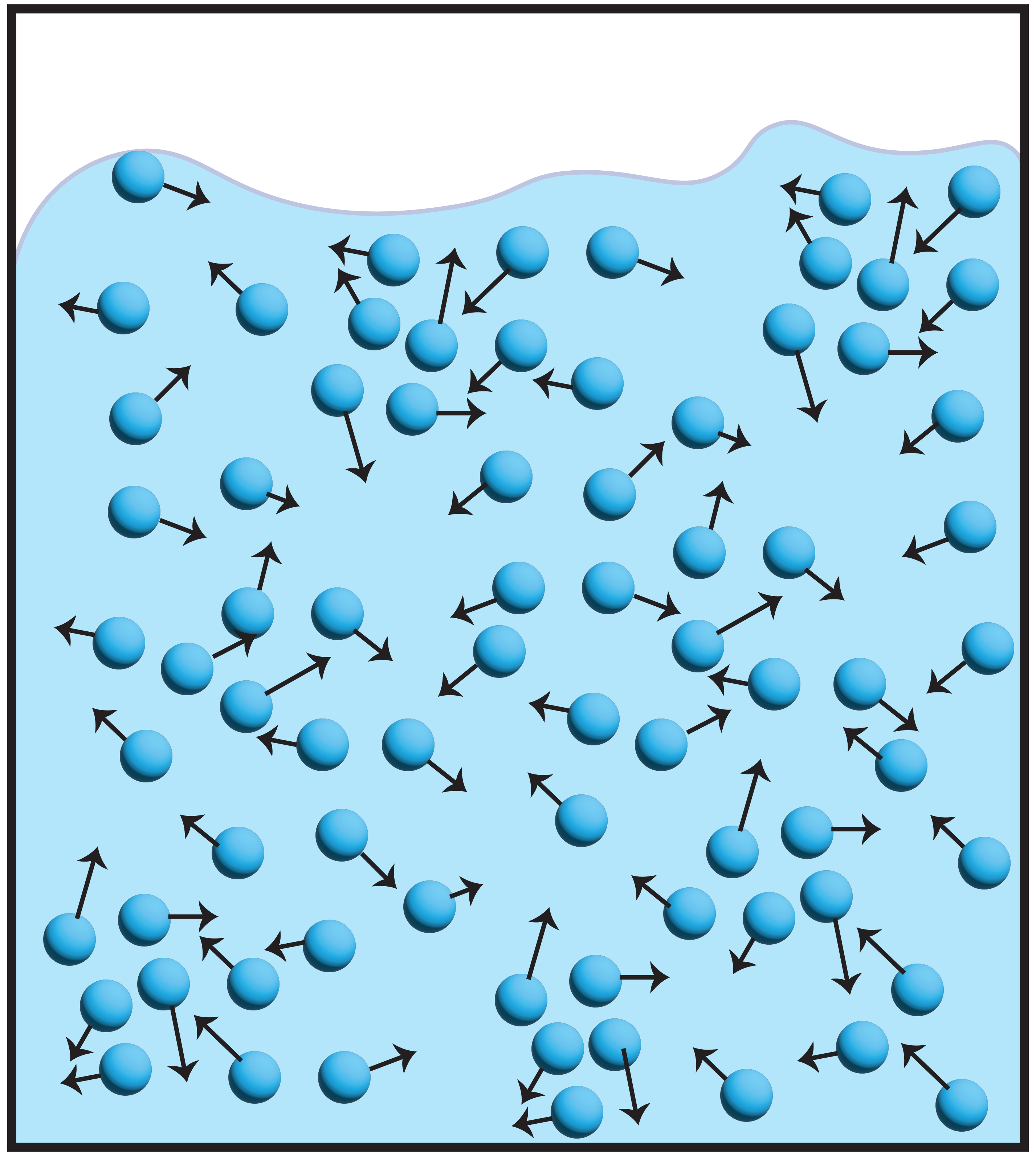


Stochastic Different equations (SDEs)

average trajectory

$$d\mathbf{x}_t = \mu(\mathbf{x}_t, t)dt + \sigma(\mathbf{x}_t, t)dW_t$$

$\mathbf{x}_{t+1} = \mathbf{x}_t$ + drift + randomness



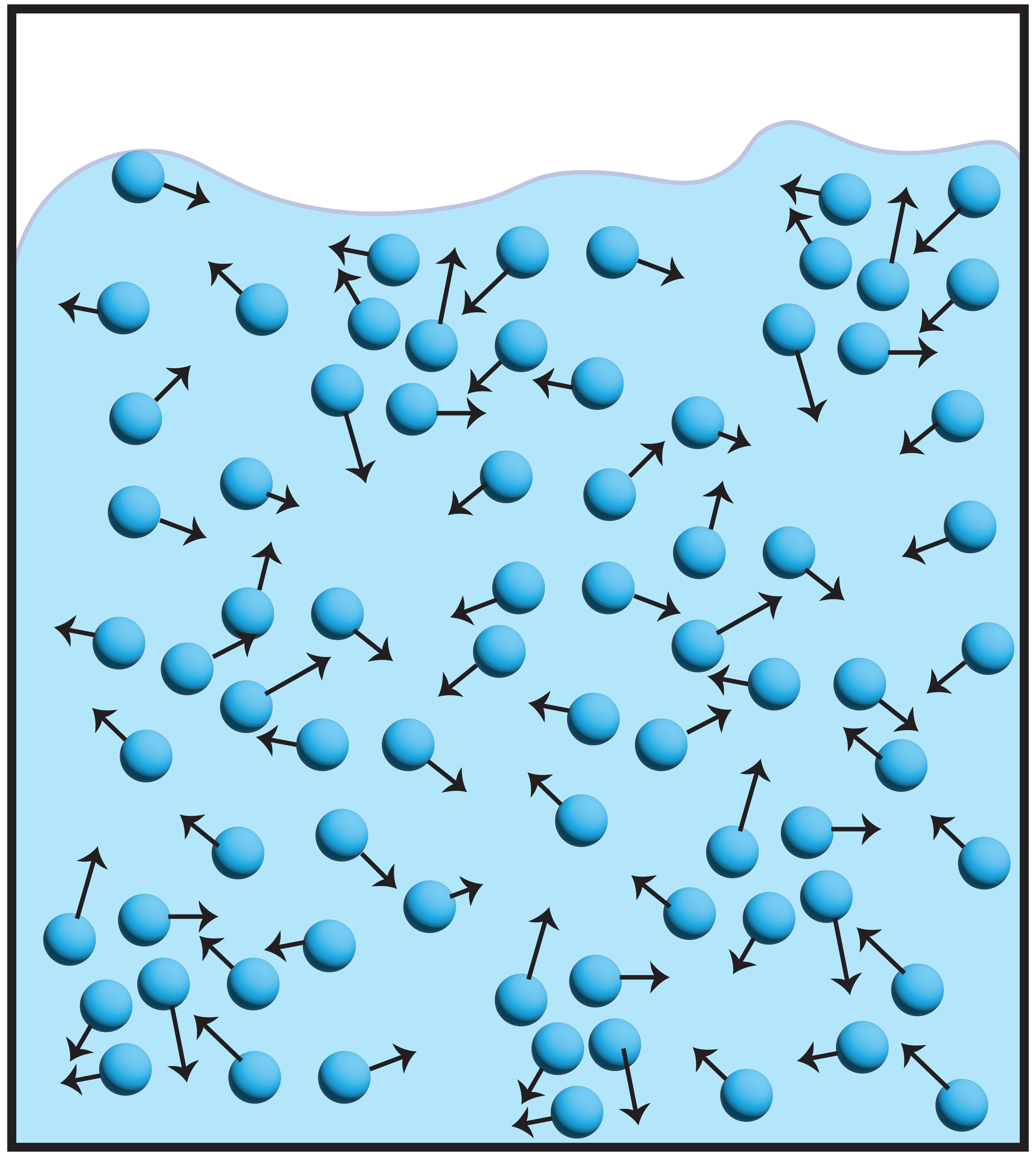
Stochastic Different equations (SDEs)

average trajectory

$$d\mathbf{x}_t = \mu(\mathbf{x}_t, t)dt + \sigma(\mathbf{x}_t, t)dW_t$$

jiggle

$\mathbf{x}_{t+1} = \mathbf{x}_t$ + drift + randomness



Stochastic Different equations (SDEs)

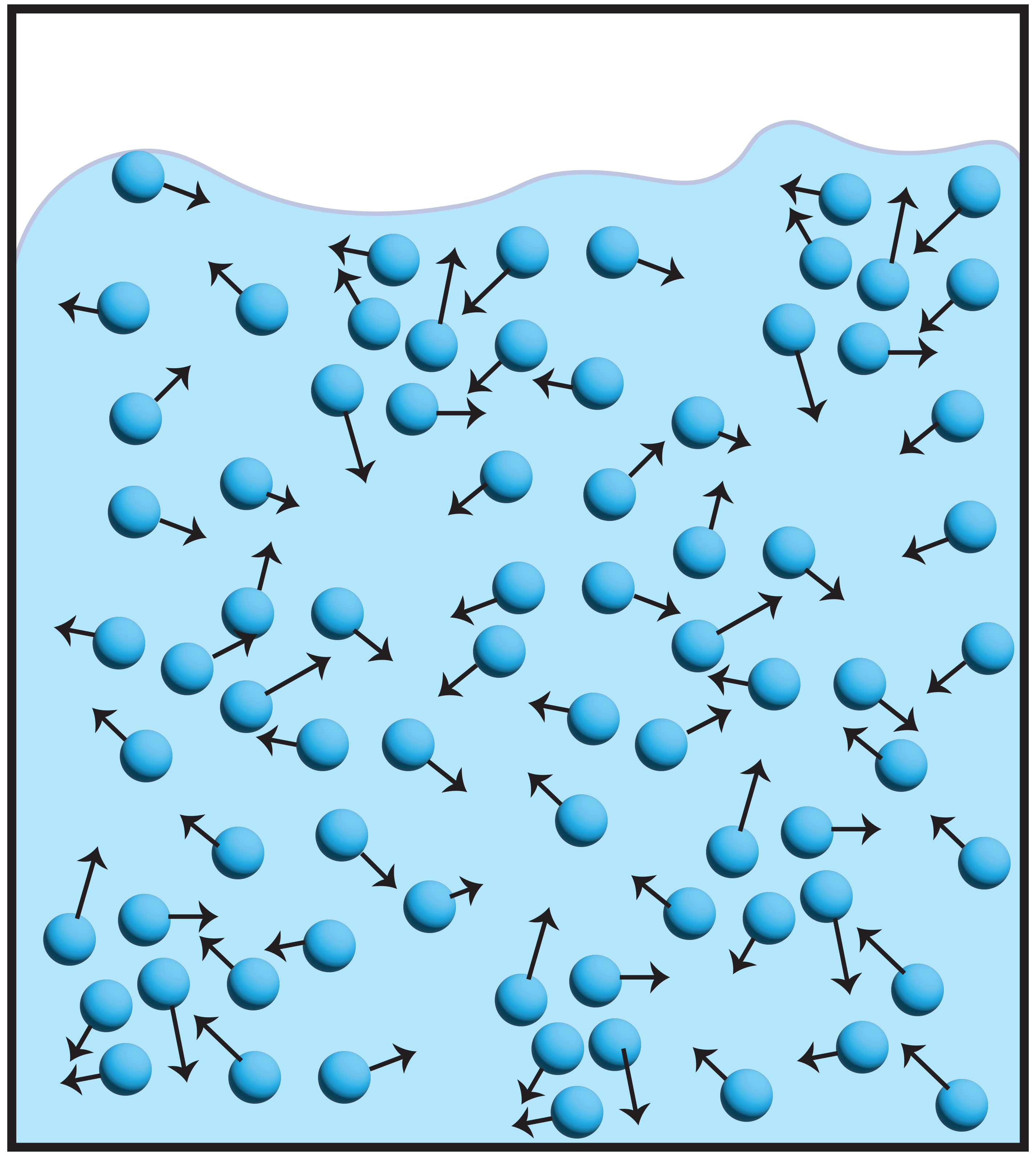
average trajectory

$$d\mathbf{x}_t = \mu(\mathbf{x}_t, t)dt + \sigma(\mathbf{x}_t, t)dW_t$$

jiggle

drift

randomness



Stochastic Different equations (SDEs)

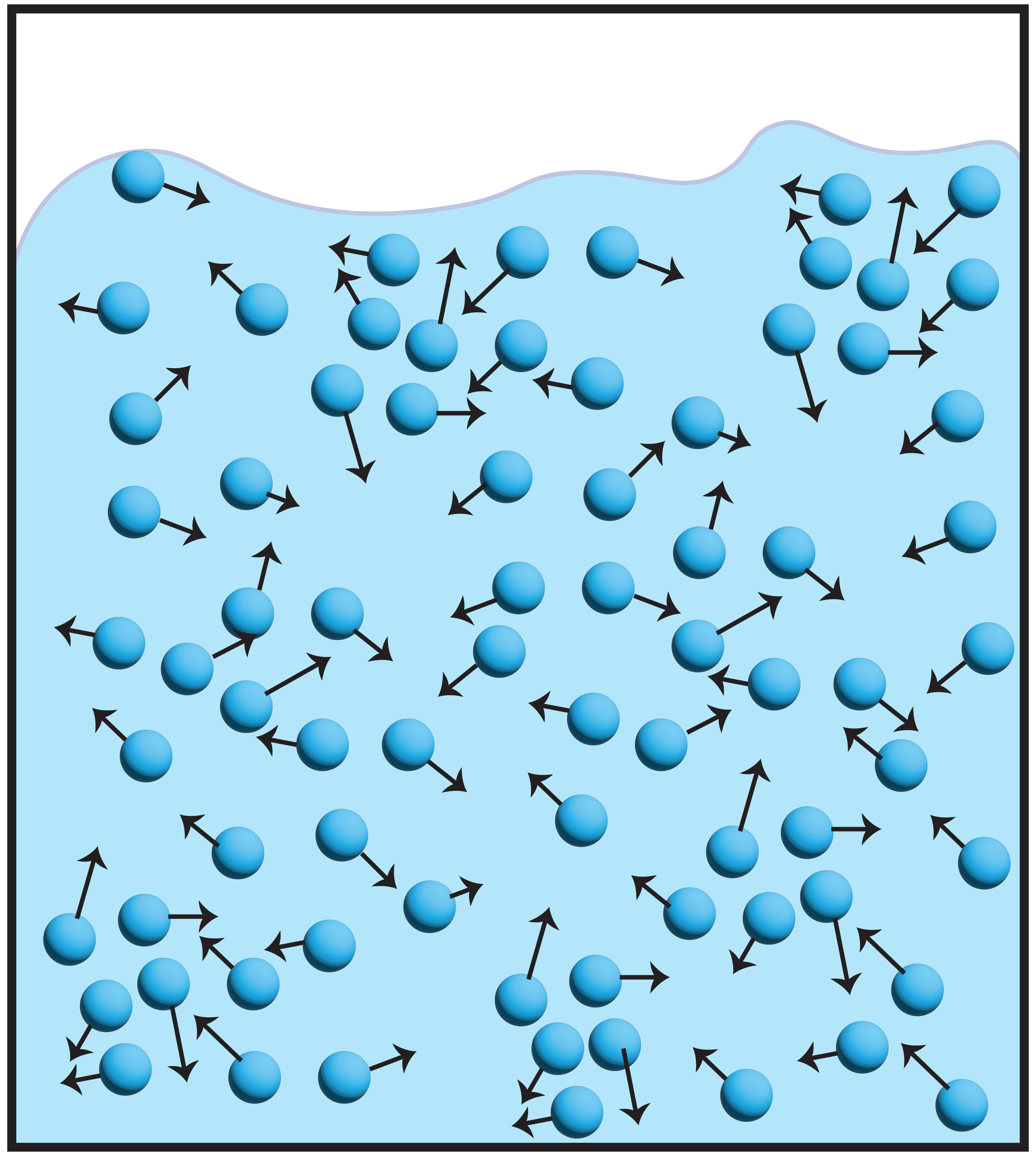
average trajectory

$$d\mathbf{x}_t = \mu(\mathbf{x}_t, t)dt + \sigma(\mathbf{x}_t, t)dW_t$$

jiggle

drift

randomness



Brownian motion: simplest form of SDE

Brownian motion: simplest form of SDE

$$d\mathbf{x}_t = \mu(\mathbf{x}_t, t)dt + \sigma(\mathbf{x}_t, t)dW_t$$

$$\boxed{\text{drift } = 0}$$

Brownian motion: simplest form of SDE

$$d\mathbf{x}_t = \ \sigma(\mathbf{x}_t, t) dW_t$$

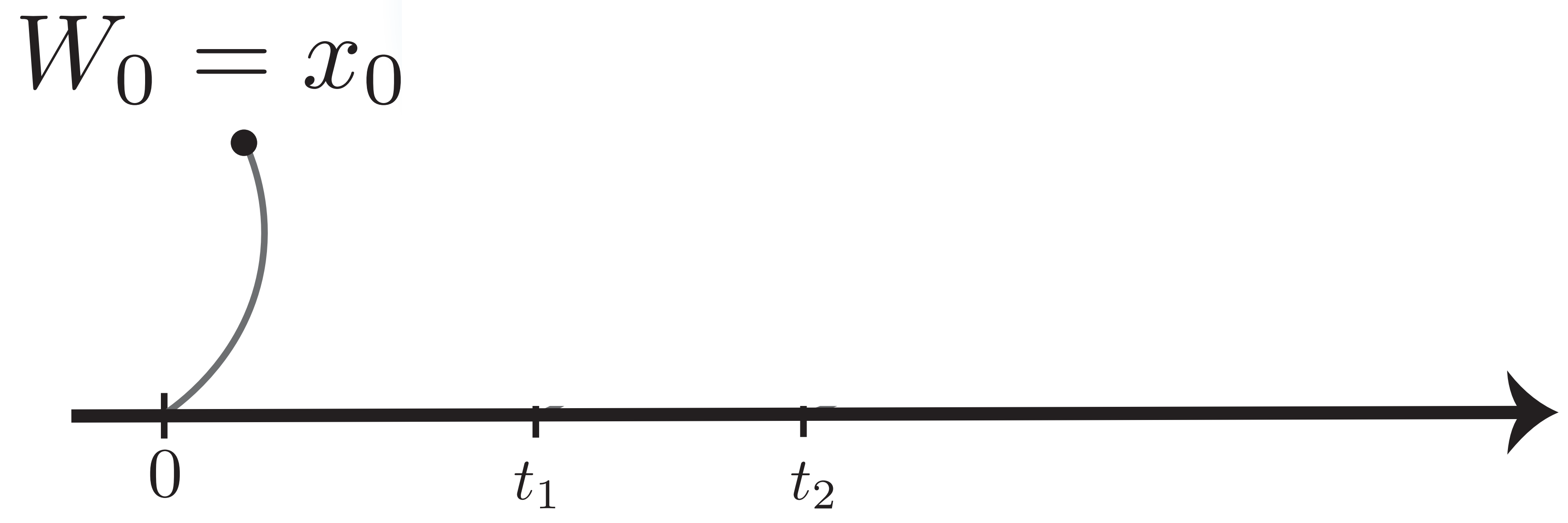


Illustration inspired from Keenan Crane notes

Brownian motion: simplest form of SDE

$$d\mathbf{x}_t = \sigma(\mathbf{x}_t, t)dW_t$$

Gaussian noise

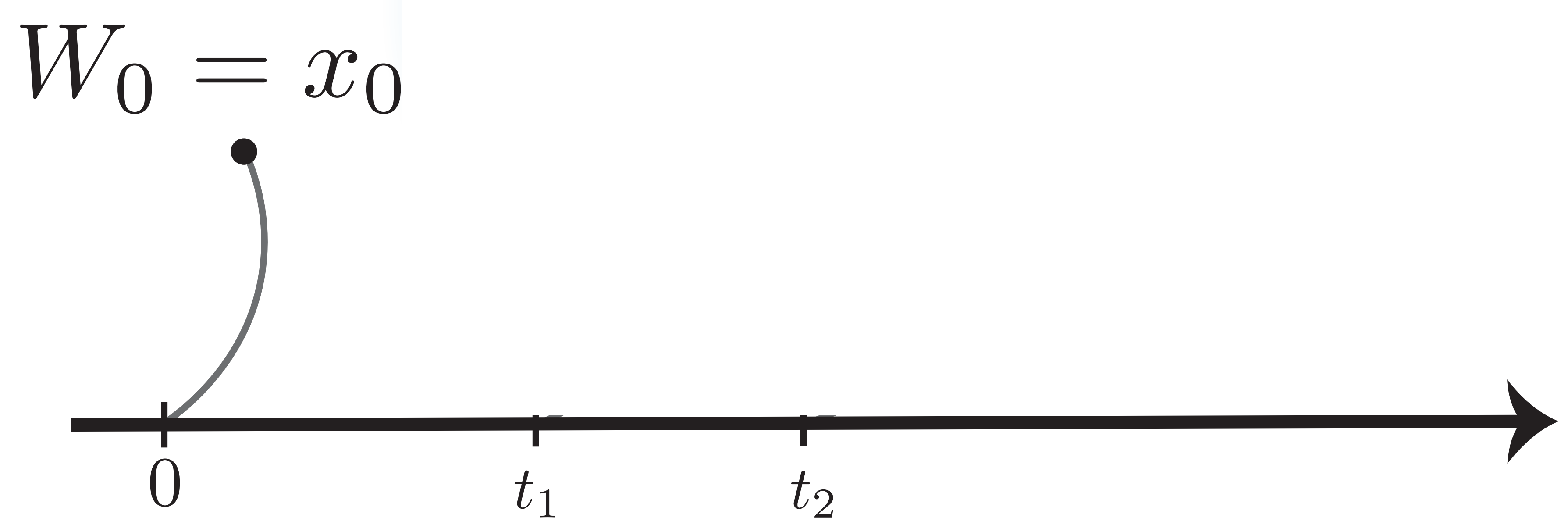


Illustration inspired from Keenan Crane notes

Brownian motion: simplest form of SDE

$$d\mathbf{x}_t = \sigma(\mathbf{x}_t, t)dW_t$$

Gaussian noise

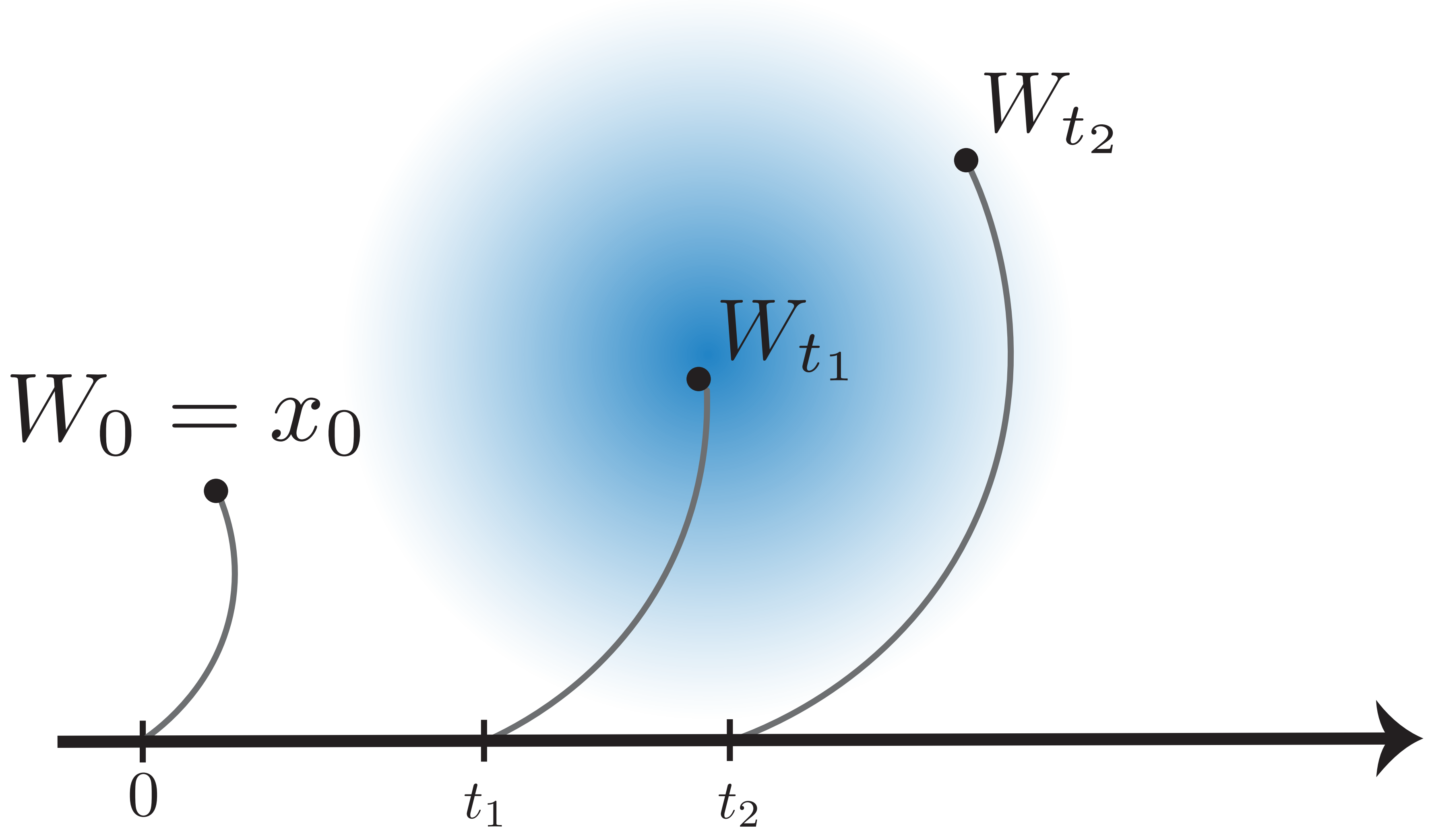


Illustration inspired from Keenan Crane notes

Brownian motion: simplest form of SDE

$$d\mathbf{x}_t = \sigma(\mathbf{x}_t, t) dW_t$$

Gaussian noise

W_{t_1} has independent samples

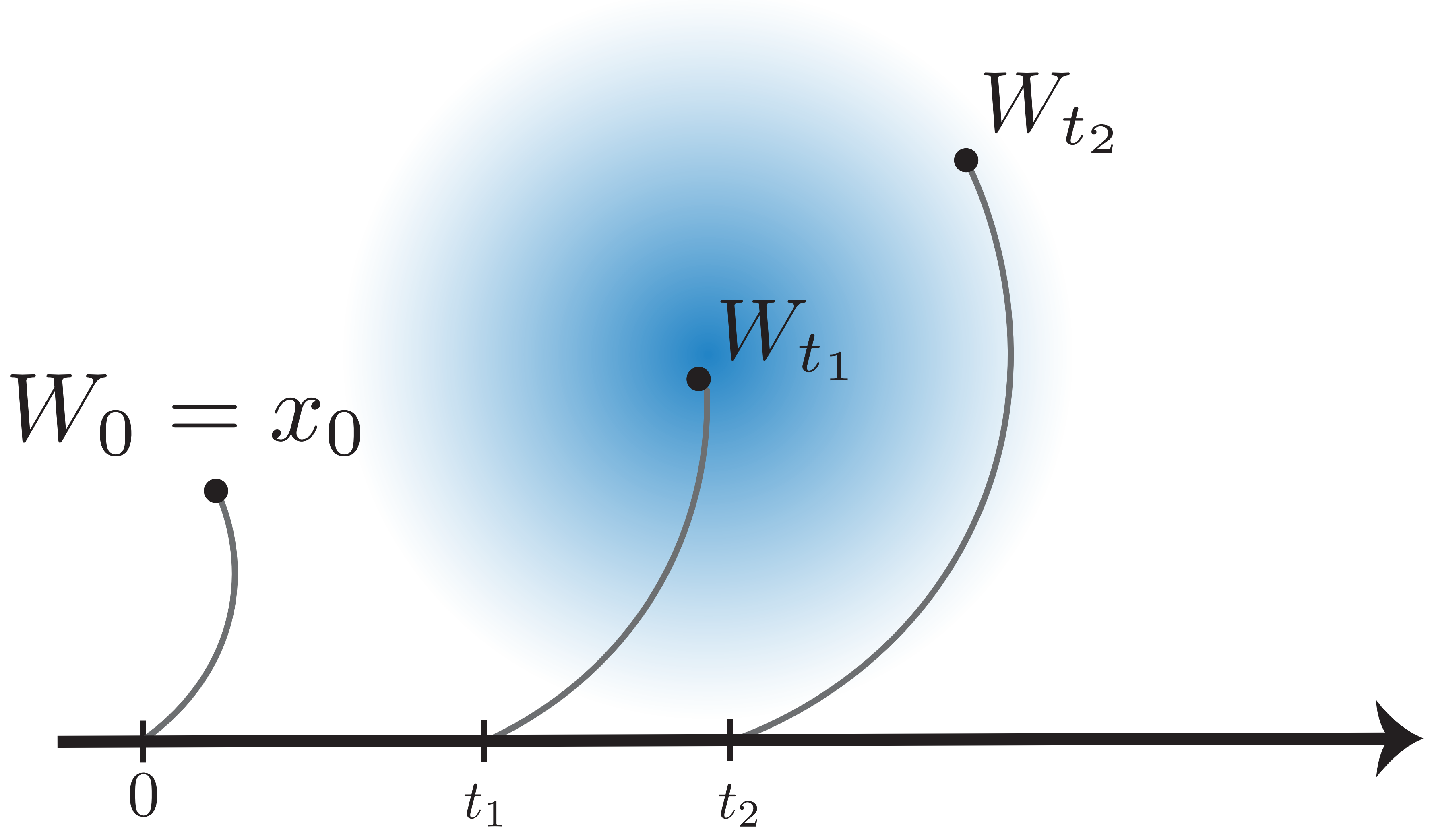
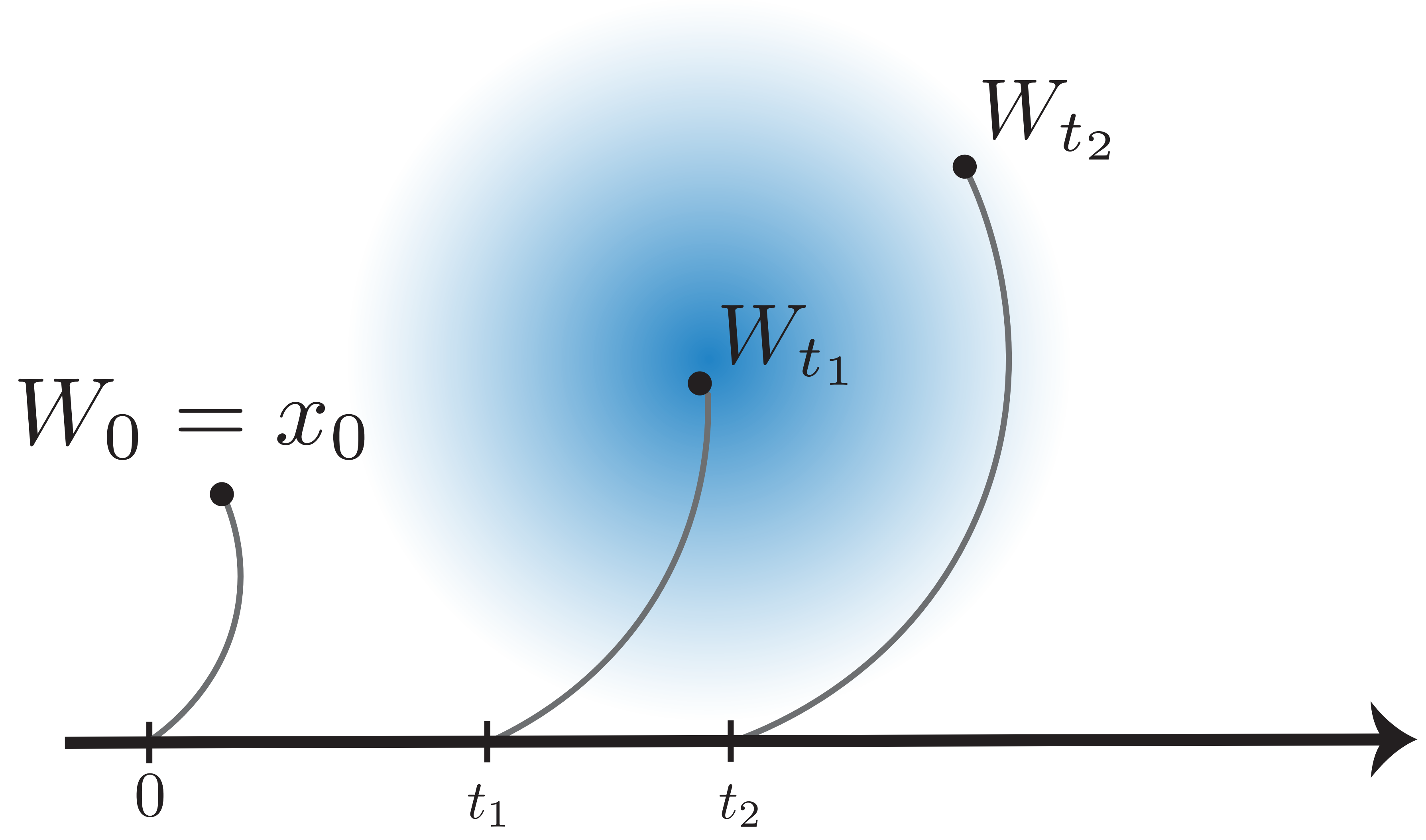


Illustration inspired from Keenan Crane notes

Brownian motion: simplest form of SDE

$$d\mathbf{x}_t = \sigma(\mathbf{x}_t, t)dW_t$$

Gaussian noise



W_{t_1} has independent samples

$$W_{t_2} - W_{t_1} \sim \mathcal{N}(0, t_2 - t_1) \text{ for } 0 \leq t_1 < t_2$$

Illustration inspired from Keenan Crane notes

Discretizing Brownian motion

$$d\mathbf{x}_t = \quad \sigma(\mathbf{x}_t, t)dW_t$$

Discretizing Brownian motion

$$d\mathbf{x}_t = \quad \sigma(\mathbf{x}_t, t)dW_t$$

- Euler Maruyama method:

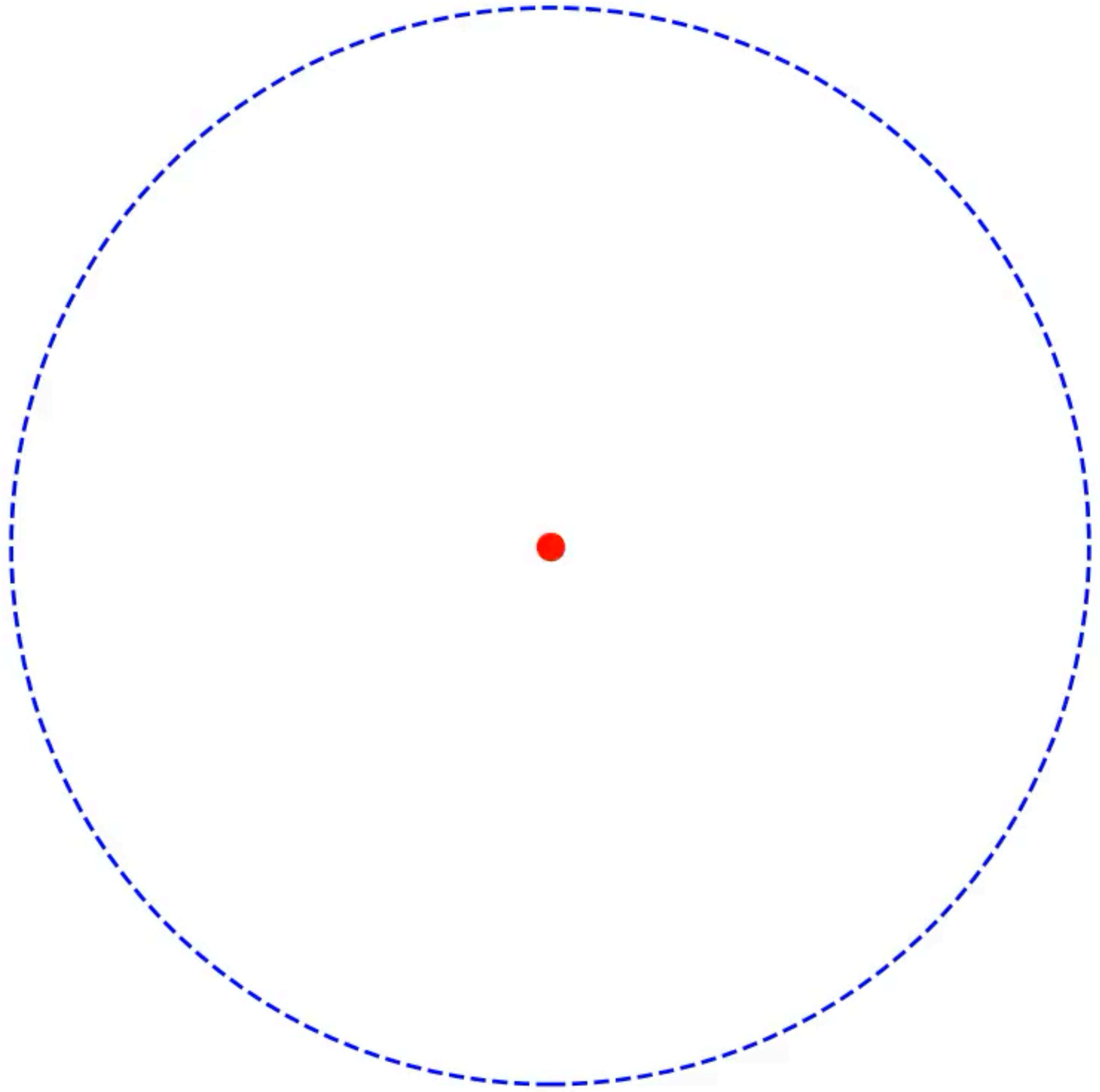
Discretizing Brownian motion

$$d\mathbf{x}_t = \quad \sigma(\mathbf{x}_t, t)dW_t$$

- Euler Maruyama method:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \xi \quad \text{where} \quad \xi \sim \mathcal{N}(0,1)$$

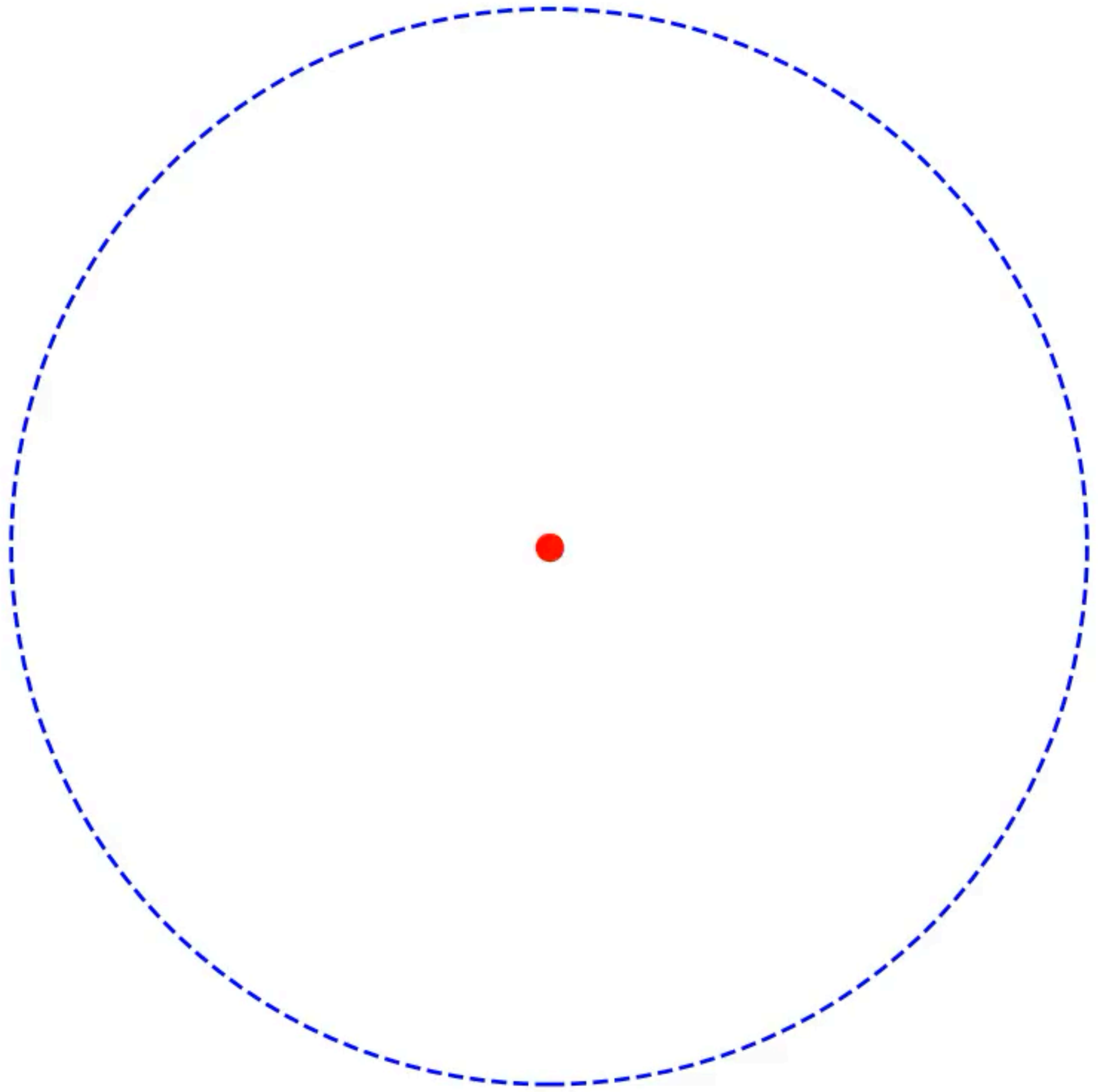
Discretized Brownian motion: Examples



Random walk in space

Given target distribution

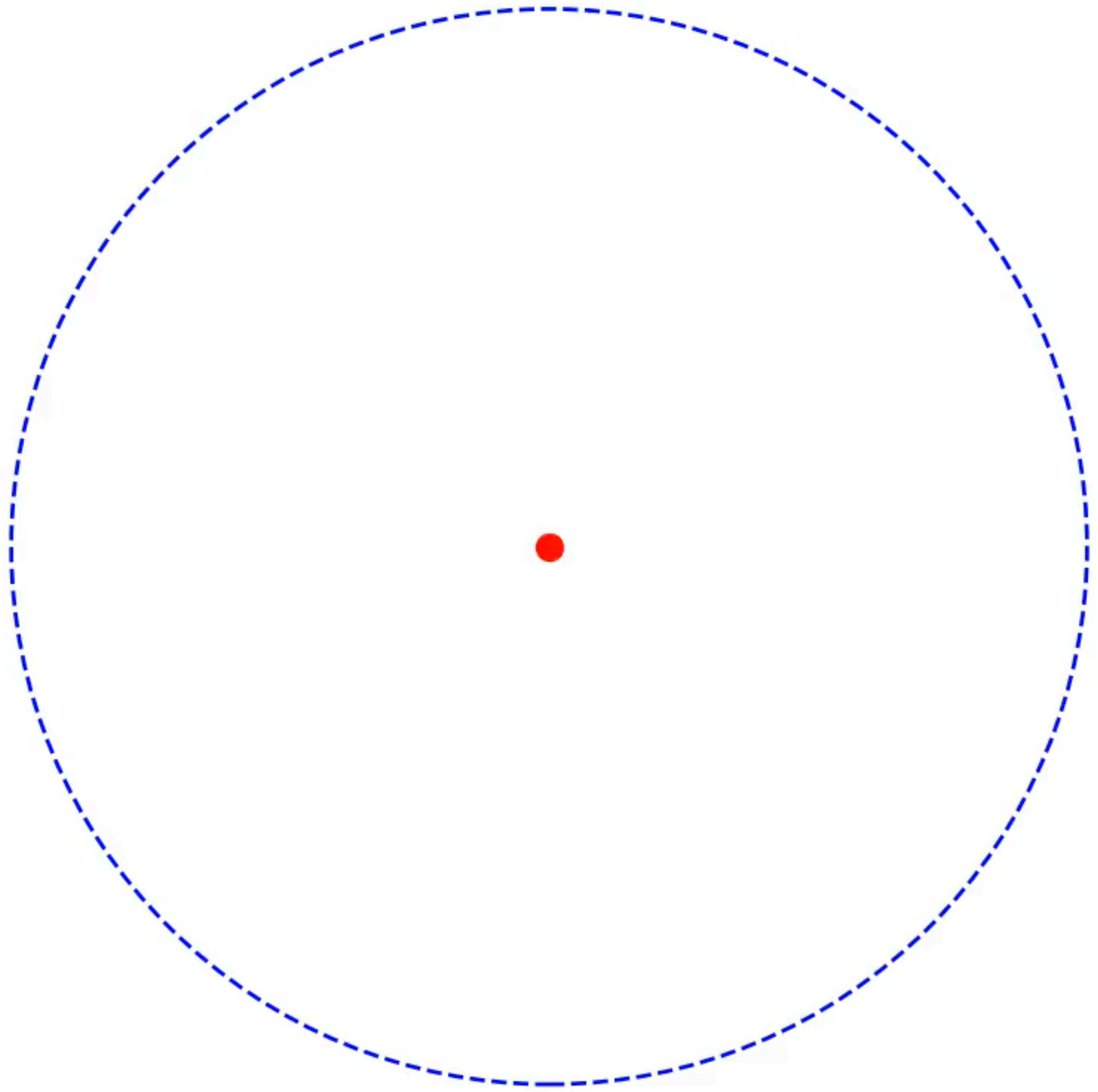
Discretized Brownian motion: Examples



Random walk in space

Given target distribution

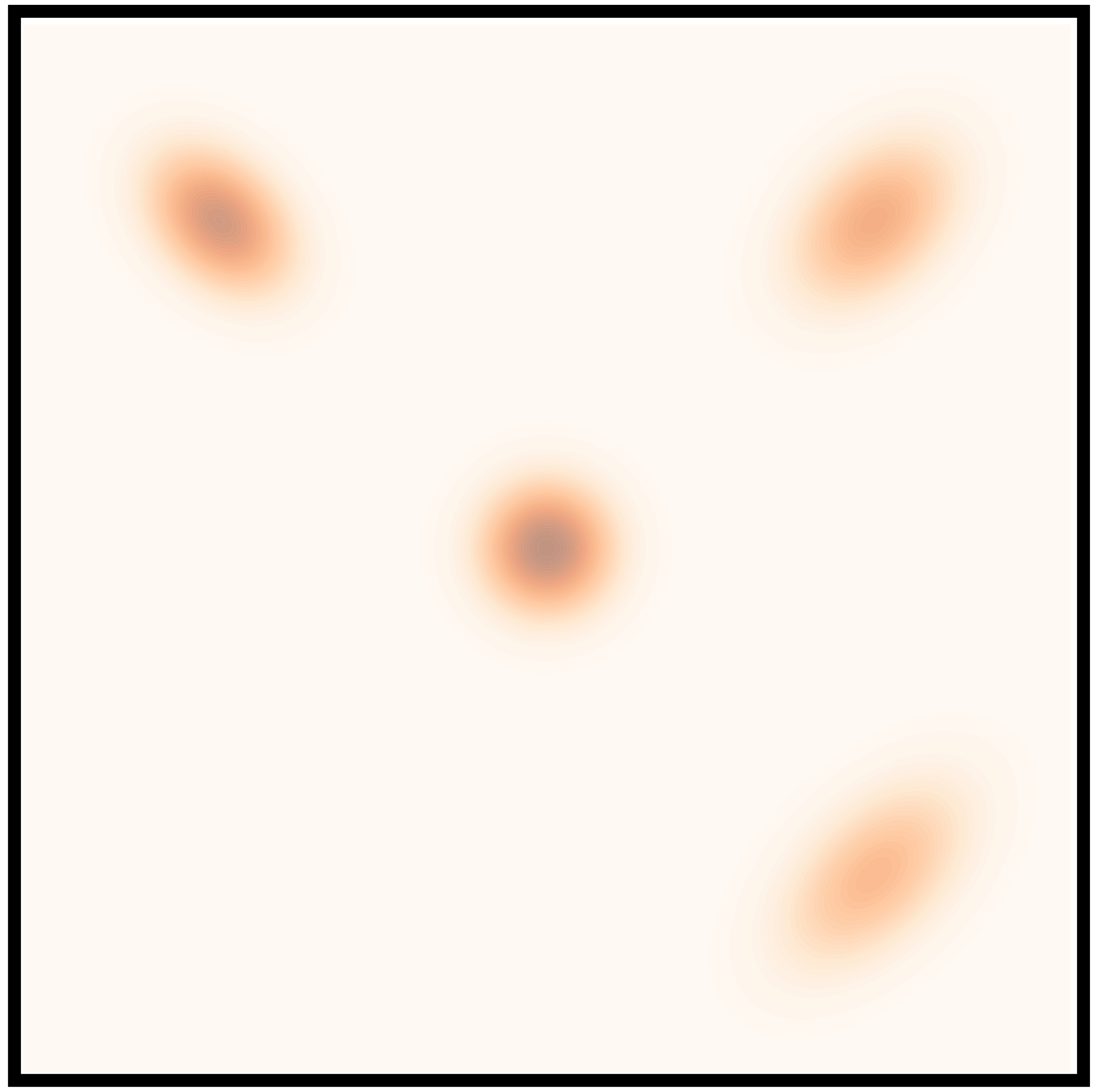
Discretized Brownian motion: Examples



Random walk in space

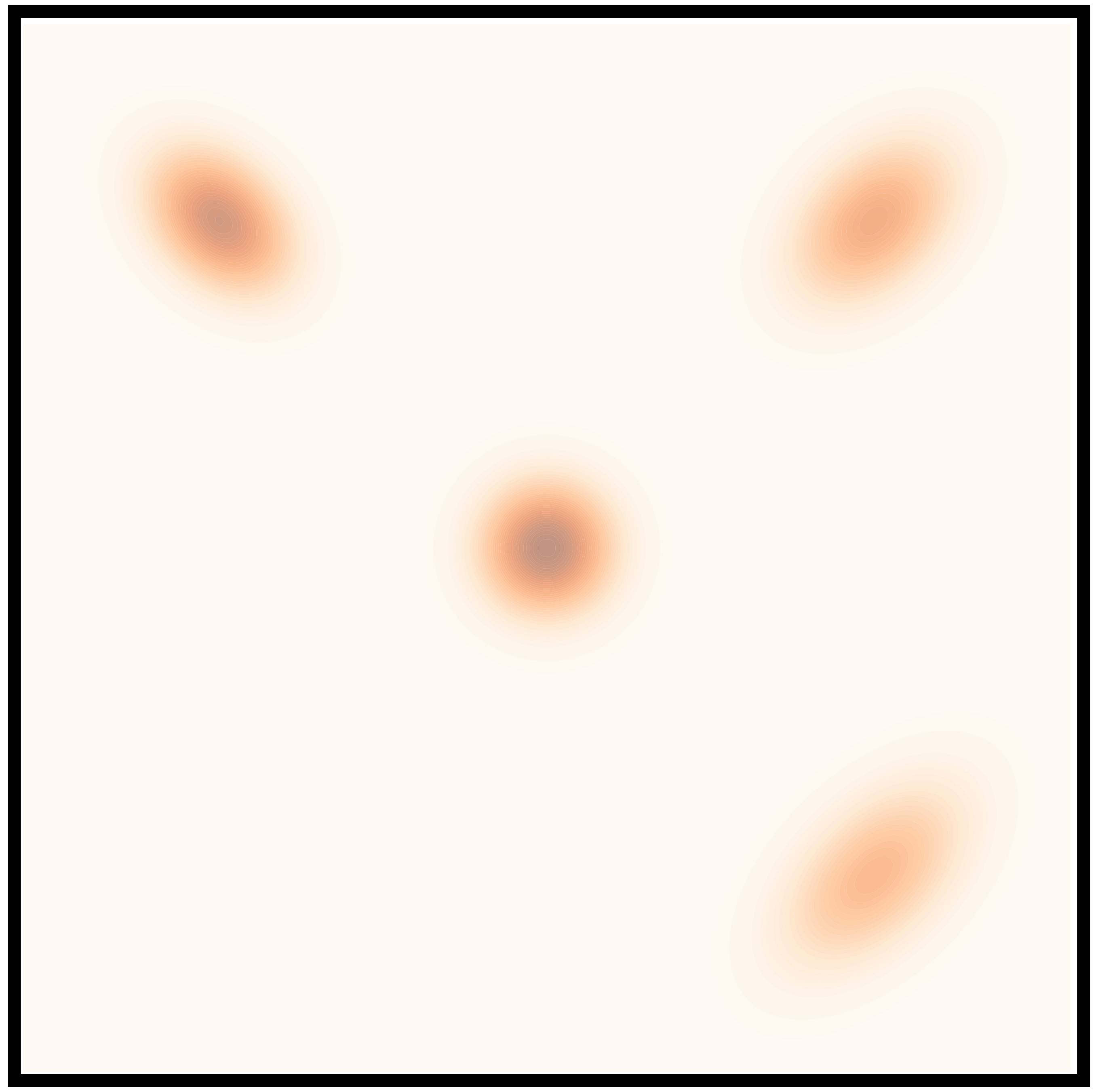
Given target distribution

Discretized Brownian motion: Examples

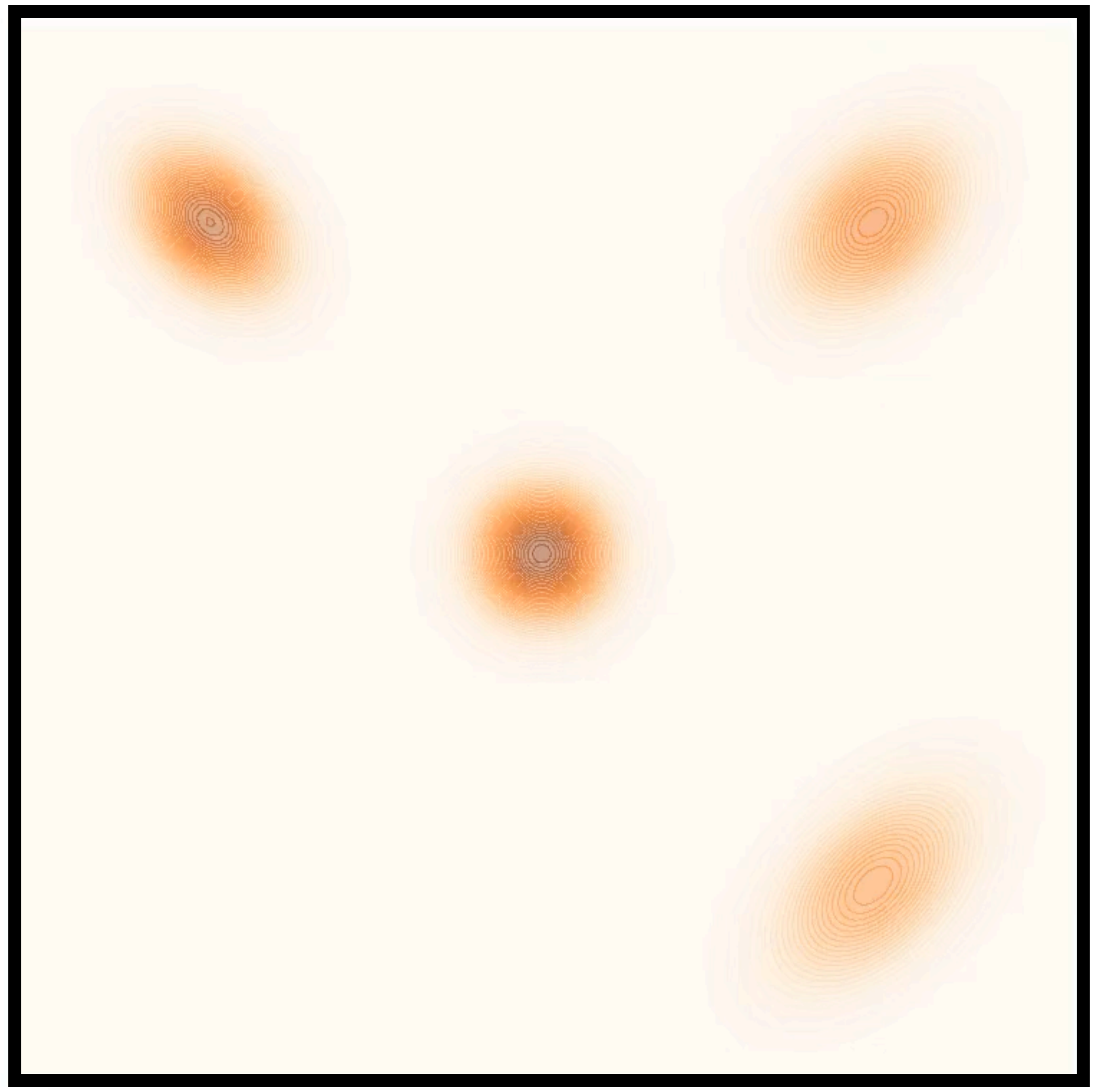


Target distribution

Discretized Brownian motion: Examples

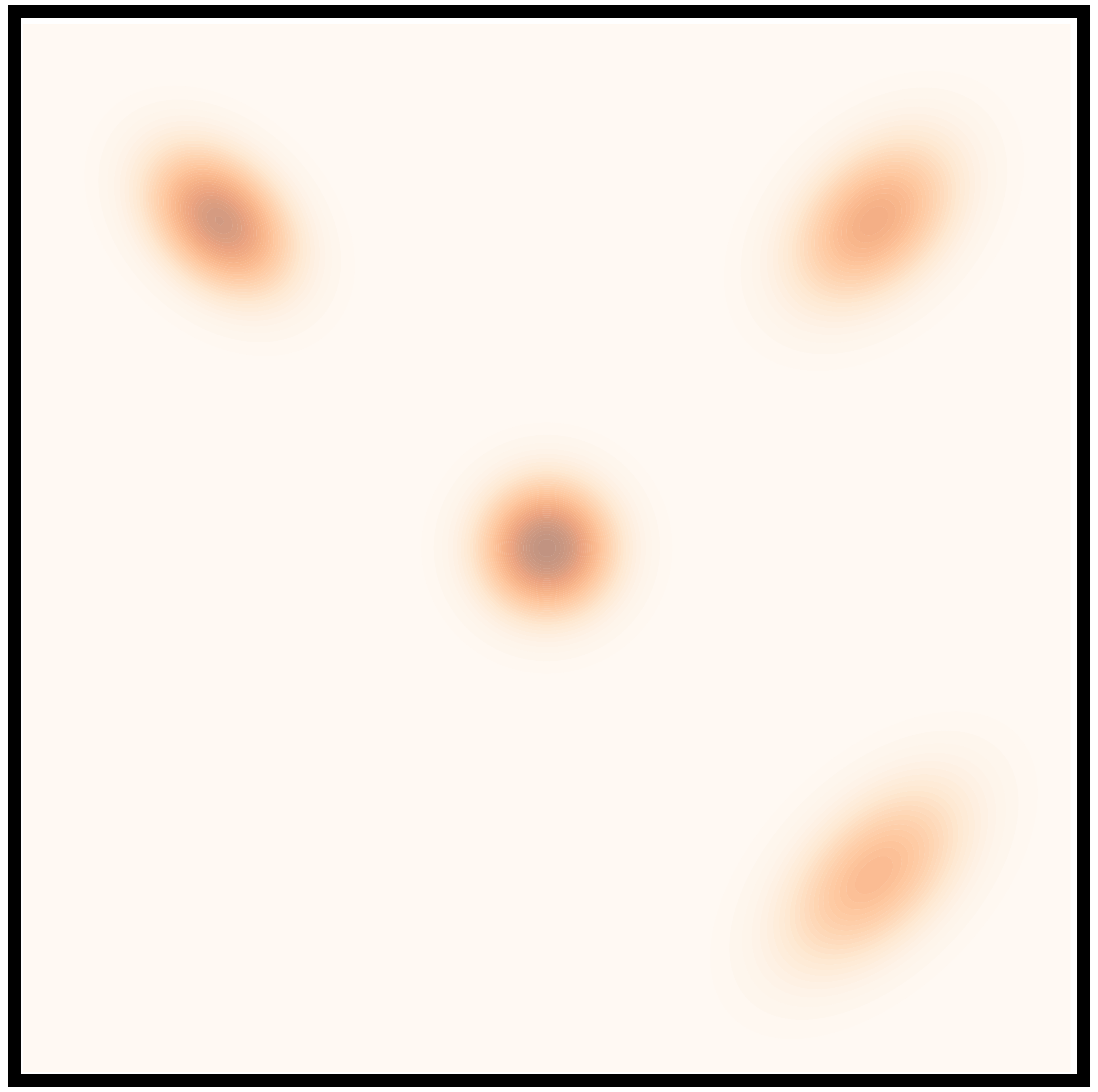


Target distribution

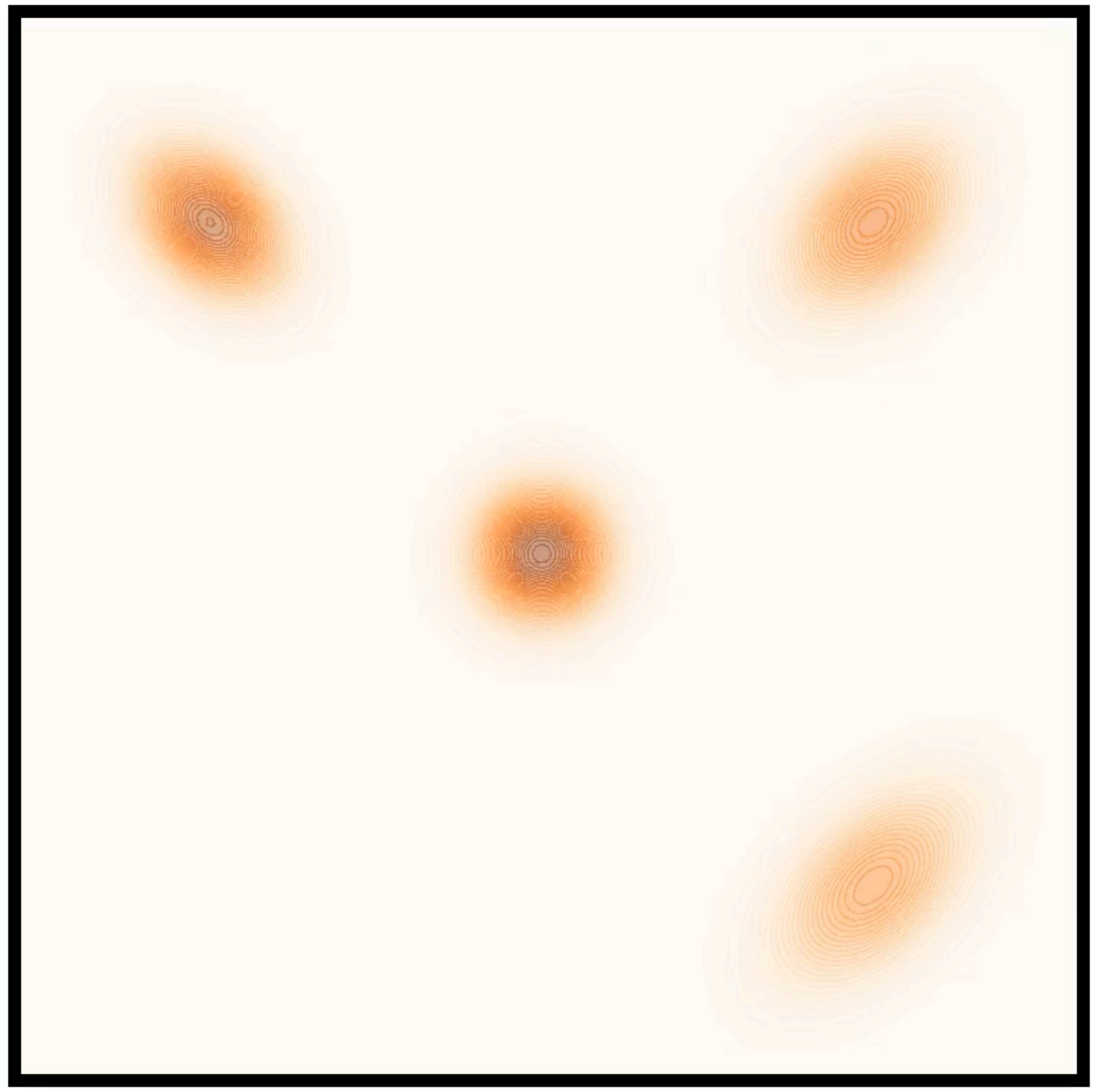


Random walk

Discretized Brownian motion: Examples

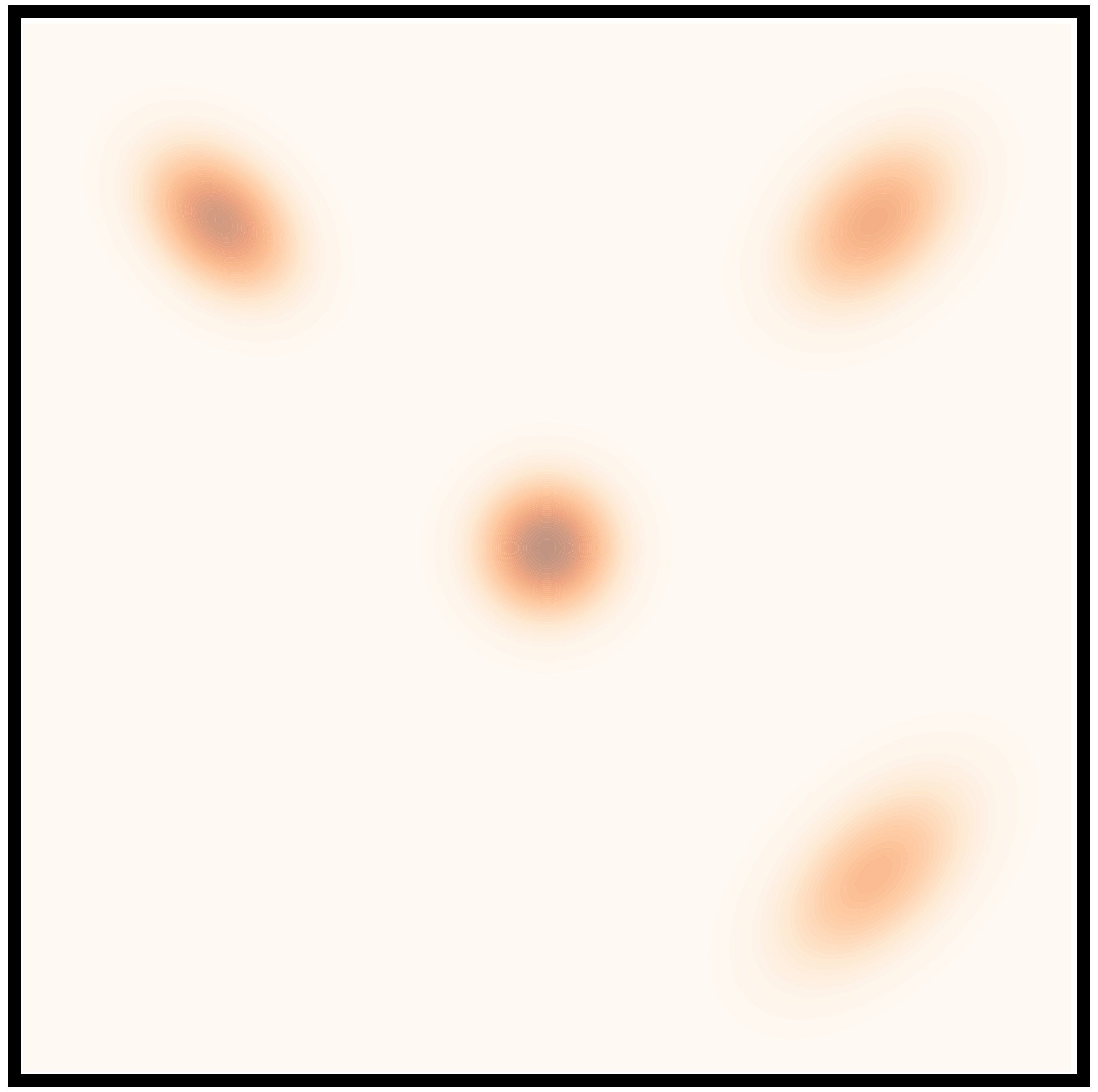


Target distribution

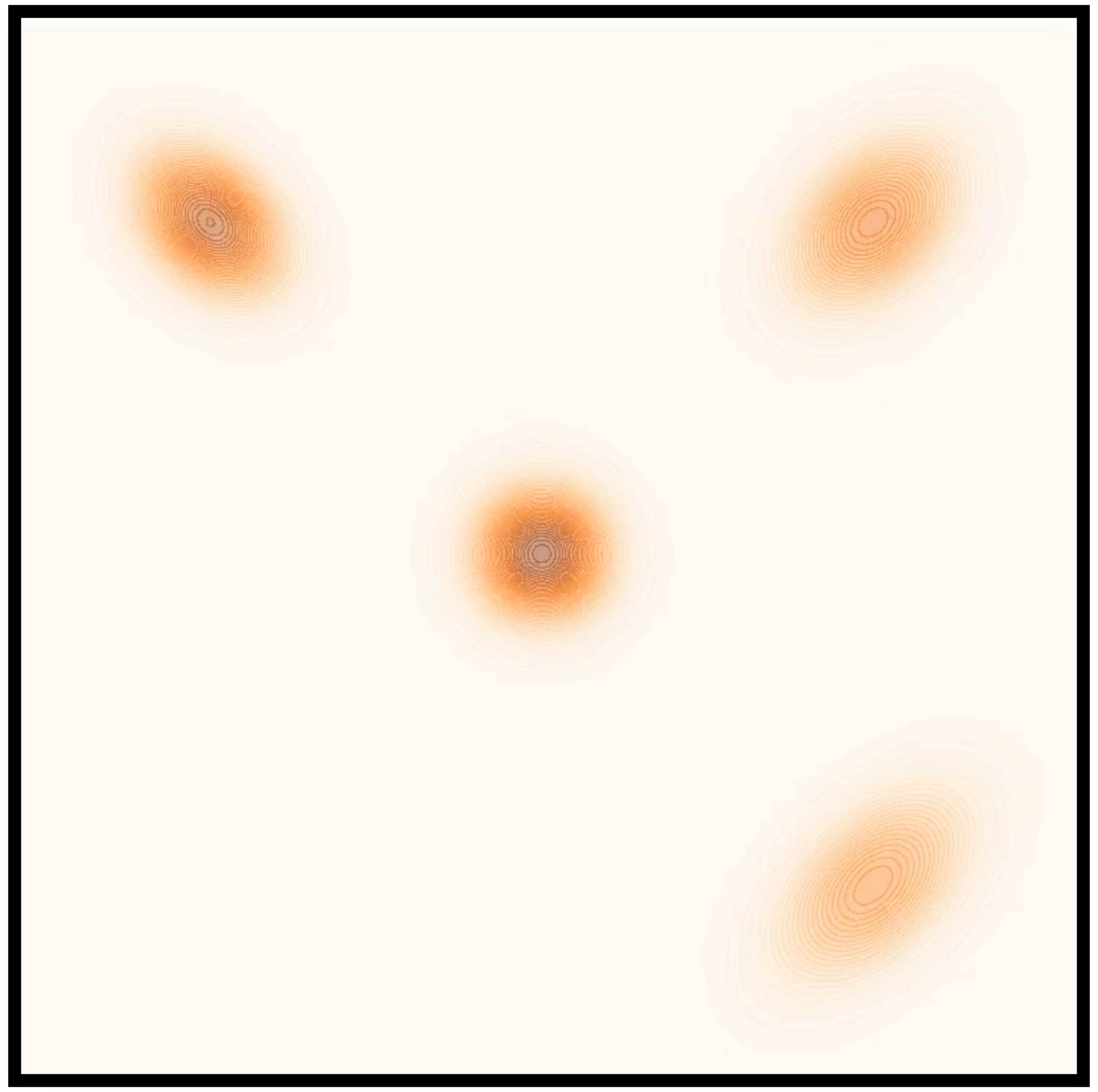


Random walk

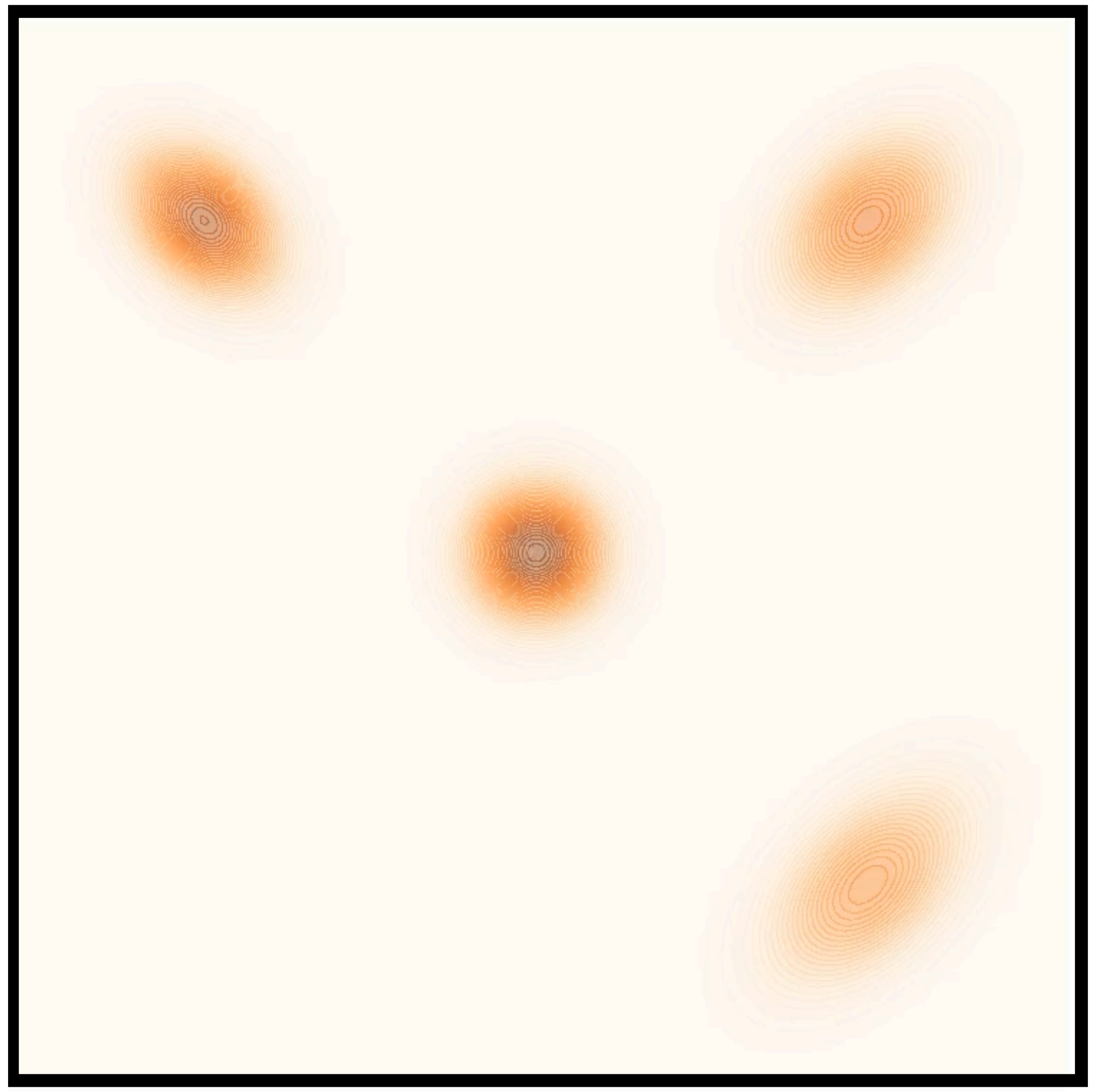
Discretized Brownian motion: Examples



Target distribution

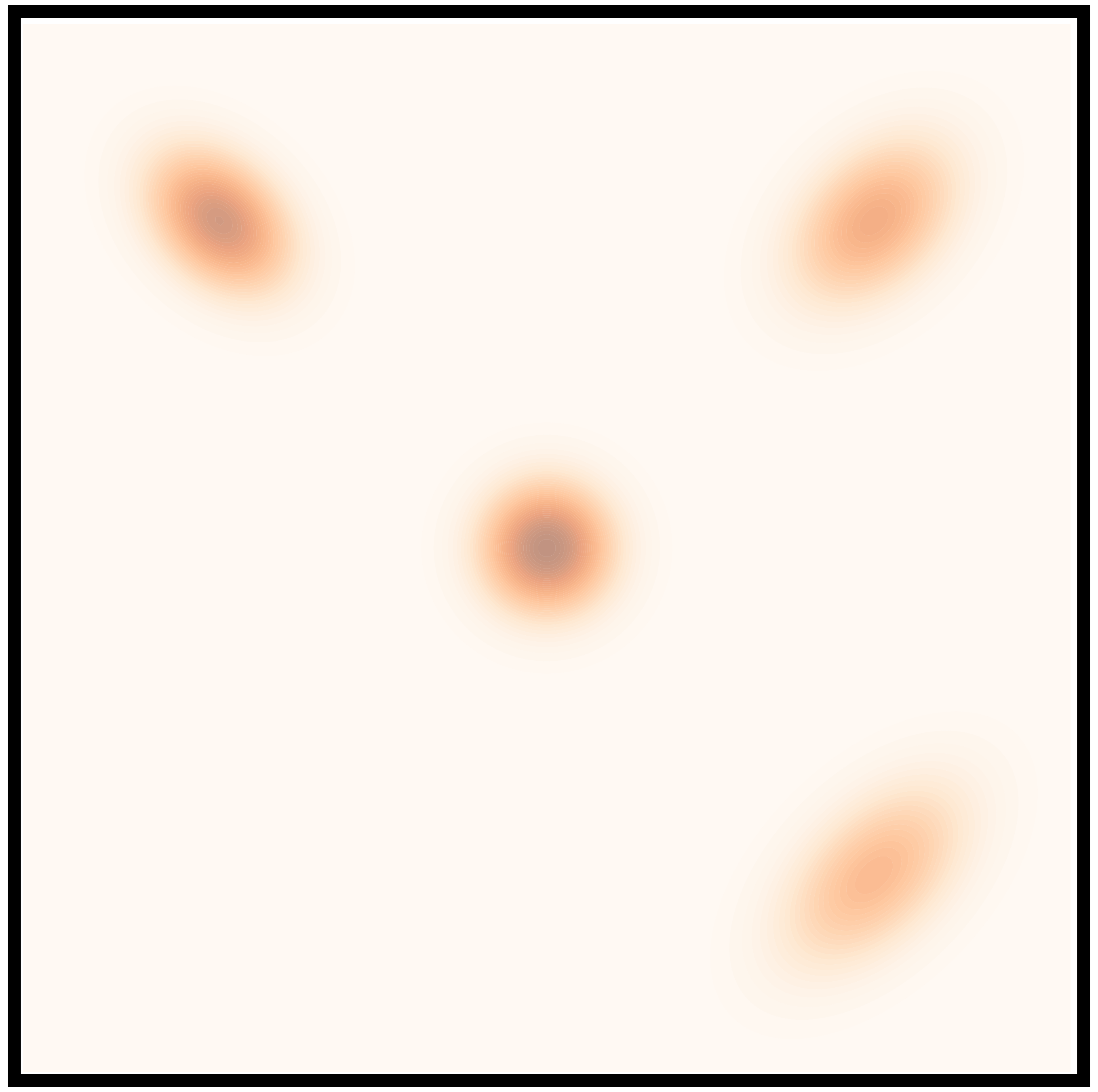


Random walk

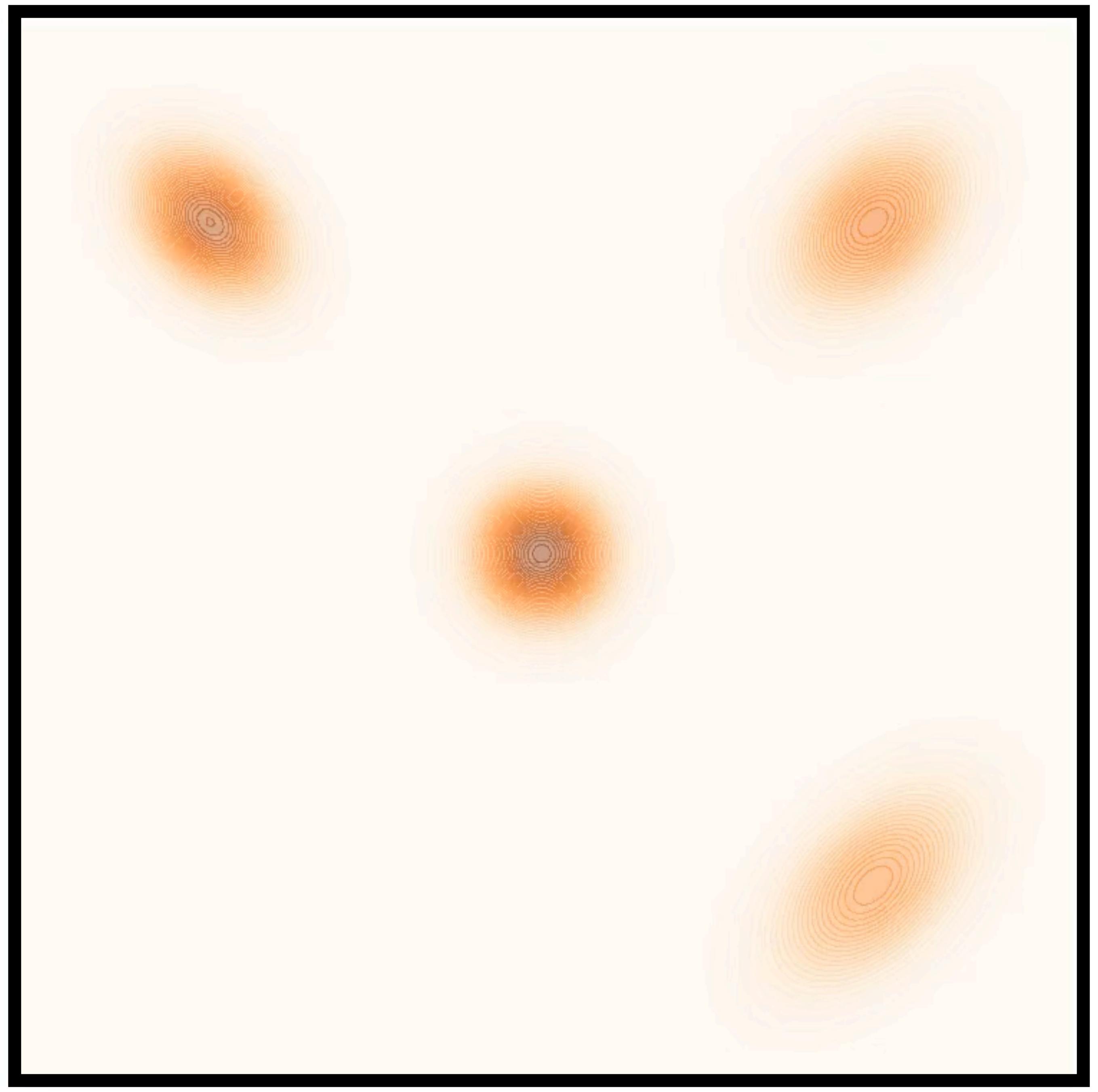


Random walk w/ jumps

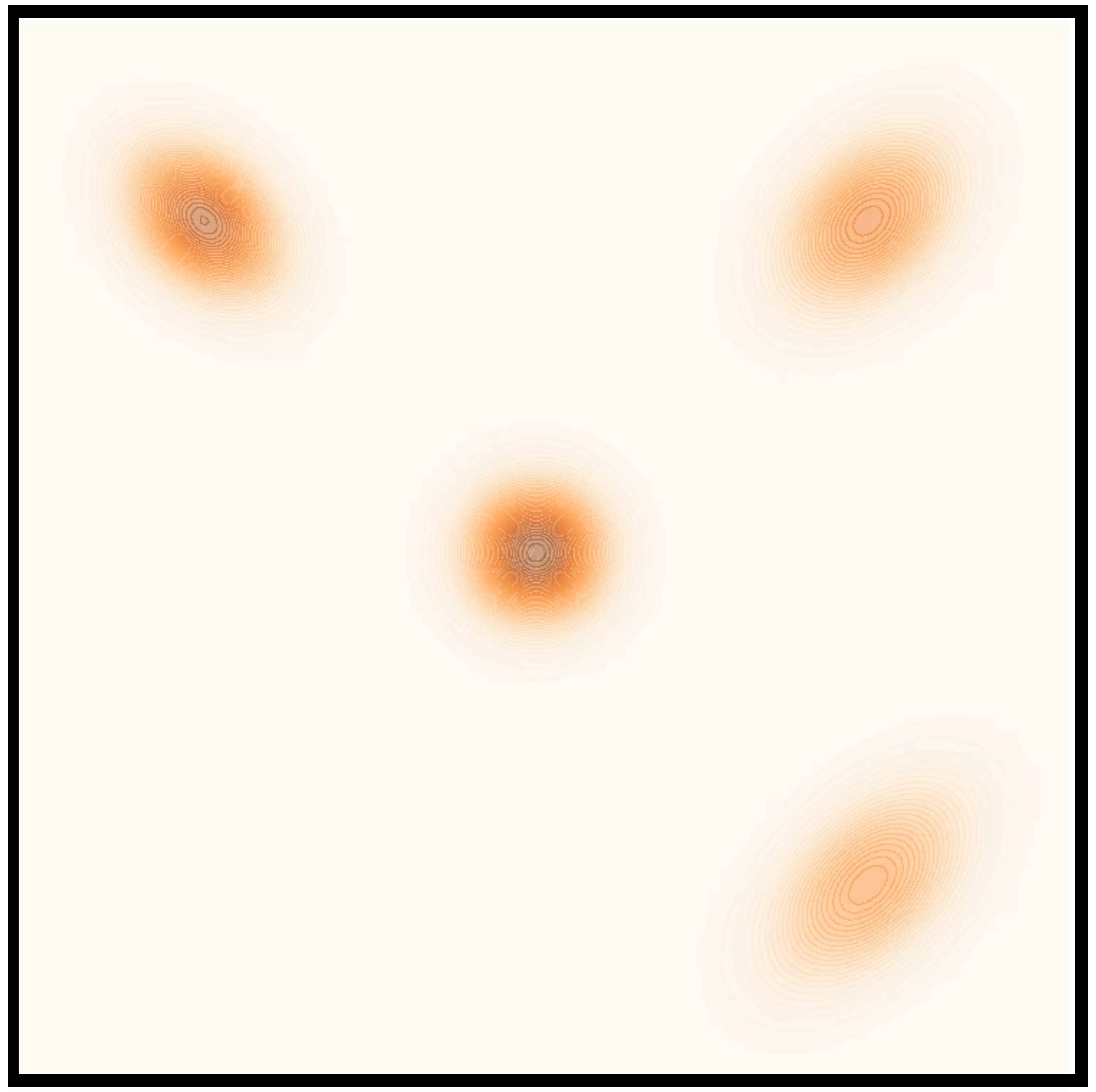
Discretized Brownian motion: Examples



Target distribution

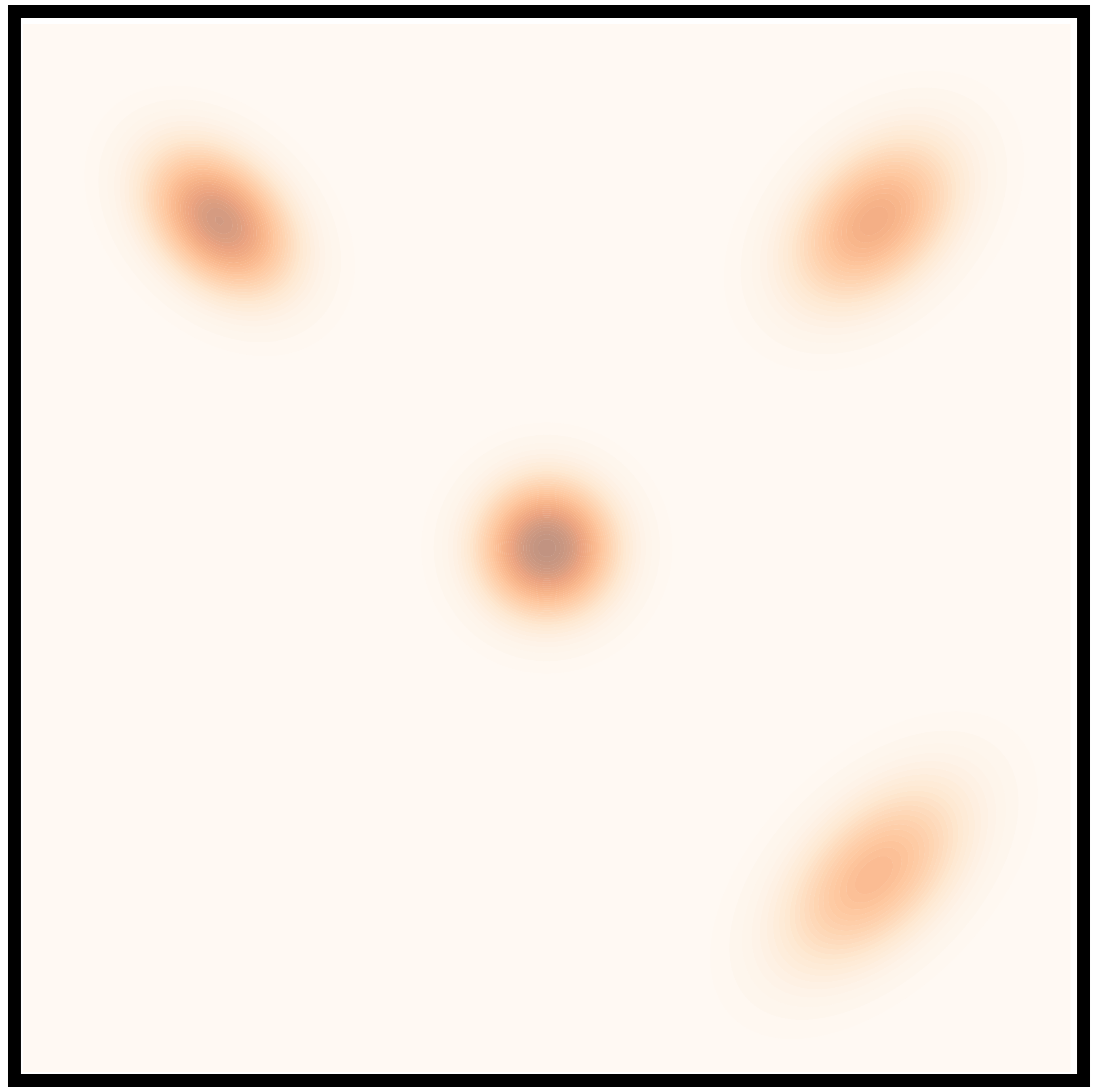


Random walk

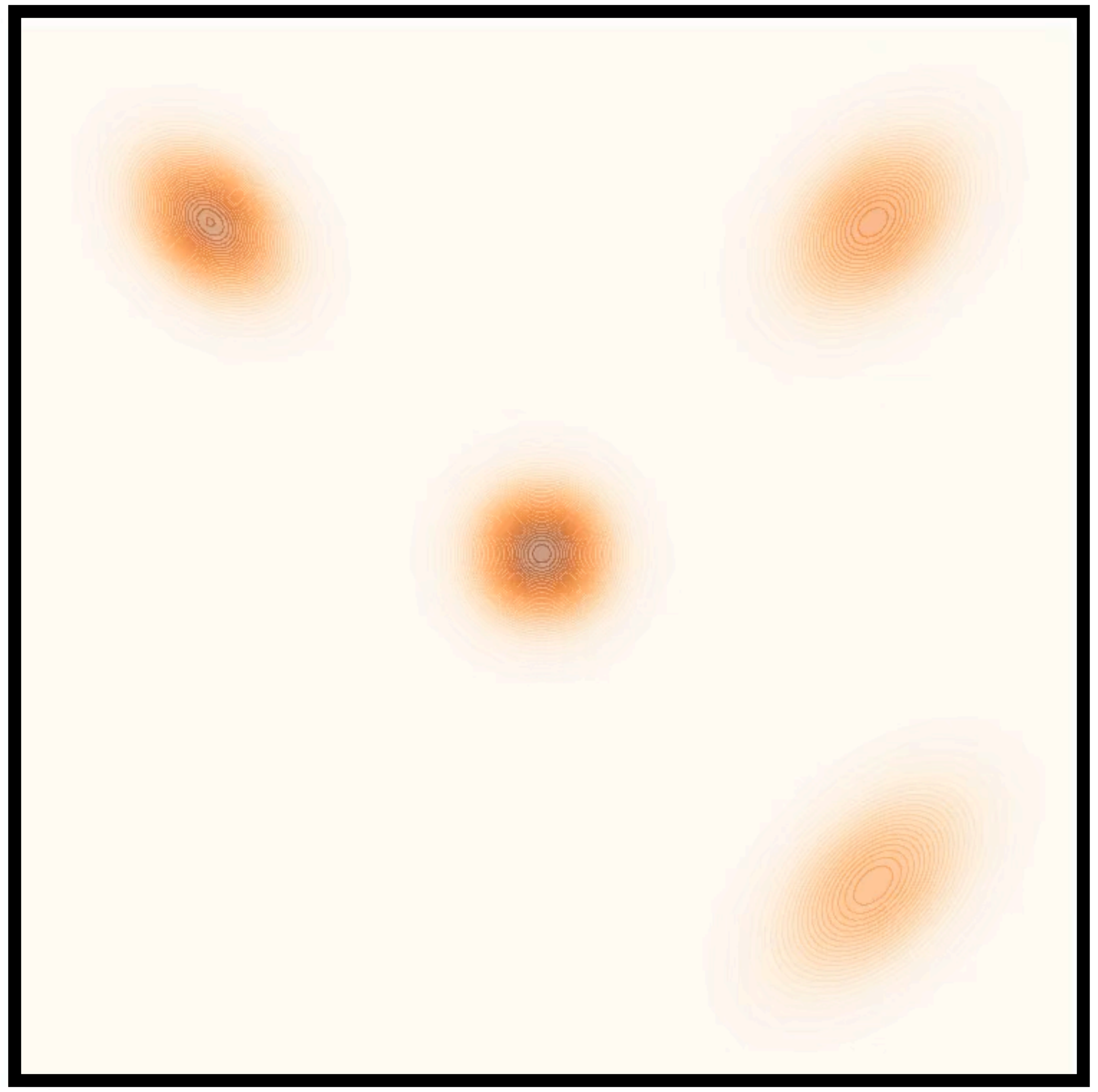


Random walk w/ jumps

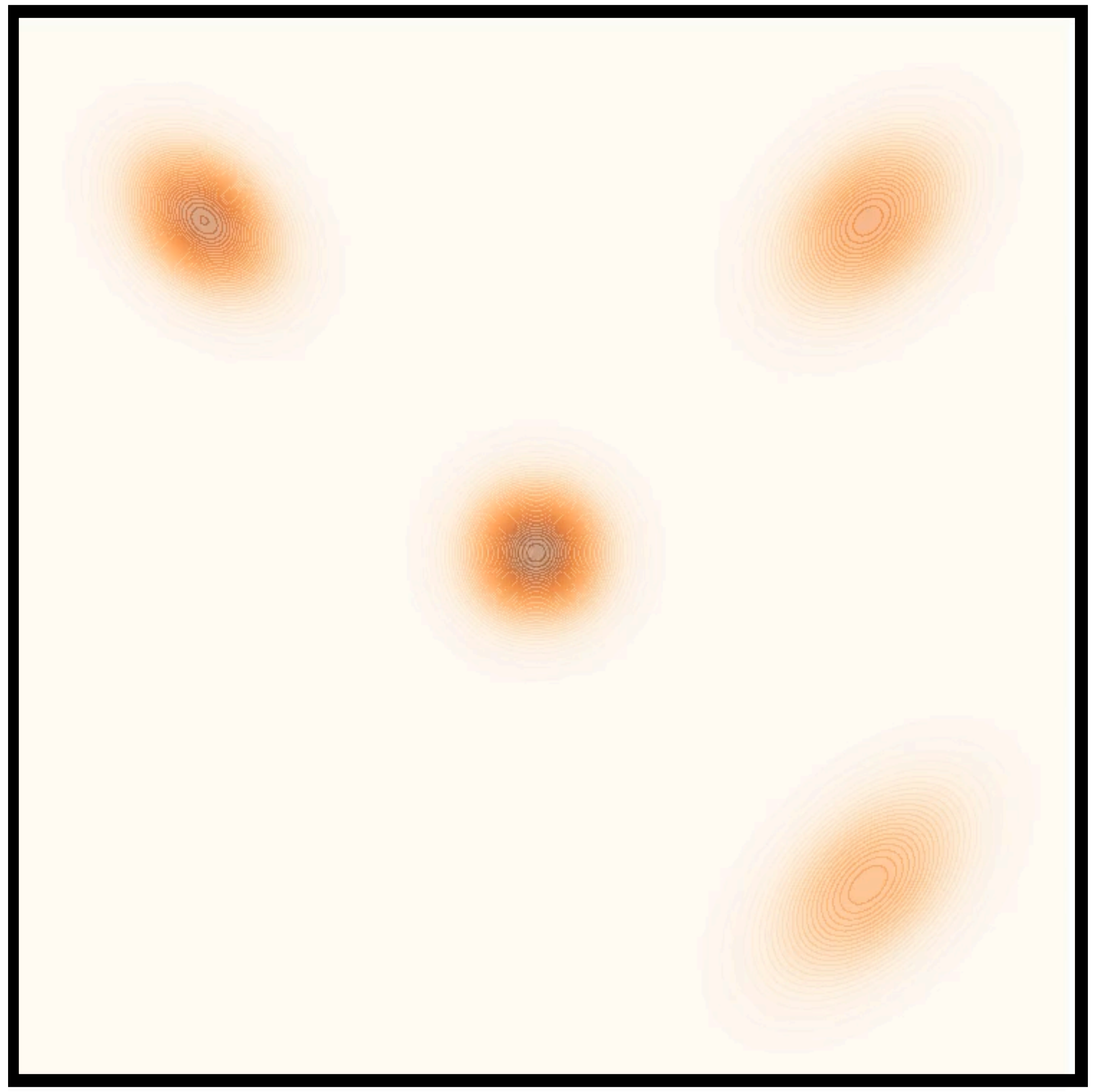
Discretized Brownian motion: Examples



Target distribution

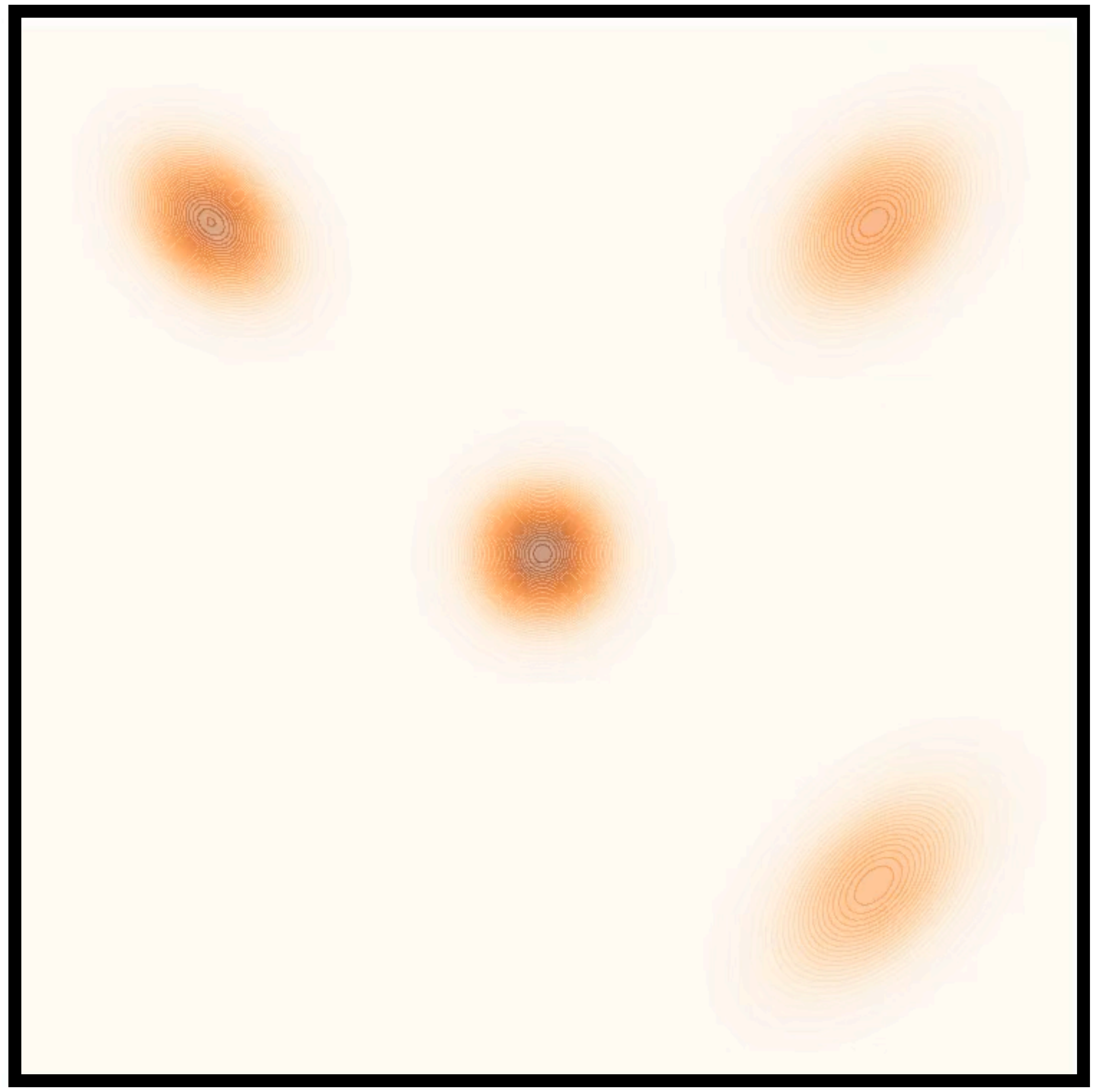


Random walk



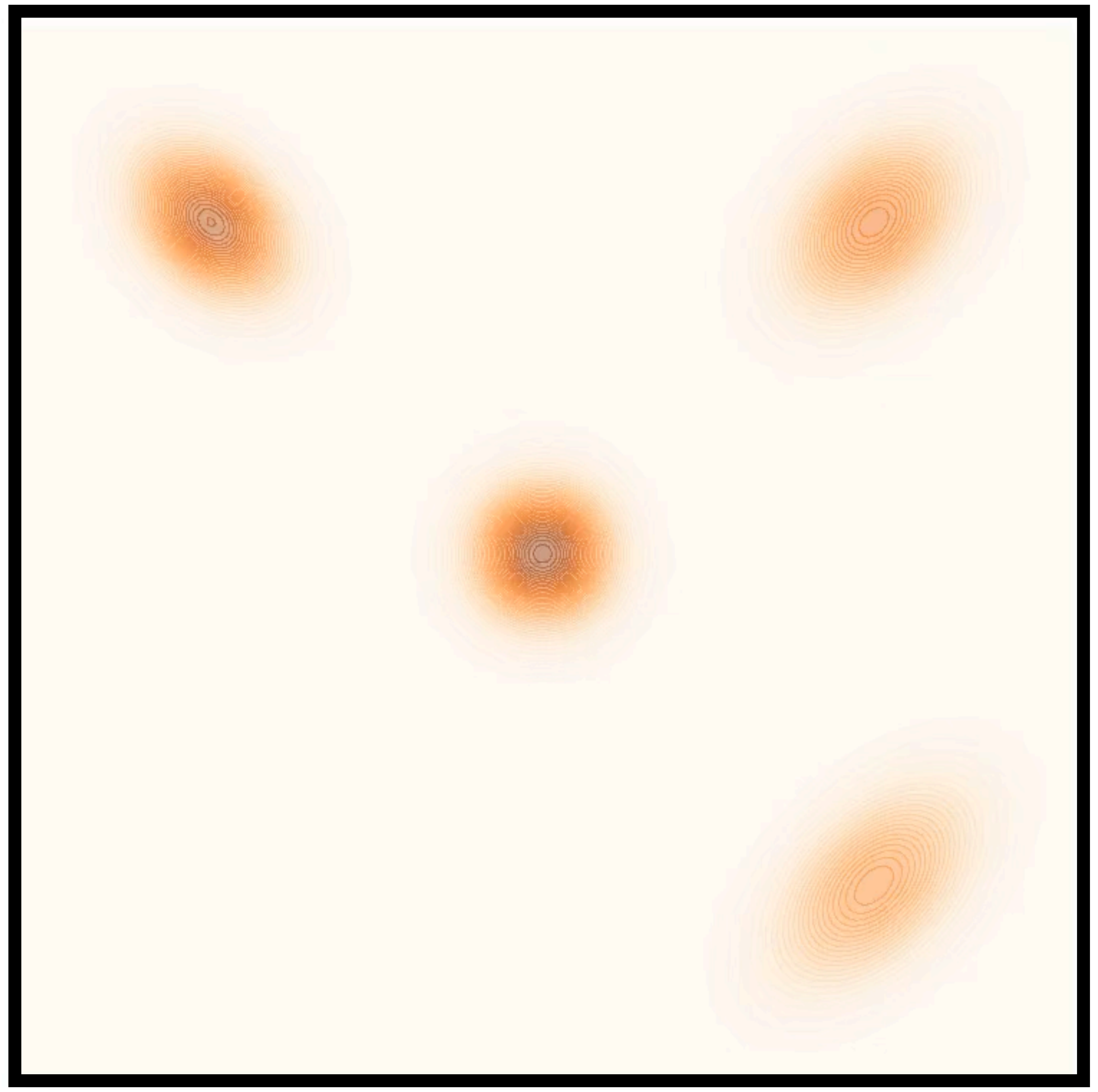
Random walk w/ jumps

Metropolis-adjusted Brownian motion



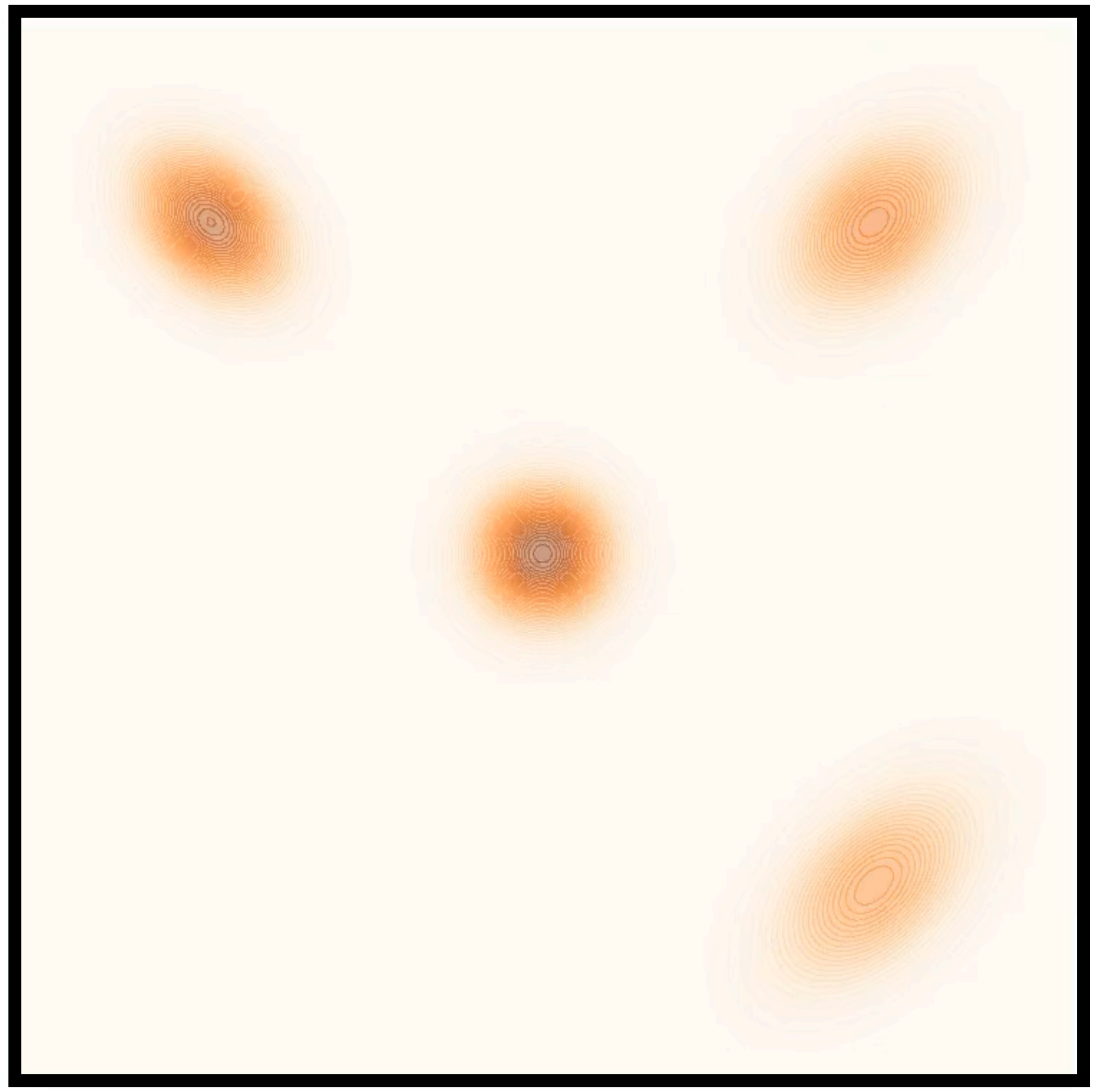
Random walk

Metropolis-adjusted Brownian motion

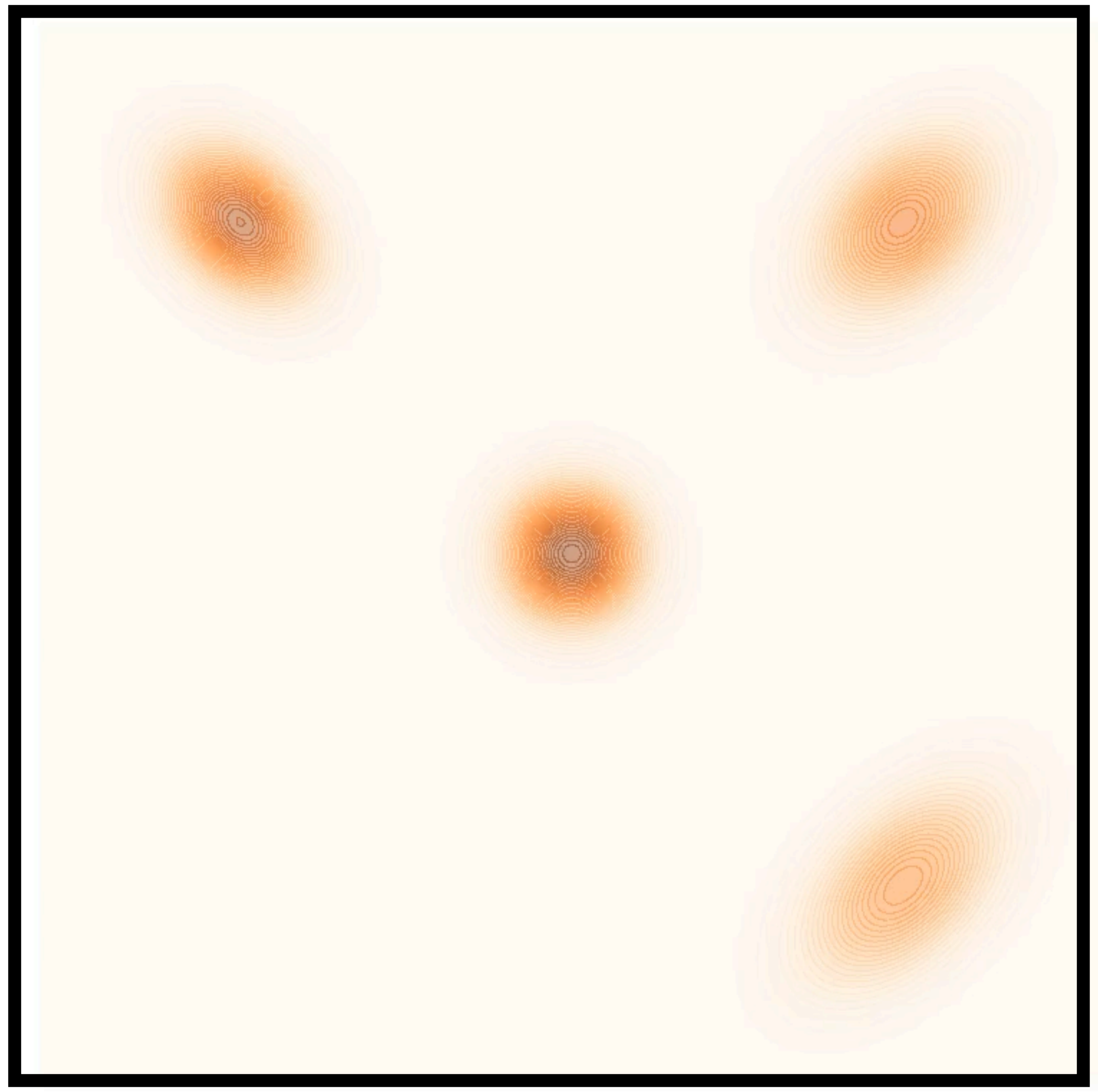


Random walk

Metropolis-adjusted Brownian motion

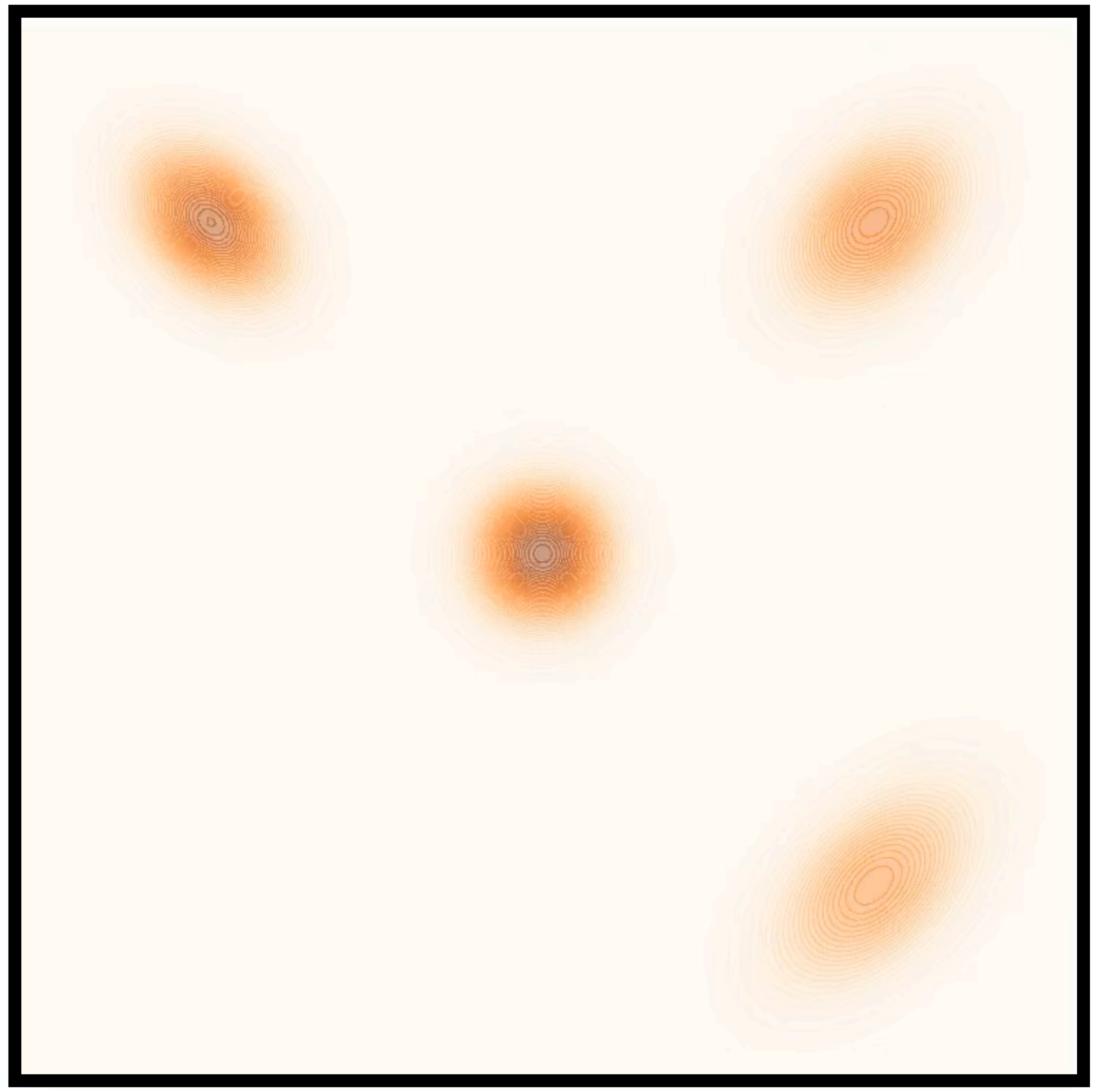


Random walk

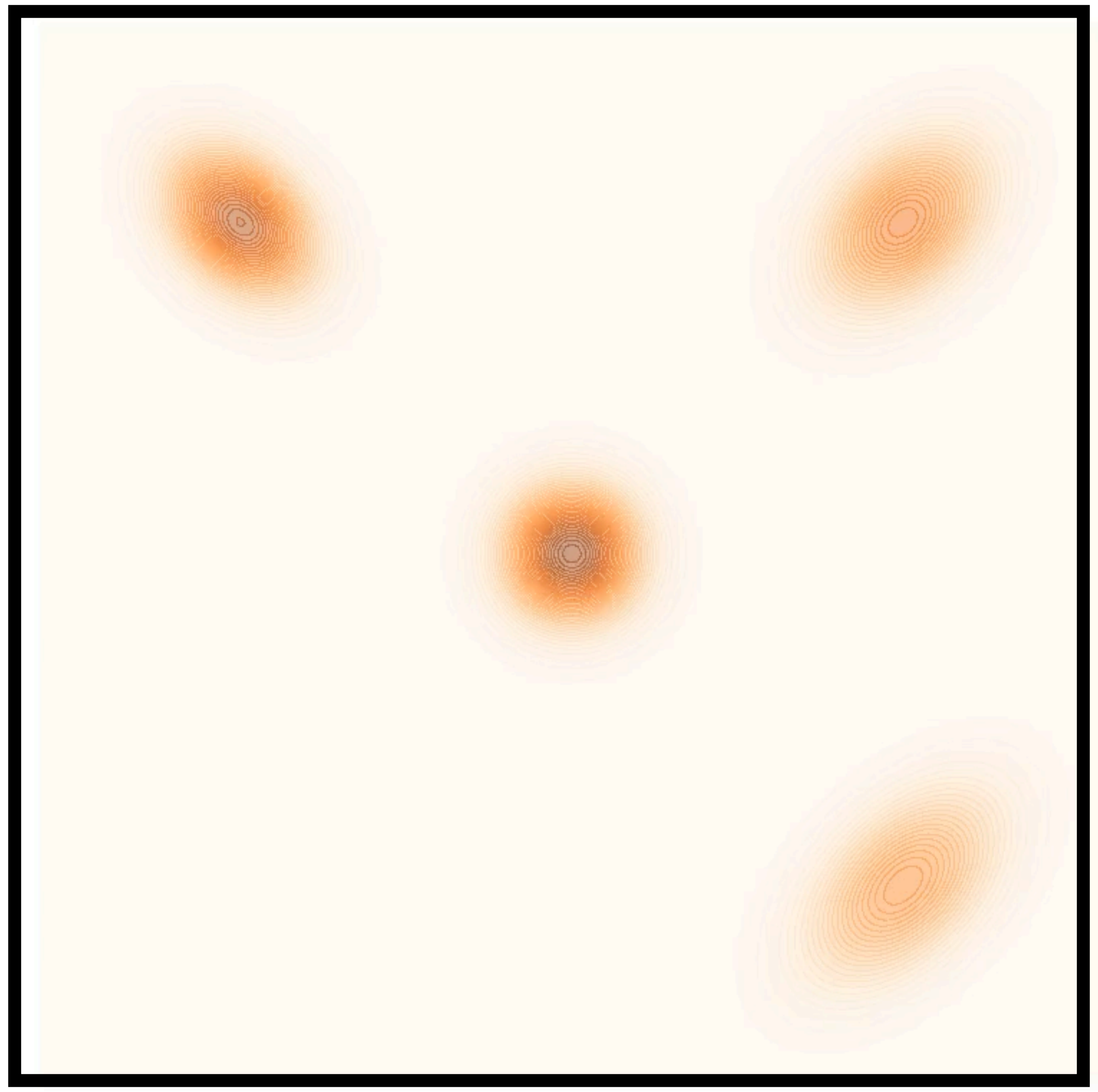


Random walk w/ MH

Metropolis-adjusted Brownian motion

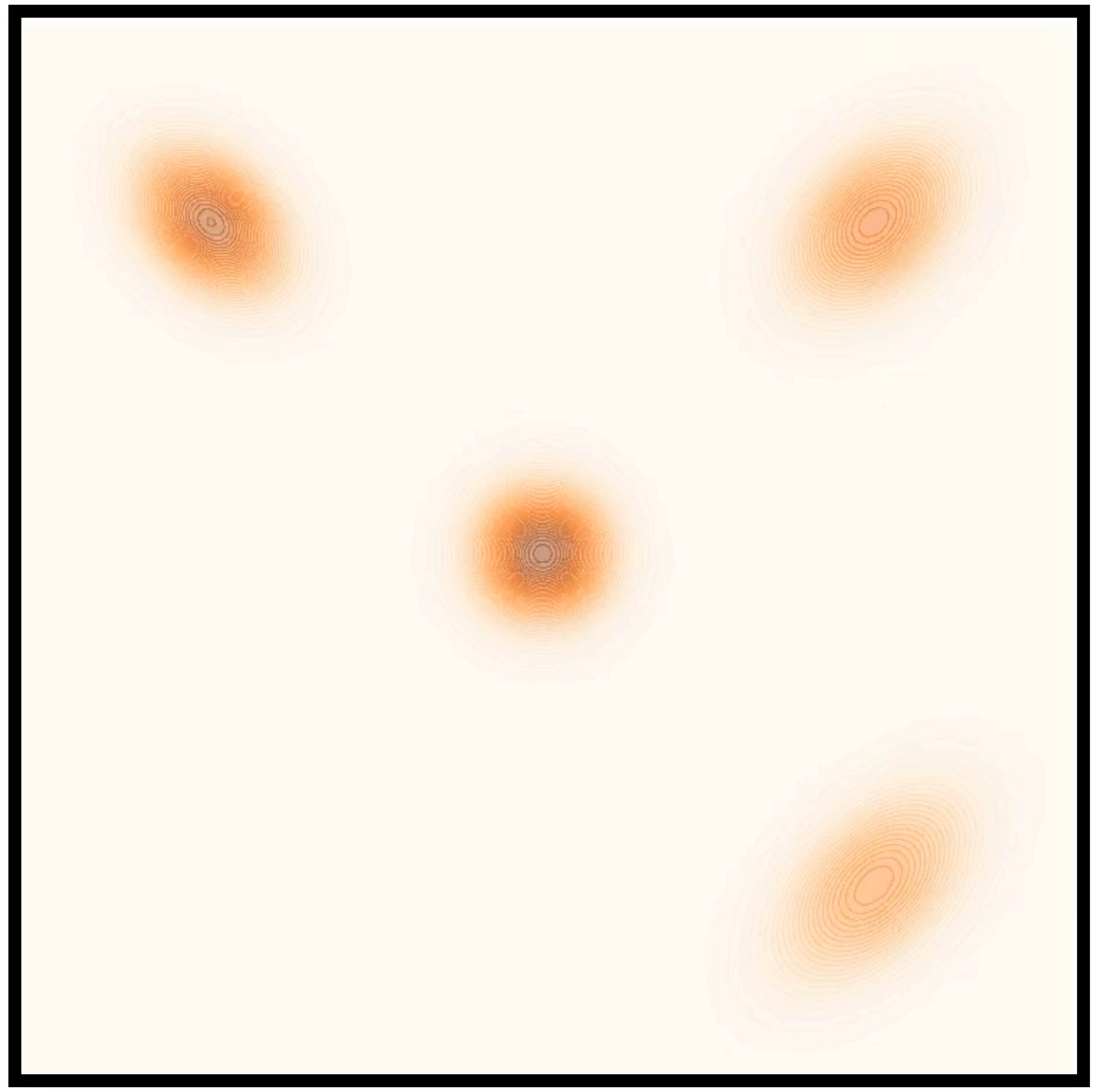


Random walk

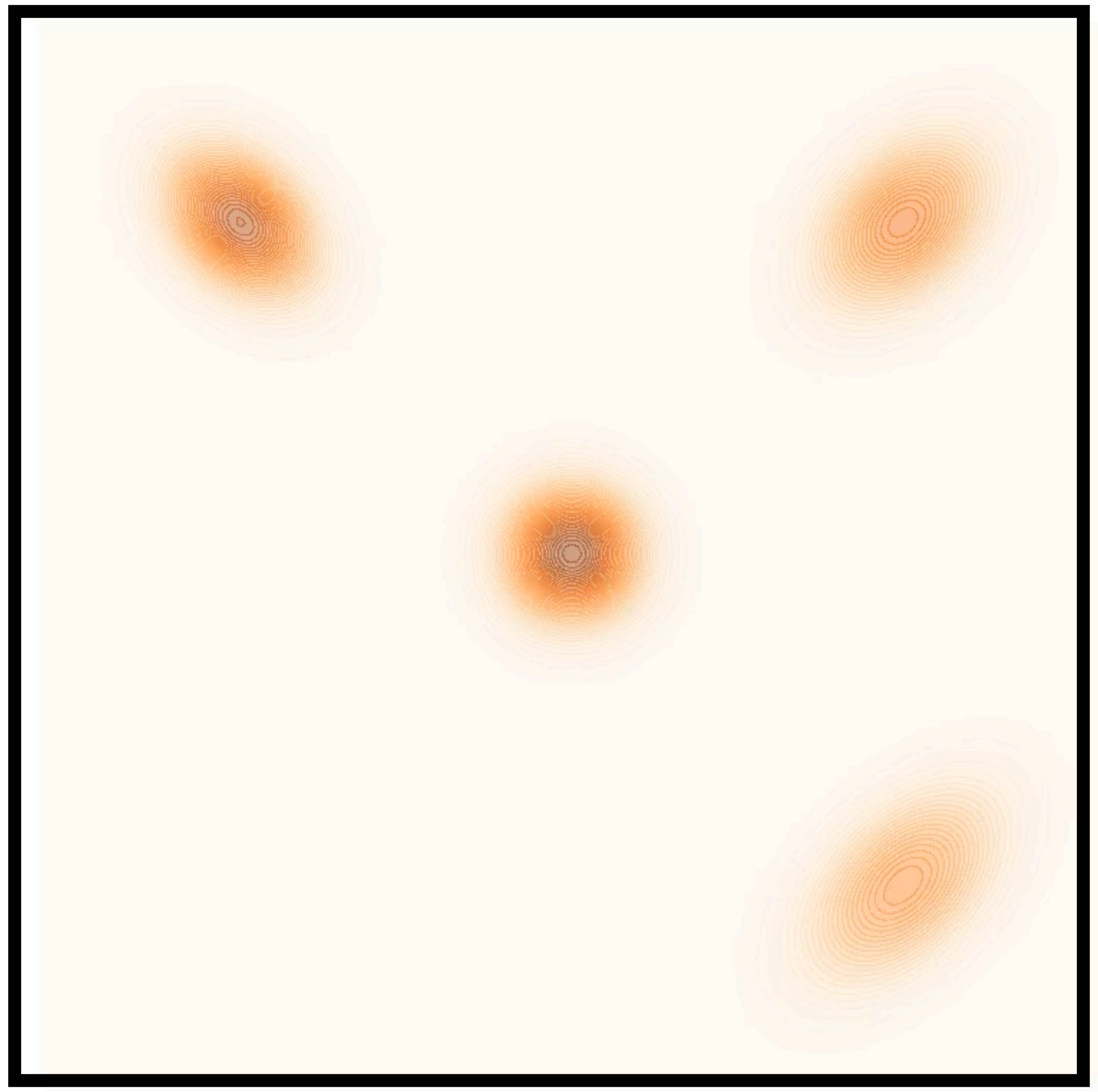


Random walk w/ MH

Metropolis-adjusted Brownian motion

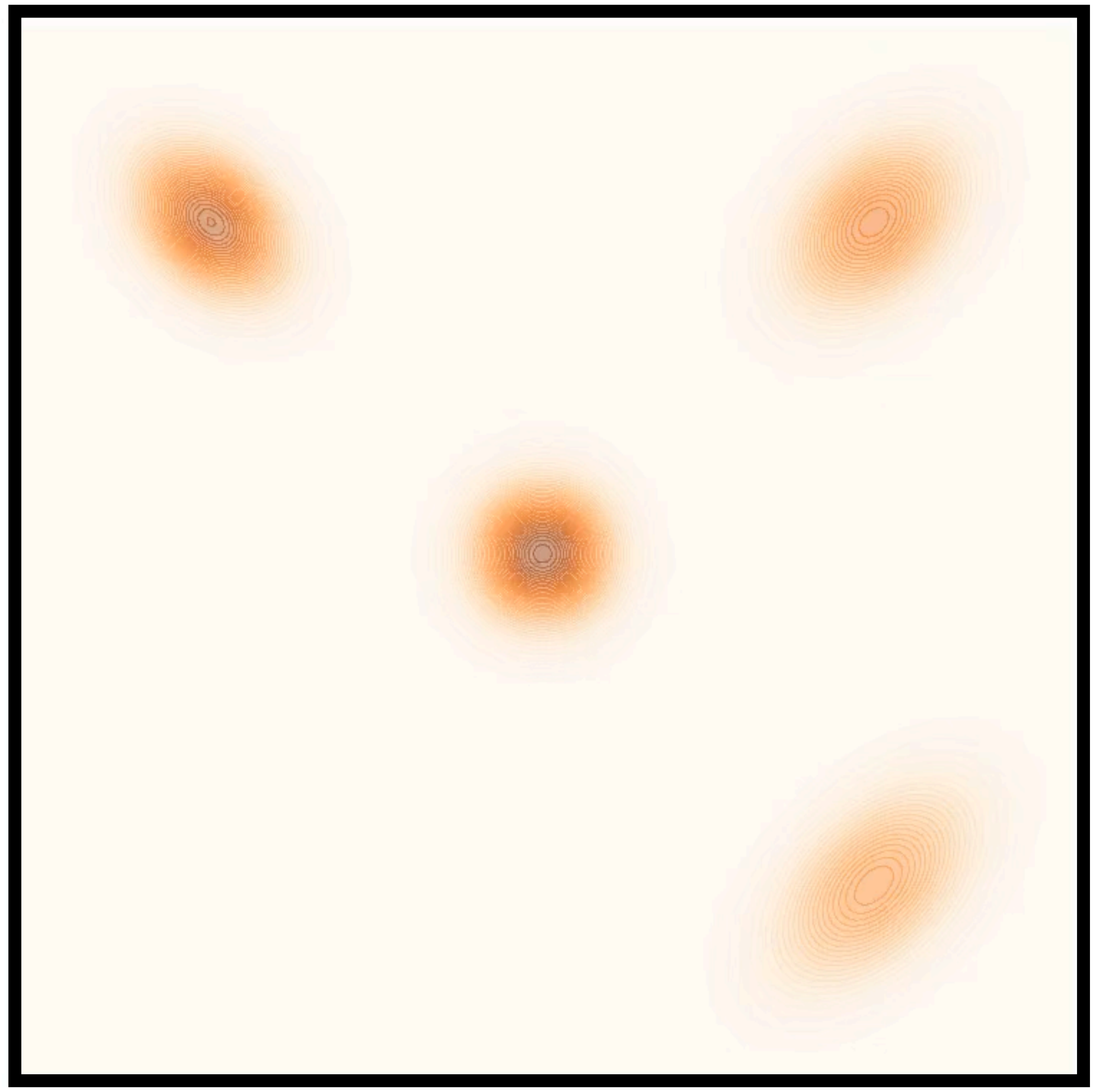


Random walk

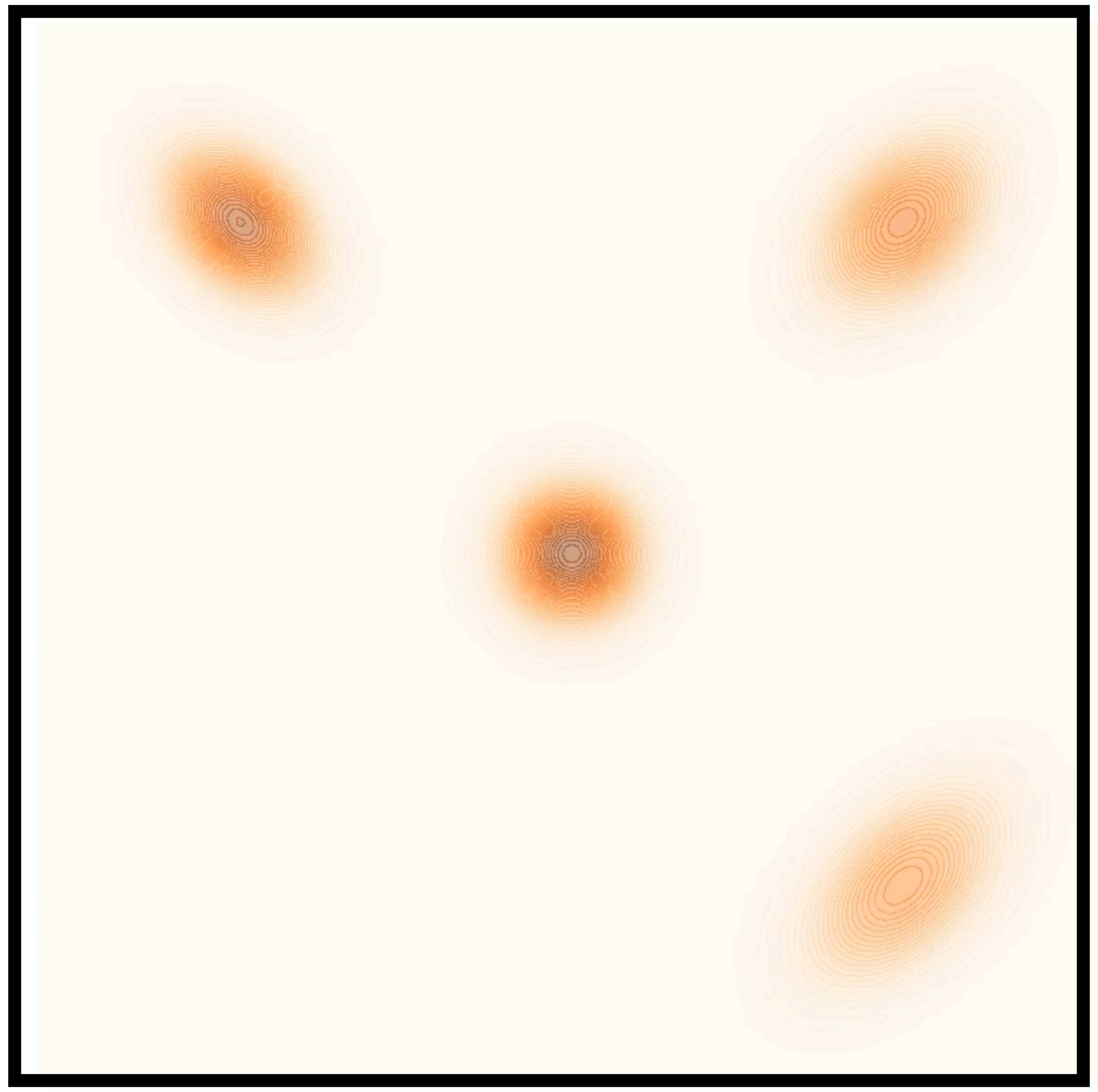


Random walk w/ MH

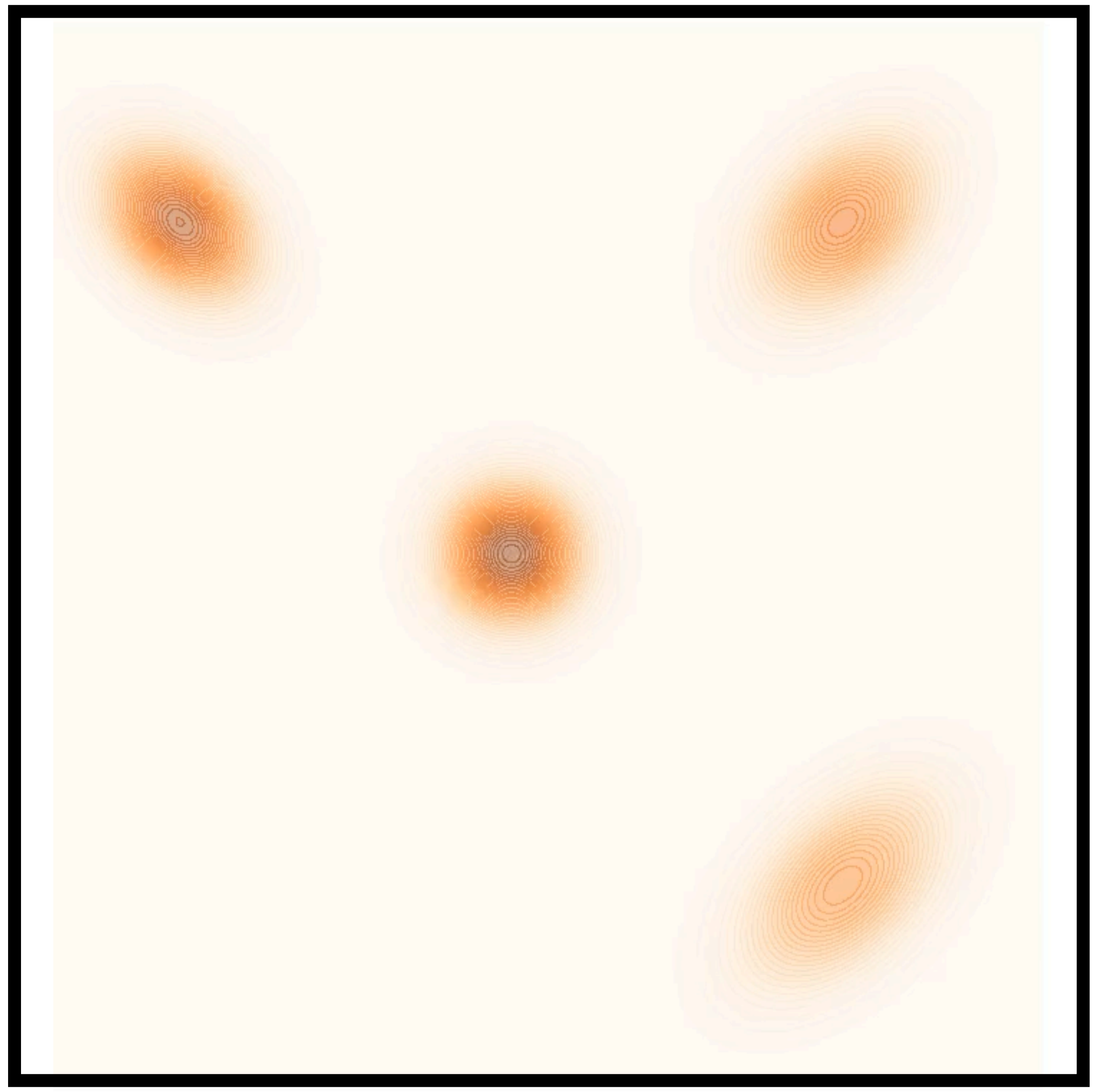
Metropolis-adjusted Brownian motion



Random walk

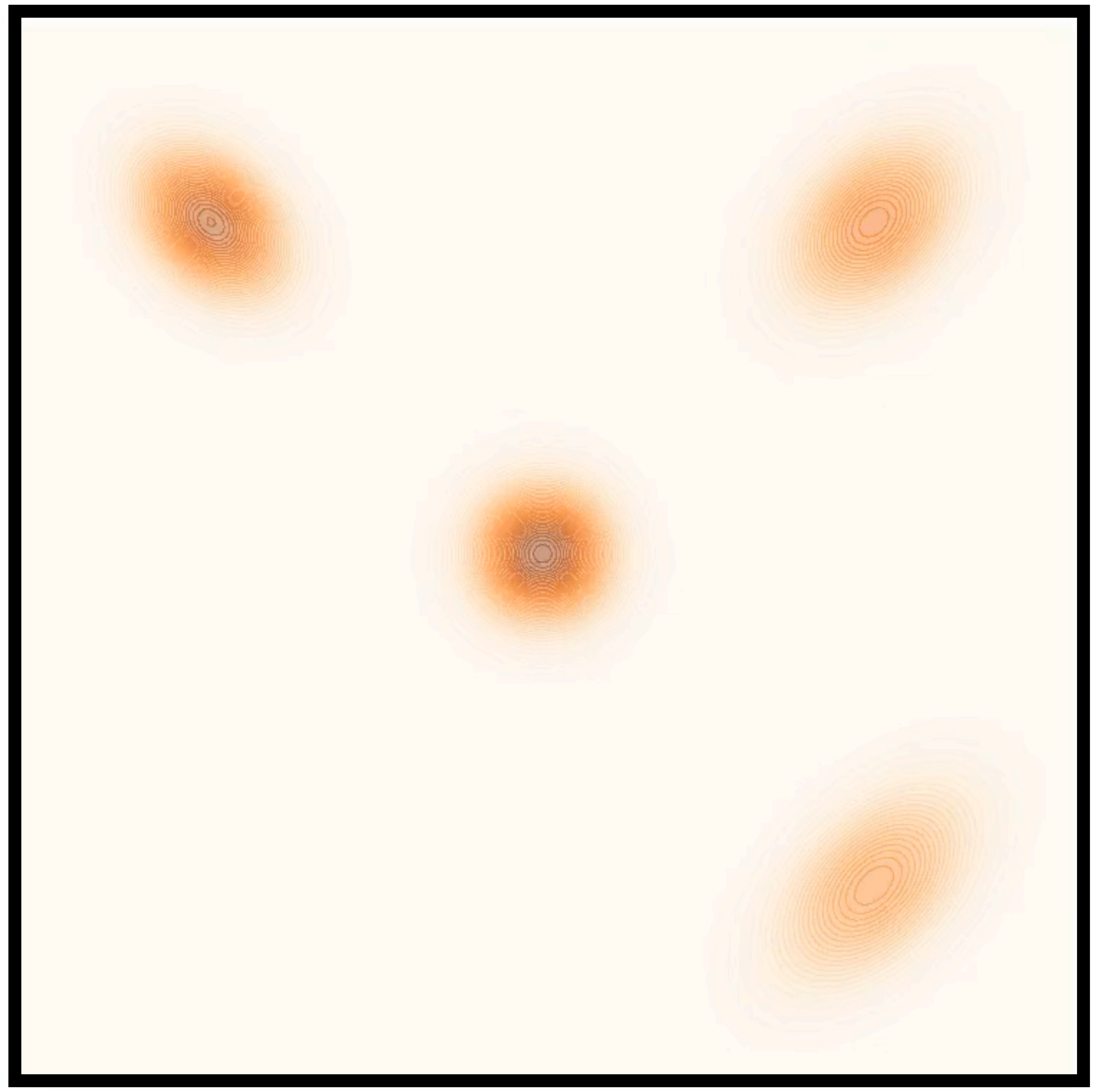


Random walk w/ MH

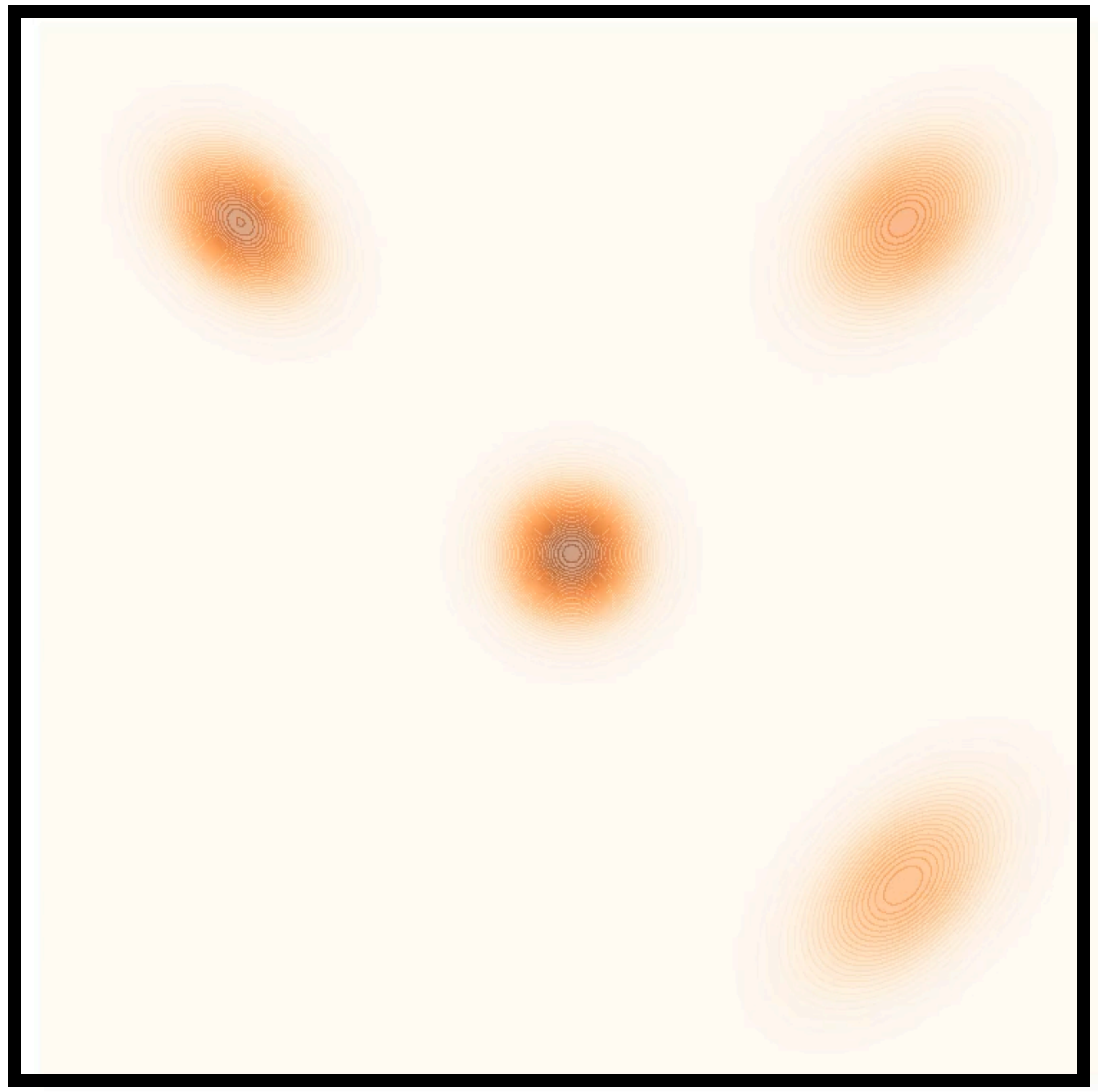


Random walk w/ MH and w/ jumps

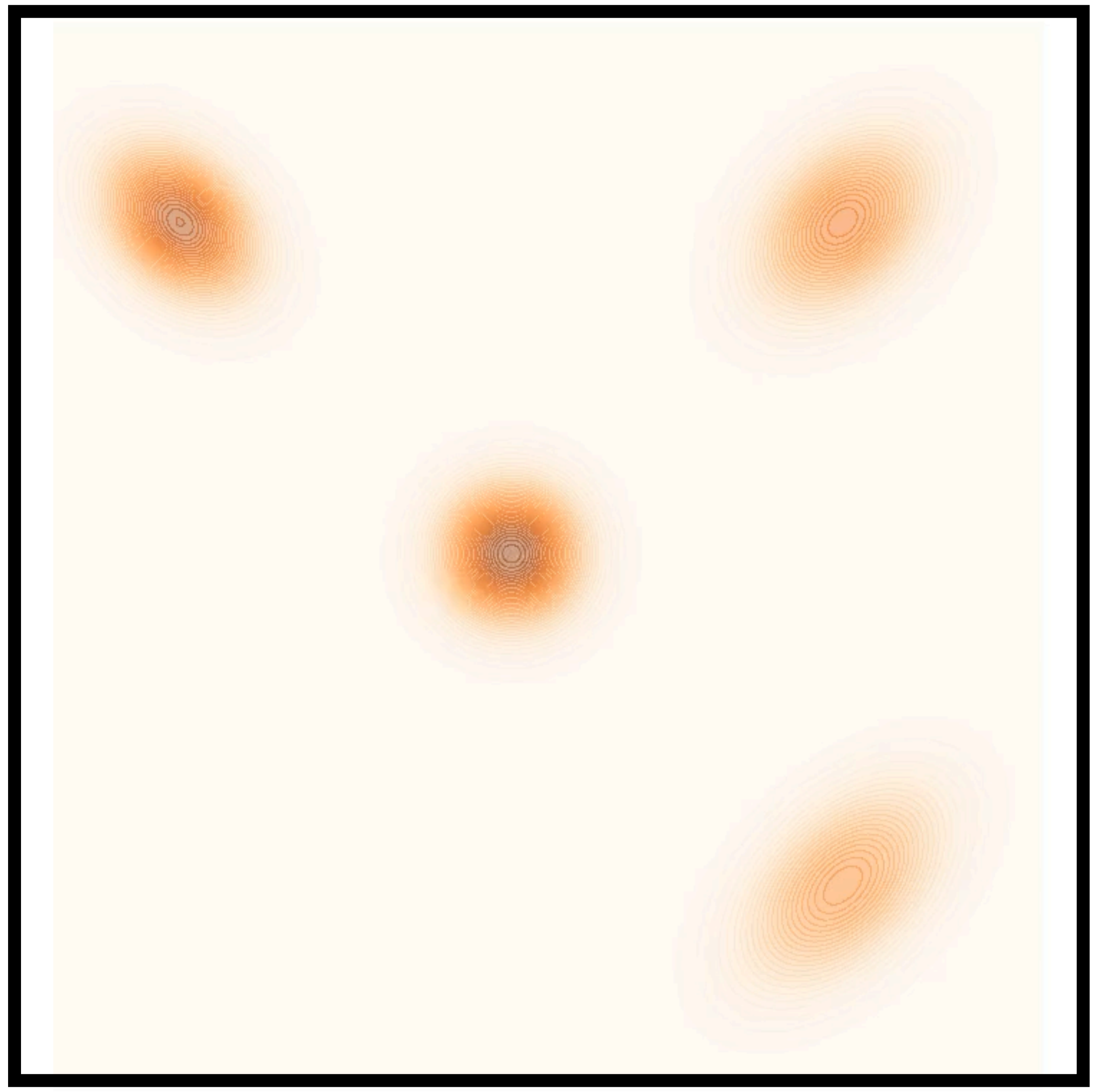
Metropolis-adjusted Brownian motion



Random walk

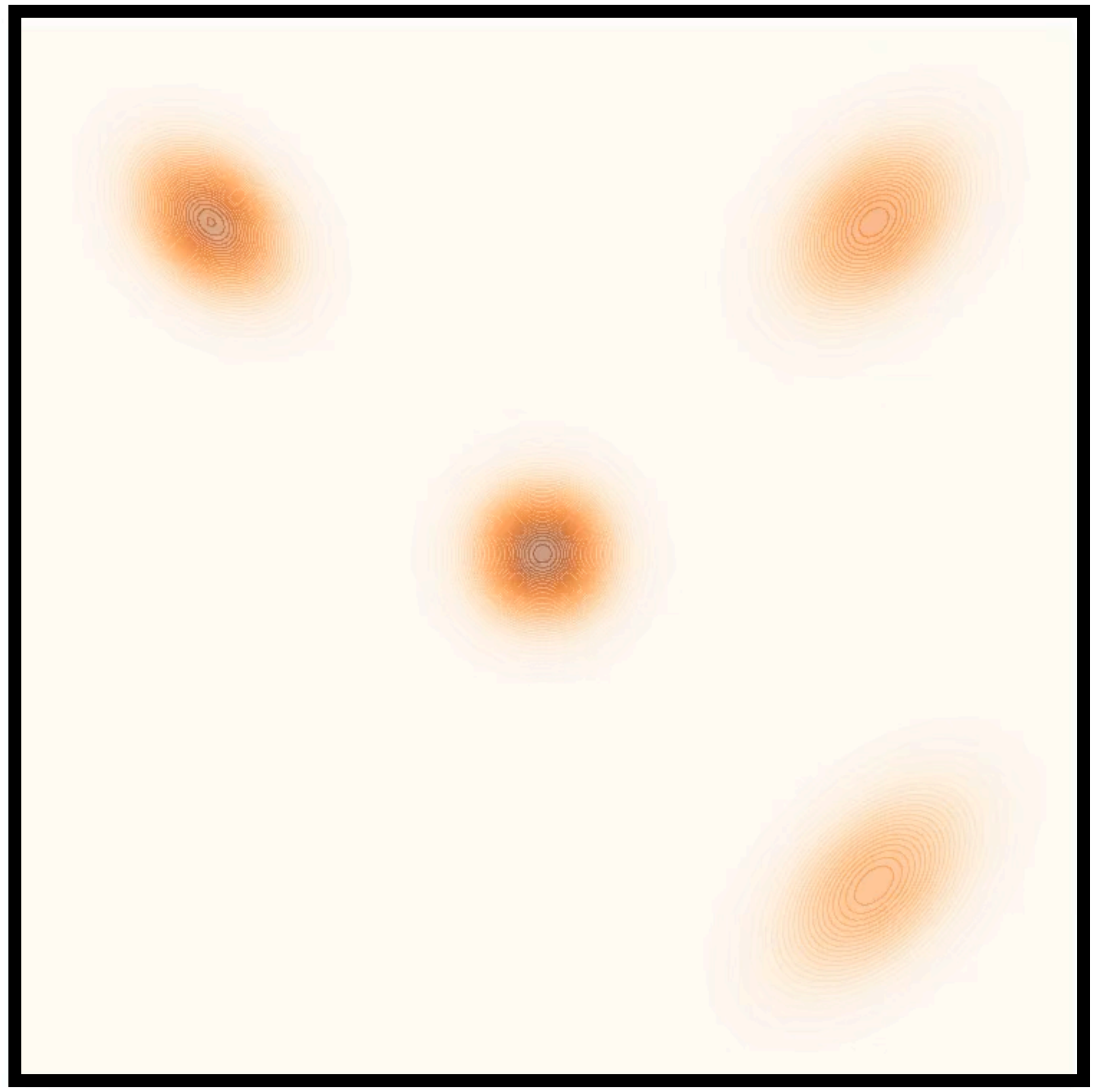


Random walk w/ MH

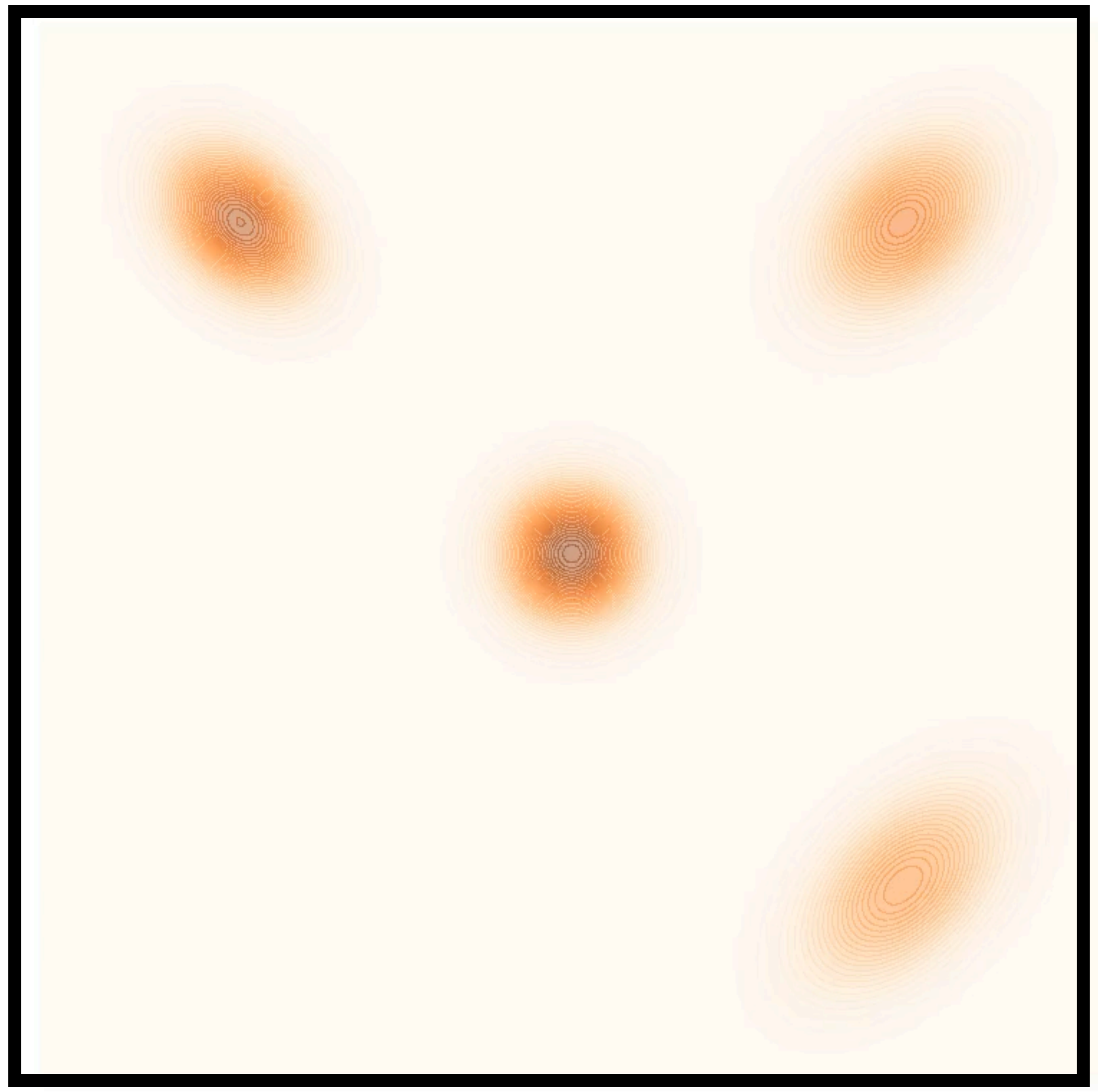


Random walk w/ MH and w/ jumps

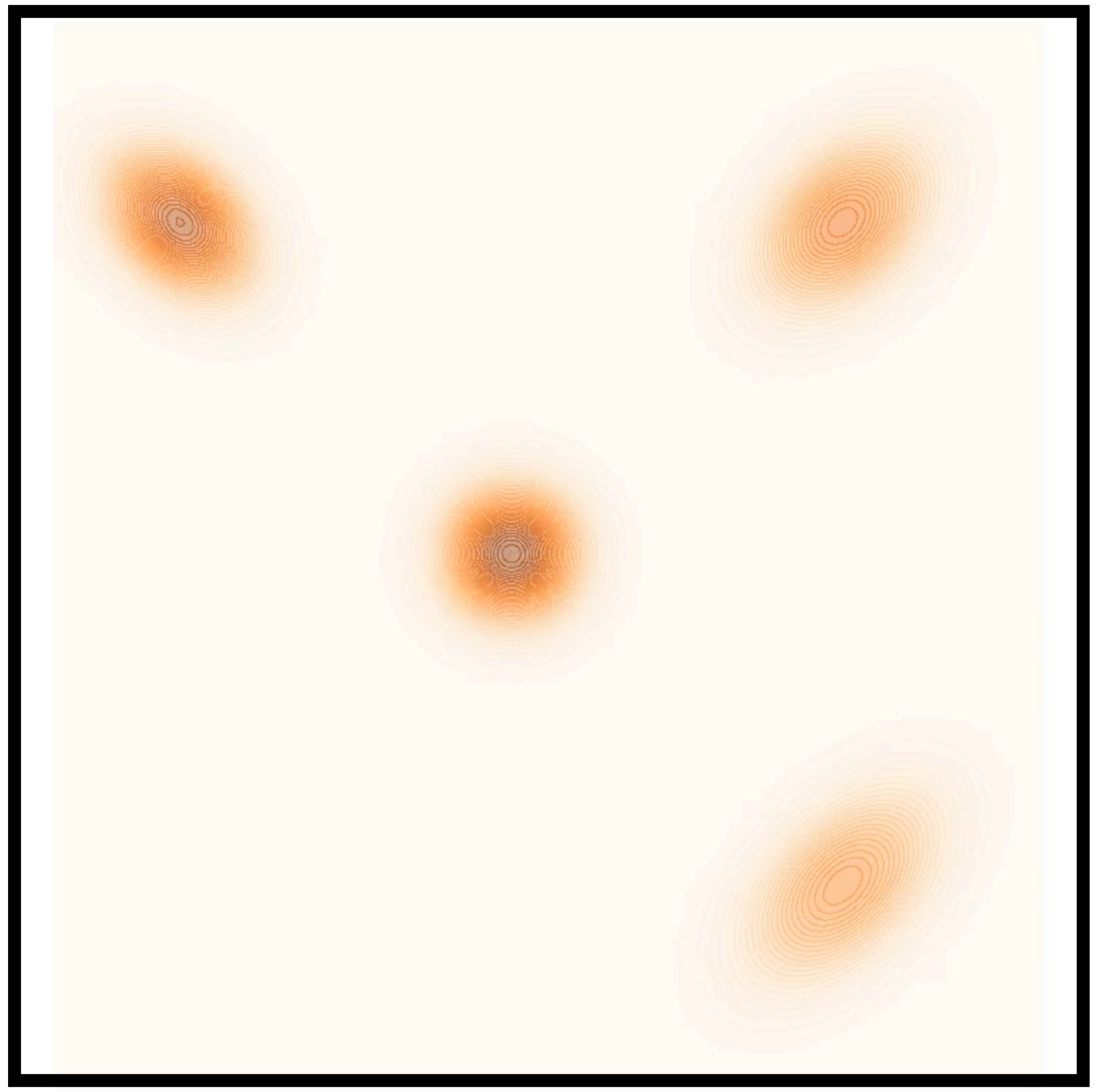
Metropolis-adjusted Brownian motion



Random walk

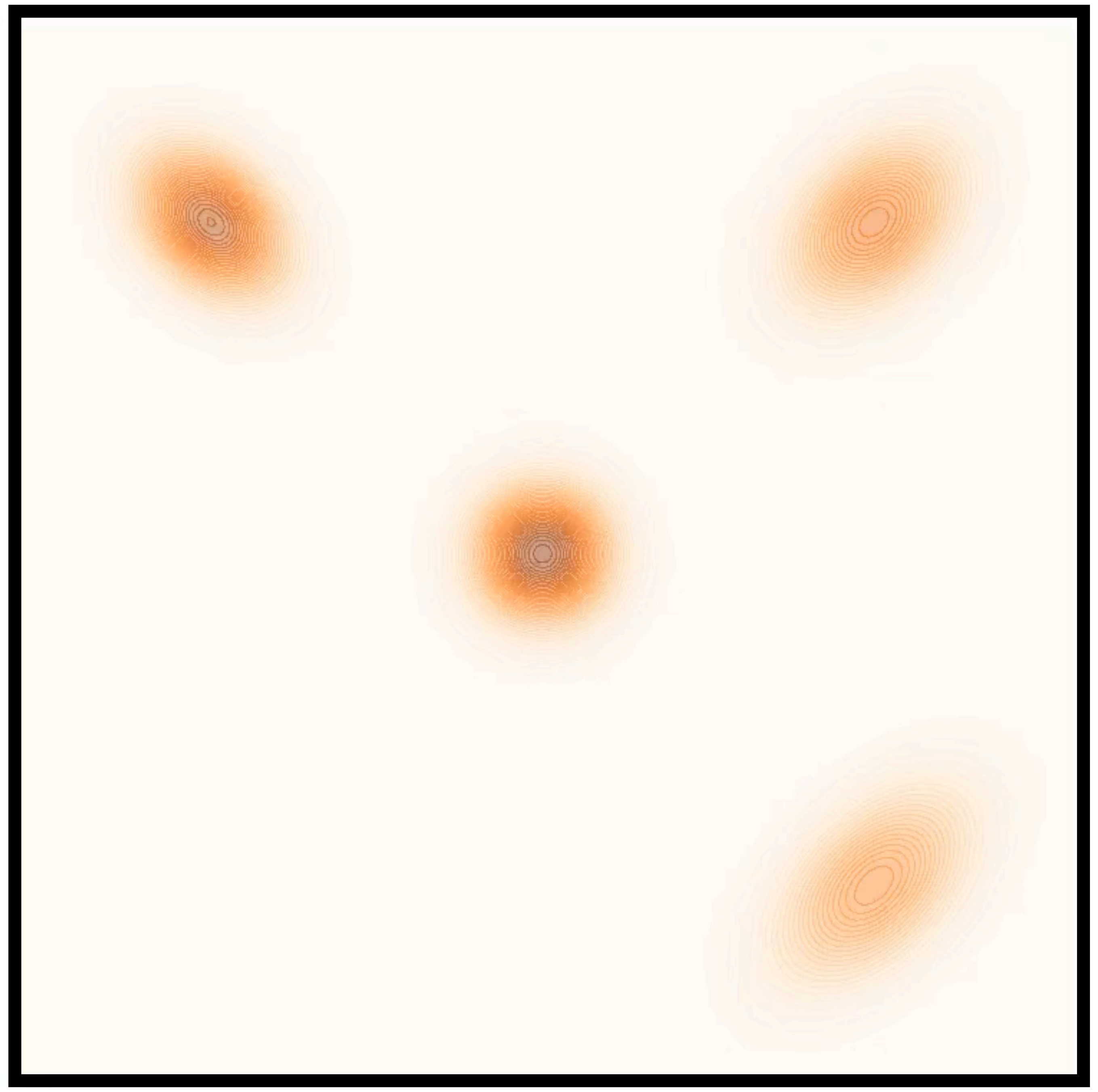


Random walk w/ MH

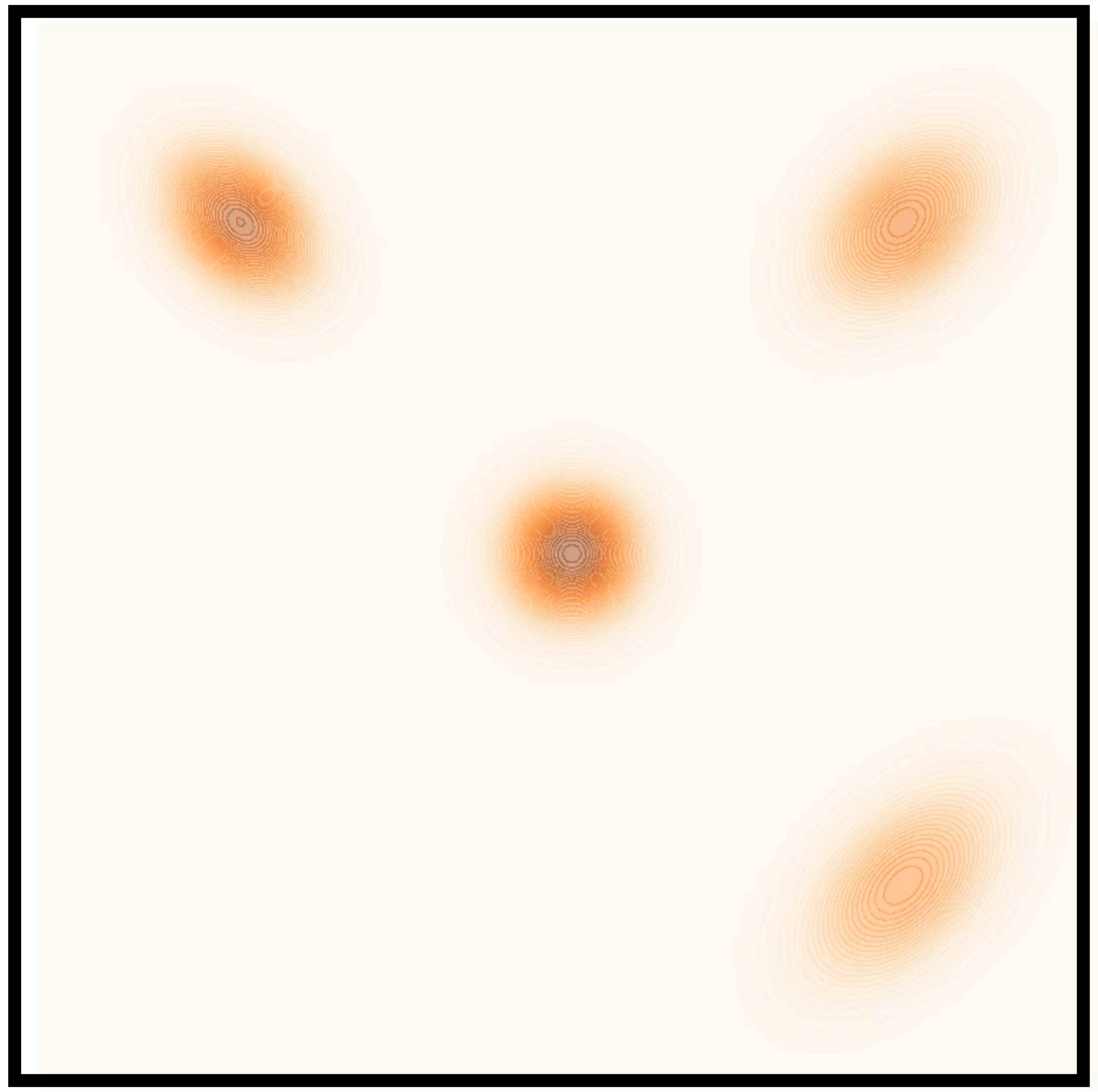


Random walk w/ MH and w/ jumps

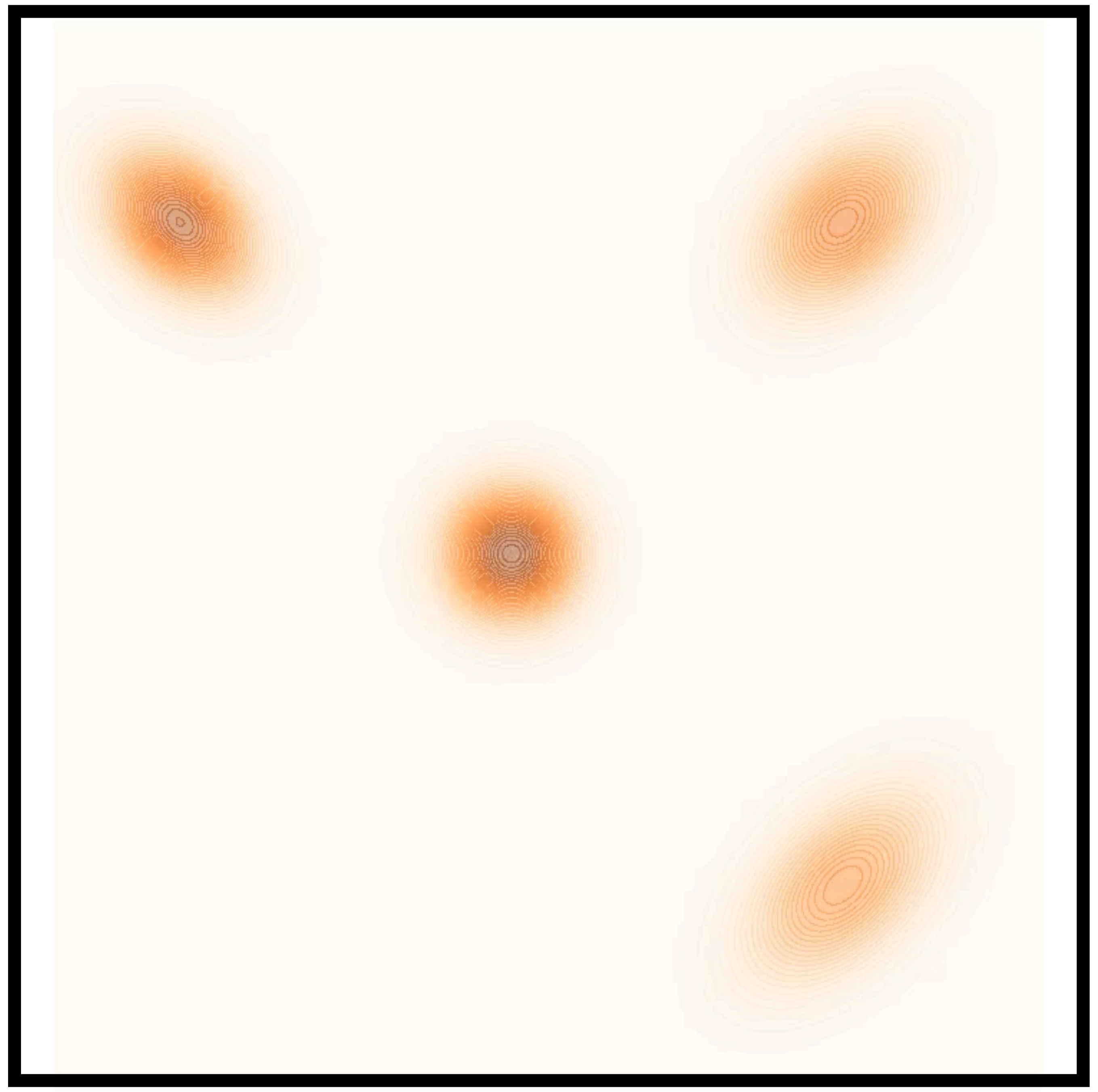
Metropolis-adjusted Brownian motion



Random walk



Random walk w/ MH



Random walk w/ MH and w/ jumps

Stochastic Different equations (SDEs)

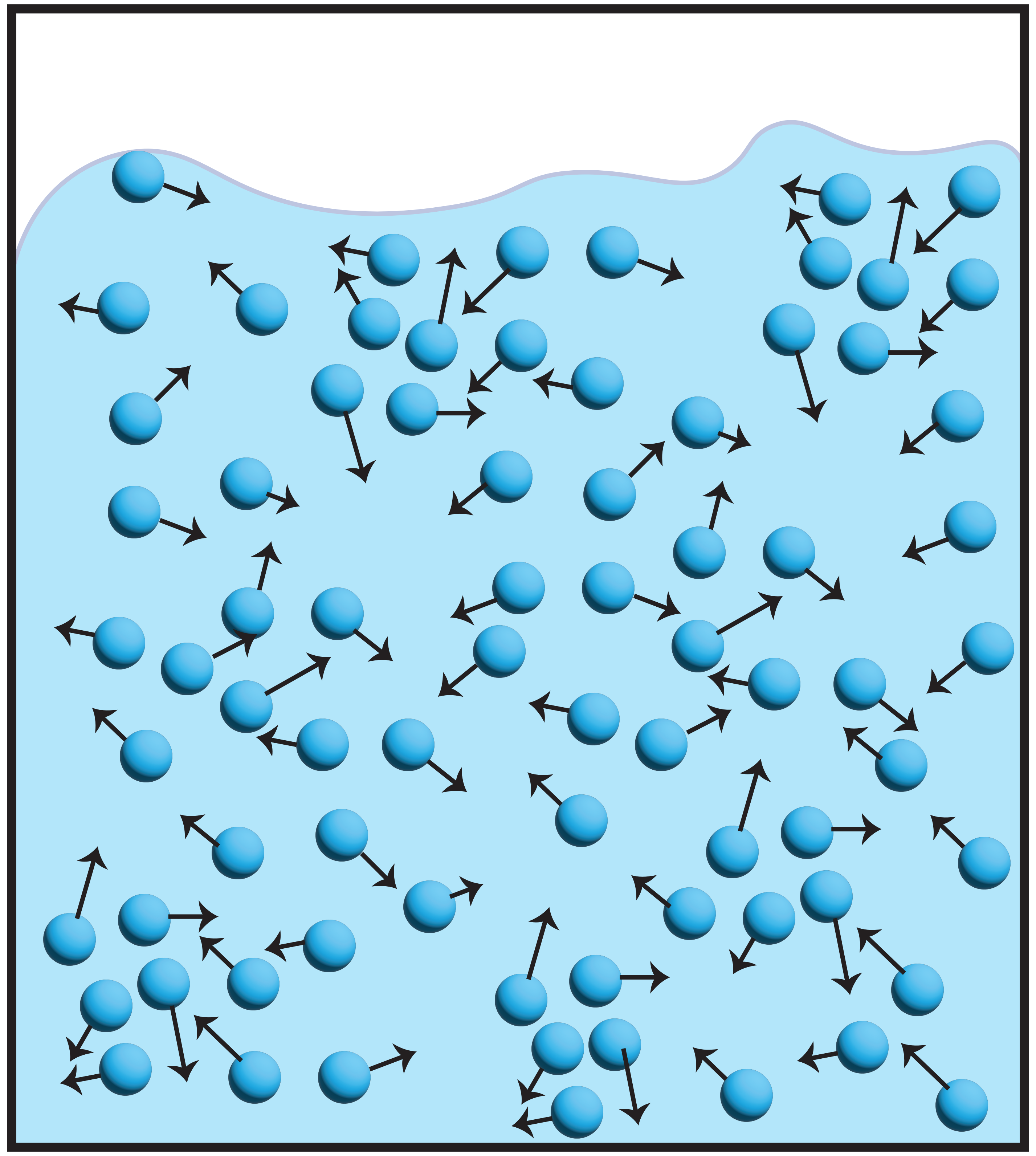
average trajectory

$$d\mathbf{x}_t = \mu(\mathbf{x}_t, t)dt + \sigma(\mathbf{x}_t, t)dW_t$$

jiggle

drift

randomness



Stochastic Different equations (SDEs)

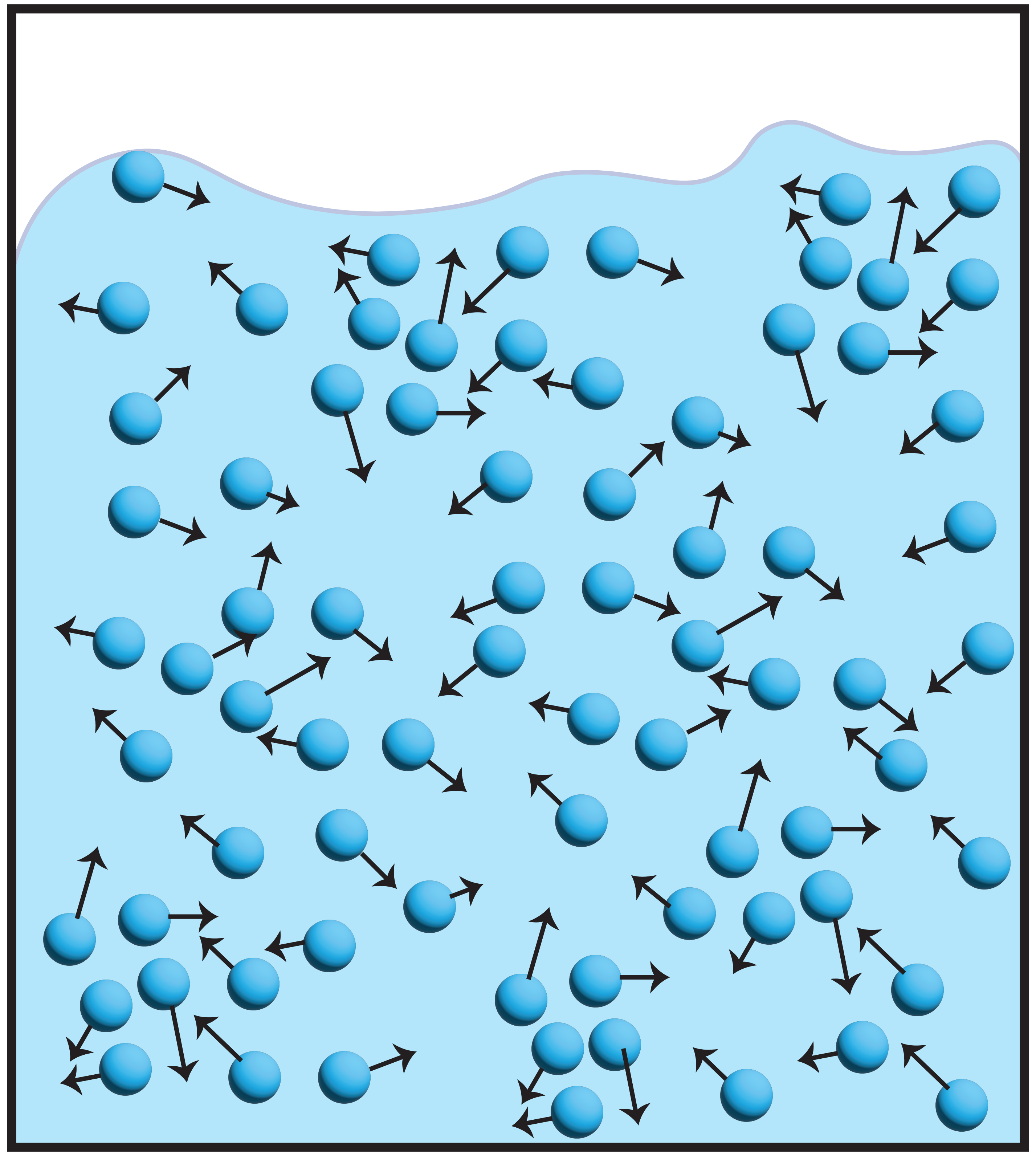
average trajectory

$$d\mathbf{x}_t = \mu(\mathbf{x}_t, t)dt + \sigma(\mathbf{x}_t, t)dW_t$$

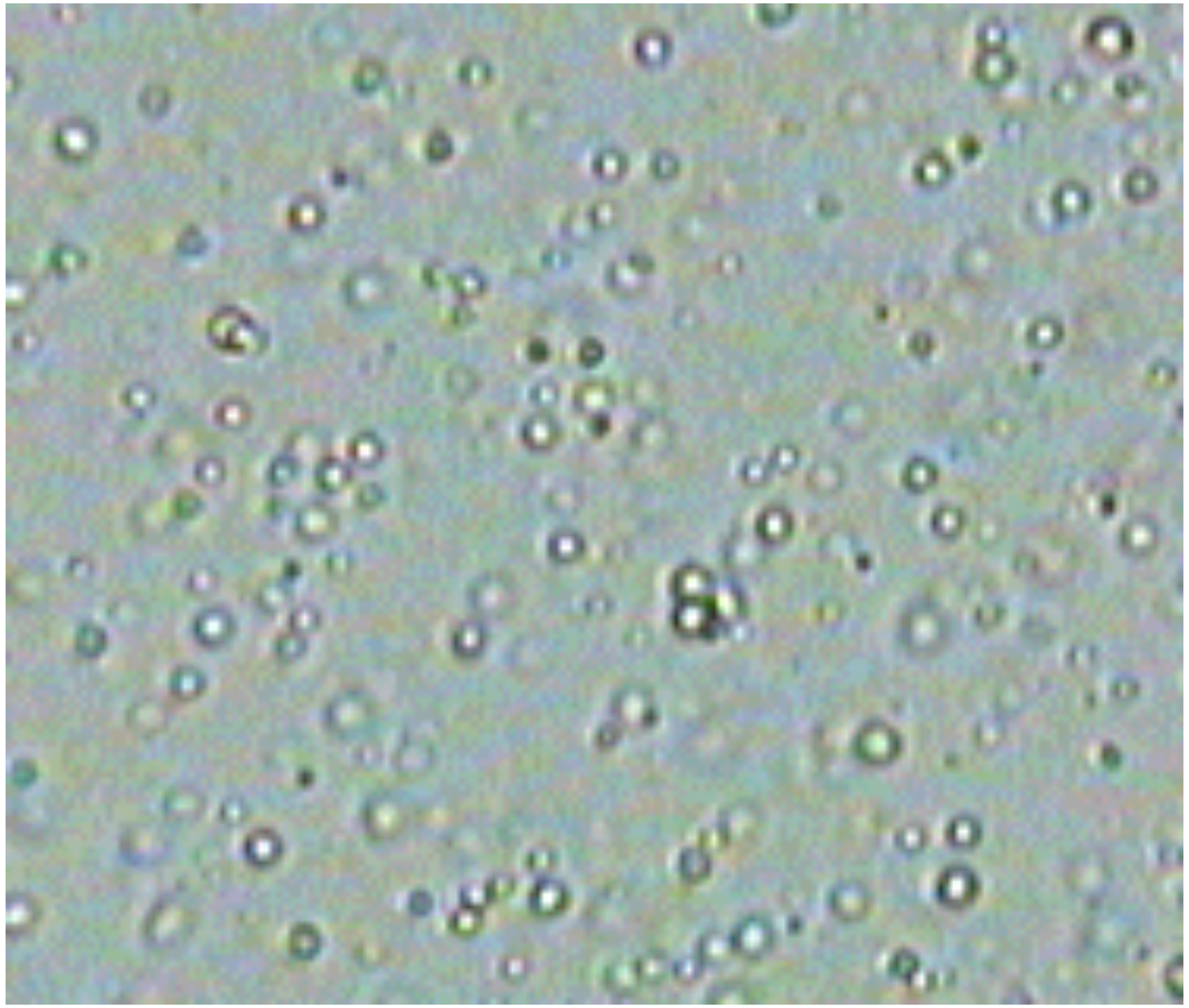
jiggle

drift

randomness

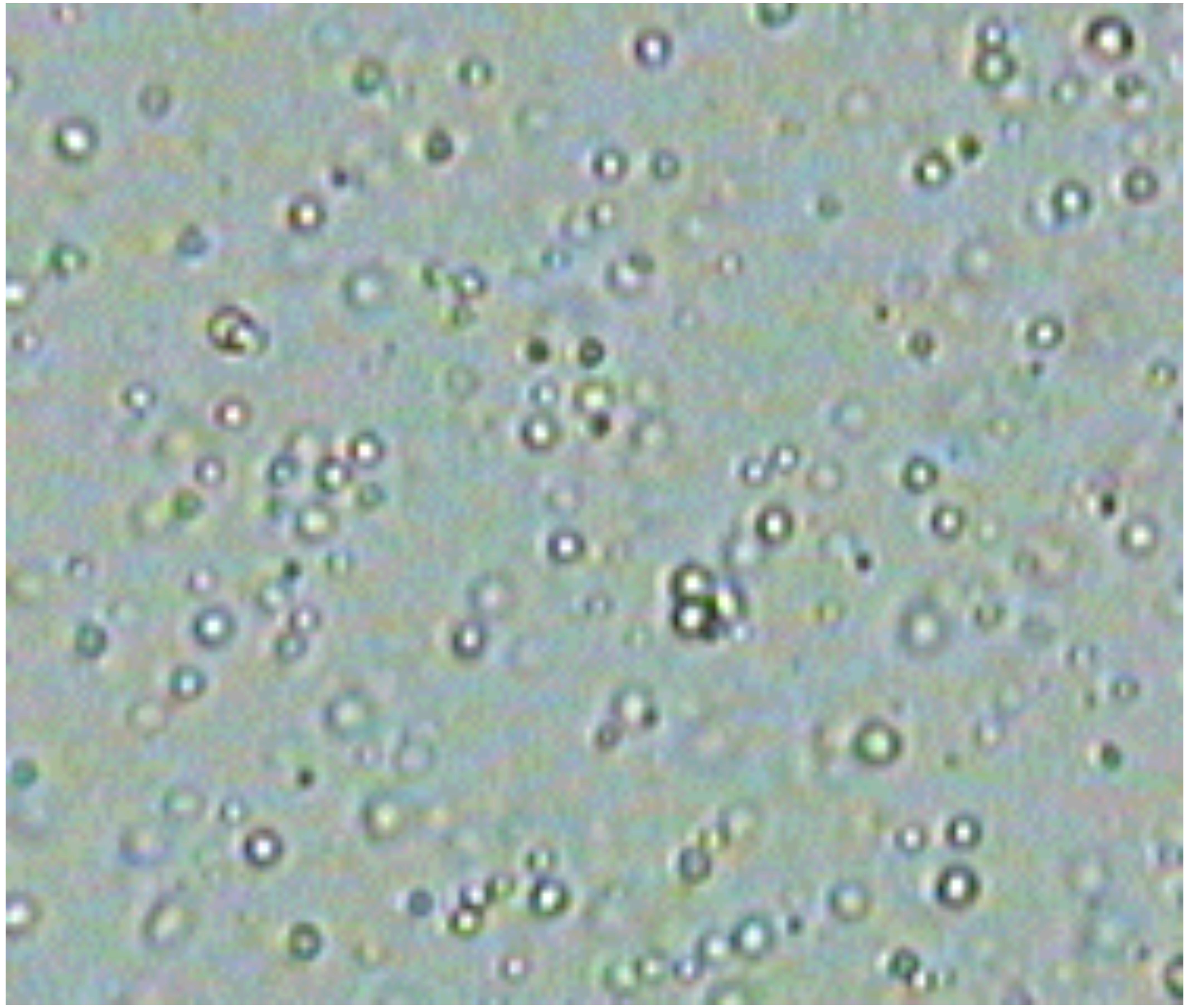


Stochastic Different equations (SDEs)



Stochastic Different equations (SDEs)

- When the particles are jiggling, we need to model & simulate the forces that induce Jiggling (“Langevin dynamics”)



Langevin dynamics

Langevin dynamics

Extends Brownian motion by adding a *drift term* that represents a deterministic force

Langevin dynamics

Extends Brownian motion by adding a *drift term* that represents a deterministic force

$$d\mathbf{x}_t = \ \sigma(\mathbf{x}_t, t) dW_t$$

Langevin dynamics

Extends Brownian motion by adding a *drift term* that represents a deterministic force

$$d\mathbf{x}_t = \overbrace{\sigma(\mathbf{x}_t, t)dW_t}^{\text{randomness}}$$

Langevin dynamics

Extends Brownian motion by adding a *drift term* that represents a deterministic force

$$d\mathbf{x}_t = \overset{\text{drift}}{\mu(\mathbf{x}_t, t)dt} + \overset{\text{randomness}}{\underline{\sigma(\mathbf{x}_t, t)dW_t}}$$

Langevin dynamics

Extends Brownian motion by adding a *drift term* that represents a deterministic force

$$d\mathbf{x}_t = \overbrace{\mu(\mathbf{x}_t, t)dt}^{\text{drift}} + \overbrace{\sigma(\mathbf{x}_t, t)dW_t}^{\text{randomness}}$$

Langevin dynamics

Extends Brownian motion by adding a *drift term* that represents a deterministic force

$$d\mathbf{x}_t = \overset{\text{drift}}{\overline{\mu(\mathbf{x}_t, t)dt}} + \overset{\text{randomness}}{\overline{\sigma(\mathbf{x}_t, t)dW_t}}$$

Langevin dynamics

Extends Brownian motion by adding a *drift term* that represents a deterministic force

$$d\mathbf{x}_t = \overbrace{\mu(\mathbf{x}_t, t)dt}^{\text{drift}} + \overbrace{\sigma(\mathbf{x}_t, t)dW_t}^{\text{randomness}}$$

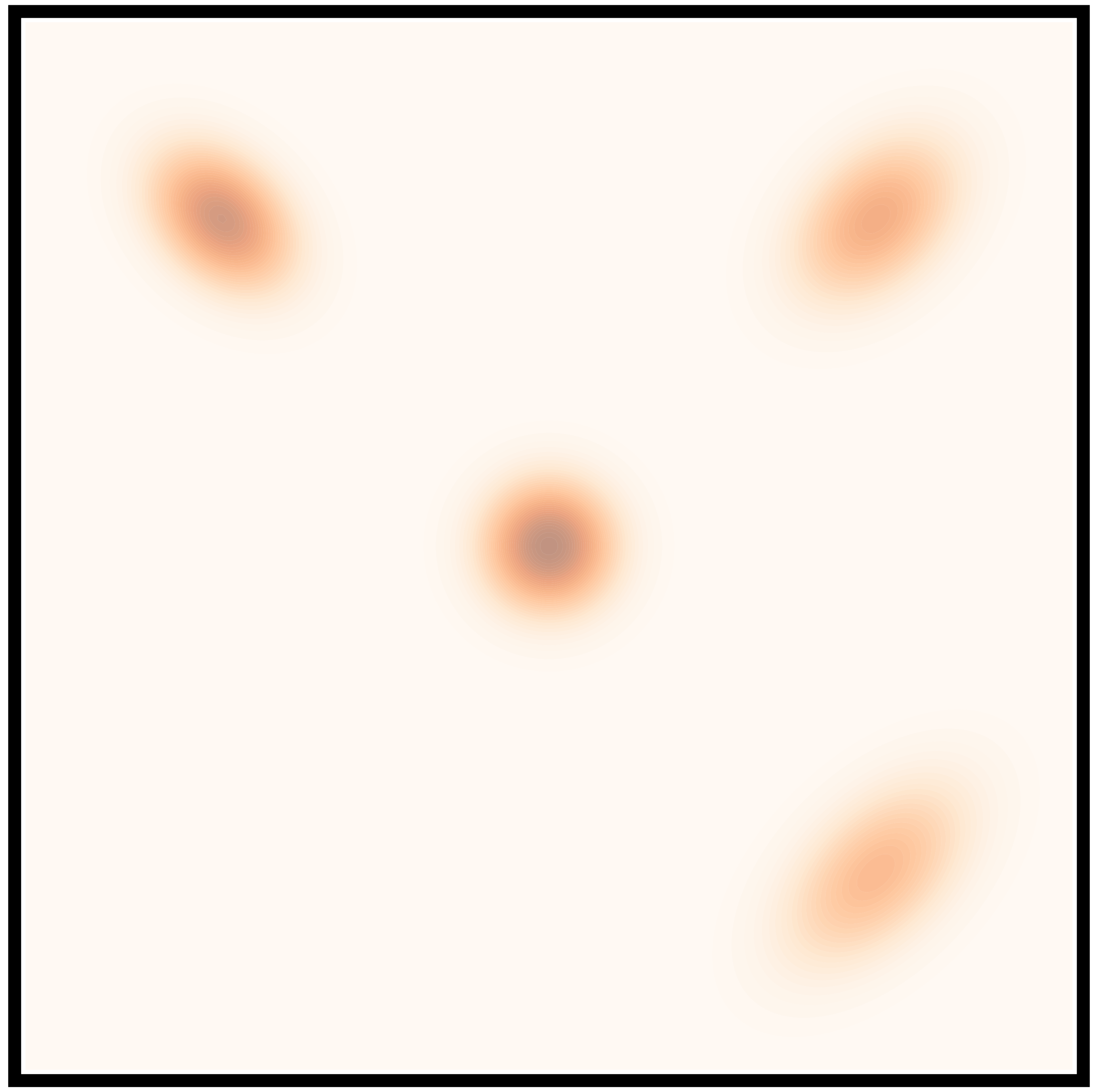
$$d\mathbf{x}_t = \nabla \log p(\mathbf{x}) + \sqrt{2} dW_t$$

Langevin dynamics

Extends Brownian motion by adding a *drift term* that represents a deterministic force

$$d\mathbf{x}_t = \overbrace{\mu(\mathbf{x}_t, t)dt}^{\text{drift}} + \overbrace{\sigma(\mathbf{x}_t, t)dW_t}^{\text{randomness}}$$

$$d\mathbf{x}_t = \nabla \log p(\mathbf{x}) + \sqrt{2} dW_t$$

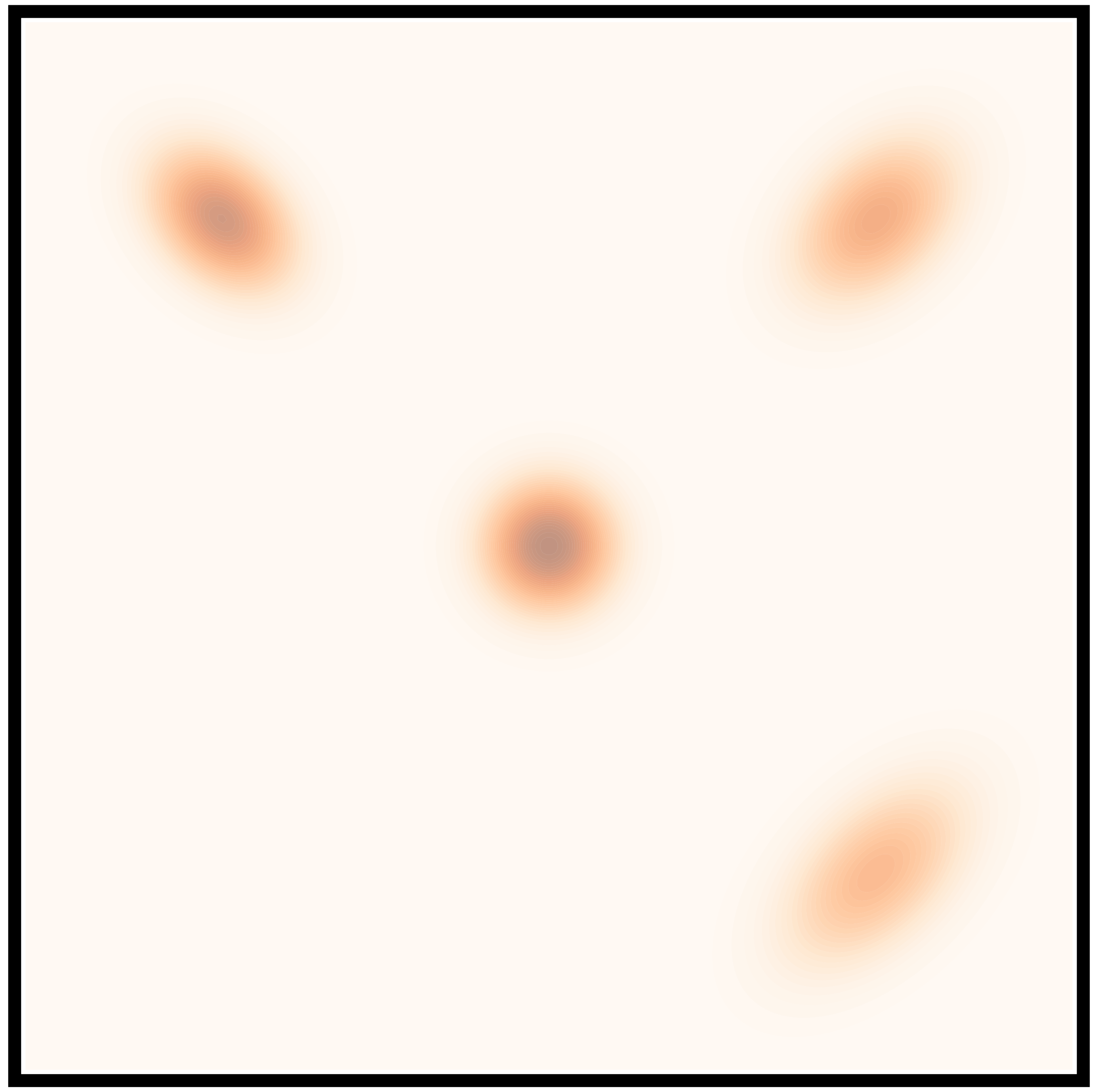


Target distribution

Langevin dynamics

Extends Brownian motion by adding a *drift term* that represents a deterministic force

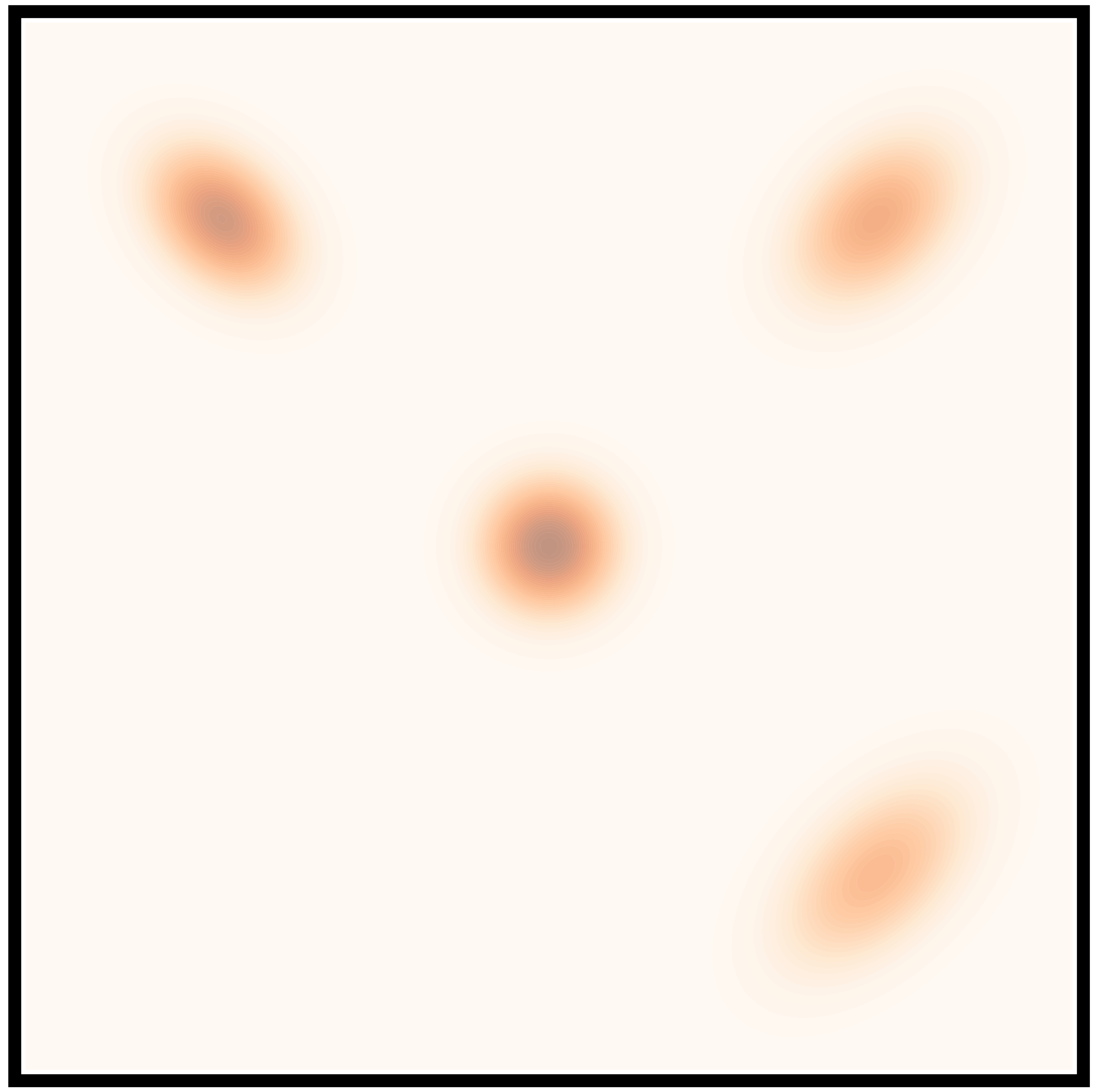
$$d\mathbf{x}_t = \overbrace{\nabla \log p(\mathbf{x})}^{\text{drift}} + \overbrace{\sqrt{2}dW_t}^{\text{randomness}}$$



Target distribution

Simulating Langevin diffusion

$$d\mathbf{x}_t = \nabla \log p(\mathbf{x}) + \sqrt{2} dW_t$$

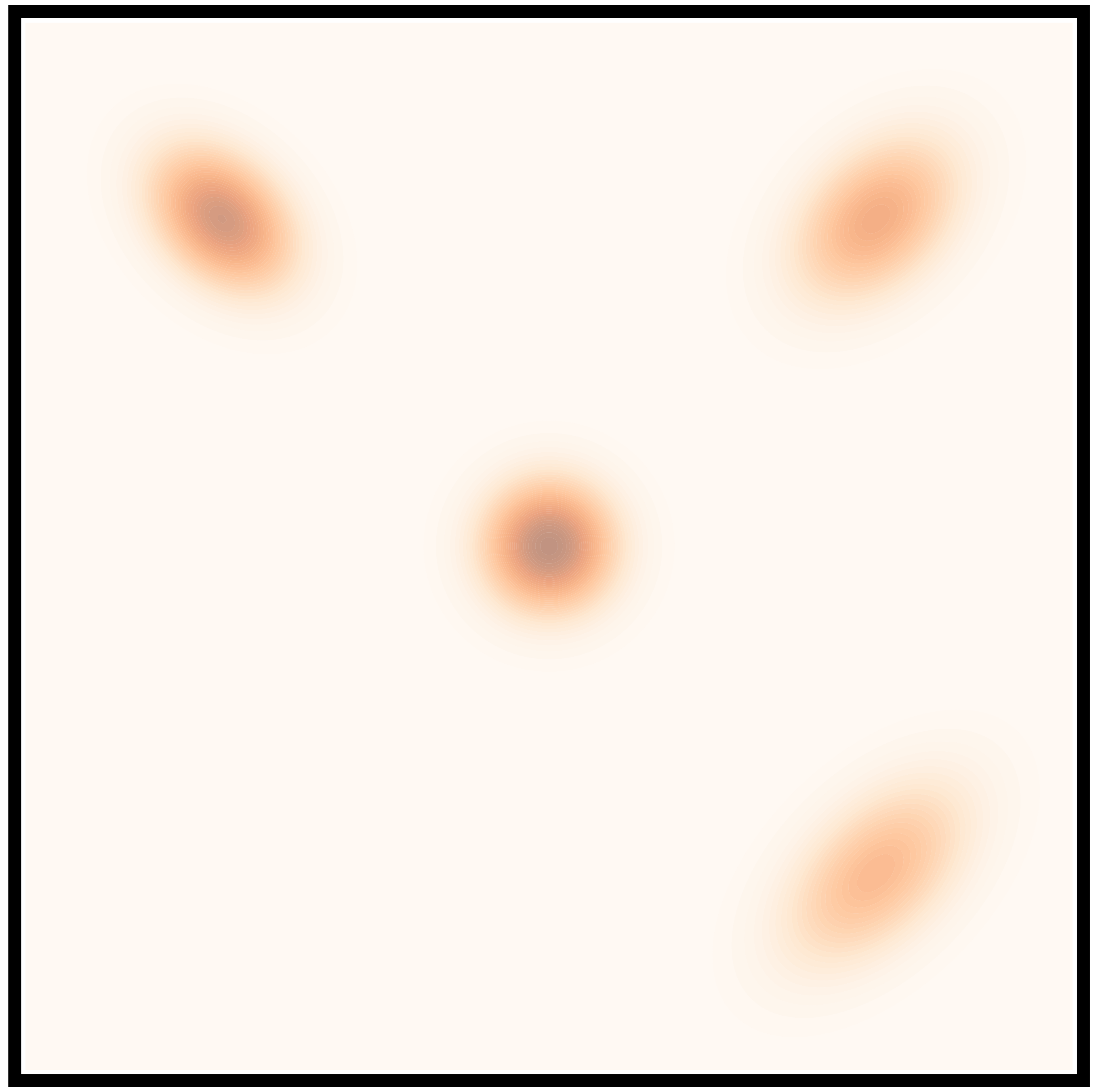


$p(\mathbf{x})$

Simulating Langevin diffusion

$$d\mathbf{x}_t = \nabla \log p(\mathbf{x}) + \sqrt{2} dW_t$$

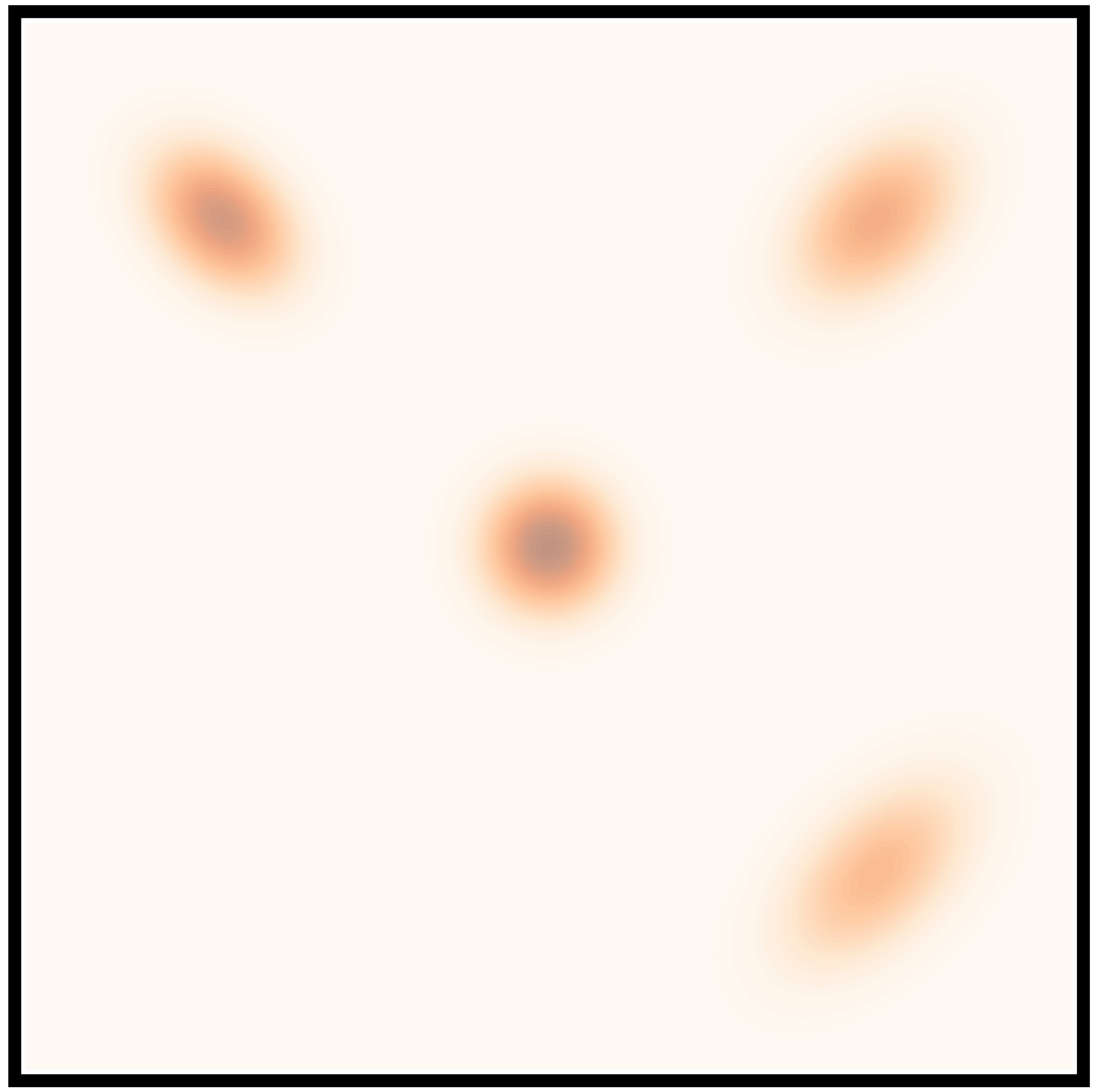
Euler-Maruyama method to simulate Langevin diffusion:



$p(\mathbf{x})$

Simulating Langevin diffusion

$$d\mathbf{x}_t = \nabla \log p(\mathbf{x}) + \sqrt{2} dW_t$$



$p(\mathbf{x})$

Euler-Maruyama method to simulate Langevin diffusion:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \tau \nabla \log p(\mathbf{x}) + \sqrt{2\tau}\xi$$

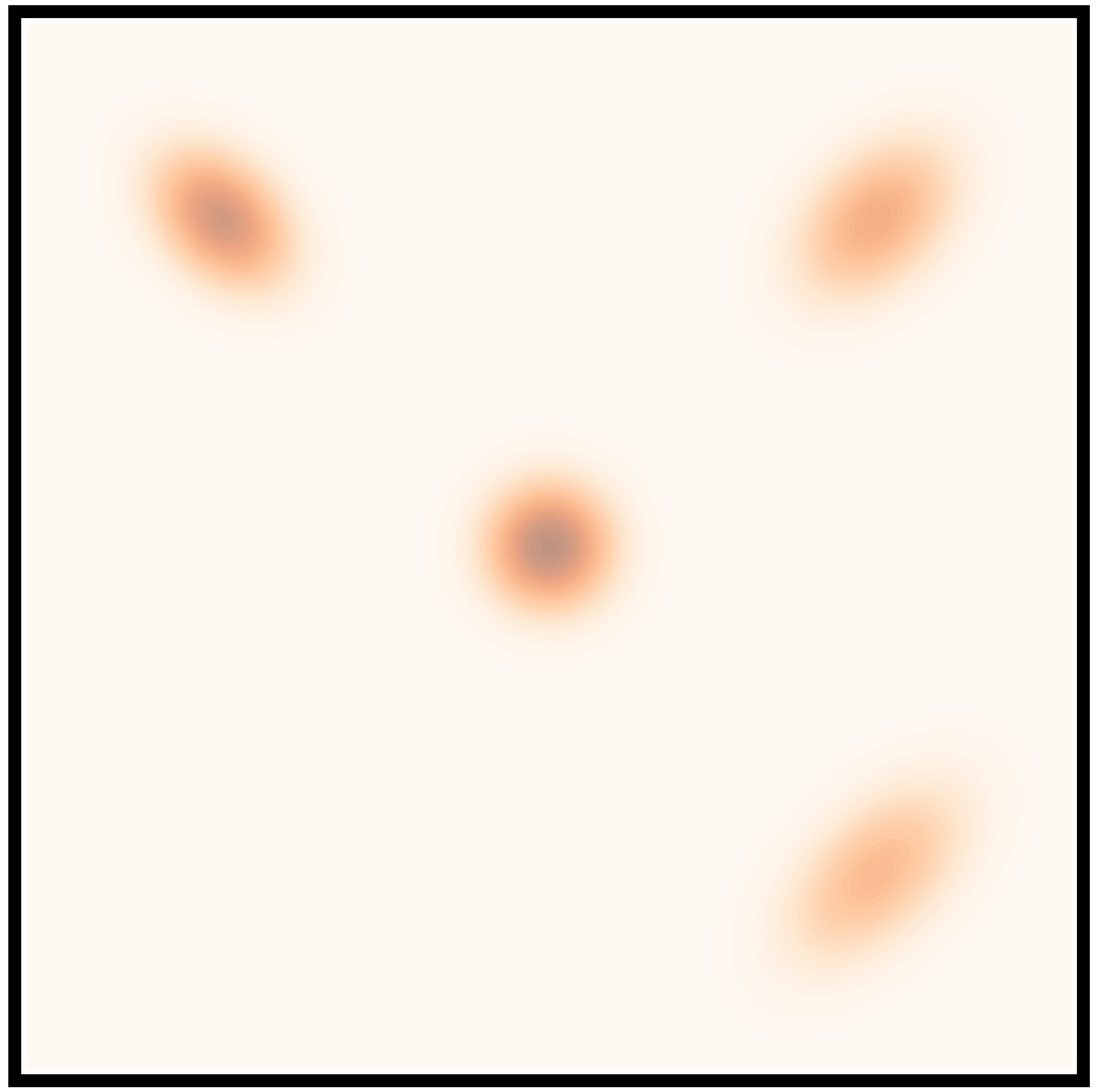
Simulating Langevin diffusion

$$d\mathbf{x}_t = \nabla \log p(\mathbf{x}) + \sqrt{2} dW_t$$

Euler-Maruyama method to simulate Langevin diffusion:

Step size

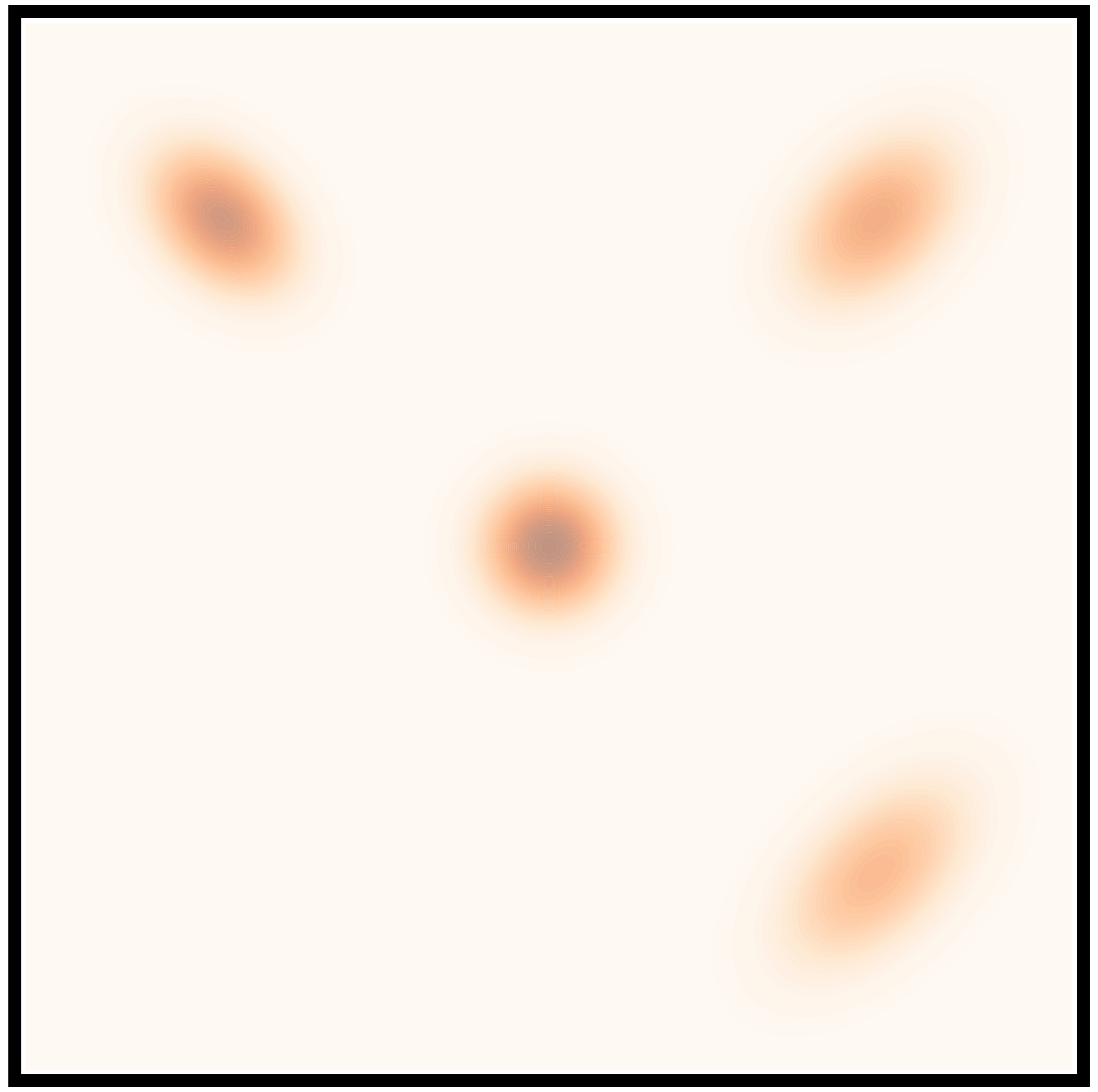
$$\mathbf{x}_{t+1} = \mathbf{x}_t + \tau \nabla \log p(\mathbf{x}) + \sqrt{2\tau}\xi$$



$p(\mathbf{x})$

Simulating Langevin diffusion

$$d\mathbf{x}_t = \nabla \log p(\mathbf{x}) + \sqrt{2} dW_t$$



$p(\mathbf{x})$

Euler-Maruyama method to simulate Langevin diffusion:

Step size

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \tau \nabla \log p(\mathbf{x}) + \sqrt{2\tau}\xi$$

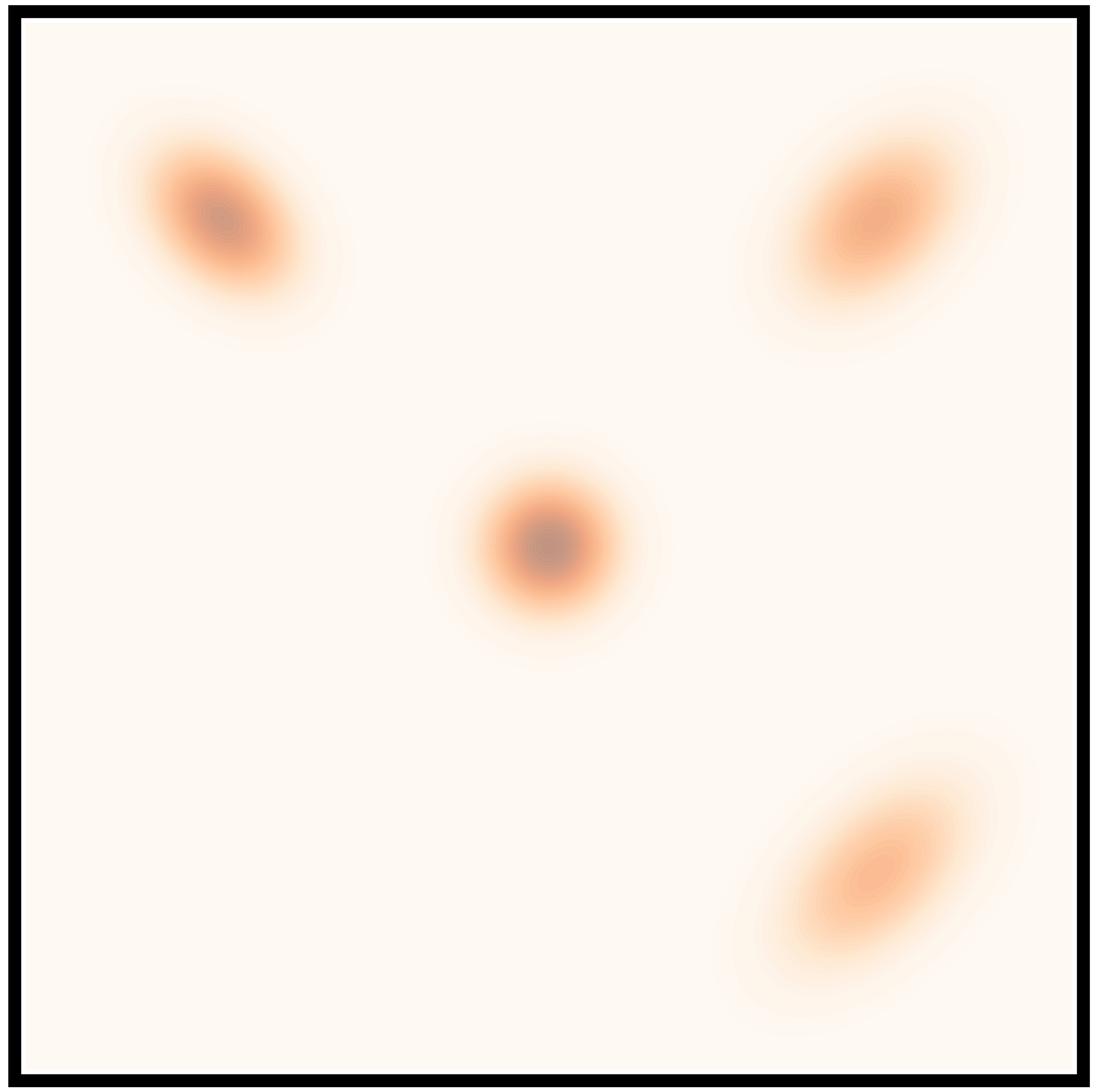
Gaussian noise

Simulating Langevin diffusion

$$d\mathbf{x}_t = \nabla \log p(\mathbf{x}) + \sqrt{2} dW_t$$

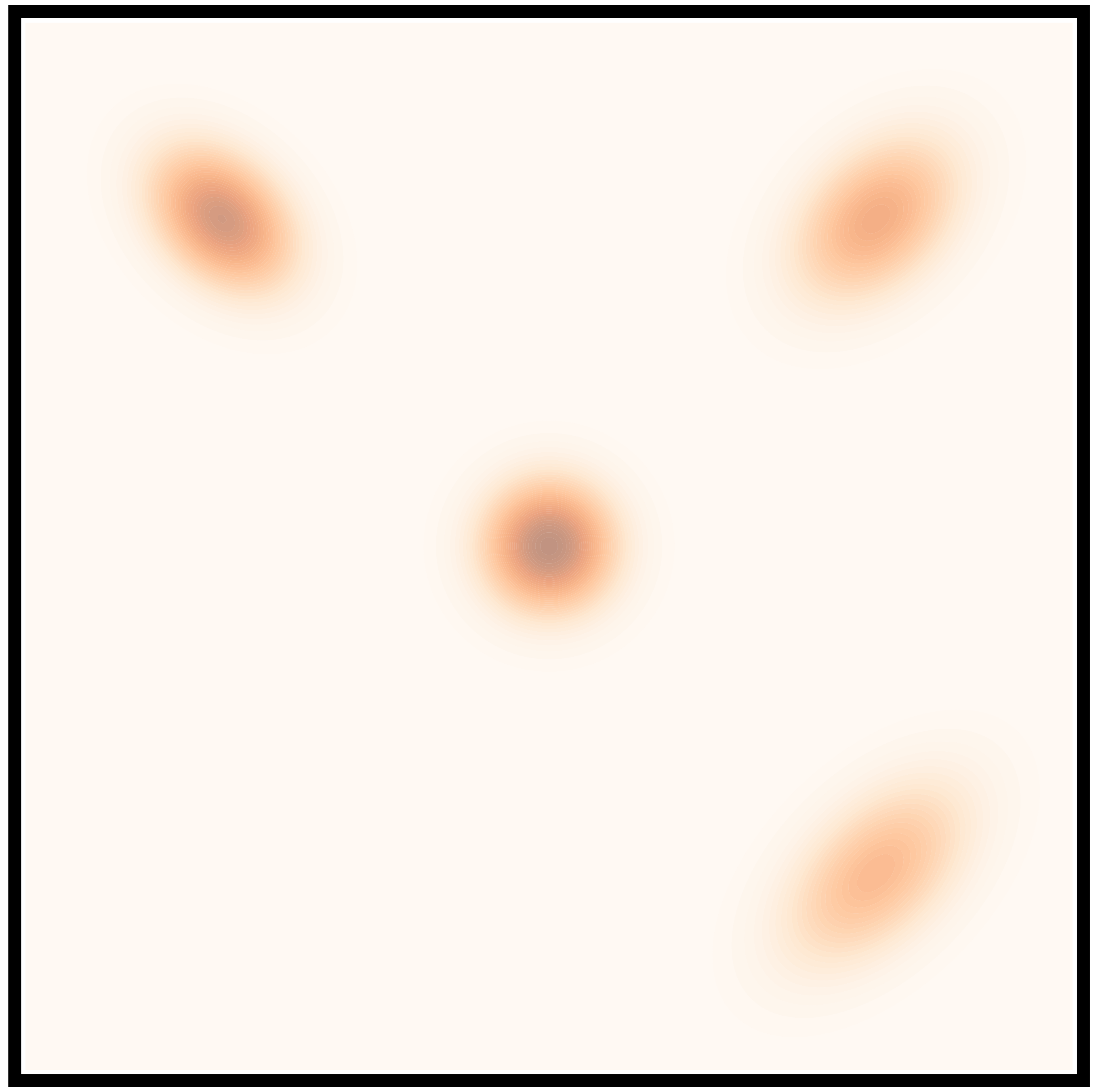
Euler-Maruyama method to simulate Langevin diffusion:

$$\mathbf{x}_{t+1} = \underline{\mathbf{x}_t + \tau \nabla \log p(\mathbf{x})} + \underline{\sqrt{2\tau}\xi}$$



Simulating Langevin diffusion

$$d\mathbf{x}_t = \nabla \log p(\mathbf{x}) + \sqrt{2} dW_t$$

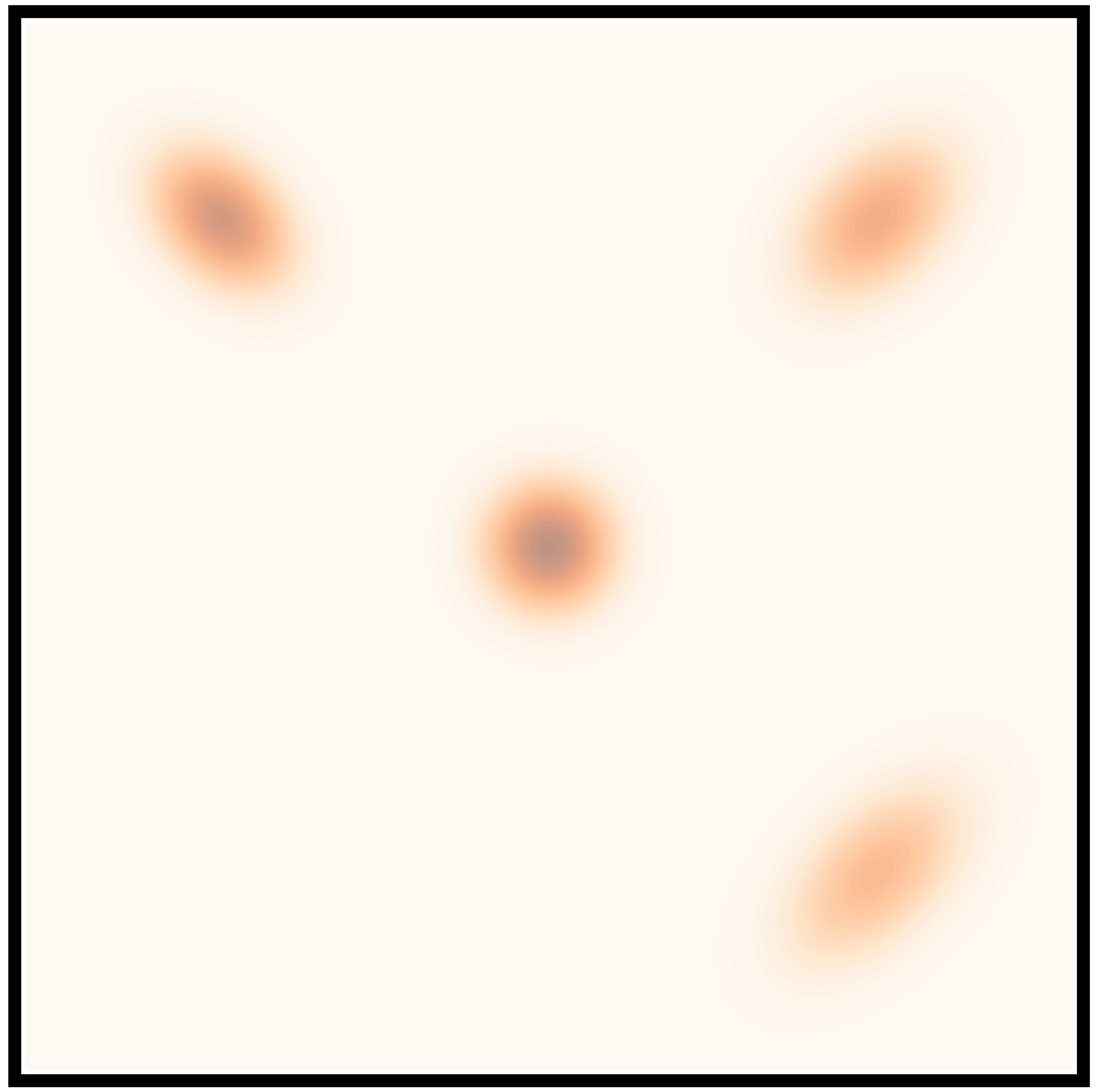


Euler-Maruyama method to simulate Langevin diffusion:

$$\mathbf{x}_{t+1} = \underbrace{\mathbf{x}_t + \tau \nabla \log p(\mathbf{x})}_{\text{Mean of the gaussian}} + \underbrace{\sqrt{2\tau}\xi}_{\text{Covariance}}$$

Simulating Langevin diffusion

$$d\mathbf{x}_t = \nabla \log p(\mathbf{x}) + \sqrt{2} dW_t$$

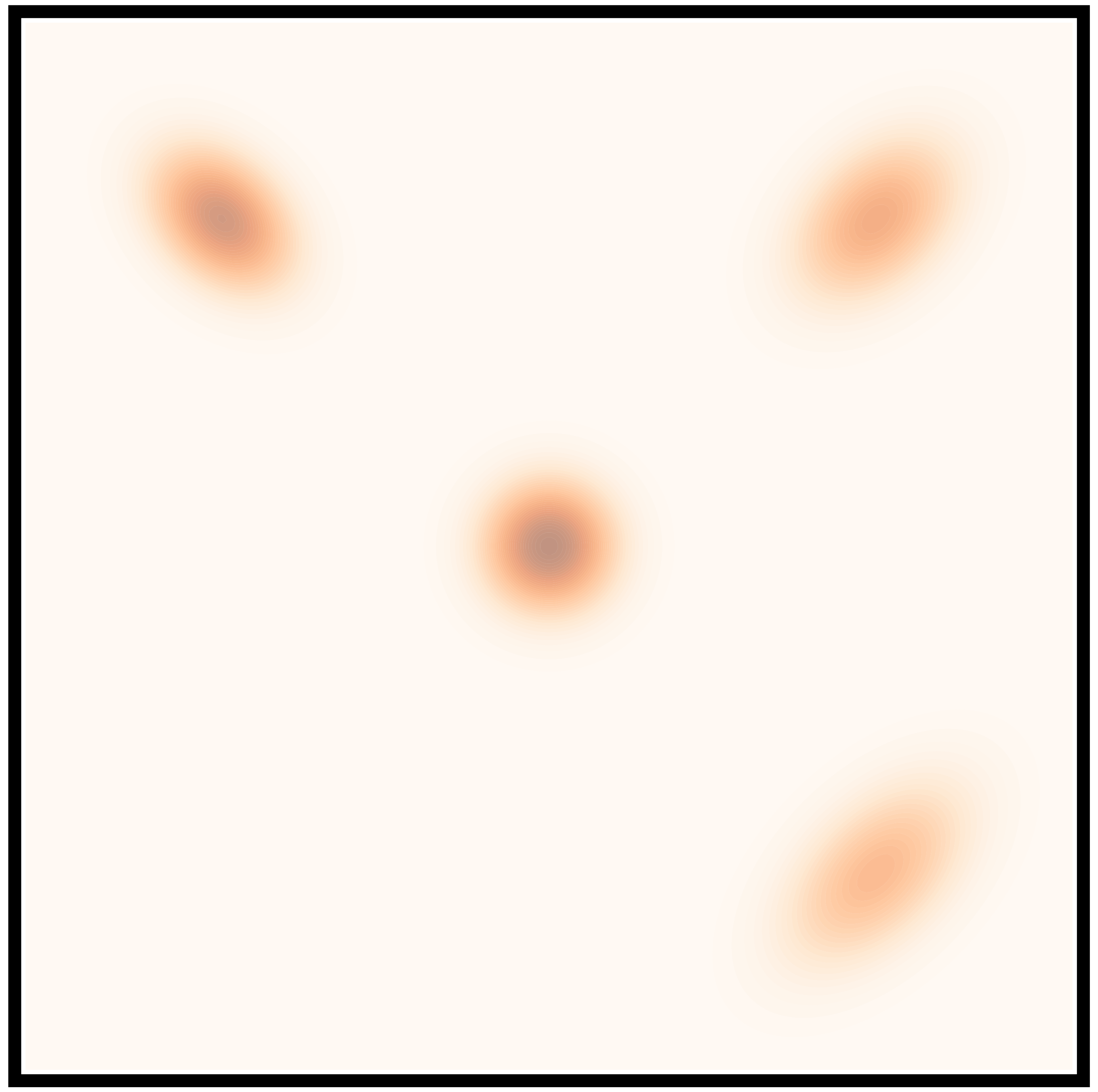


Euler-Maruyama method to simulate Langevin diffusion:

$$\mathbf{x}_{t+1} = \underline{\mathbf{x}_t + \tau \nabla \log p(\mathbf{x})} + \underline{\sqrt{2\tau}\xi}$$

Simulating Langevin diffusion

$$d\mathbf{x}_t = \nabla \log p(\mathbf{x}) + \sqrt{2} dW_t$$



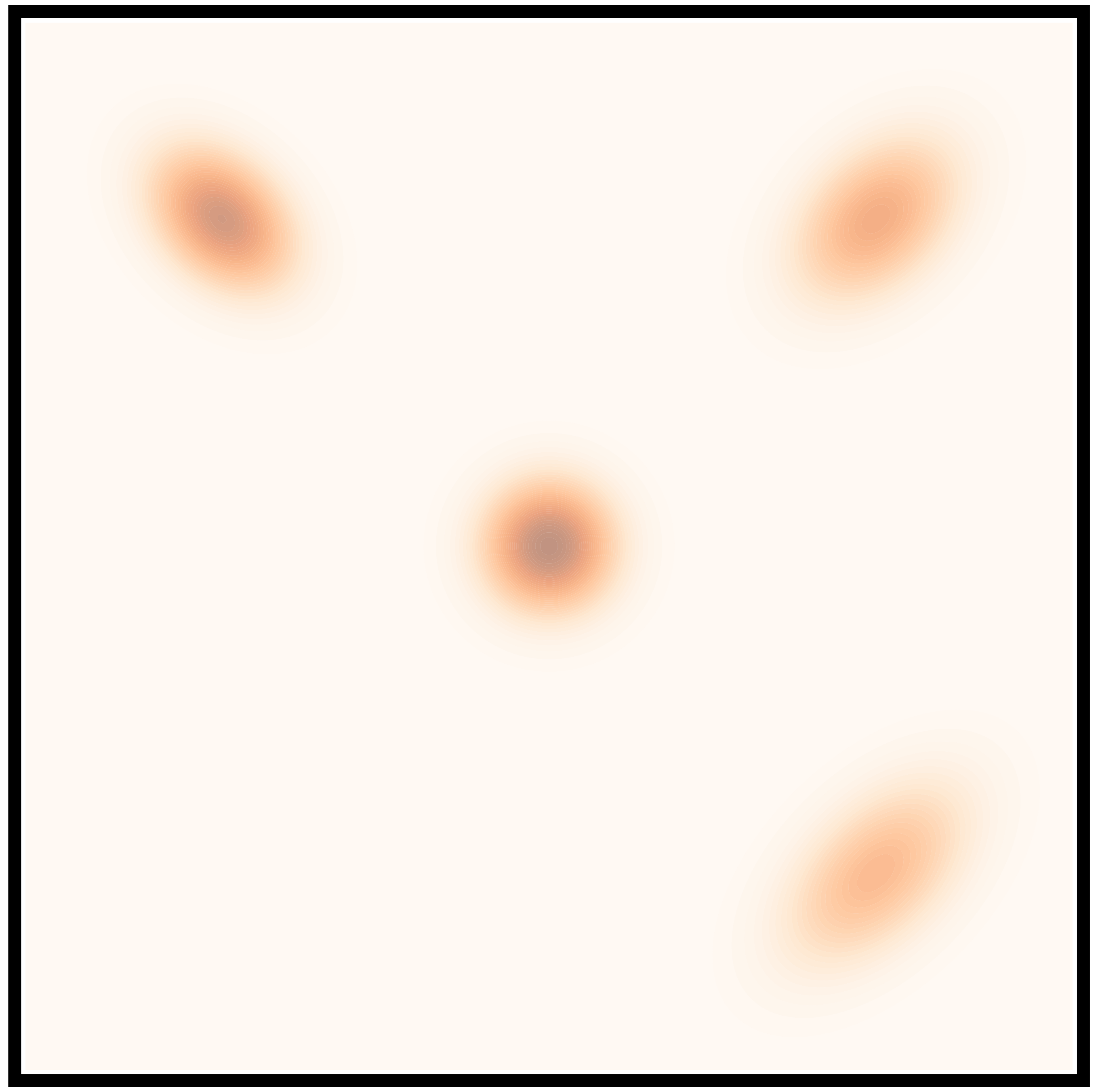
Euler-Maruyama method to simulate Langevin diffusion:

$$\mathbf{x}_{t+1} = \underline{\mathbf{x}_t + \tau \nabla \log p(\mathbf{x})} + \underline{\sqrt{2\tau}\xi}$$

Metropolis-adjusted Langevin update (MALA):

Simulating Langevin diffusion

$$d\mathbf{x}_t = \nabla \log p(\mathbf{x}) + \sqrt{2} dW_t$$



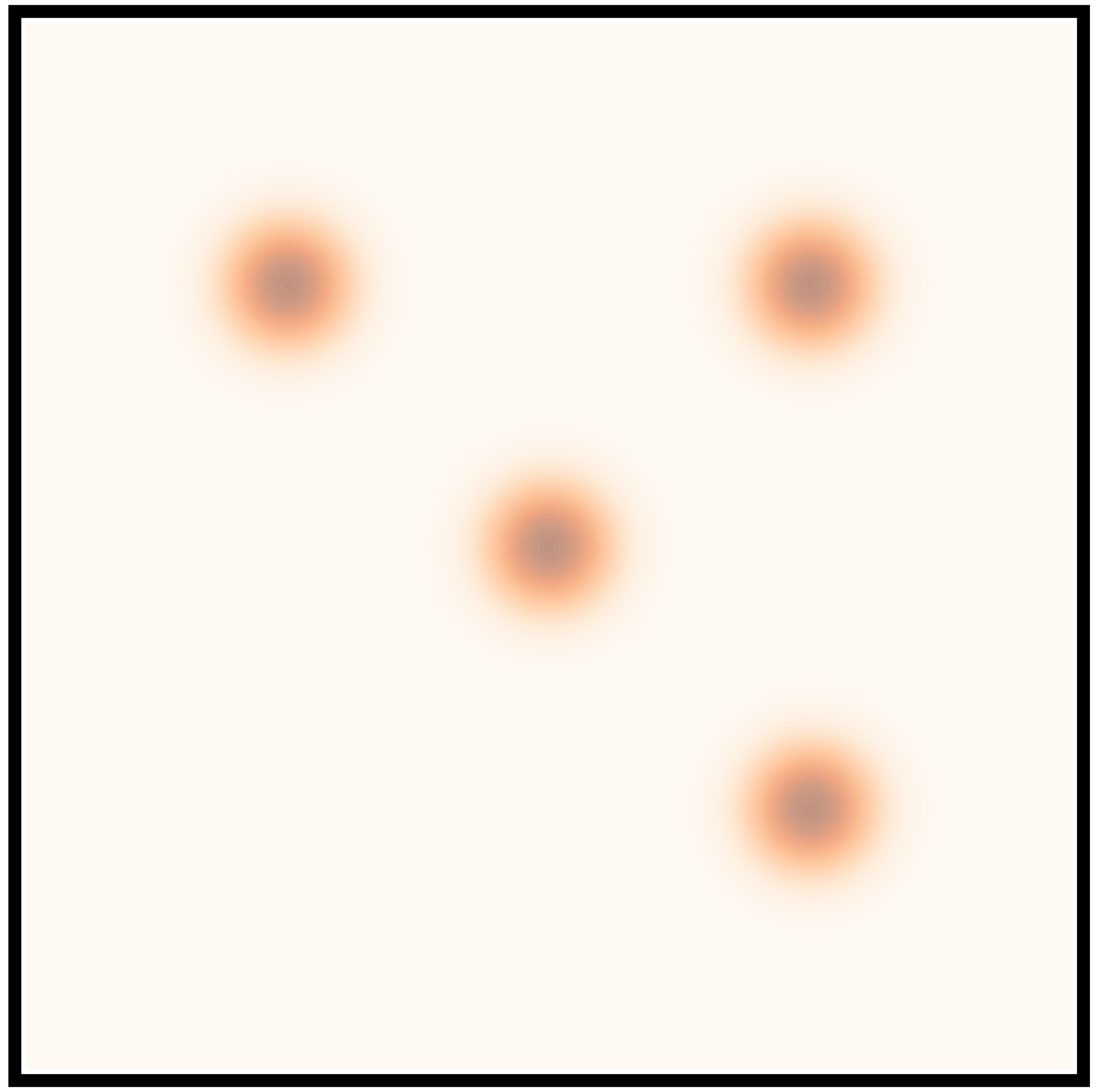
Euler-Maruyama method to simulate Langevin diffusion:

$$\mathbf{x}_{t+1} = \underline{\mathbf{x}_t + \tau \nabla \log p(\mathbf{x})} + \underline{\sqrt{2\tau}\xi}$$

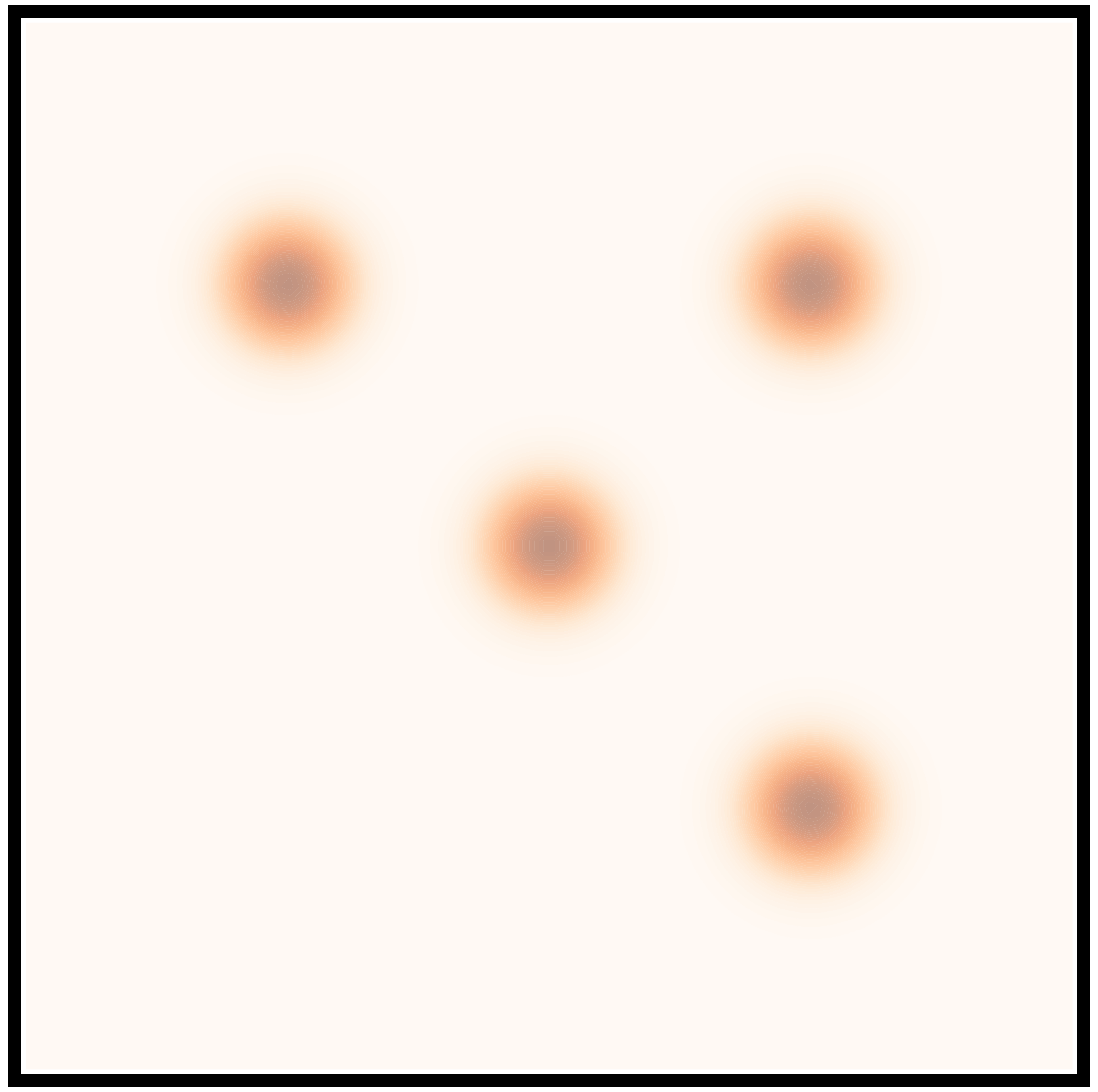
Metropolis-adjusted Langevin update (MALA):

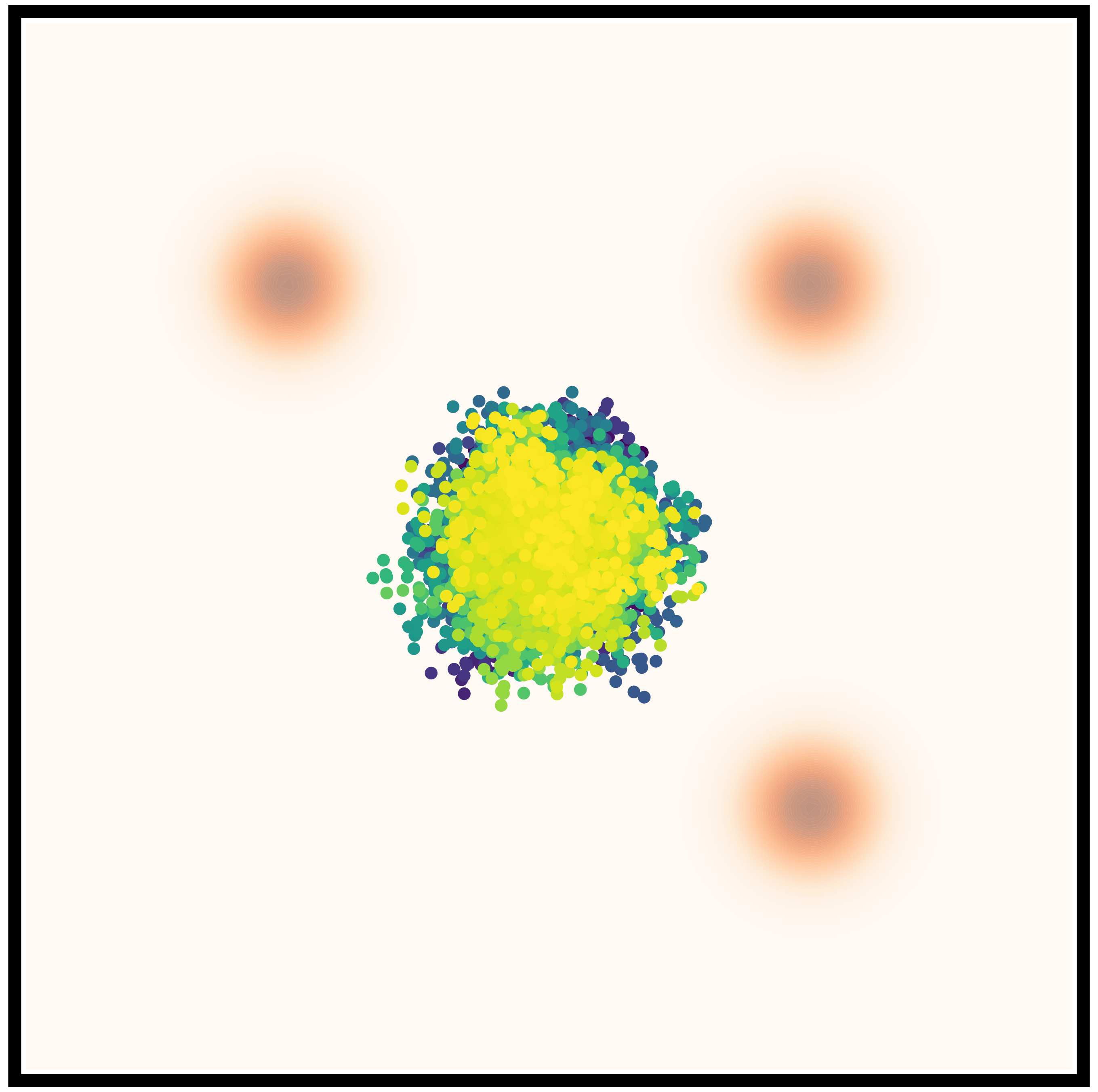
$\mathbf{x}_{t+1}$ is accepted based on the Metropolis-Hastings acceptance prob.

Langevin dynamics: Examples



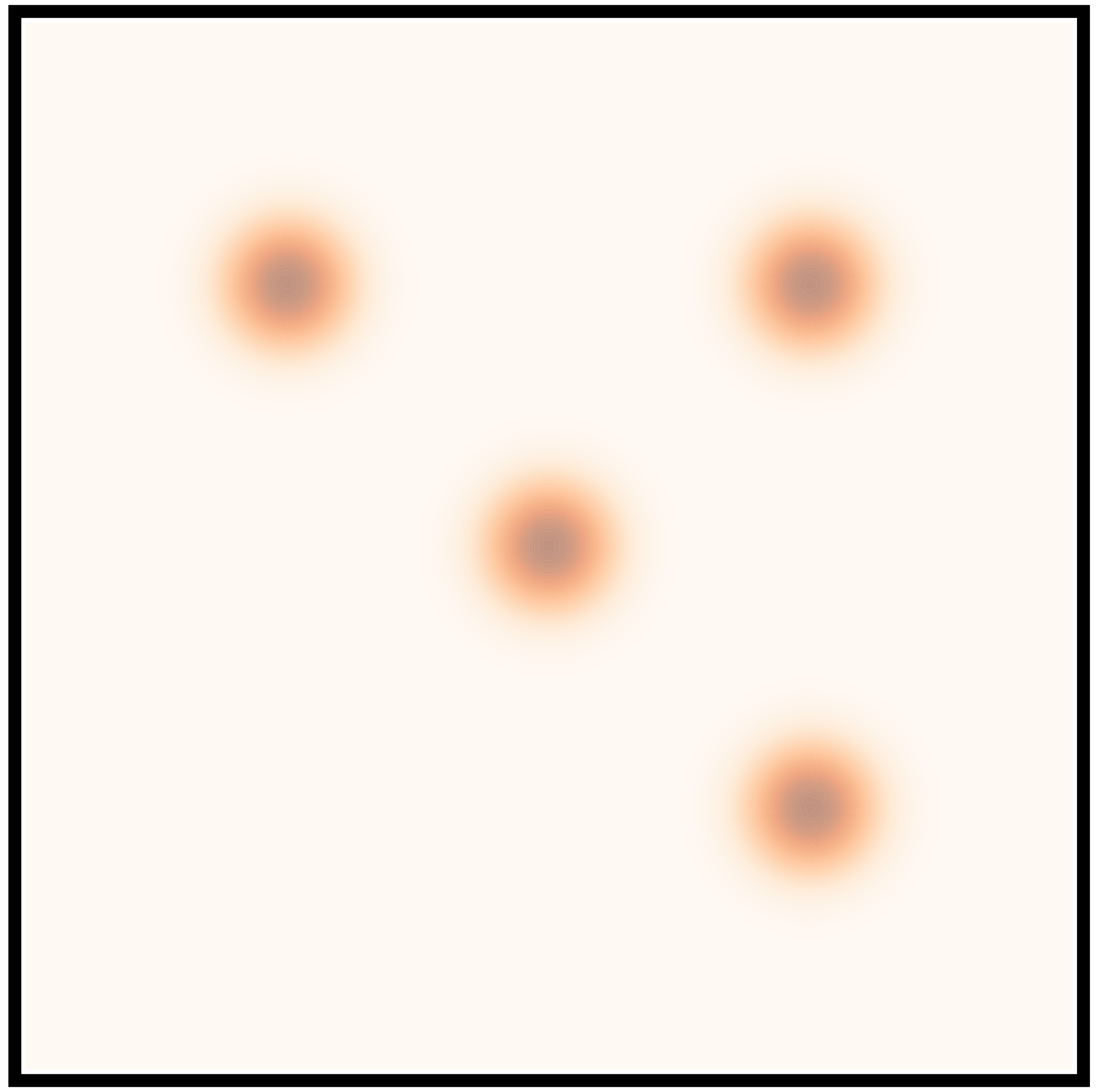
Langevin dynamics: Examples

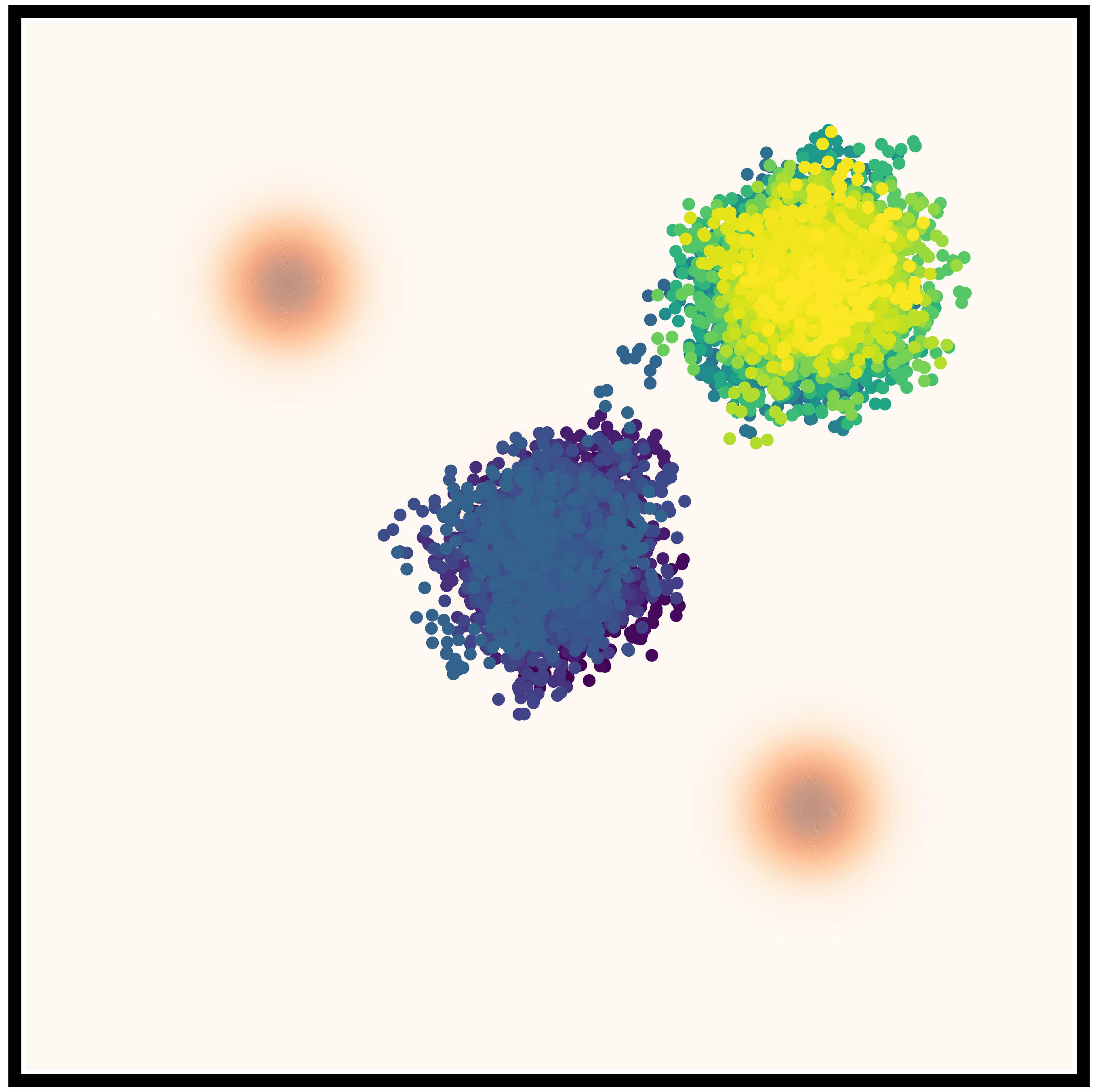




Step size τ=0.1

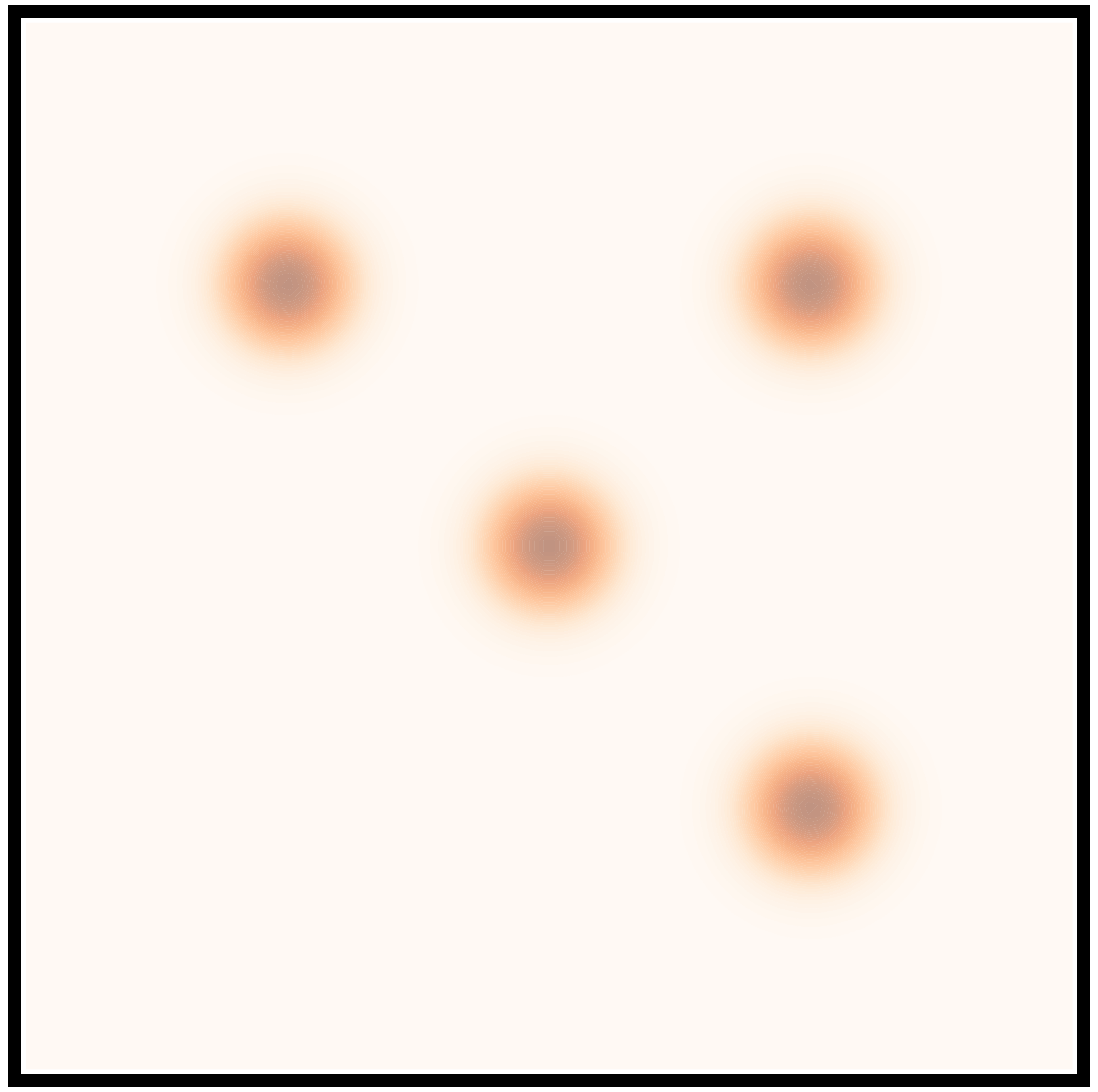
Langevin dynamics: Examples

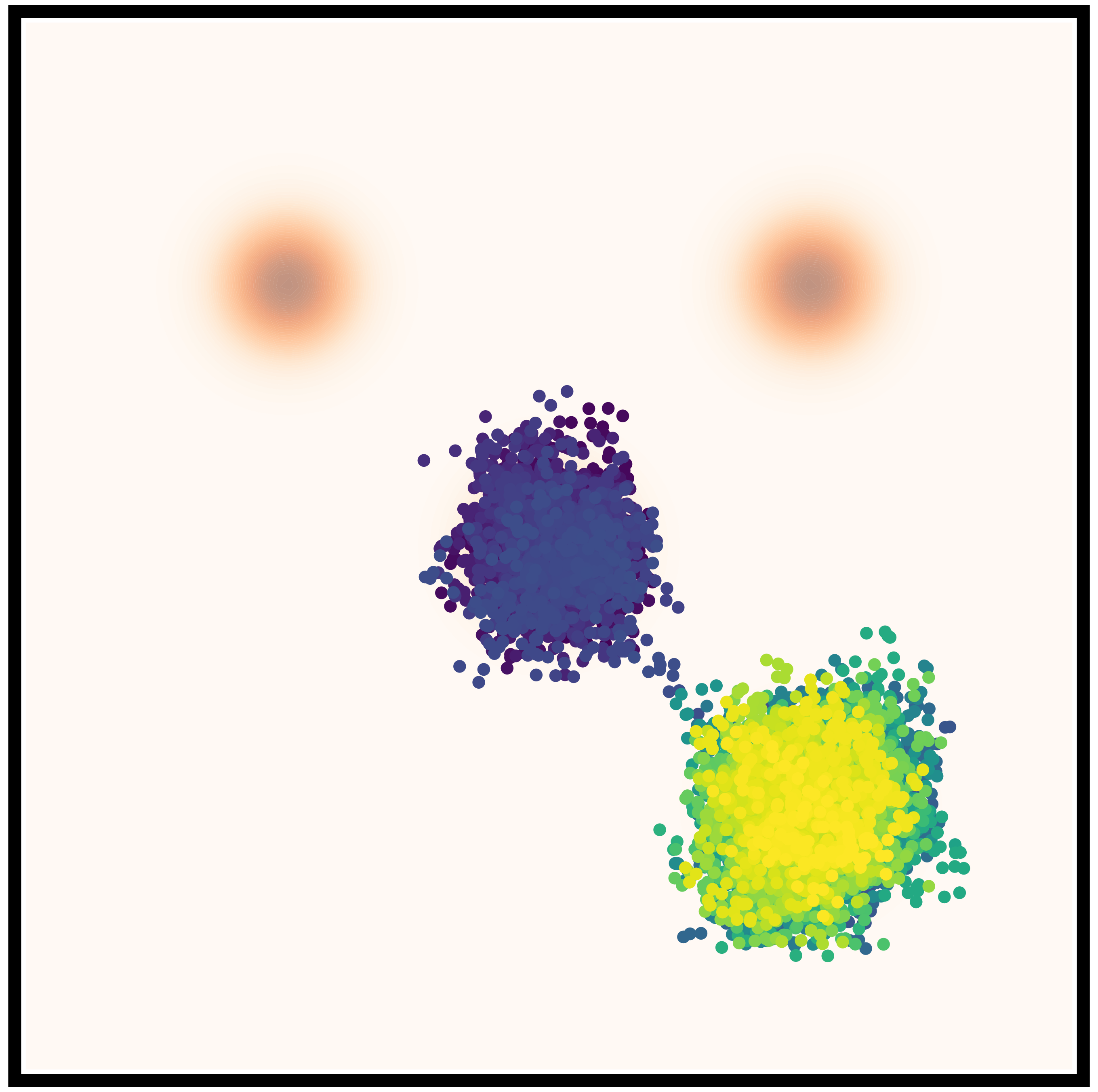




Step size τ=0.1

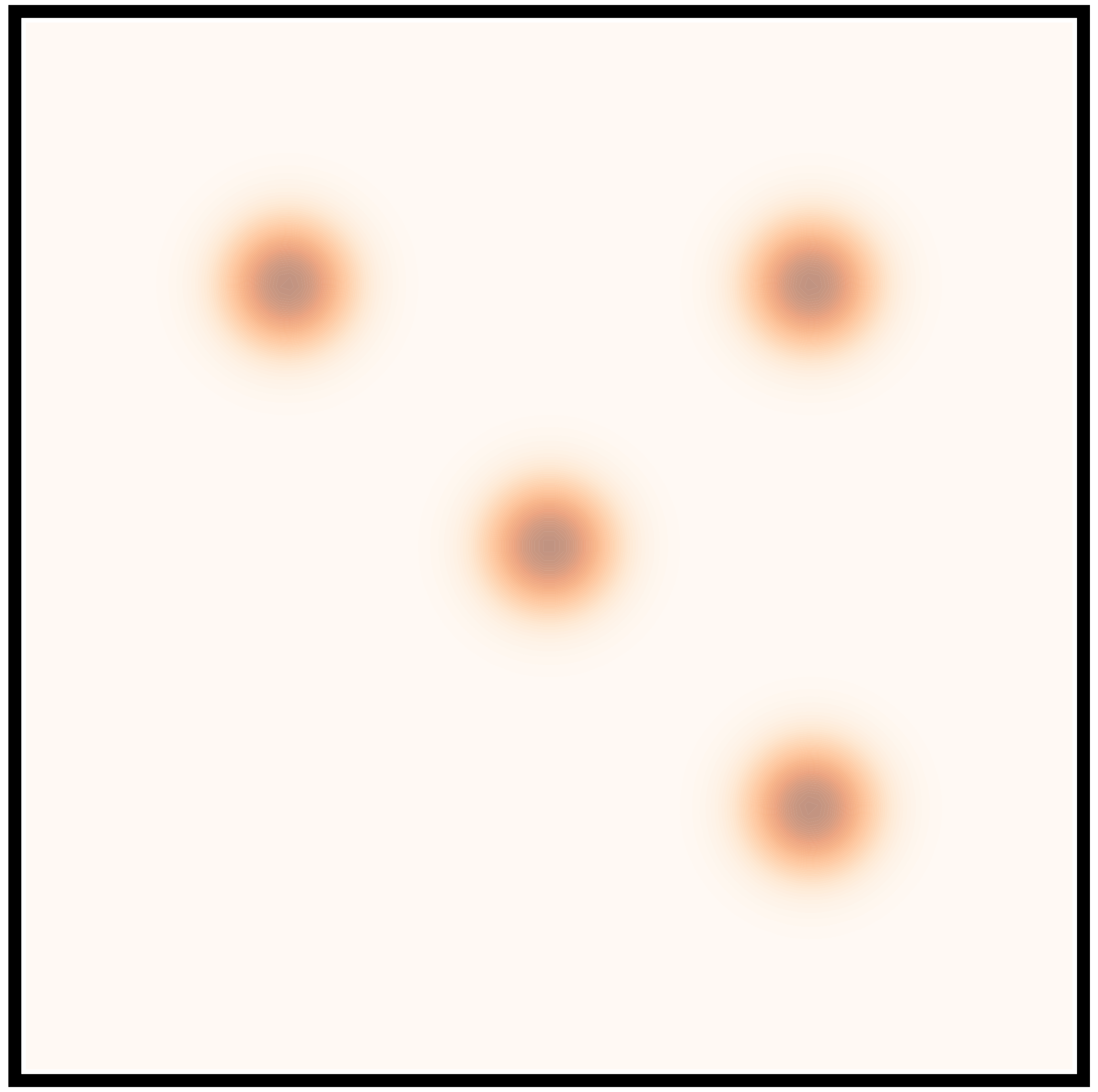
Langevin dynamics: Examples

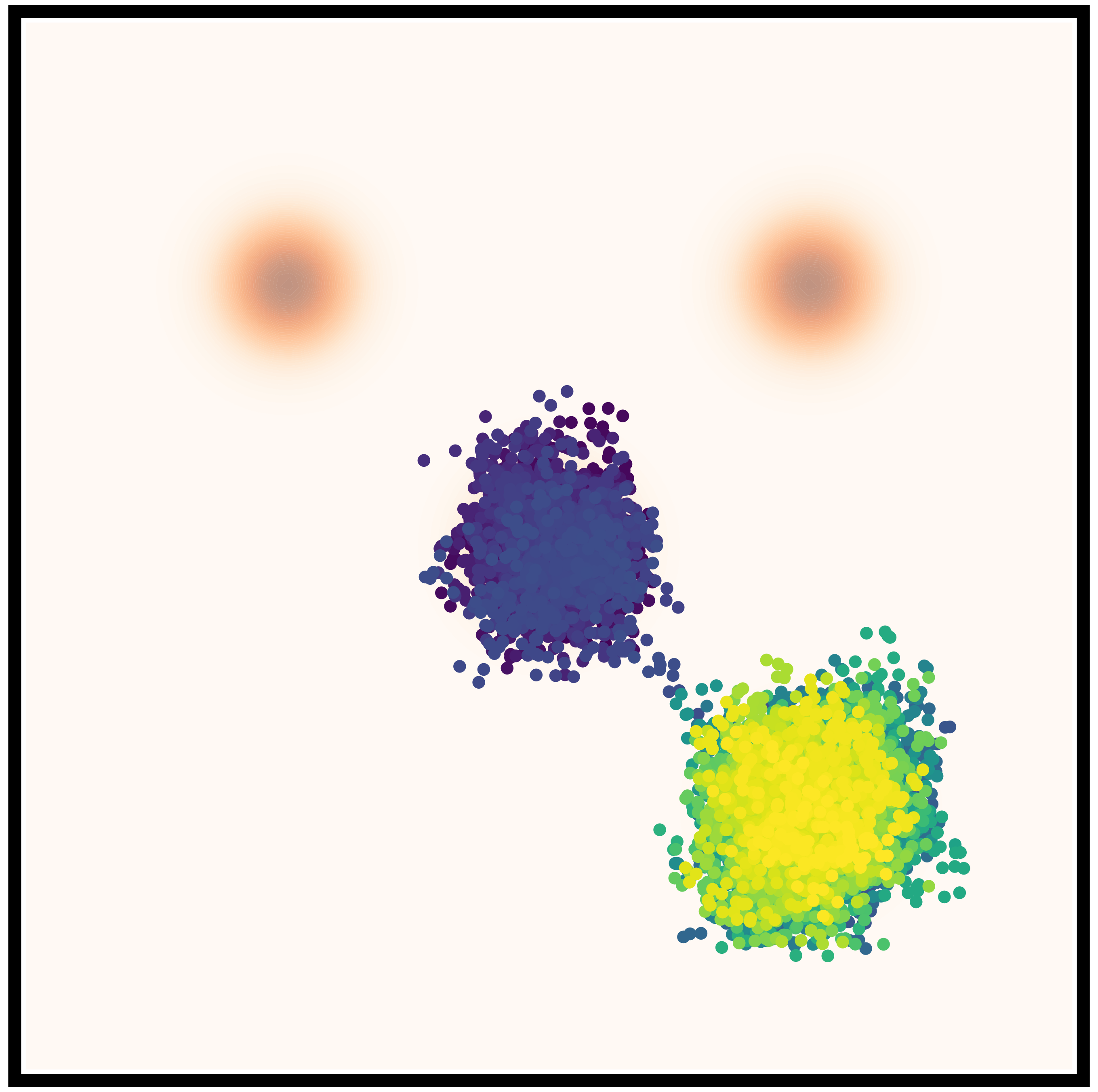




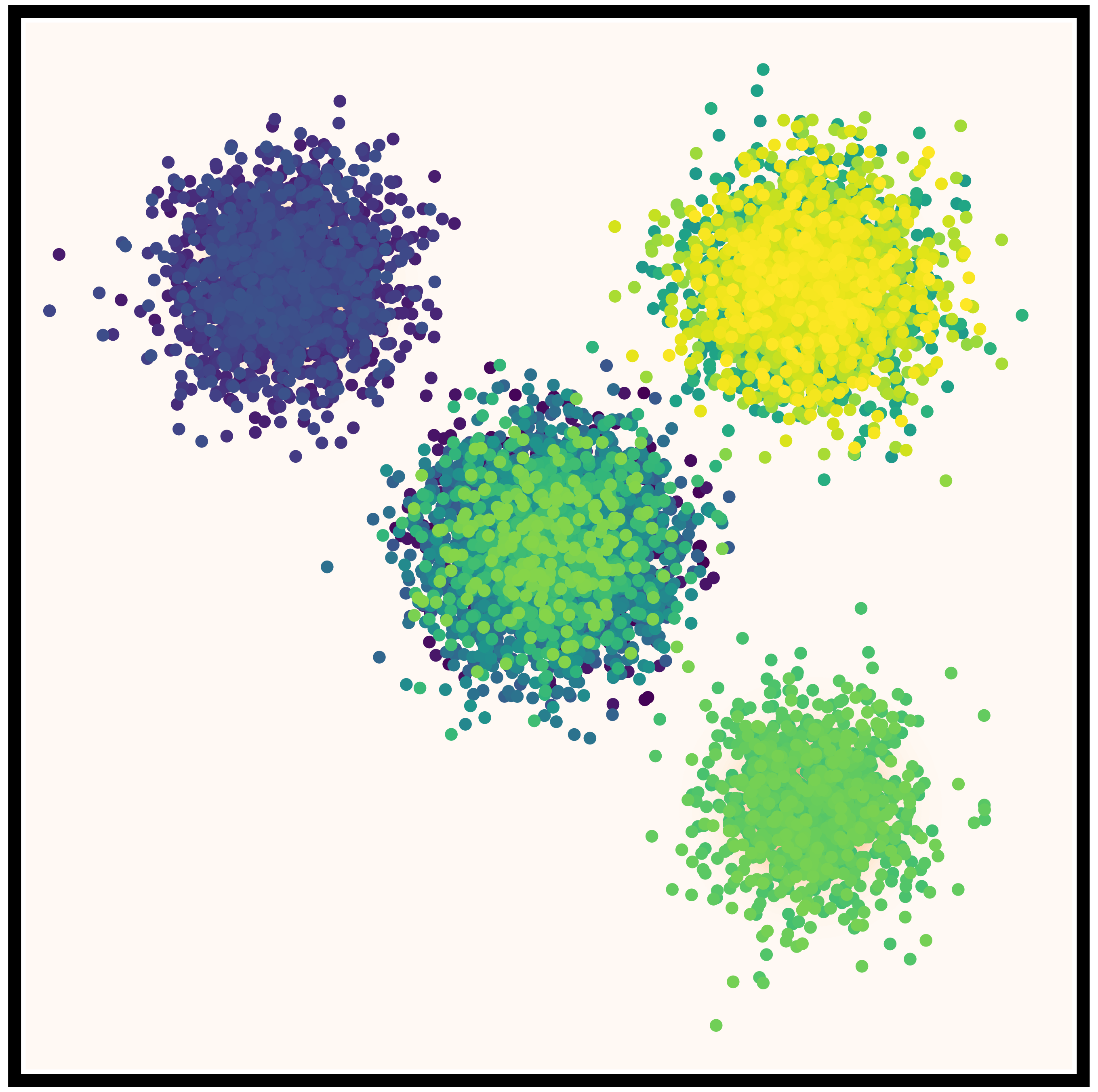
Step size τ=0.1

Langevin dynamics: Examples





Step size τ=0.1



Step size τ=1

Recap

Recap

- Introduced MCMC

Recap

- Introduced MCMC
- Introduced Stochastic Differential Equations (SDEs)

Recap

- Introduced MCMC
- Introduced Stochastic Differential Equations (SDEs)
- MCMC methods

Recap

- Introduced MCMC
- Introduced Stochastic Differential Equations (SDEs)
- MCMC methods
    - Metropolis-Hastings

Recap

- Introduced MCMC
- Introduced Stochastic Differential Equations (SDEs)
- MCMC methods
  - Metropolis-Hastings
- Brownian motion : a simple SDE

Recap

- Introduced MCMC
- Introduced Stochastic Differential Equations (SDEs)
- MCMC methods
    - Metropolis-Hastings
- Brownian motion : a simple SDE
- Langevin Dynamics

Applications

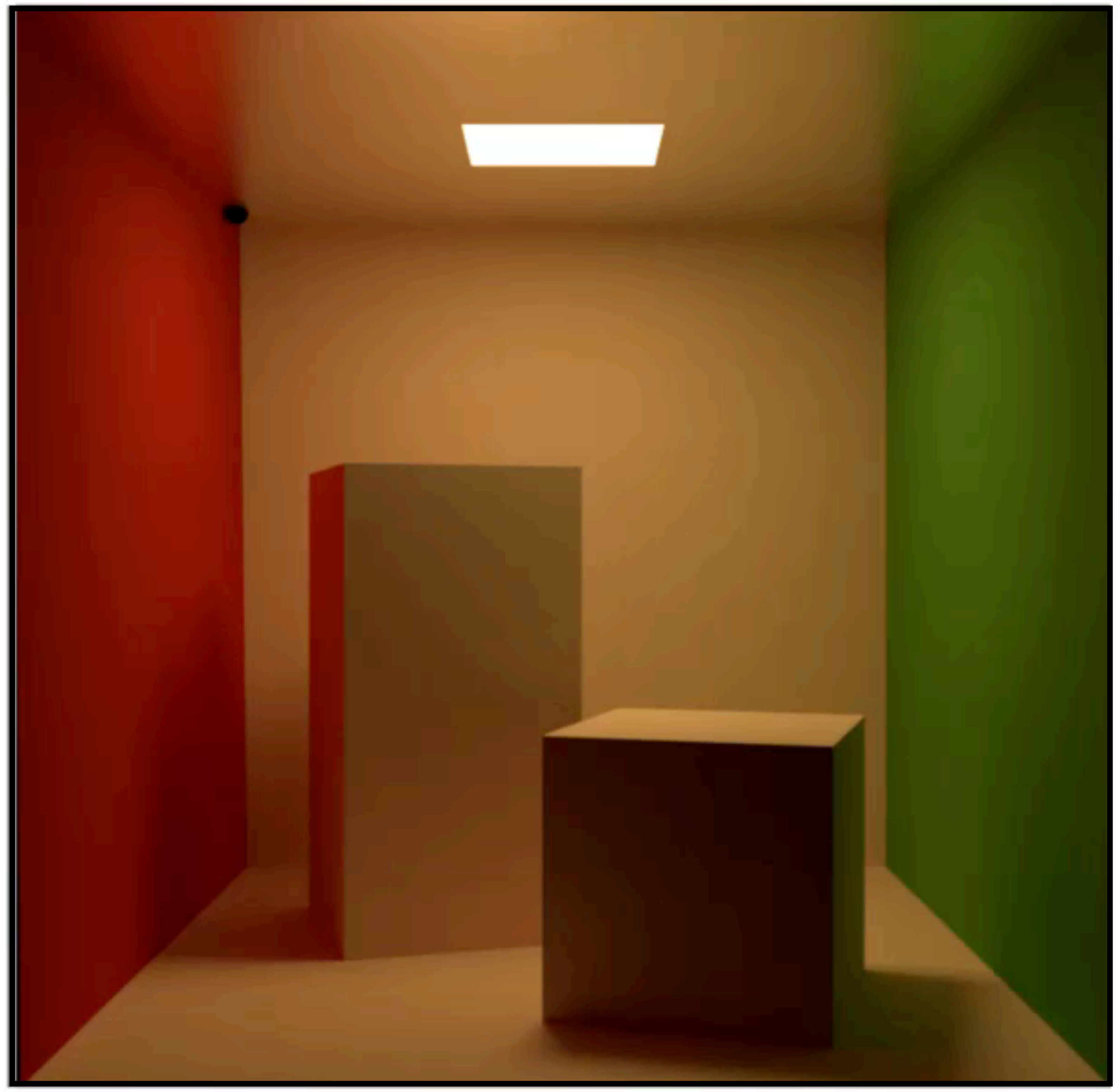
MCMC in Rendering

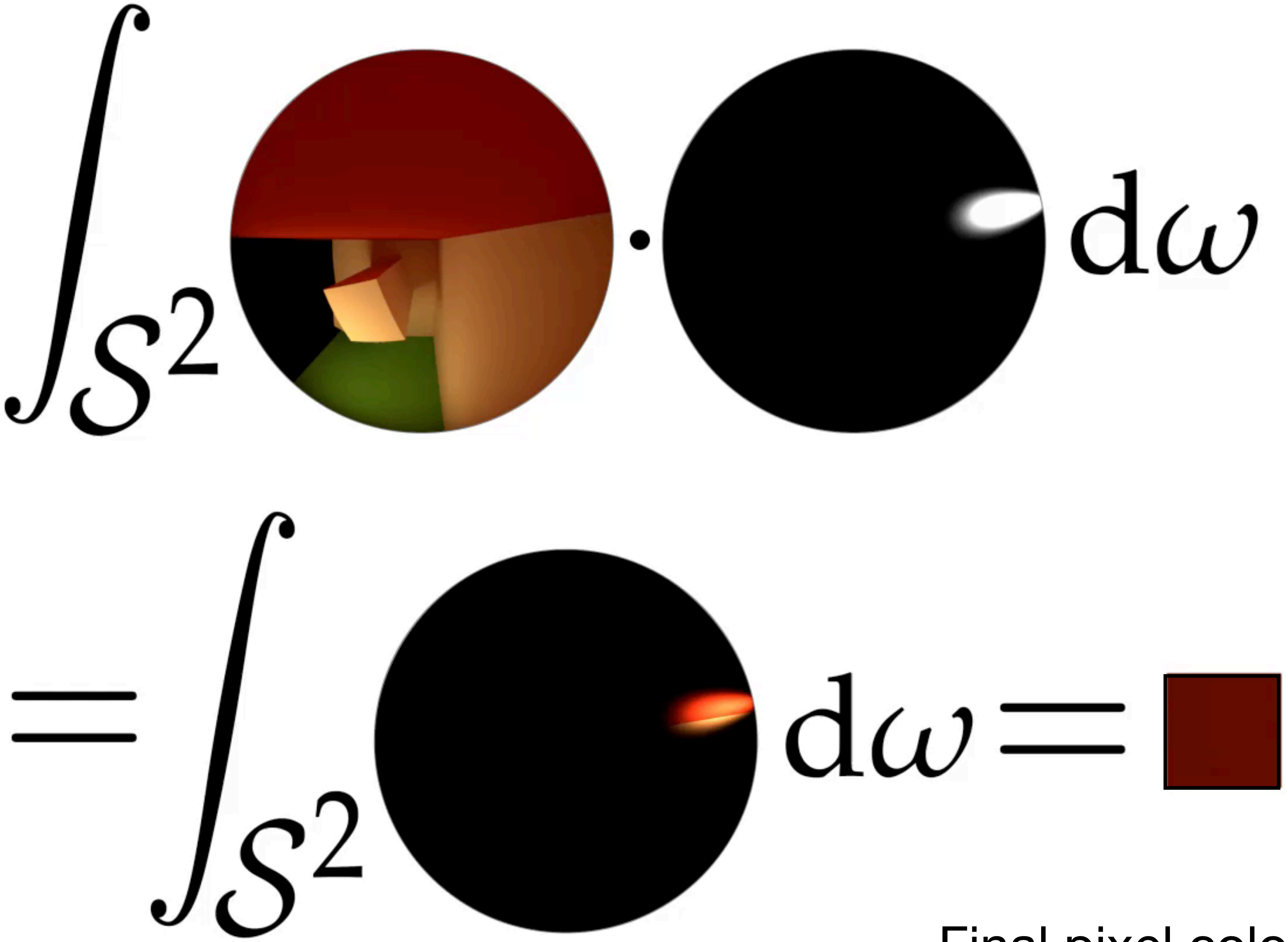
MC Integration / MIS / Limitations

Metropolis light Transport

MC & MCMC Rendering

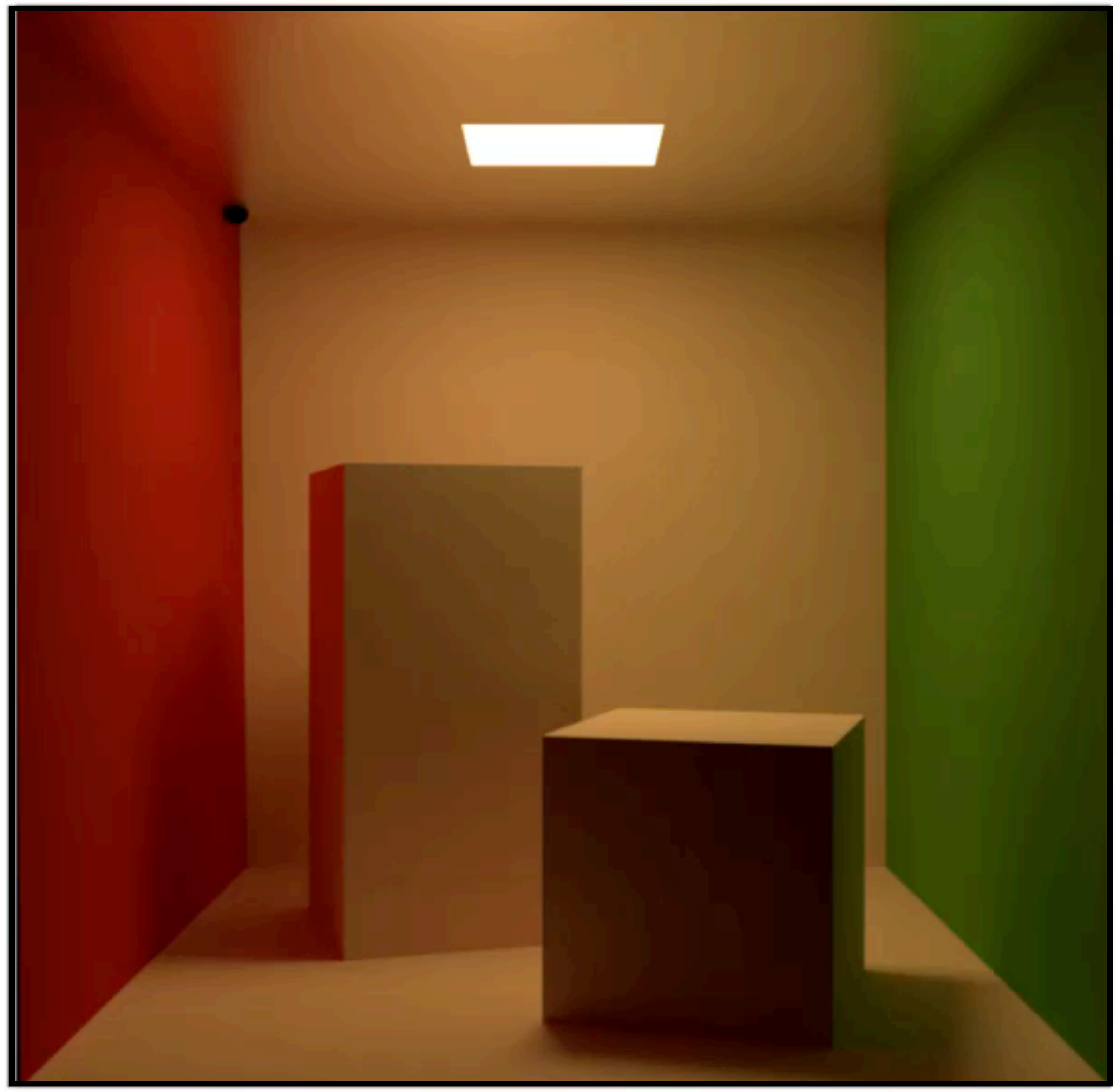
Light Transport 101

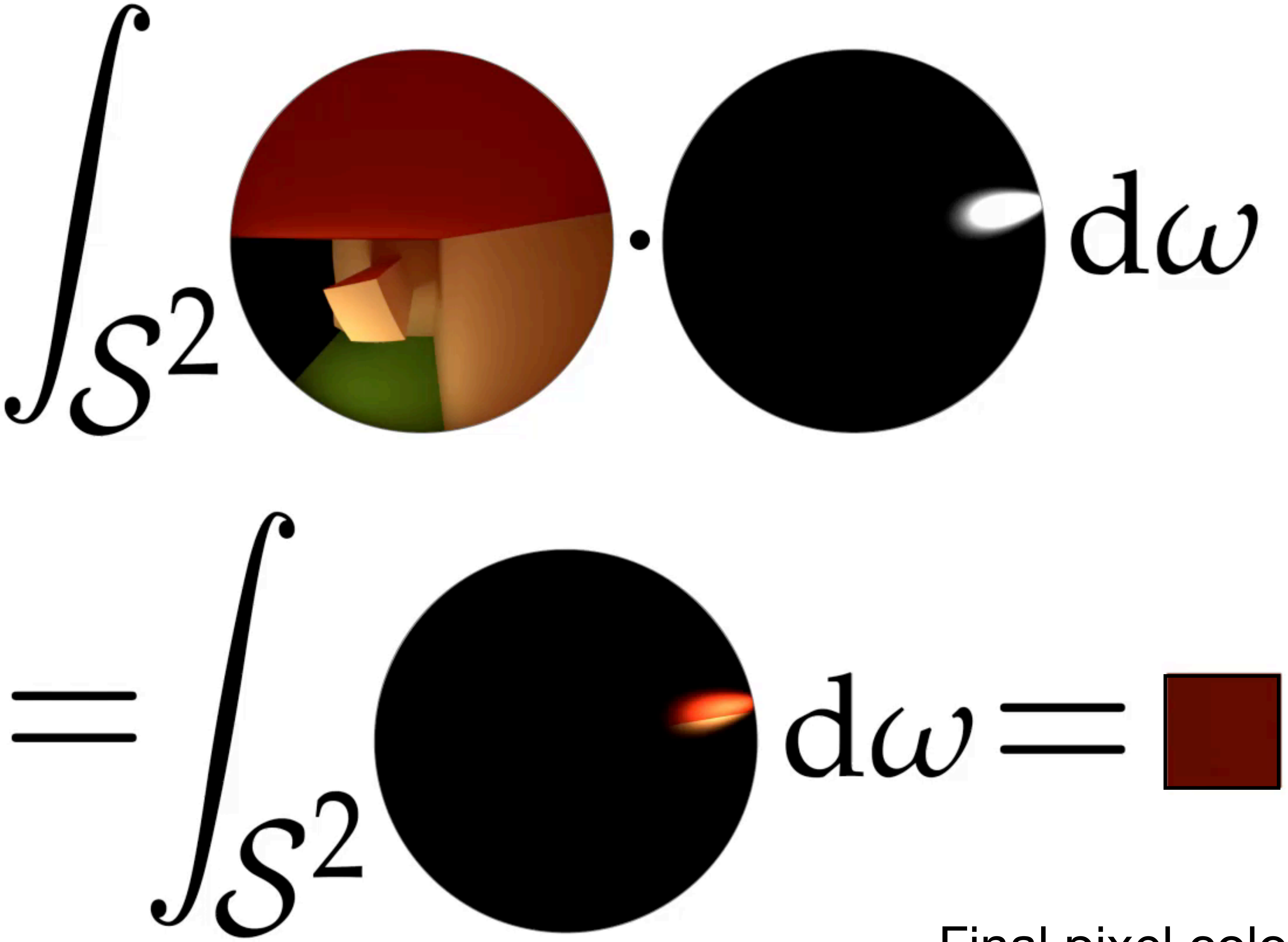




Final pixel color

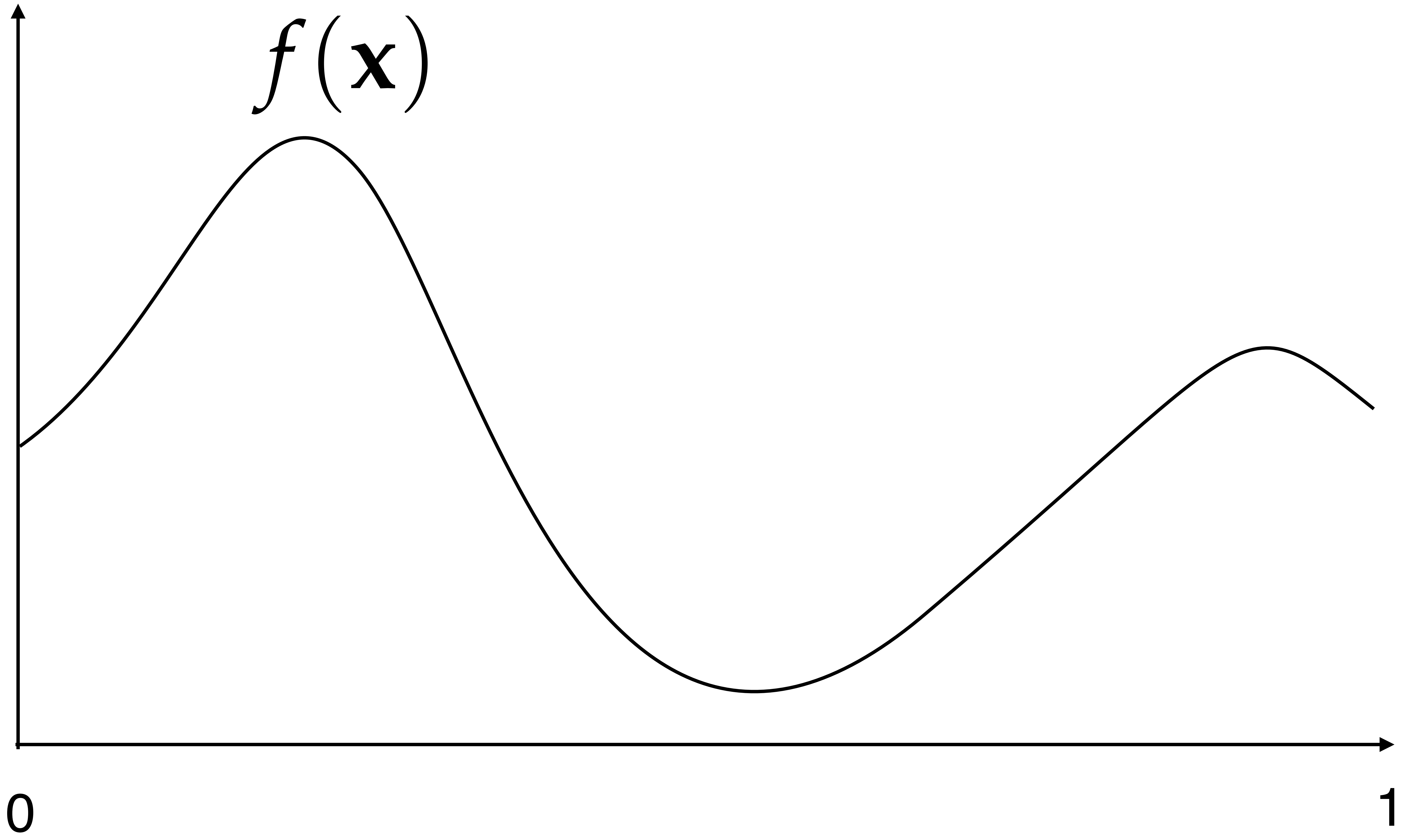
Light Transport 101



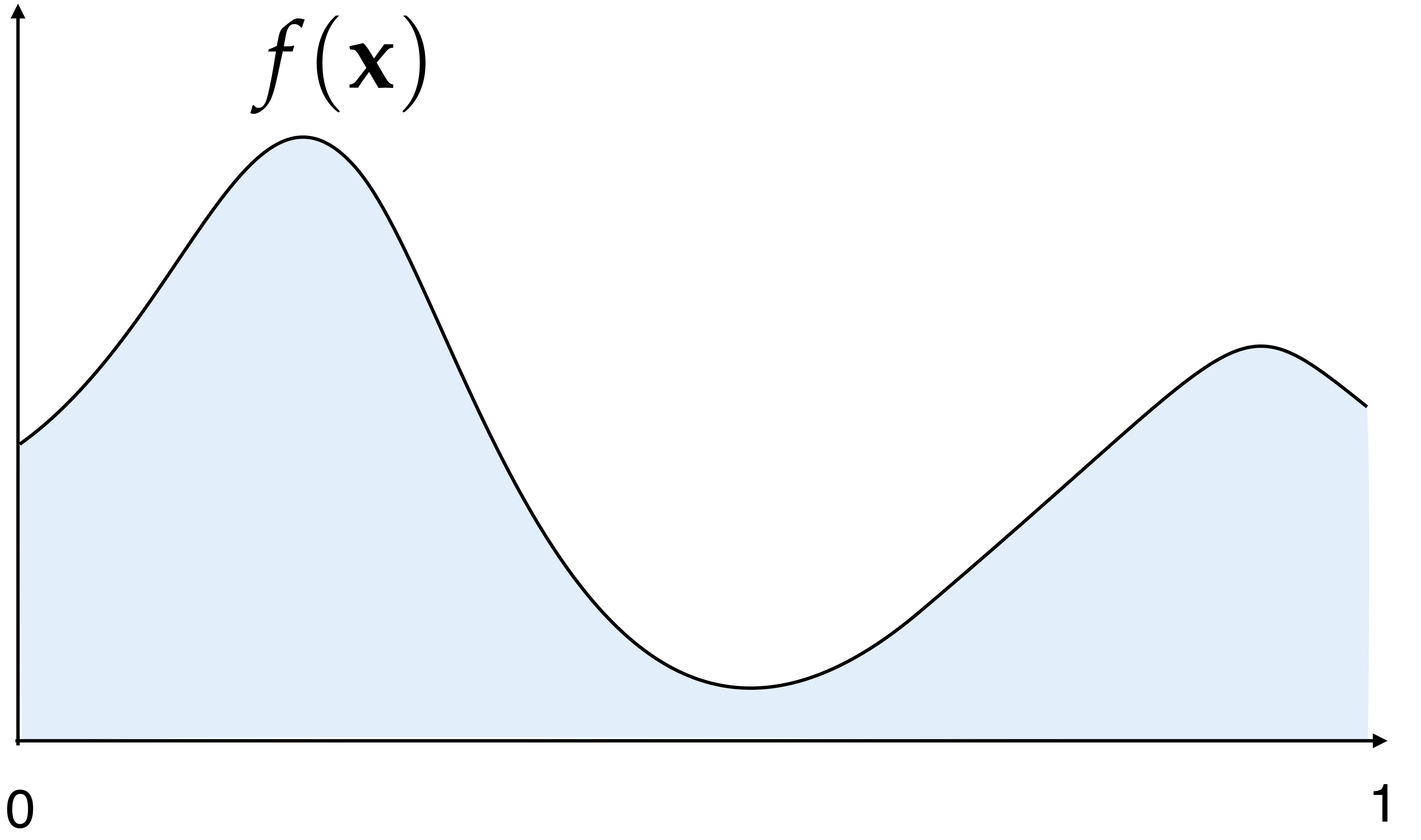


Final pixel color

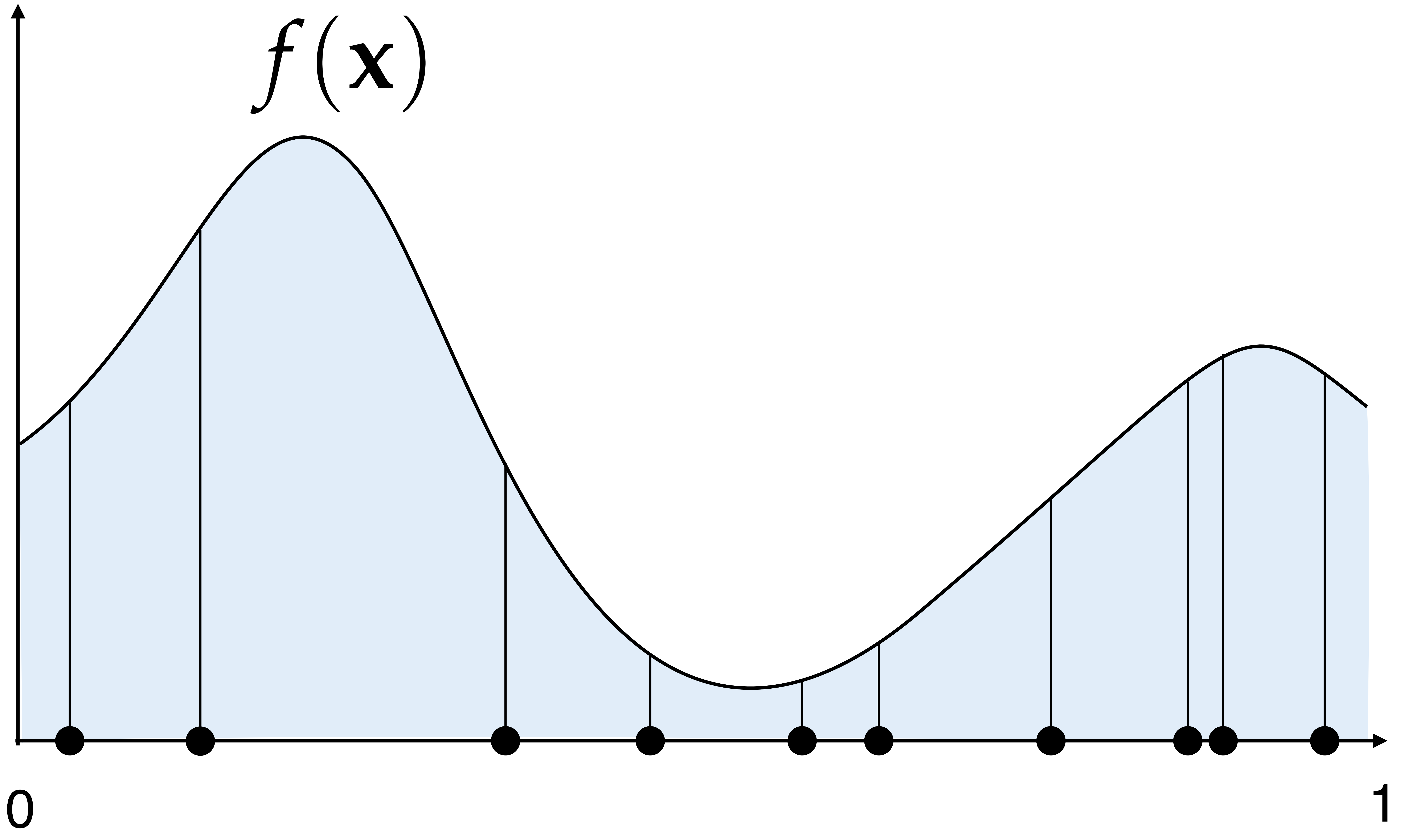
Monte Carlo integration



Monte Carlo integration

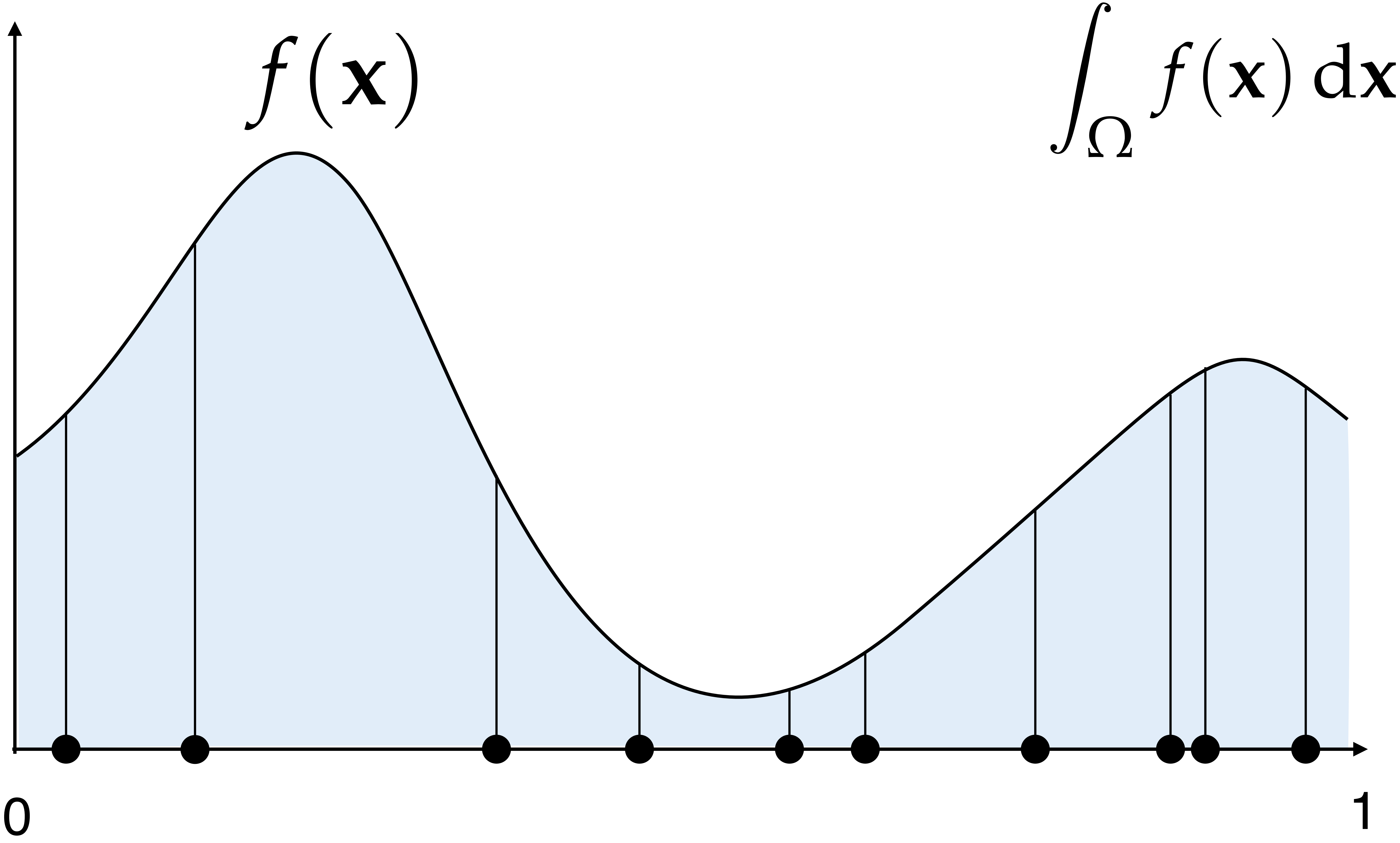


Monte Carlo integration

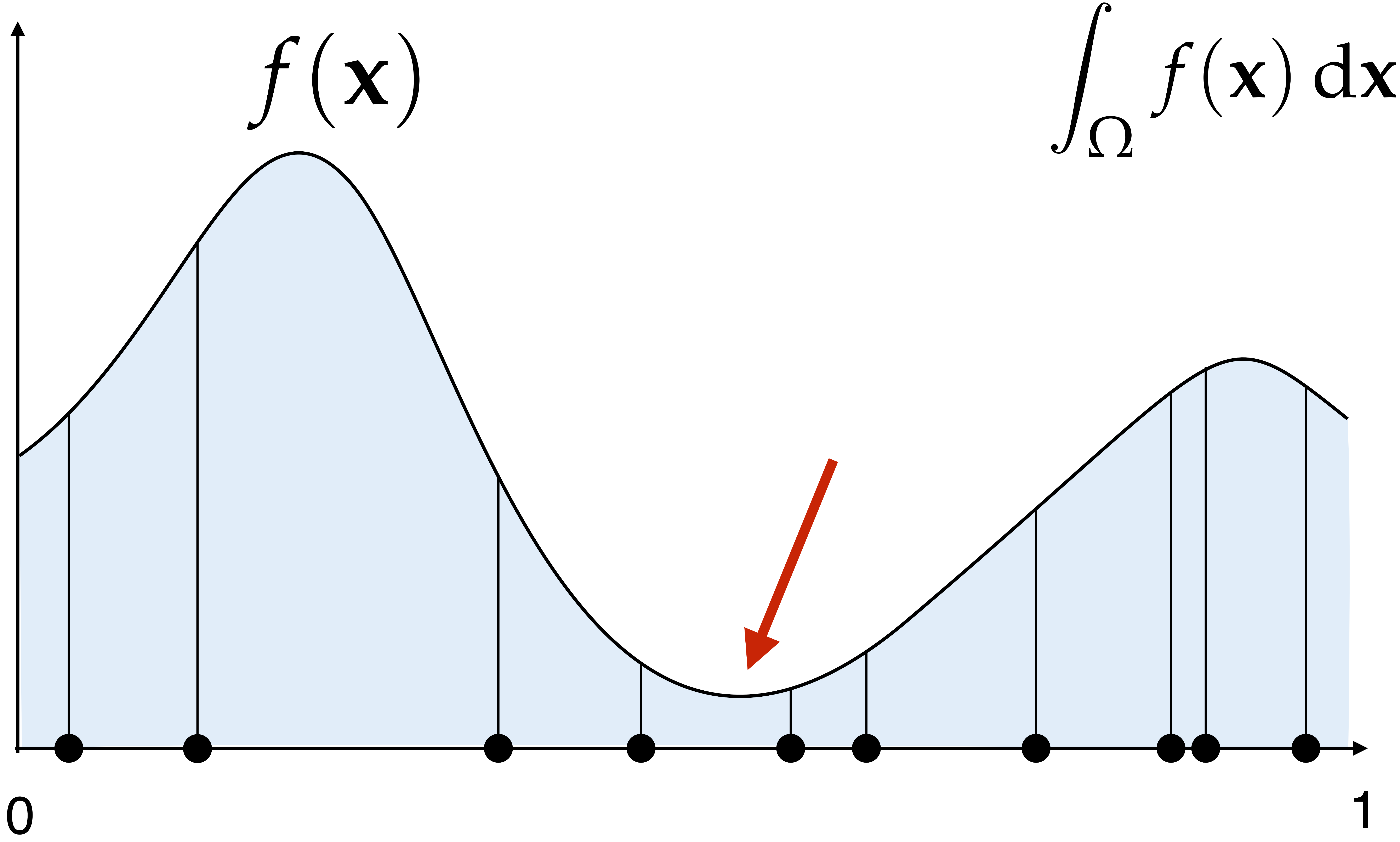


Monte Carlo integration

$$\int_{\Omega} f(\mathbf{x})\,\mathrm{d}\mathbf{x} \approx \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x}_i)$$

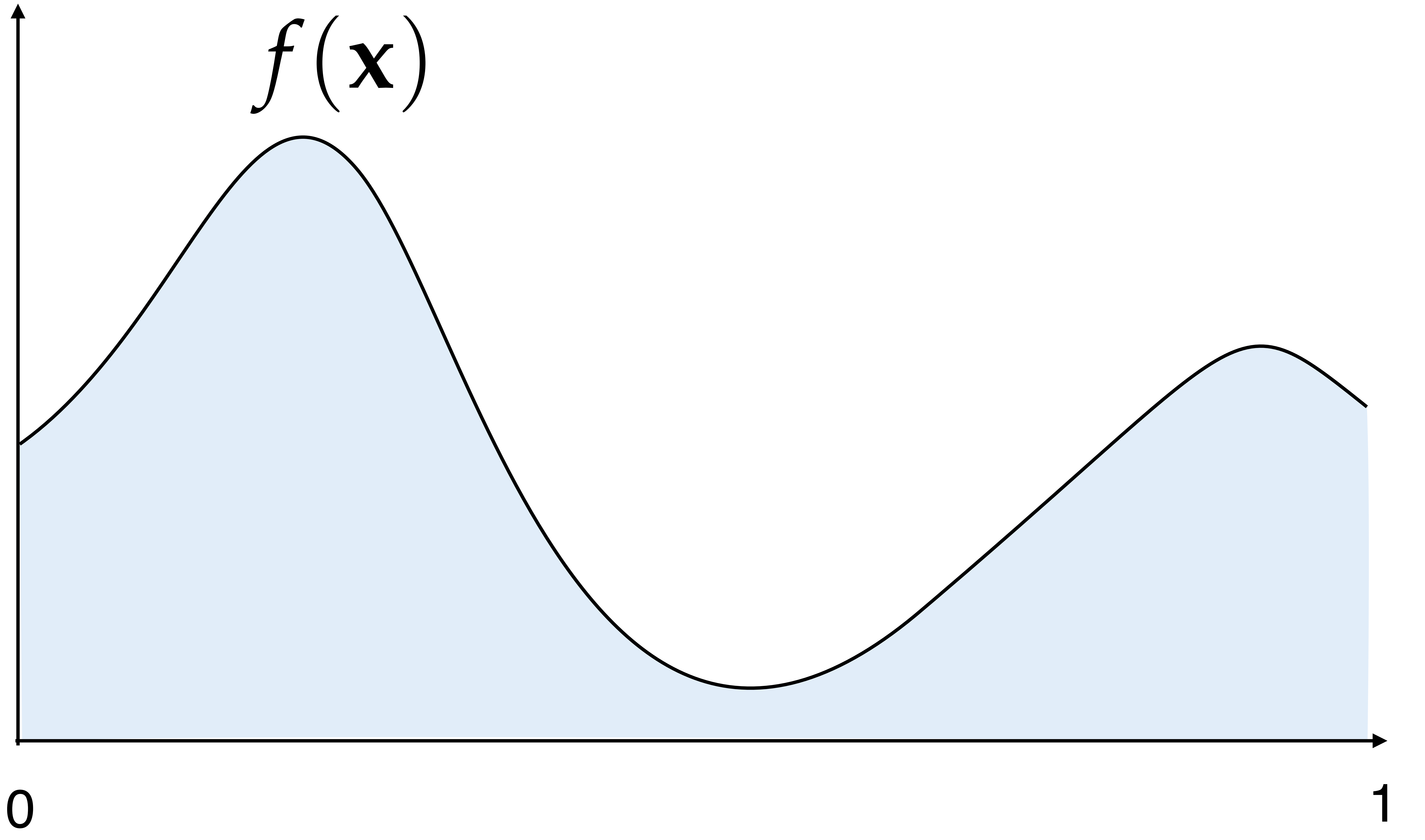


Monte Carlo integration

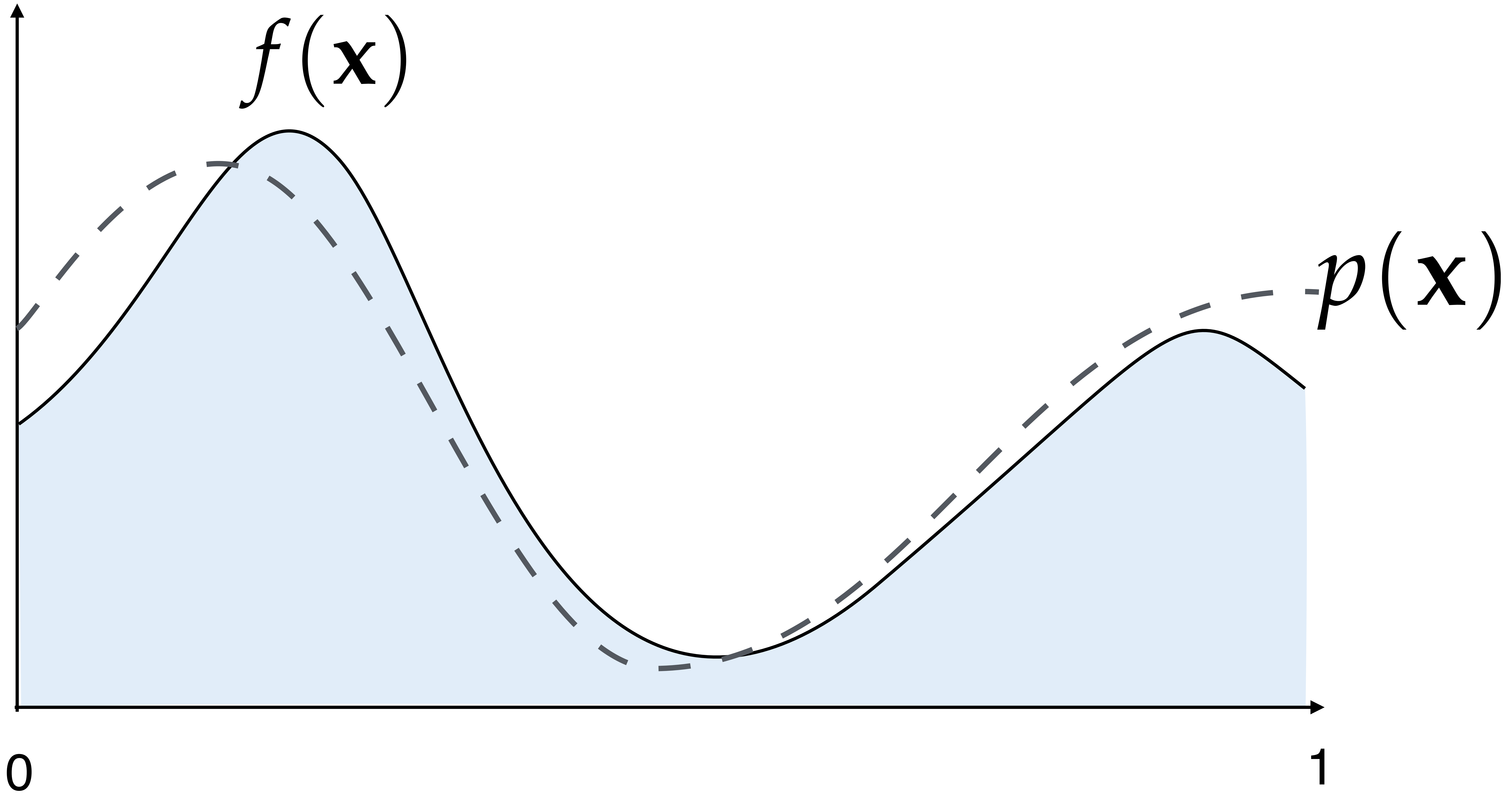


$$\int_\Omega f(\mathbf{x})\,\mathrm{d}\mathbf{x} \approx \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x}_i)$$

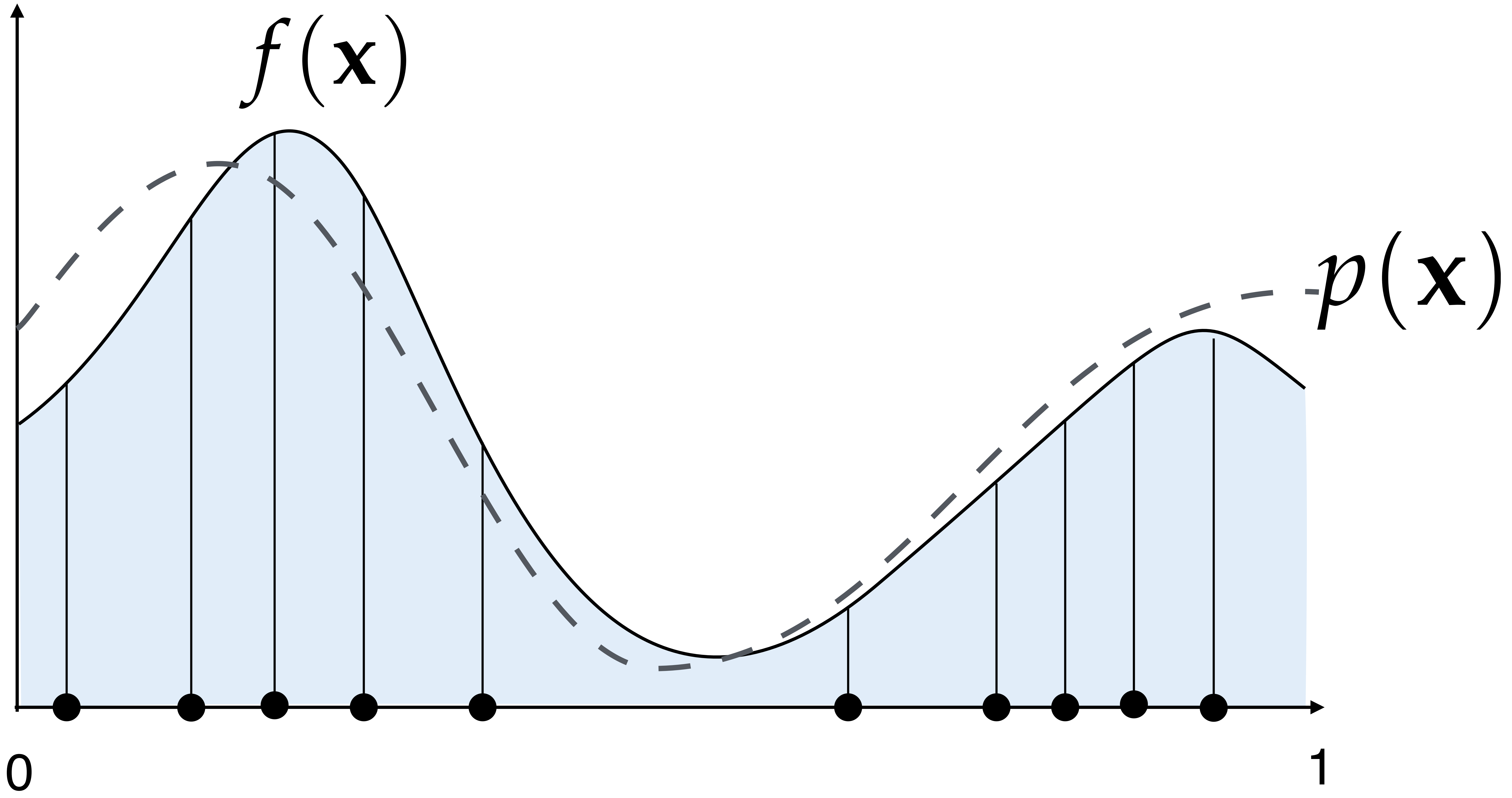
Importance sampling



Importance sampling

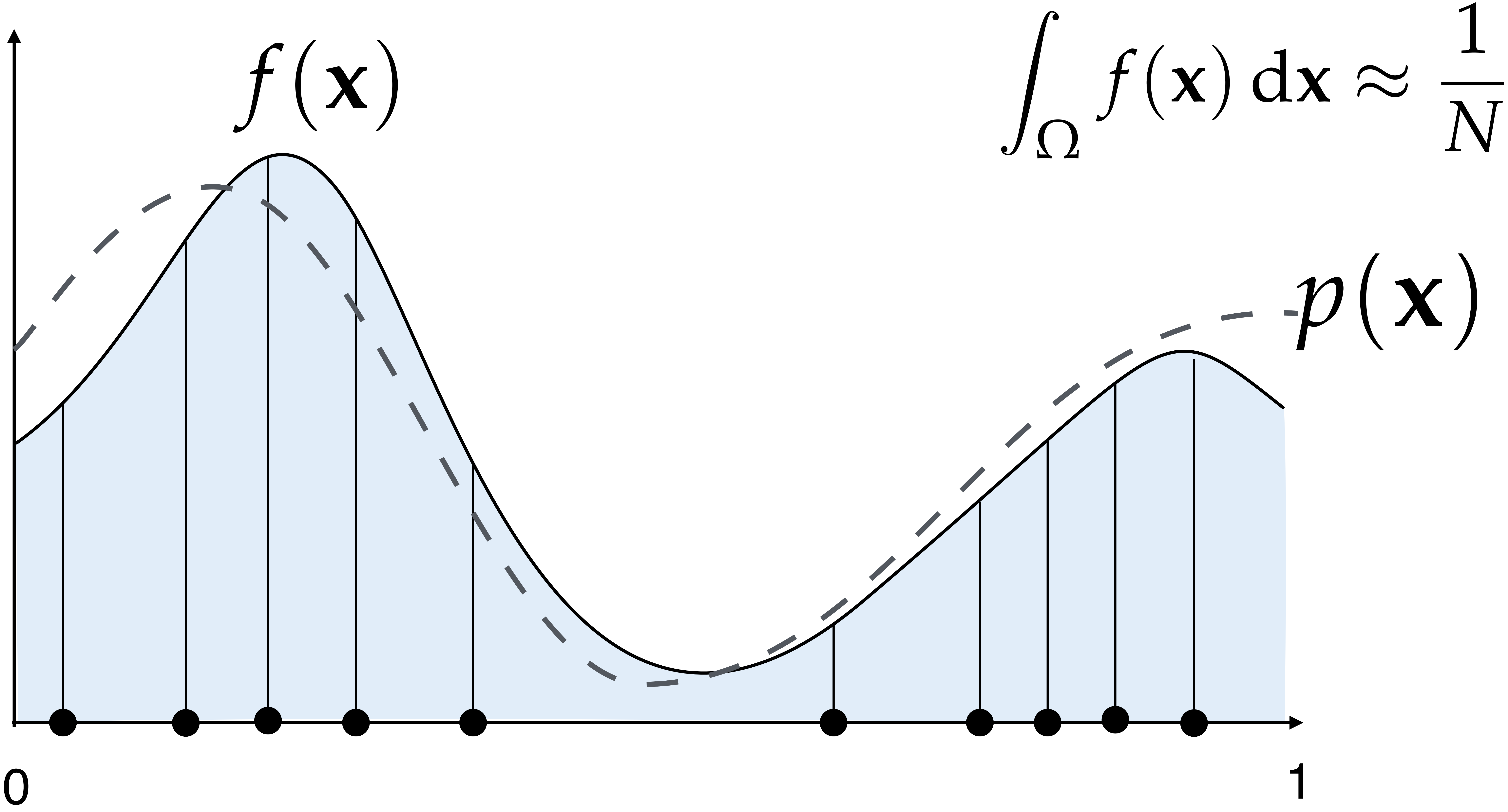


Importance sampling

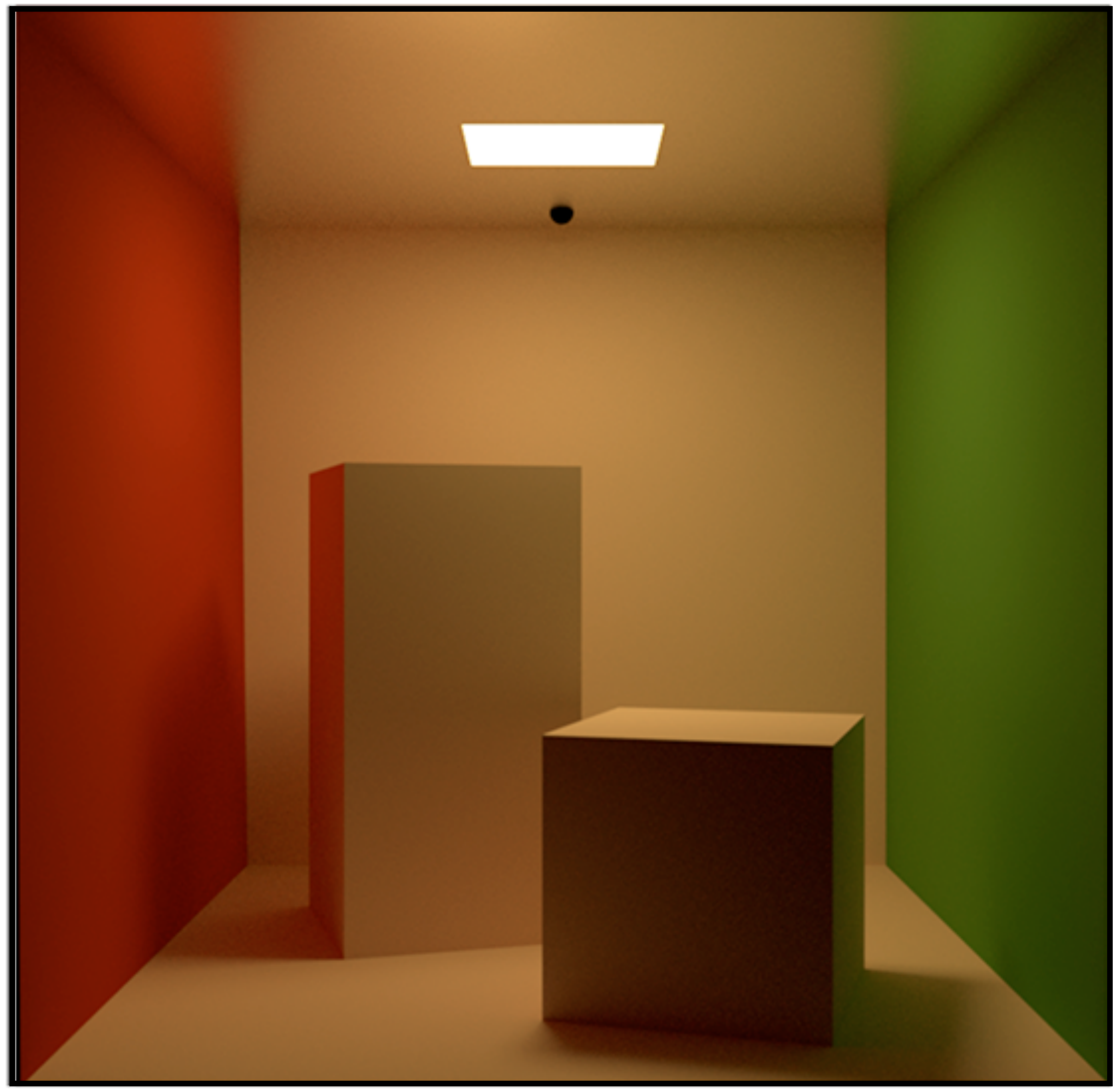


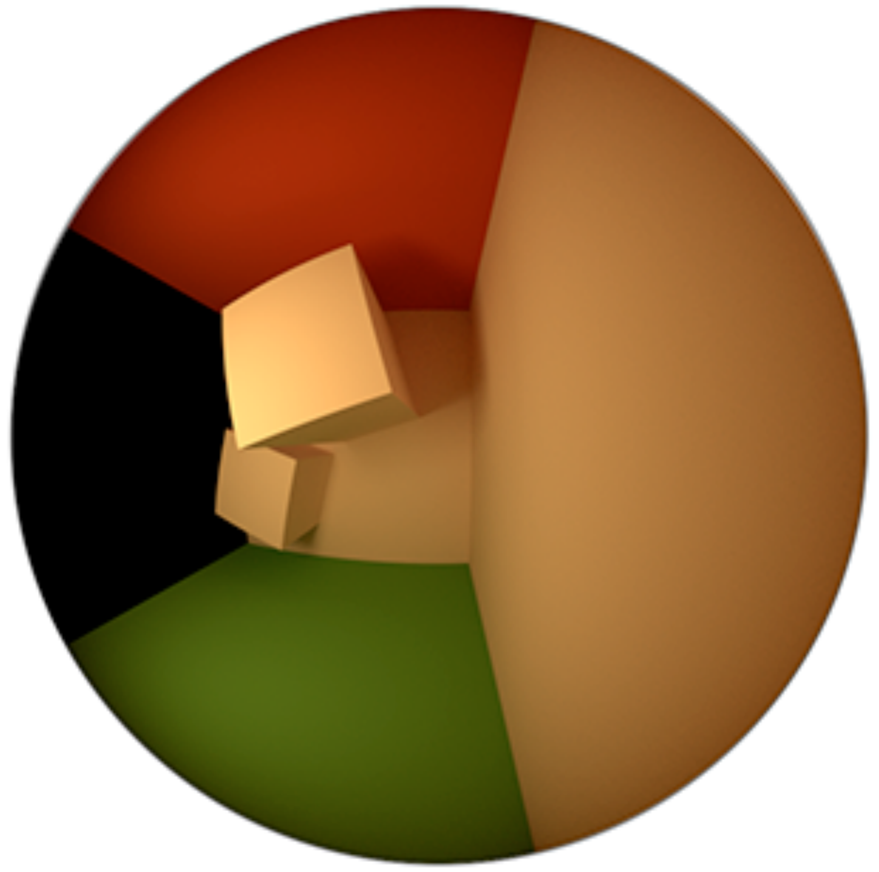
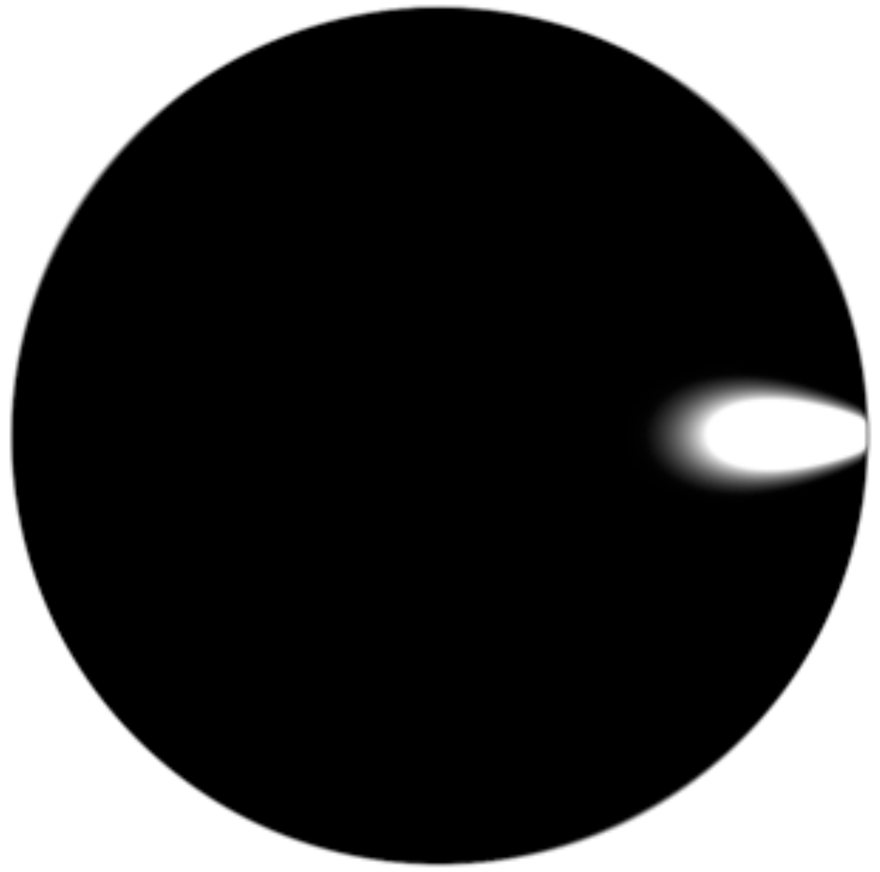
Importance sampling

$$\int_{\Omega} f(\mathbf{x})\,\mathrm{d}\mathbf{x} \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(\mathbf{x}_i)}{p(\mathbf{x}_i)}$$

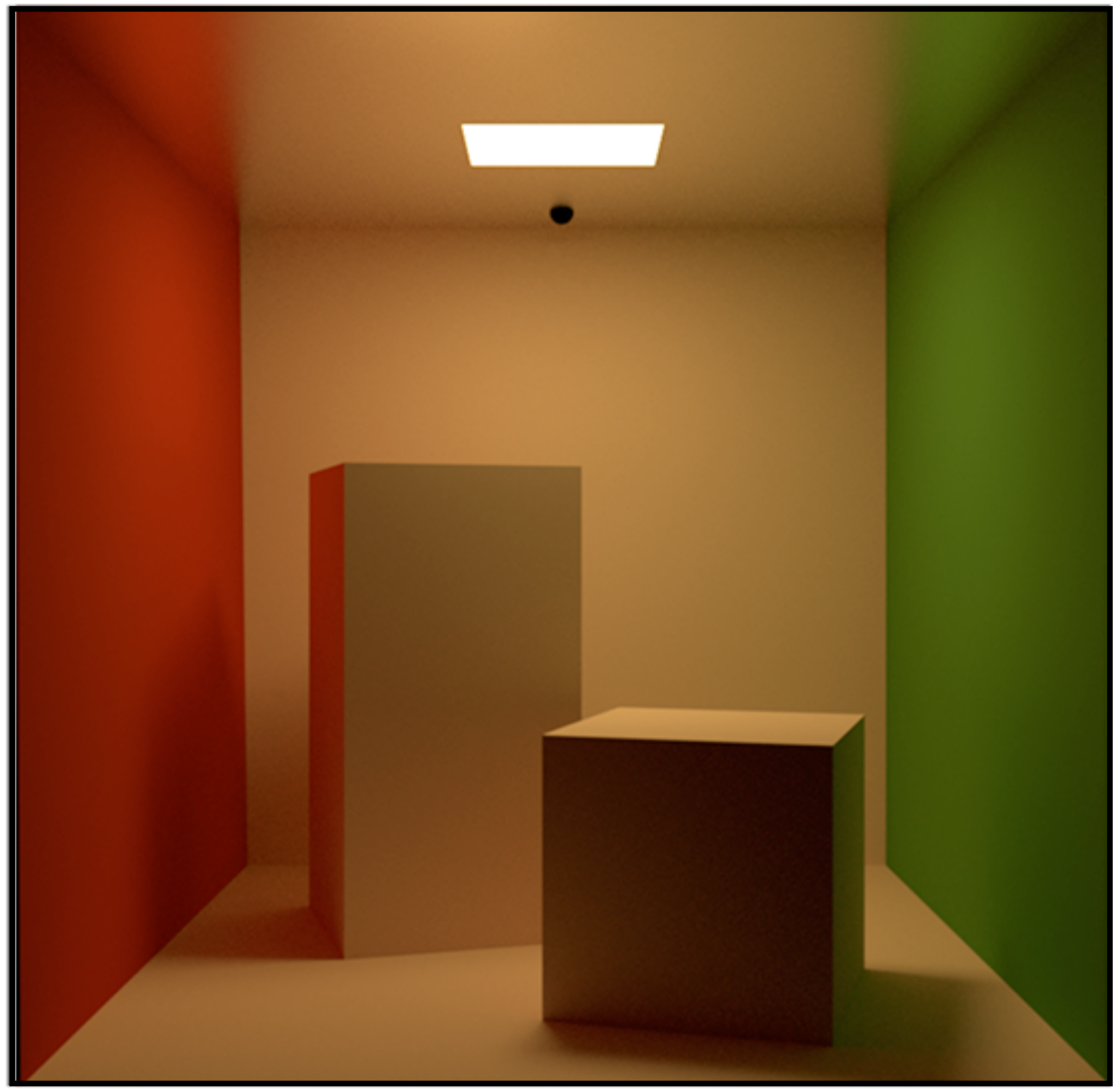


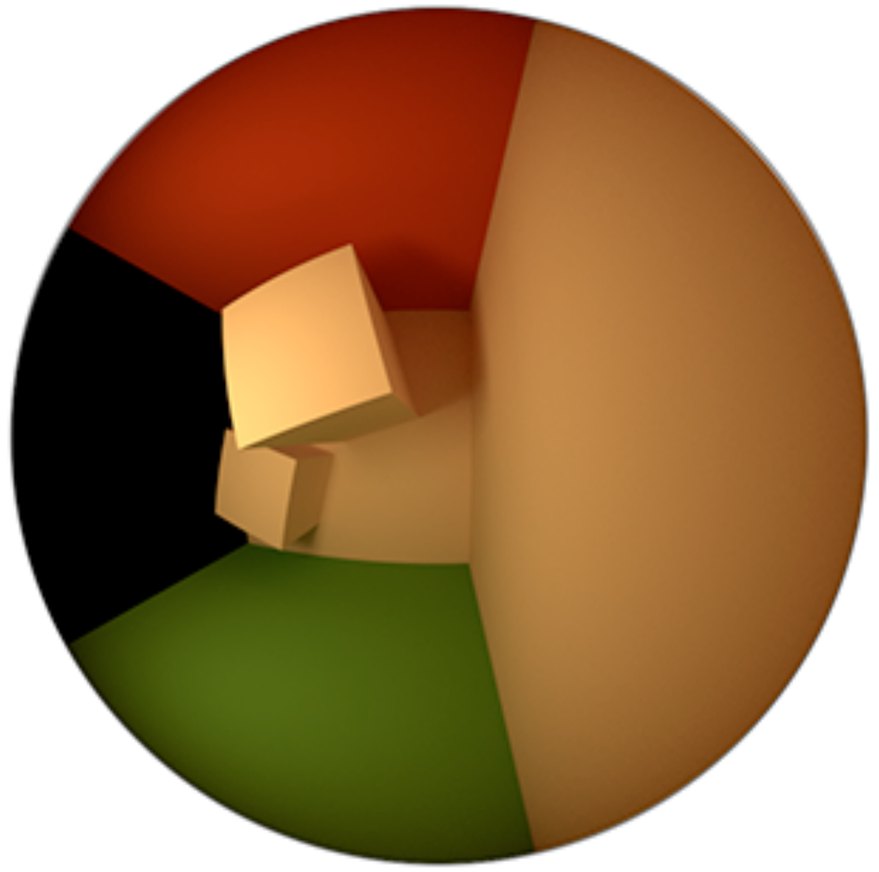
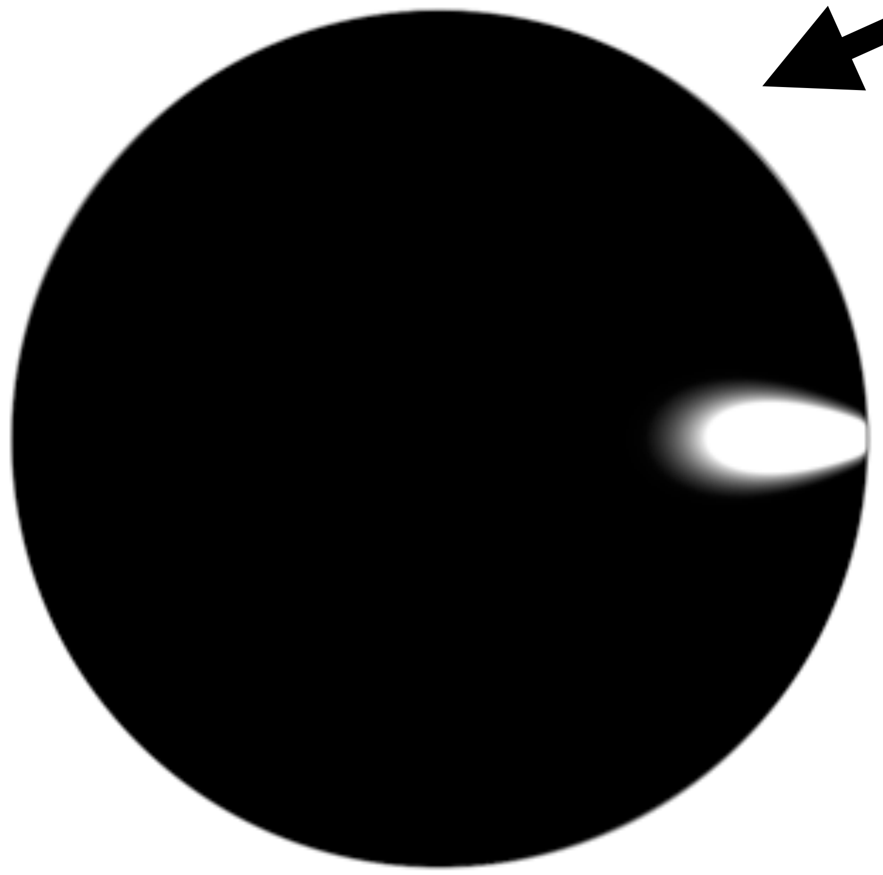
Sampling the integral



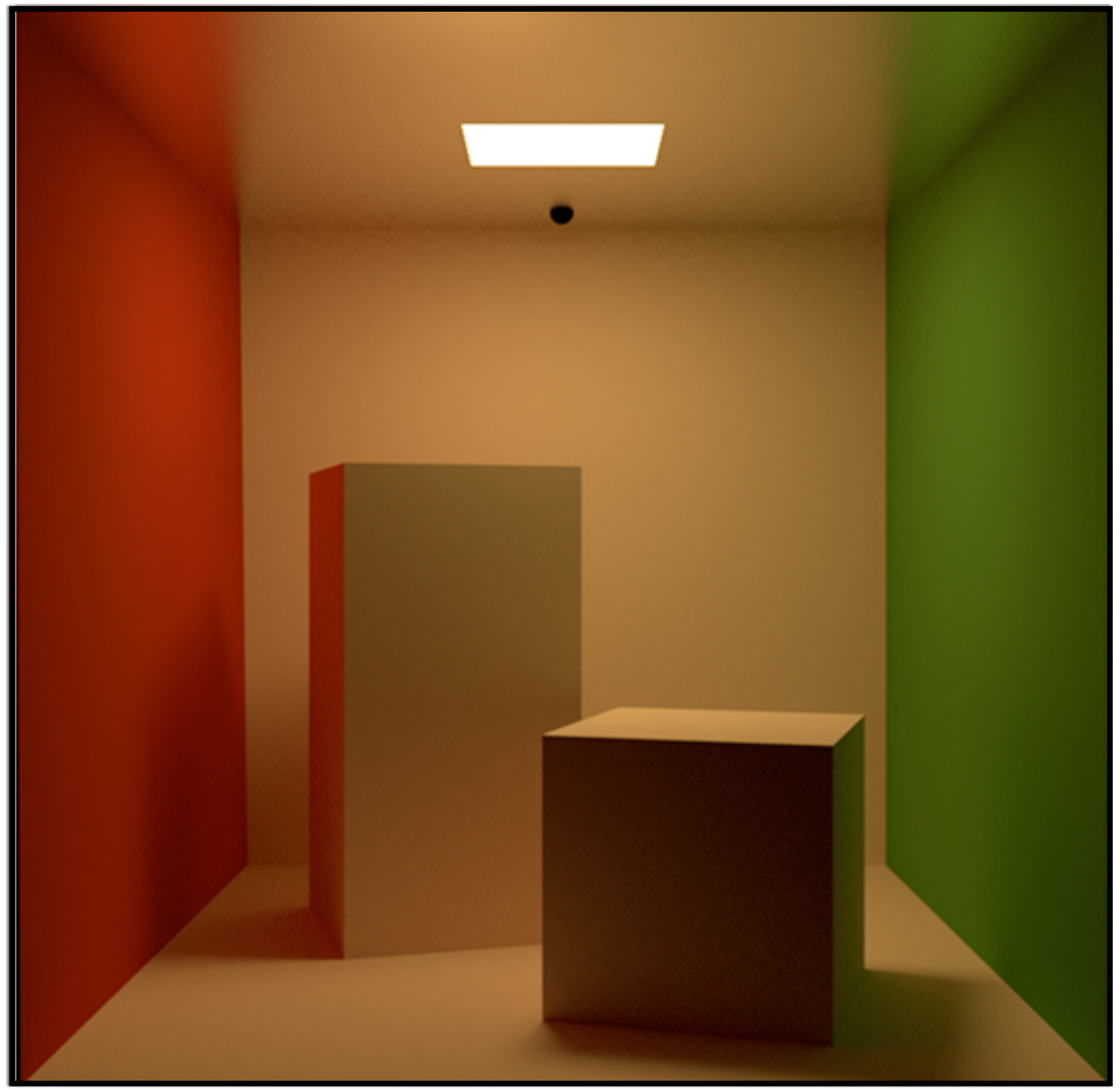
$\int_{\mathcal{S}^2}$  $\cdot$  $\mathrm{d}\omega$

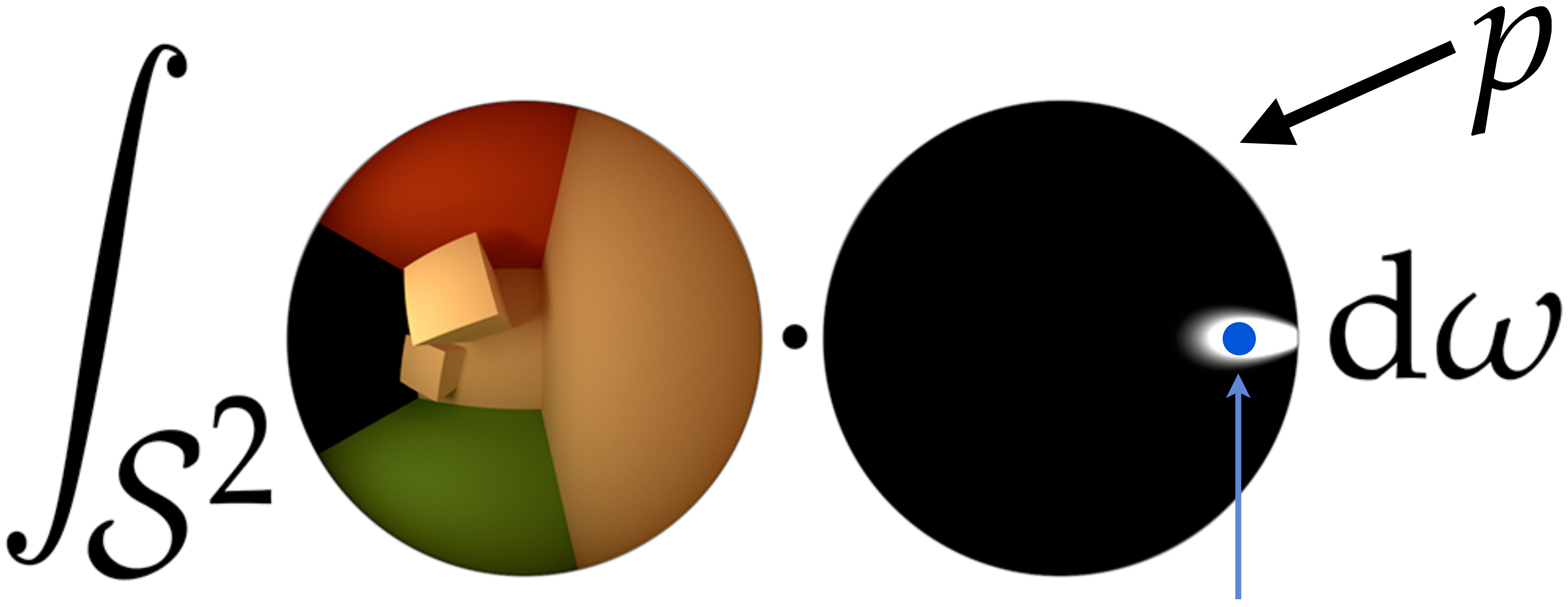
Sampling the integral



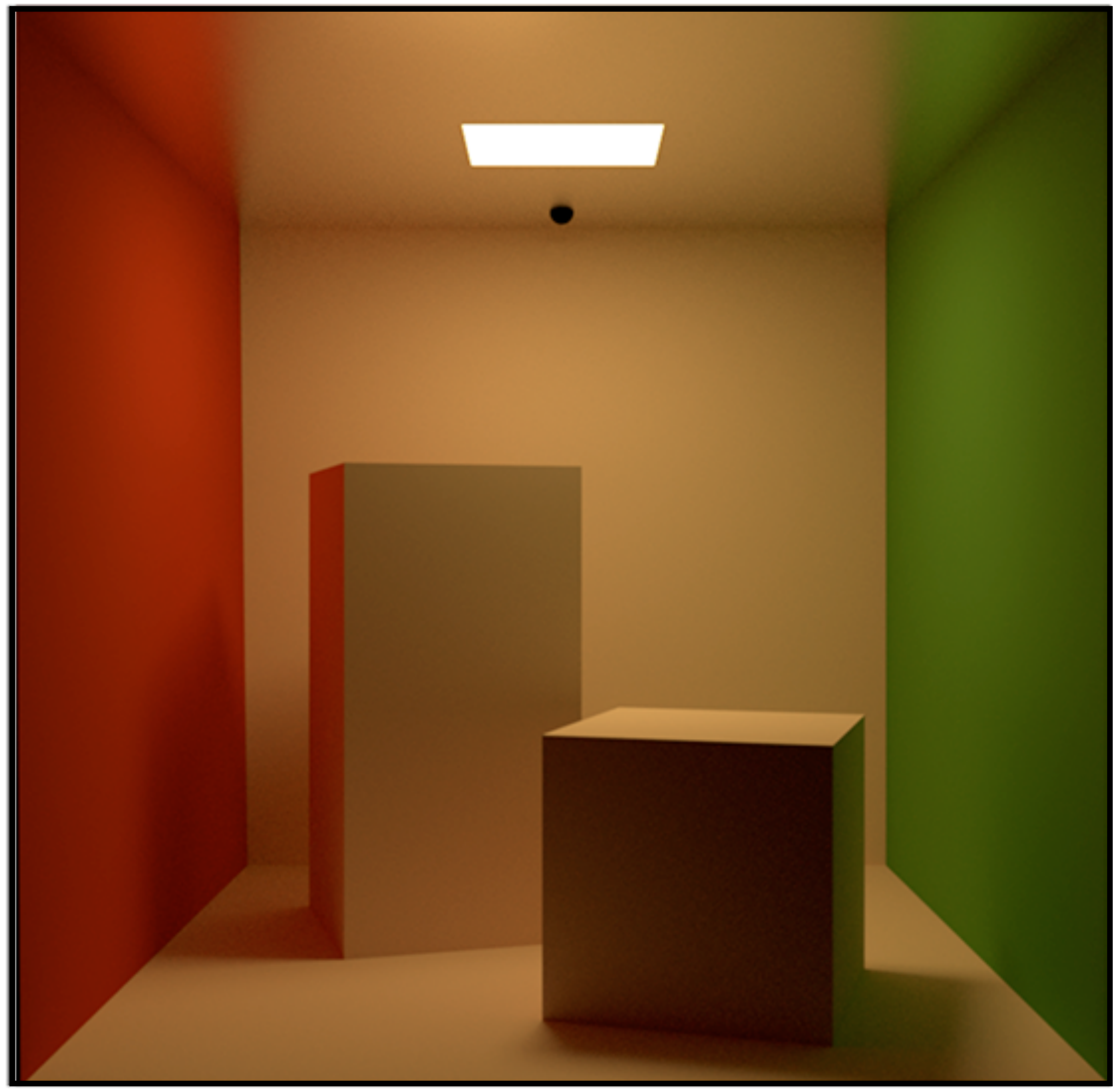
$\int_{S^2}$  $\cdot$  $\mathrm{d}\omega$

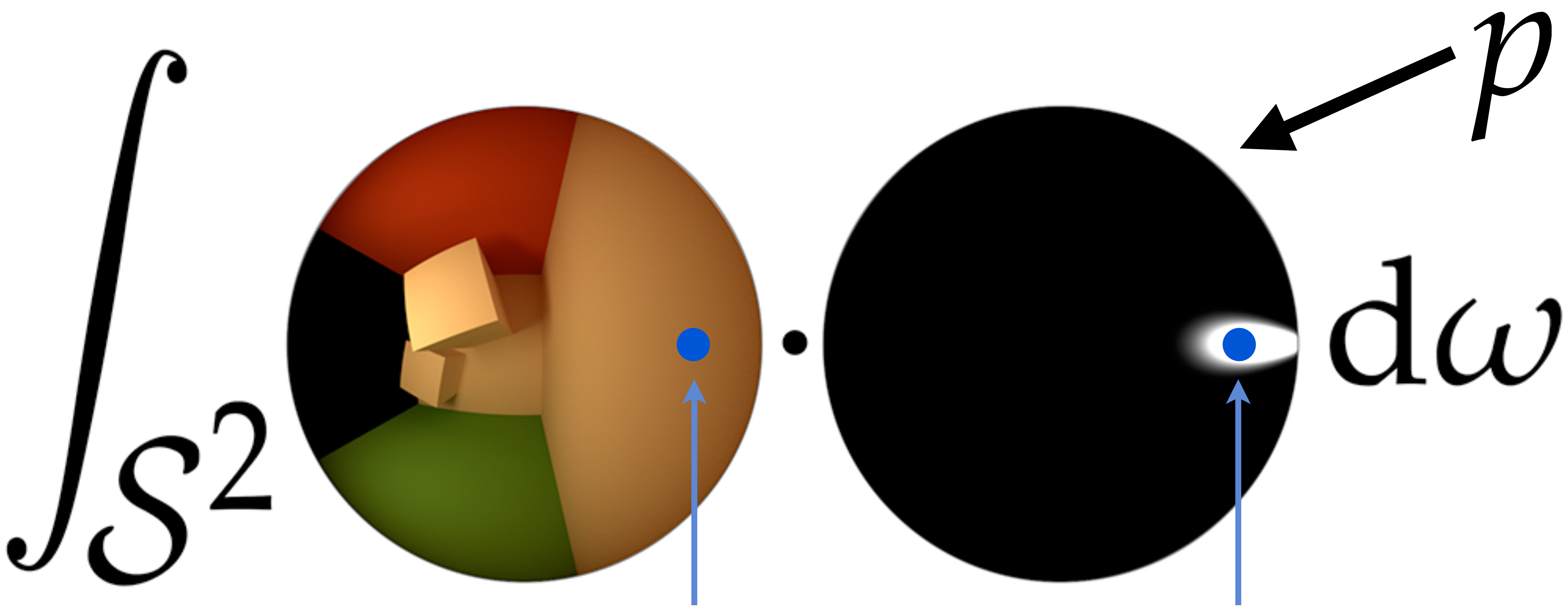
Sampling the integral



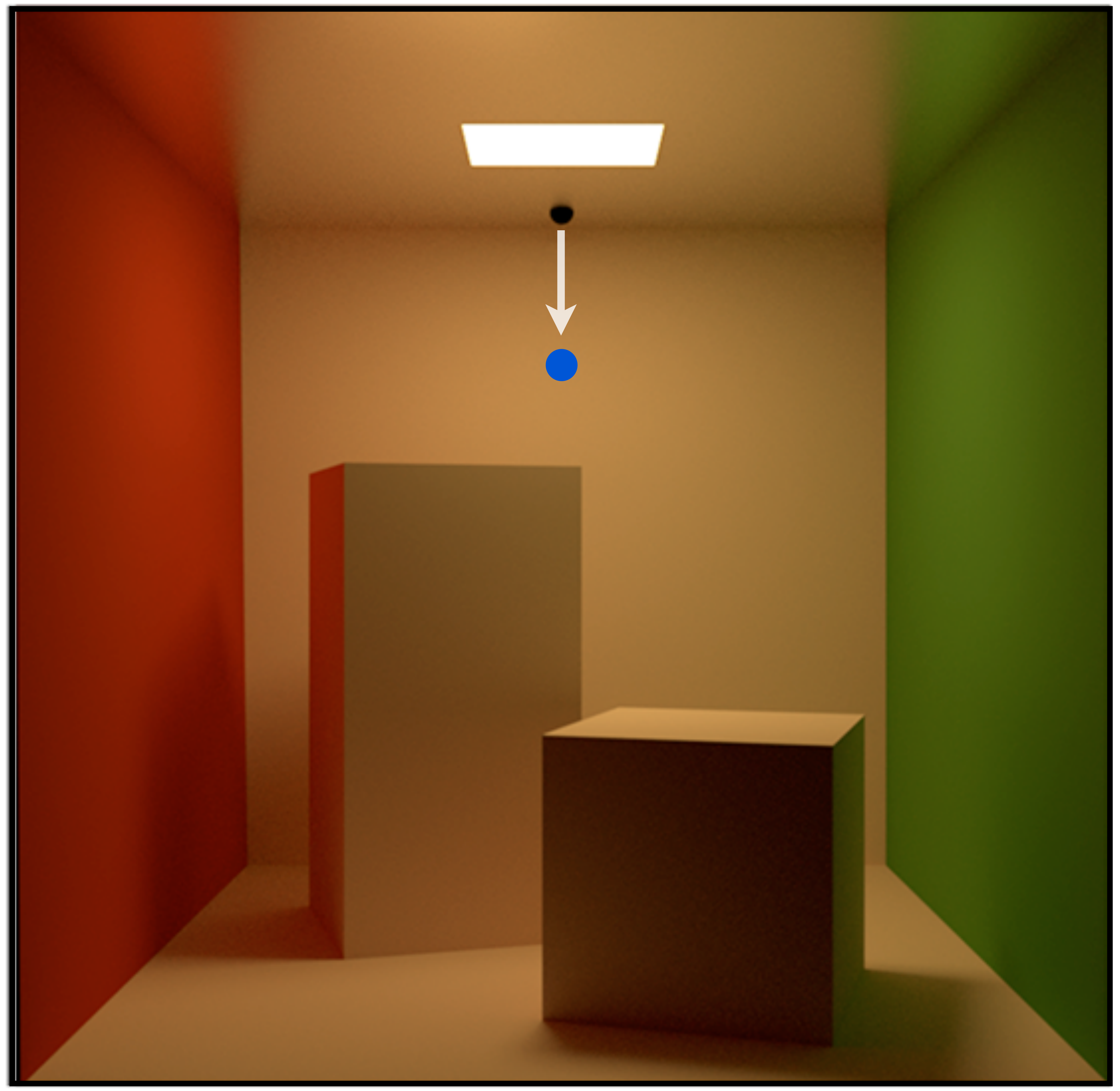


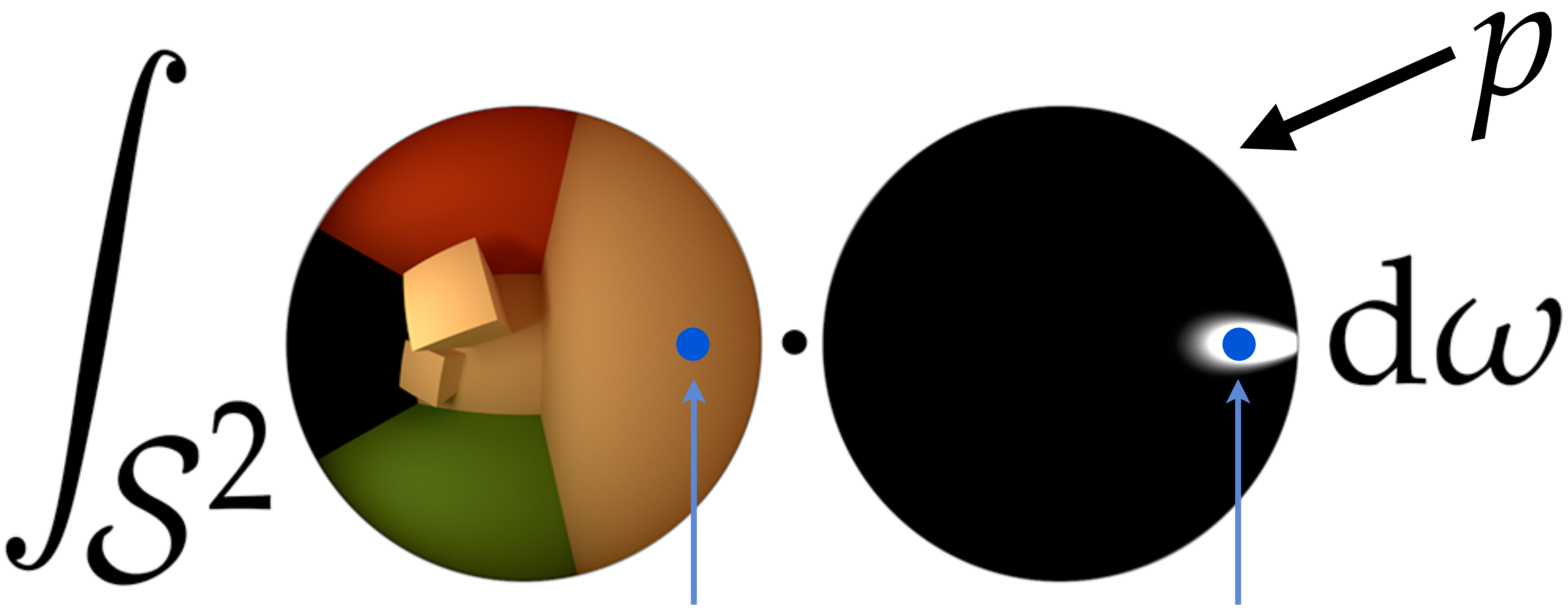
Sampling the integral



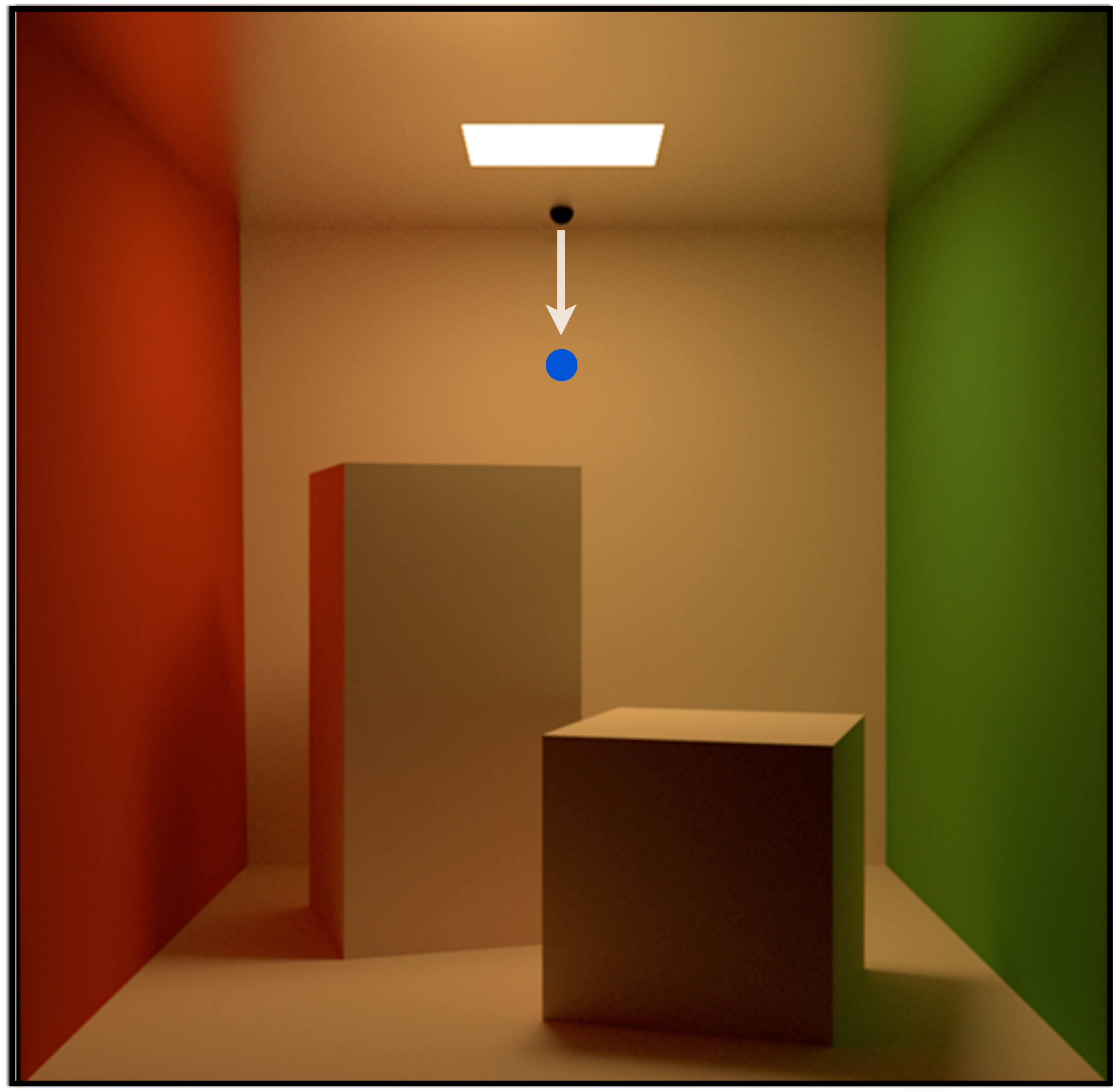


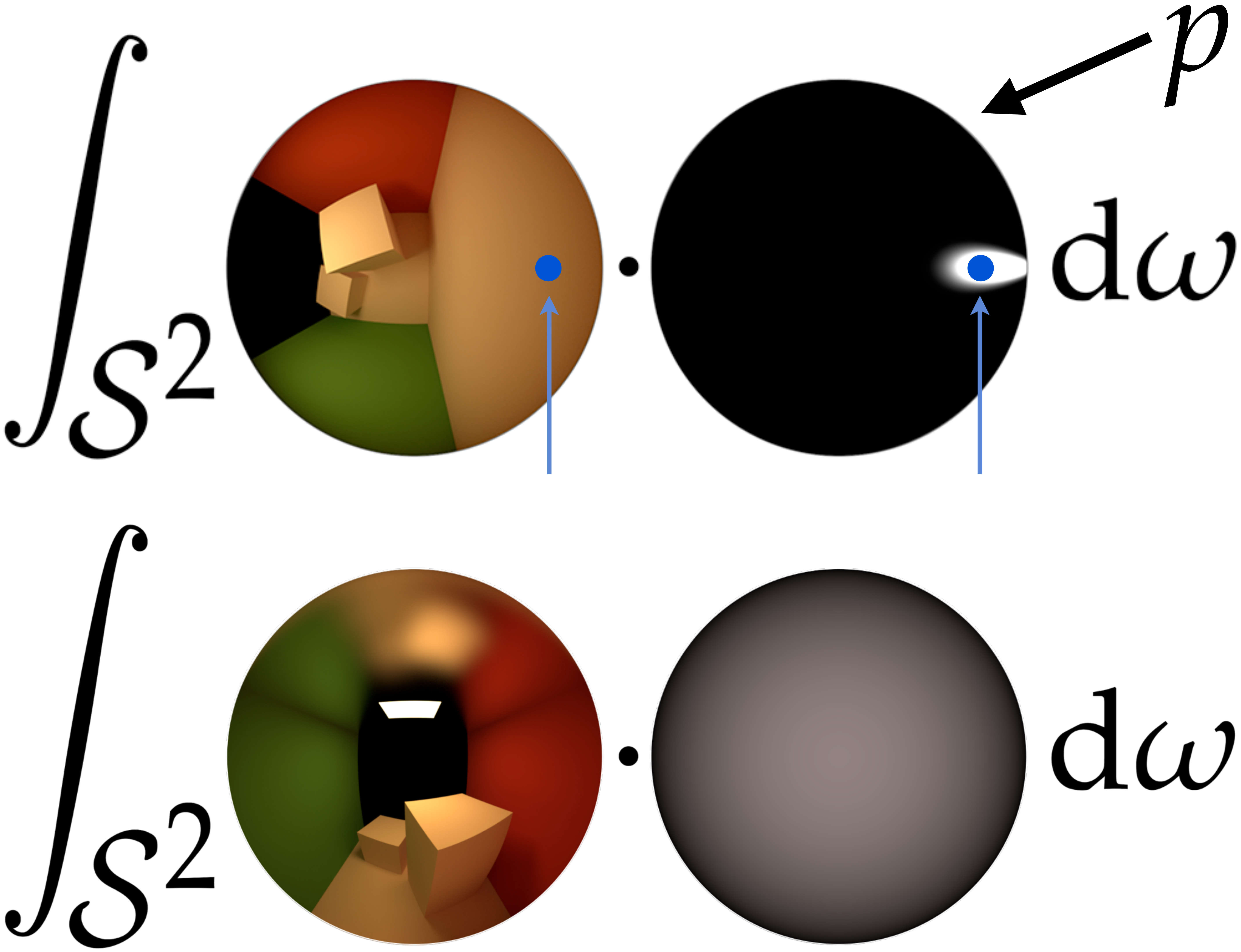
Sampling the integral



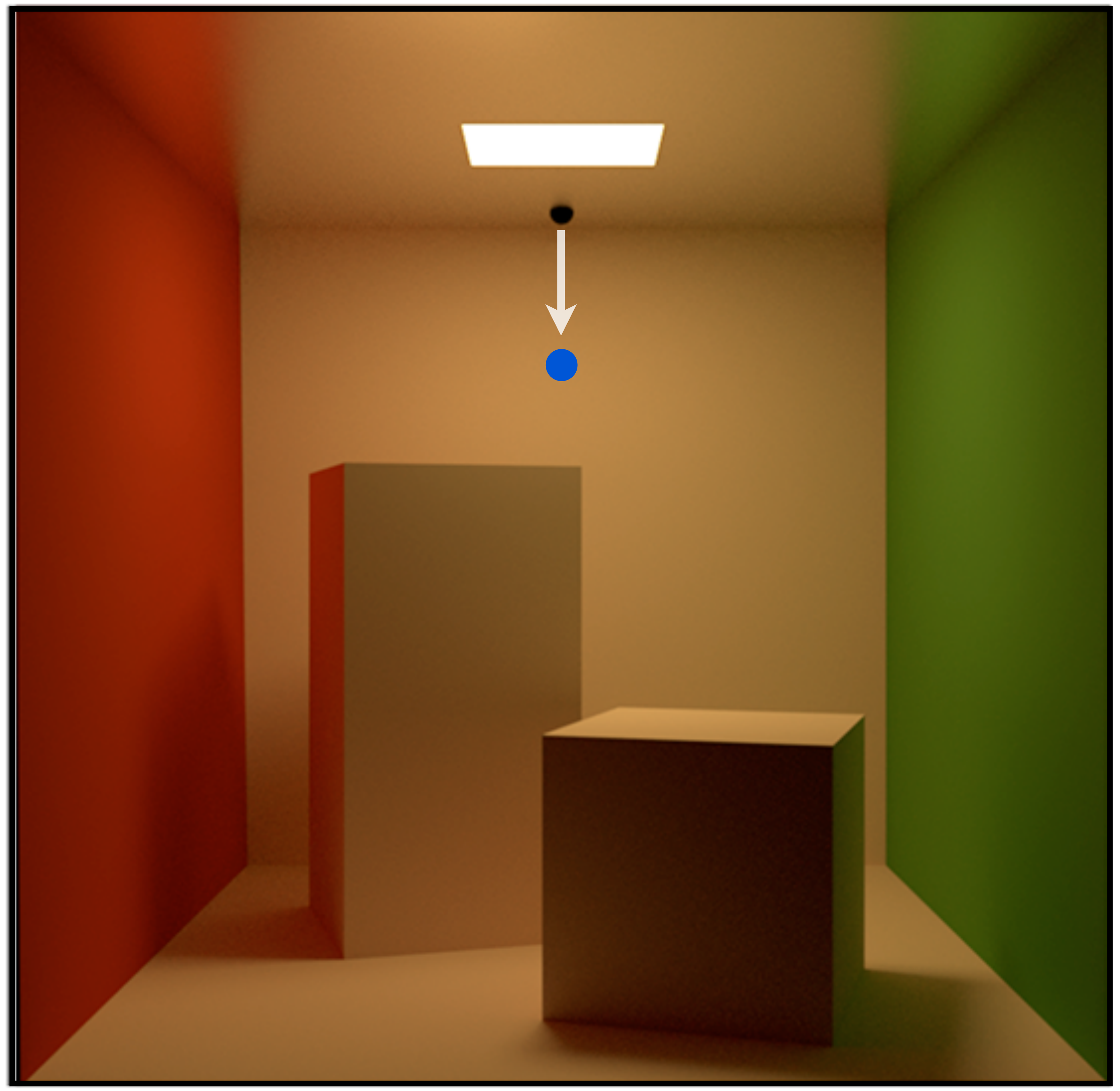


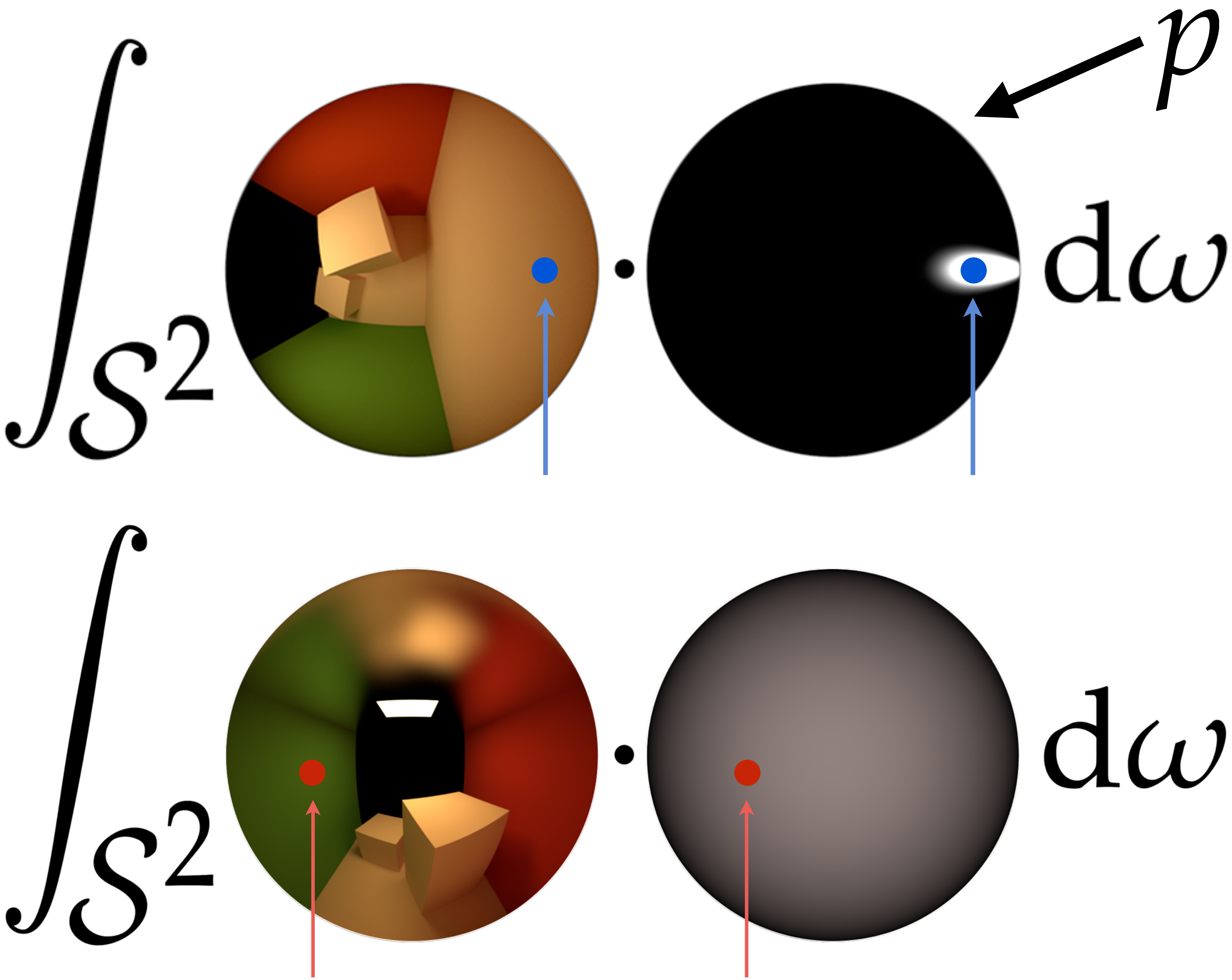
Sampling the integral



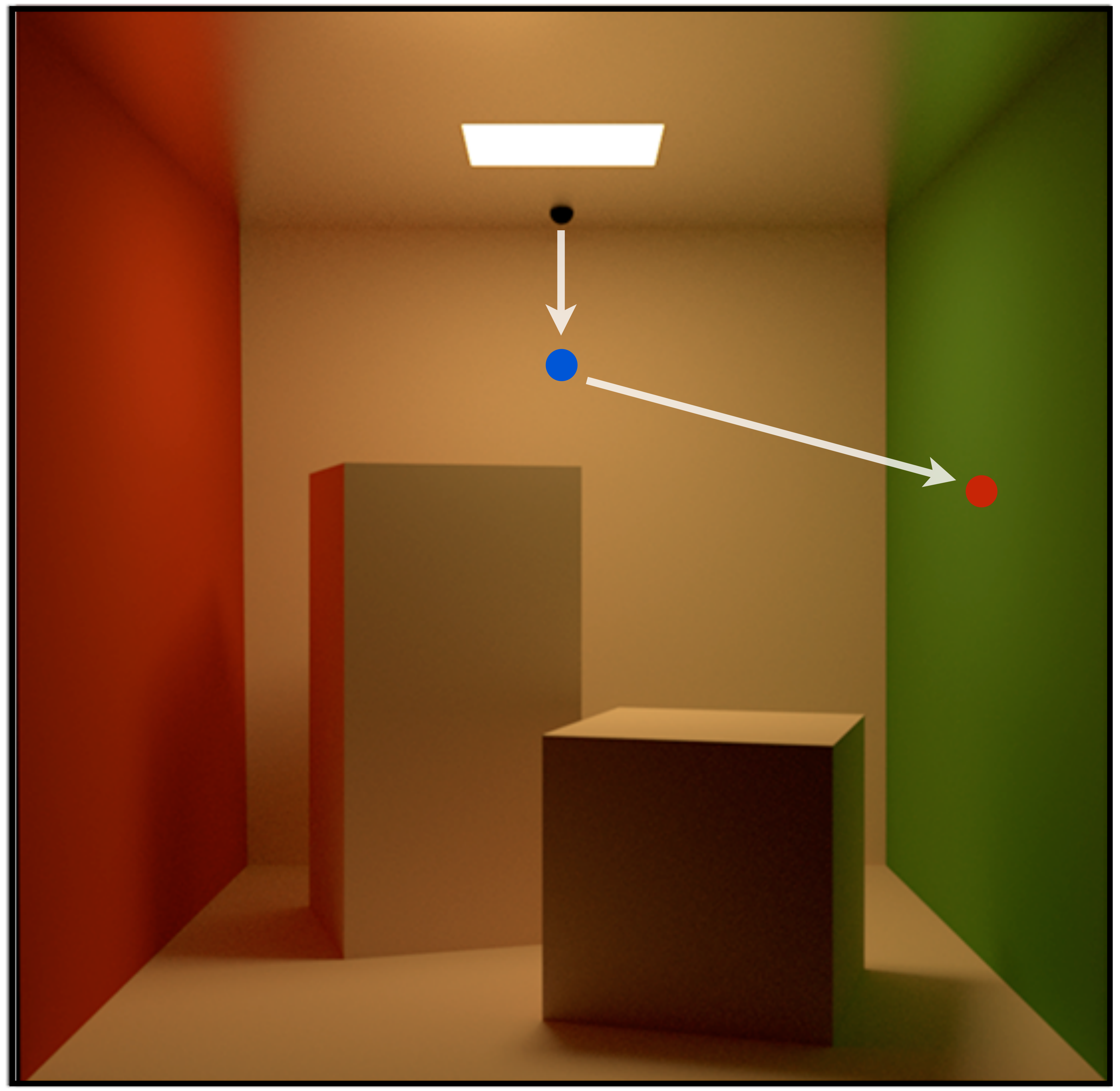


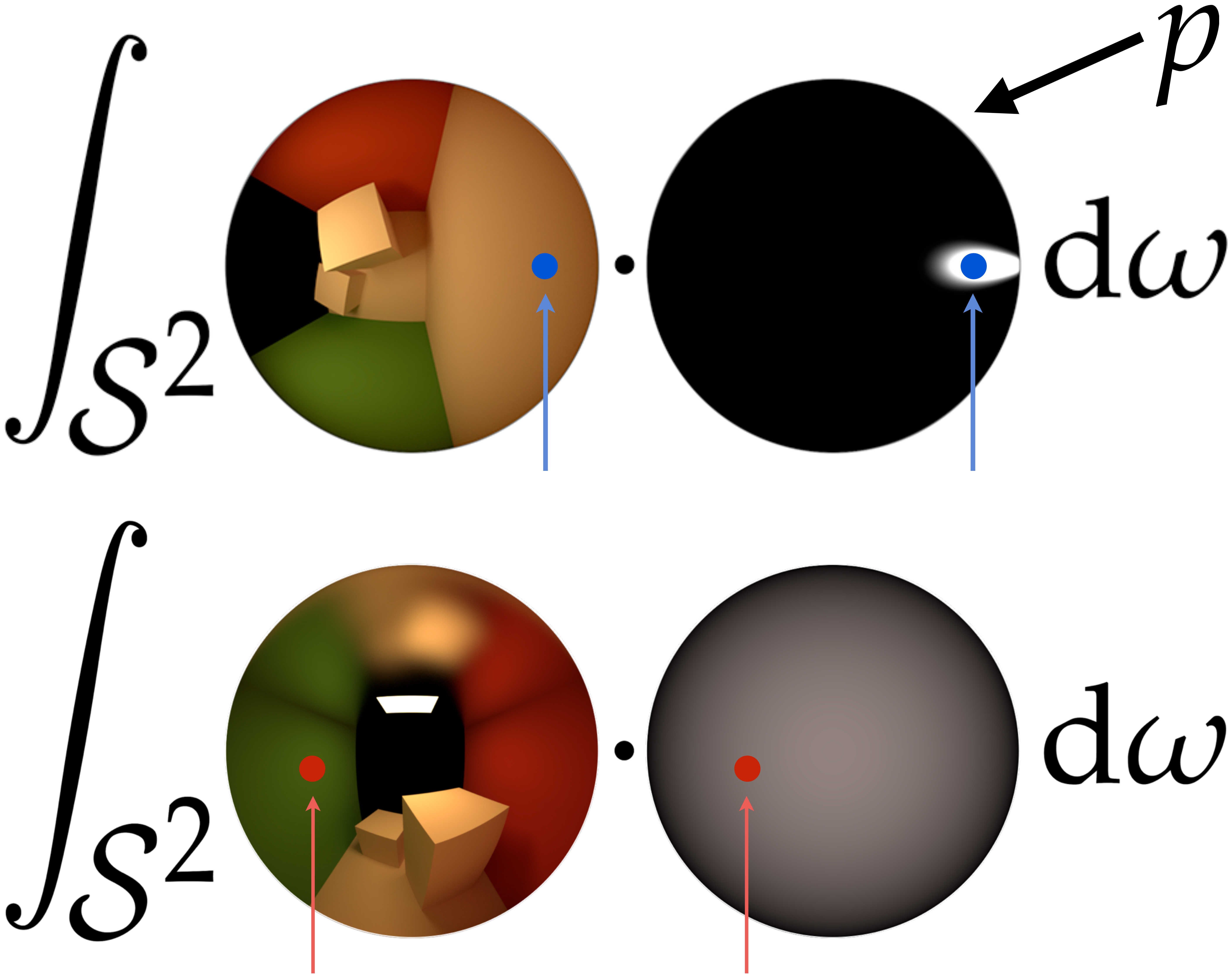
Sampling the integral



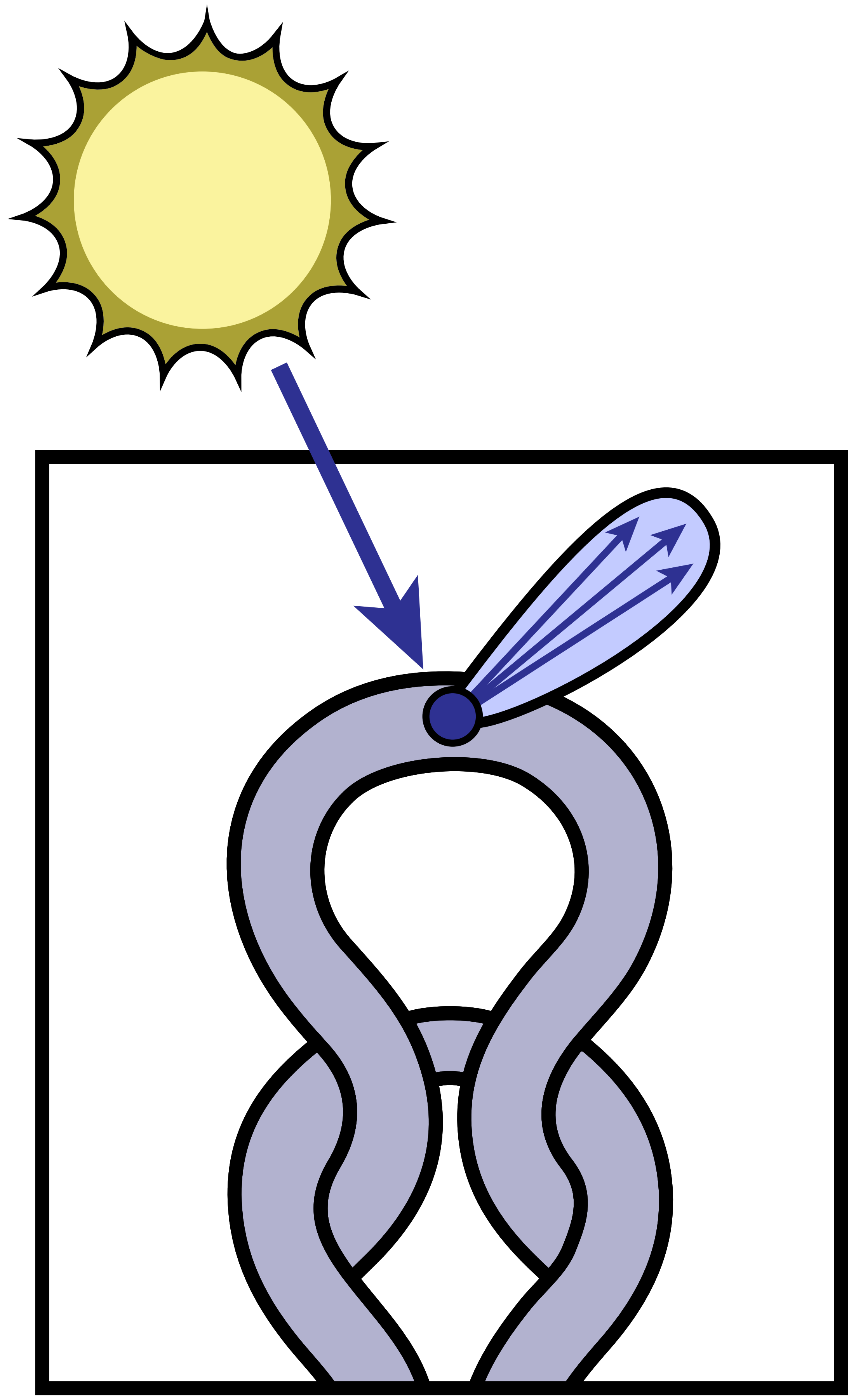


Sampling the integral

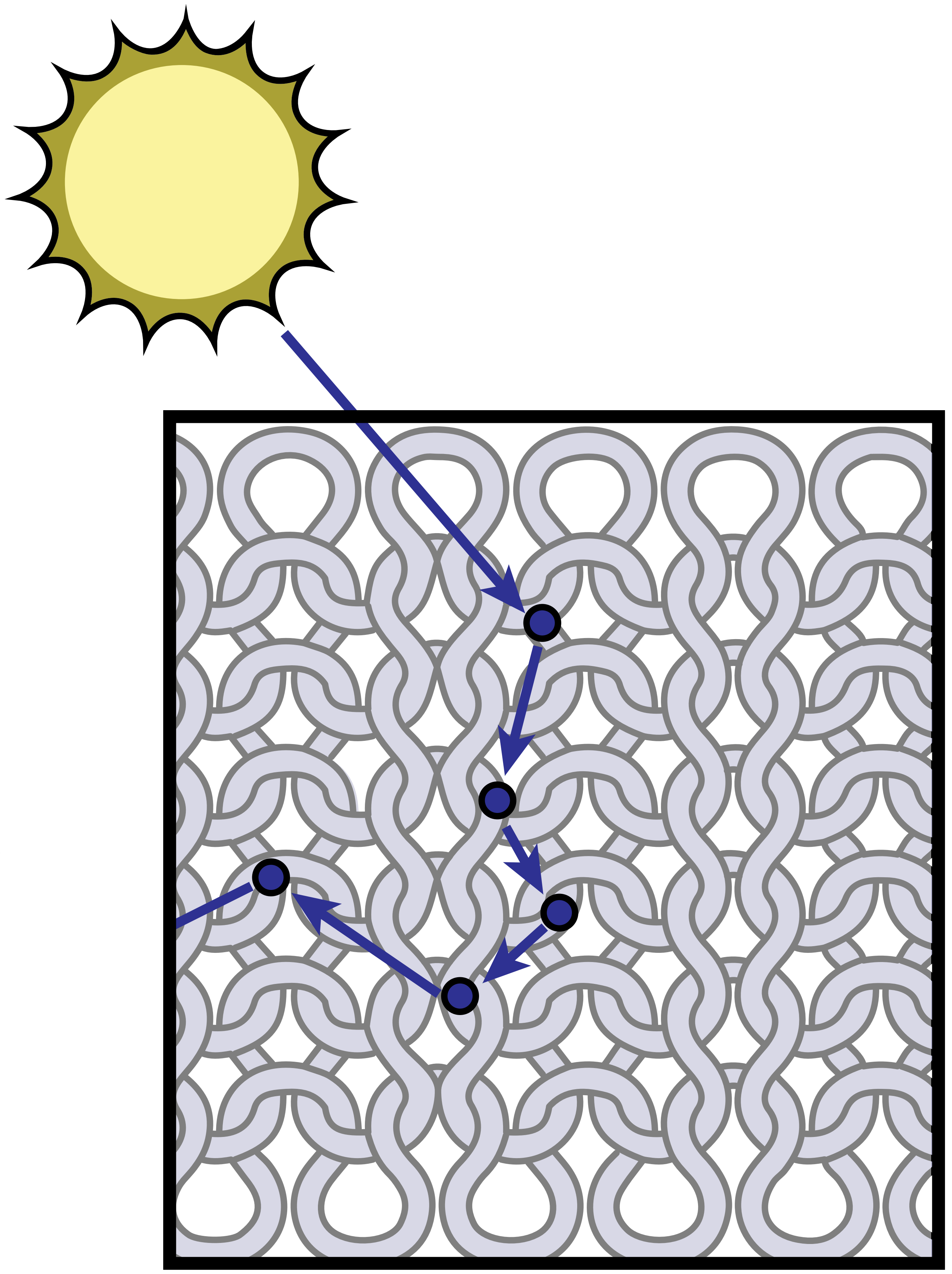




Light paths



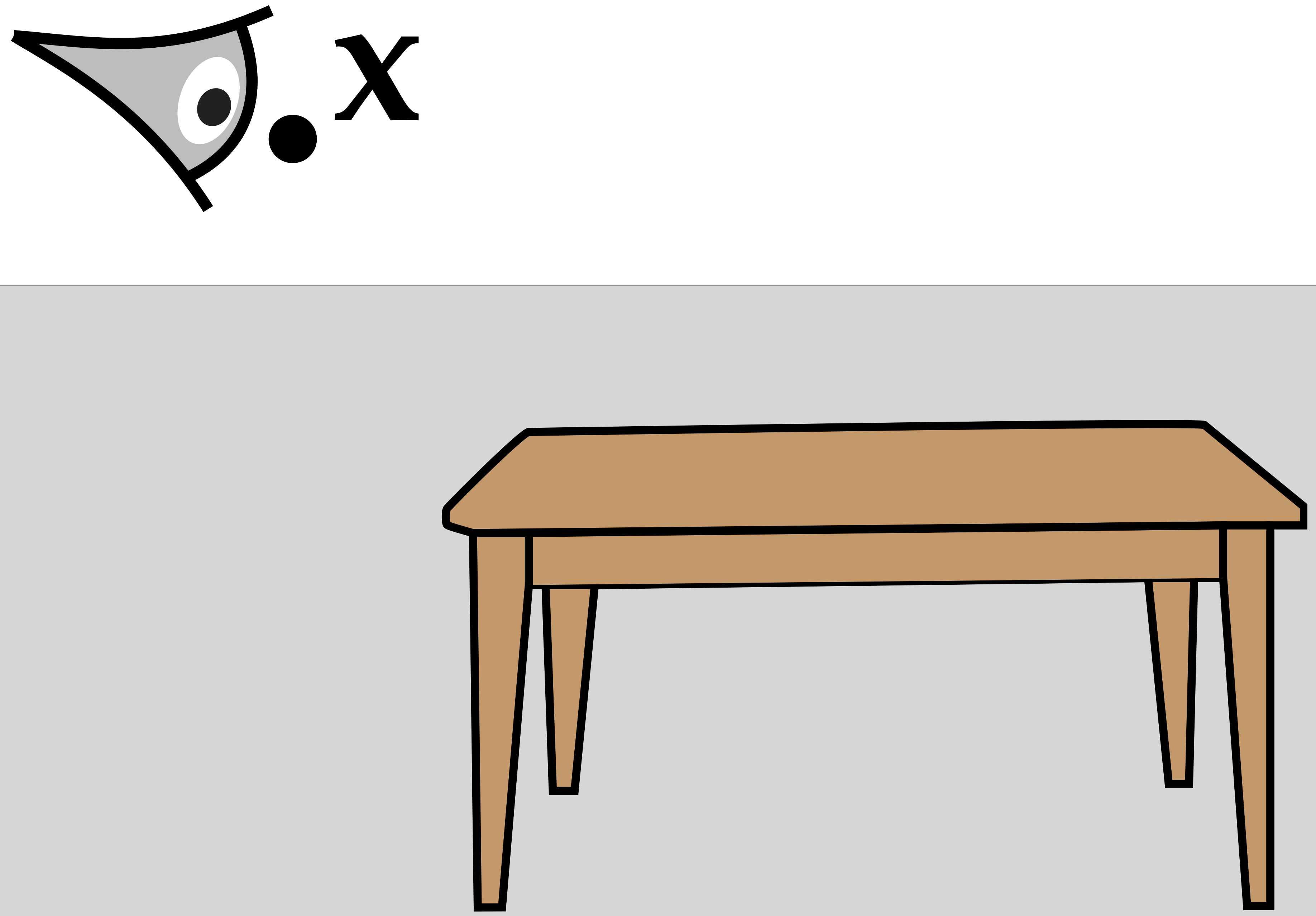
Material model



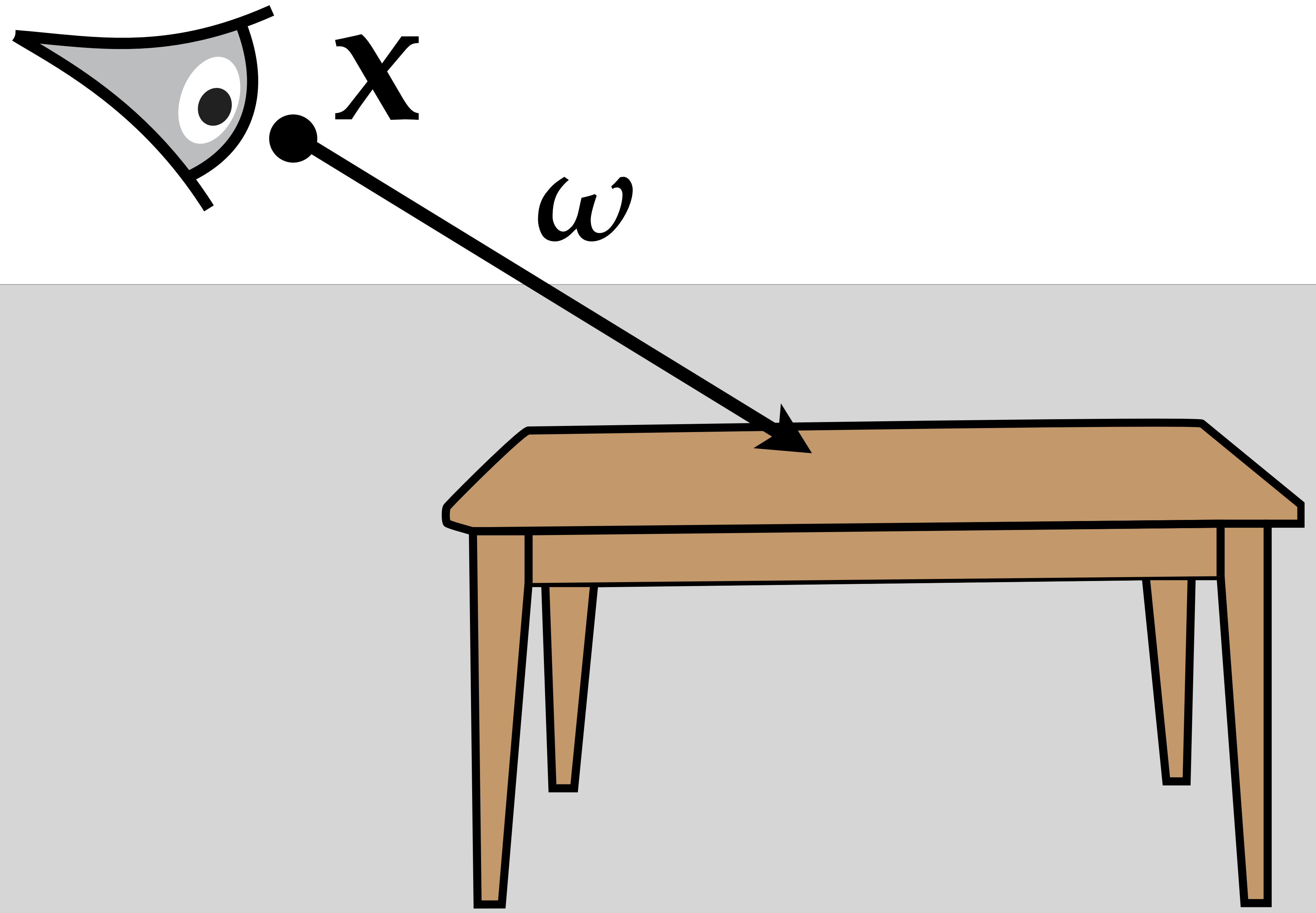
Material-Material interactions

Final rendering

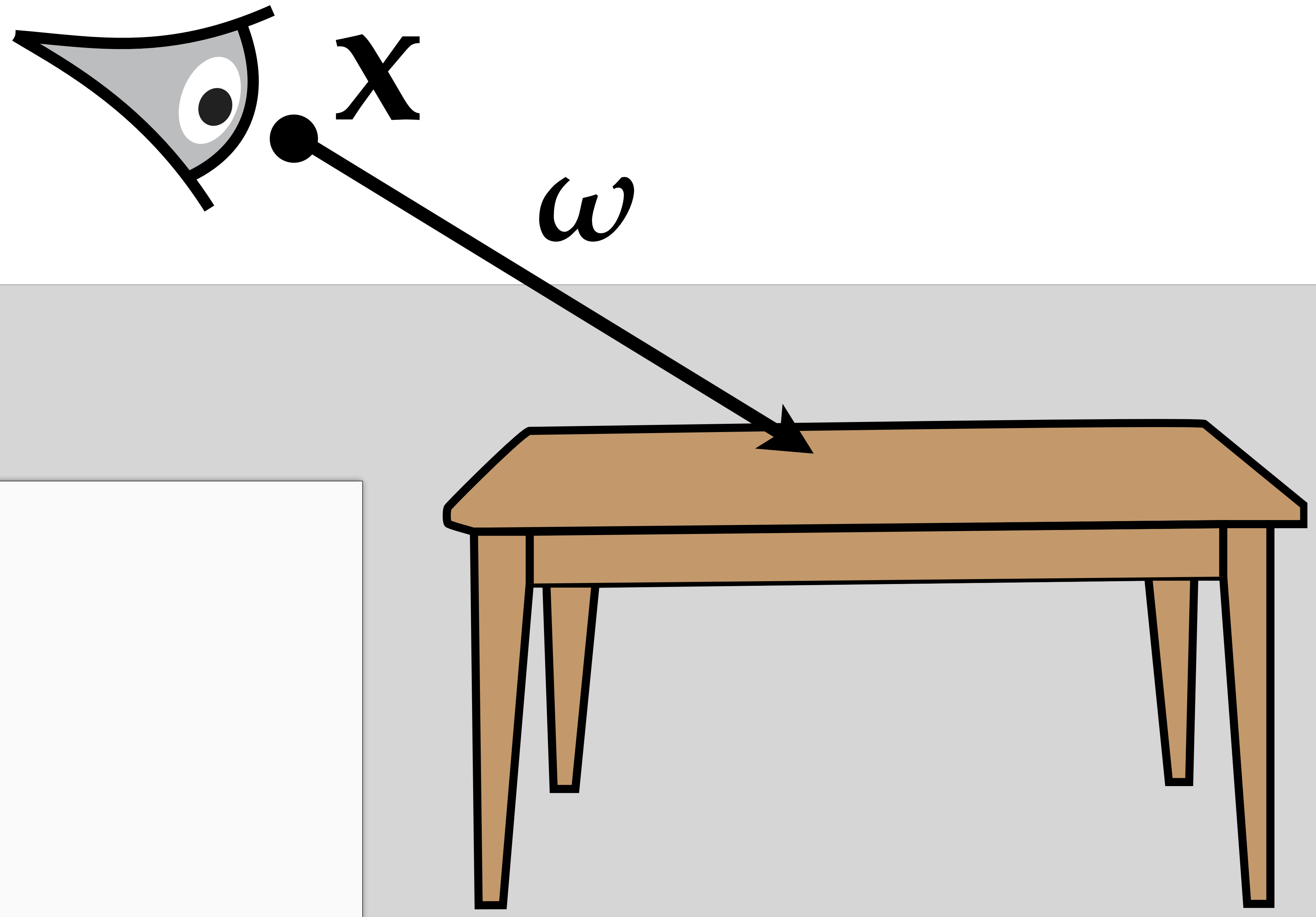
The path tracing algorithm



The path tracing algorithm

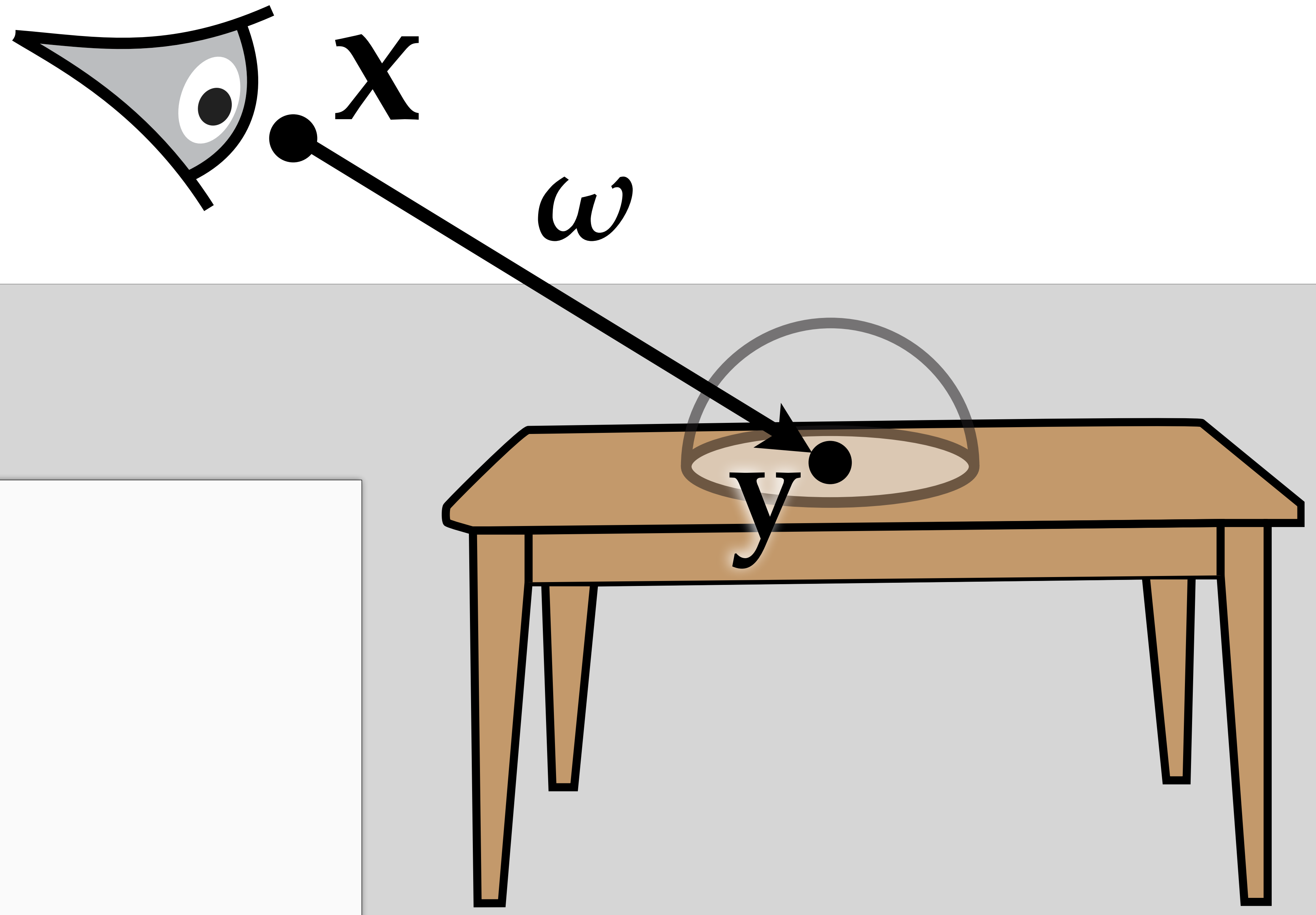


The path tracing algorithm



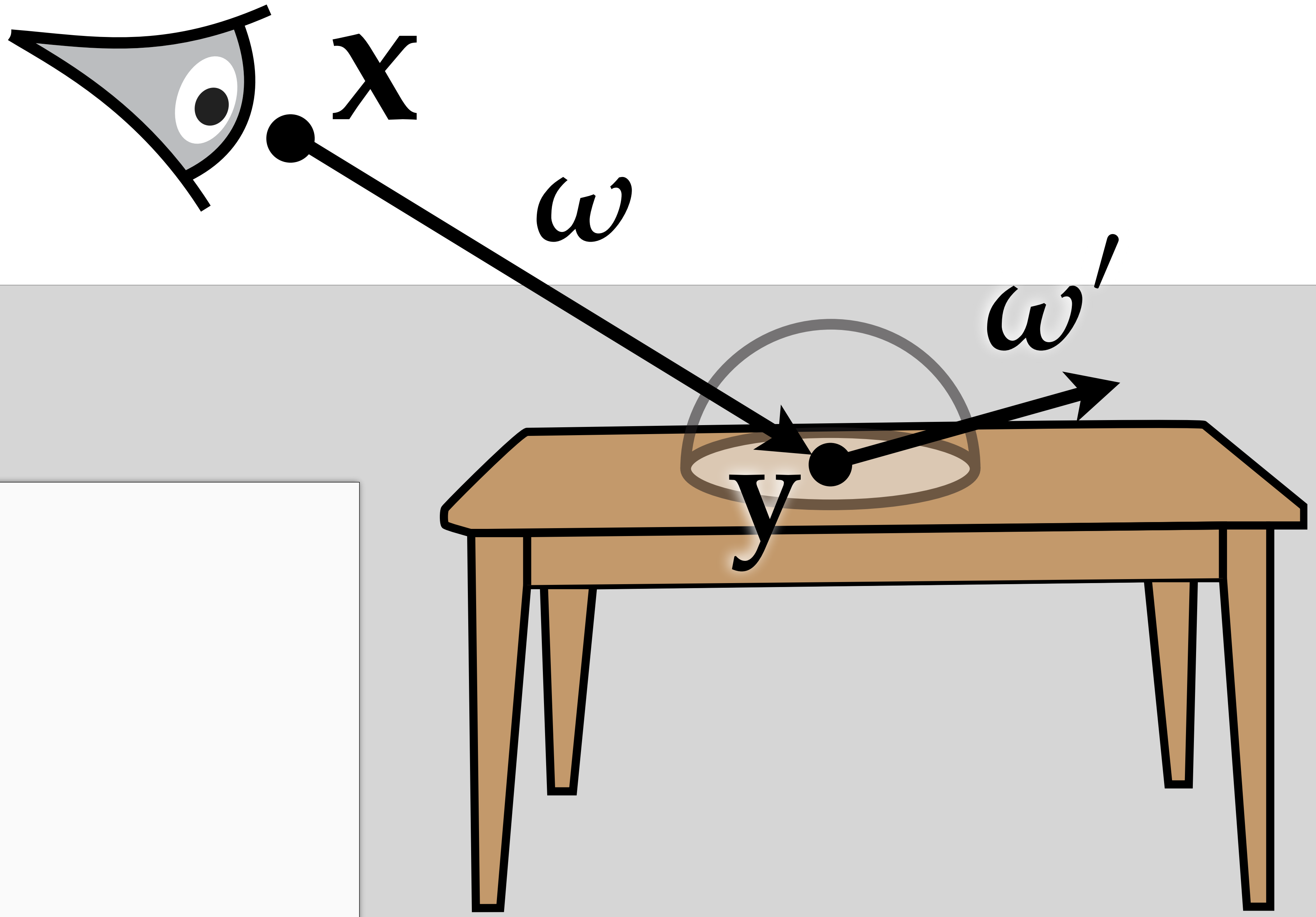
```
def L(x, ω):
    y = intersect(x, ω)
```

The path tracing algorithm



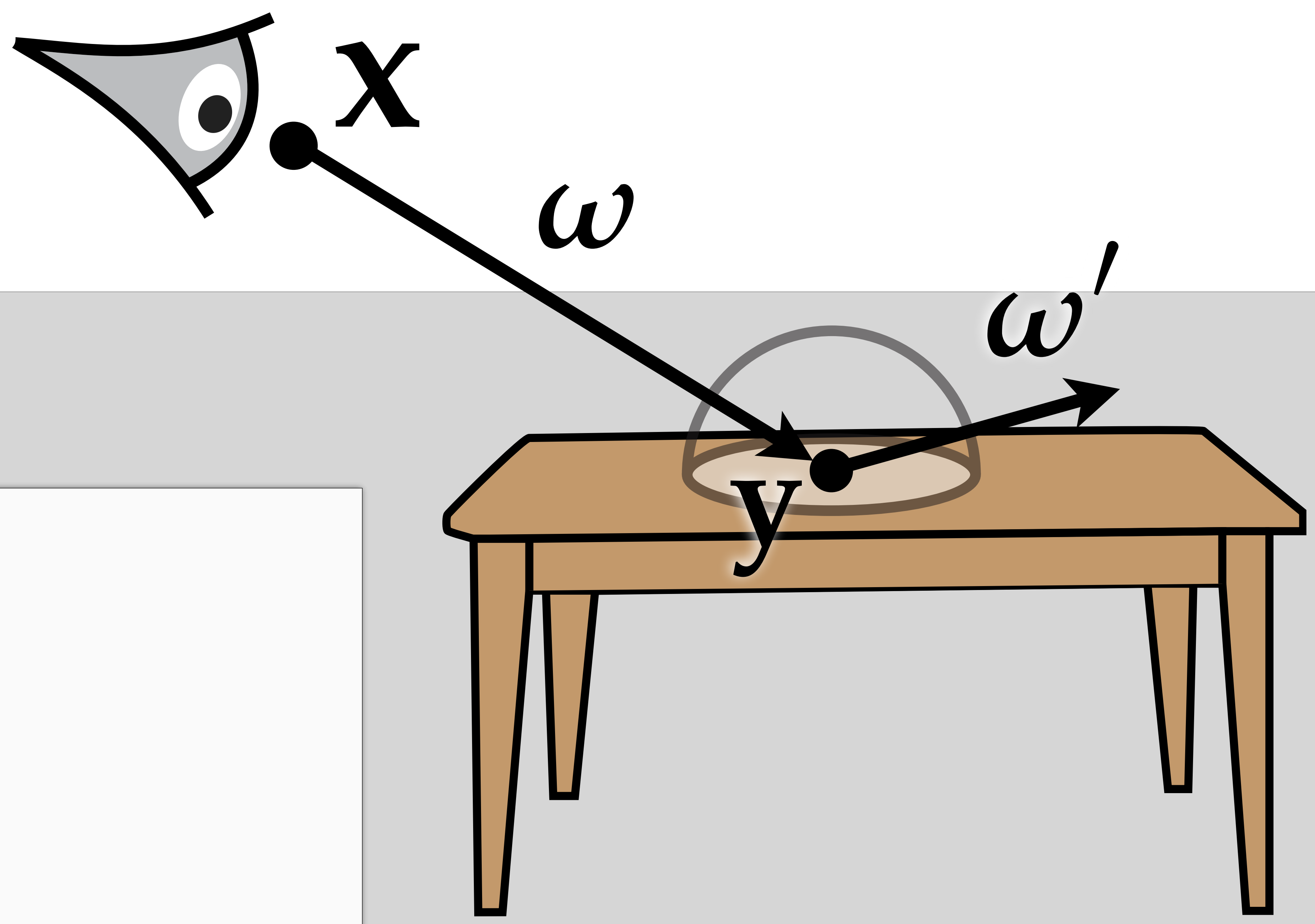
```
def L(x, ω):
    y = intersect(x, ω)
```

The path tracing algorithm



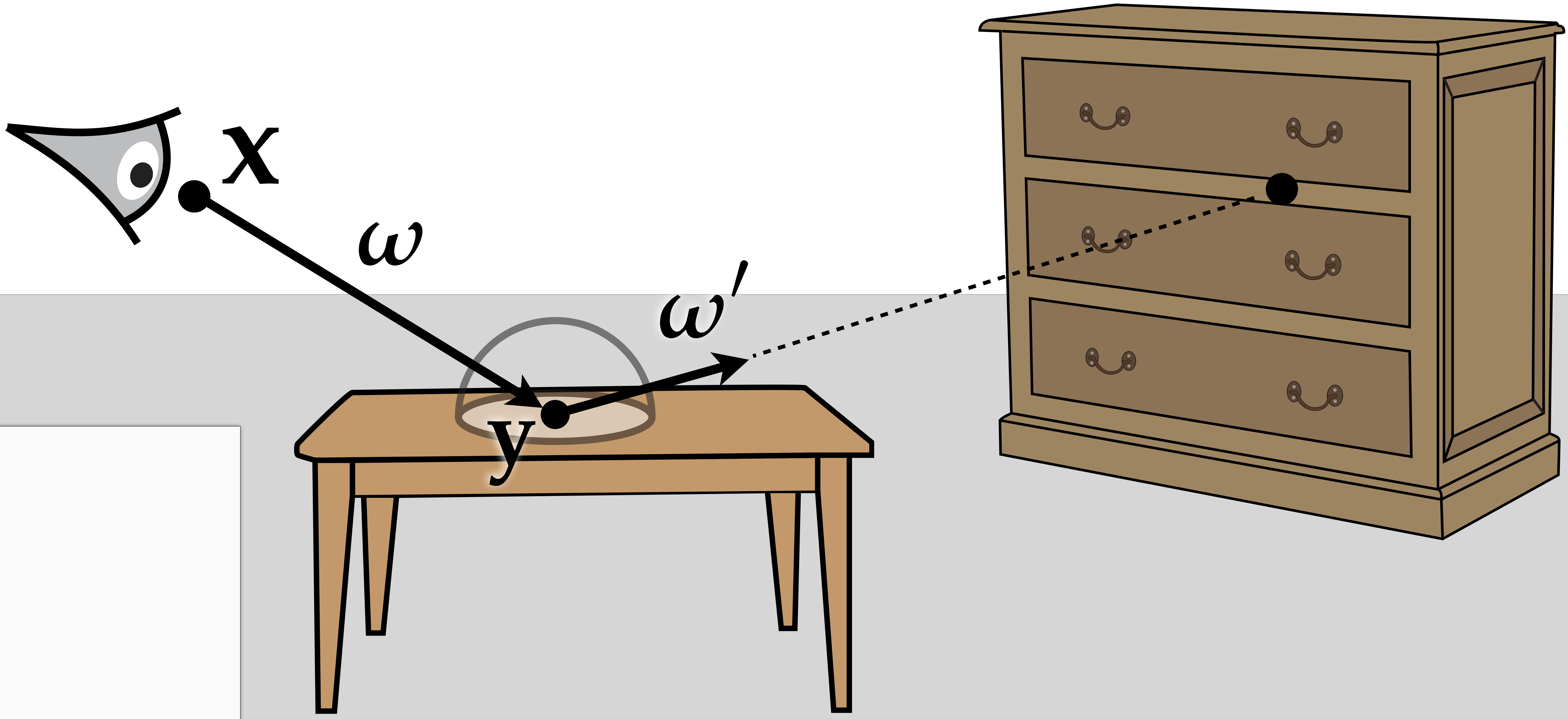
```
def L(x, ω):
    y = intersect(x, ω)

    ω’, weight = scatter(x, ω)
```

The path tracing algorithm



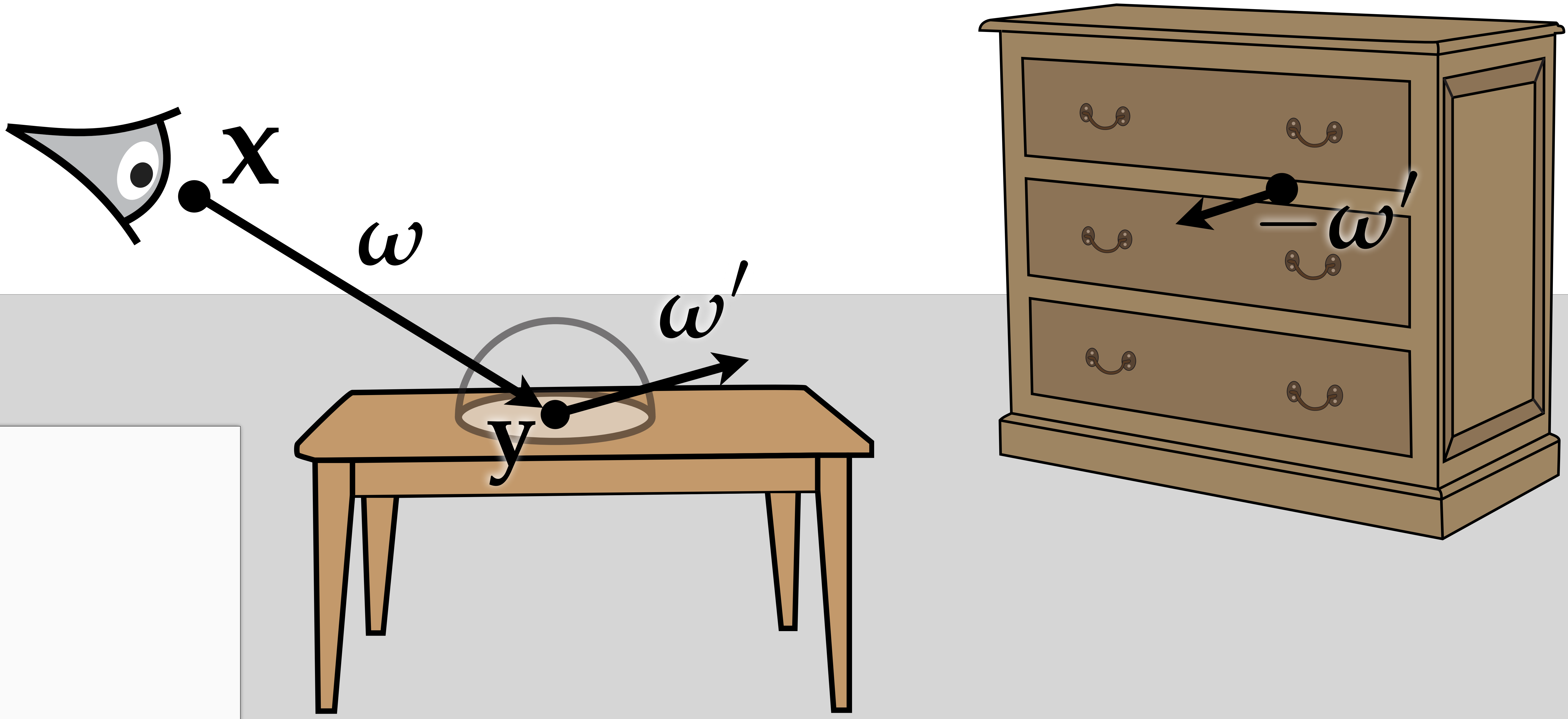
```
def L(x, ω):
    y = intersect(x, ω)

    ω’, weight = scatter(x, ω)

    return y.emission +
           weight * L(y, -ω’)
```

The path tracing algorithm



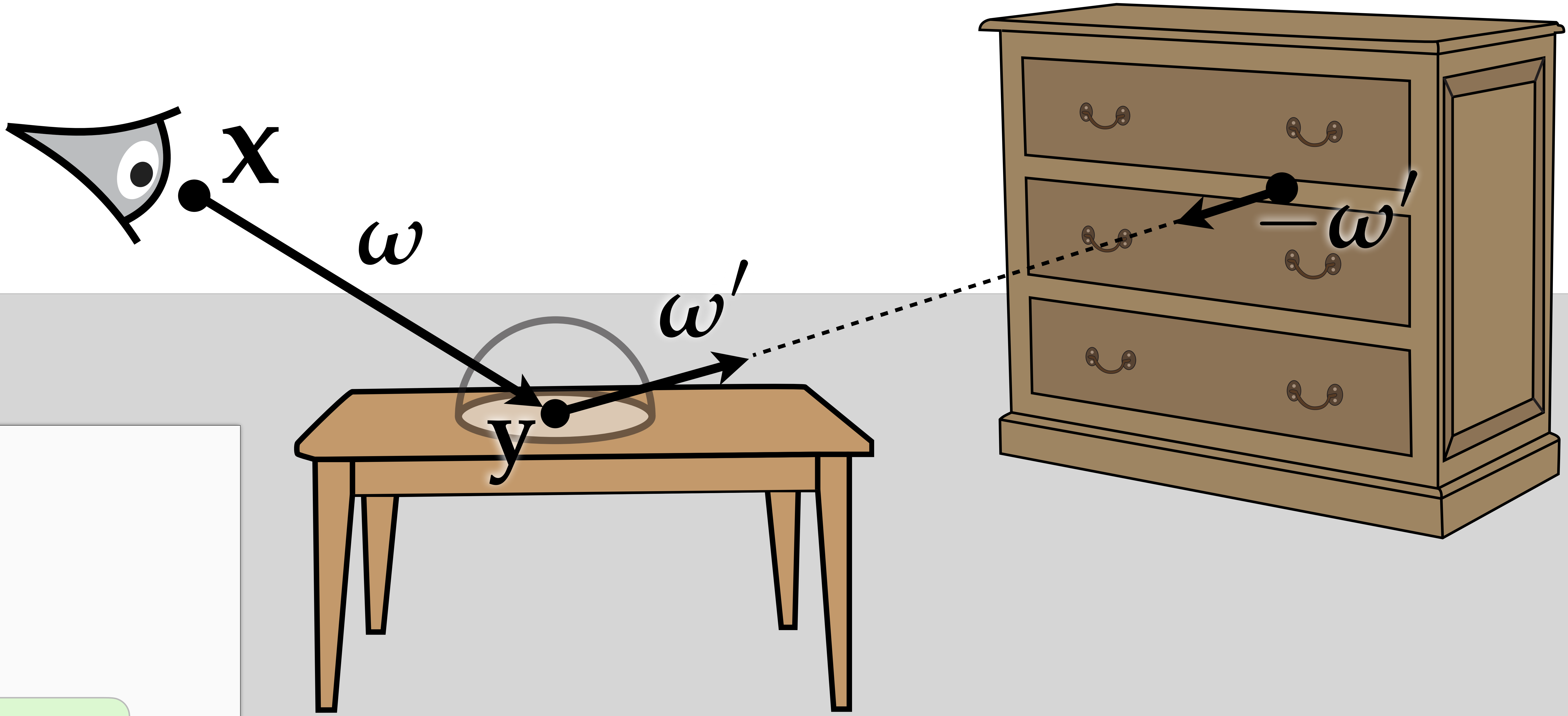
```
def L(x, ω):
    y = intersect(x, ω)

    ω’, weight = scatter(x, ω)

    return y.emission +
           weight * L(y, -ω’)
```

The path tracing algorithm



```
def L(x, ω):
    y = intersect(x, ω)

    ω’, weight = scatter(x, ω)

    return y.emission +
           weight * L(y, -ω’)
```

The path tracing algorithm



```
def L(x, ω):
    y = intersect(x, ω)

    ω’, weight = scatter(x, ω)

    return y.emission +
           weight * L(y, -ω’)
```

uses randomness

The path tracing algorithm

```
def L(x, ω):
    y = intersect(x, ω)

    ω’, weight = scatter(x, ω)

    return y.emission +
           weight * L(y, -ω’)
```

uses randomness

The path tracing algorithm

```
def L($x$, $\omega$, $u_1 \ldots u_n$):
    y = intersect($x$, $\omega$)

    $\omega'$, weight = scatter($x$, $\omega$, $u_1$)

    return y.emission +
          weight * L($y$, $-\omega'$, $u_2 \ldots u_n$)
```

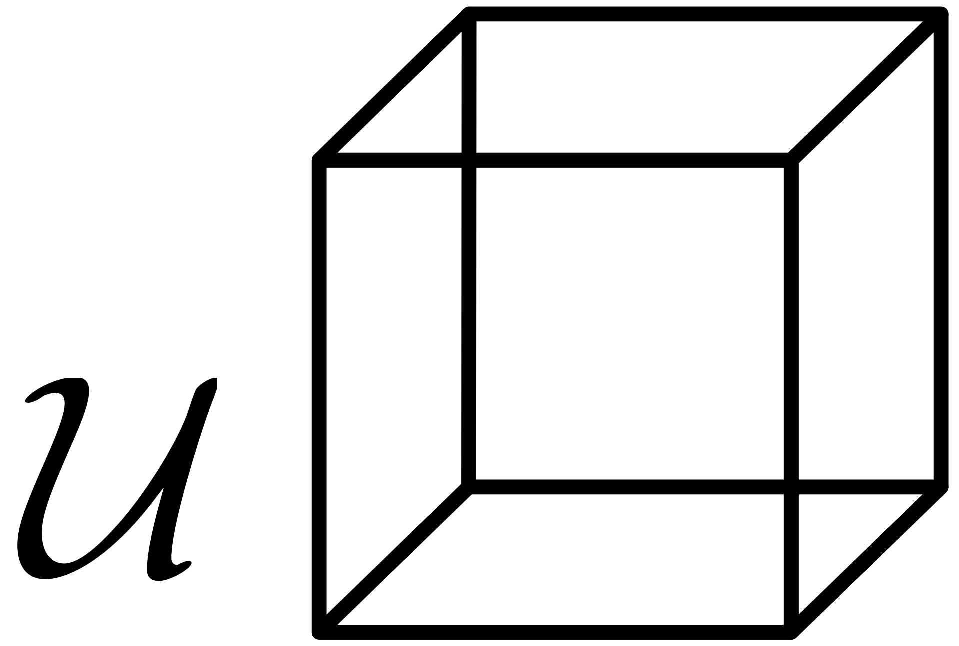
The path tracing algorithm

```
def L(x, ω, u):
    y = intersect(x, ω)

    ω’, weight = scatter(x, ω, $u_1$)

    return y.emission +
           weight * L(y, -ω’, $u_2 \dots u_n$)
```

Another interpretation

```
def L(x, ω, $\mathbf{u}$):
    y = intersect(x, ω)

    ω’, weight = scatter(x, ω, $u_1$)

    return y.emission +
           weight * L(y, -ω’, $u_2 \ldots u_n$)
```

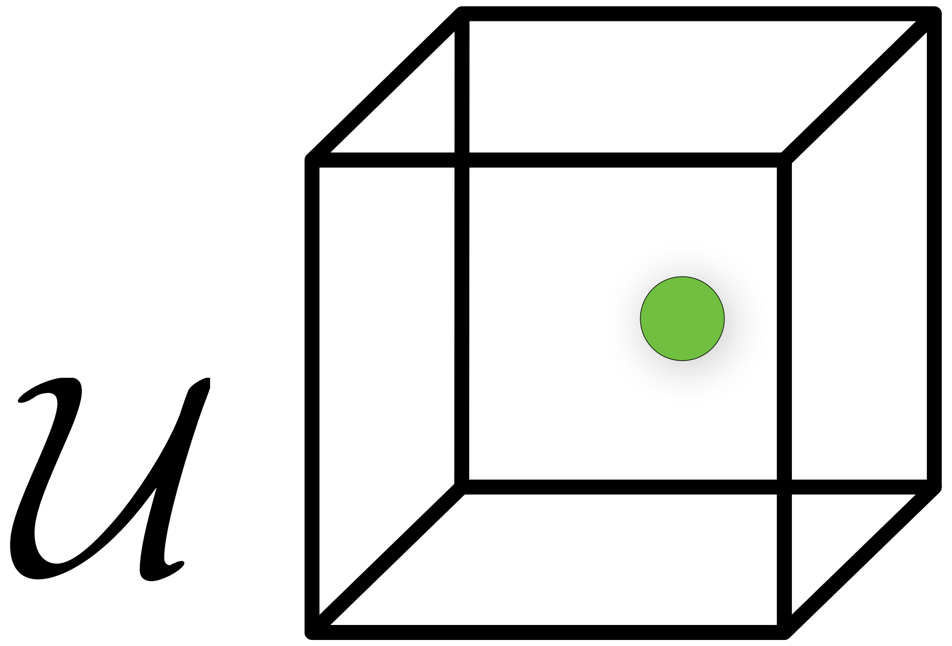
Another interpretation



Hypercube of
“random numbers”

```
def L(x, ω, u):
    y = intersect(x, ω)

    ω’, weight = scatter(x, ω, u1)

    return y.emission +
           weight * L(y, -ω’, u2…un)
```

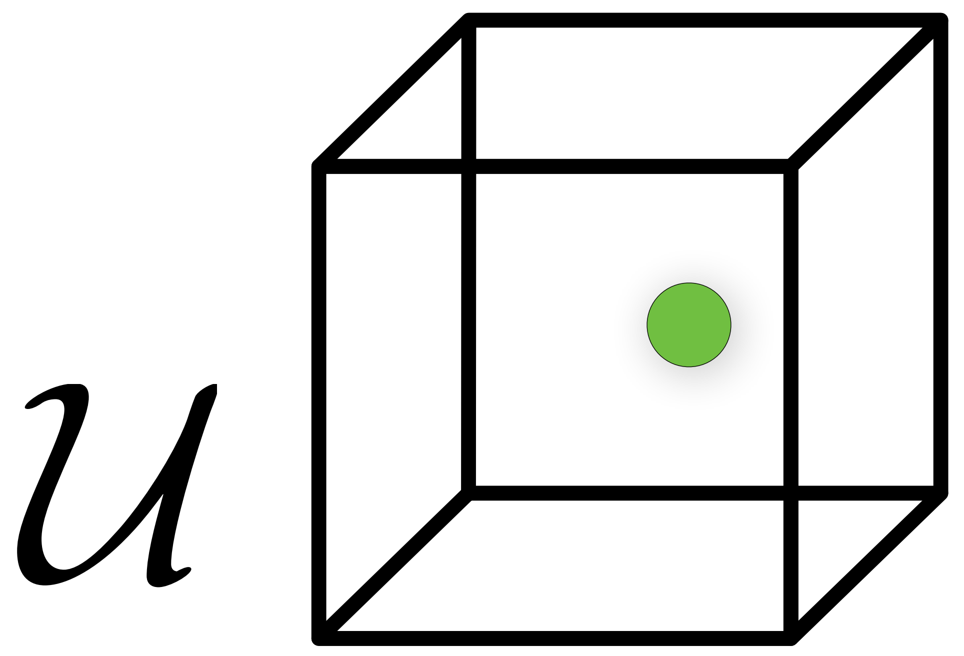
Another interpretation



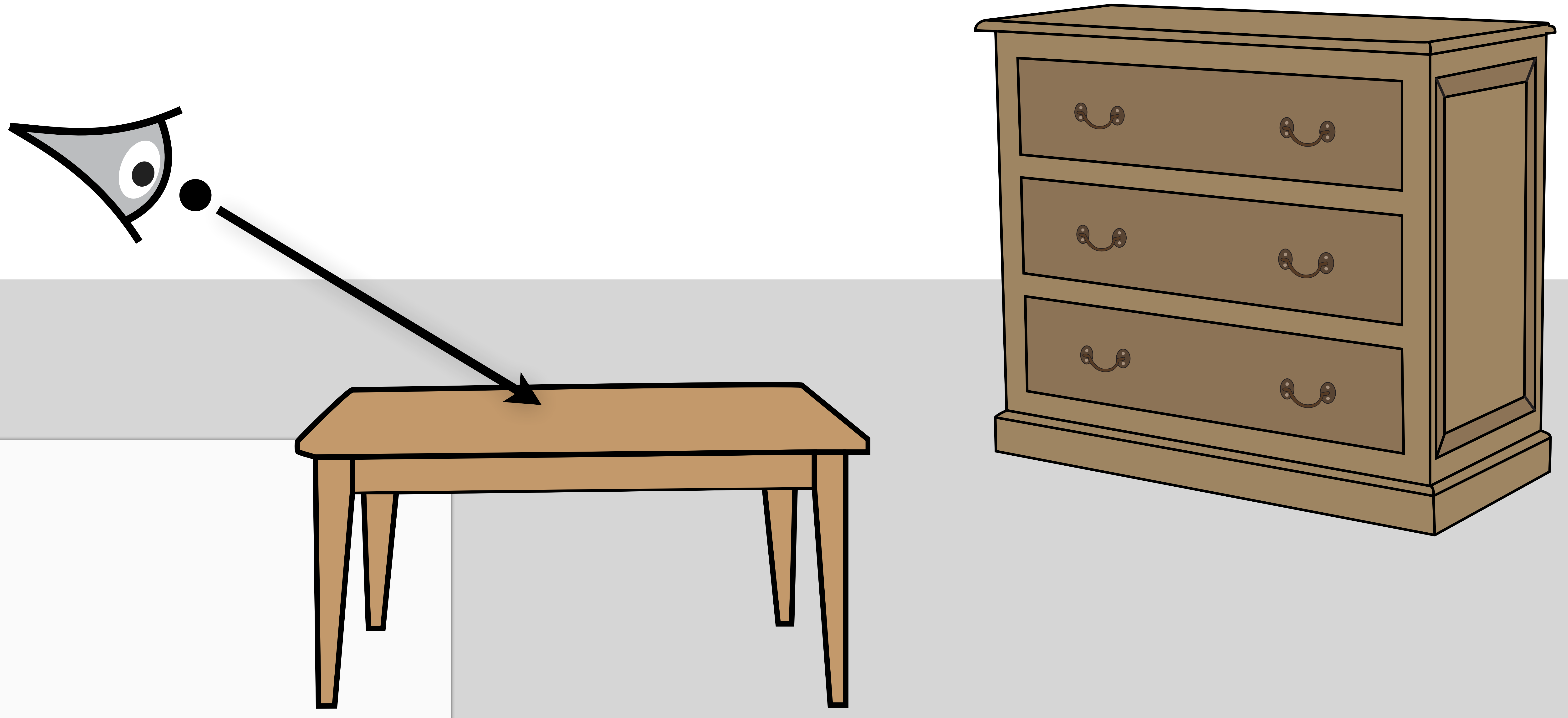
Hypercube of
“random numbers”

```
def L(x, ω, u):
    y = intersect(x, ω)

    ω’, weight = scatter(x, ω, u1)

    return y.emission +
           weight * L(y, -ω’, u2…un)
```

Another interpretation

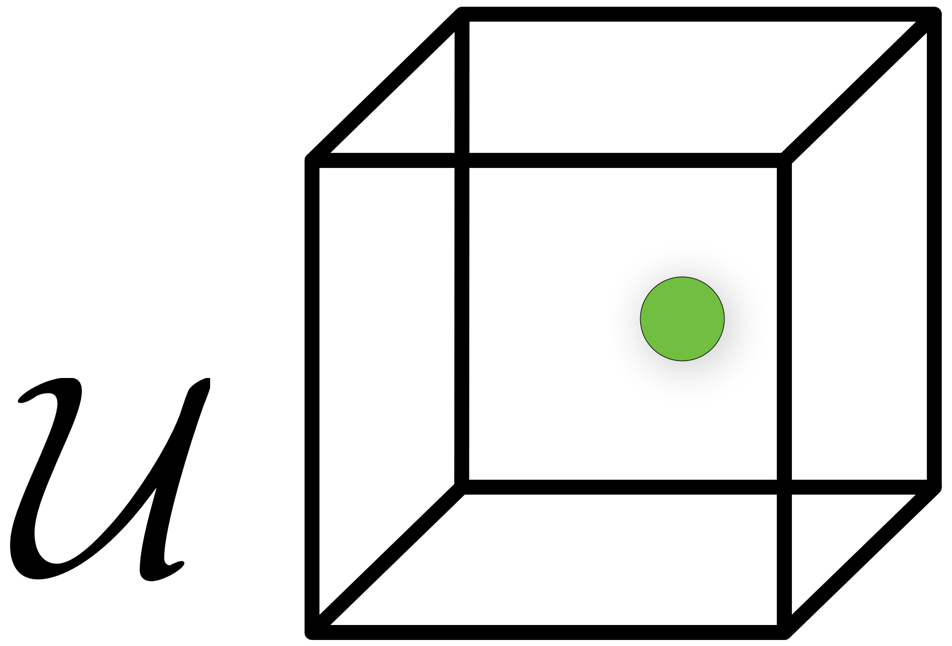


Hypercube of
"random numbers"

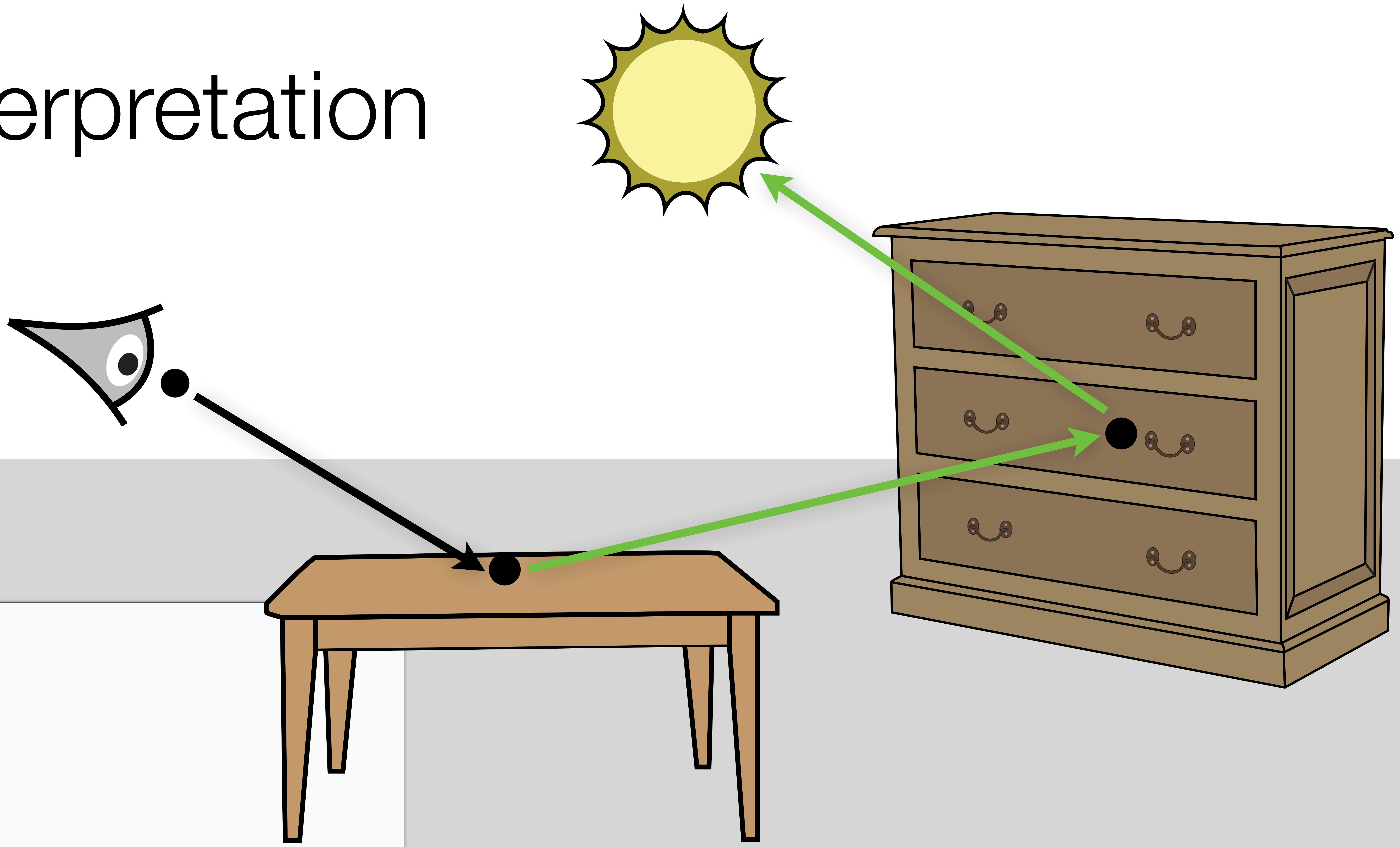


```
def L(x, ω, u):
    y = intersect(x, ω)

    ω', weight = scatter(x, ω, u1)

    return y.emission +
           weight * L(y, -ω', u2…un)
```

Another interpretation

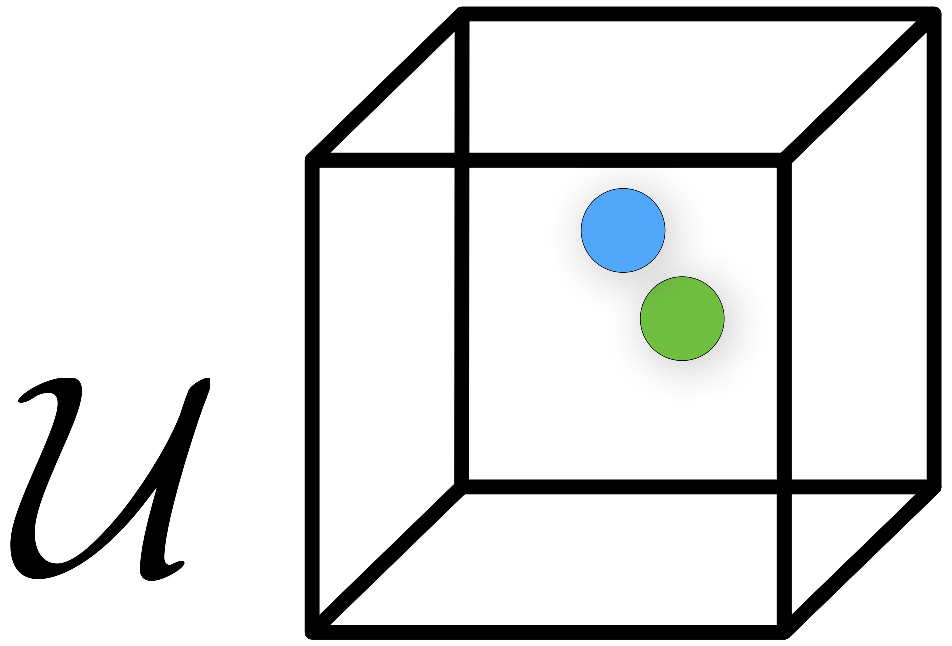


Hypercube of
"random numbers"

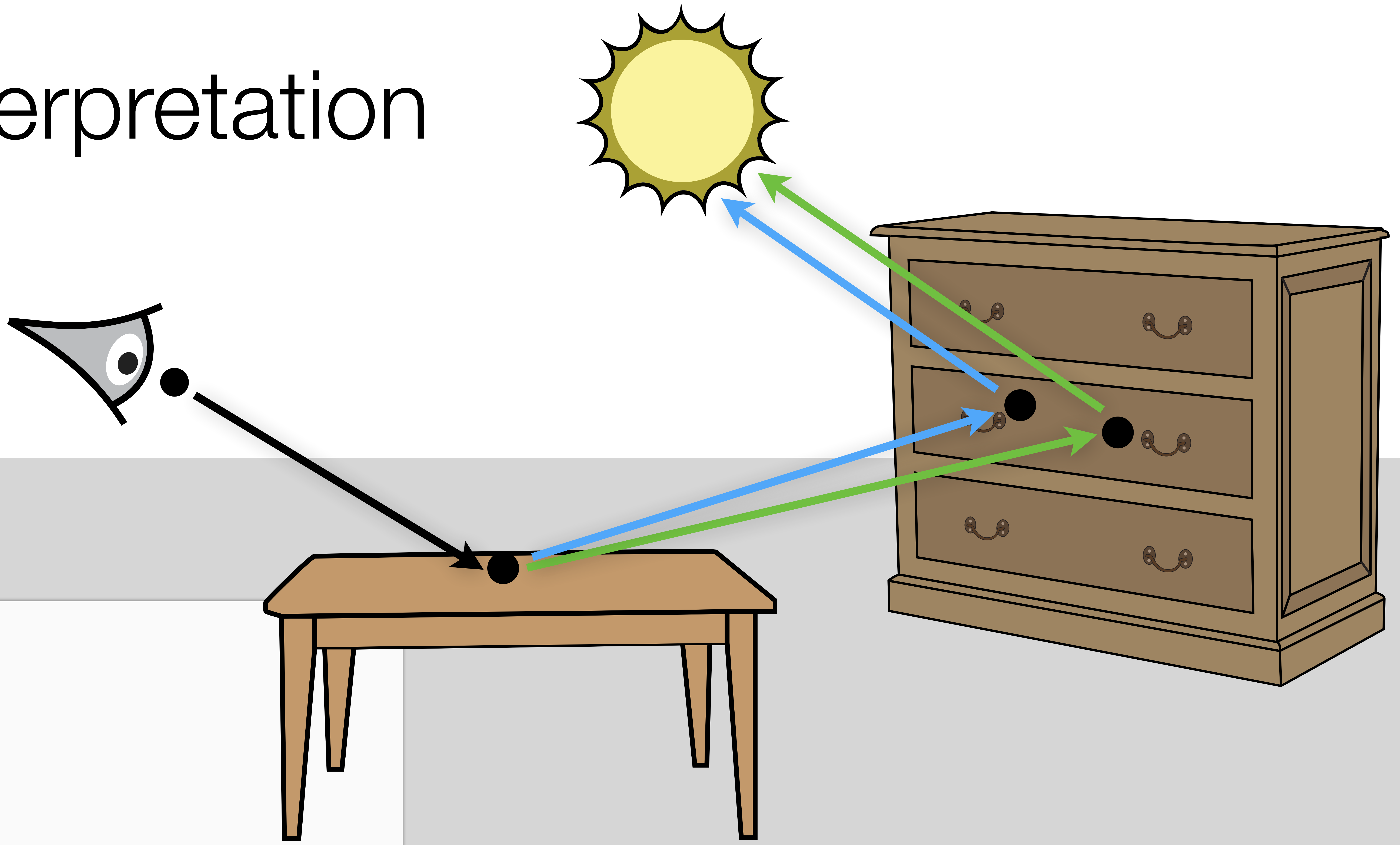


```
def L(x, ω, u):
    y = intersect(x, ω)

    ω’, weight = scatter(x, ω, u1)

    return y.emission +
           weight * L(y, -ω’, u2…un)
```

Another interpretation

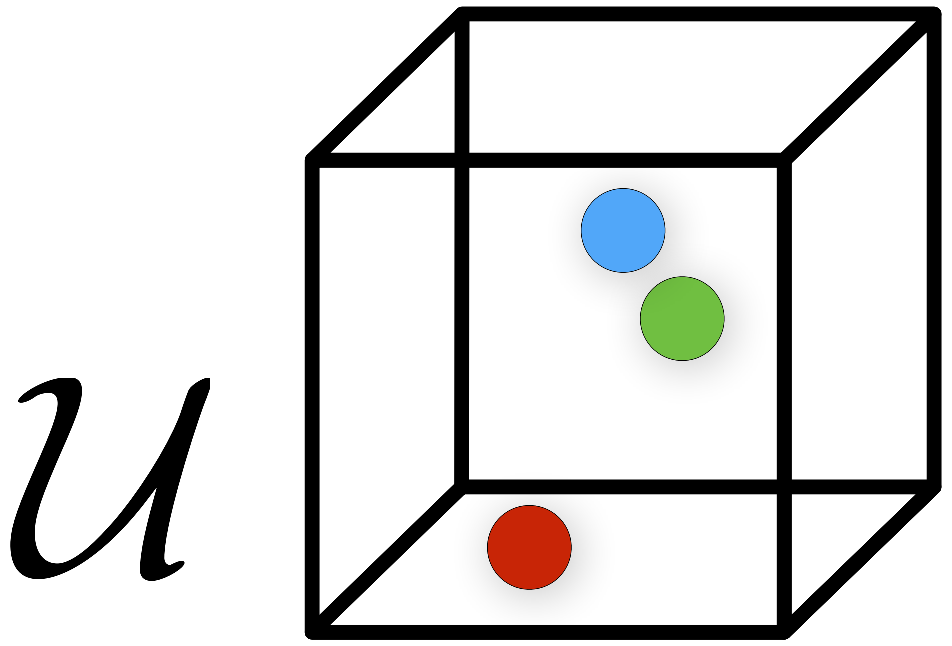


Hypercube of
“random numbers”

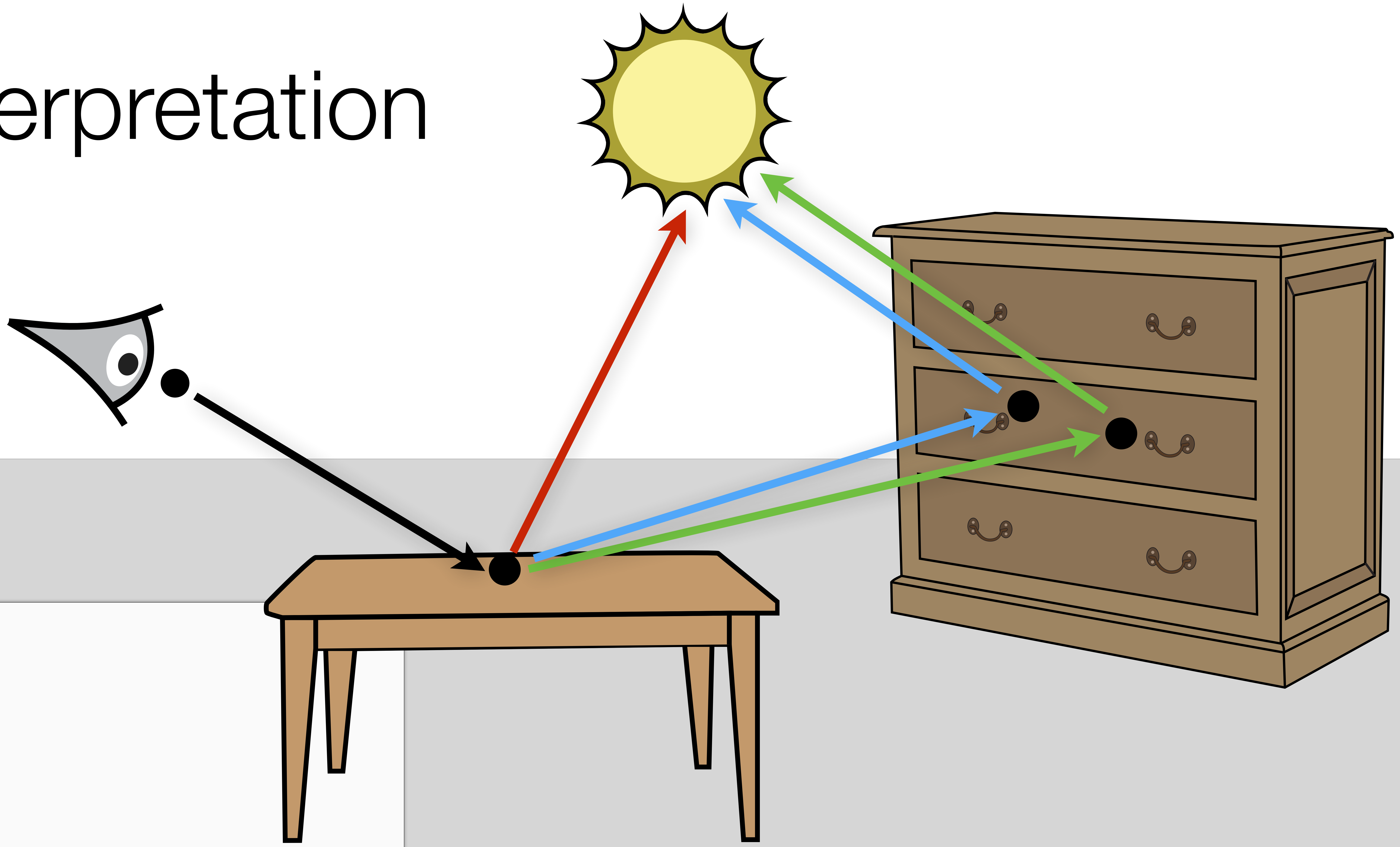


```
def L(x, ω, u):
    y = intersect(x, ω)

    ω’, weight = scatter(x, ω, u1)

    return y.emission +
           weight * L(y, -ω’, u2…un)
```

Another interpretation

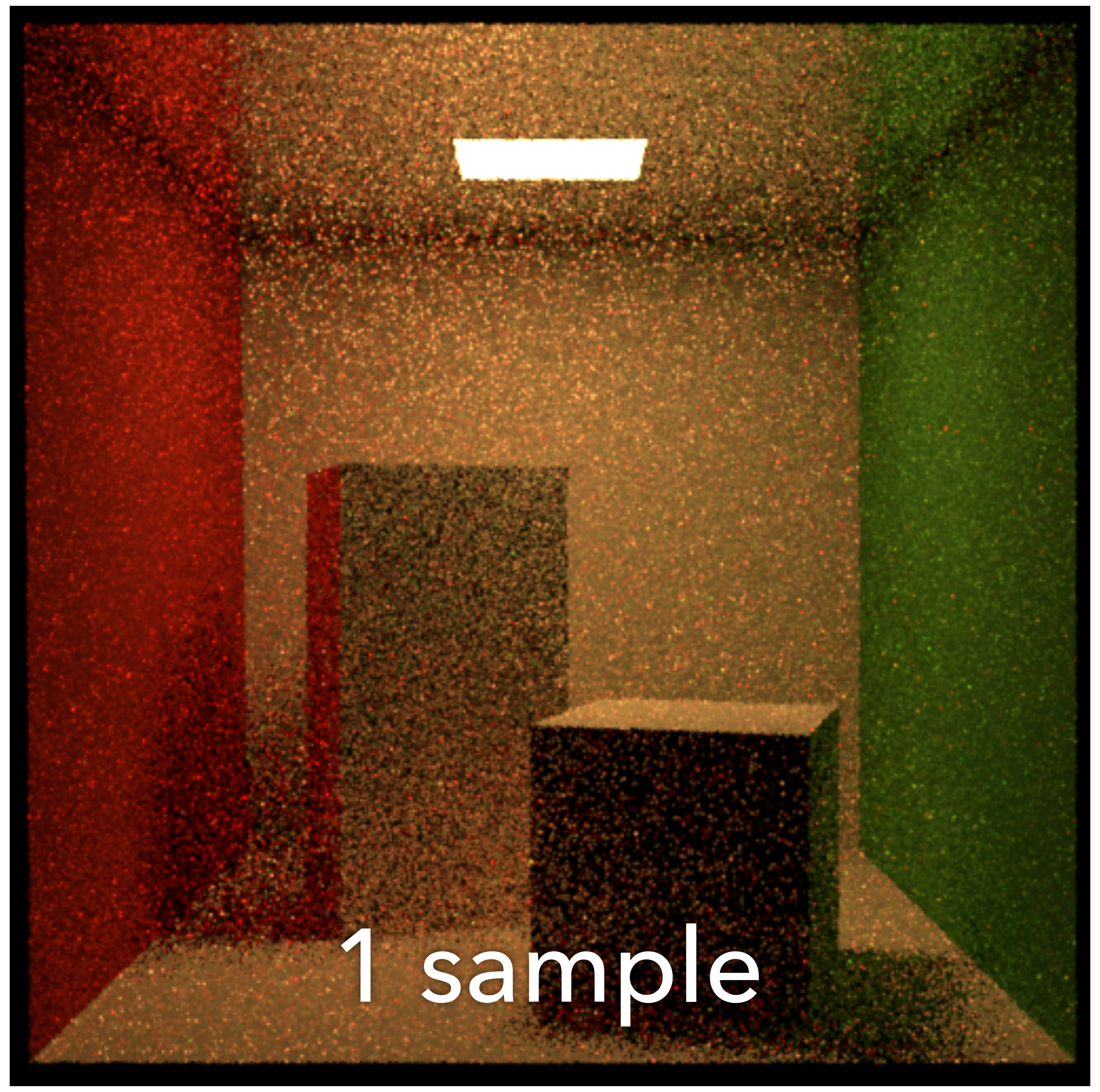


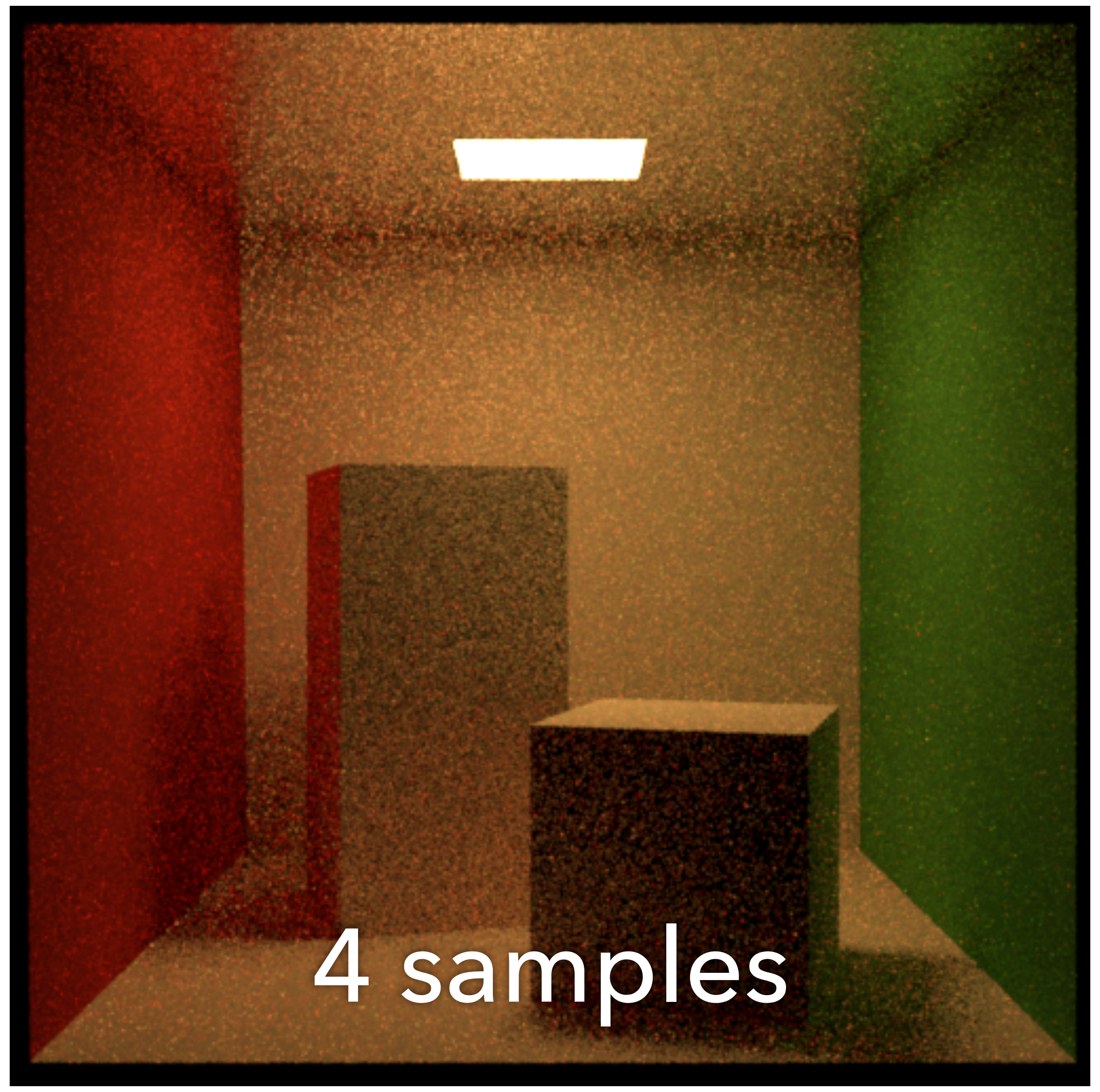
Hypercube of
“random numbers”

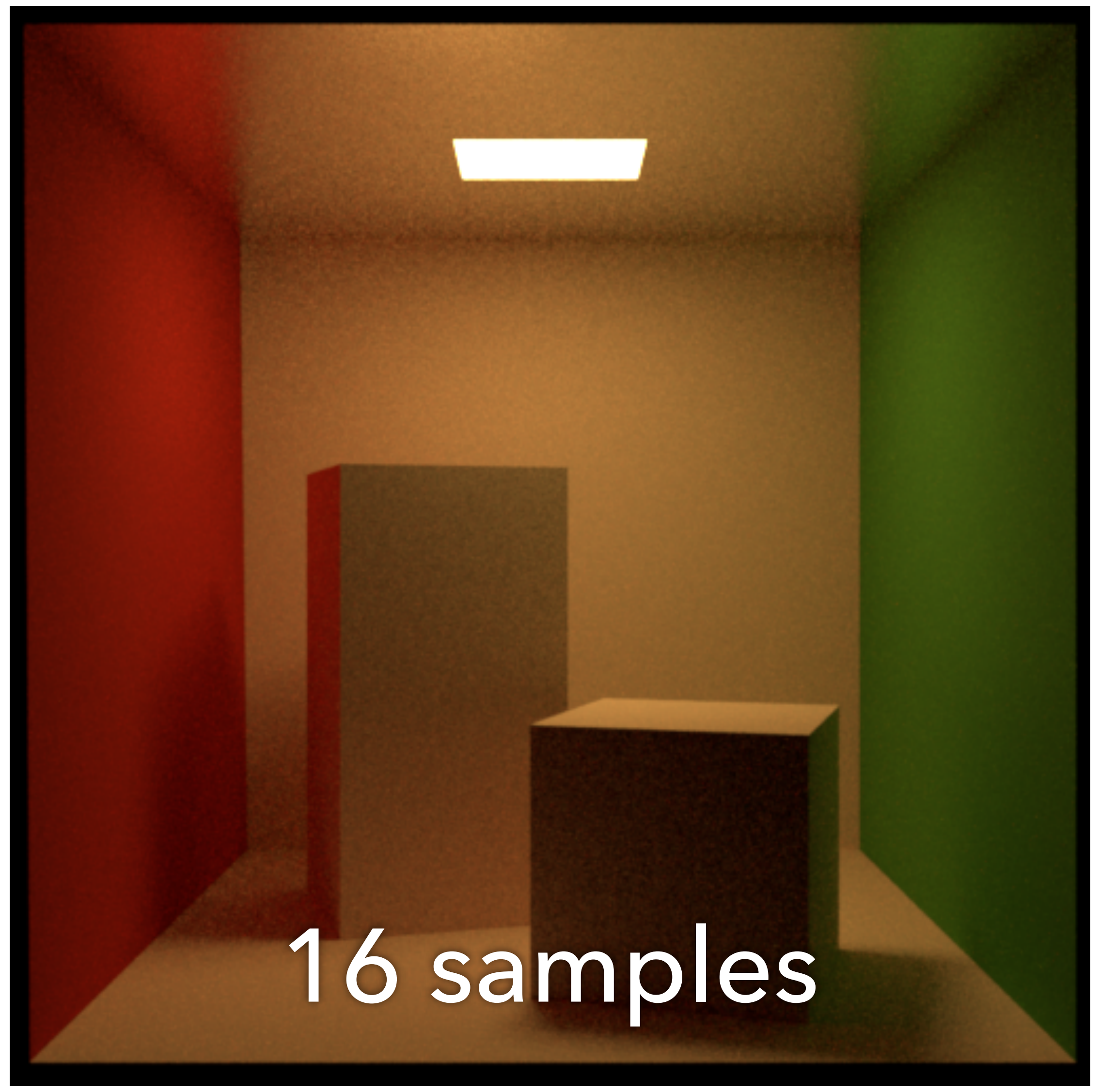


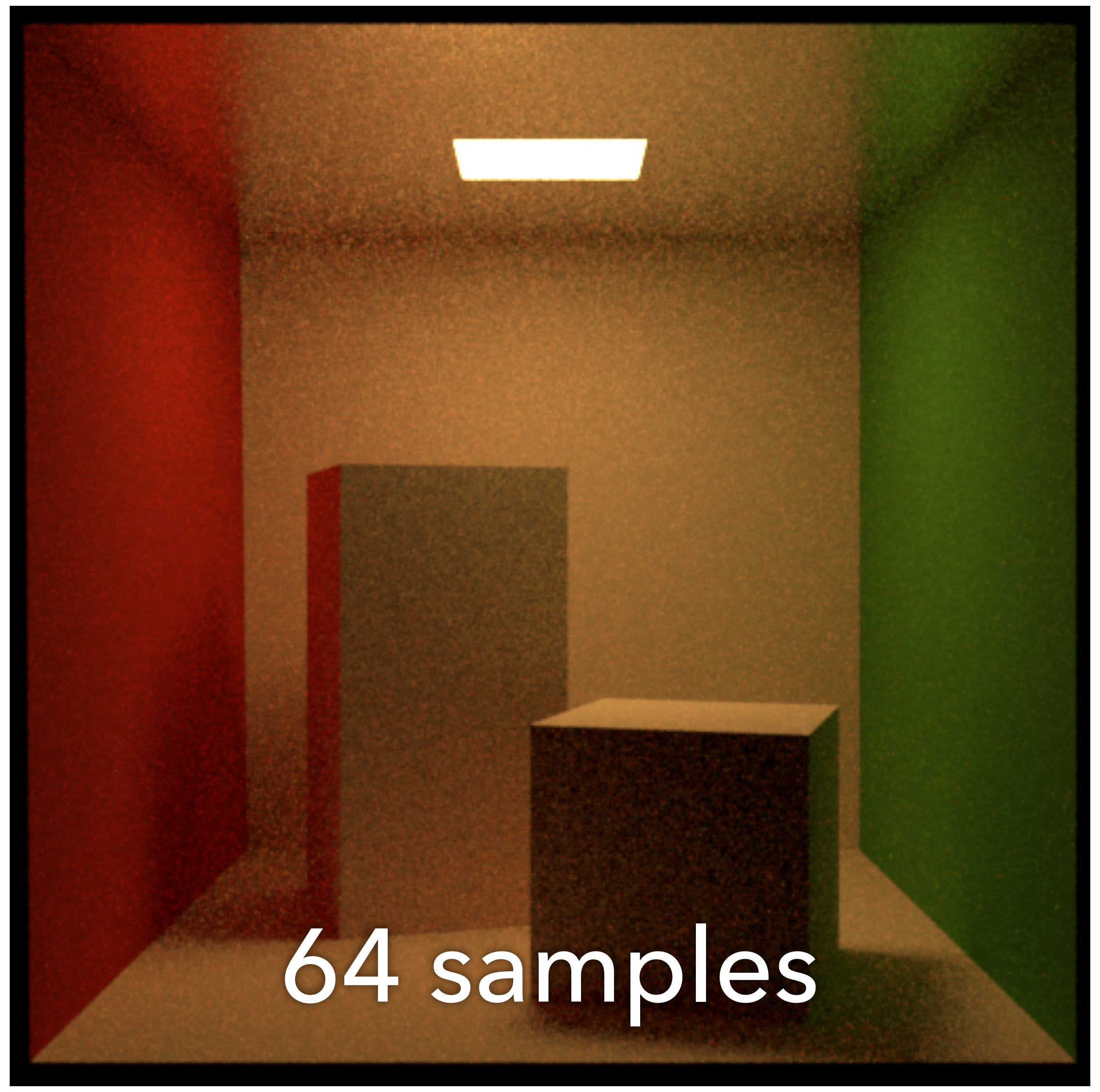
```
def L(x, ω, u):
    y = intersect(x, ω)

    ω’, weight = scatter(x, ω, u1)

    return y.emission +
           weight * L(y, -ω’, u2…un)
```

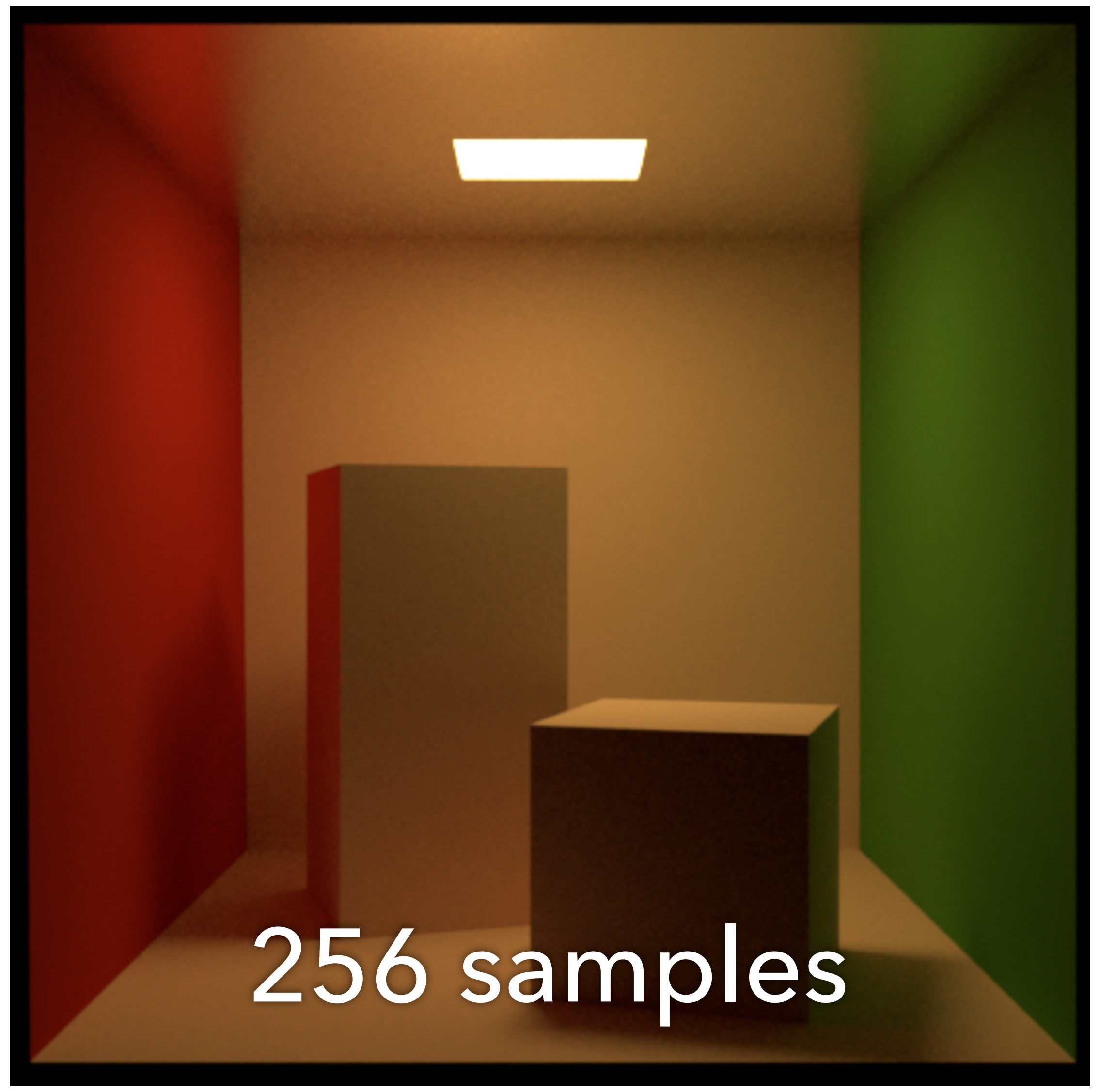
Convergence

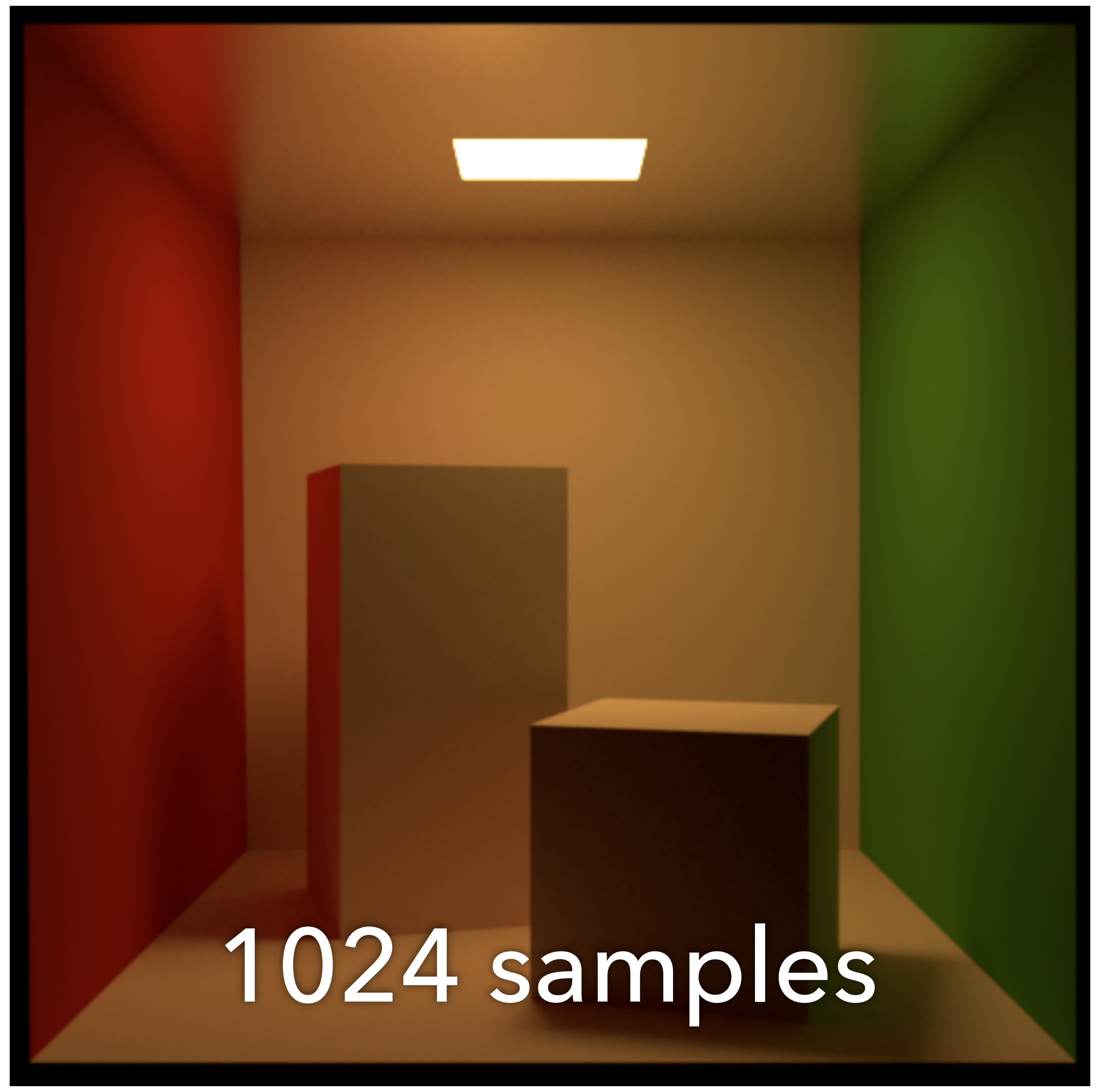




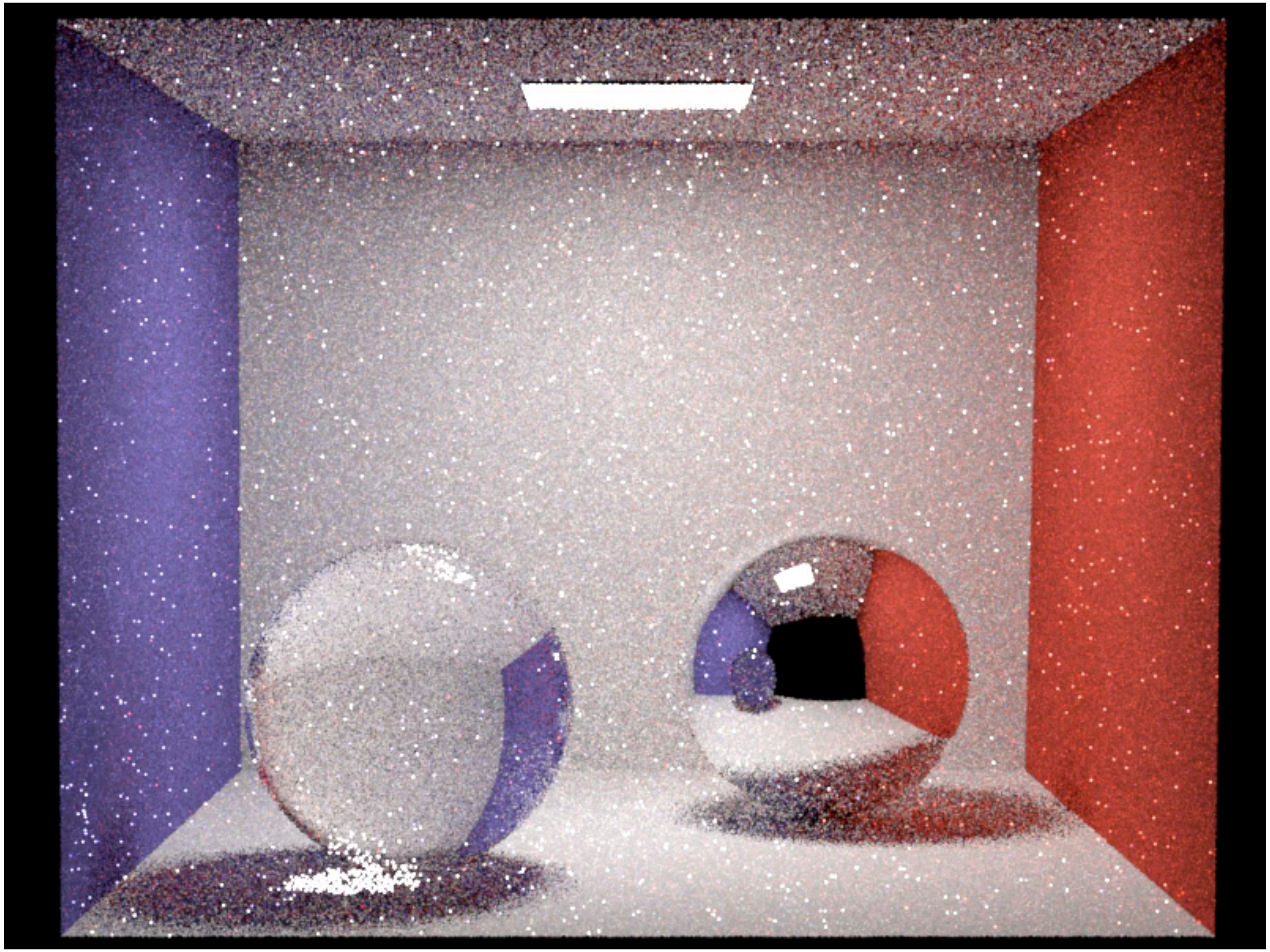




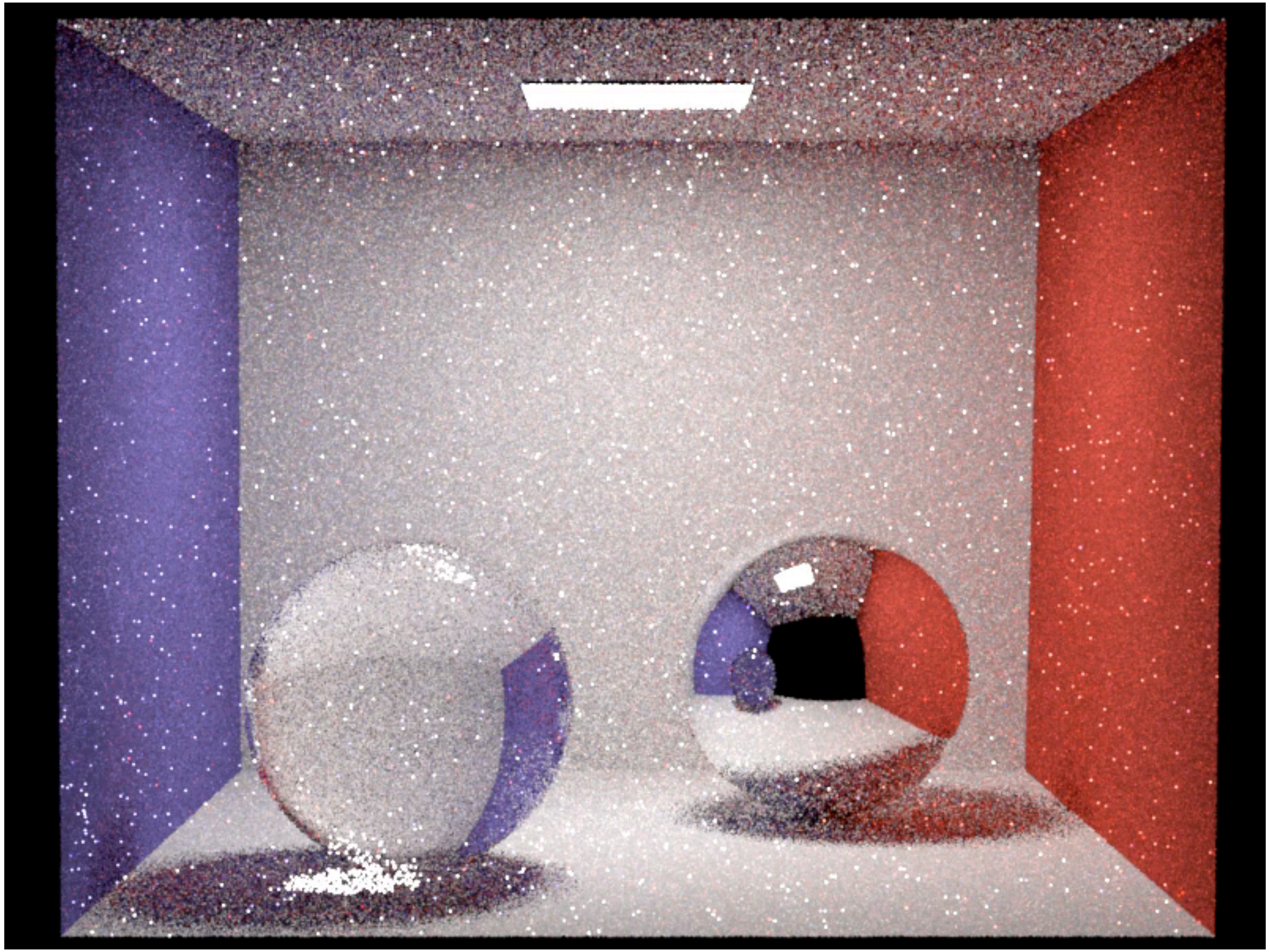




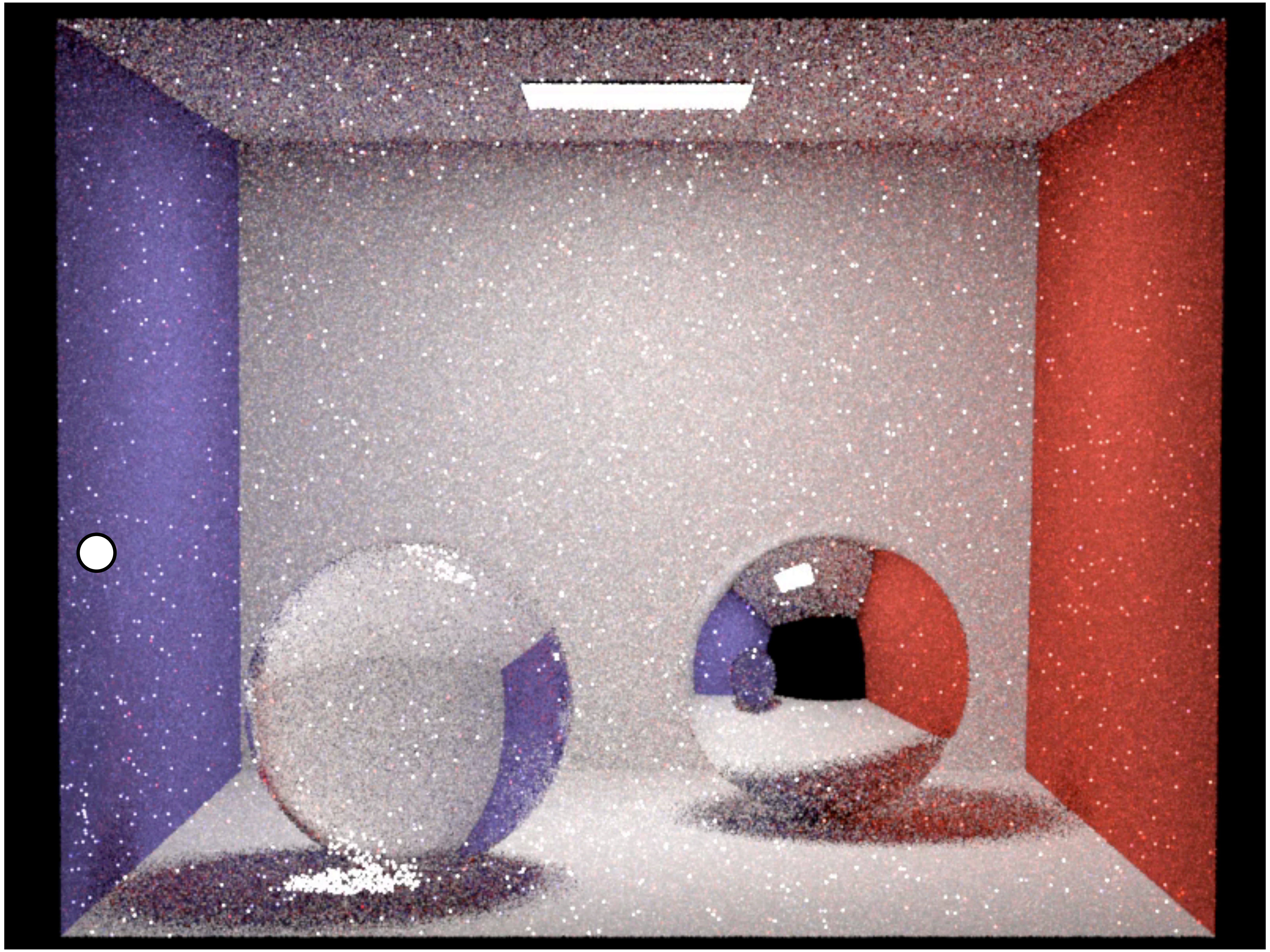
But not all is well..



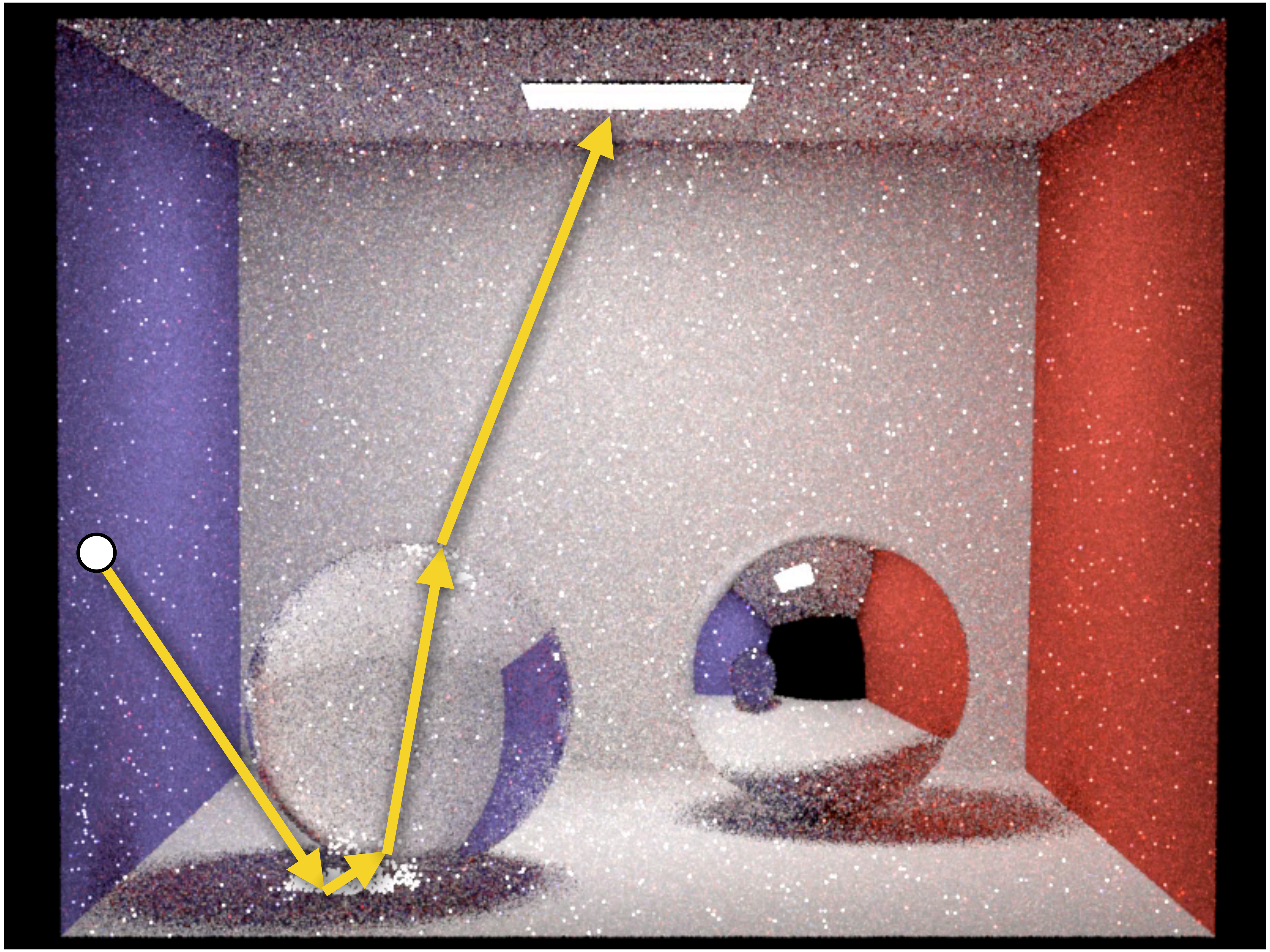
But not all is well..



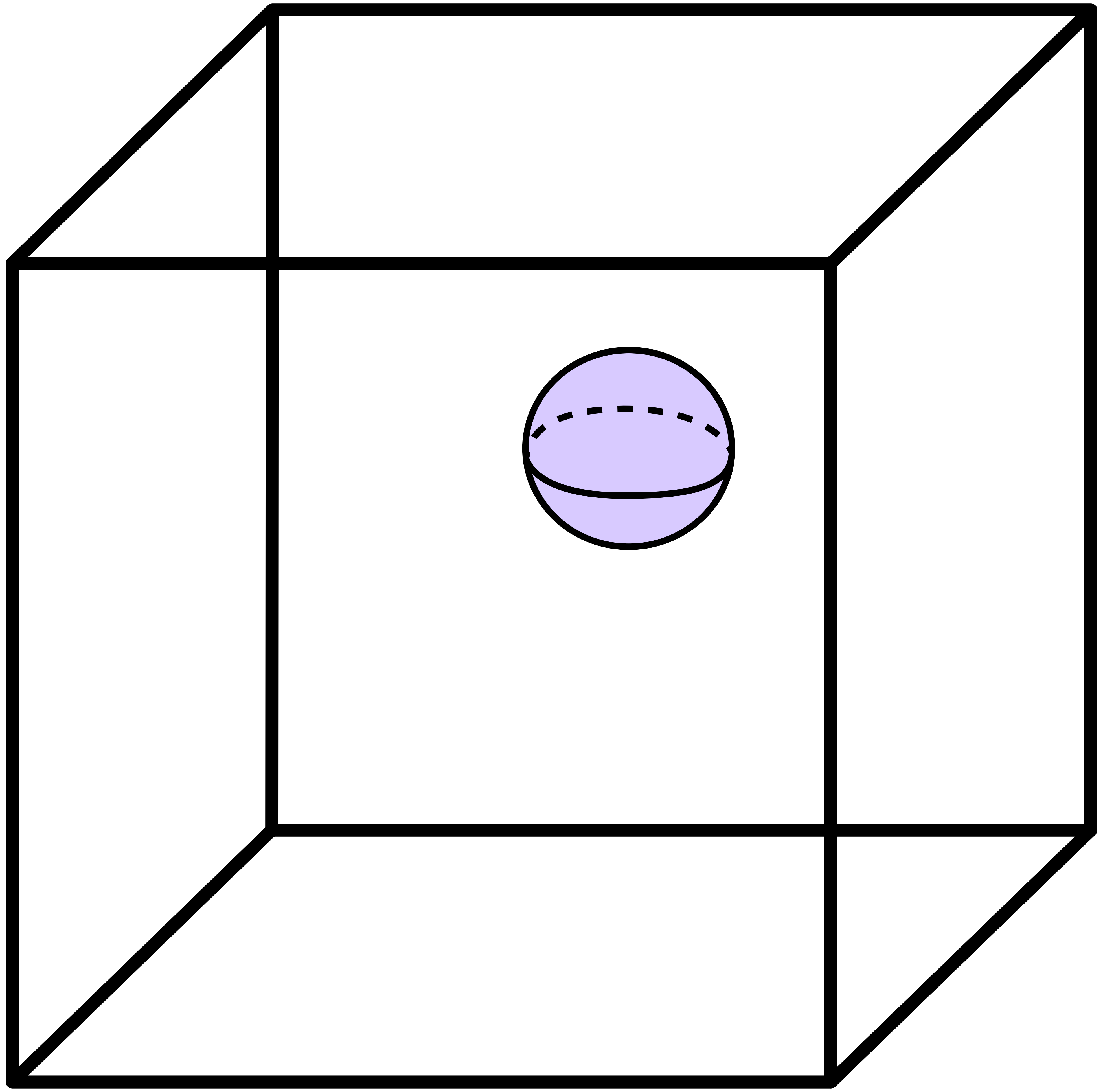
But not all is well..

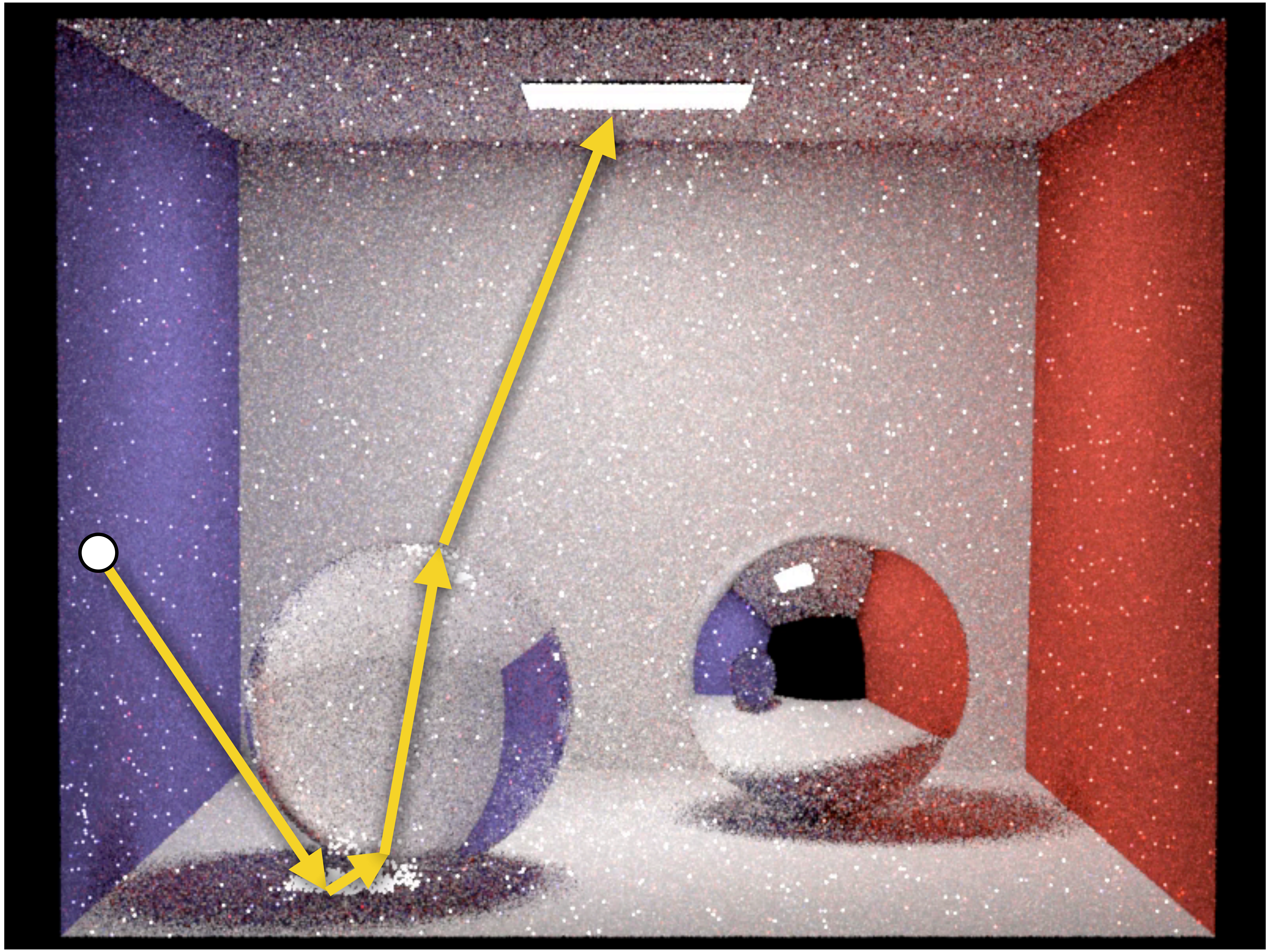


But not all is well..

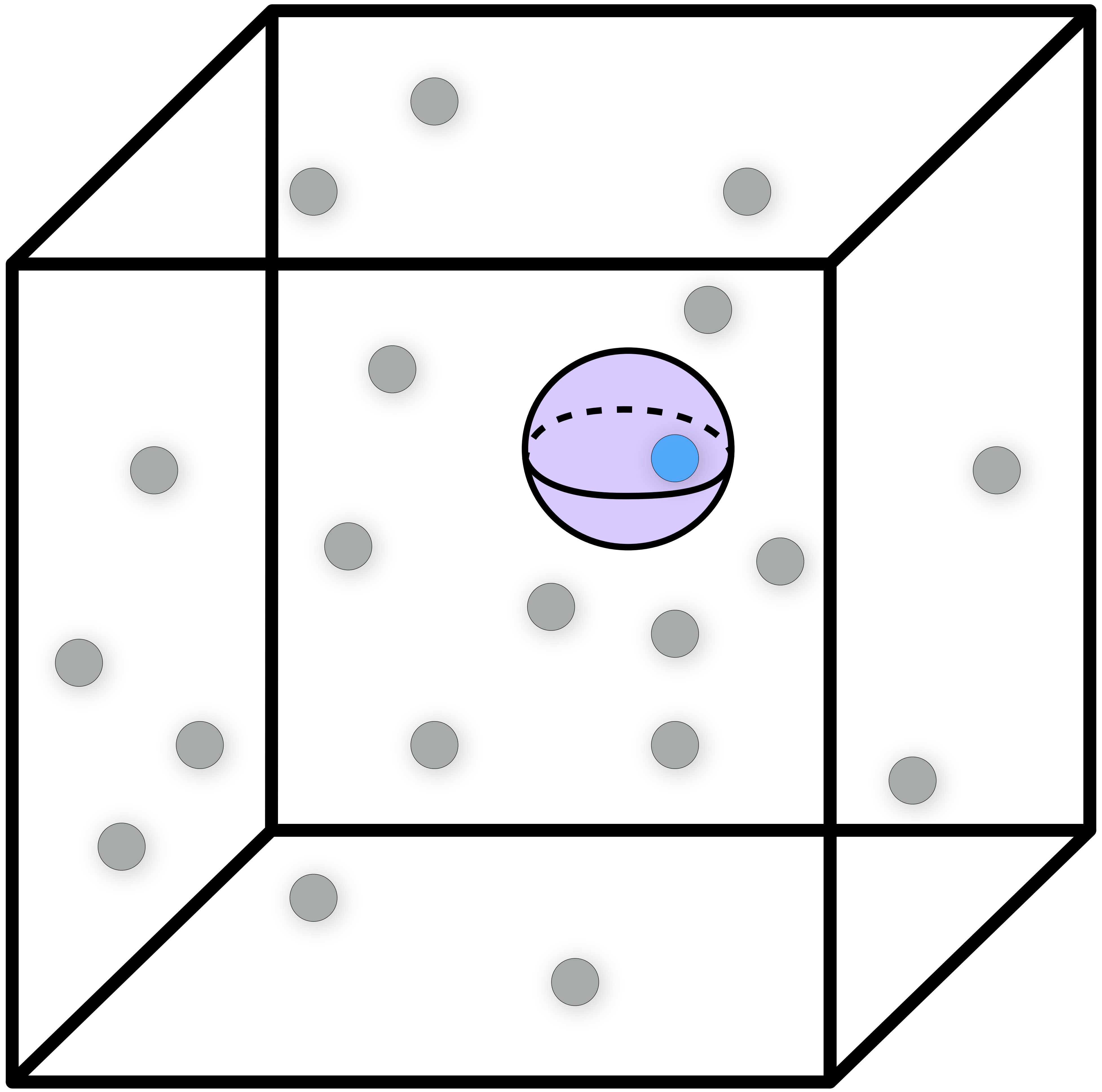


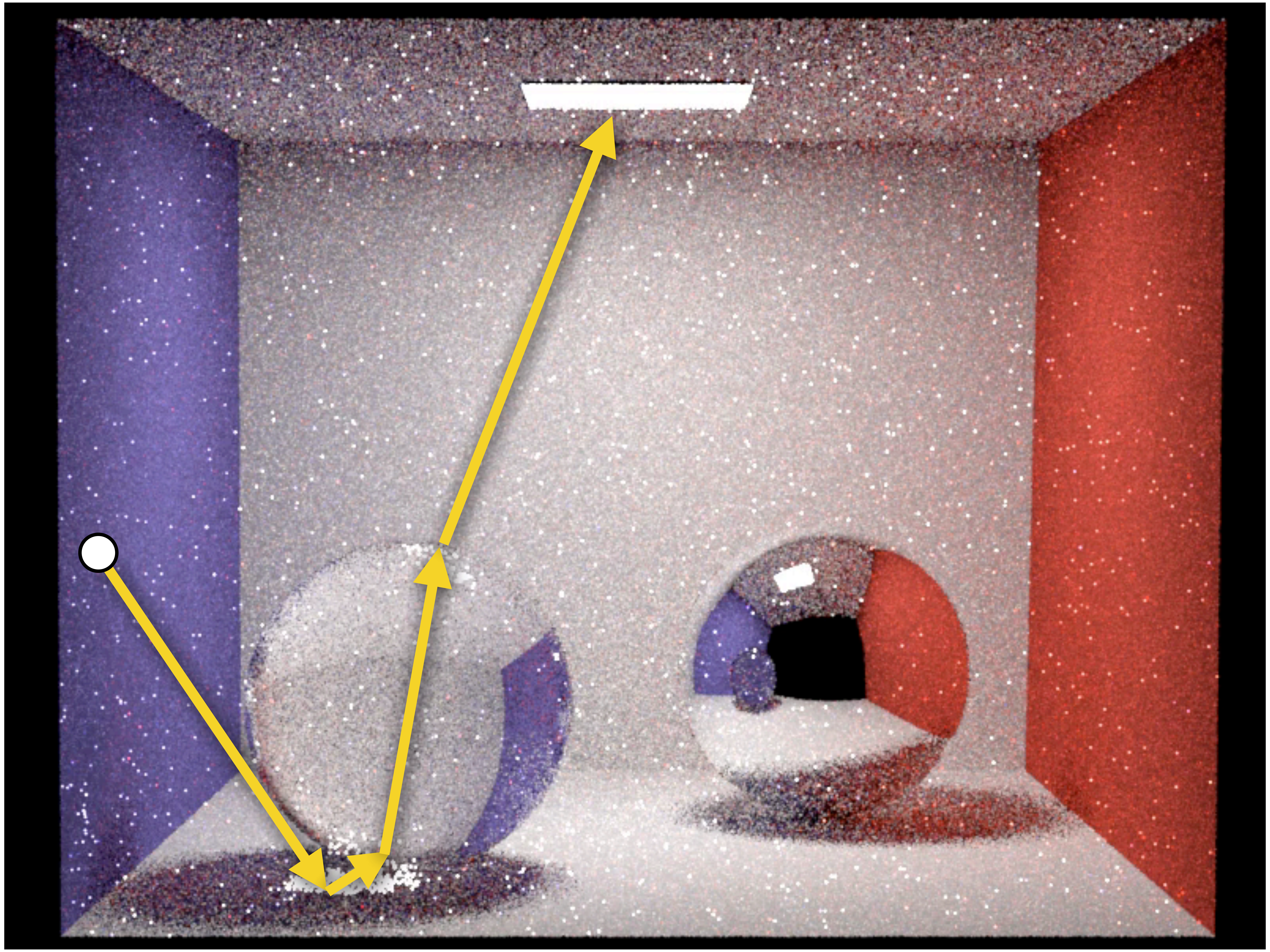
But not all is well..





But not all is well..





A more challenging case

A more challenging case

A more challenging case

A more challenging case

Paper tree

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

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

Figure 5: All of the light in the visible portion of this scene comes through a door that is slightly ajar, such that about 0.1% of the light in the adjacent room comes through the doorway. The light source is a diffuse ceiling panel at the far end of a large adjacent room, so that virtually all of the light coming through the doorway has bounced several times. The MLT algorithm efficiently generates paths that go through the small opening between the rooms, by always preserving a path segment that goes through the doorway. The images are 900 by 500 pixels, and include the effects of all paths up to length 10.

deleted subpath can be regenerated. (This is the factor $p_{d,2}$ mentioned in Section 5.3.1.)

Let $\bar{x}$ be the current path, and consider a mutation that deletes the subpath $\mathbf{x}_s \ldots \mathbf{x}_t$. The insight is that given only the deleted subpath, it is already possible to compute some of the factors in the acceptance probability $a(\bar{y}\,|\,\bar{x})$. In particular, from (13) we see that $a(\bar{y} \mid \bar{x})$ is proportional to $1 \,/\, R(\bar{x}\,|\,\bar{y})$, and that it is possible to compute all the components of $R(\bar{x}\,|\,\bar{y})$ except for the discrete probabilities p_d and p_a (these apply to $\bar{y}$, which has not been generated yet). The computable factors of $1 \,/\, R(\bar{x}\,|\,\bar{y})$ are denoted $p_{d,2}$. In the example of Section 5.3.1, for instance, we have

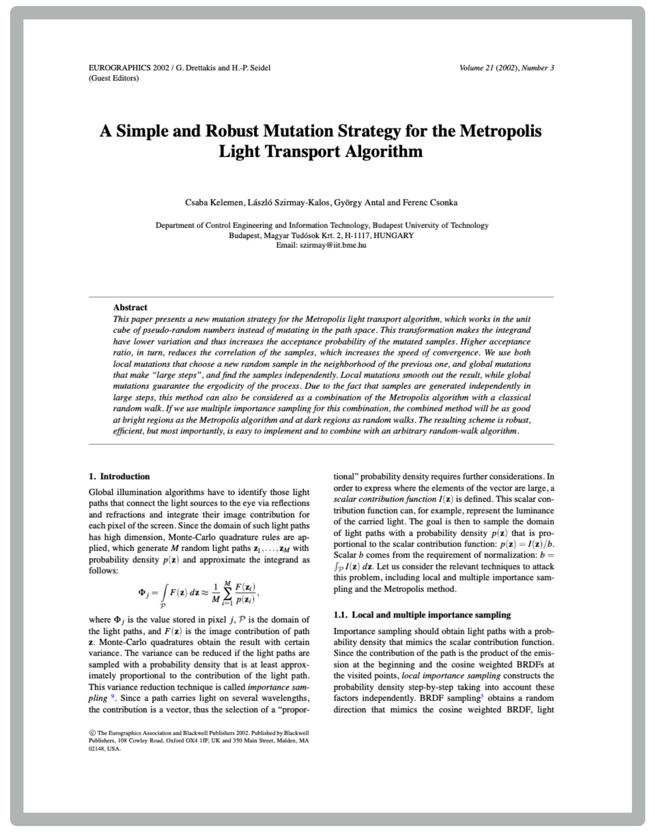
$$p_{d,2}[1,2] = 1/C(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \,.$$

The discrete probabilities for each mutation type are weighted by this factor, before a mutation is selected.

Metropolis Light Transport
[Veach & Guibas 1997]

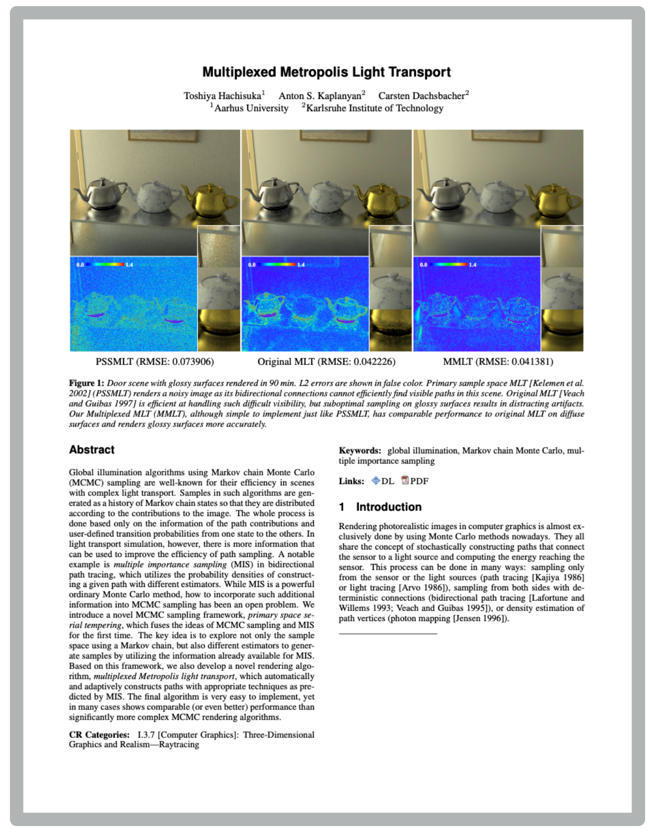
Paper tree

Metropolis Light Transport
[Veach & Guibas 1997]



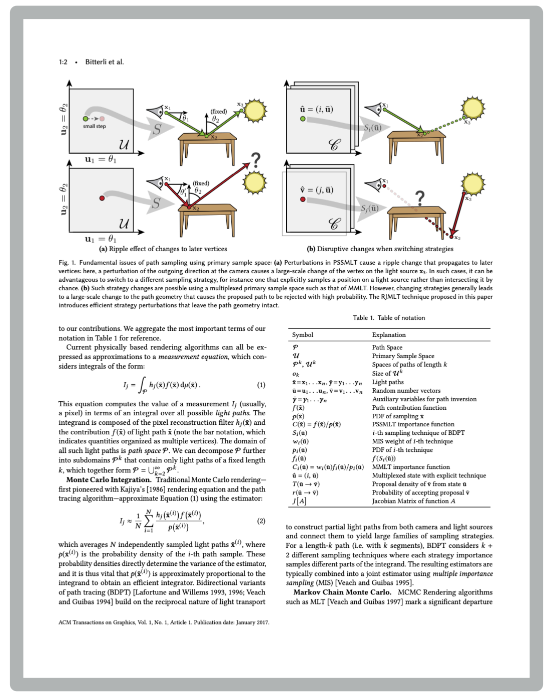
A Simple and Robust Mutation Strategy for the Metropolis Light Transport Algorithm

Primary Sample Space MLT
[Kelemen et al. 2002]



Multiplexed Metropolis Light Transport

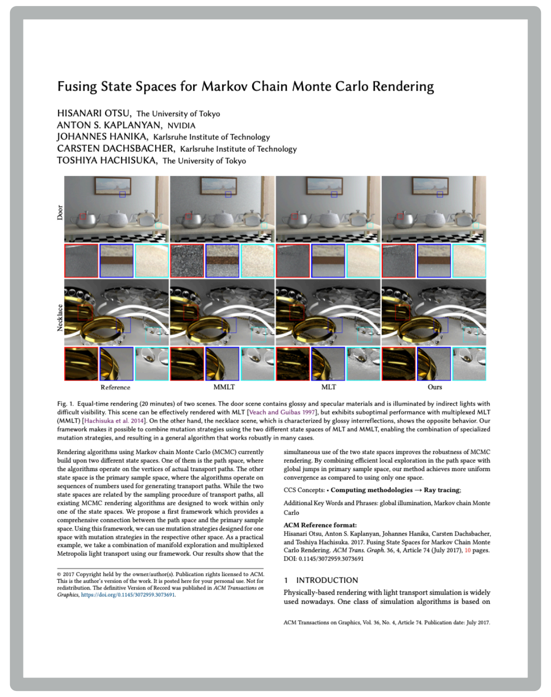
Multiplexed MLT
[Hachisuka et al. 2014]



Reversible Jump MLT
[Bitterli et al. 2017]

Charted Metropolis Light Transport

Chartered MLT
[Pantaleoni et al. 2017]

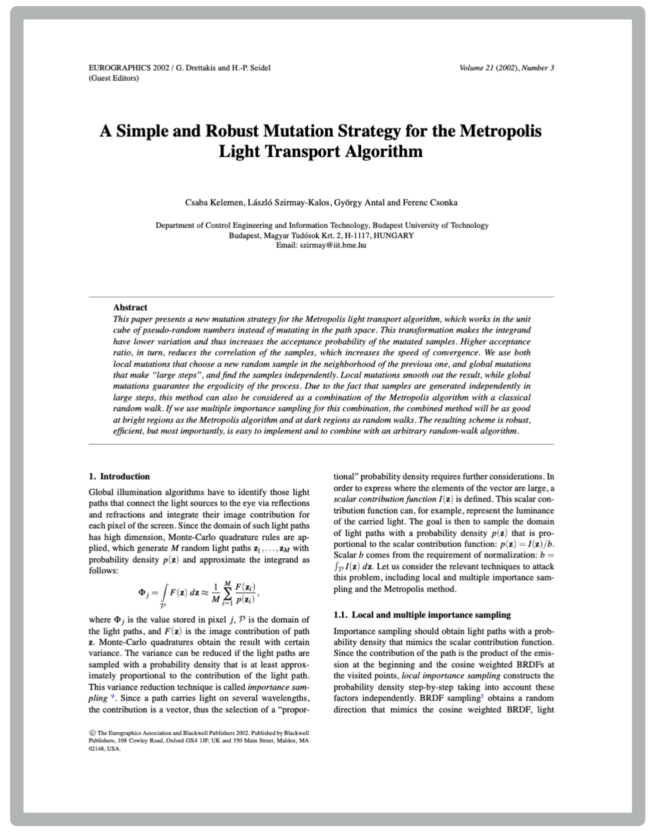


Fusing State Spaces for Markov Chain Monte Carlo Rendering

Fusing State Spaces
[Otsu et al. 2017]

Paper tree

Metropolis Light Transport
[Veach & Guibas 1997]



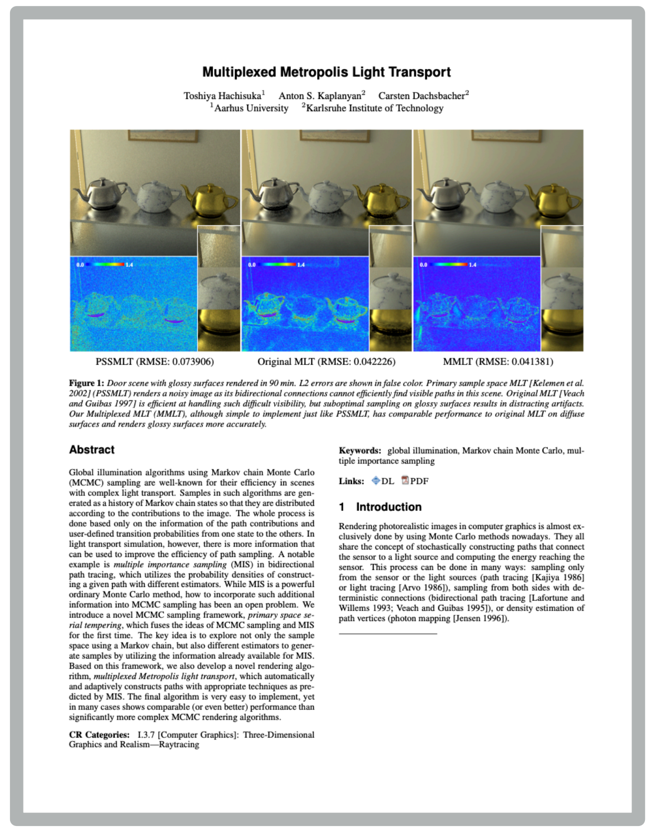
EUROGRAPHICS 2002 / G. Drettakis and H.-P. Seidel (Guest Editors)

Volume 21 (2002), Number 3

A Simple and Robust Mutation Strategy for the Metropolis Light Transport Algorithm

Csaba Kelemen, László Szirmay-Kalos, György Antal and Ferenc Csonka

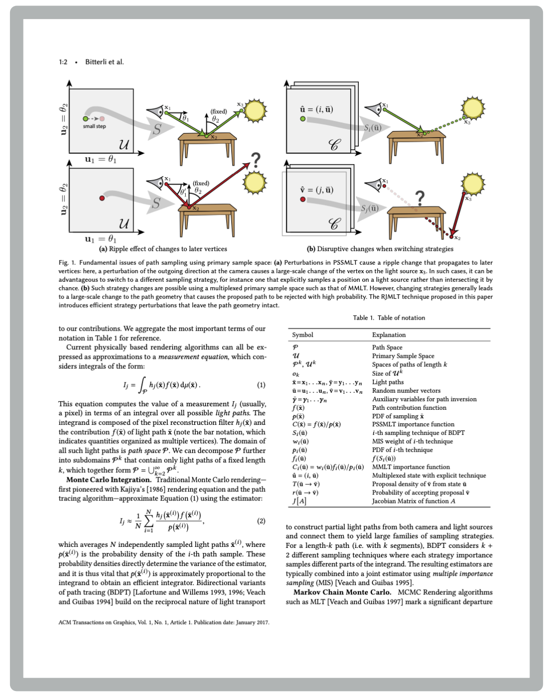
Primary Sample Space MLT
[Kelemen et al. 2002]



Multiplexed Metropolis Light Transport

Toshiya Hachisuka, Anton S. Kaplanyan, Carsten Dachsbacher

Multiplexed MLT
[Hachisuka et al. 2014]



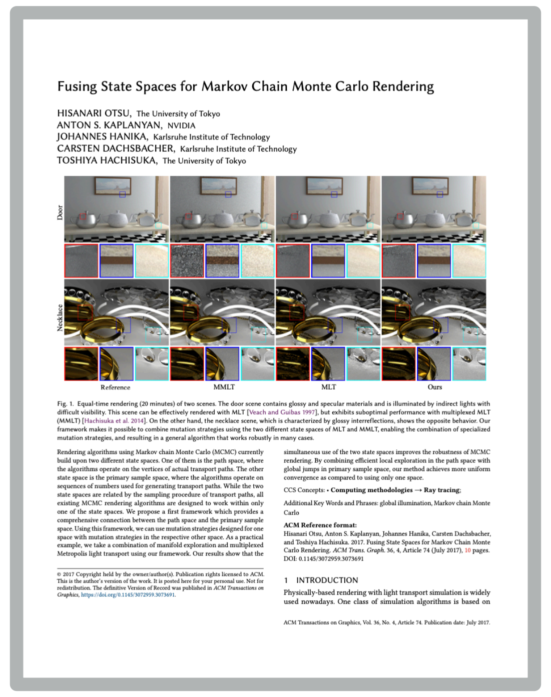
Reversible Jump MLT
[Bitterli et al. 2017]

Charted Metropolis Light Transport

Jacopo Pantaleoni

NVIDIA

Chartered MLT
[Pantaleoni et al. 2017]



Fusing State Spaces for Markov Chain Monte Carlo Rendering

Fusing State Spaces
[Otsu et al. 2017]

To appear in ACM Transactions on Graphics 31(4).

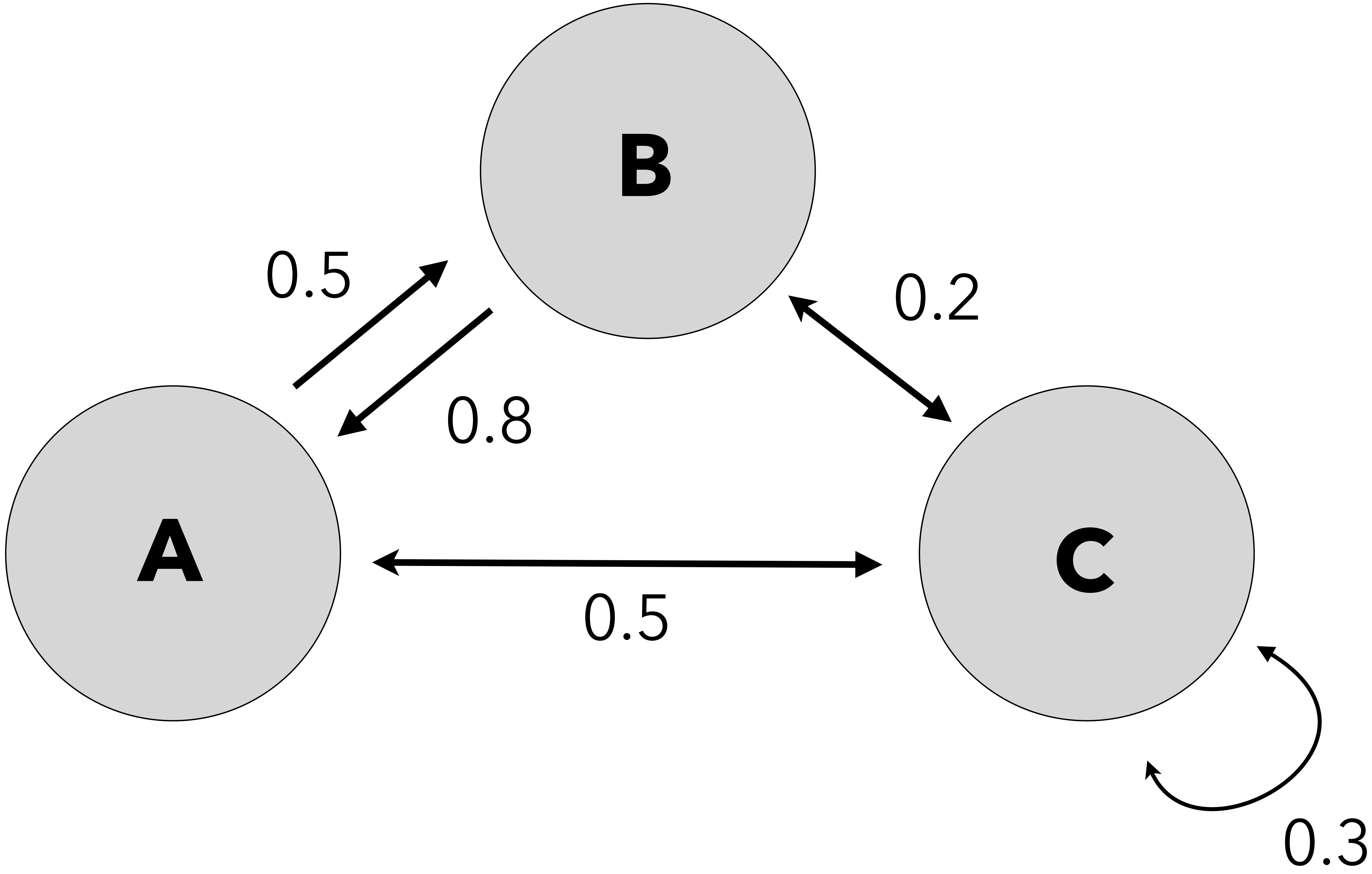
Manifold Exploration: A Markov Chain Monte Carlo Technique for Rendering Scenes with Difficult Specular Transport

Wenzel Jakob, Steve Marschner

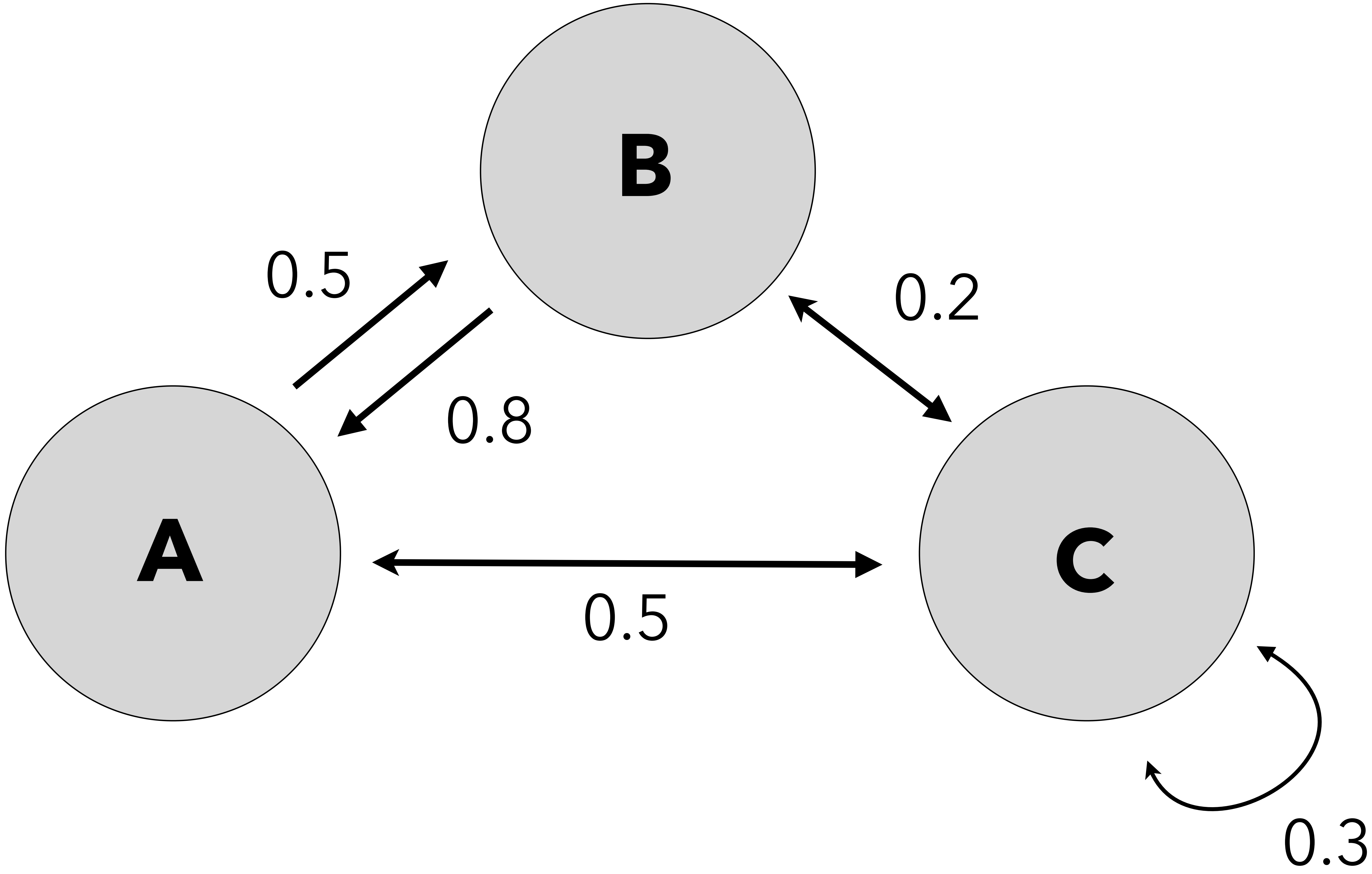
Cornell University

Manifold Exploration
[Jakob & Marschner 2012]

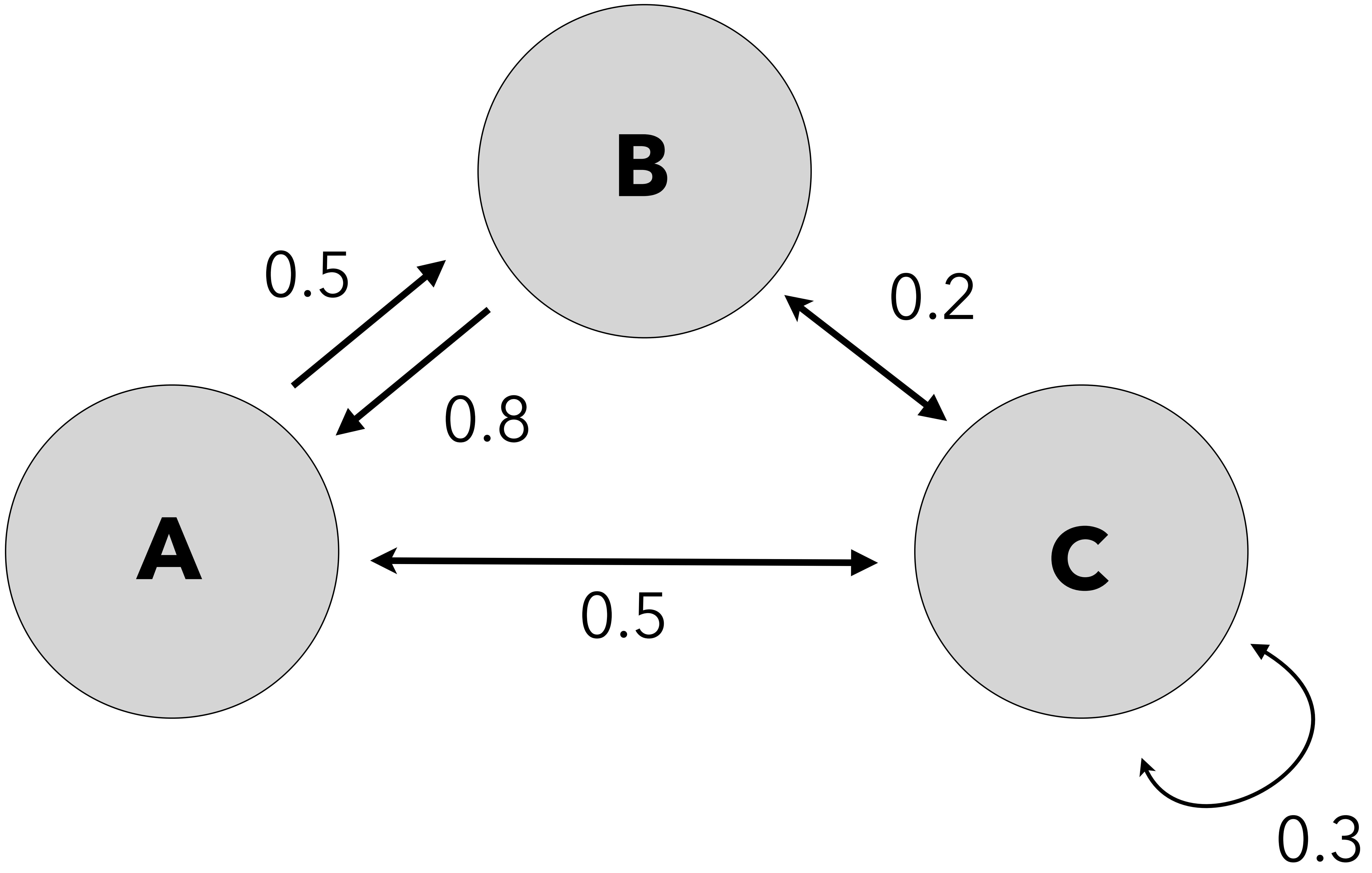
Markov Chain review (*discrete case*)



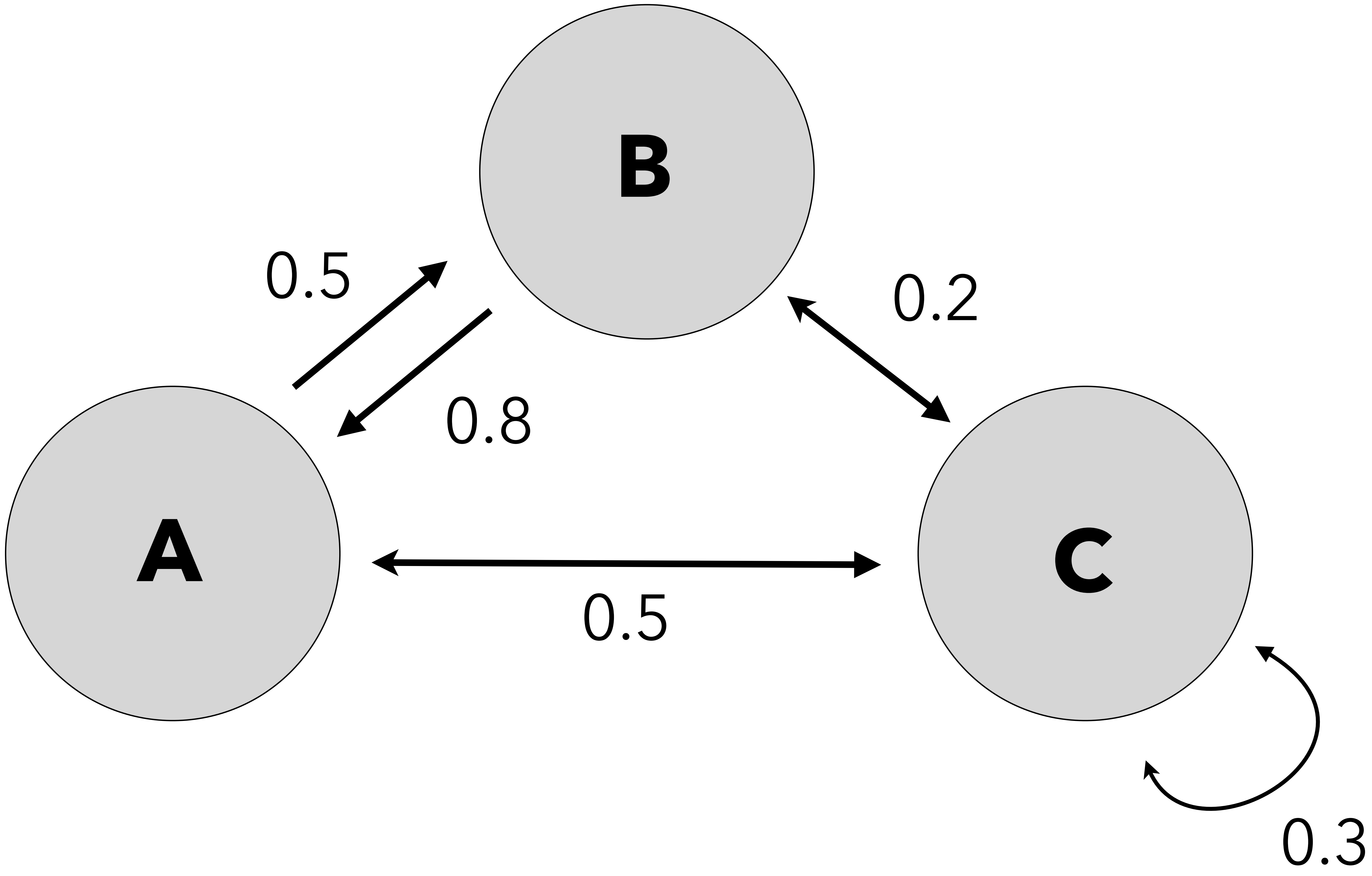
Markov Chain review (*discrete case*)

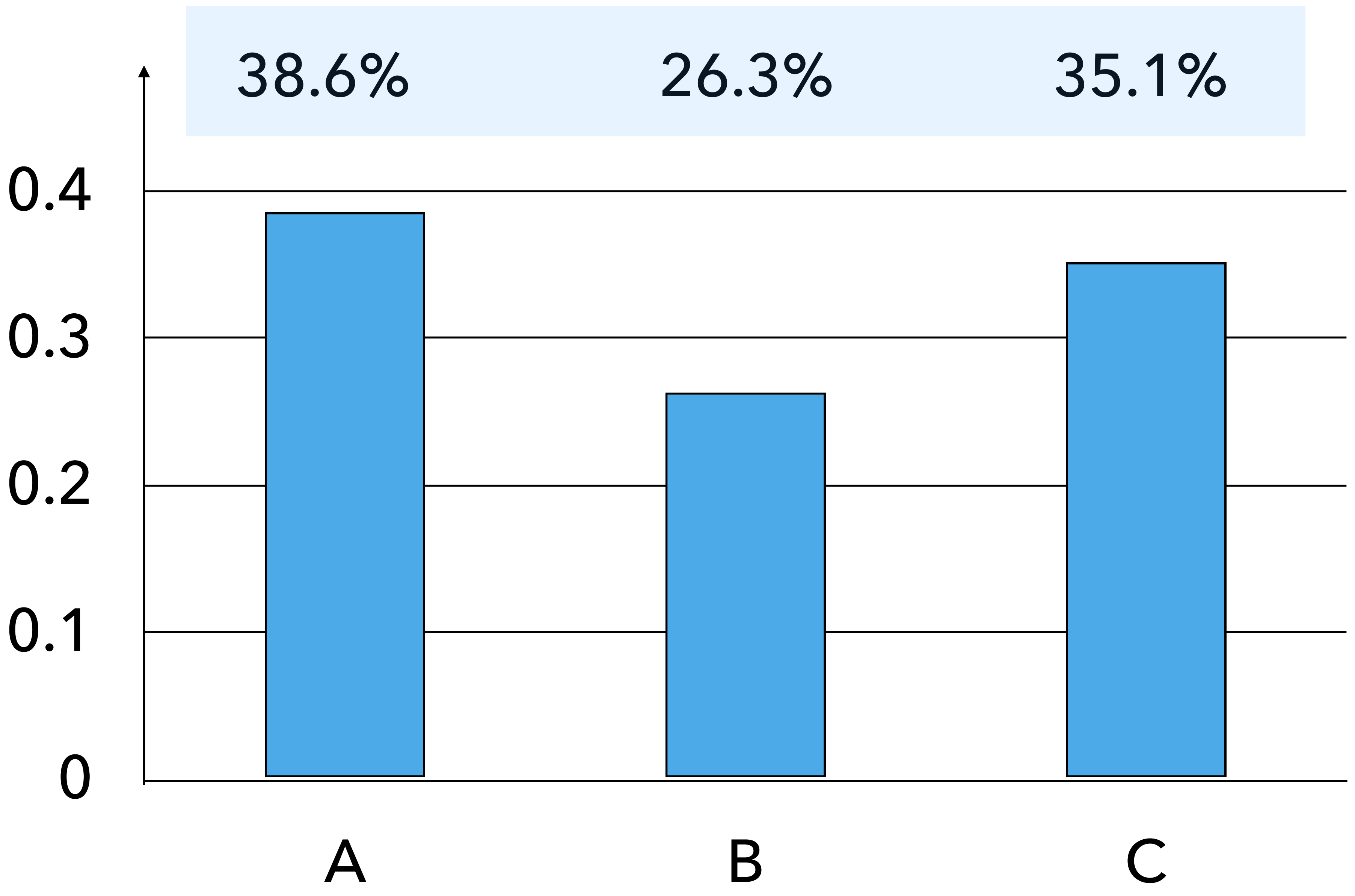


Markov Chain review (*discrete case*)

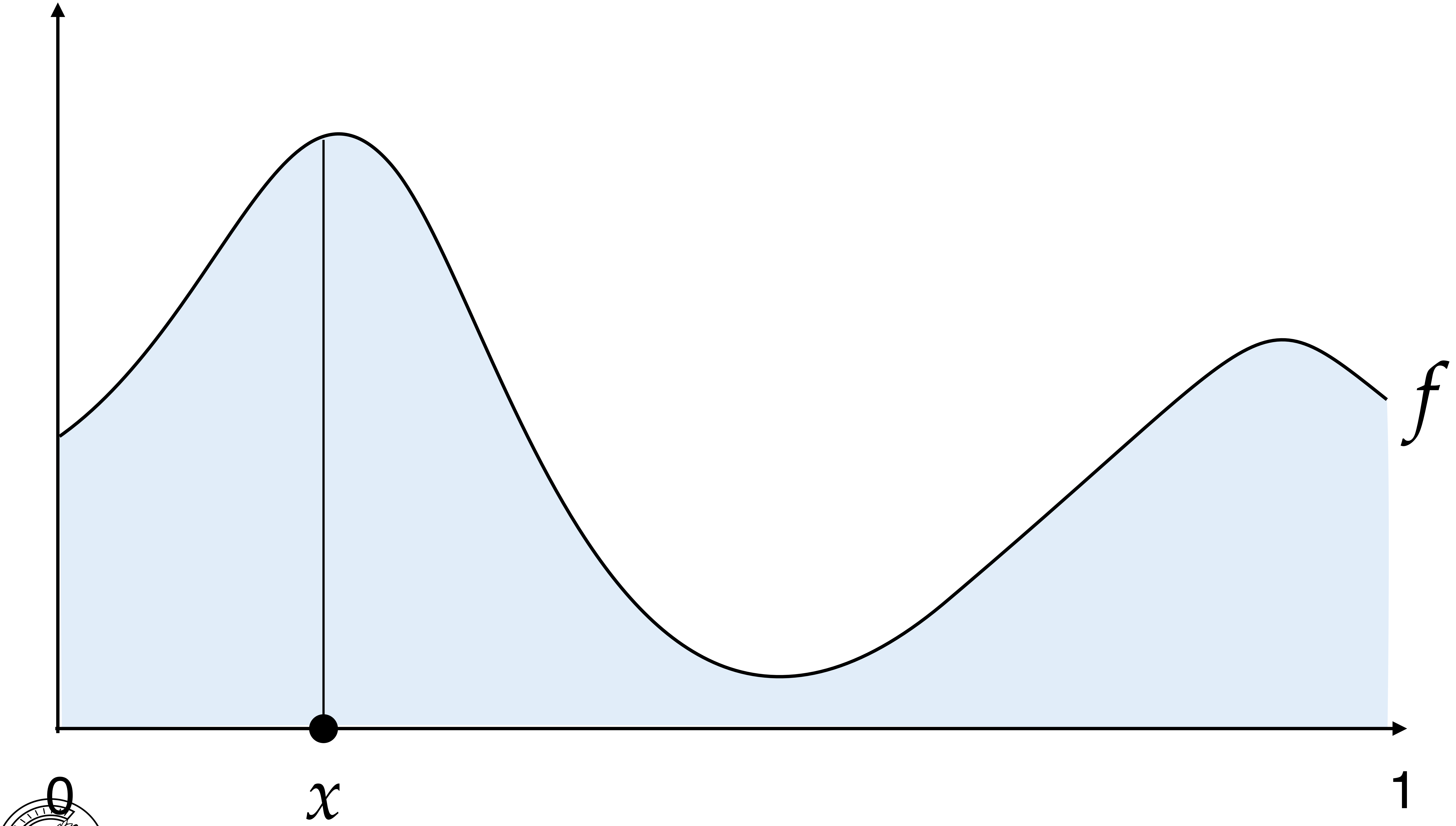


Markov Chain review (*discrete case*)

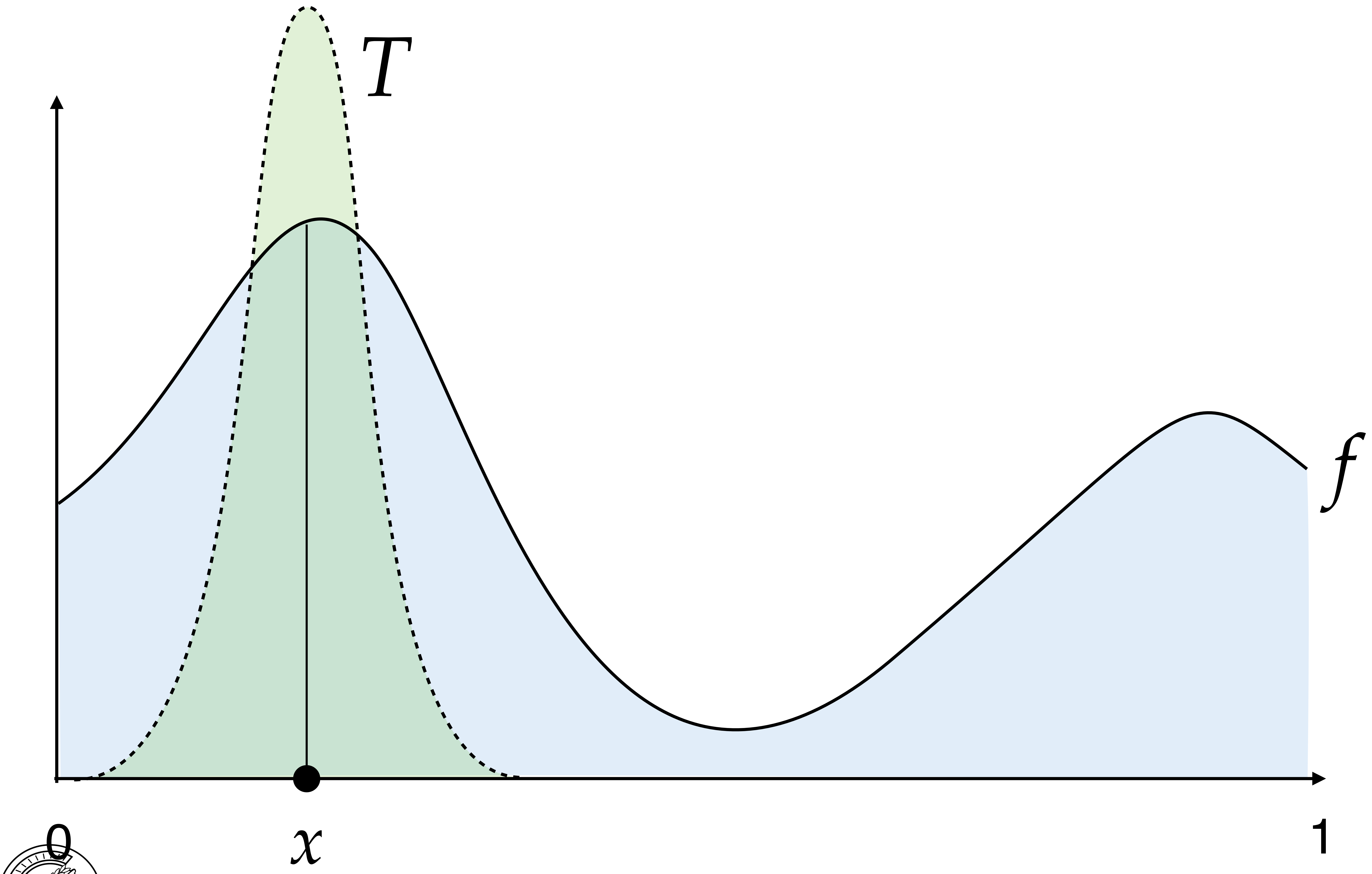




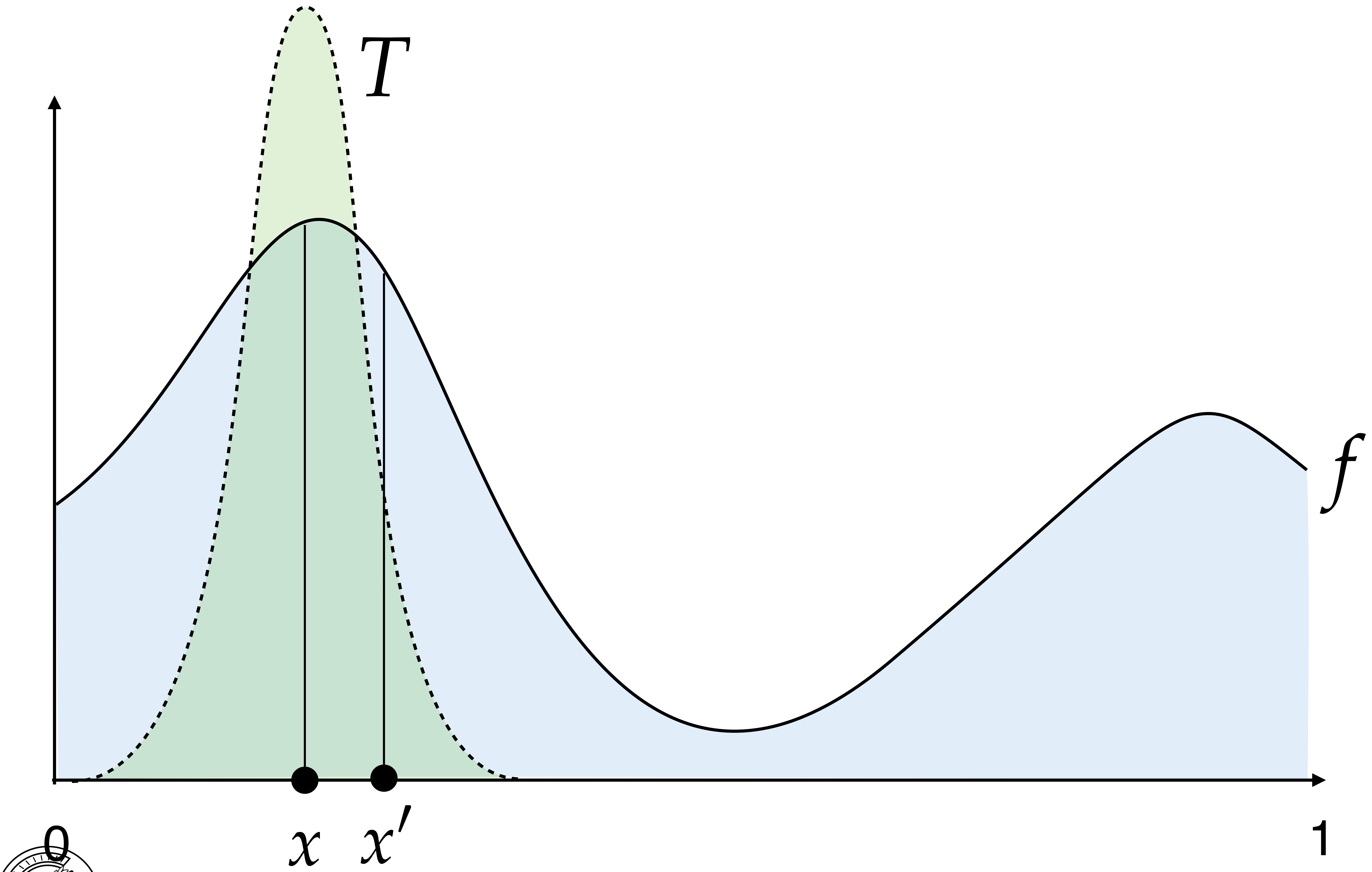
Metropolis-Hastings algorithm (*continuous case*)



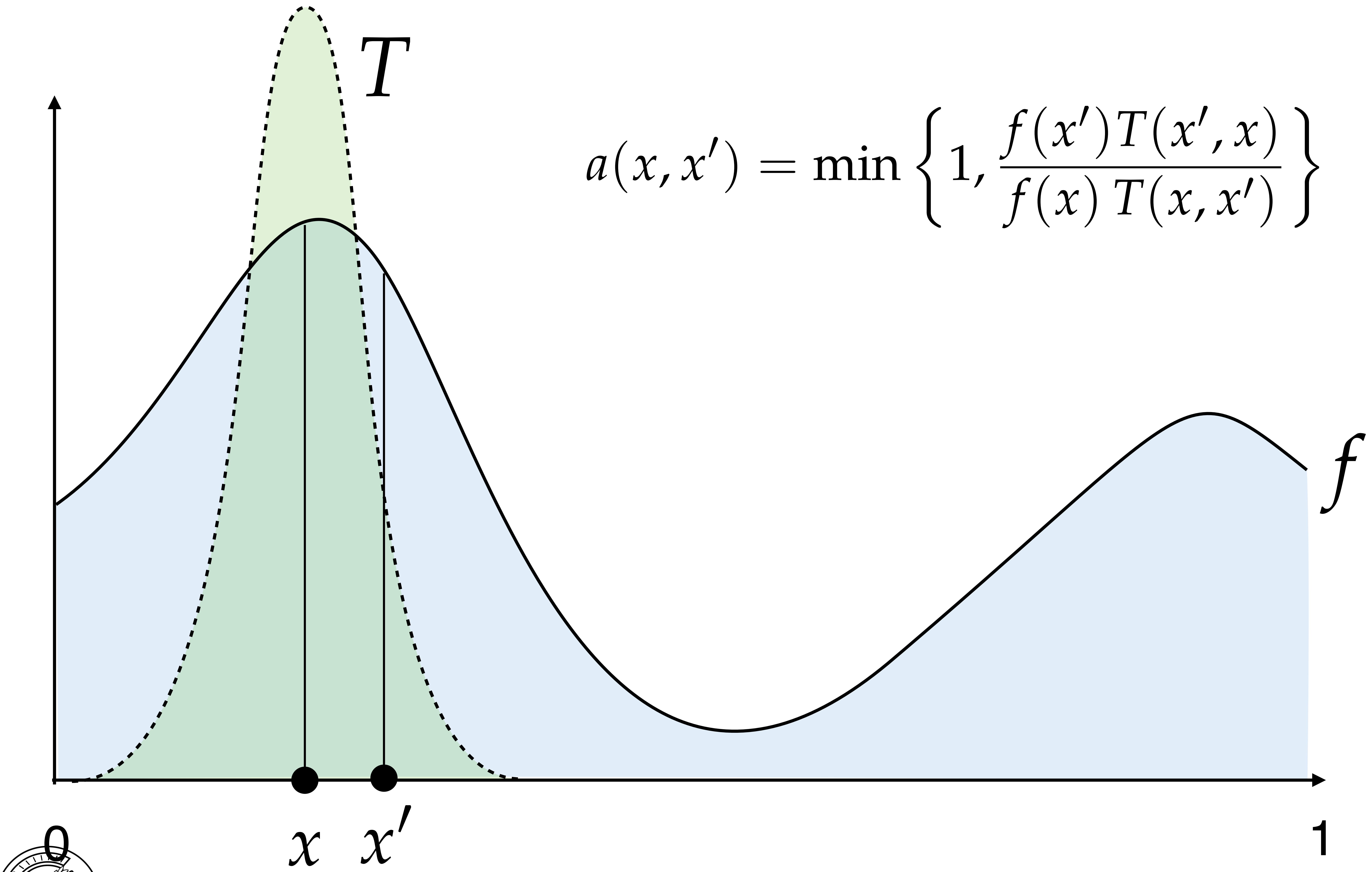
Metropolis-Hastings algorithm (*continuous case*)



Metropolis-Hastings algorithm (*continuous case*)

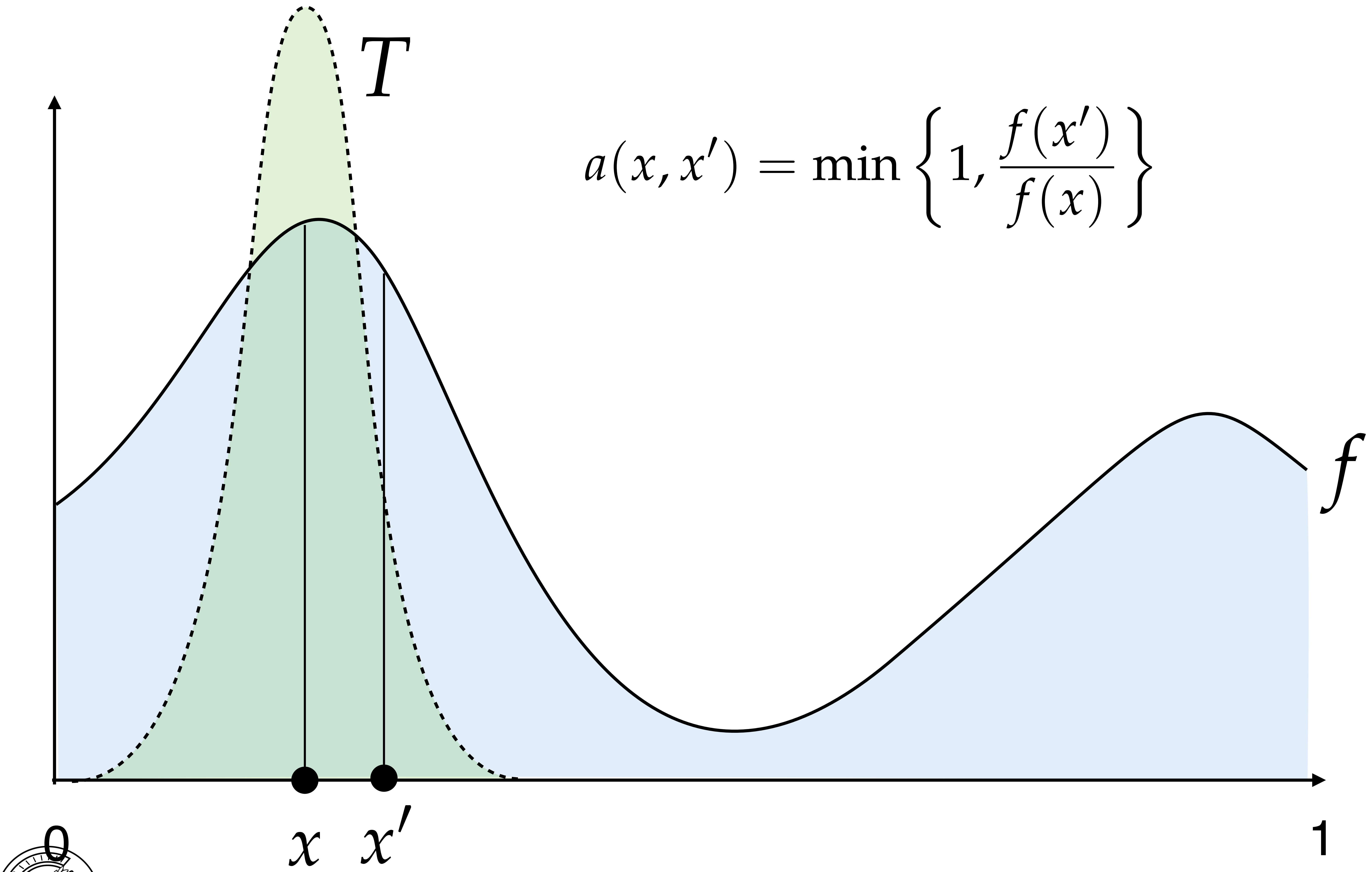


Metropolis-Hastings algorithm (*continuous case*)



$$a(x,x') = \min\left\{1, \frac{f(x')T(x',x)}{f(x)\,T(x,x')}\right\}$$

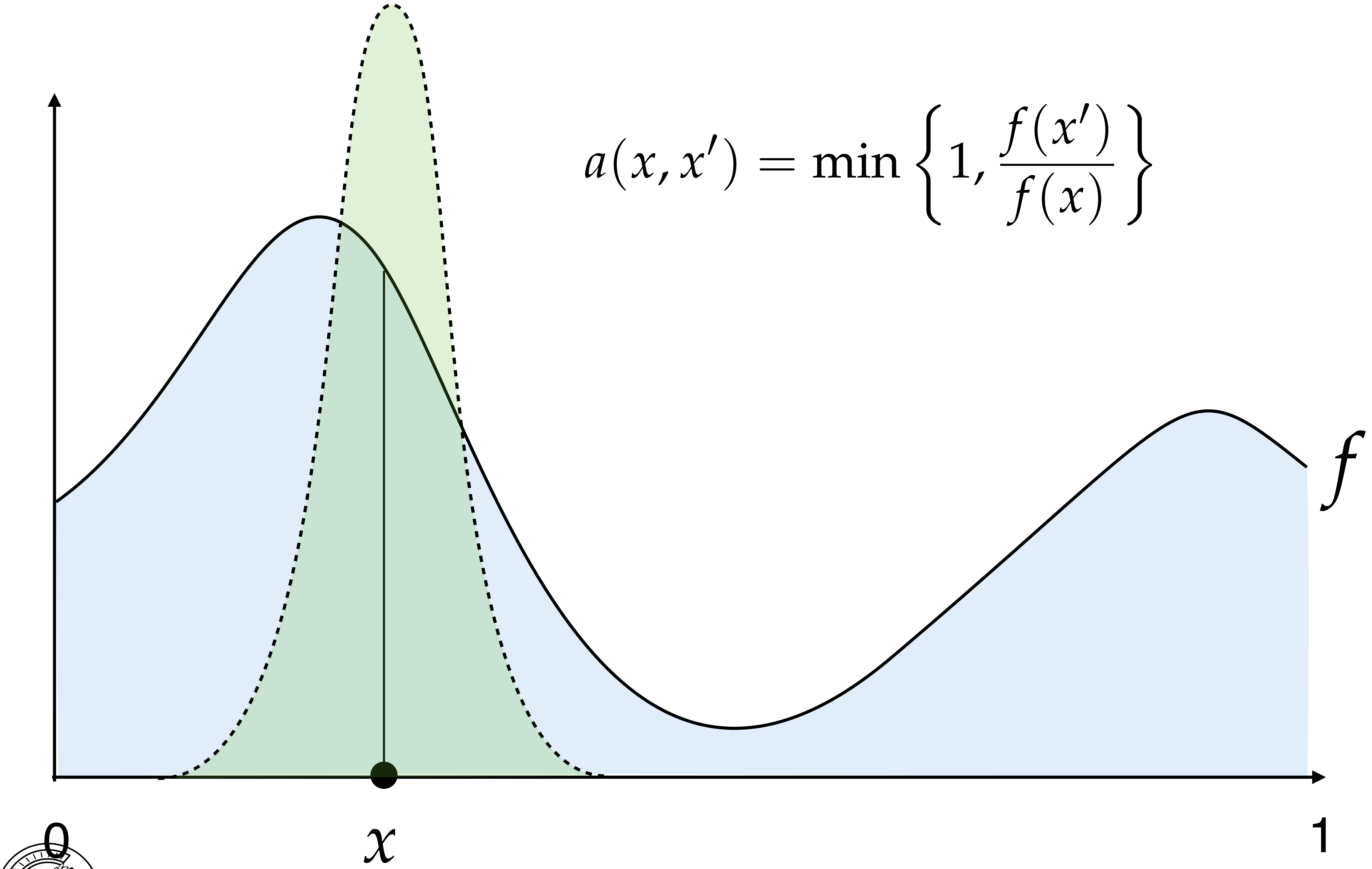
Metropolis-Hastings algorithm (*continuous case*)



$$a(x, x') = \min\left\{1, \frac{f(x')}{f(x)}\right\}$$

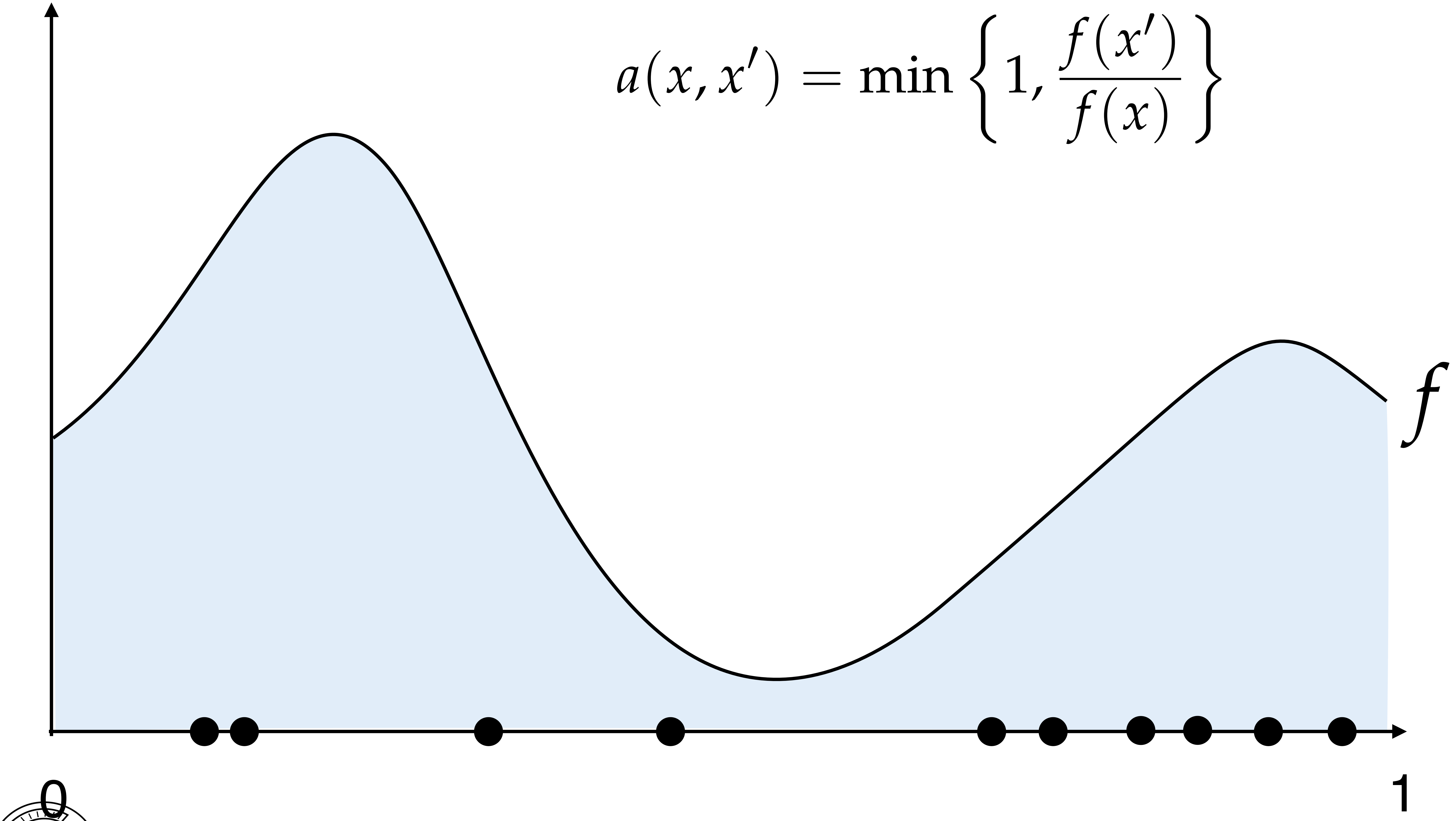
Metropolis-Hastings algorithm (*continuous case*)

$$a(x, x') = \min\left\{1, \frac{f(x')}{f(x)}\right\}$$



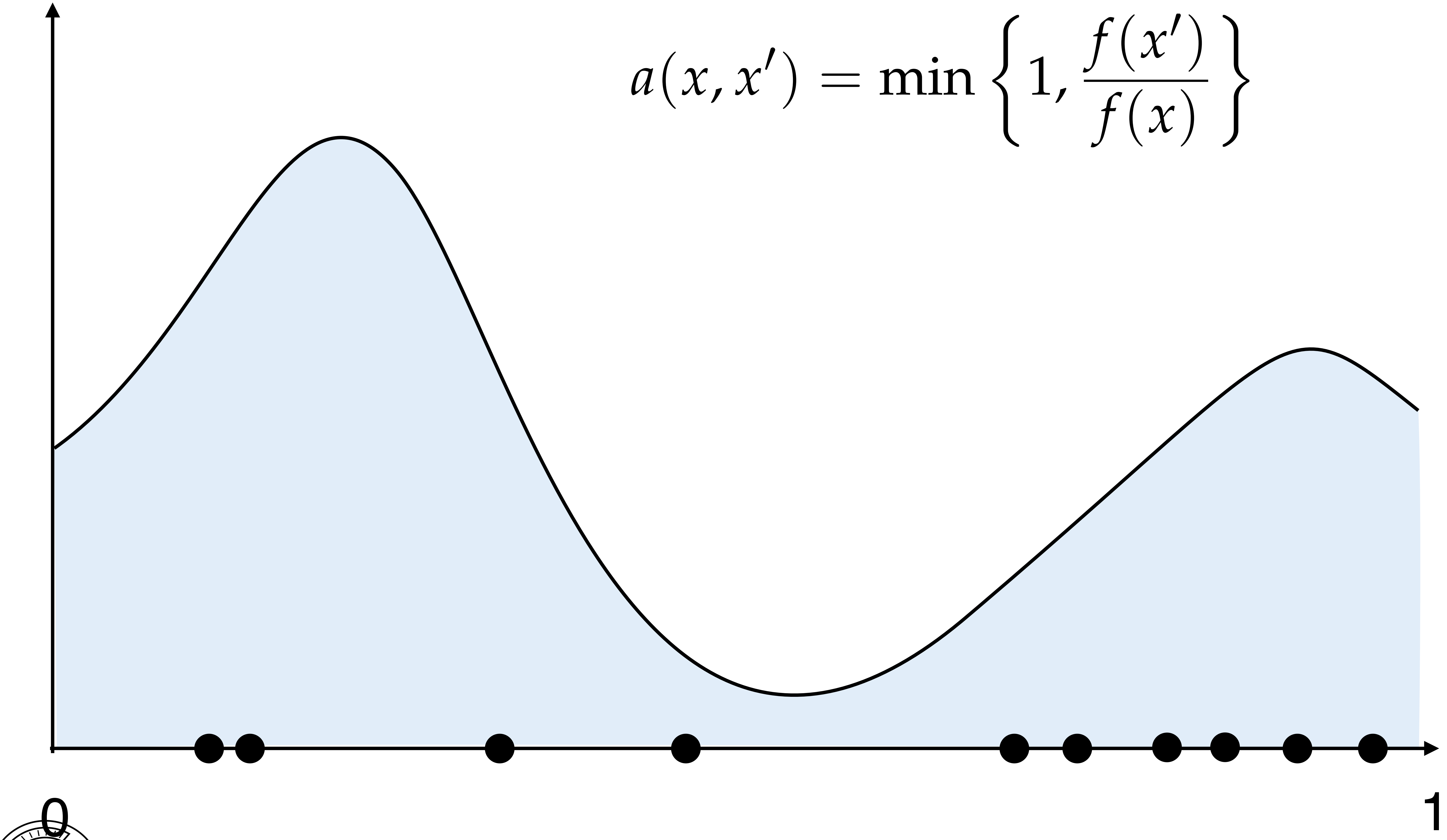
Metropolis-Hastings algorithm (*continuous case*)

$$a(x, x') = \min\left\{1, \frac{f(x')}{f(x)}\right\}$$



Metropolis-Hastings algorithm (*continuous case*)

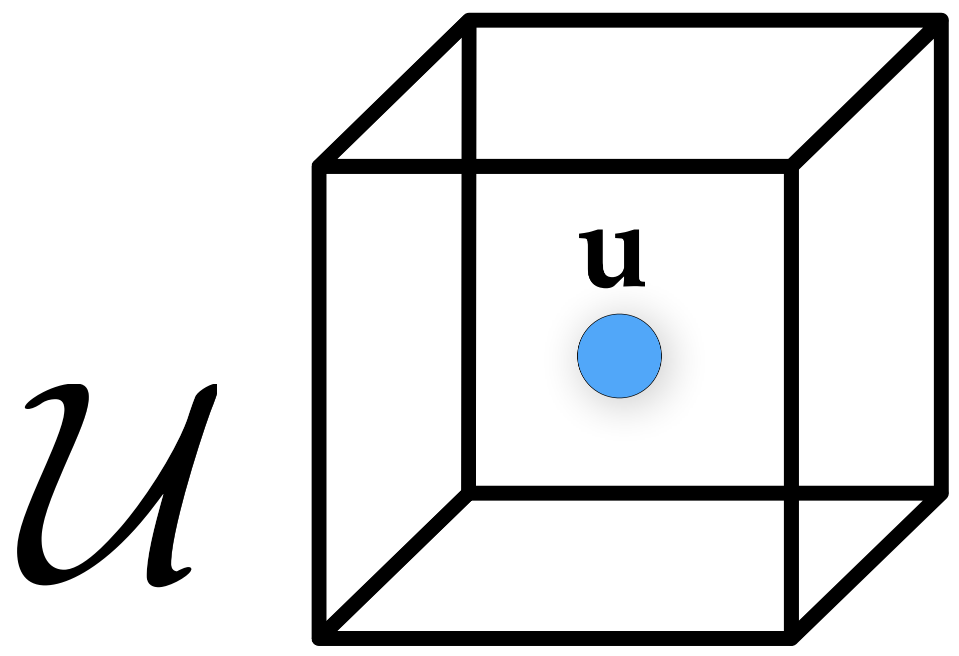
$$a(x, x') = \min\left\{1, \frac{f(x')}{f(x)}\right\}$$



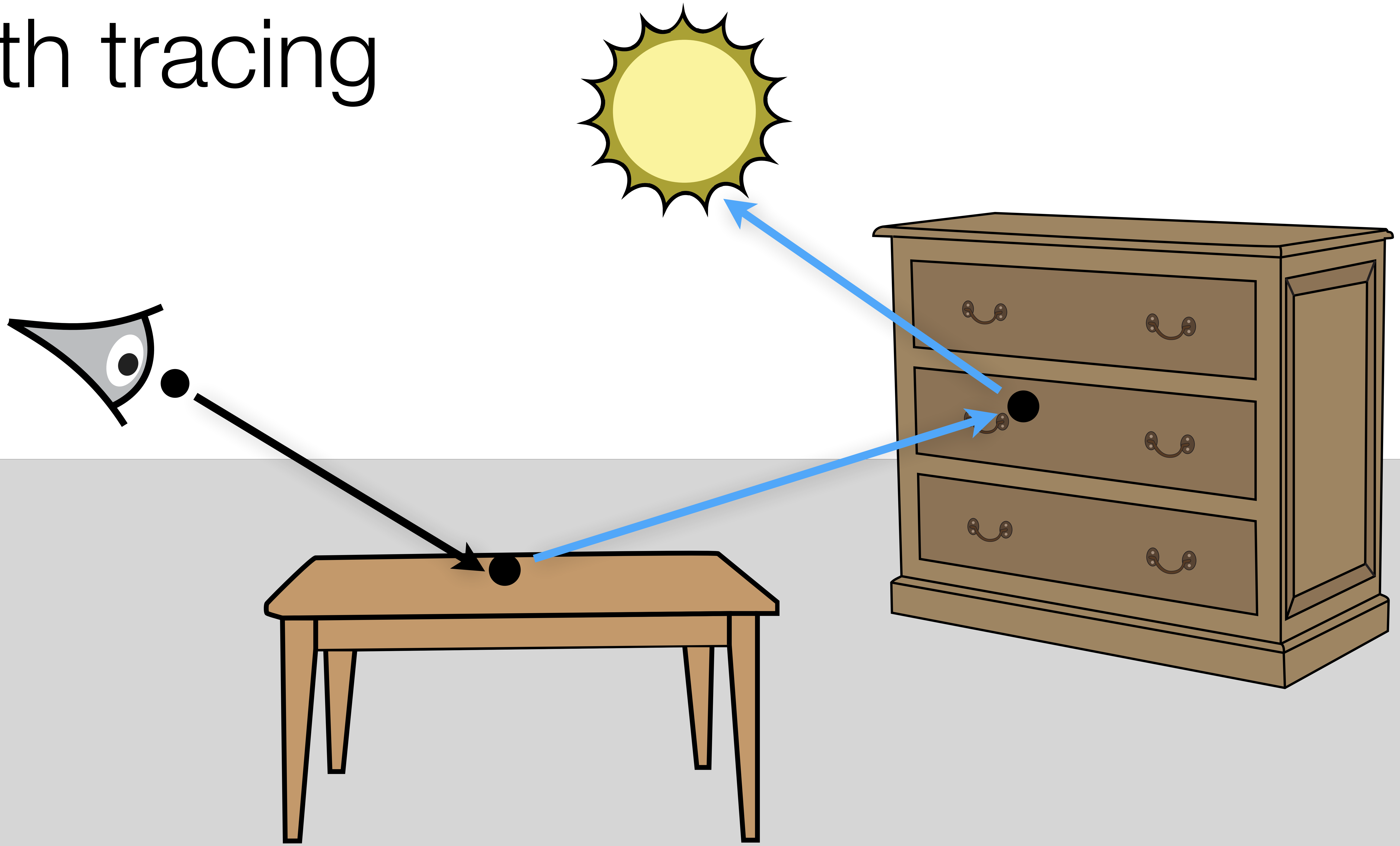
Interesting properties:

1. Samples are correlated
2. Algorithm tends to explore local maxima
3. Can be combined with classical MC algorithms

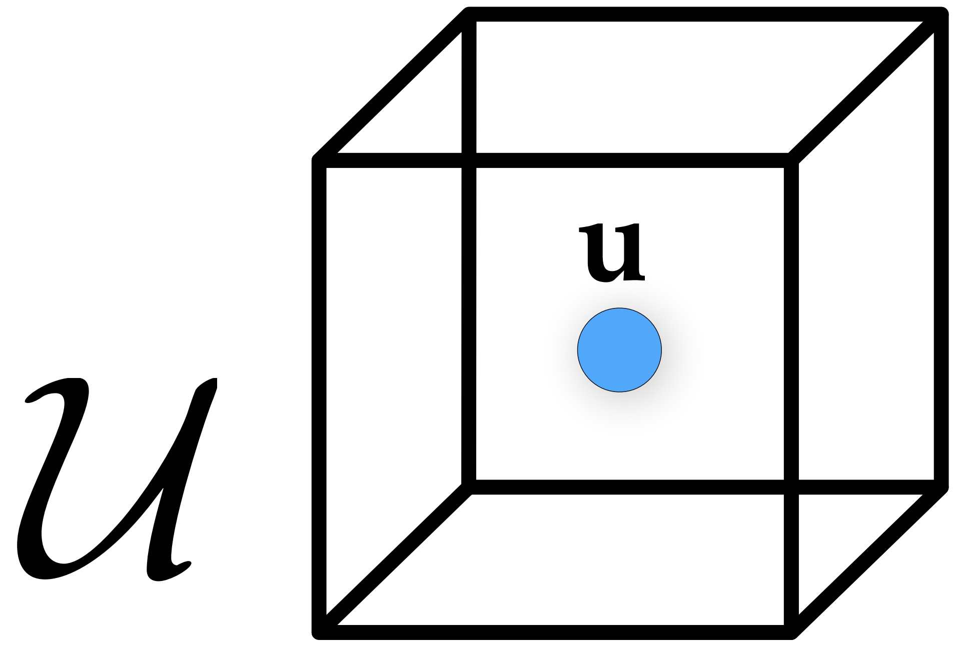
Application to path tracing



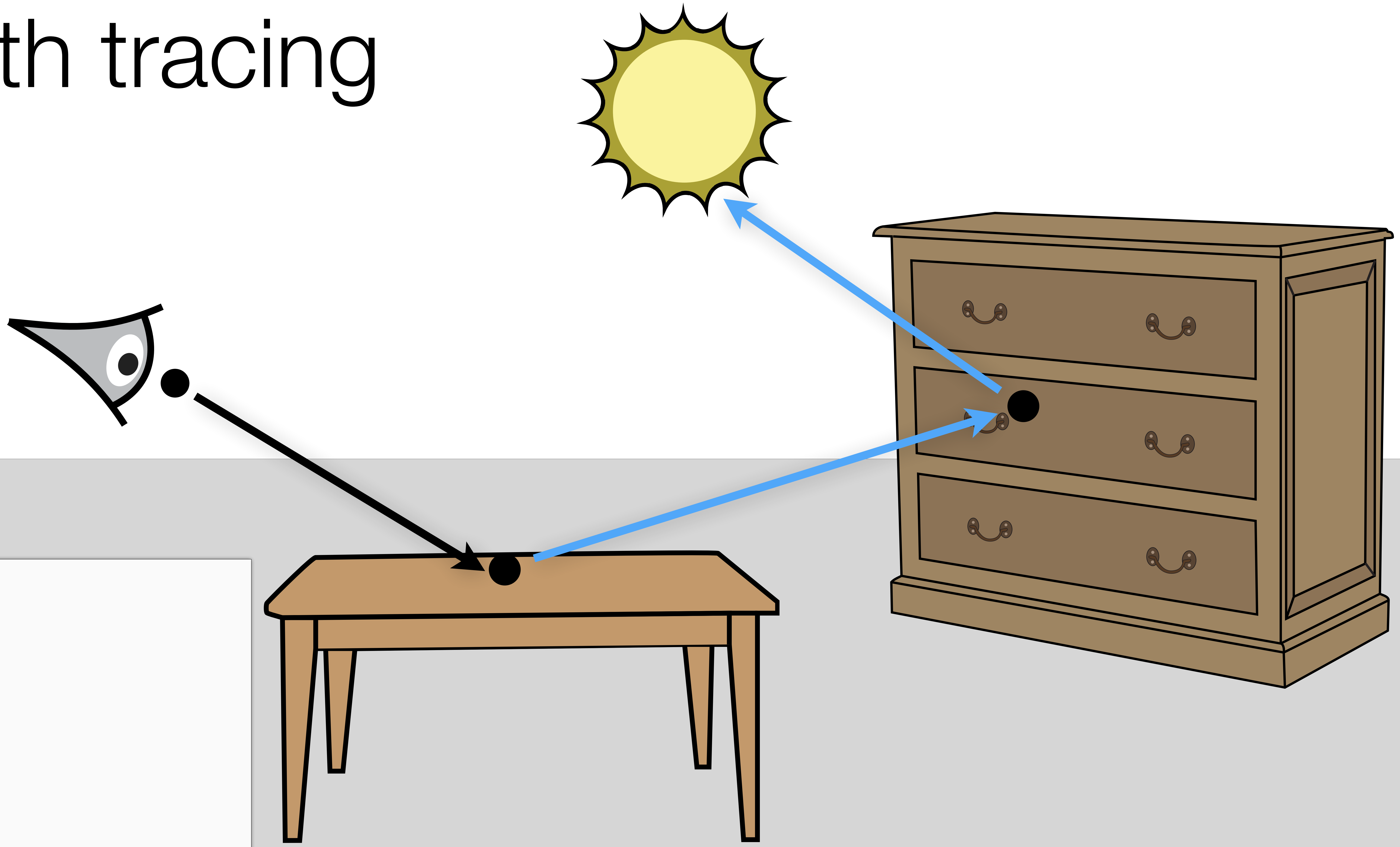
Hypercube of
“random numbers”



Application to path tracing

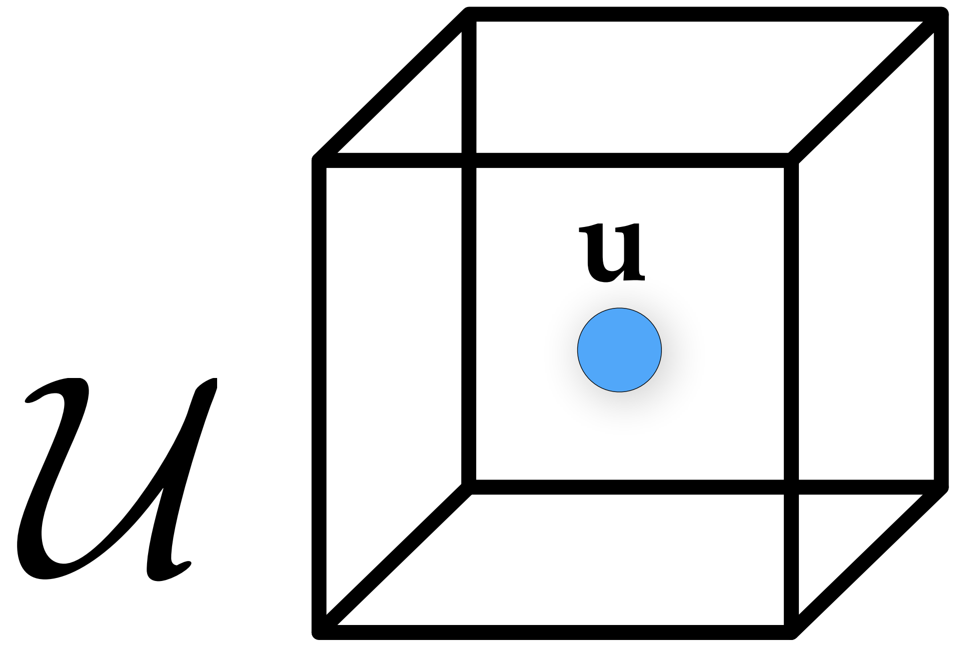


Hypercube of
“random numbers”

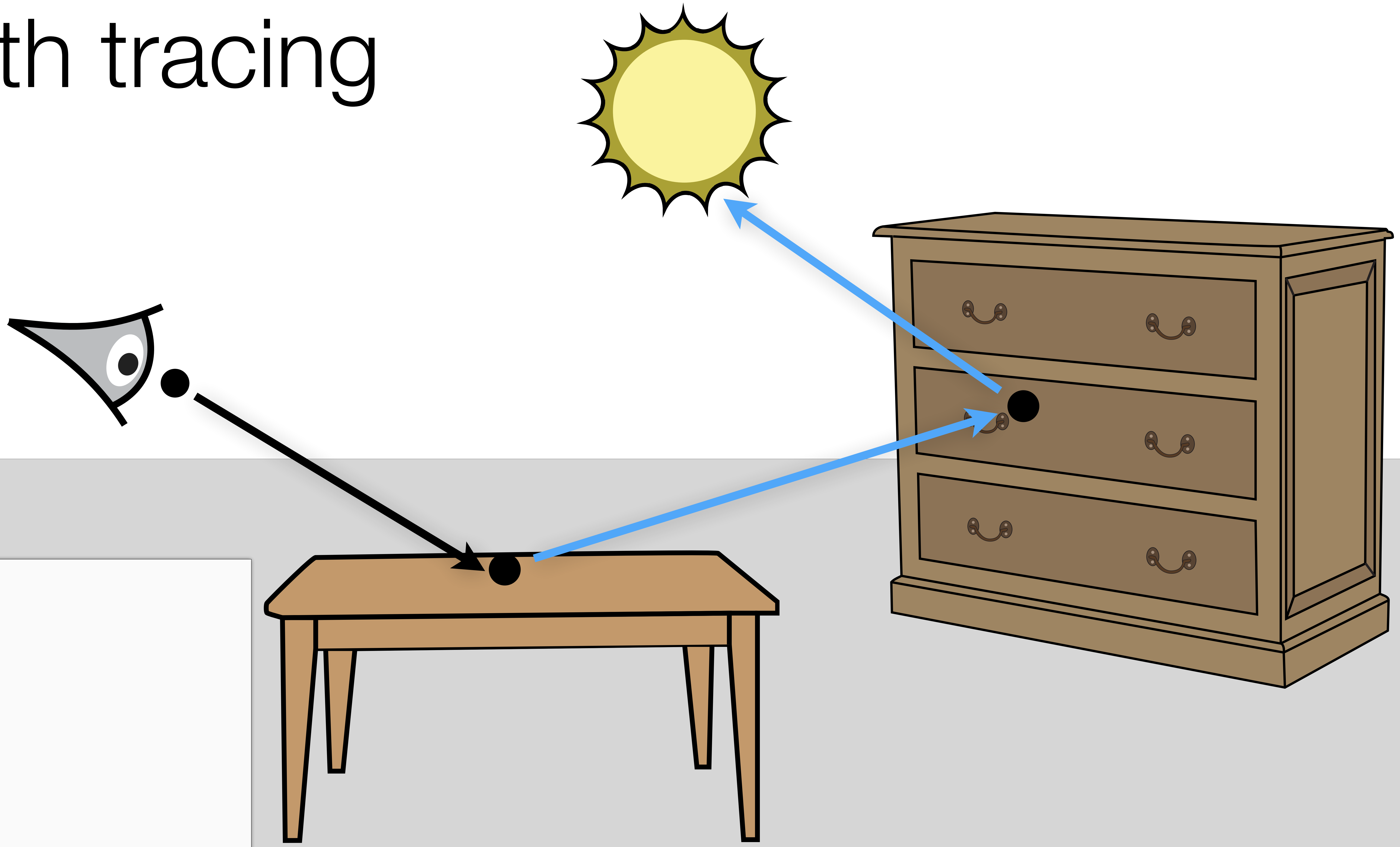


```
def mcmc_path_tracer():
    u = [0.5, …, 0.5]
```

Application to path tracing

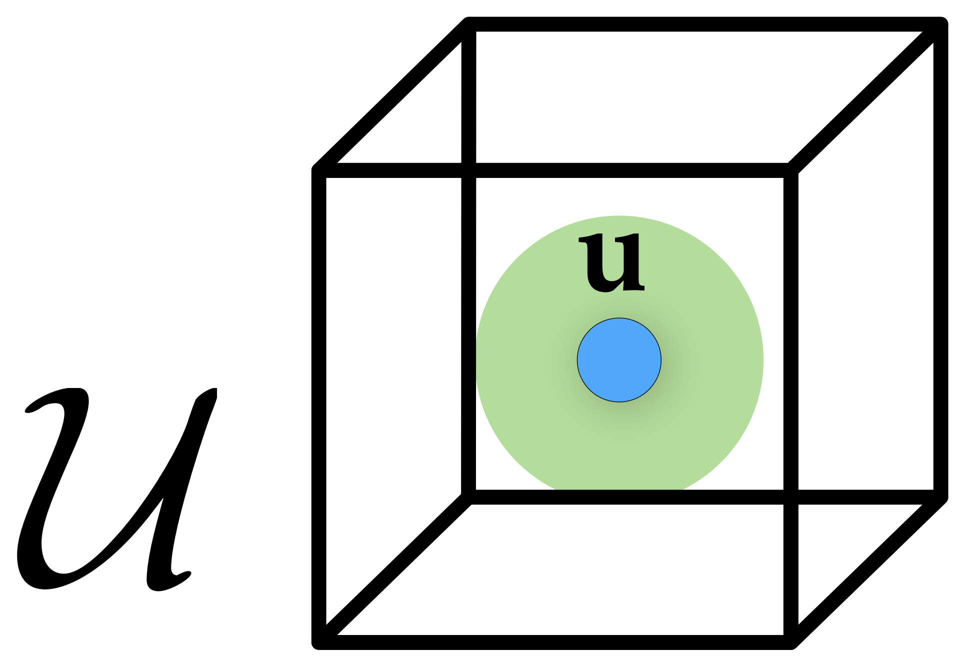


Hypercube of
“random numbers”

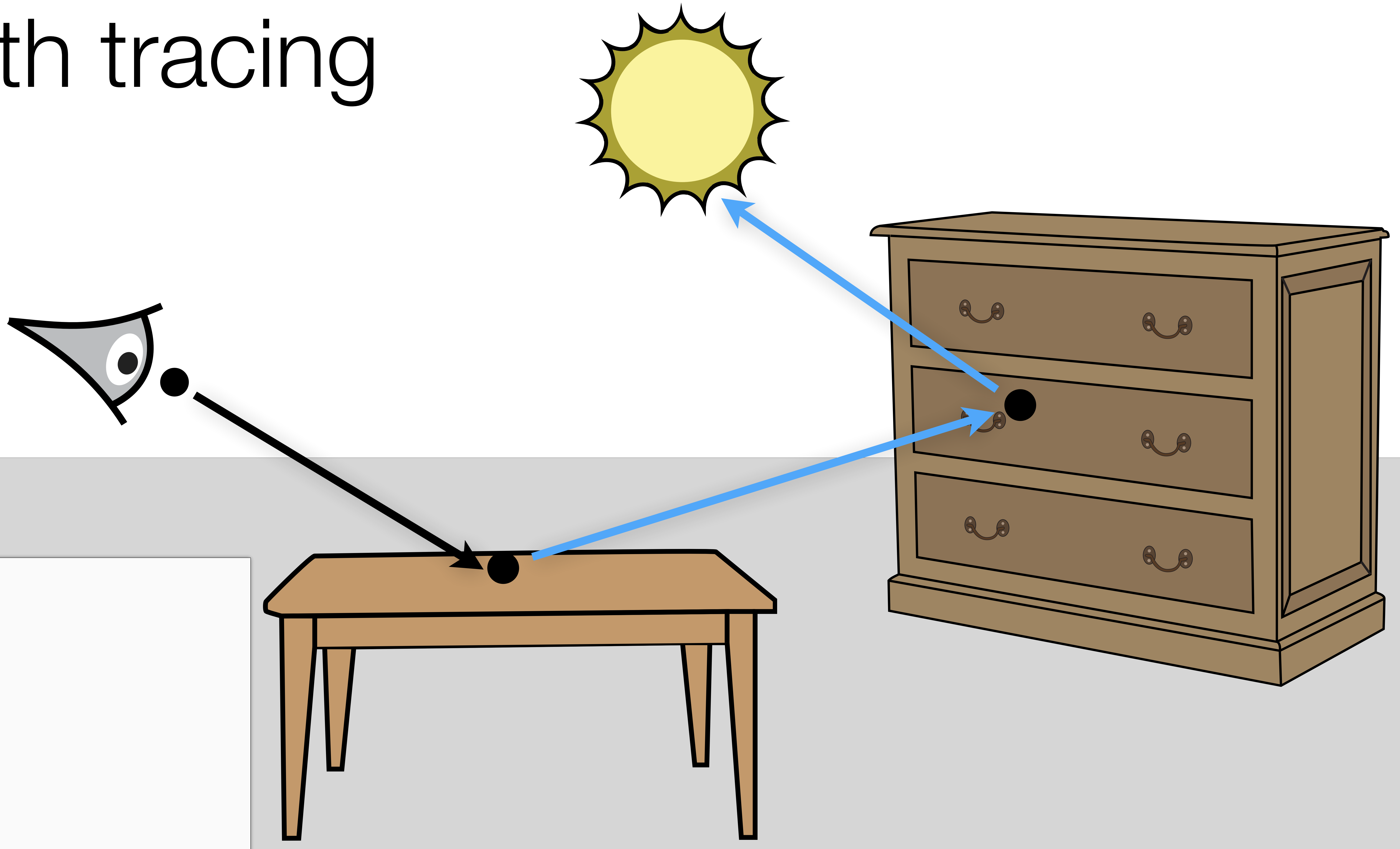


```
def mcmc_path_tracer():
    u = [0.5, …, 0.5]

    while !done:
```

Application to path tracing

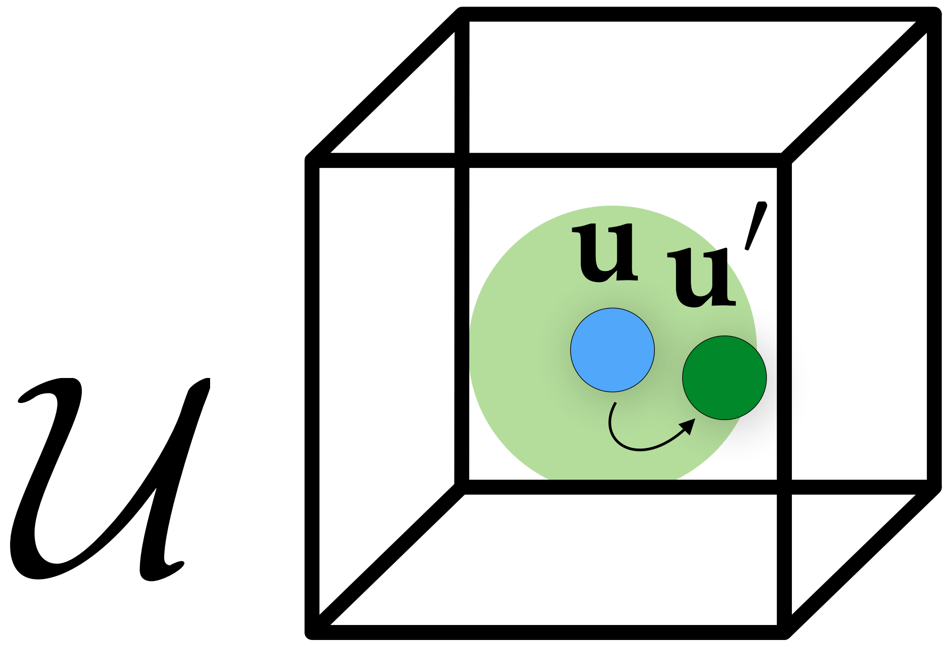


Hypercube of
“random numbers”

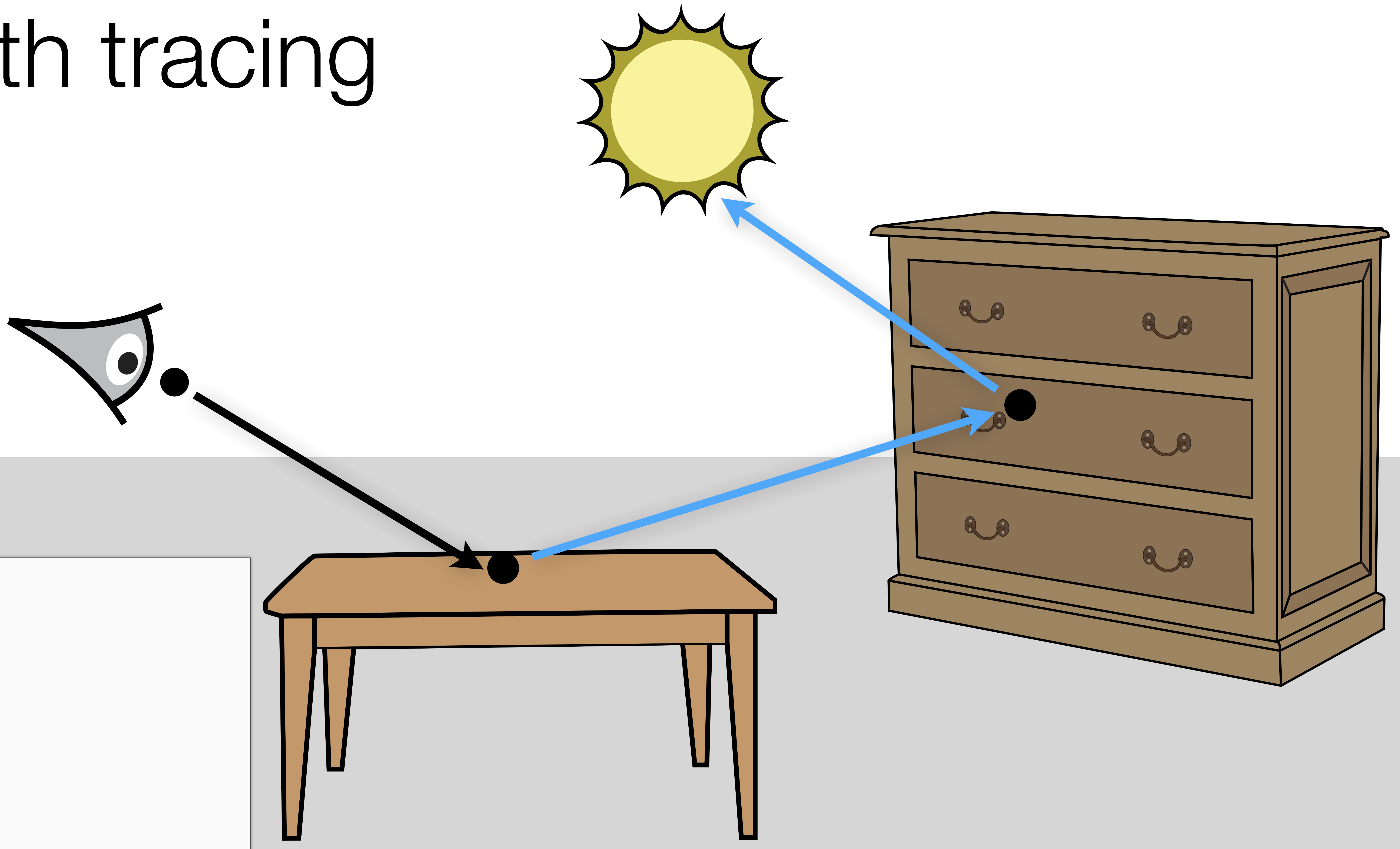


```
def mcmc_path_tracer():
    u = [0.5, …, 0.5]

    while !done:
        u’ = perturb(u)
```

Application to path tracing

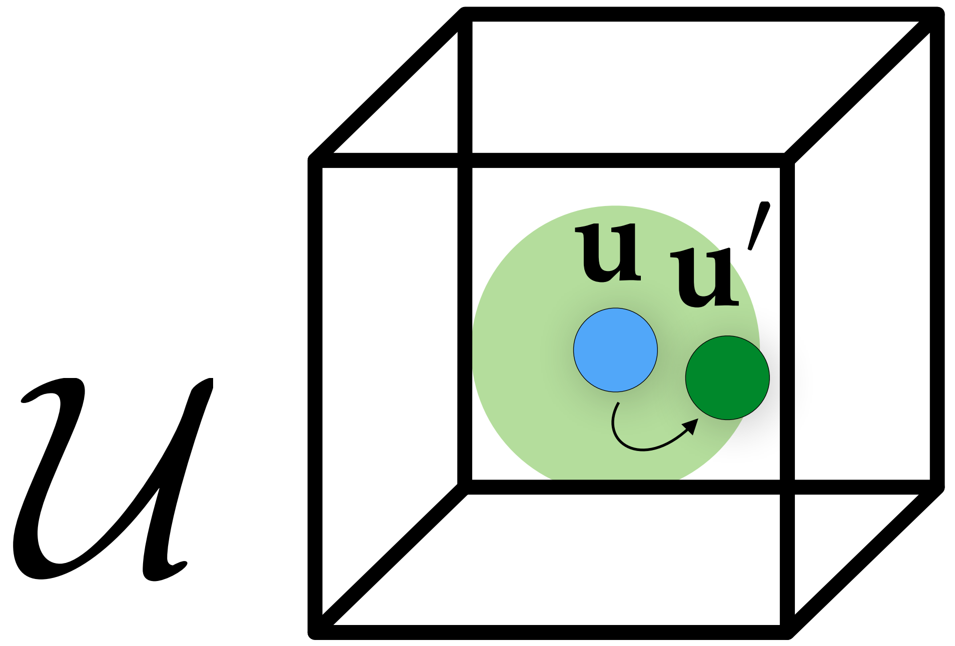


Hypercube of
"random numbers"

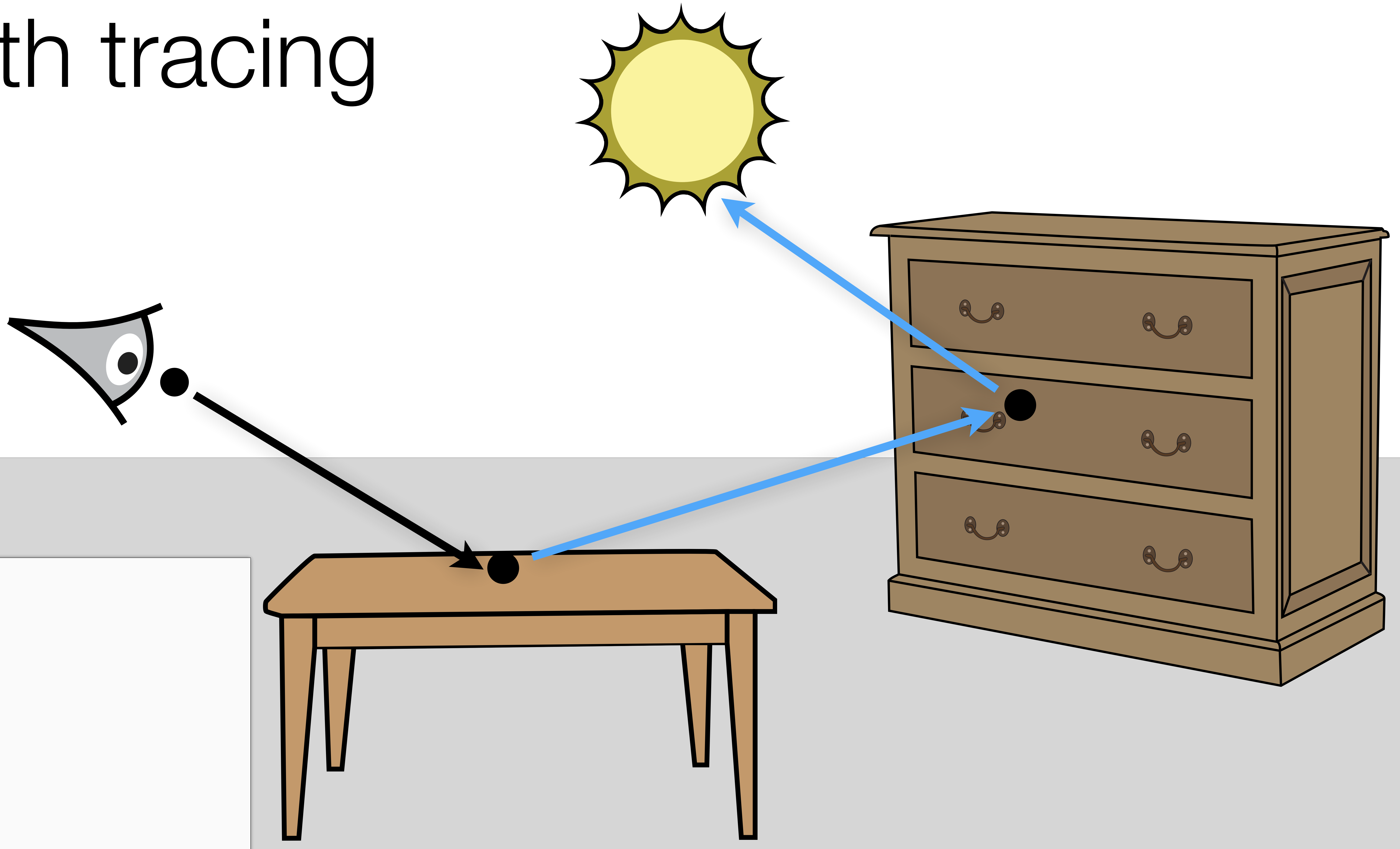


```
def mcmc_path_tracer():
    u = [0.5, …, 0.5]

    while !done:
        u’ = perturb(u)
```

Application to path tracing

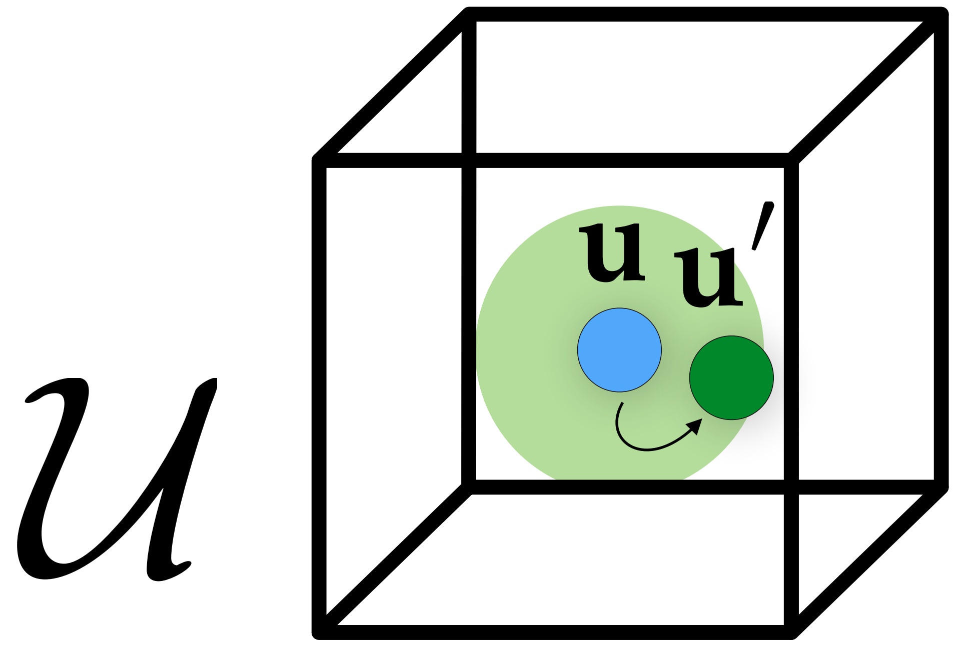


Hypercube of
“random numbers”

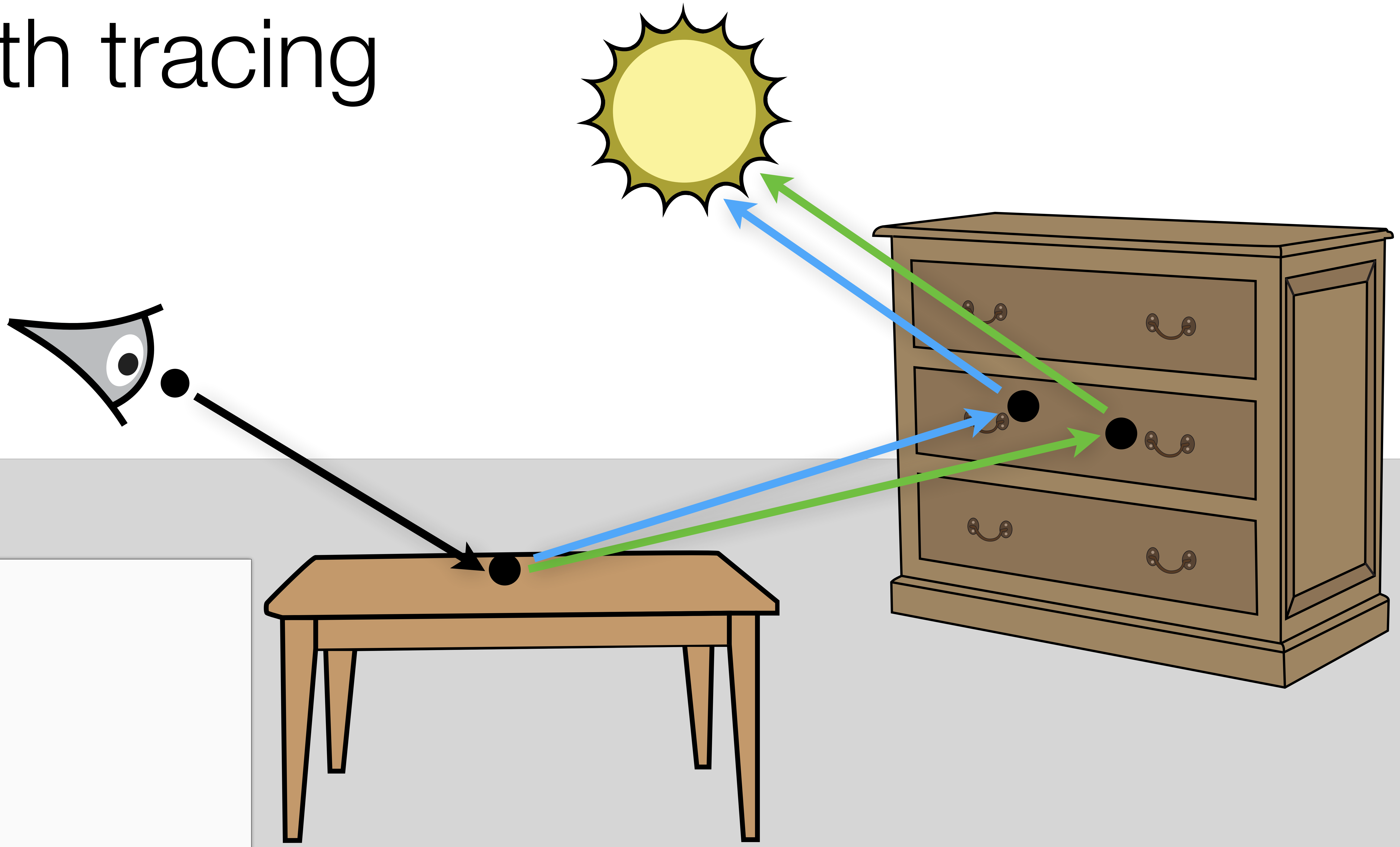


```
def mcmc_path_tracer():
    u = [0.5, …, 0.5]

    while !done:
        u’ = perturb(u)

        # Acceptance probability
        a = L(u’) / L(u)
```

Application to path tracing

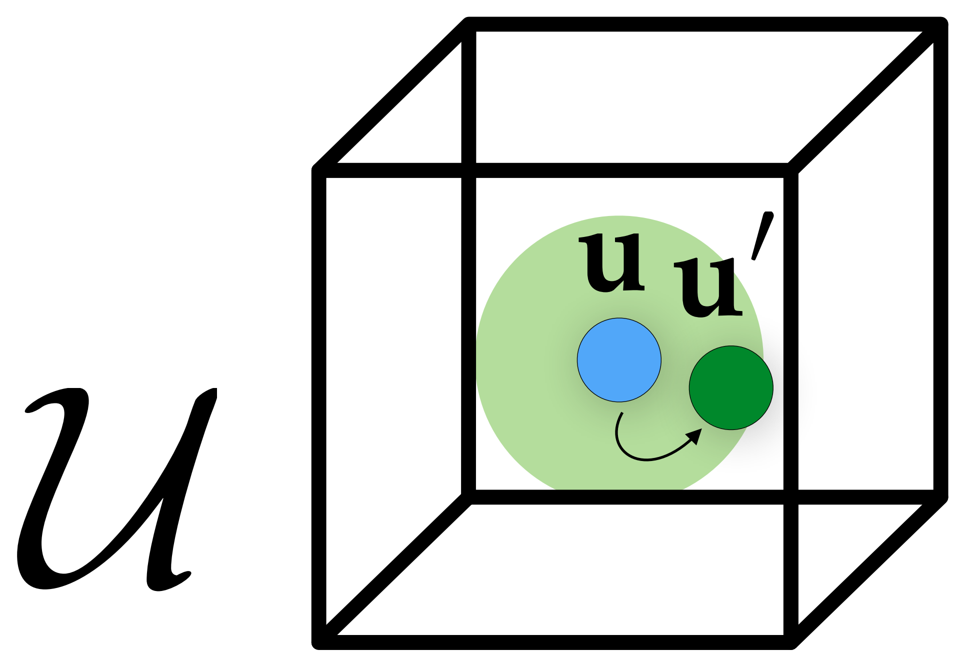


Hypercube of
"random numbers"



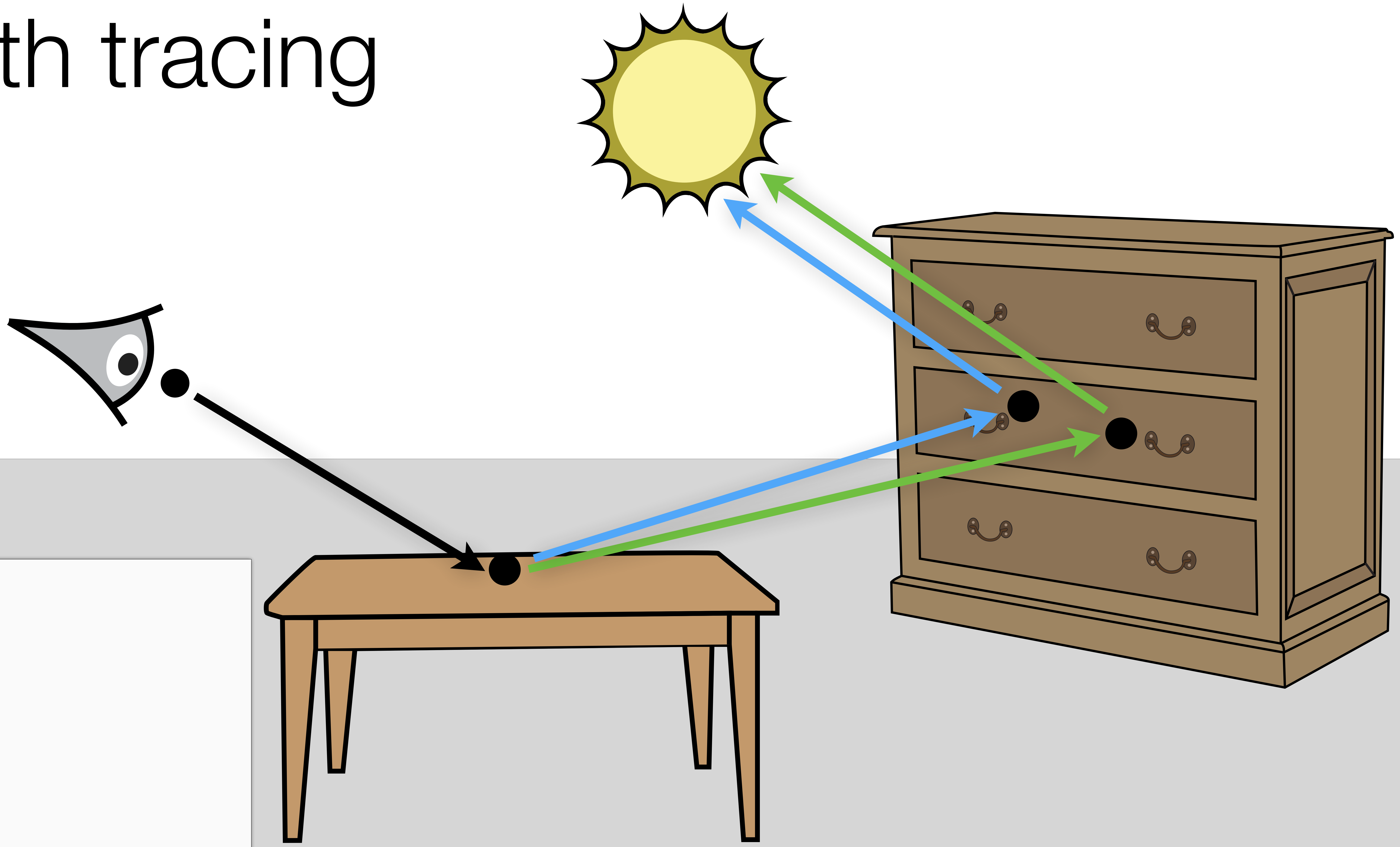
```
def mcmc_path_tracer():
    u = [0.5, …, 0.5]

    while !done:
        u' = perturb(u)

        # Acceptance probability
        a = L(u') / L(u)
```

Application to path tracing

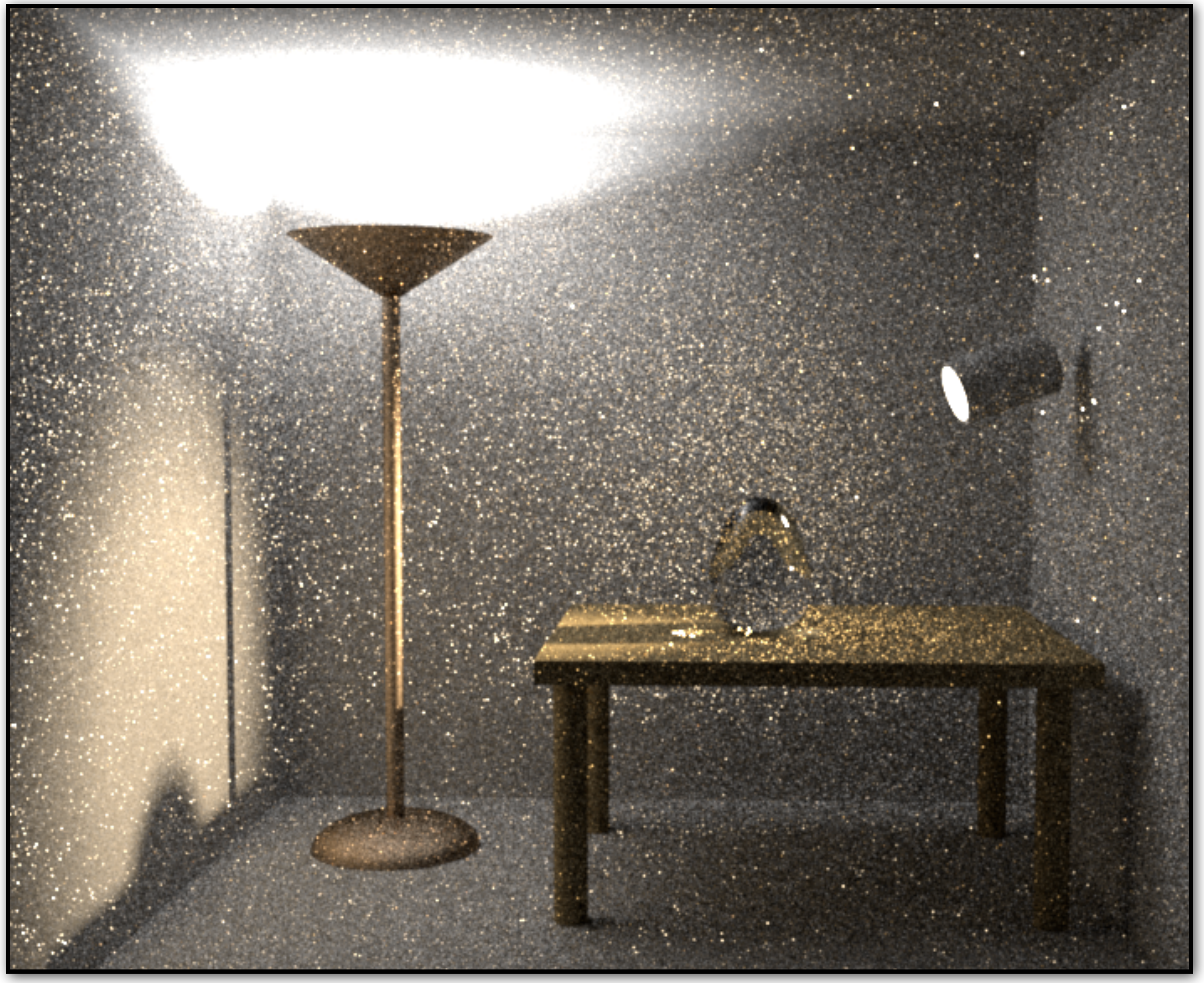


Hypercube of
“random numbers”

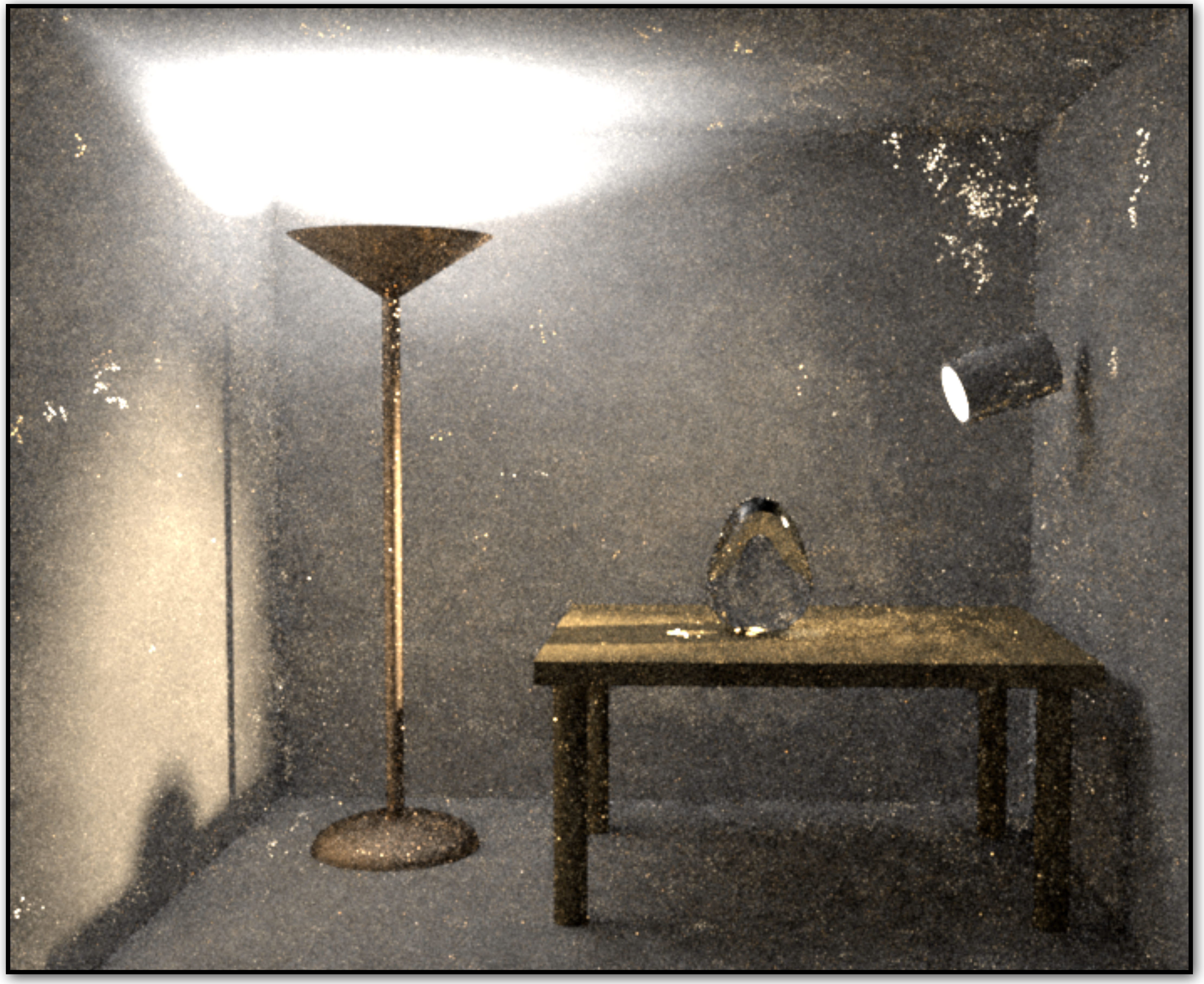
```
def mcmc_path_tracer():
    u = [0.5, …, 0.5]

    while !done:
        u’ = perturb(u)

        # Acceptance probability
        a = L(u’) / L(u)

        if rand() < a:
            u = u’
```



Equal-time comparison

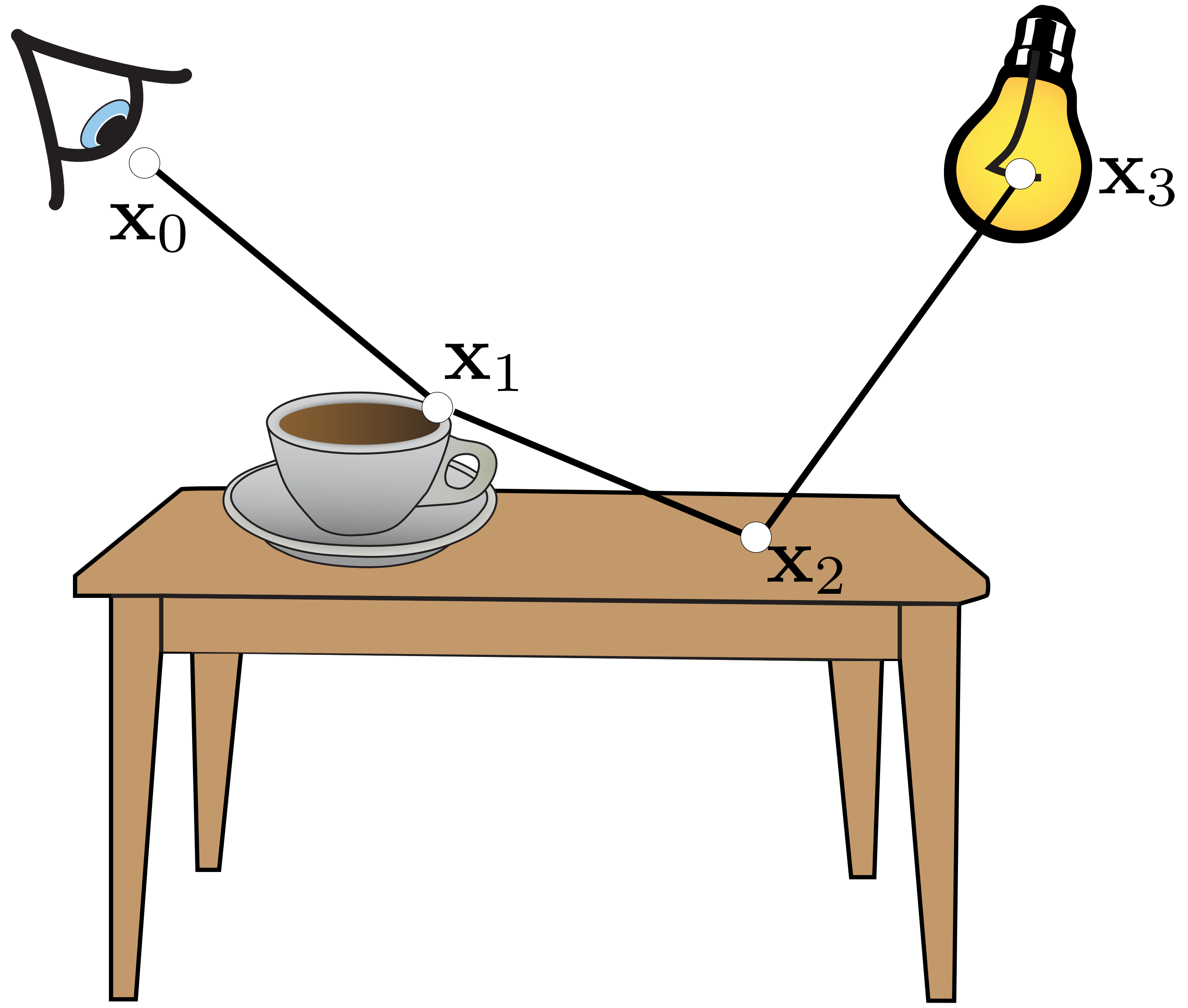


Path tracing

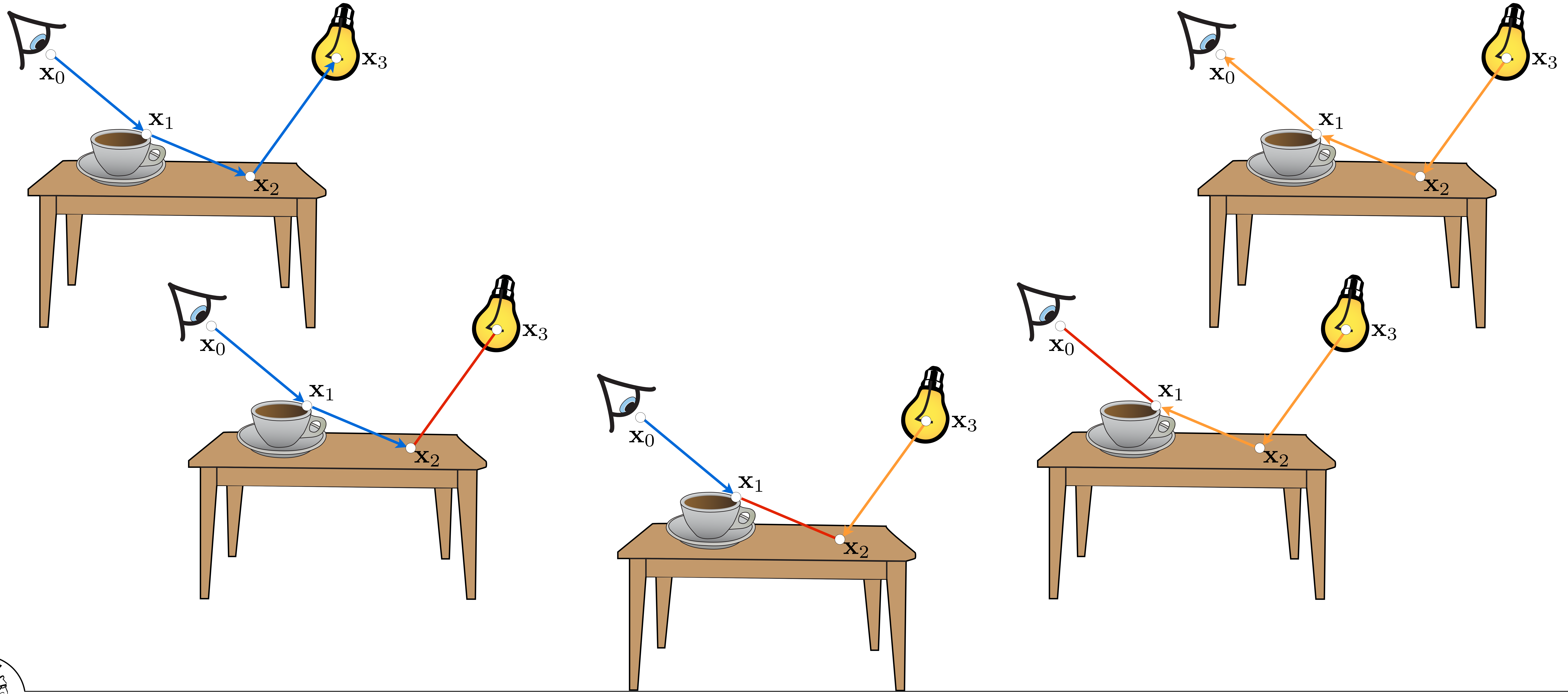


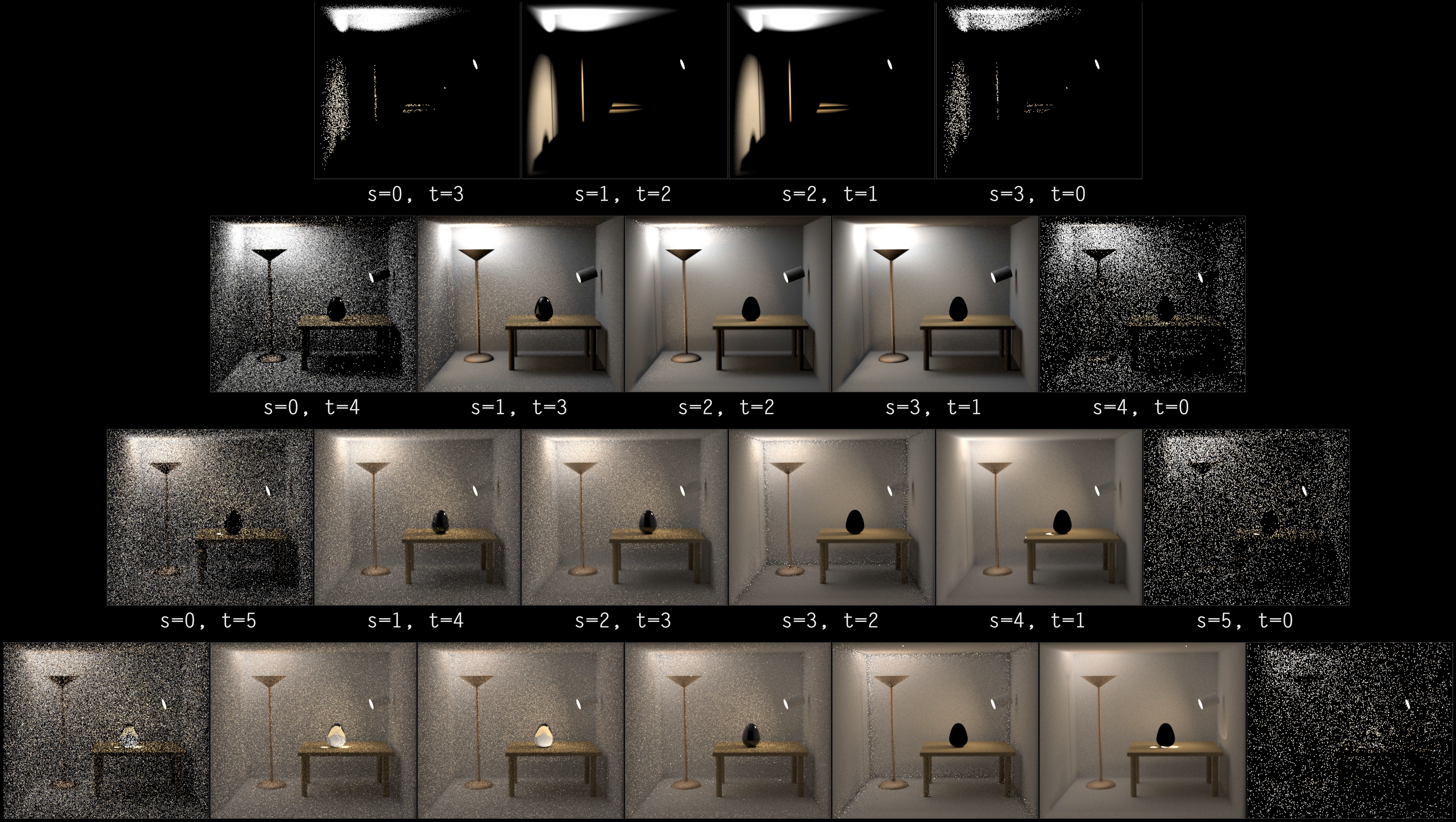
"Metropolized" Path tracing

Sampling light paths bidirectionally



Sampling light paths bidirectionally



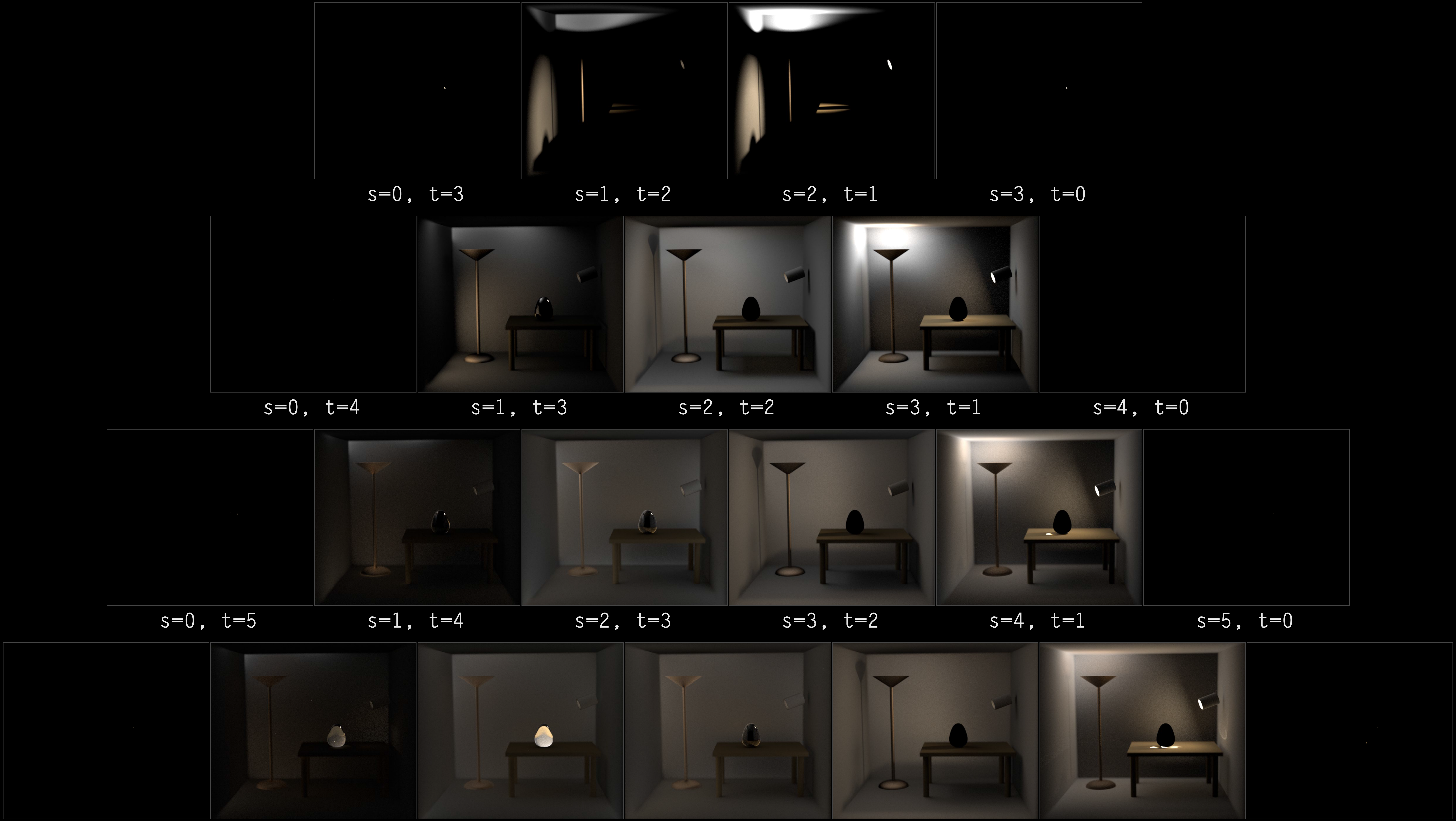


s=0, t=3 s=1, t=2 s=2, t=1 s=3, t=0

s=0, t=4 s=1, t=3 s=2, t=2 s=3, t=1 s=4, t=0

s=0, t=5 s=1, t=4 s=2, t=3 s=3, t=2 s=4, t=1 s=5, t=0

s=0, t=6 s=1, t=5 s=2, t=4 s=3, t=3 s=4, t=2 s=5, t=1 s=6, t=0



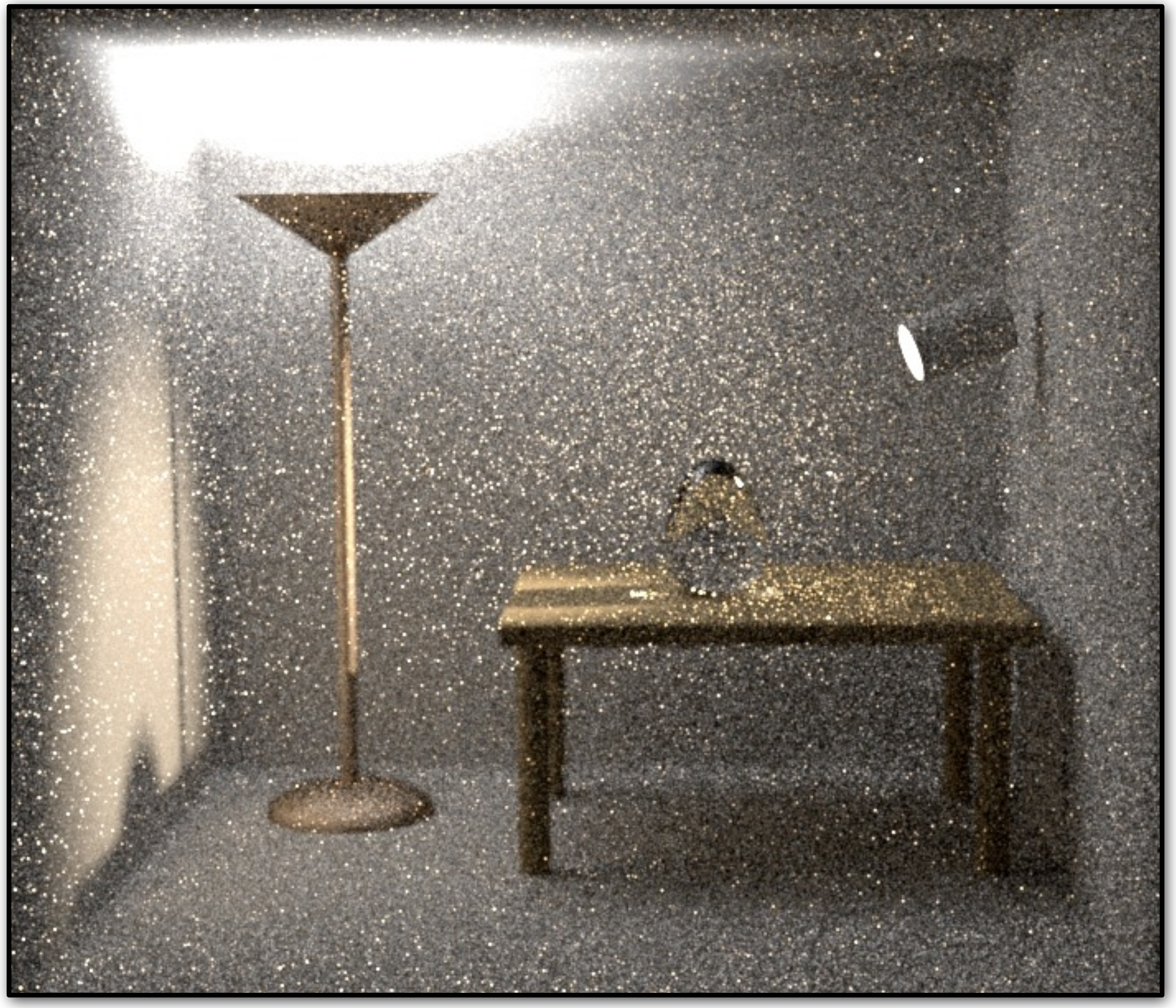
s=0, t=3 s=1, t=2 s=2, t=1 s=3, t=0

s=0, t=4 s=1, t=3 s=2, t=2 s=3, t=1 s=4, t=0

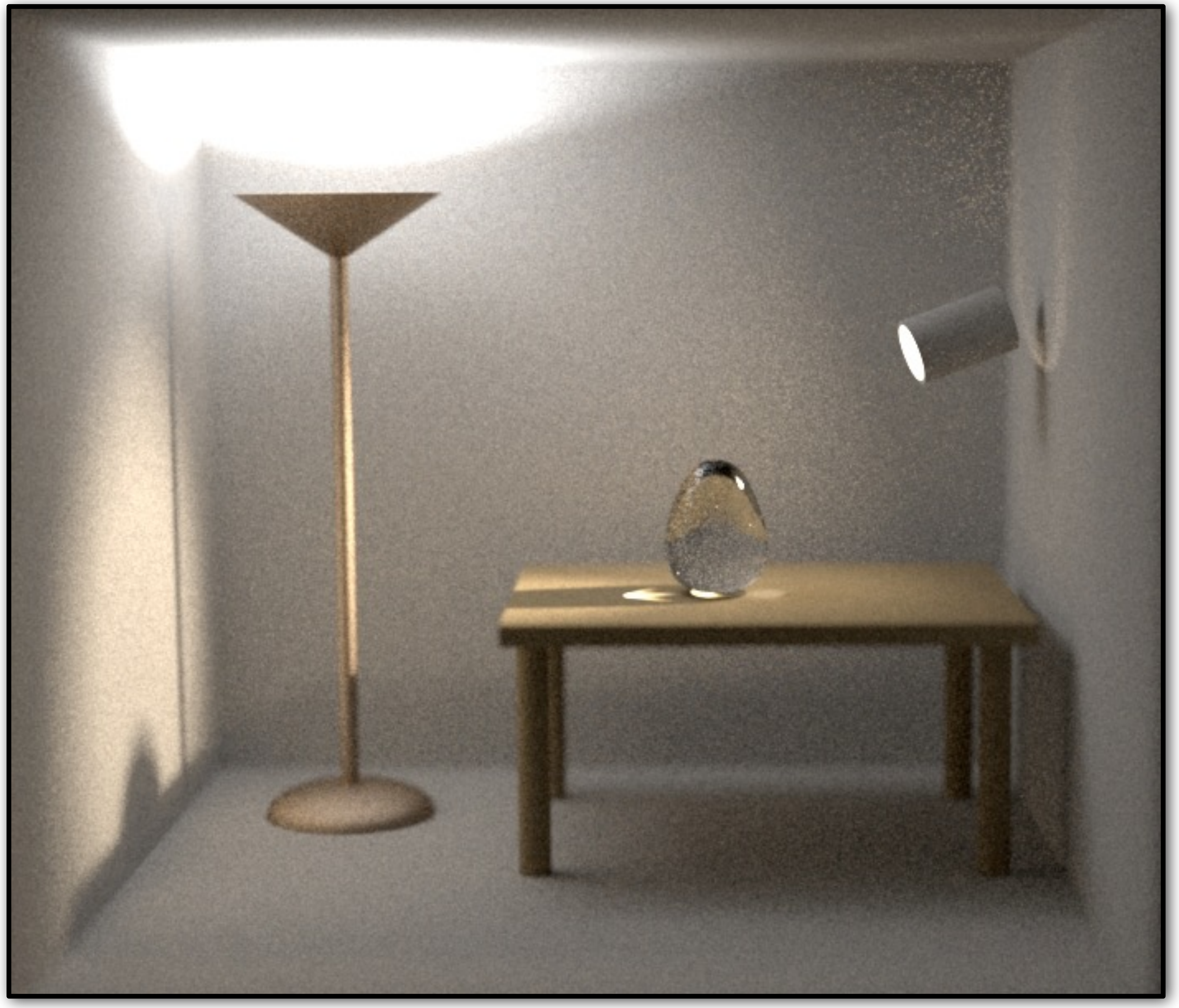
s=0, t=5 s=1, t=4 s=2, t=3 s=3, t=2 s=4, t=1 s=5, t=0

s=0, t=6 s=1, t=5 s=2, t=4 s=3, t=3 s=4, t=2 s=5, t=1 s=6, t=0

Equal-time comparison



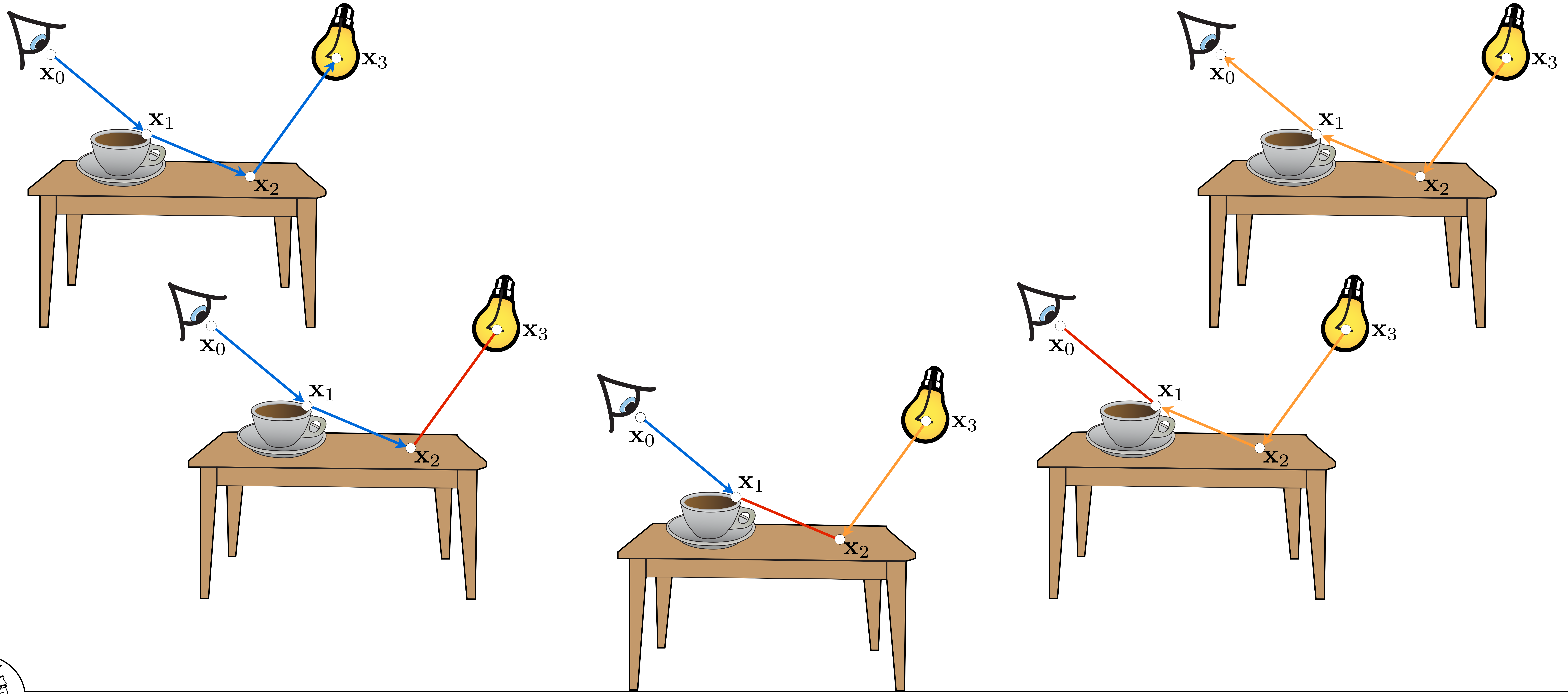
Path tracing



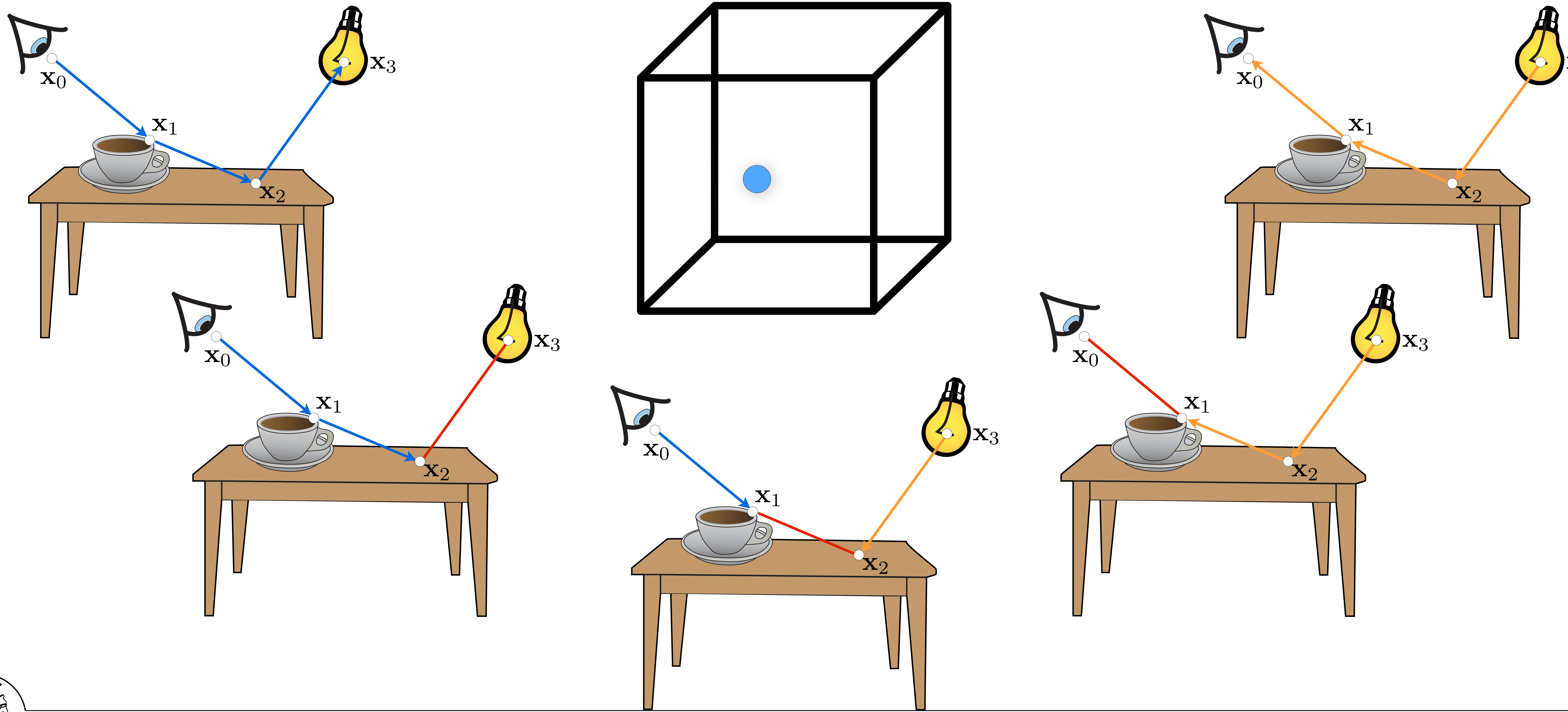
Bidirectional tracing

Multiplexing: exploration using multiple strategies

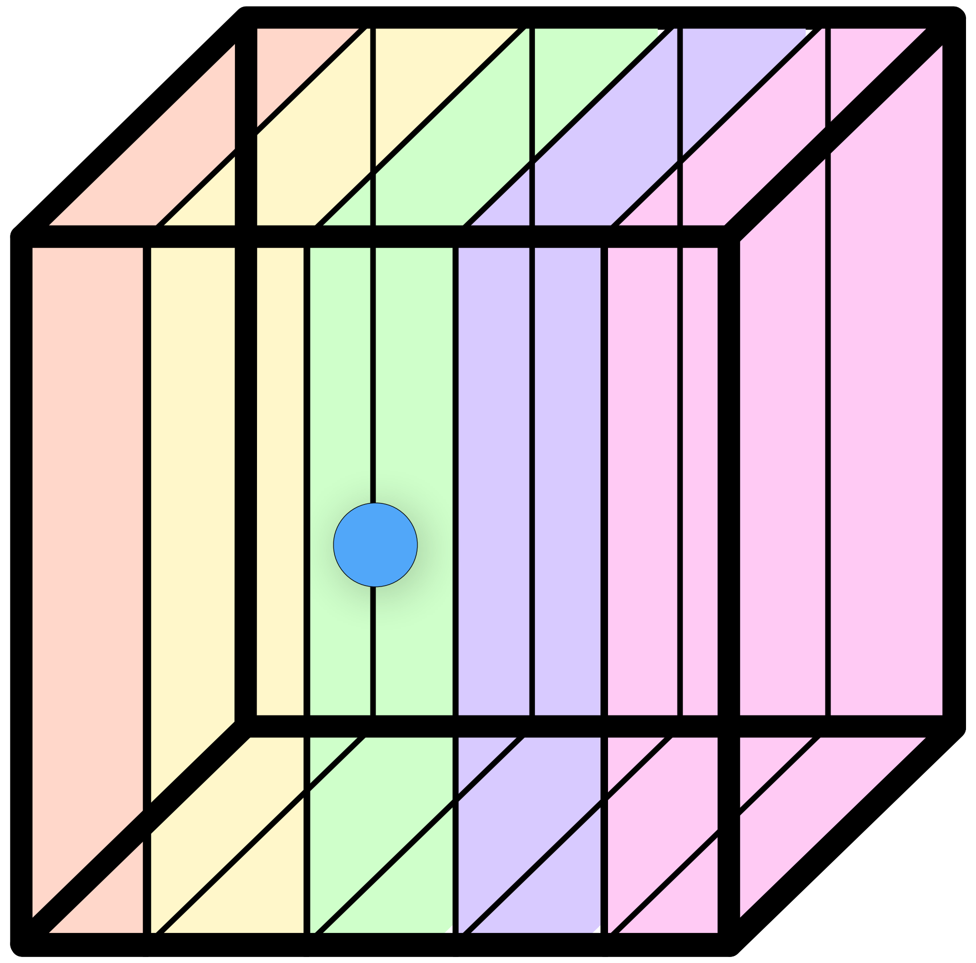
Multiplexing: exploration using multiple strategies

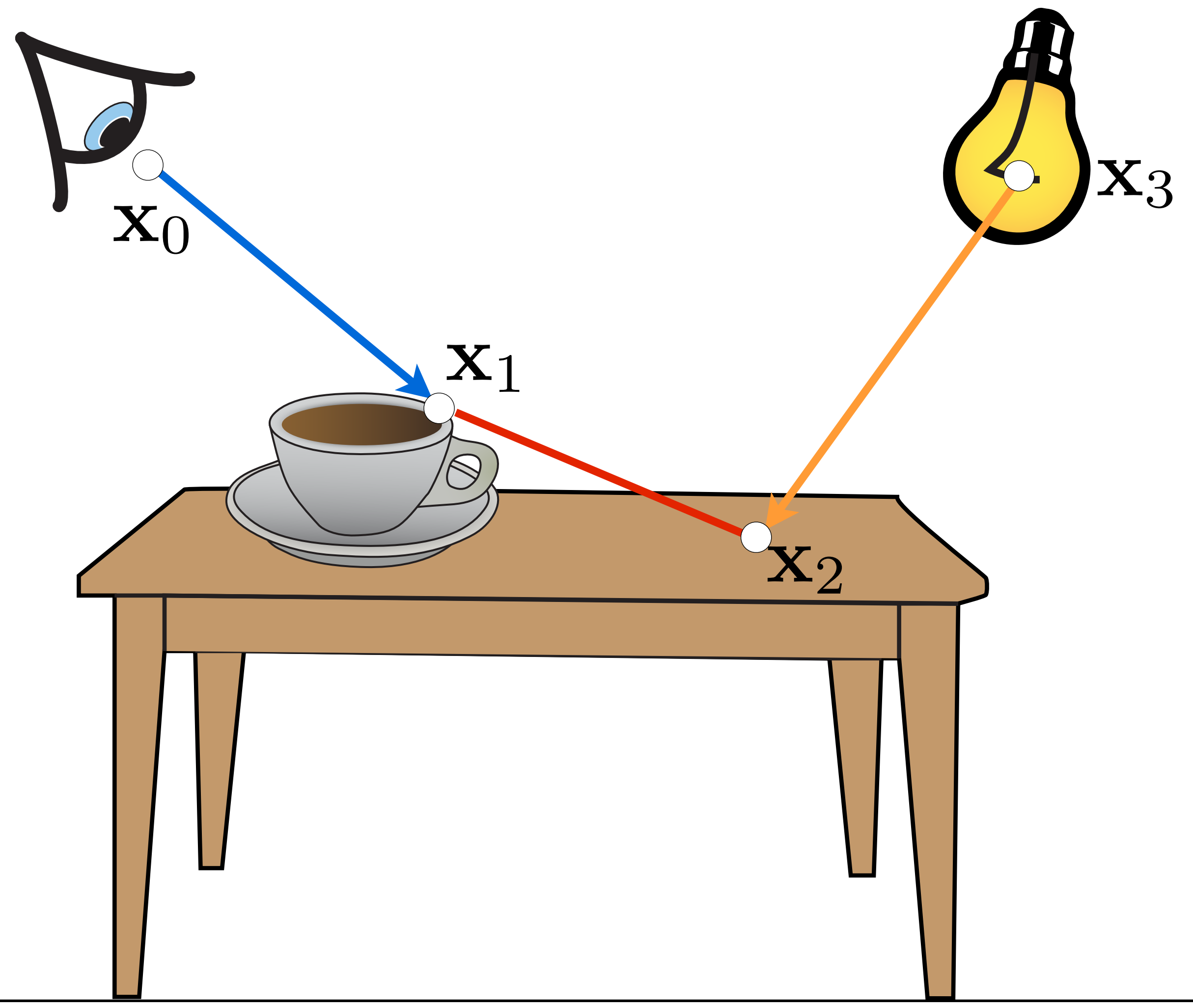


Multiplexing: exploration using multiple strategies



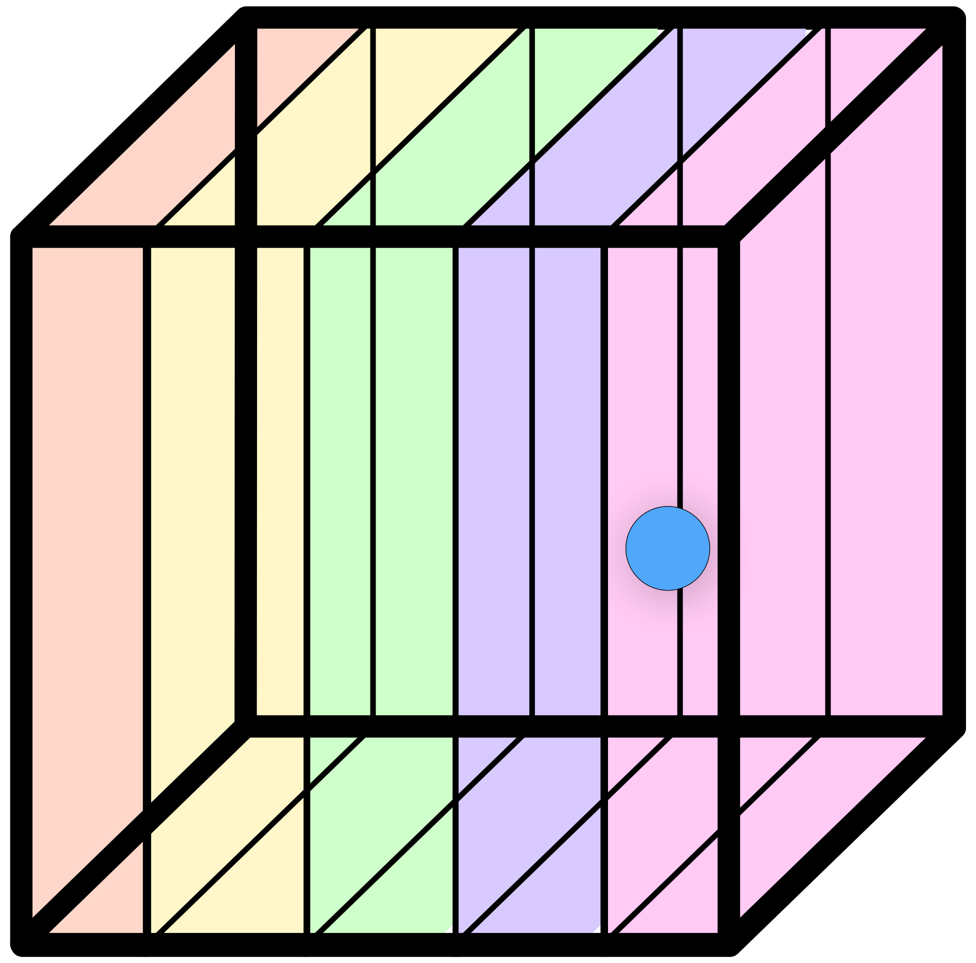
Multiplexing: exploration using multiple strategies

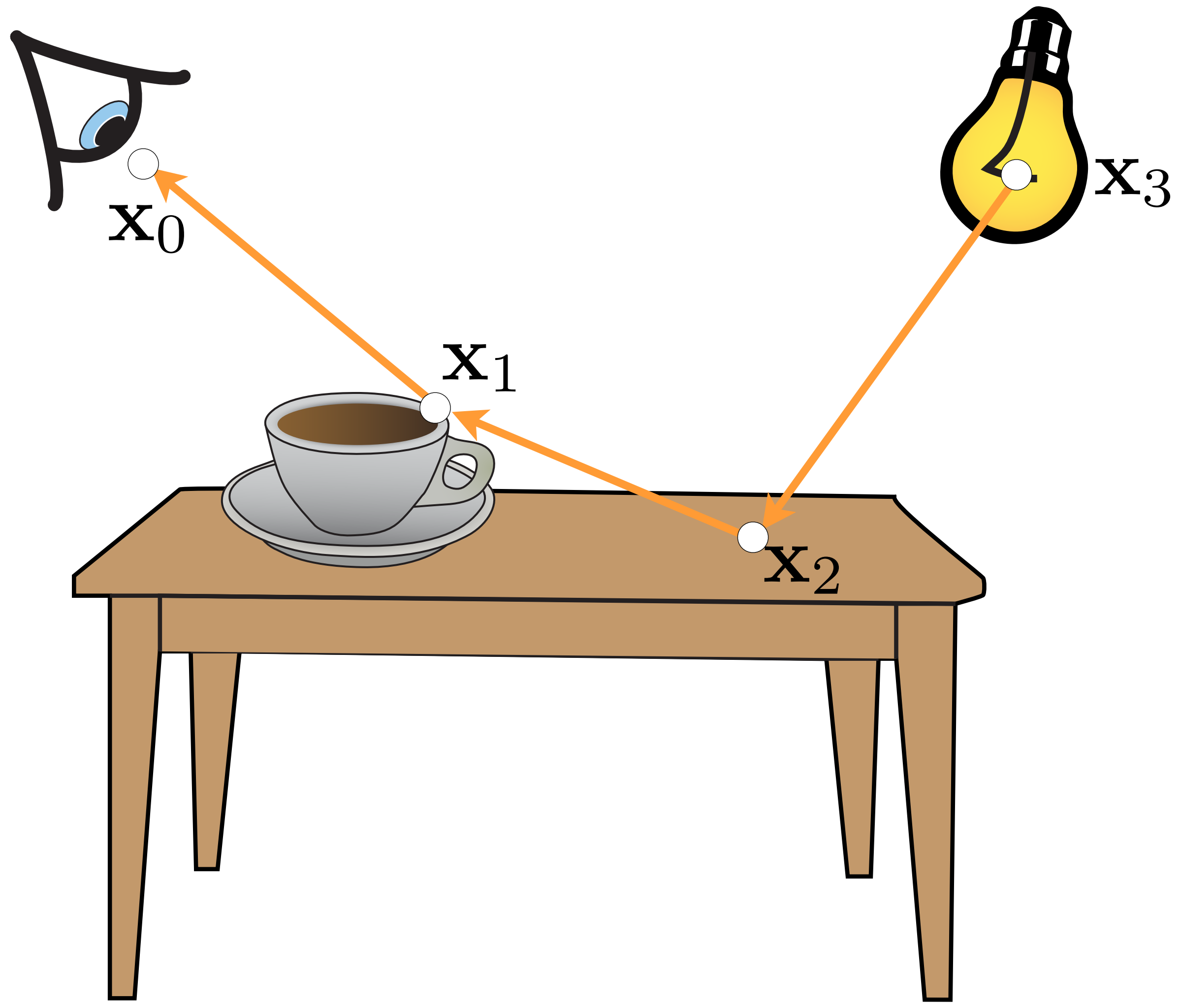




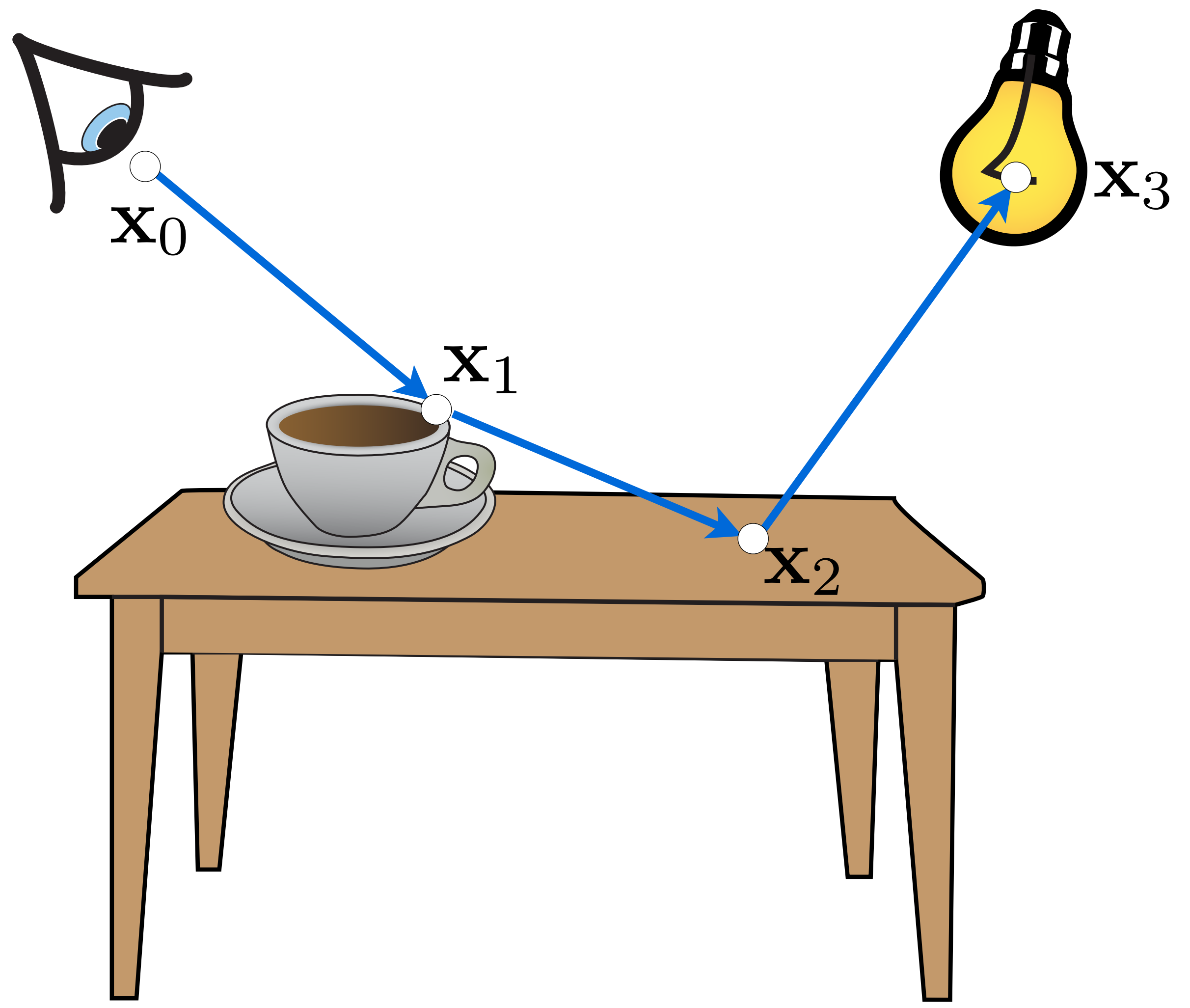
EPFL

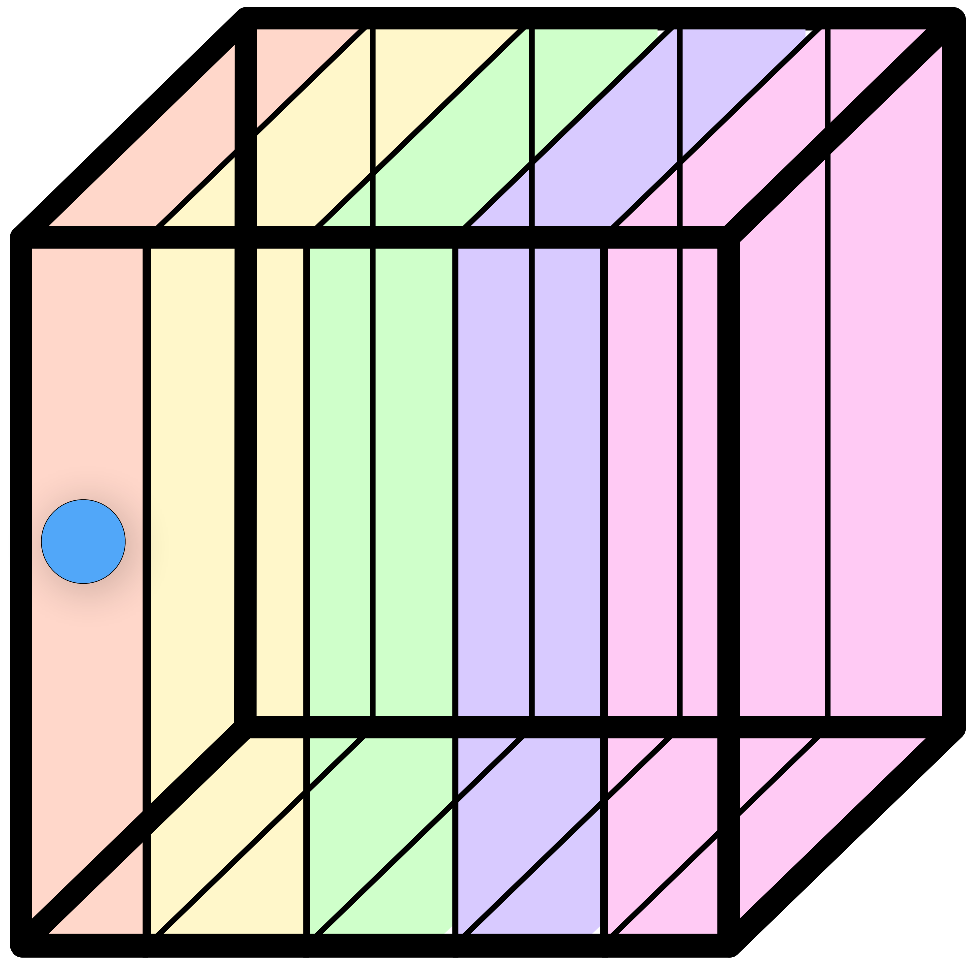
Multiplexing: exploration using multiple strategies



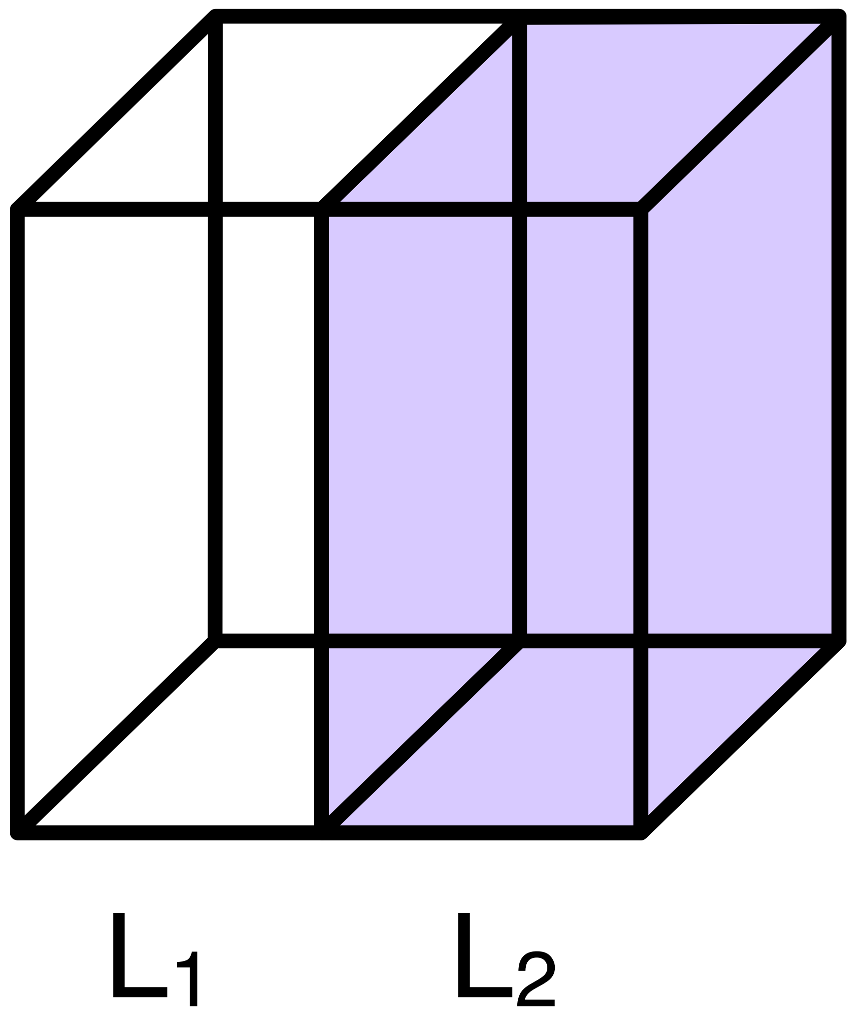


Multiplexing: exploration using multiple strategies



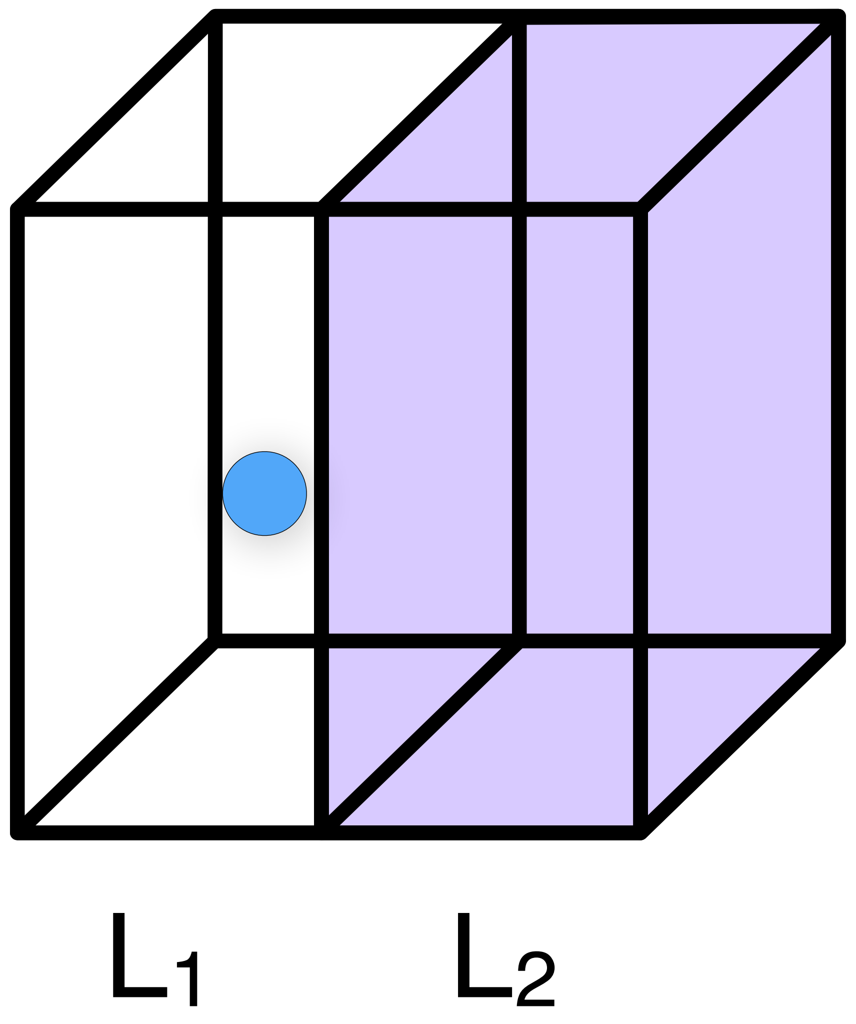


What *actually* happens :(

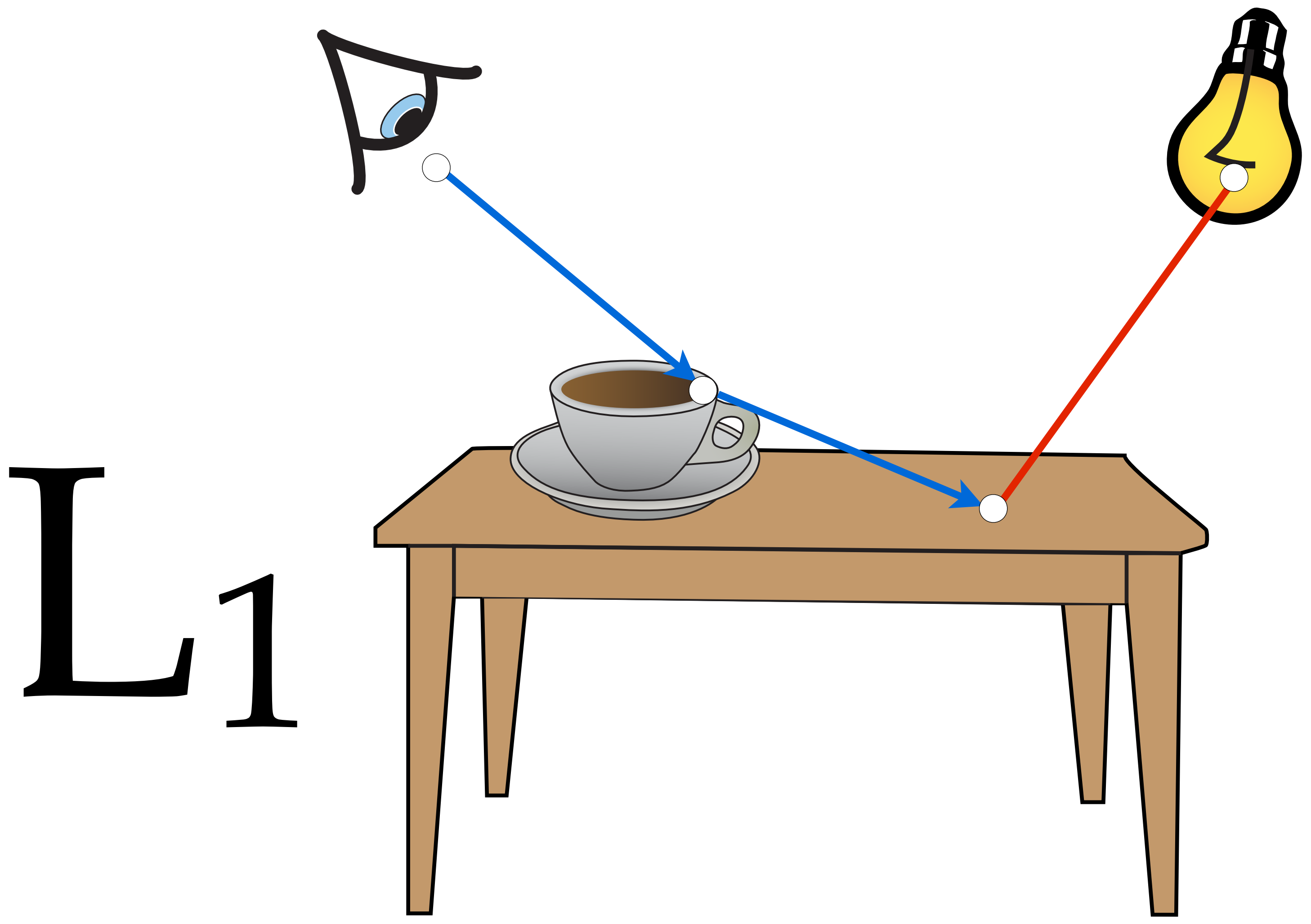


```
def L(x, ω, u):
    if u1 < 0.5:
        return L1(x, ω, u2…un)
```

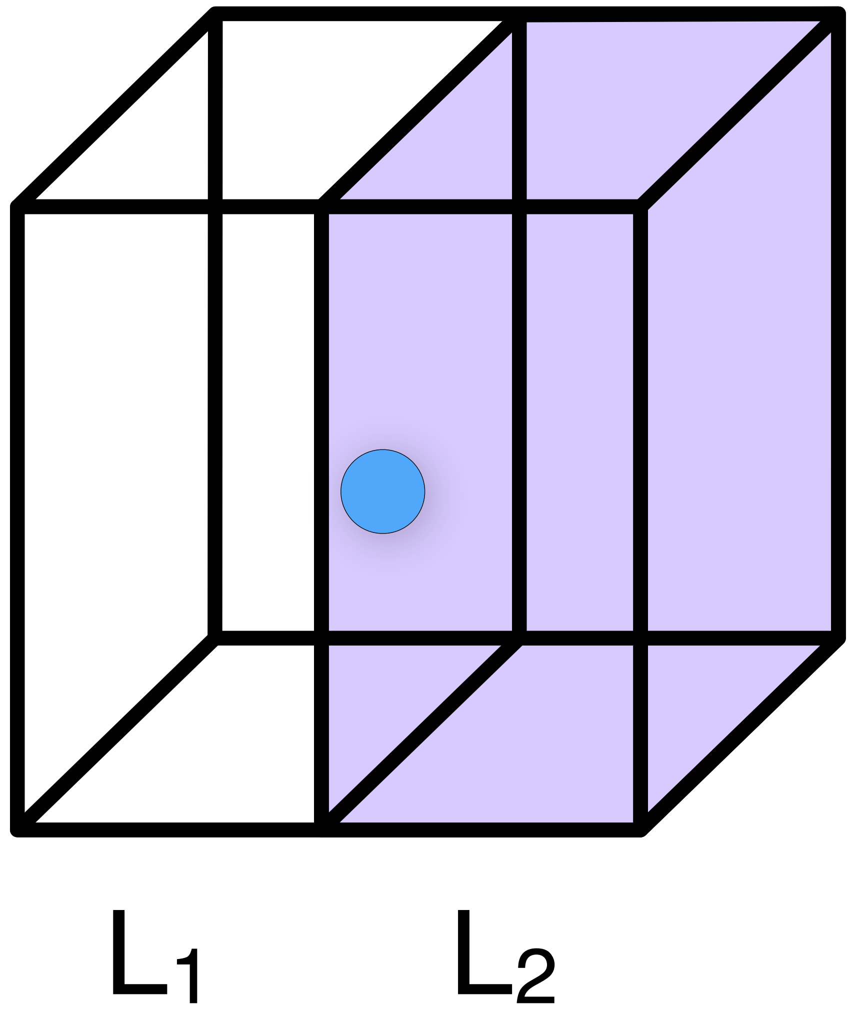
What *actually* happens :(



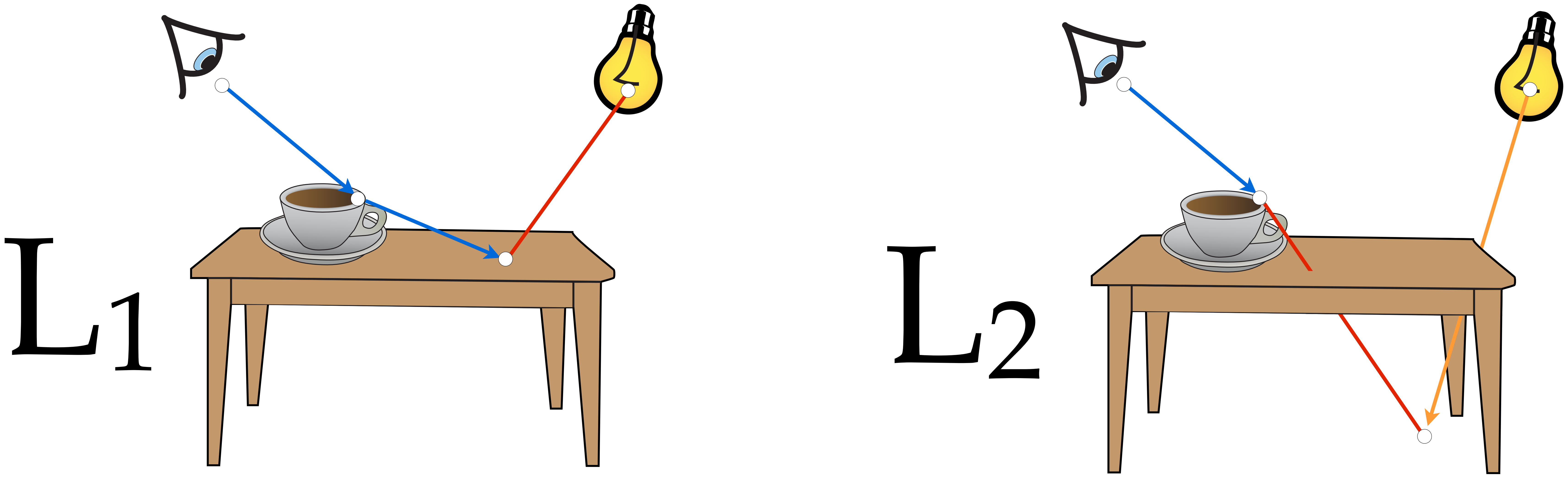
```
def L(x, ω, u):
    if $u_1 < 0.5$:
        return $L_1(x, ω, u_2 \ldots u_n)$
```



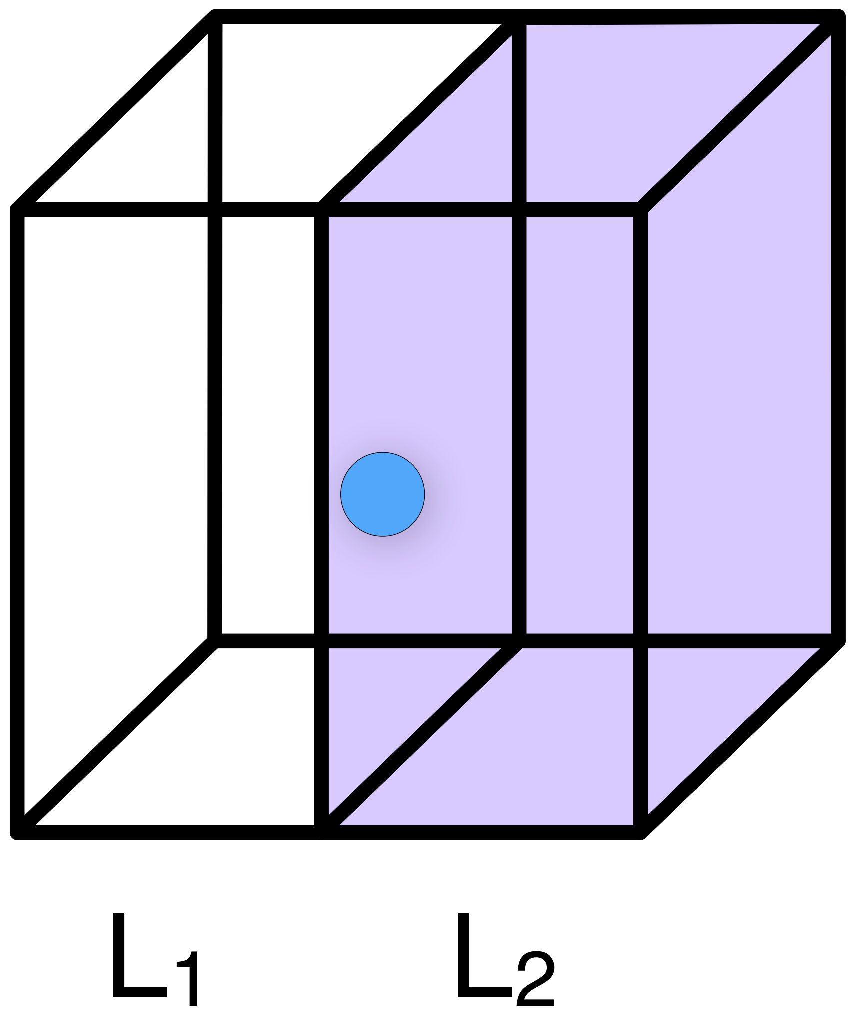
What *actually* happens :(



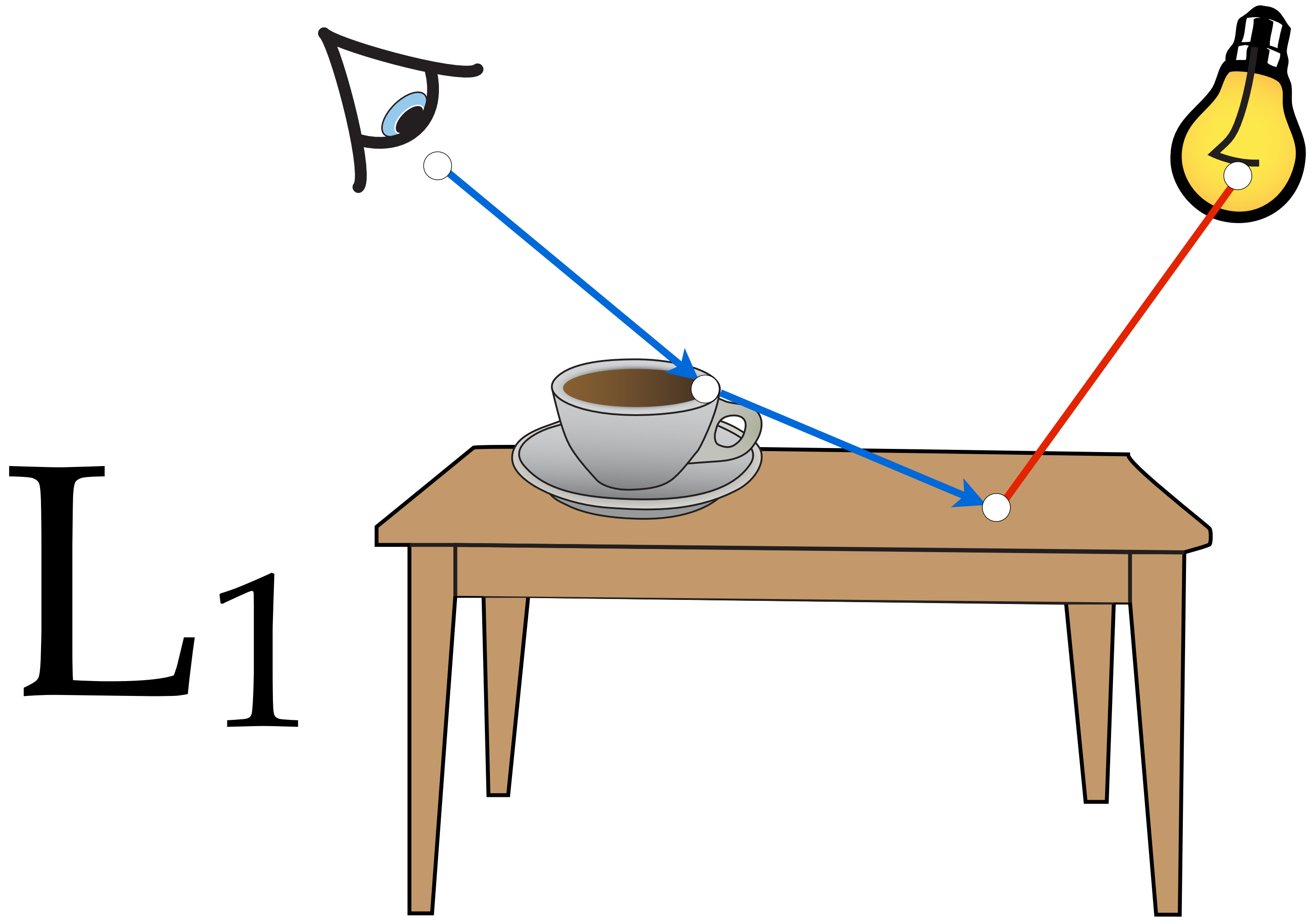
```
def L(x, ω, u):



    else:
        return L2(x, ω, u2…un)
```

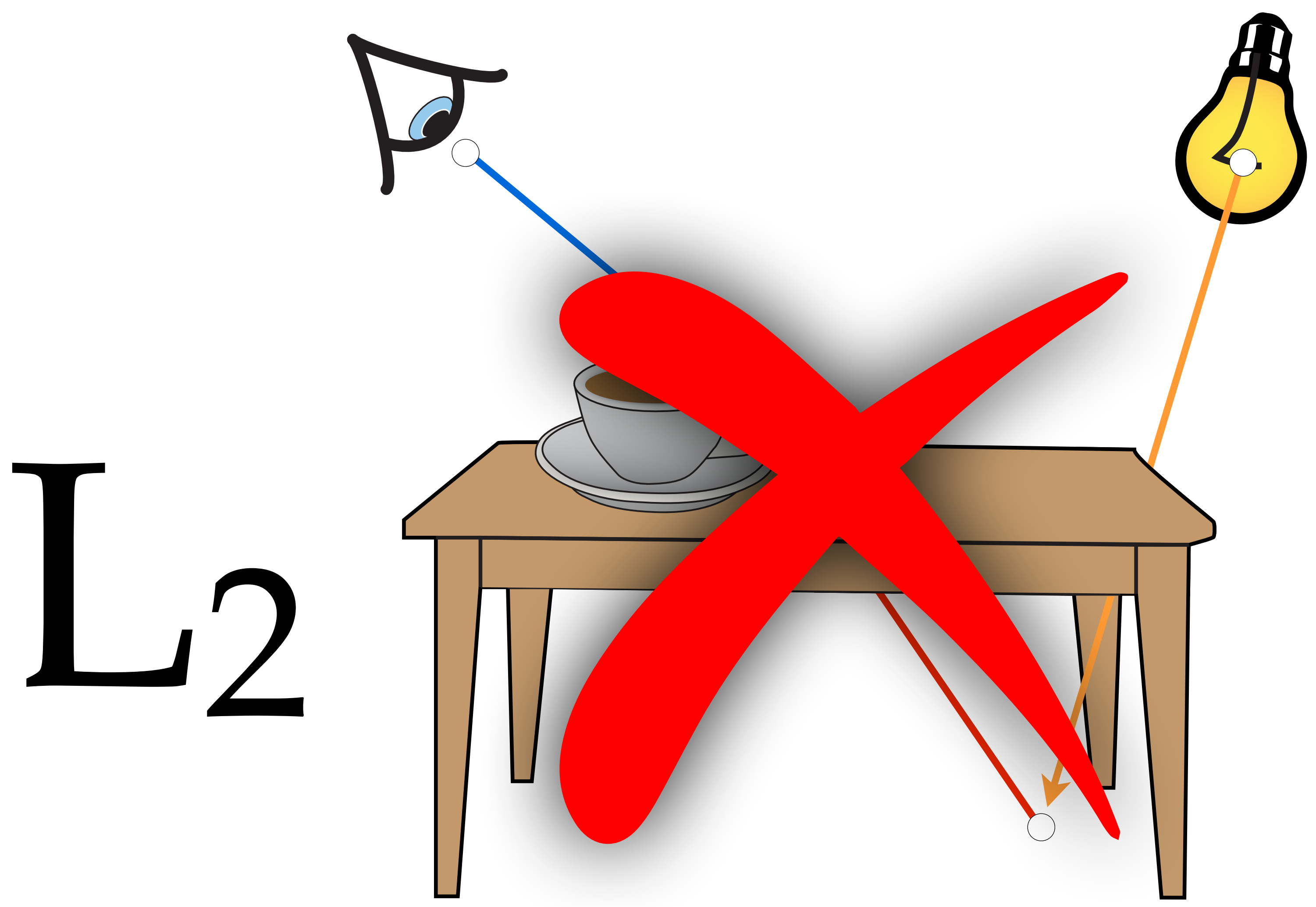


What *actually* happens :(

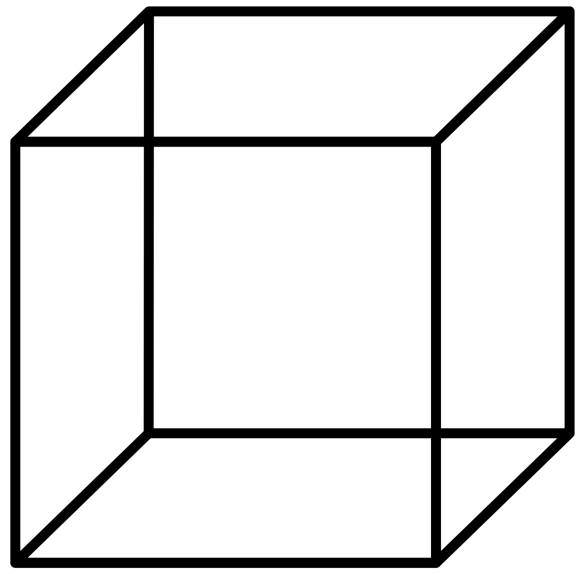


```
def L(x, ω, u):


    else:
        return L2(x, ω, u2…un)
```

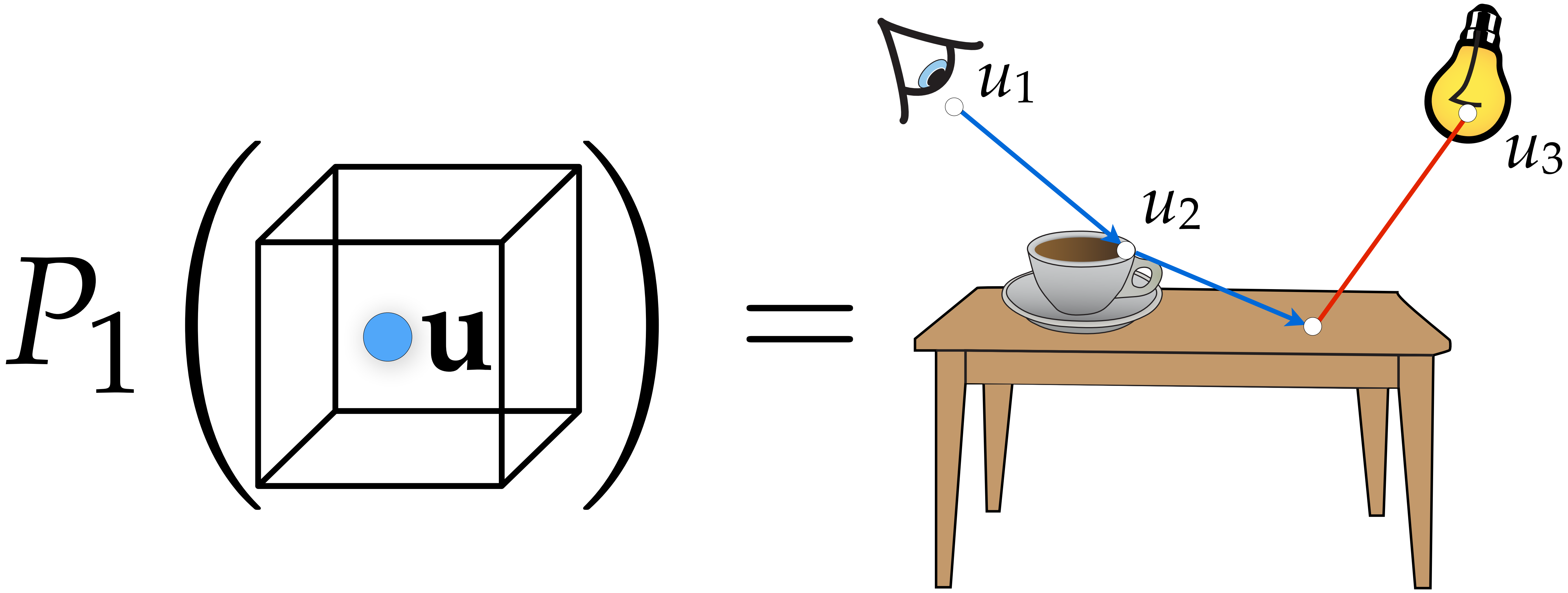




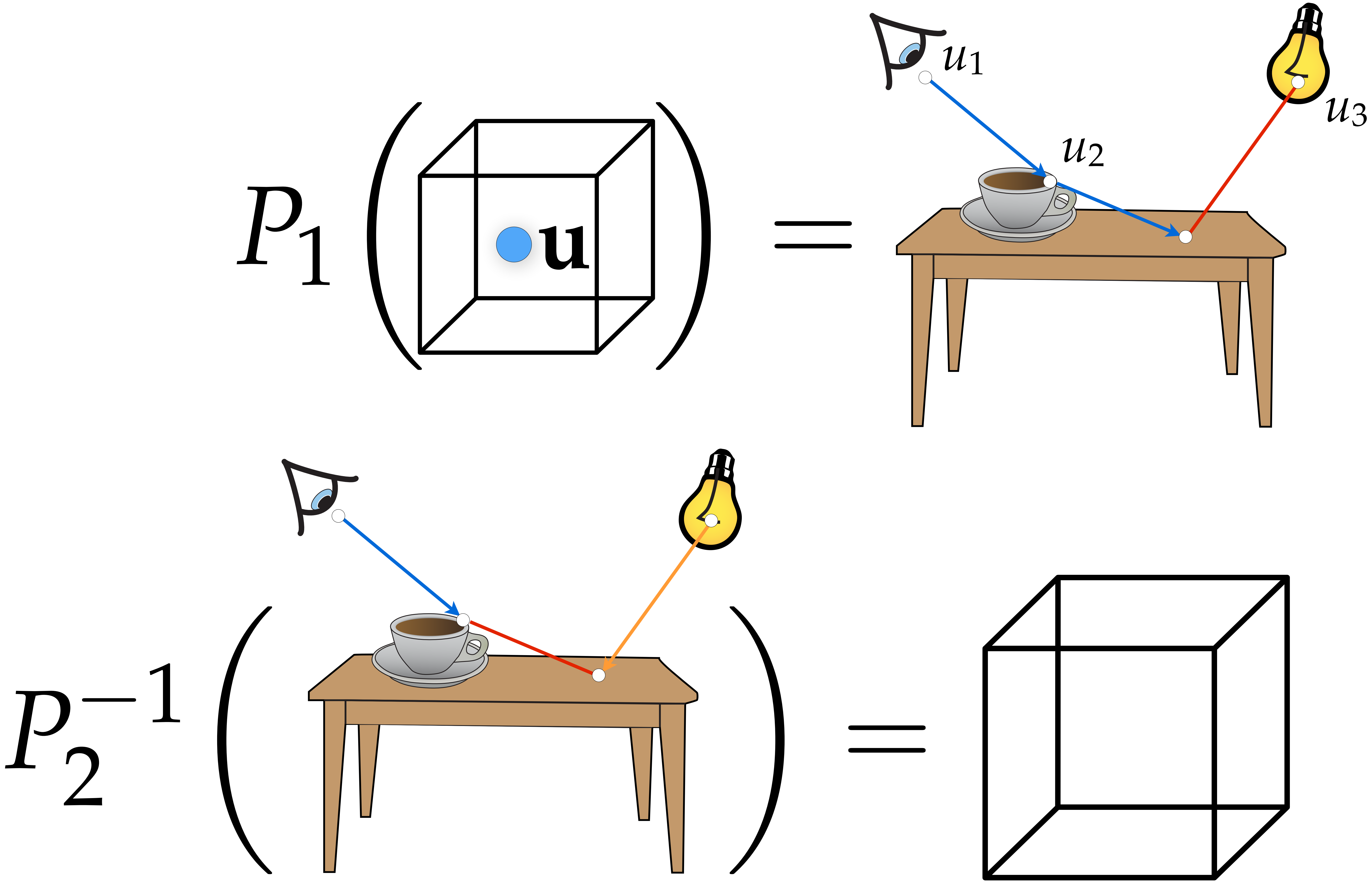
Solution: path inverses

$P_1\Bigg($  $\Bigg) =$

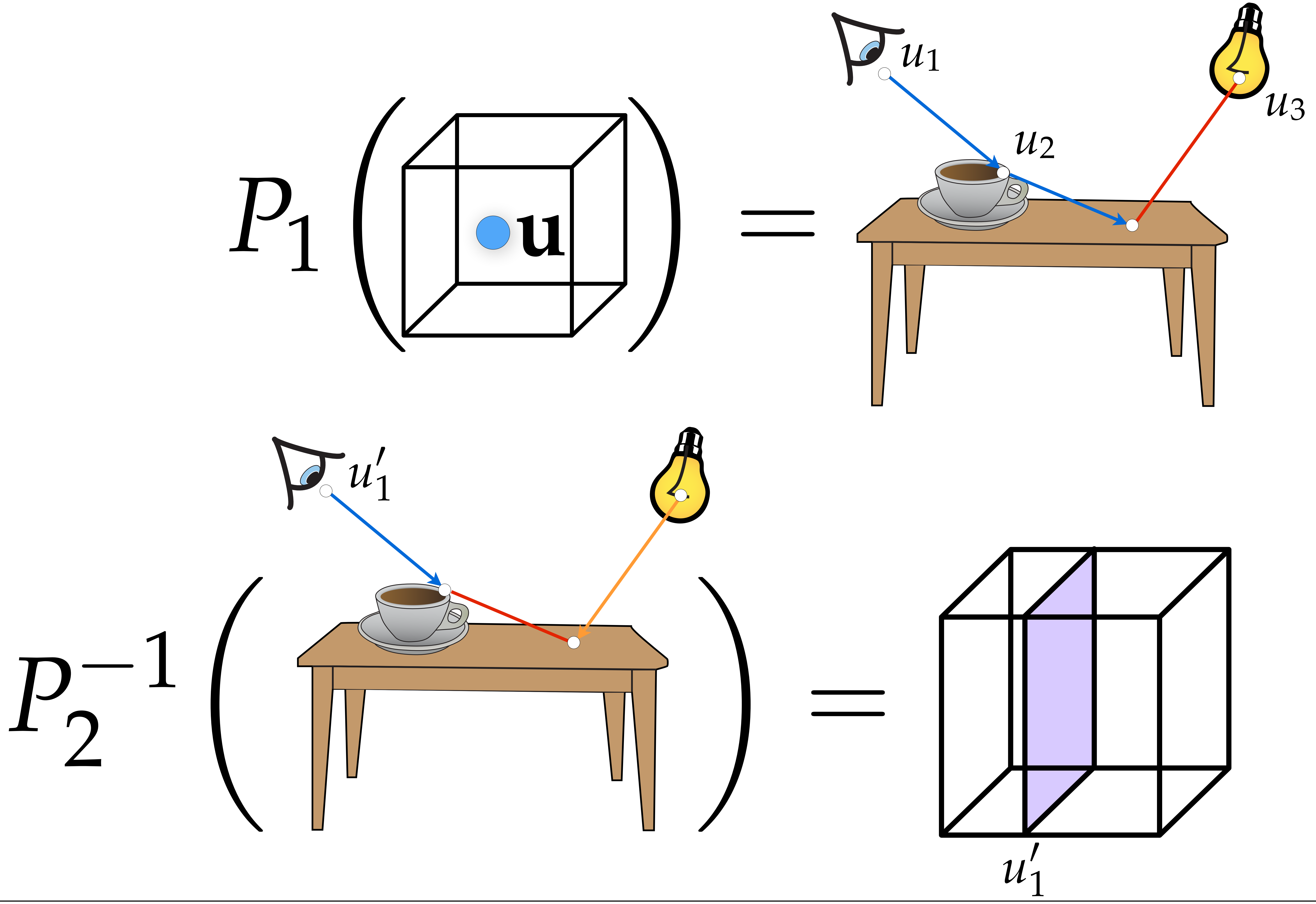
Solution: path inverses



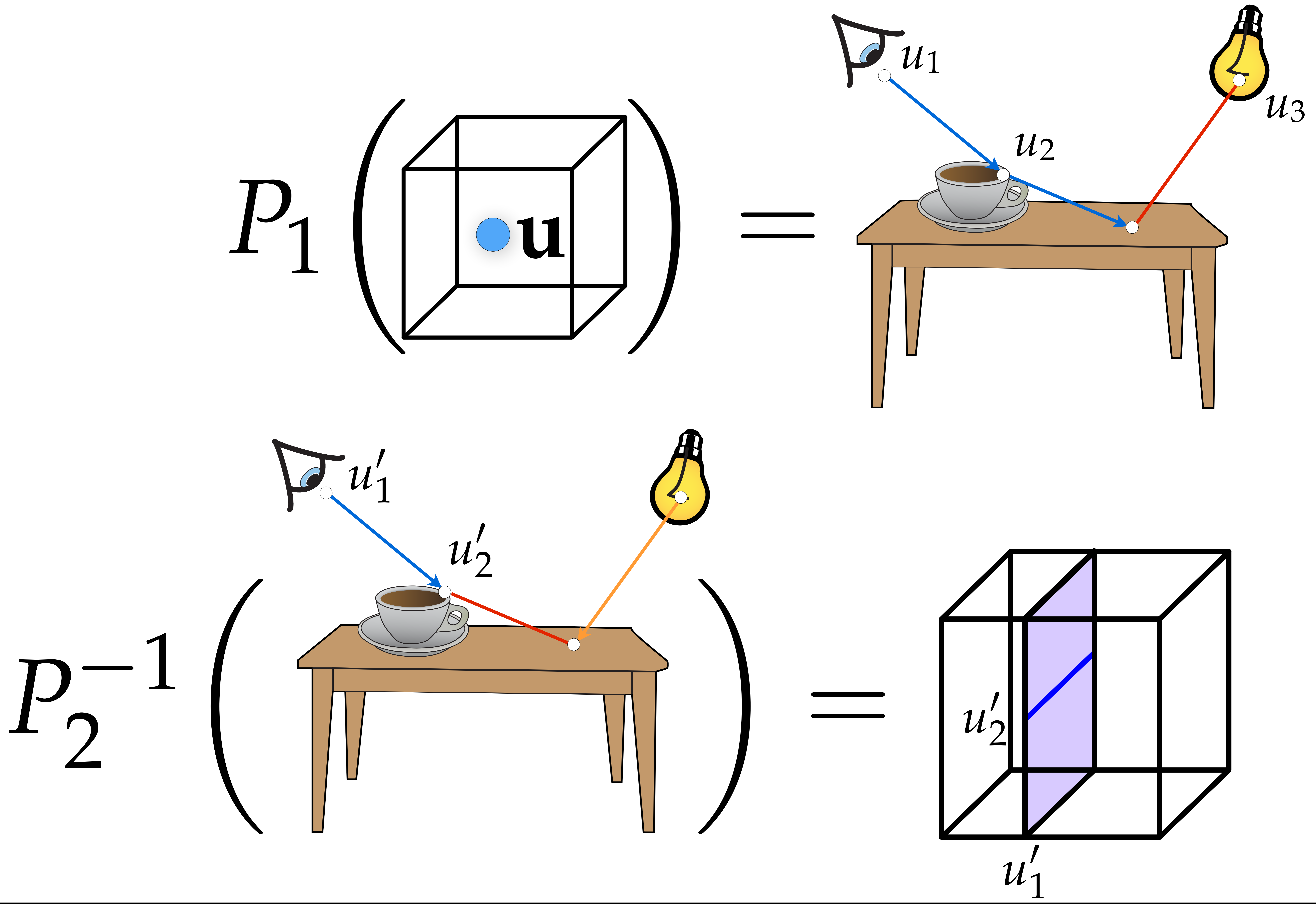
Solution: path inverses



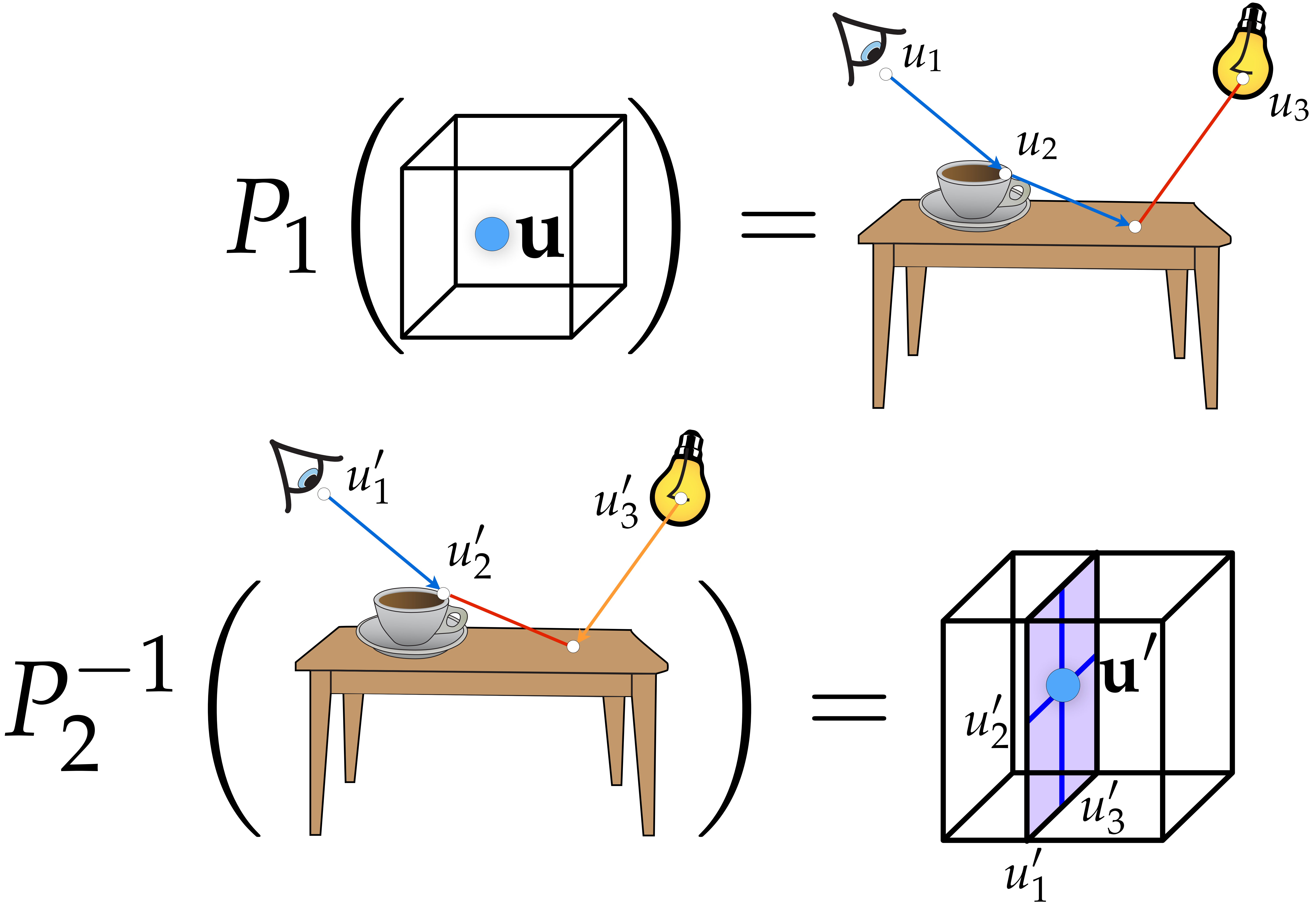
Solution: path inverses



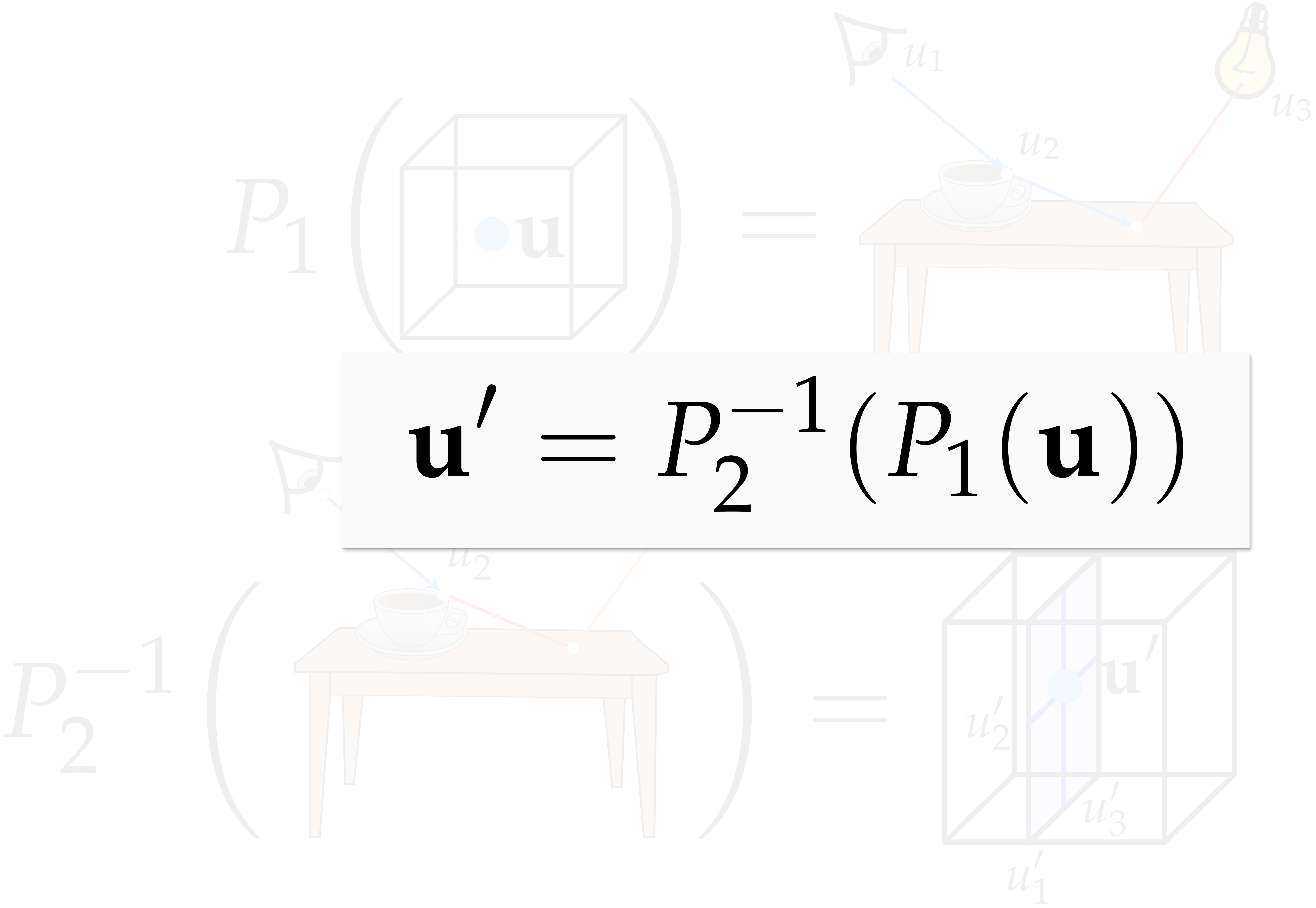
Solution: path inverses



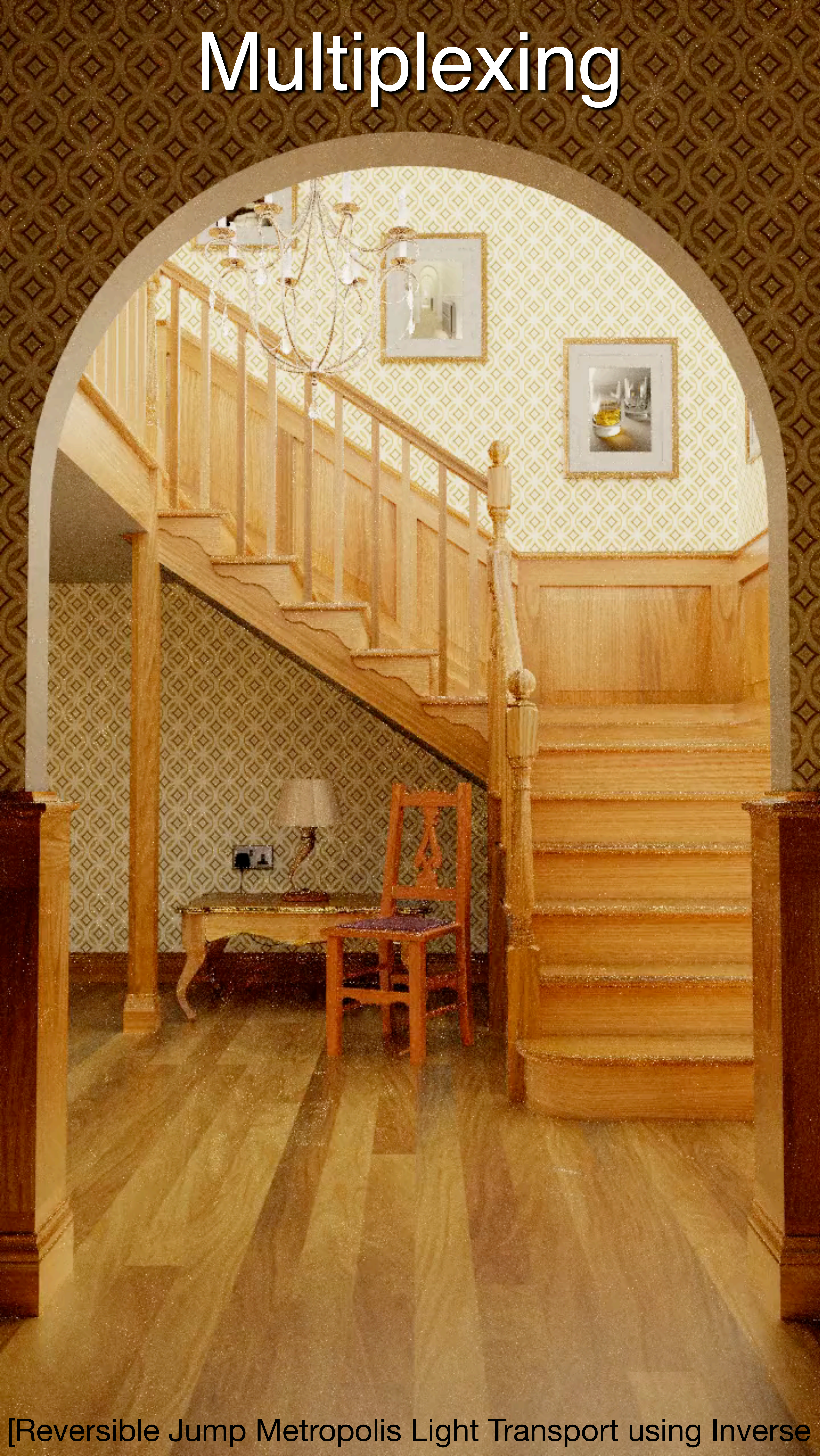
Solution: path inverses



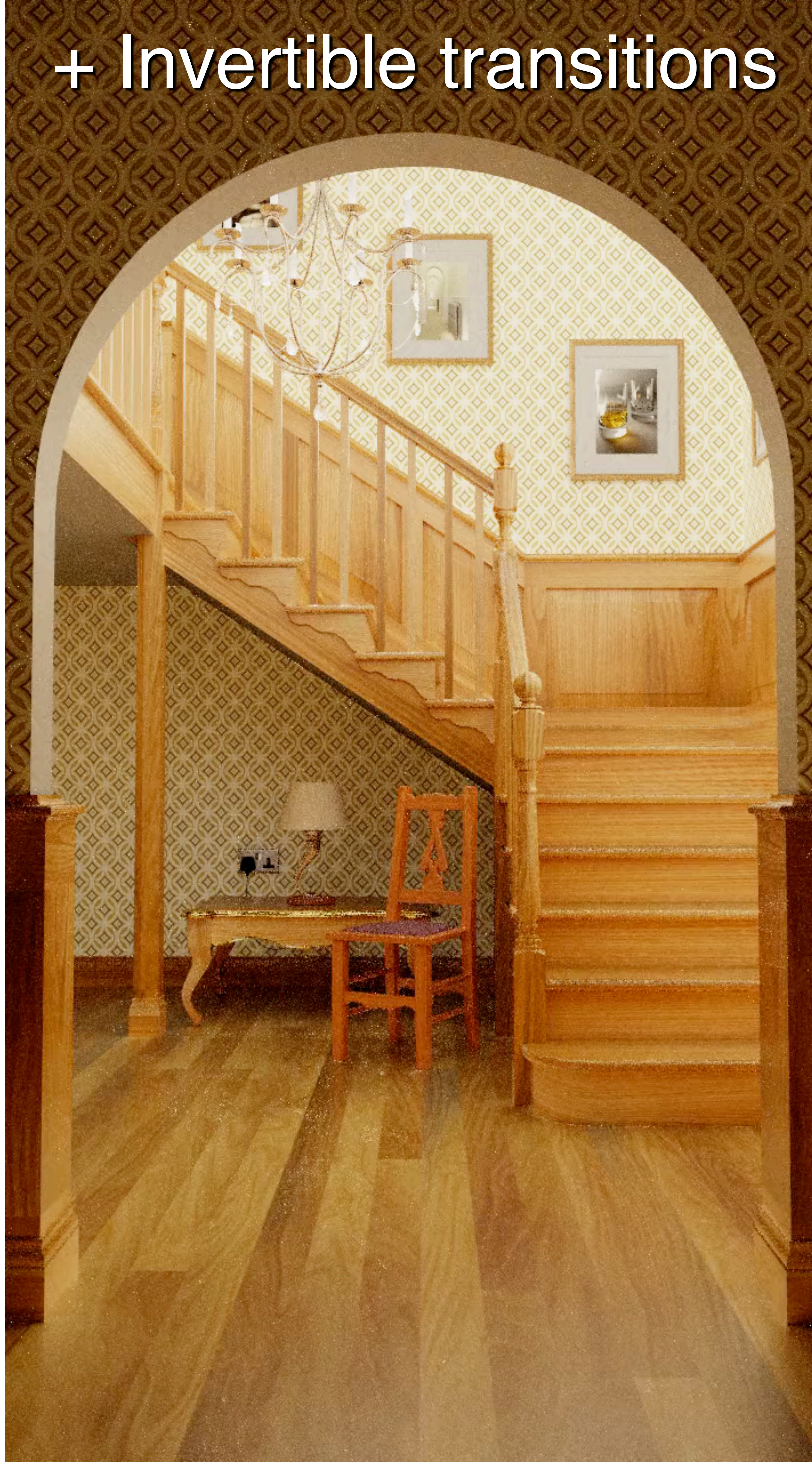
Solution: path inverses



$$\mathbf{u}' = P_2^{-1}(P_1(\mathbf{u}))$$

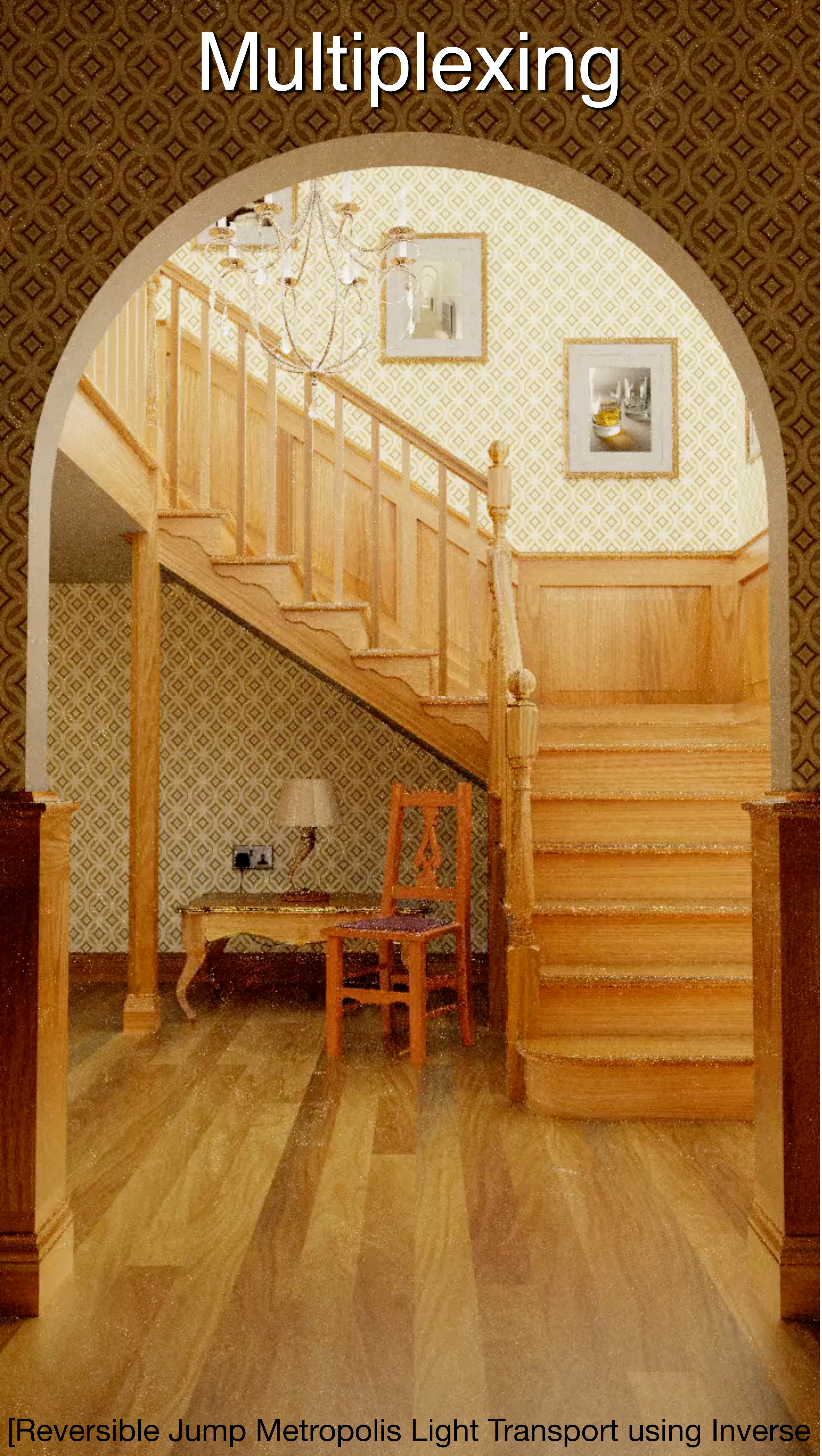


Multiplexing

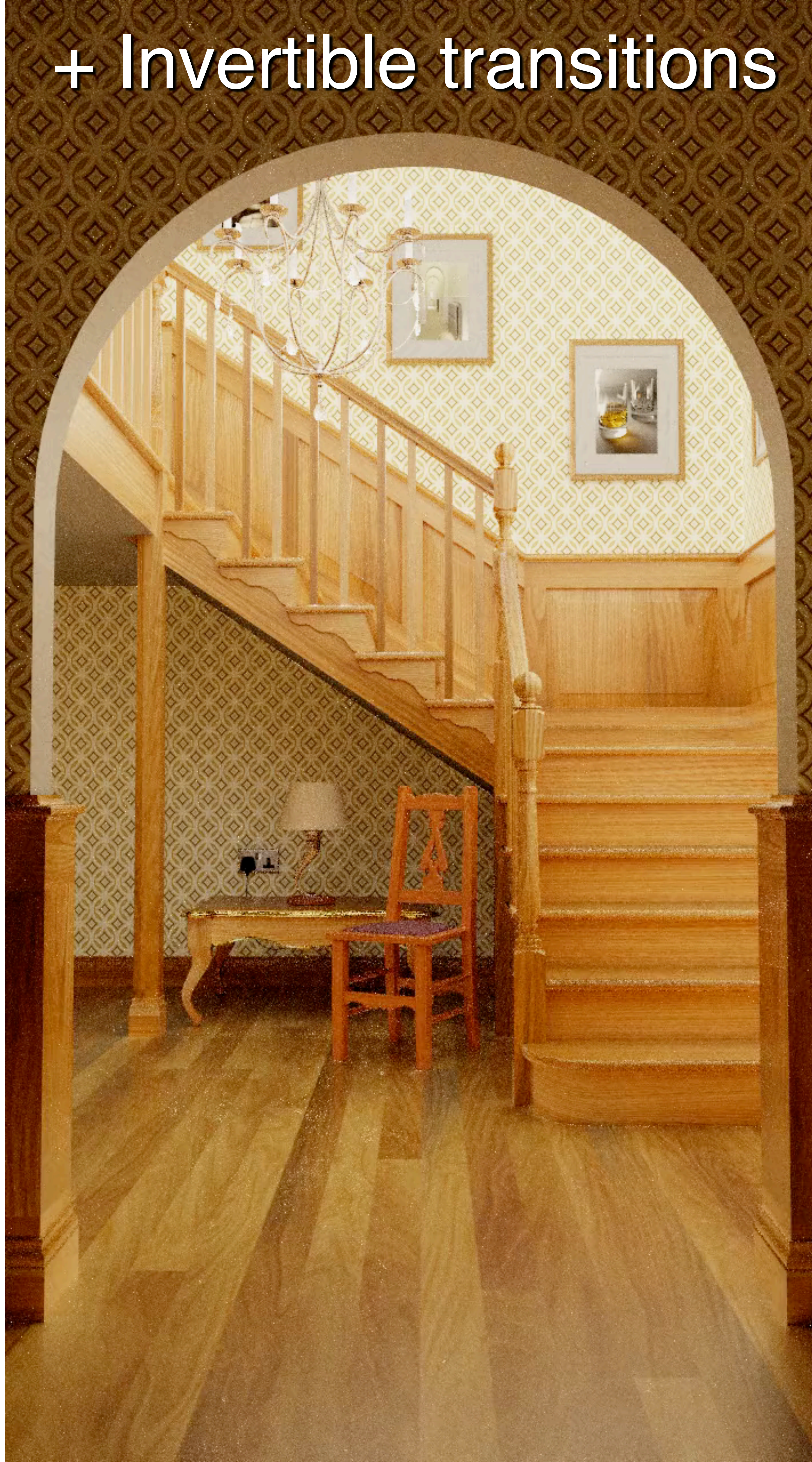


+ Invertible transitions

[Reversible Jump Metropolis Light Transport using Inverse Mappings, Bitterli 2017]

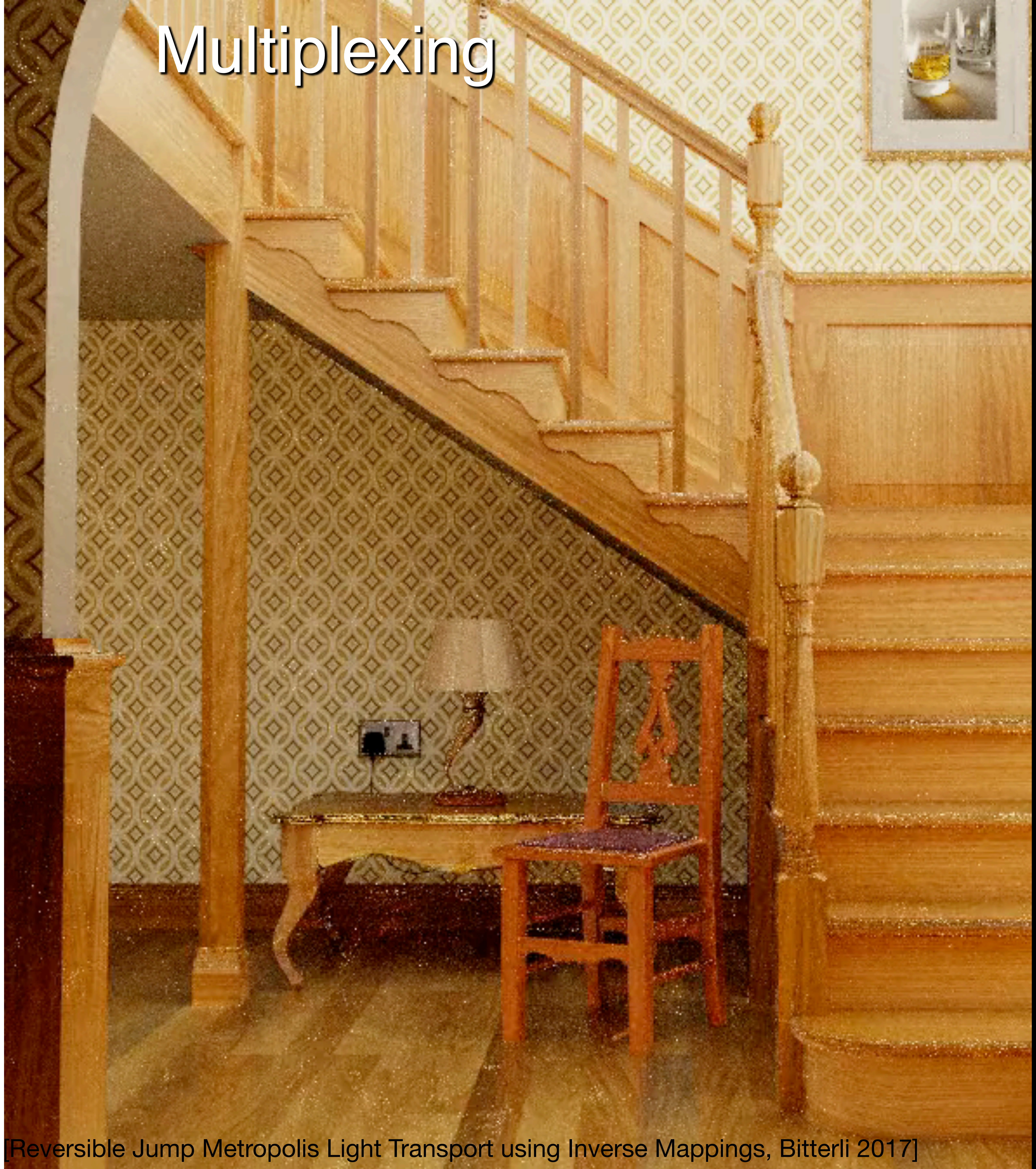


Multiplexing

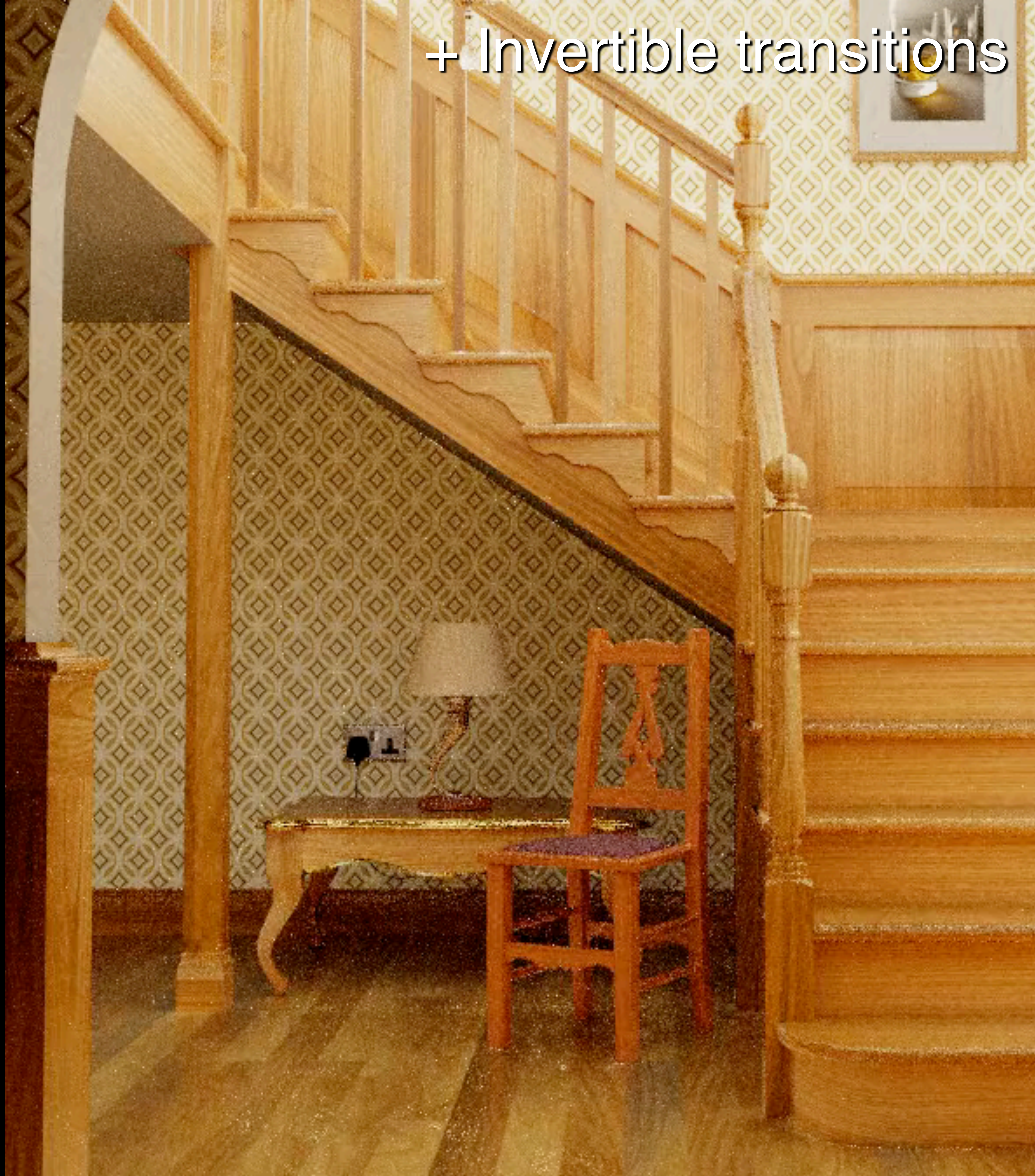


+ Invertible transitions

[Reversible Jump Metropolis Light Transport using Inverse Mappings, Bitterli 2017]



Multiplexing



+ Invertible transitions

[Reversible Jump Metropolis Light Transport using Inverse Mappings, Bitterli 2017]

The original Metropolis Light Transport Algorithm

Metropolis Light Transport
[Veach and Guibas 1997]

Mutation and Perturbation strategies

Bidirectional mutation

Mutation and Perturbation strategies

Bidirectional mutation

Mutation and Perturbation strategies

Bidirectional mutation

Mutation and Perturbation strategies

Bidirectional mutation

Mutation and Perturbation strategies

Bidirectional mutation

Mutation and Perturbation strategies

Bidirectional mutation

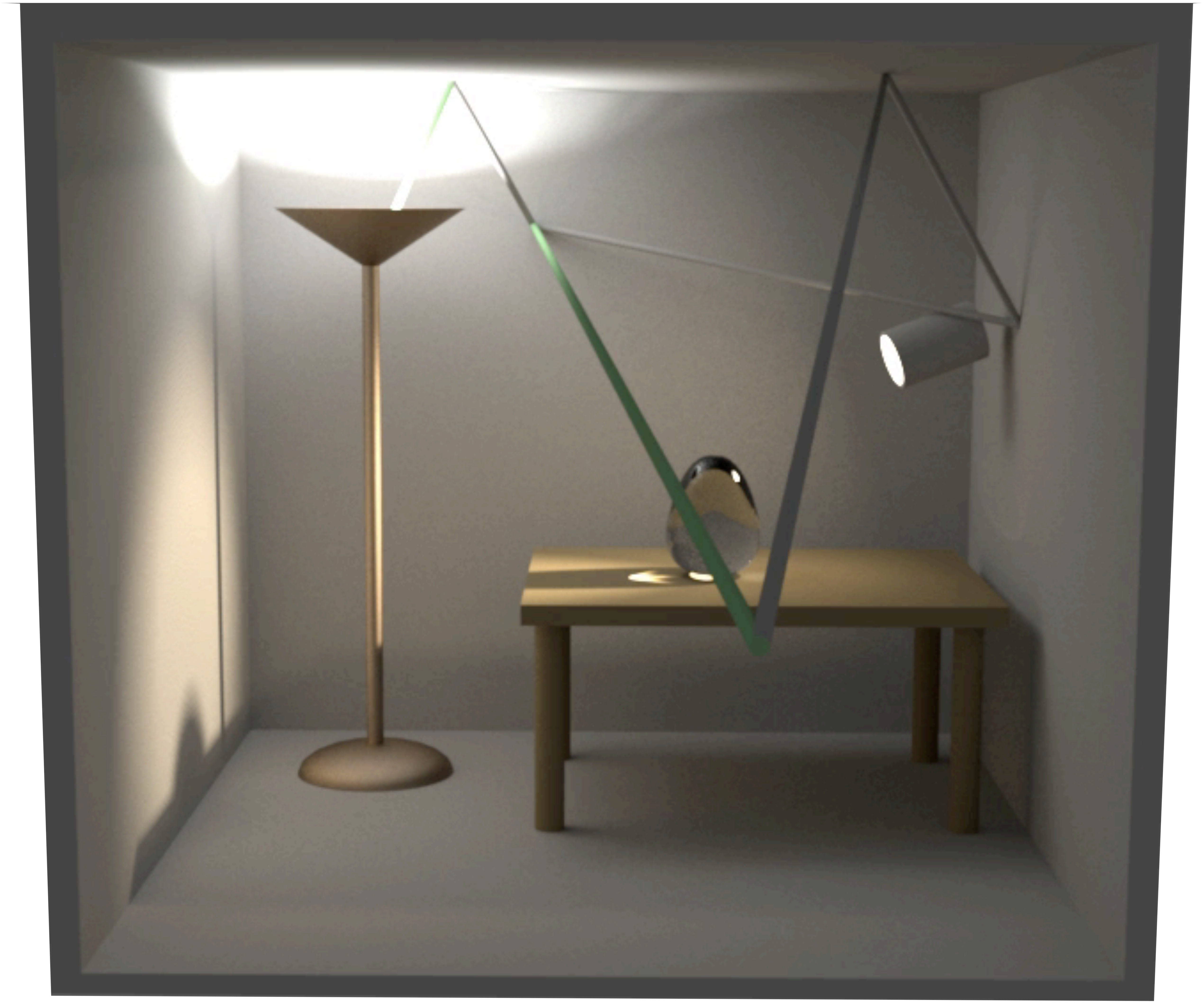
Caustic perturbation

Mutation and Perturbation strategies

Bidirectional mutation

Caustic perturbation

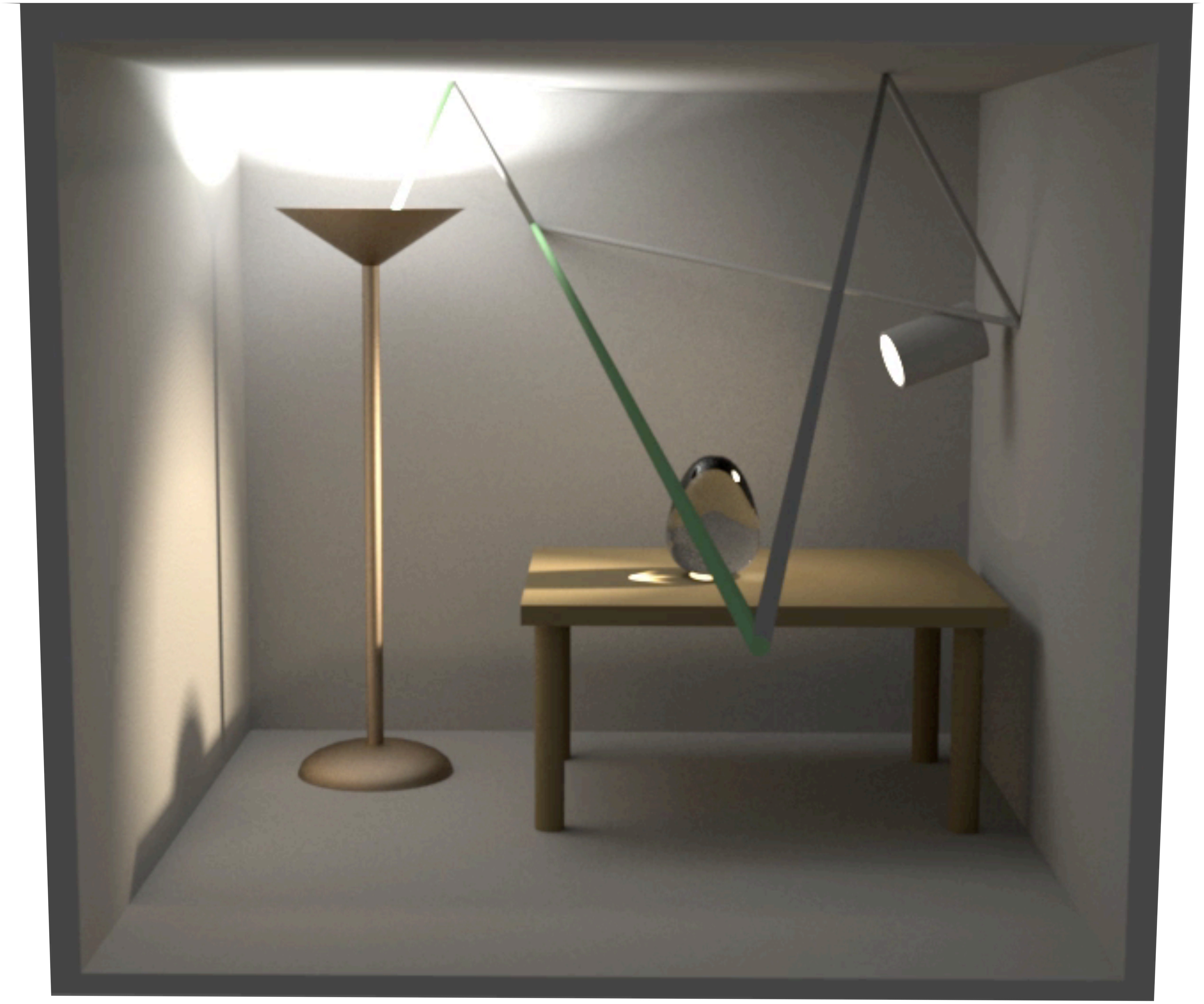
Light path visualization



gray = proposal state

green = current state

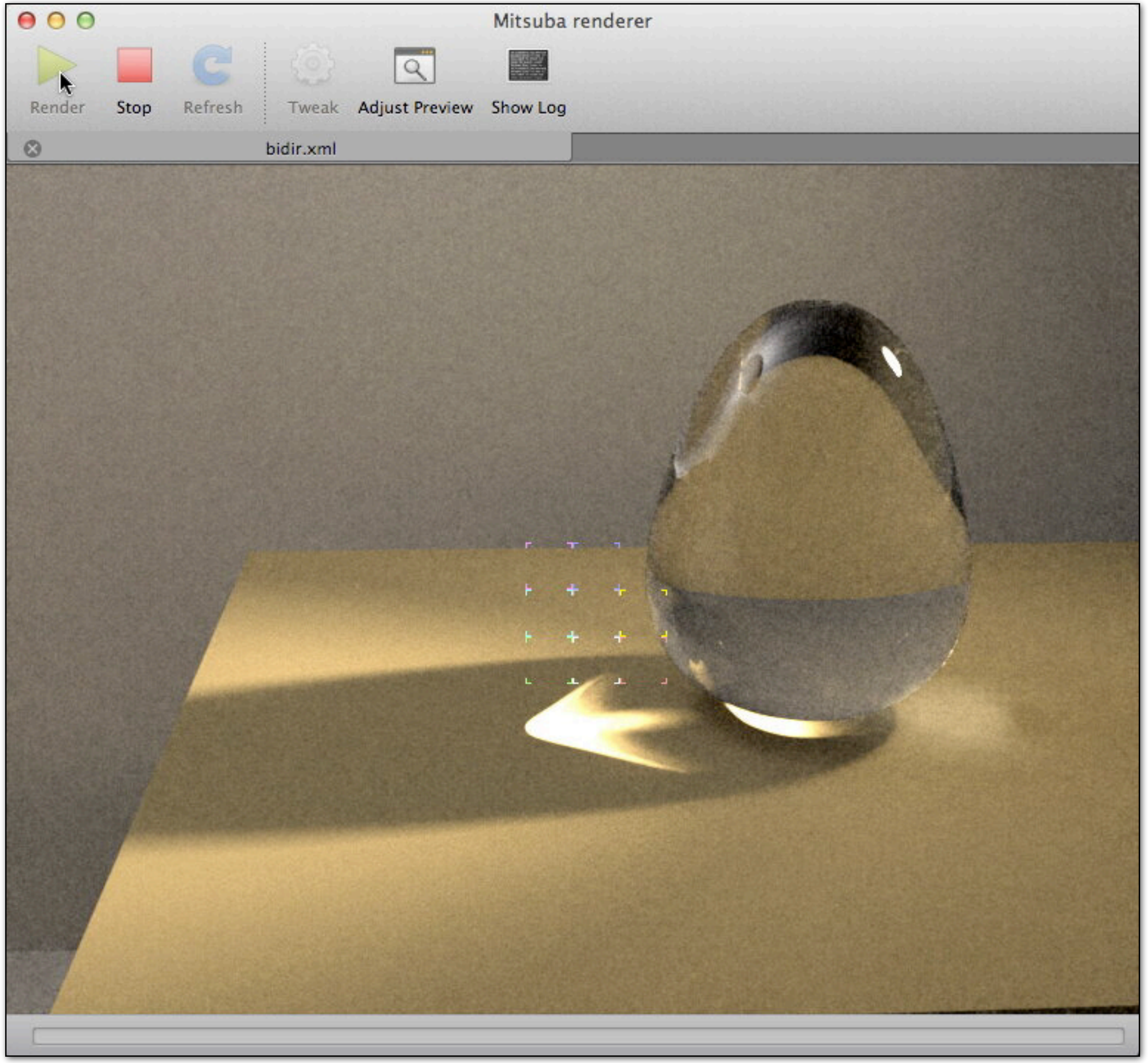
Light path visualization



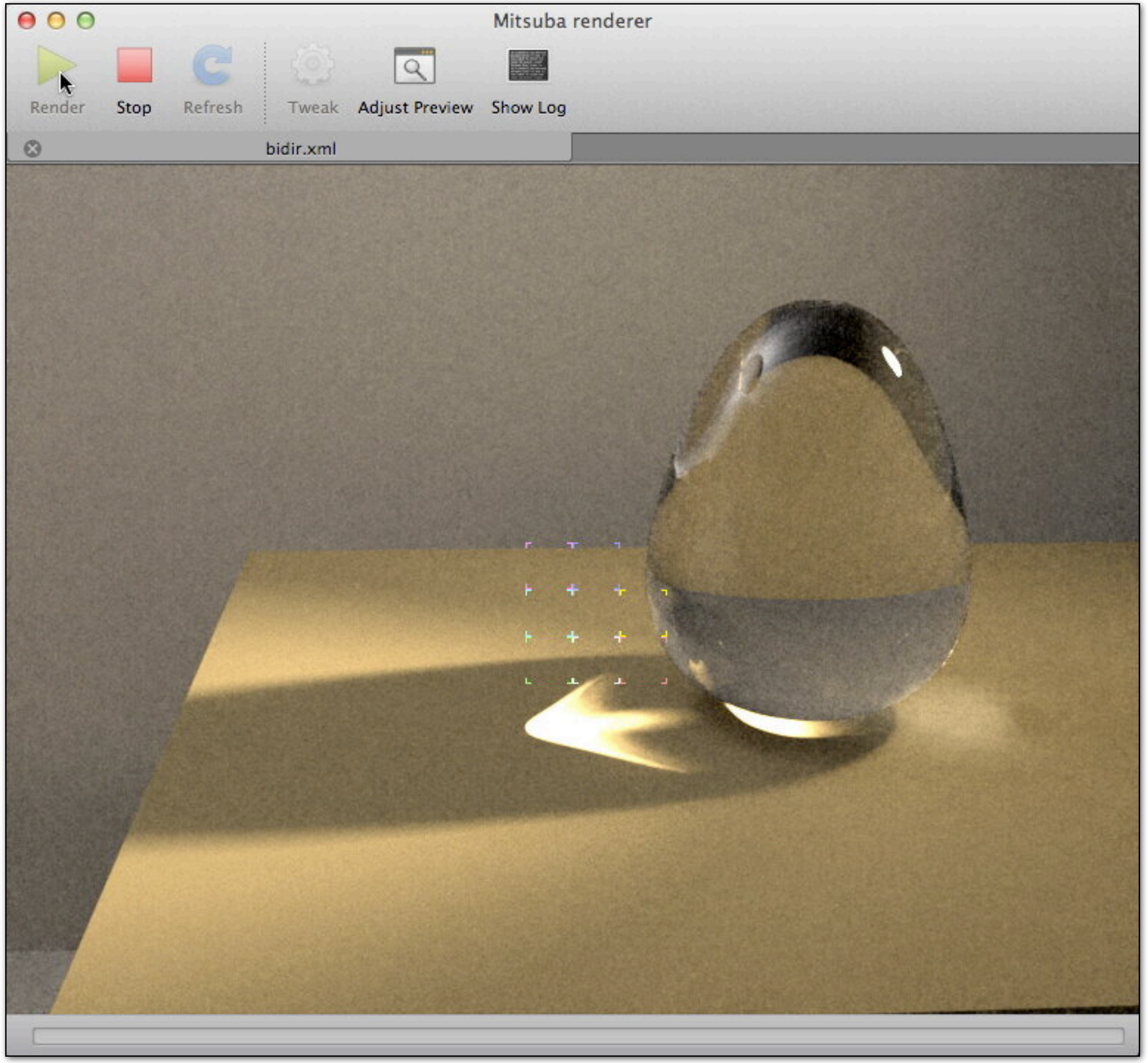
gray = proposal state

green = current state

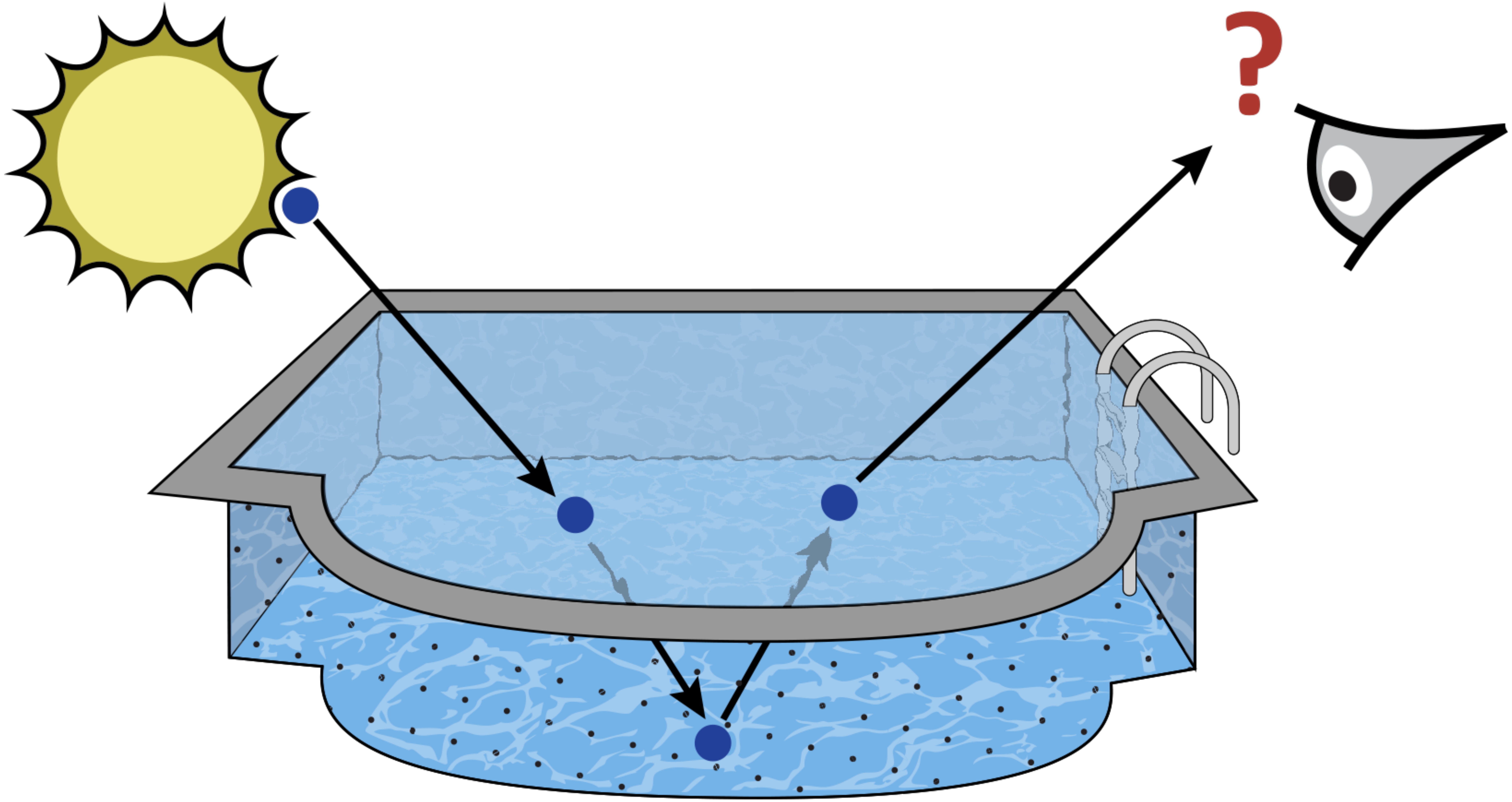
Visualization, and issues with this method



Visualization, and issues with this method

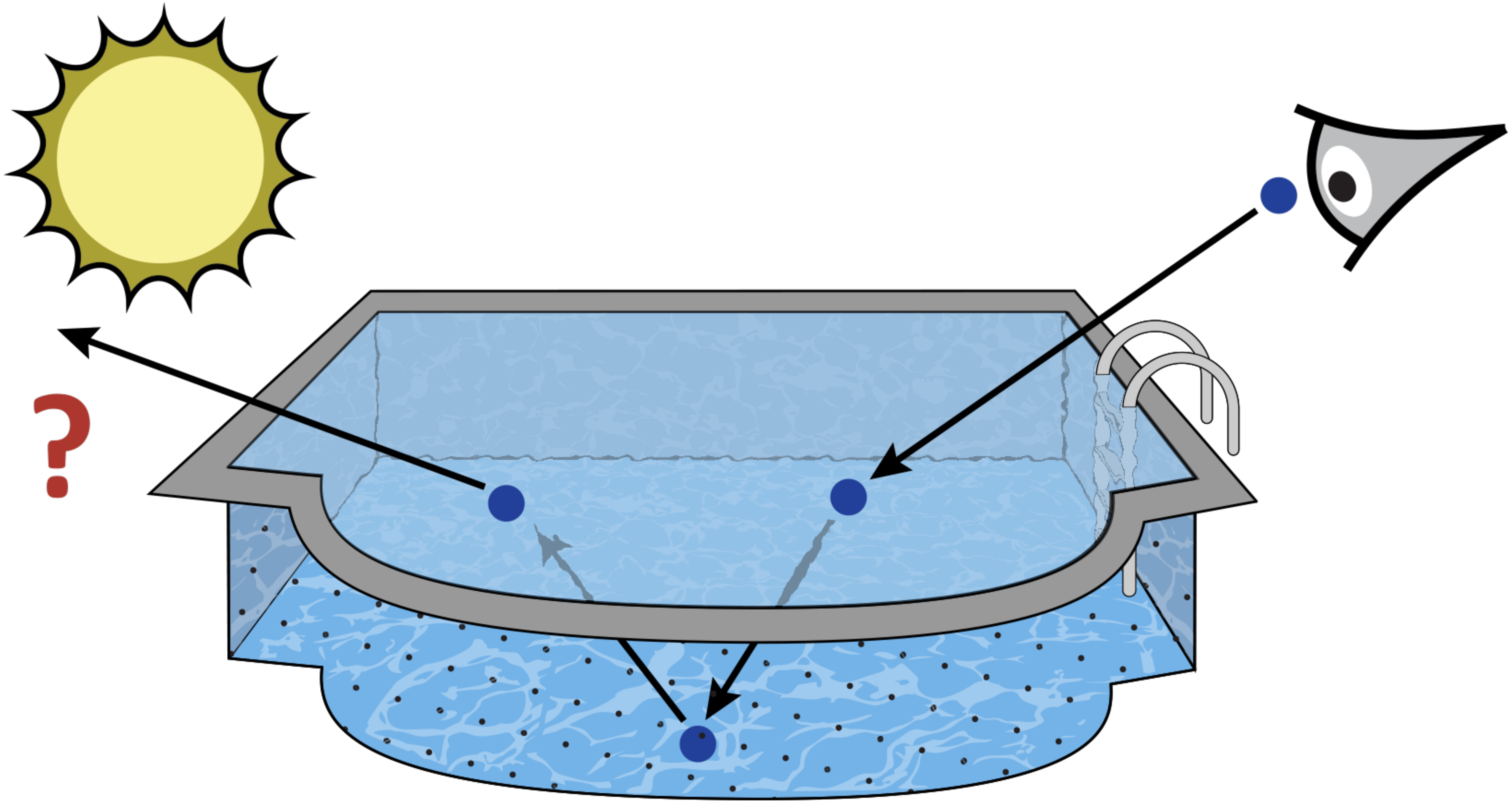


Specular paths



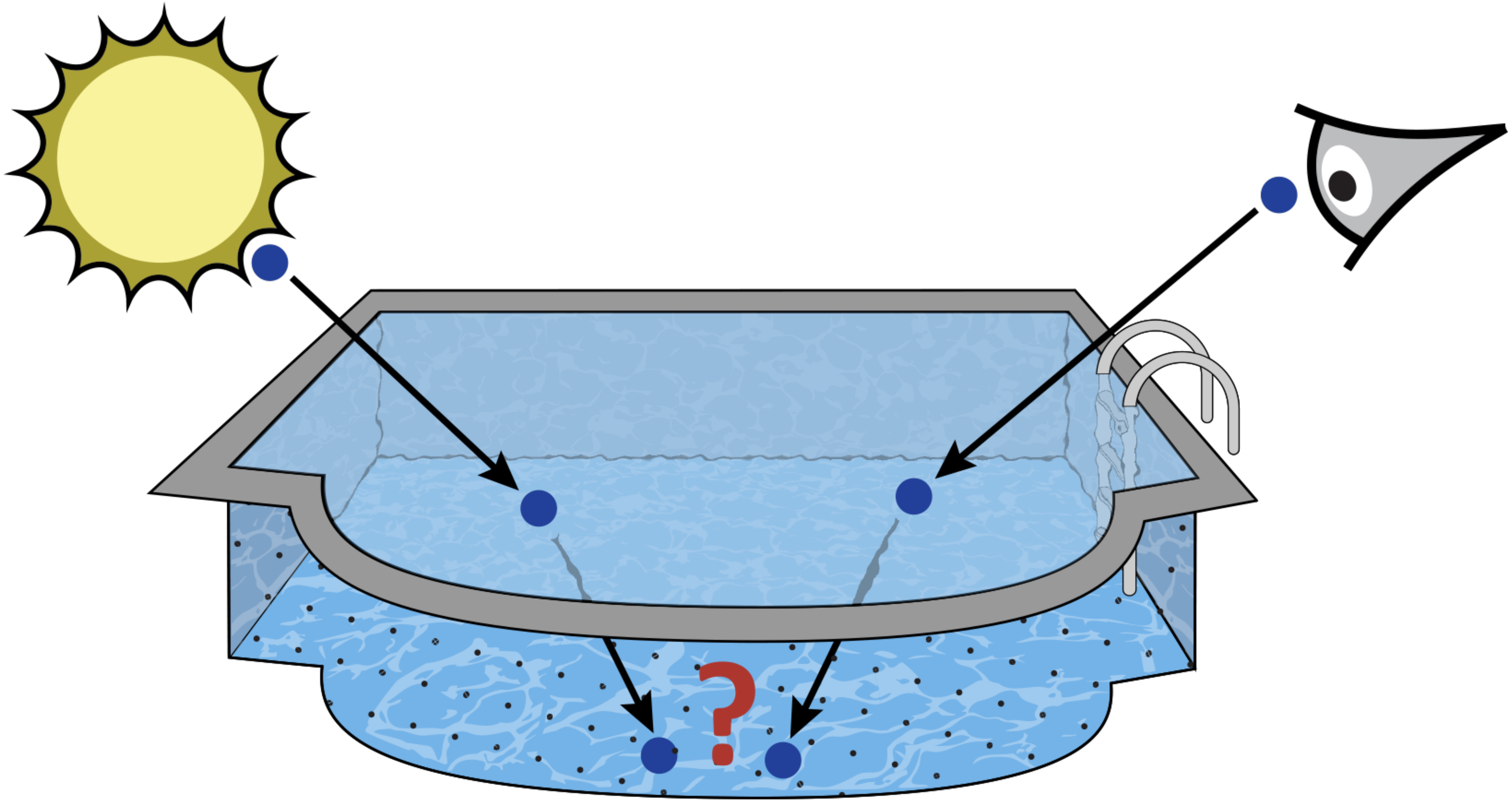
Path tracing from
the light source

Specular paths



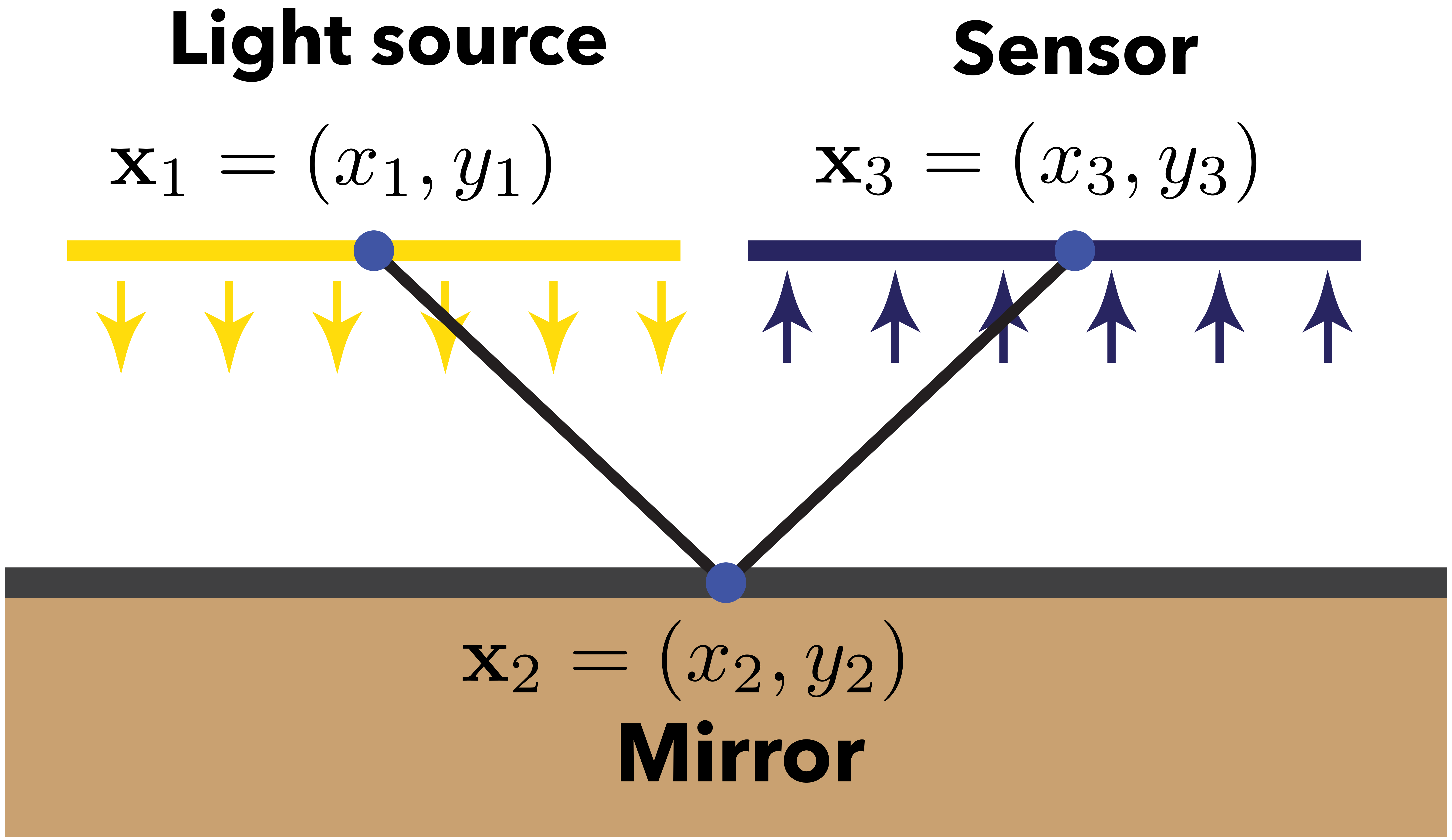
Path tracing from
the camera

Specular paths

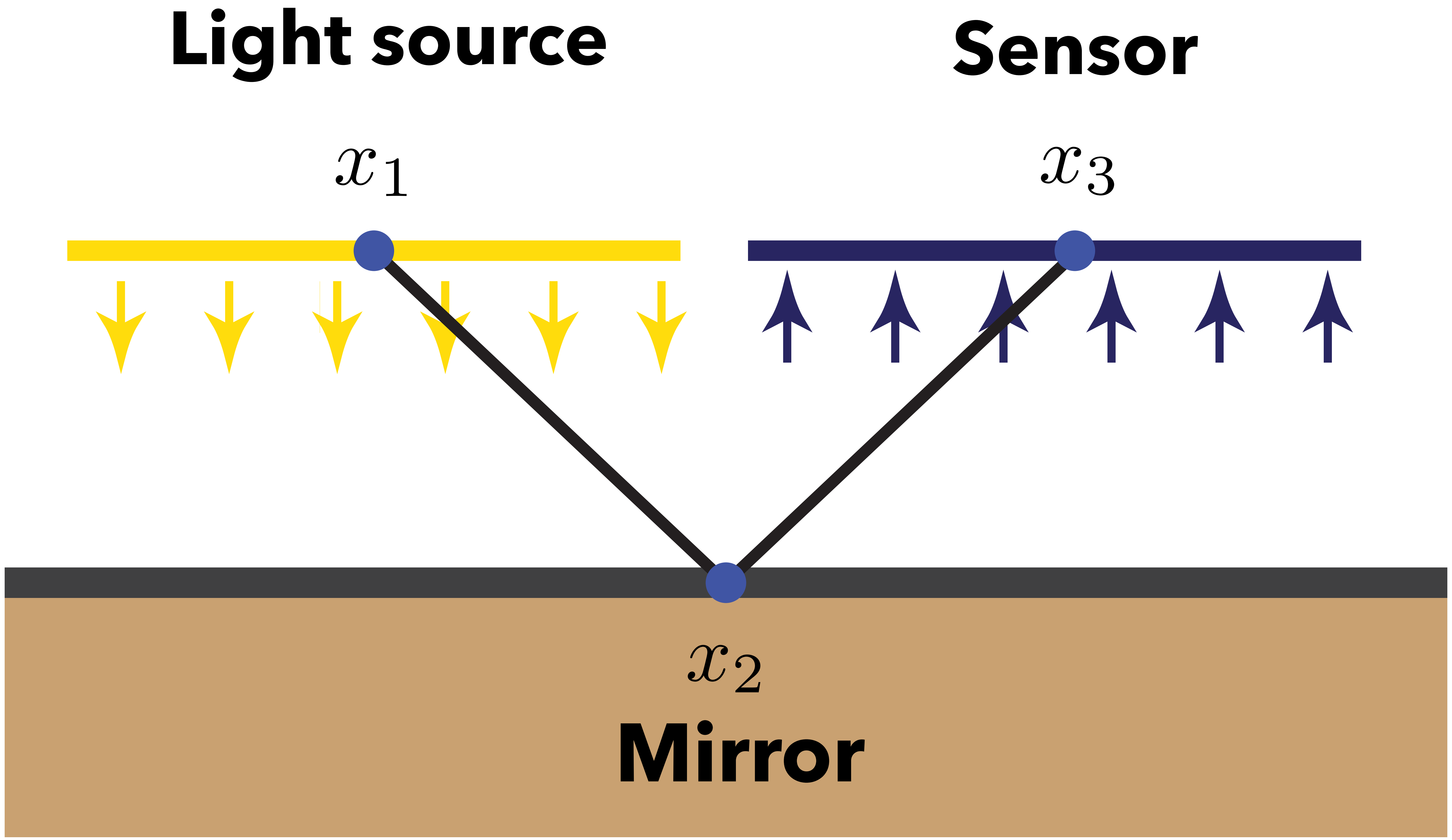


Bidirectional path tracing

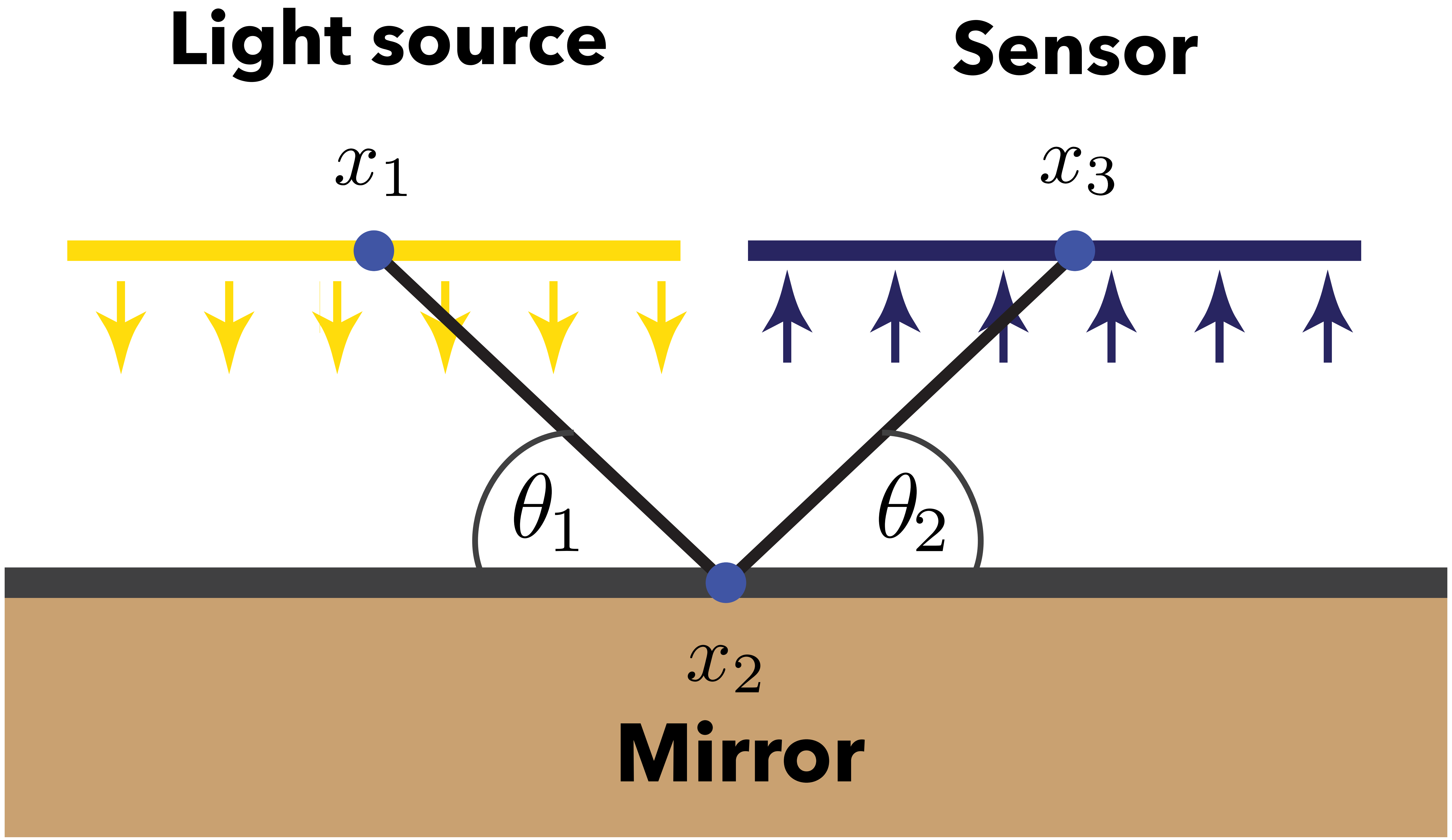
An observation in flatland


Light source
$\mathbf{x}_1 = (x_1, y_1)$
Sensor
$\mathbf{x}_3 = (x_3, y_3)$
$\mathbf{x}_2 = (x_2, y_2)$
Mirror

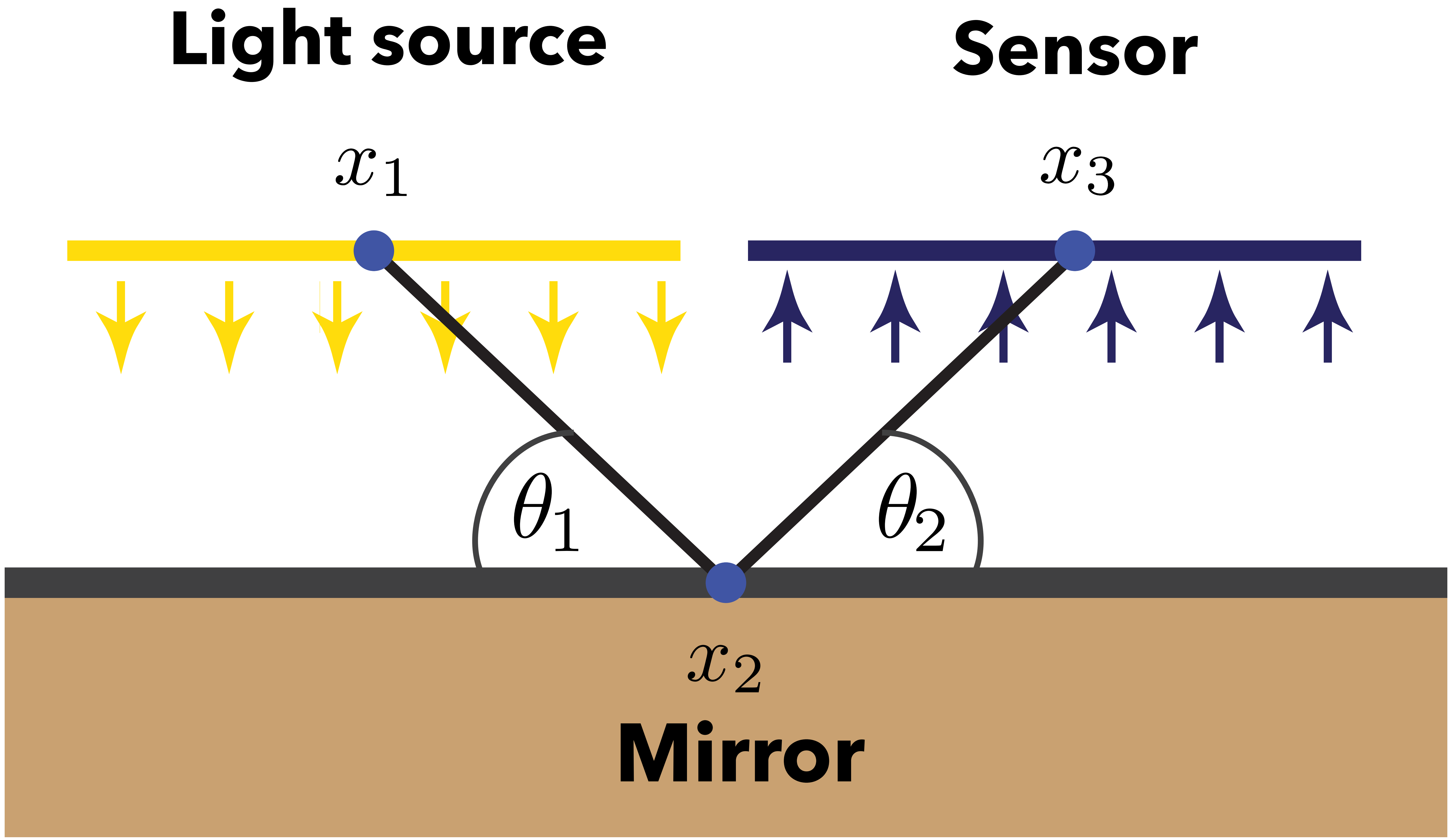
An observation in flatland



An observation in flatland

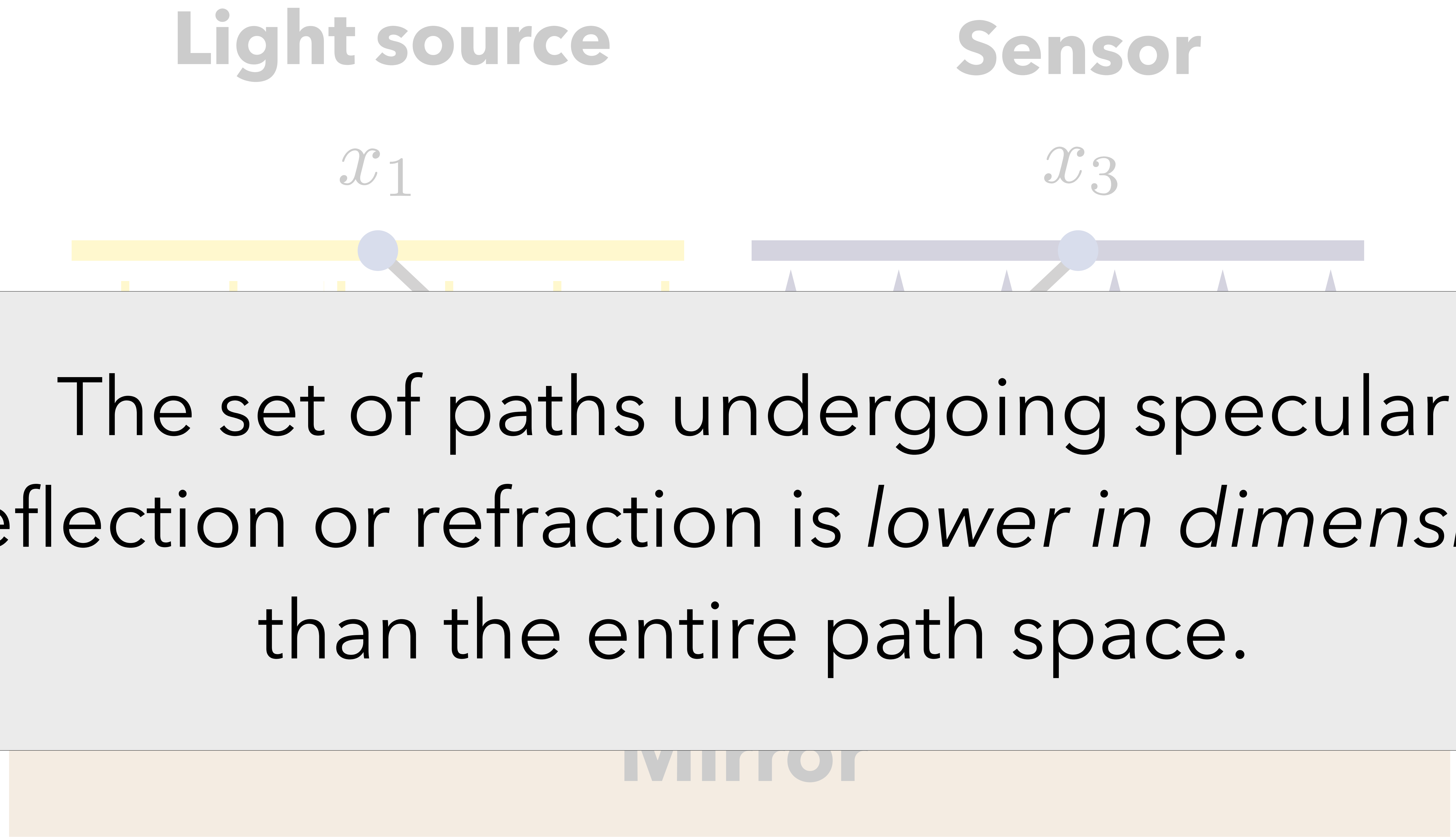


An observation in flatland



$$x_2 = \frac{1}{2}\left(x_1 + x_3\right)$$

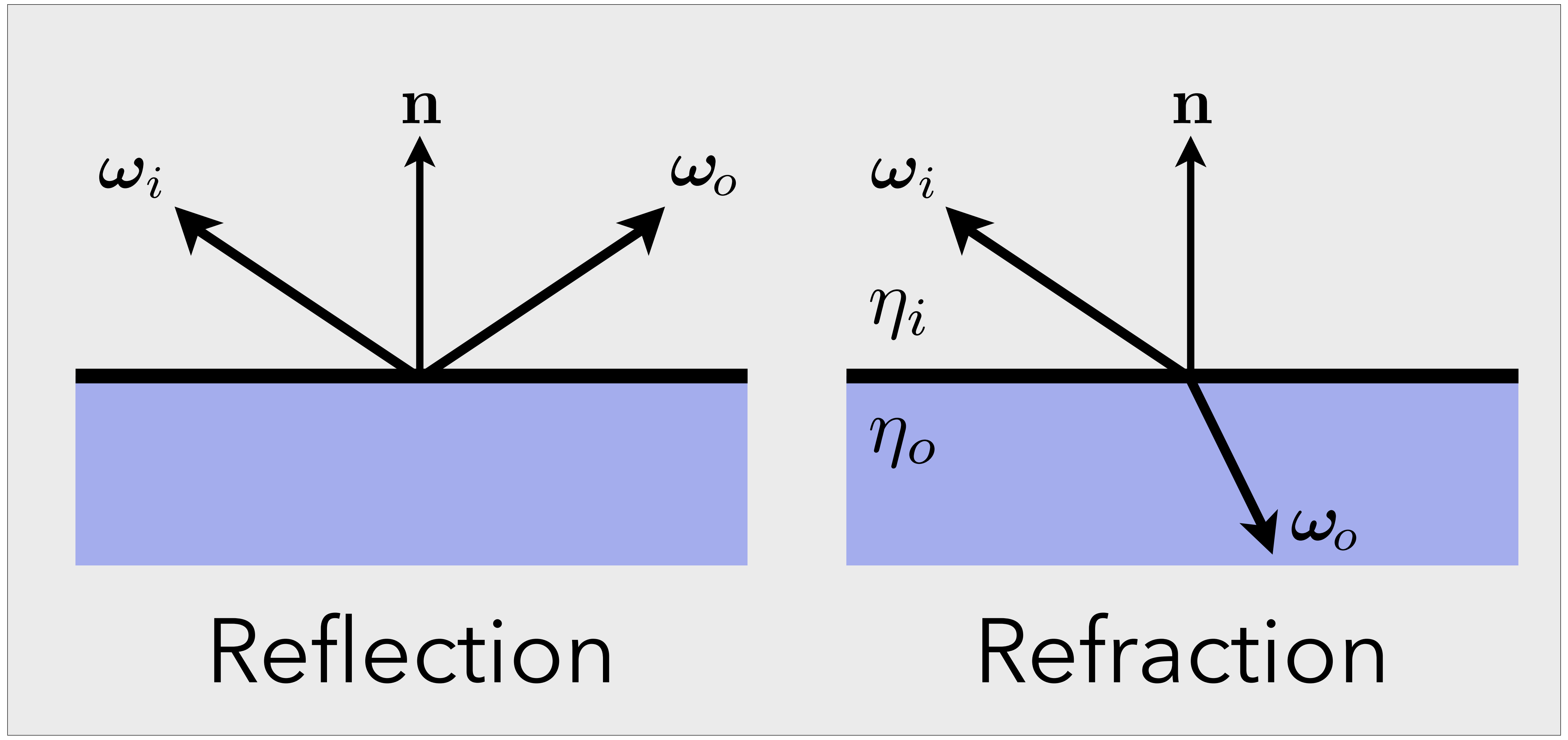
An observation in flatland



The set of paths undergoing specular reflection or refraction is *lower in dimension* than the entire path space.

$$x_2 = \frac{1}{2}(x_1 + x_3)$$

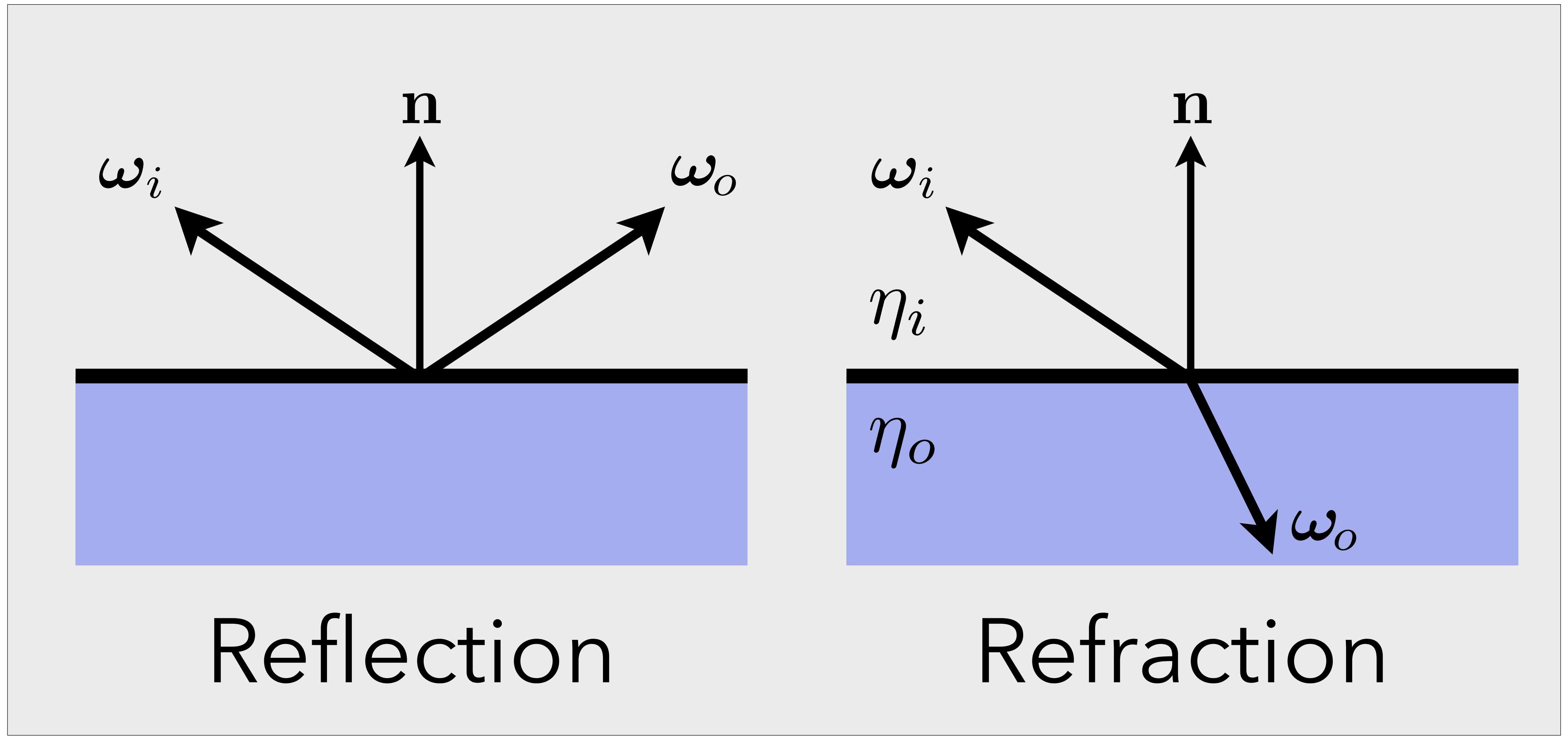
More formally



Express as constraint:

$$c_i(\mathbf{x}_{i-1},\, \mathbf{x}_i,\, \mathbf{x}_{i+1}) = 0$$

More formally



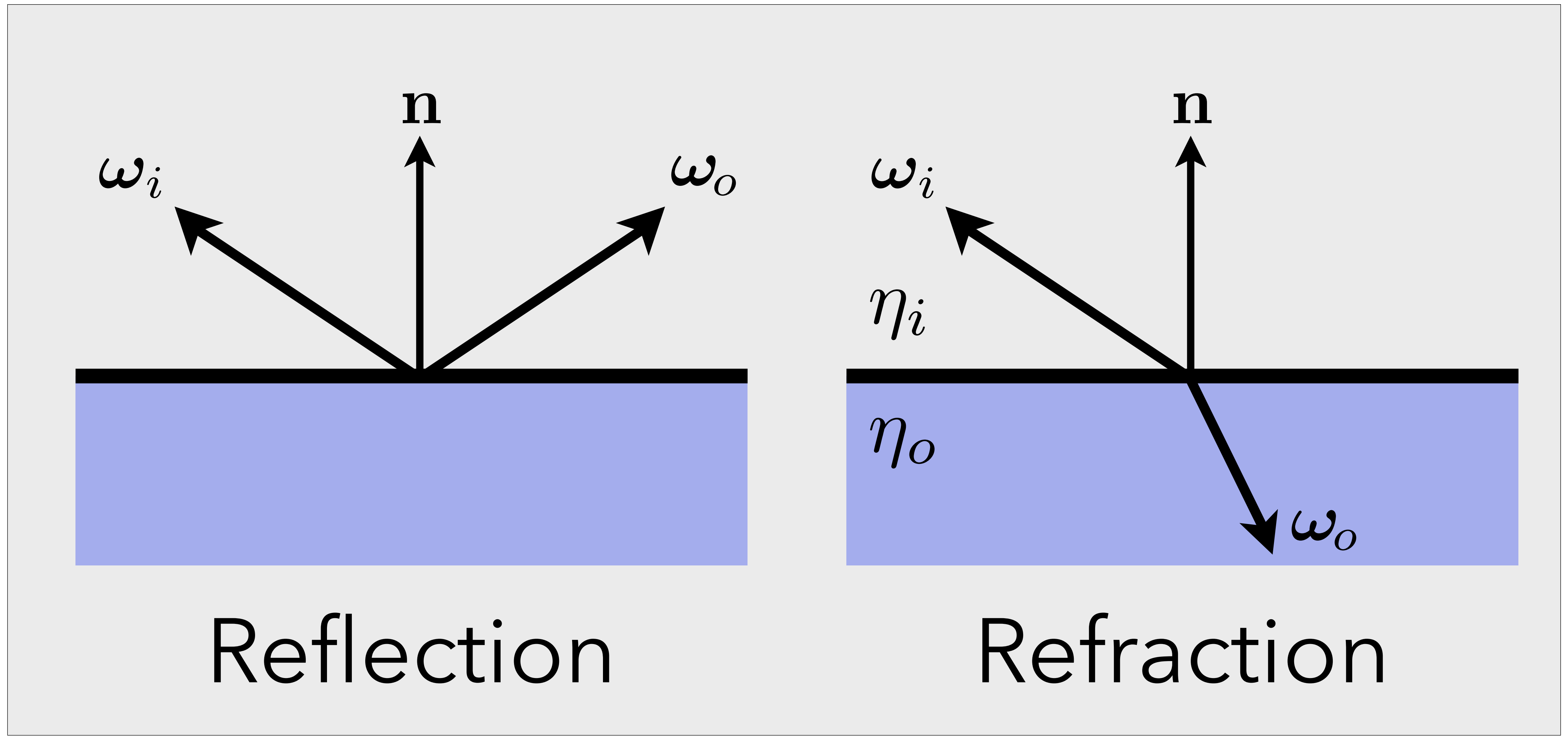
Express as constraint:

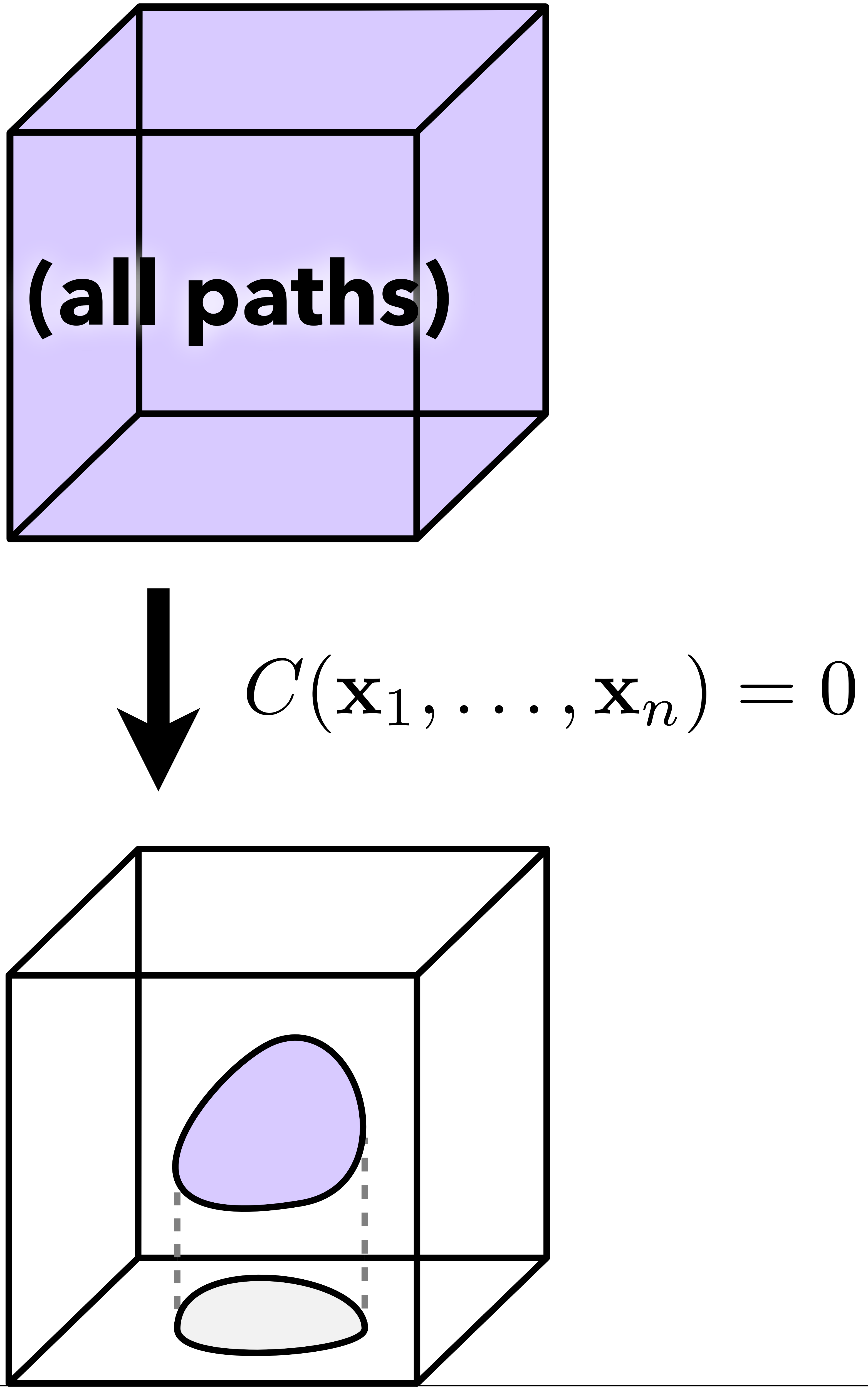
$$c_i\big(\mathbf{x}_{i-1},\ \mathbf{x}_i,\ \mathbf{x}_{i+1}\big) = 0$$

Set satisfying all constraints:

$$\mathcal{S} = \big\{\mathbf{x}_1, \dots, \mathbf{x}_n \in \Omega \mid C(\mathbf{x}_1, \dots, \mathbf{x}_n) = 0\big\}$$

More formally





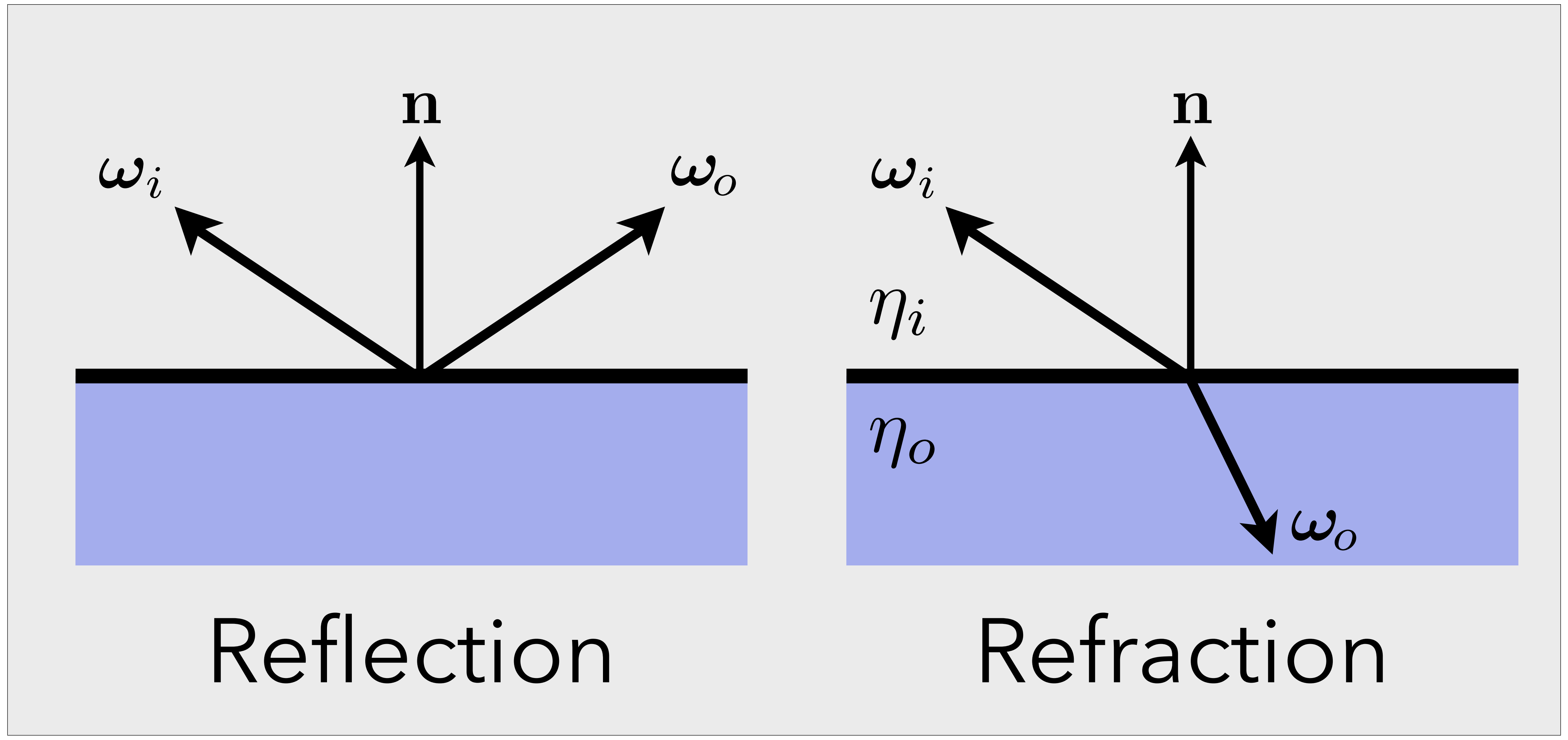
Express as constraint:

$$c_i\big(\mathbf{x}_{i-1},\ \mathbf{x}_i,\ \mathbf{x}_{i+1}\big) = 0$$

Set satisfying all constraints:

$$\mathcal{S} = \left\{\mathbf{x}_1, \ldots, \mathbf{x}_n \in \Omega \mid C(\mathbf{x}_1, \ldots, \mathbf{x}_n) = 0\right\}$$

More formally

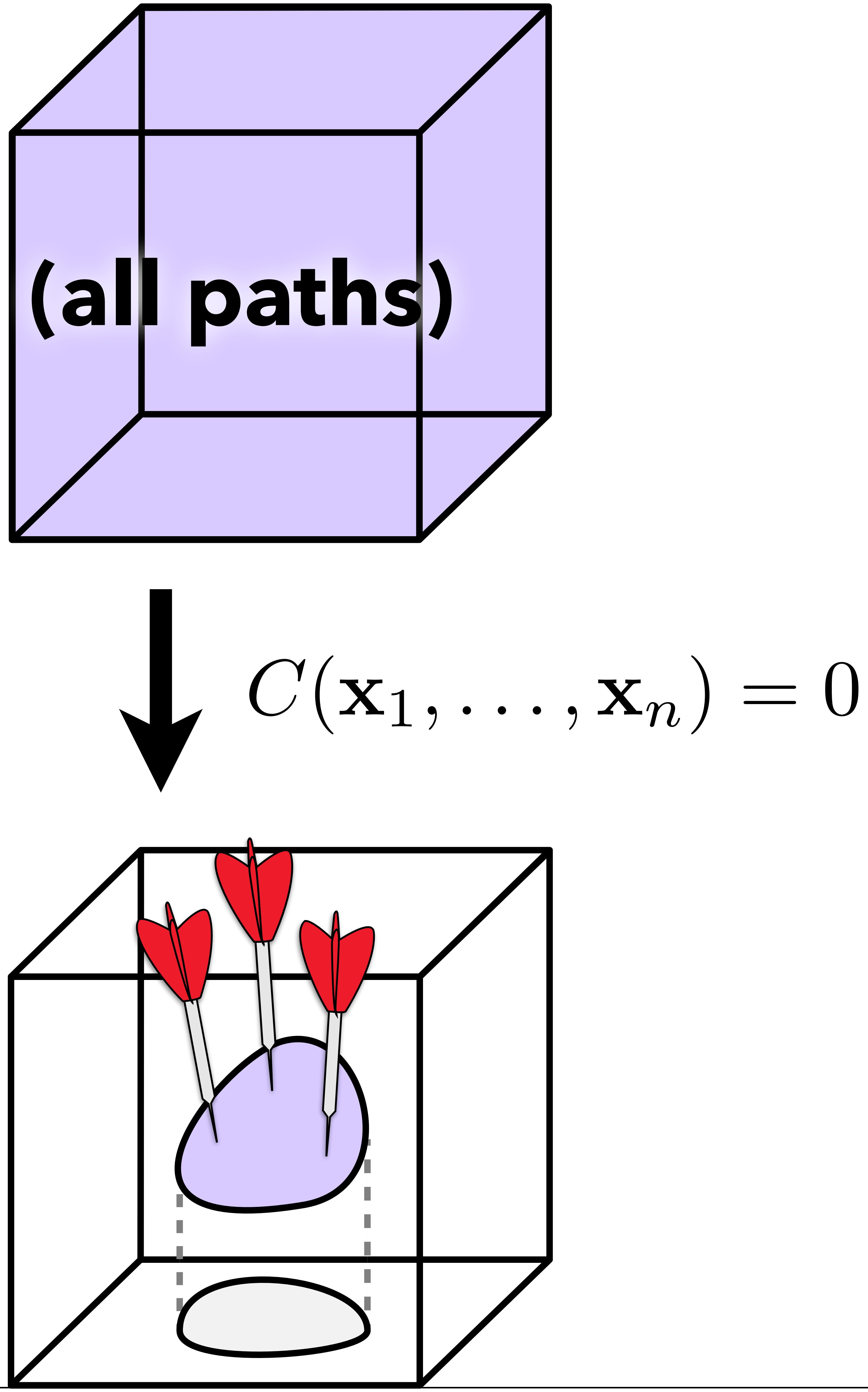


Express as constraint:

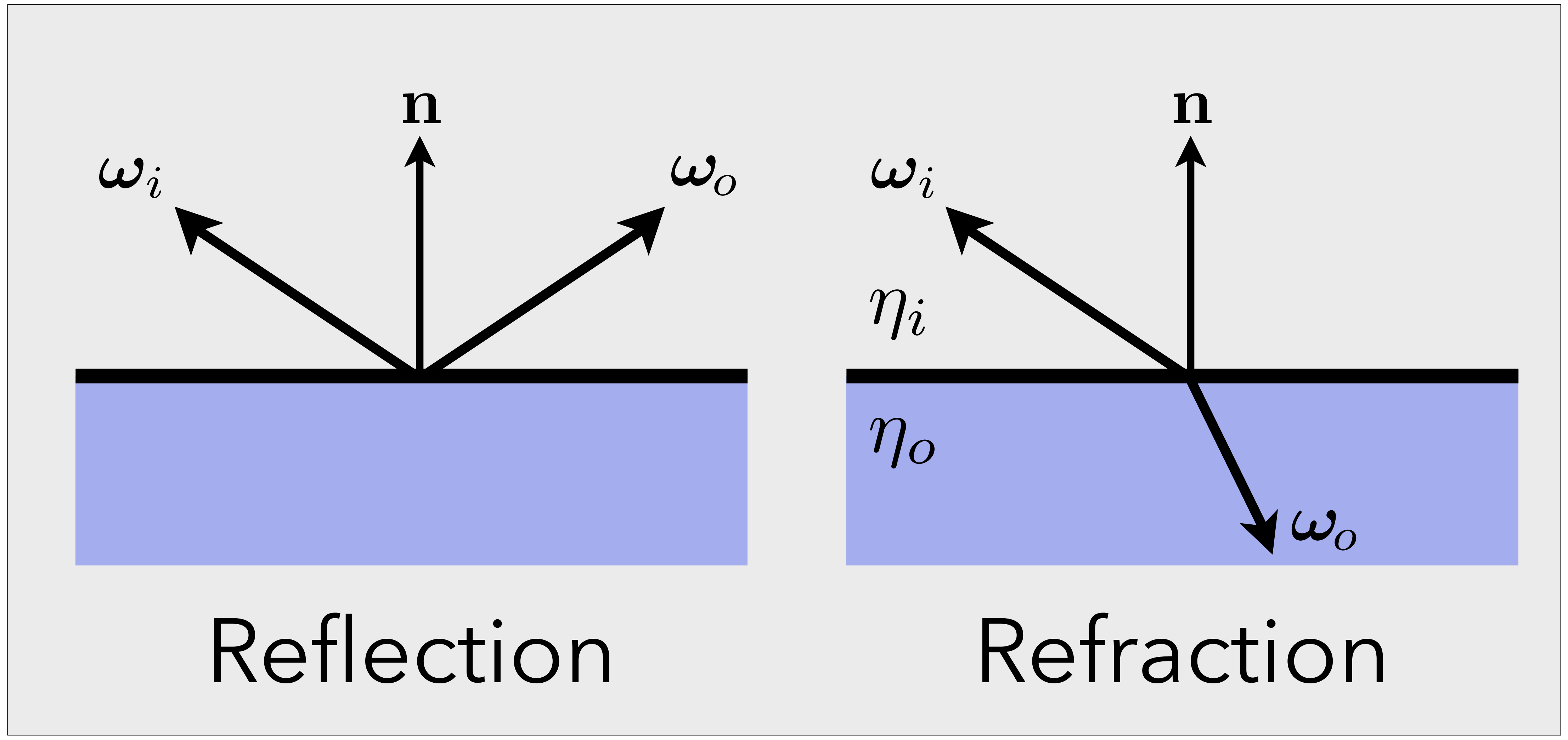
$$c_i(\mathbf{x}_{i-1},\ \mathbf{x}_i,\ \mathbf{x}_{i+1}) = 0$$

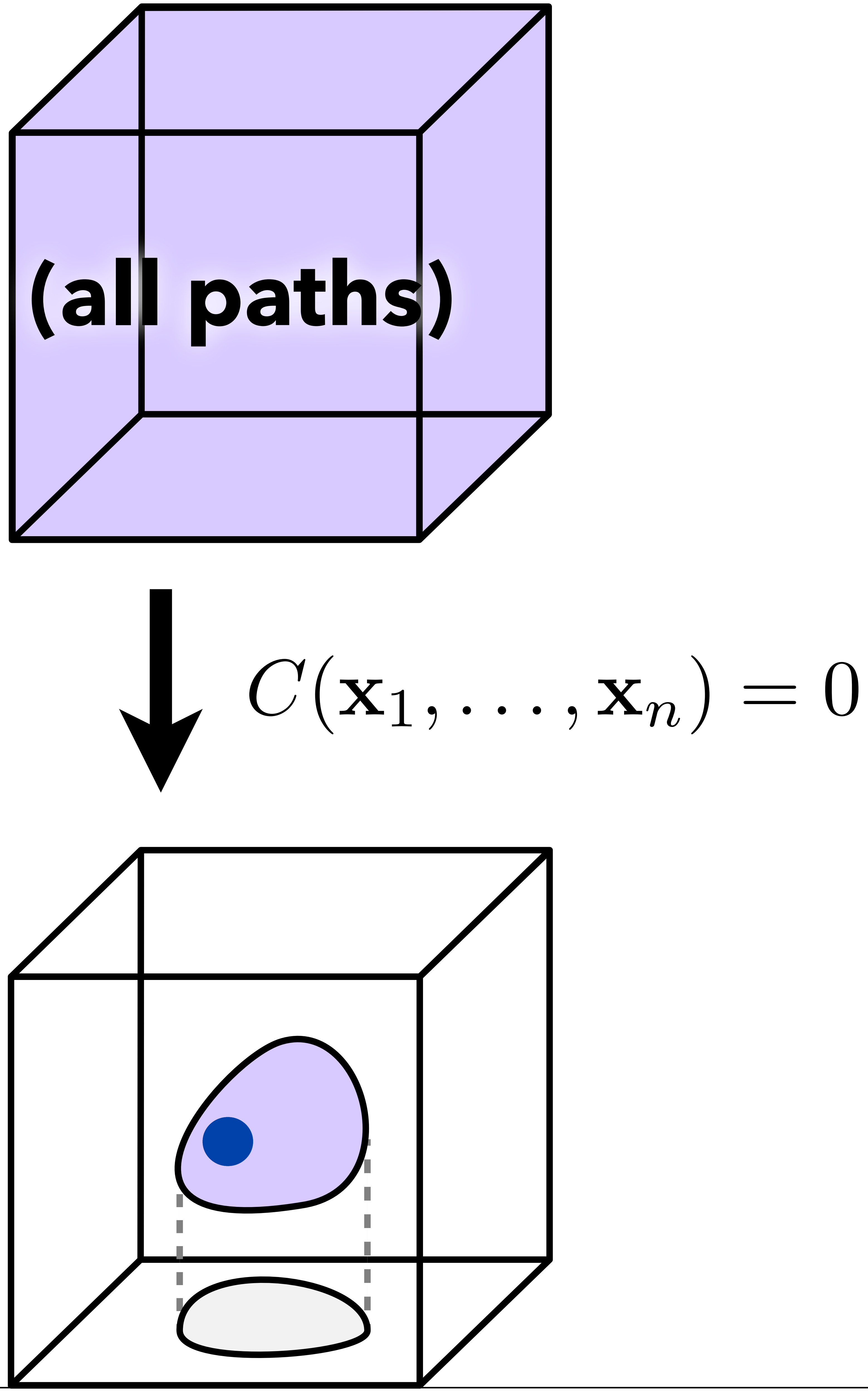
Set satisfying all constraints:

$$\mathcal{S} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n \in \Omega \mid C(\mathbf{x}_1, \ldots, \mathbf{x}_n) = 0\}$$



More formally





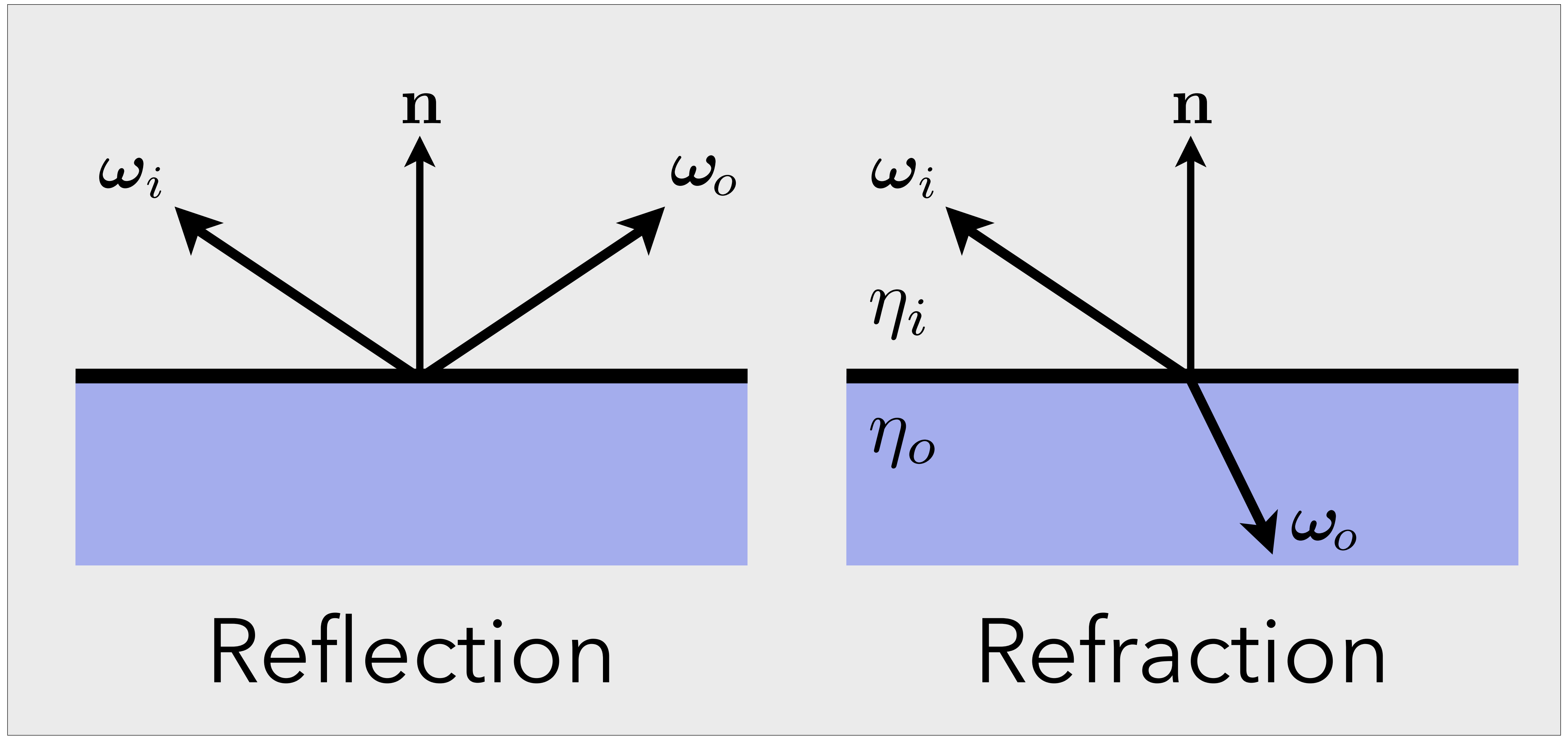
Express as constraint:

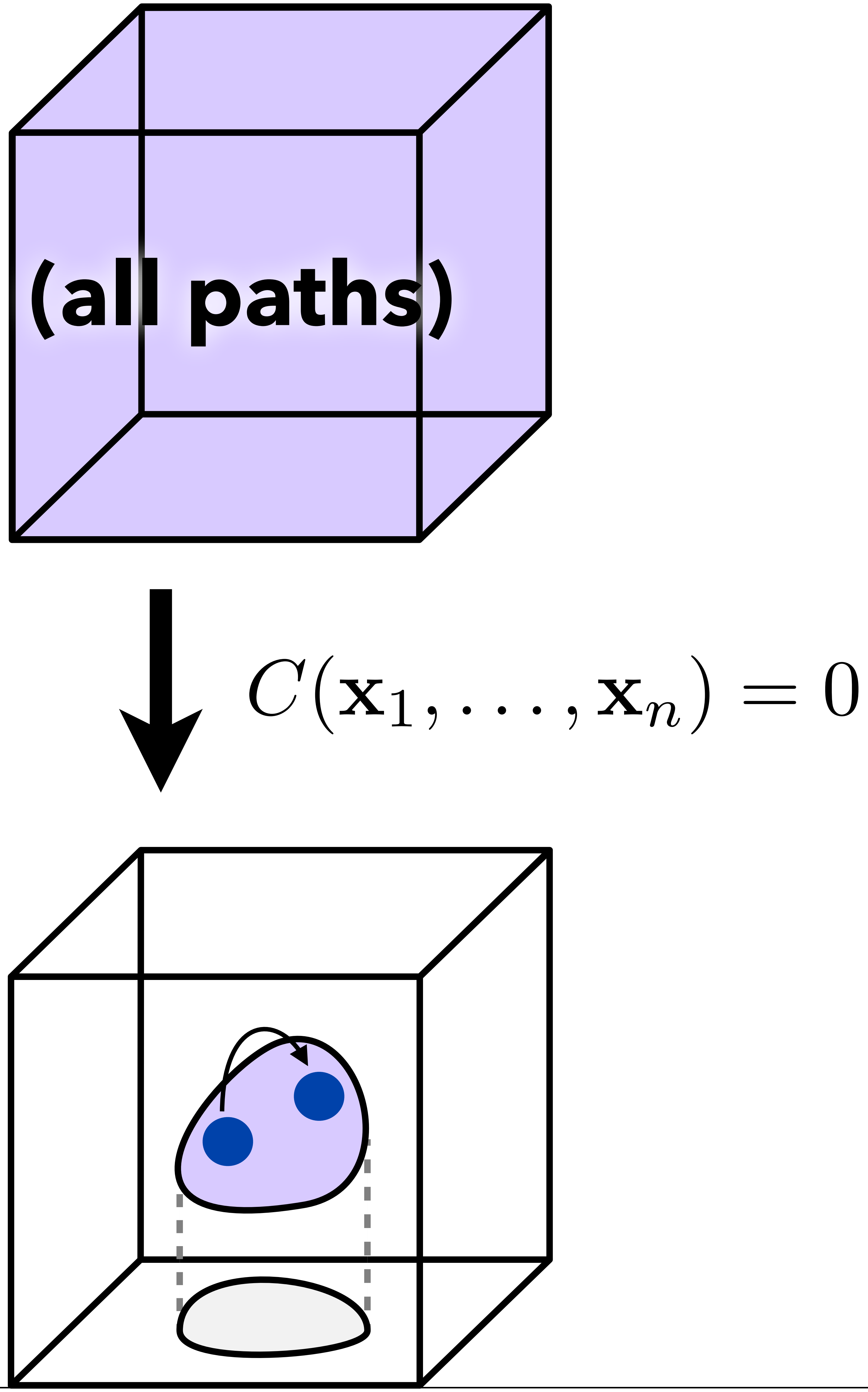
$$c_i(\mathbf{x}_{i-1},\ \mathbf{x}_i,\ \mathbf{x}_{i+1}) = 0$$

Set satisfying all constraints:

$$\mathcal{S} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n \in \Omega \mid C(\mathbf{x}_1, \ldots, \mathbf{x}_n) = 0\}$$

More formally





Express as constraint:

$$c_i(\mathbf{x}_{i-1},\ \mathbf{x}_i,\ \mathbf{x}_{i+1}) = 0$$

Set satisfying all constraints:

$$\mathcal{S} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n \in \Omega \mid C(\mathbf{x}_1, \ldots, \mathbf{x}_n) = 0\}$$

How this is used in a rendering algorithm?

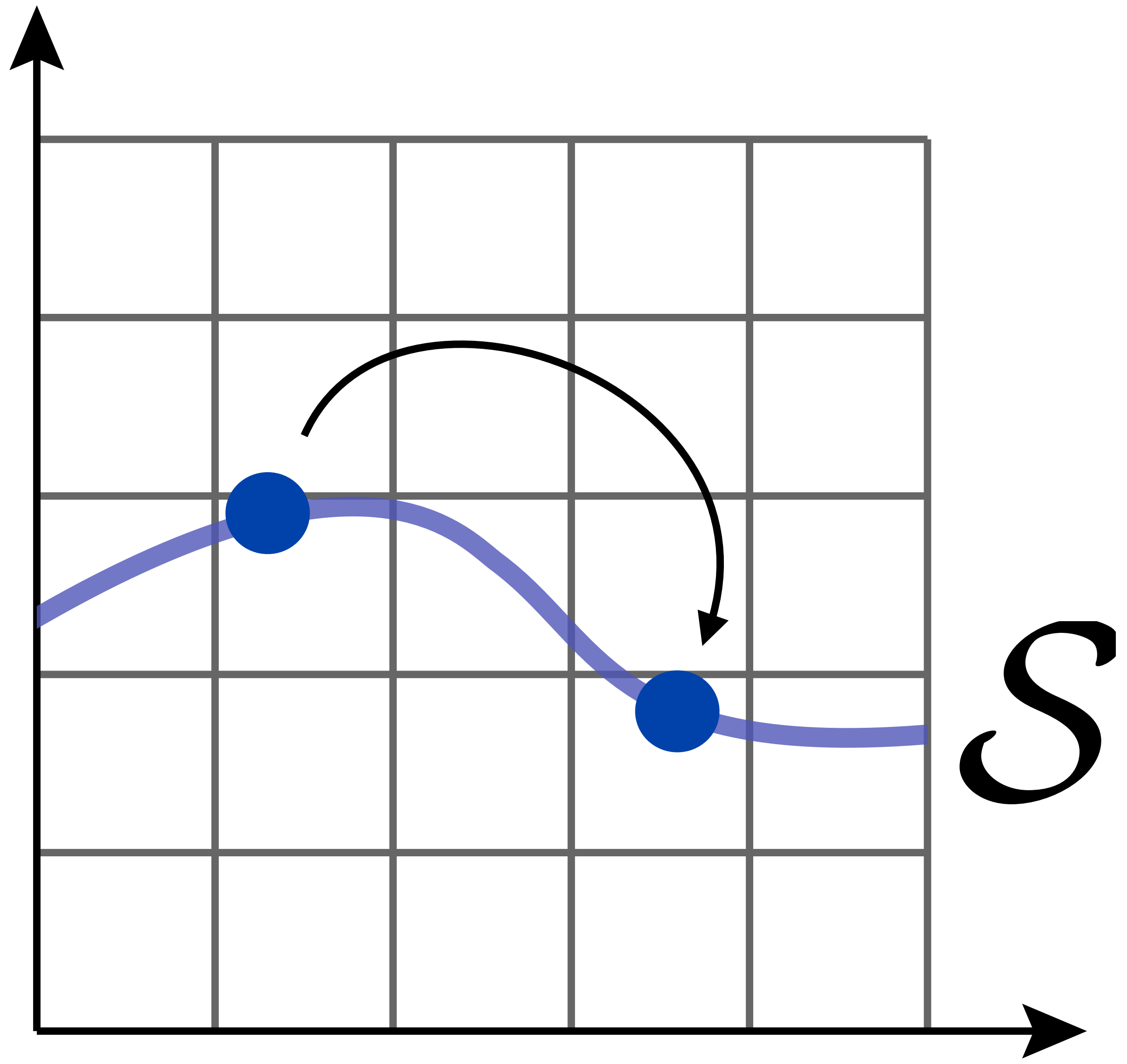
```
def MCMC_Path_Tracer():
    u = [0.5, …, 0.5]

    while !done:
        u’ = perturb(u)

        # Acceptance probability
        a = L(u’) / L(u)

        if a < rand():
            u = u’
```

How this is used in a rendering algorithm?

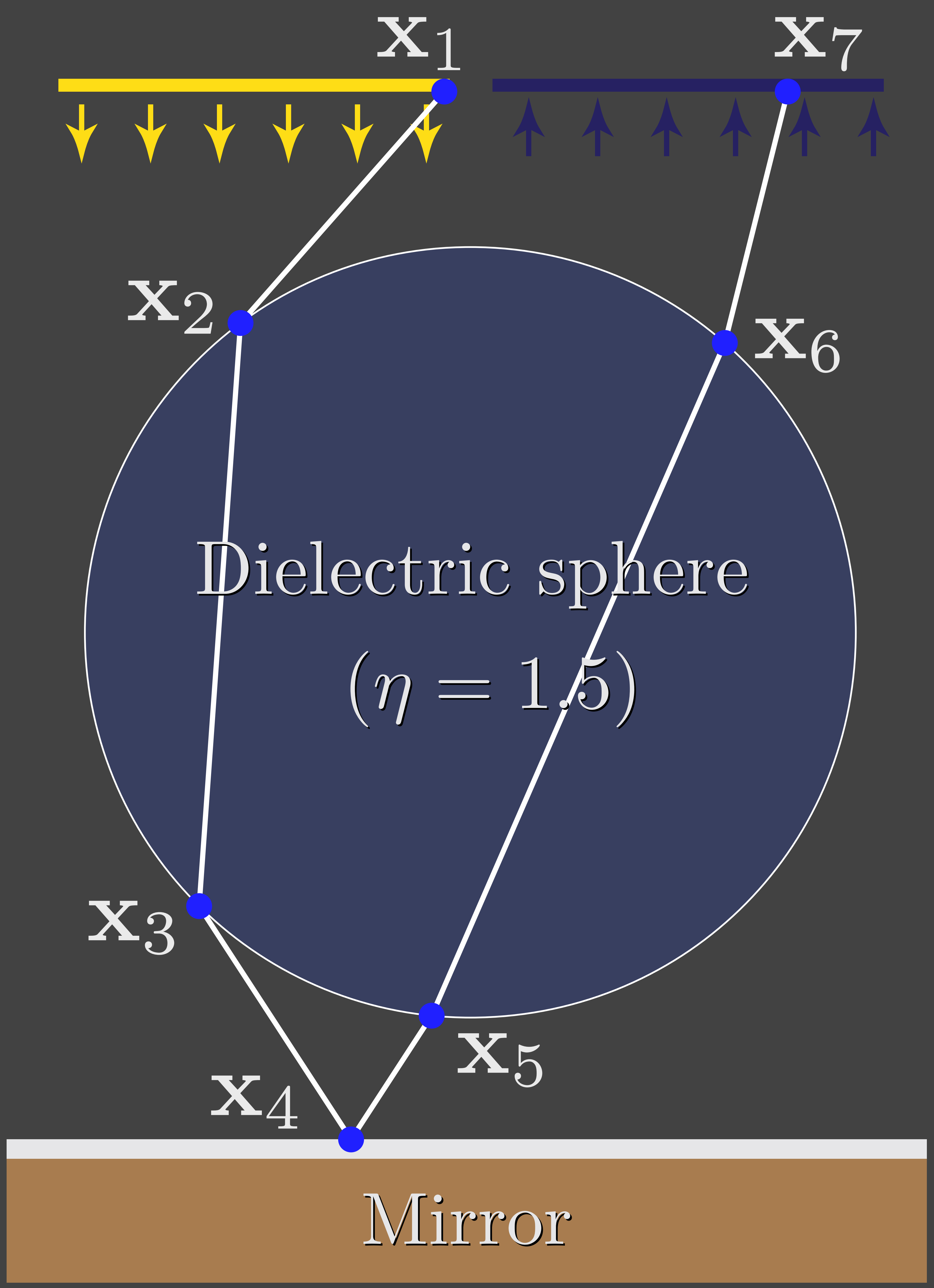
```
def MCMC_Path_Tracer():
    u = [0.5, …, 0.5]

    while !done:
        u’ = perturb(u)

        # Acceptance probability
        a = L(u’) / L(u)

        if a < rand():
            u = u’
```

How this is used in a rendering algorithm?

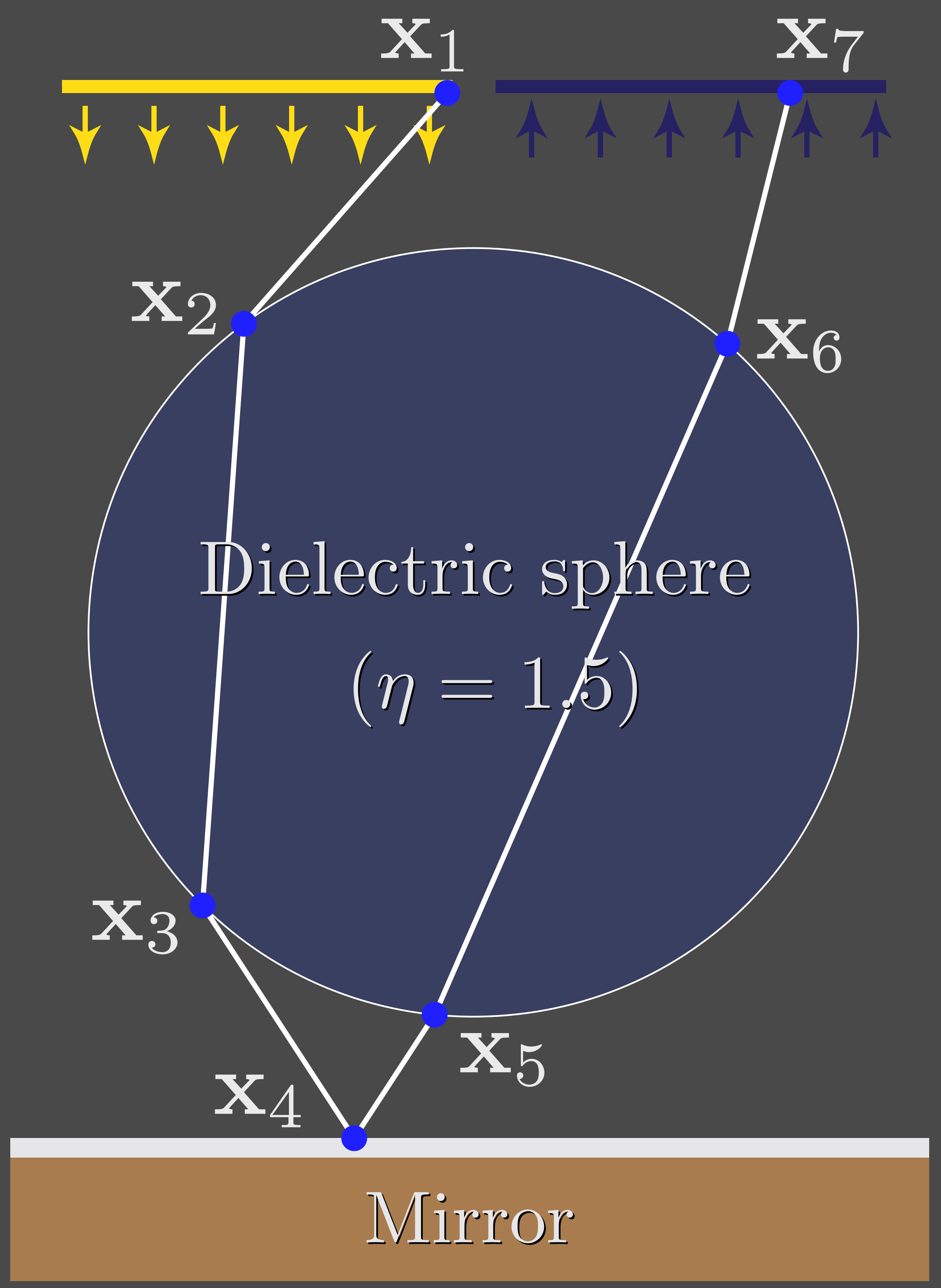
```
def MCMC_Path_Tracer():
    u = [0.5, …, 0.5]

    while !done:
        u’ = perturb(u)

        # Acceptance probability
        a = L(u’) / L(u)

        if a < rand():
            u = u’
```

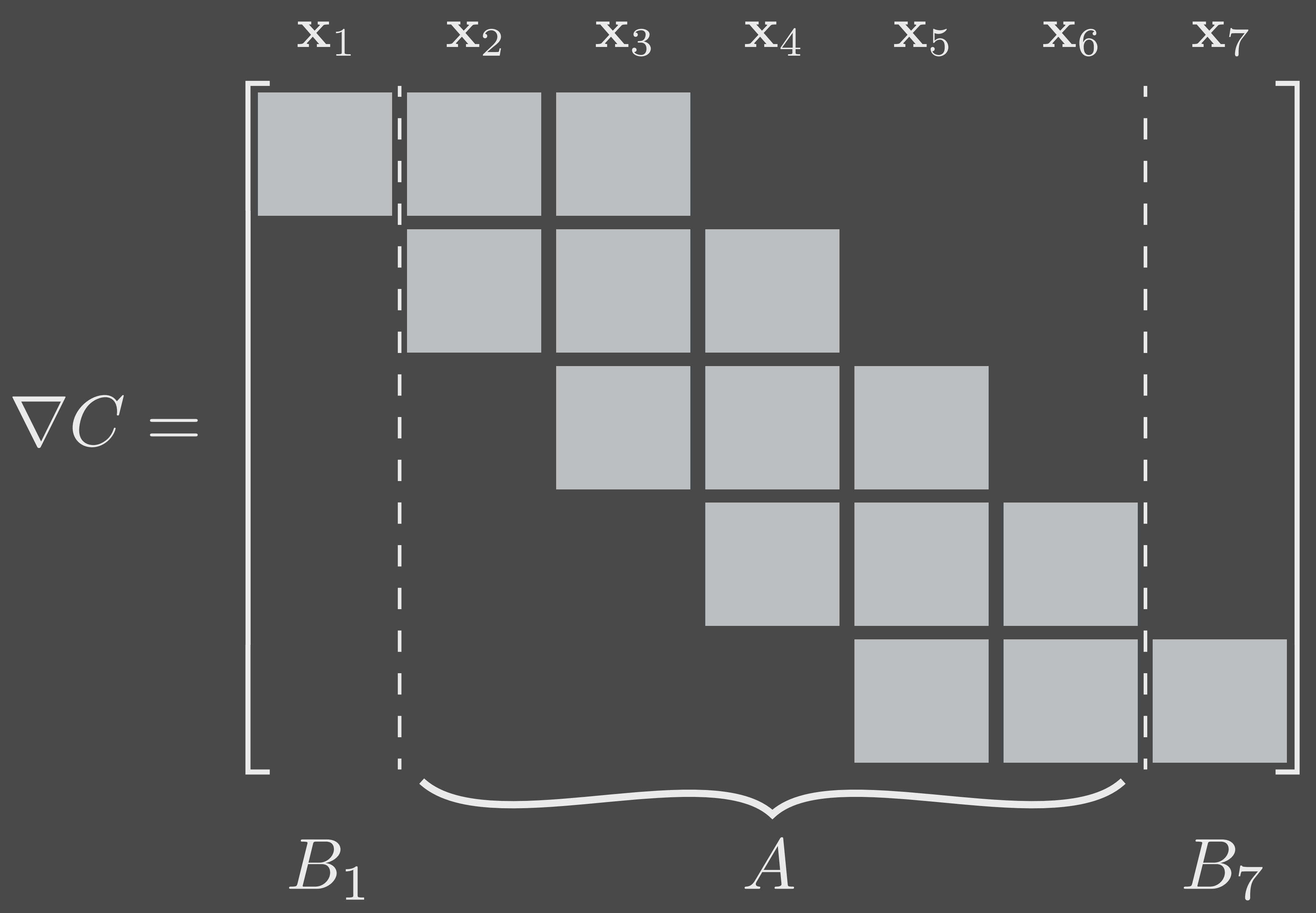


Manifold walks



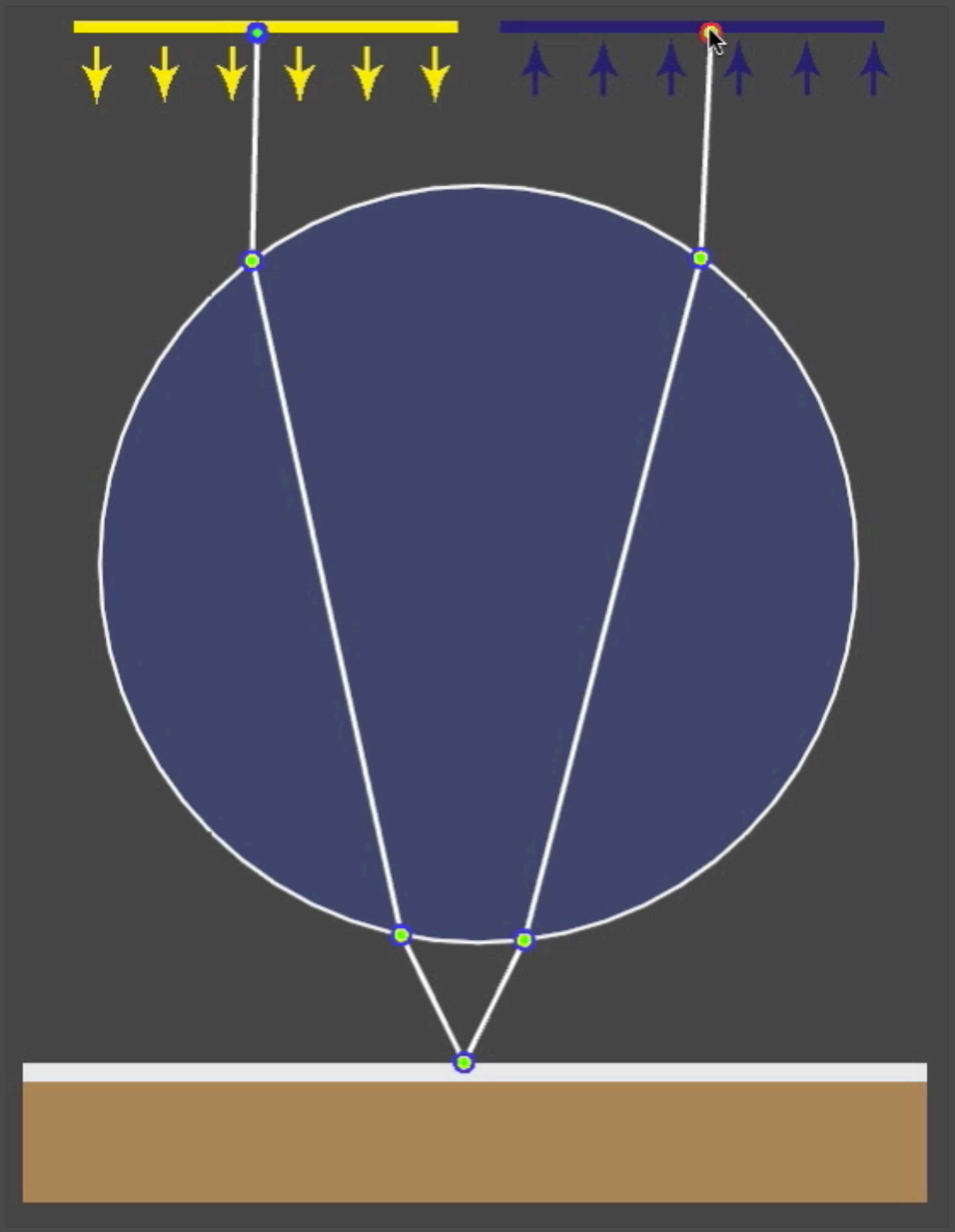
Manifold walks



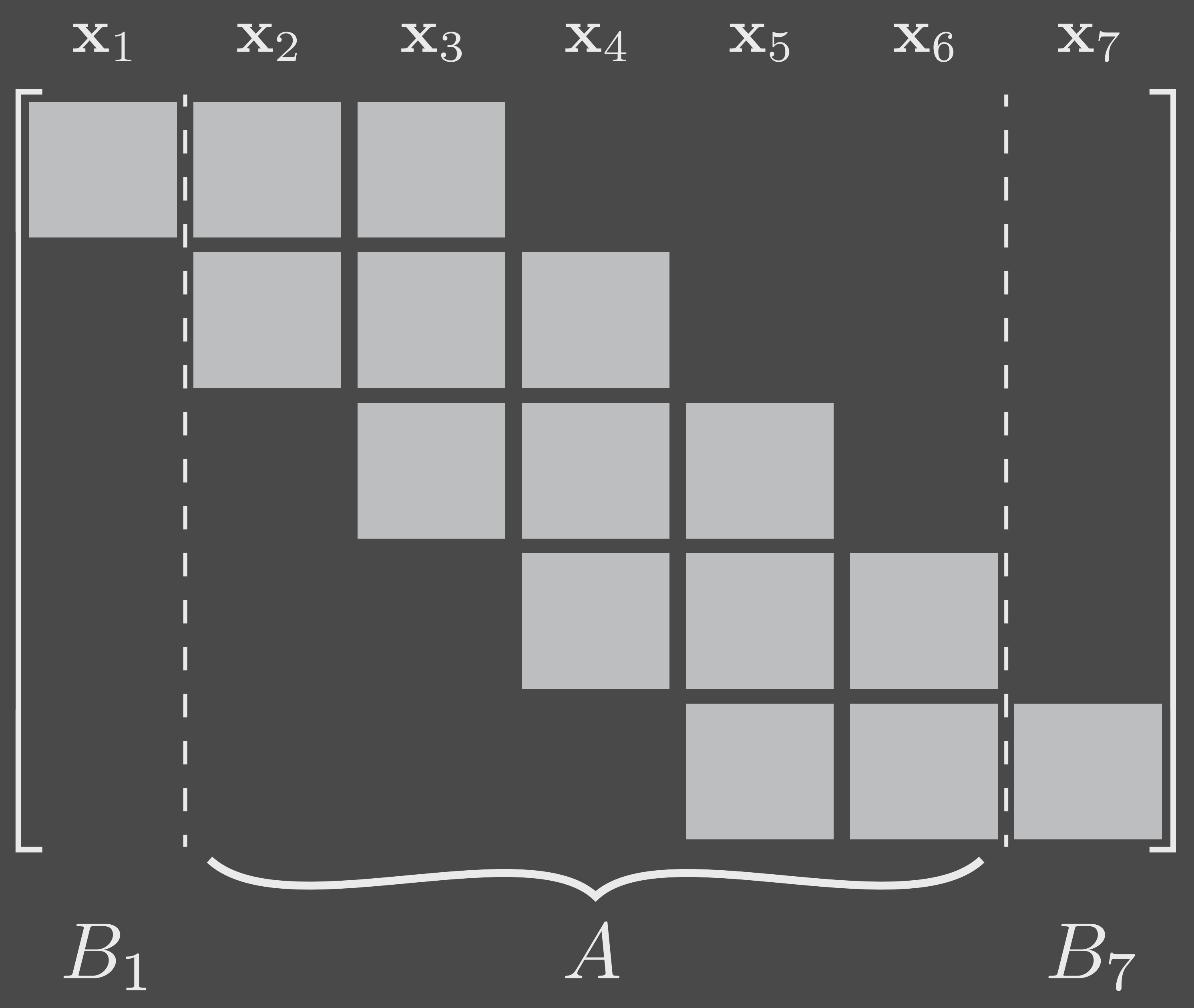


$$\frac{\partial \begin{bmatrix}\mathbf{x}_2\\ \vdots \\ \mathbf{x}_6\end{bmatrix}}{\partial \mathbf{x}_7} = -A^{-1}B_7$$

Manifold walks



$\nabla C =$

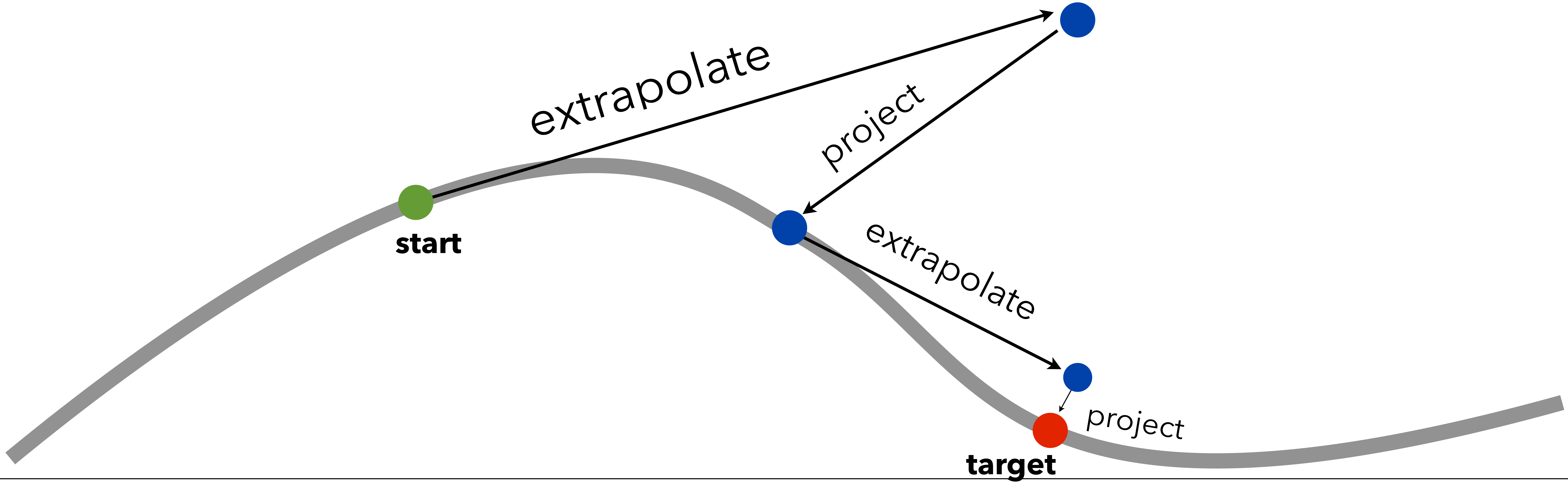


$$\frac{\partial \begin{bmatrix} \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_6 \end{bmatrix}}{\partial \mathbf{x}_7} = -A^{-1}B_7$$

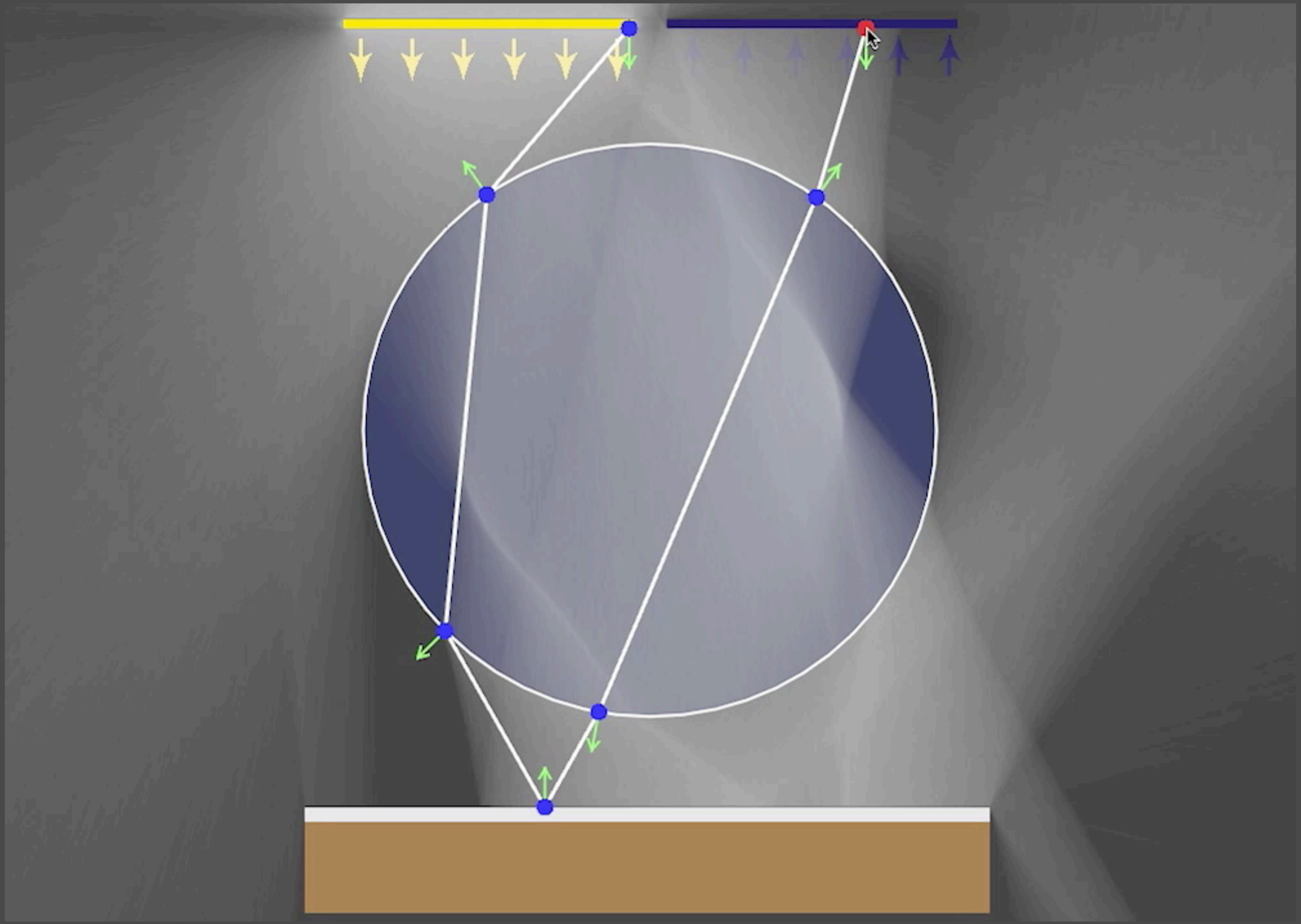
Manifold walking algorithm

Basic idea:

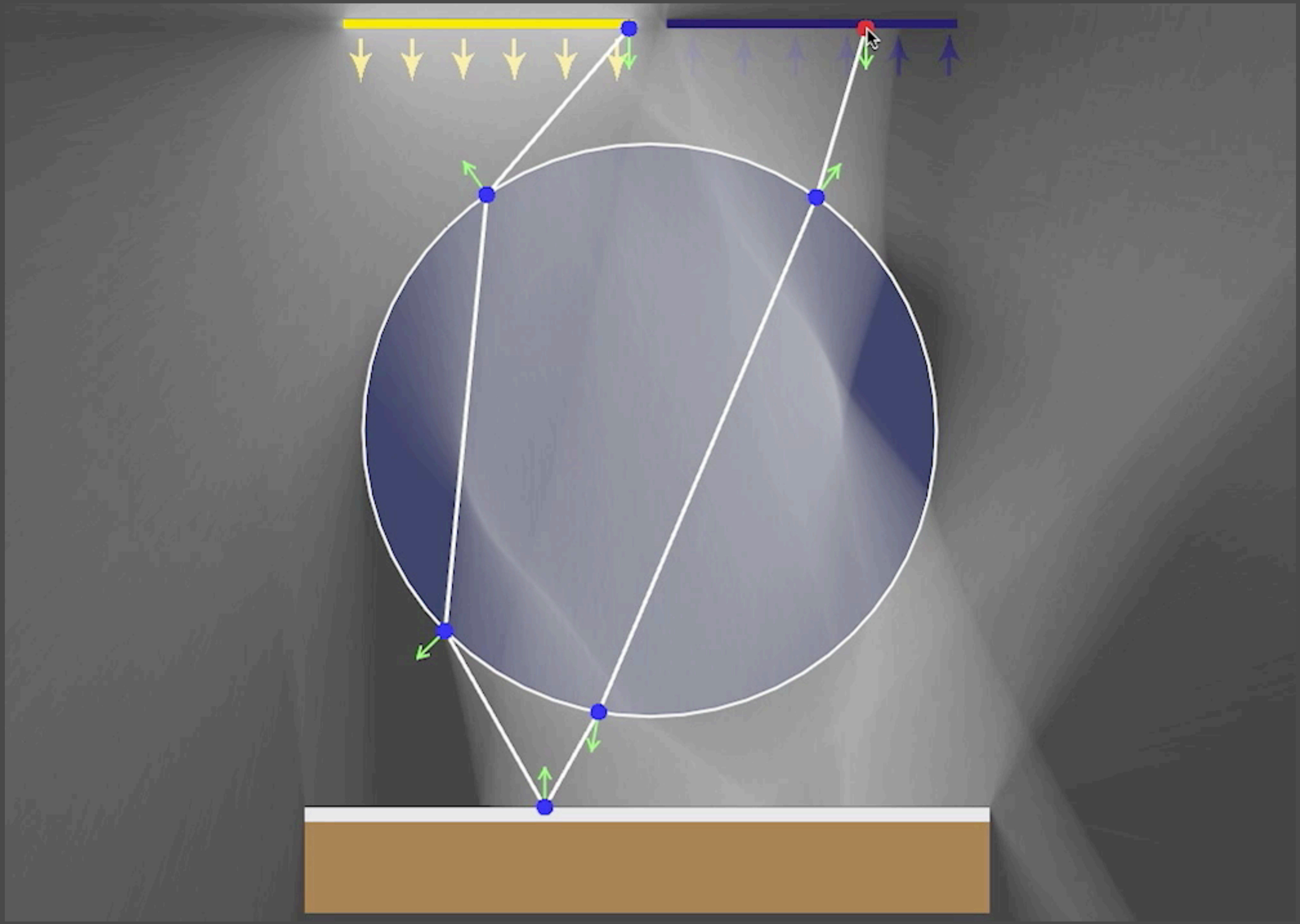
```
while not there yet:
    1. EXTRAPOLATE: Perturb vertices using manifold tangents
    2. PROJECT: re-trace extrapolated path
```



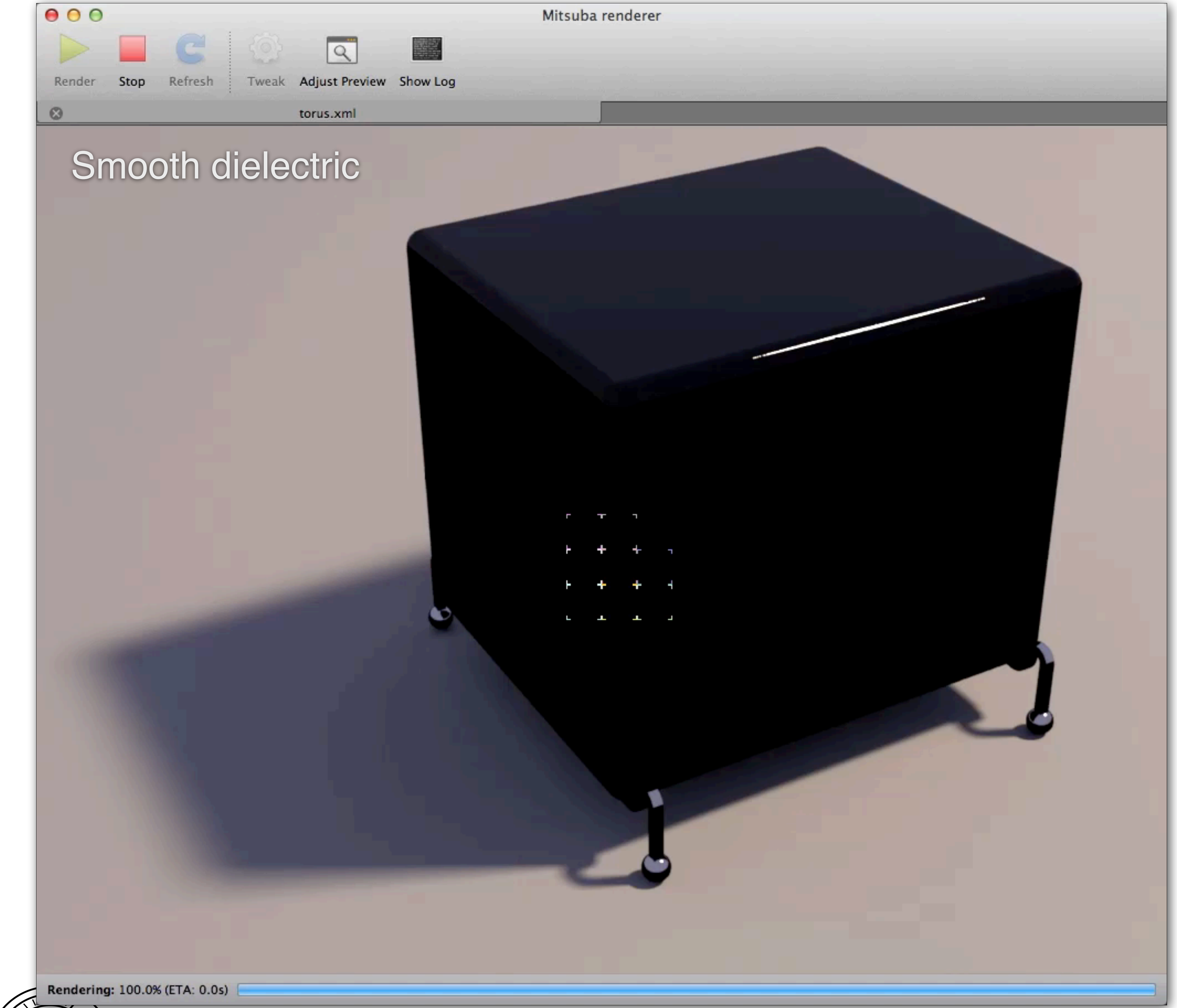
Both steps combined



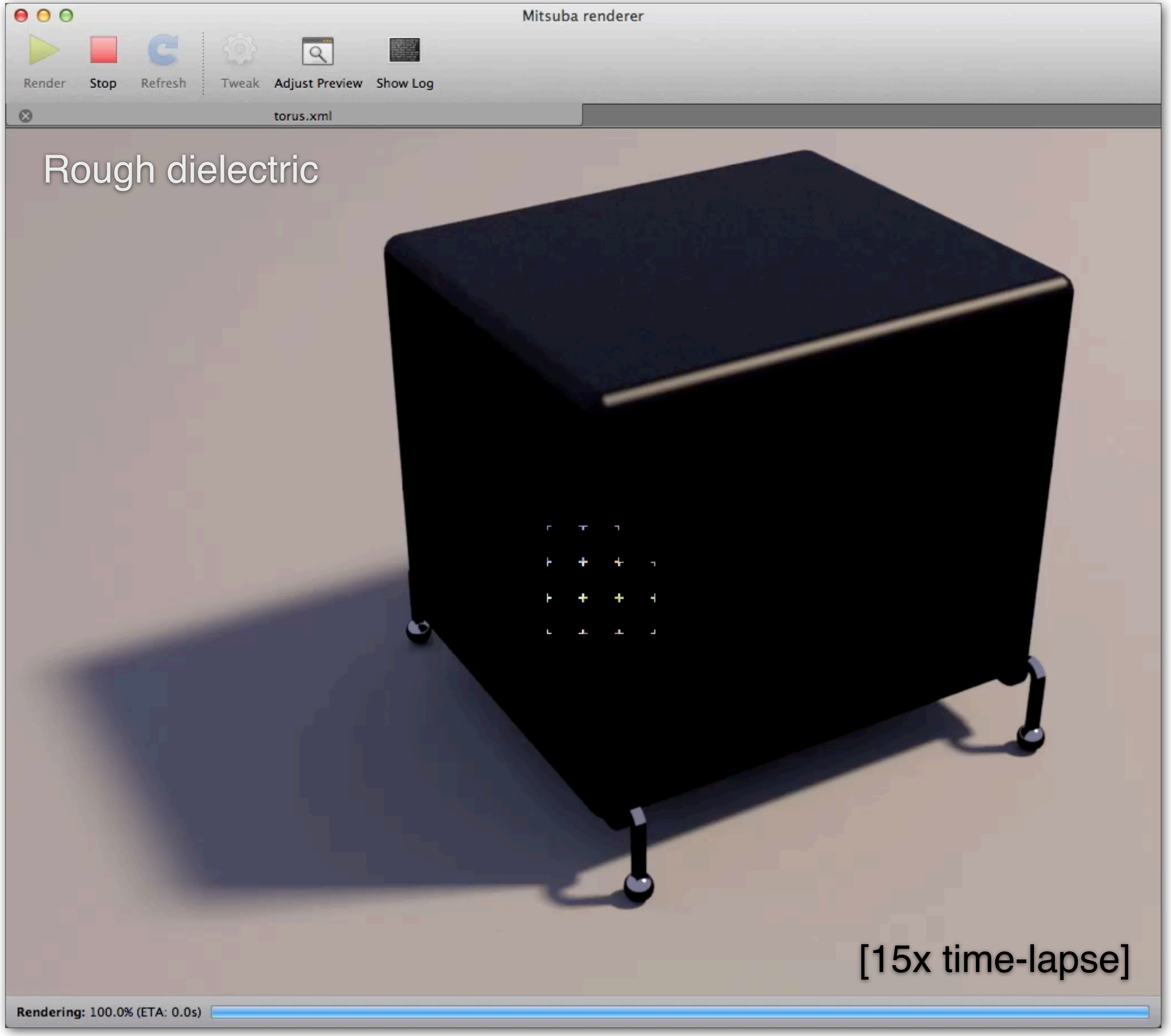
Both steps combined



Manifold Exploration Path Tracing

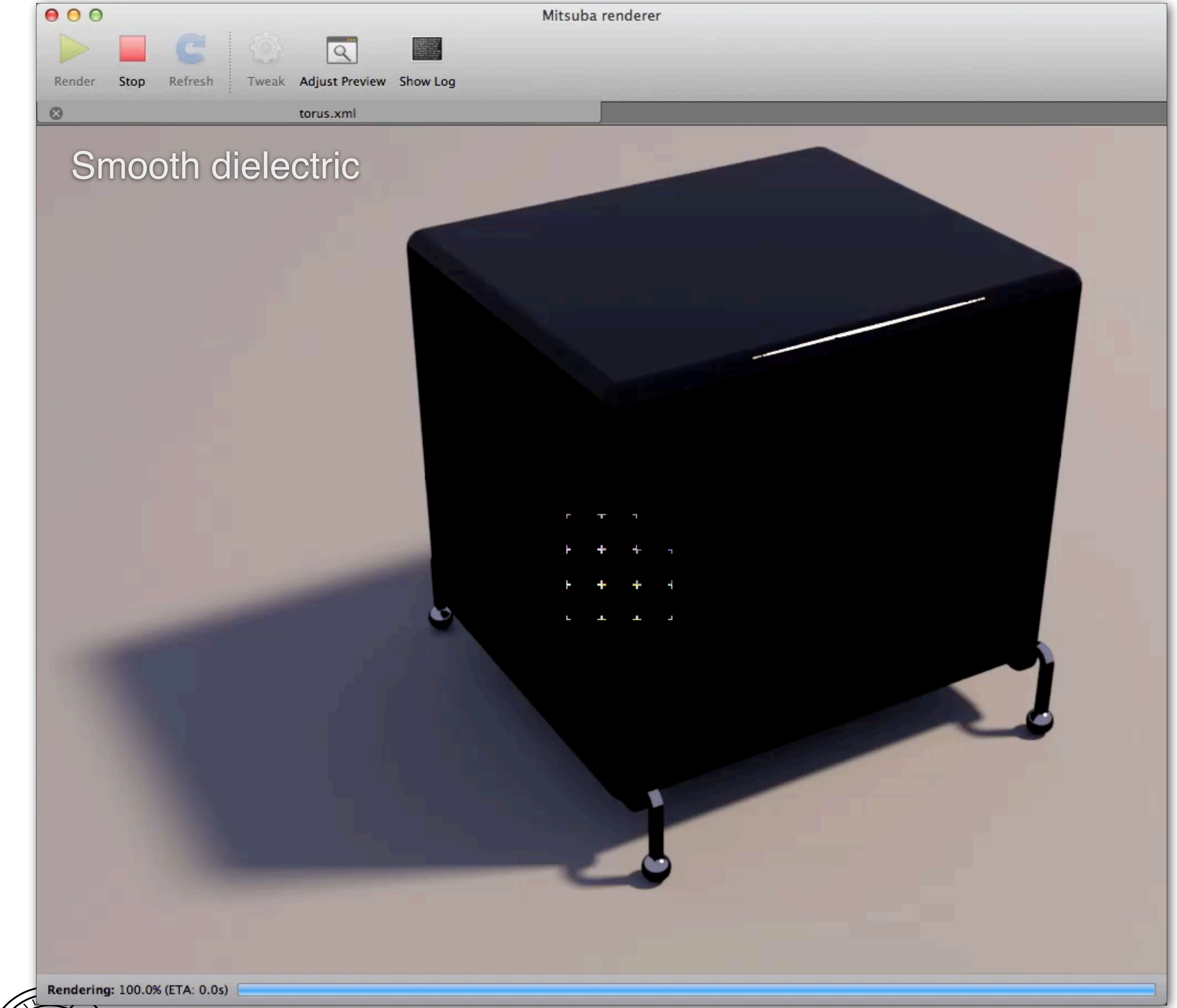


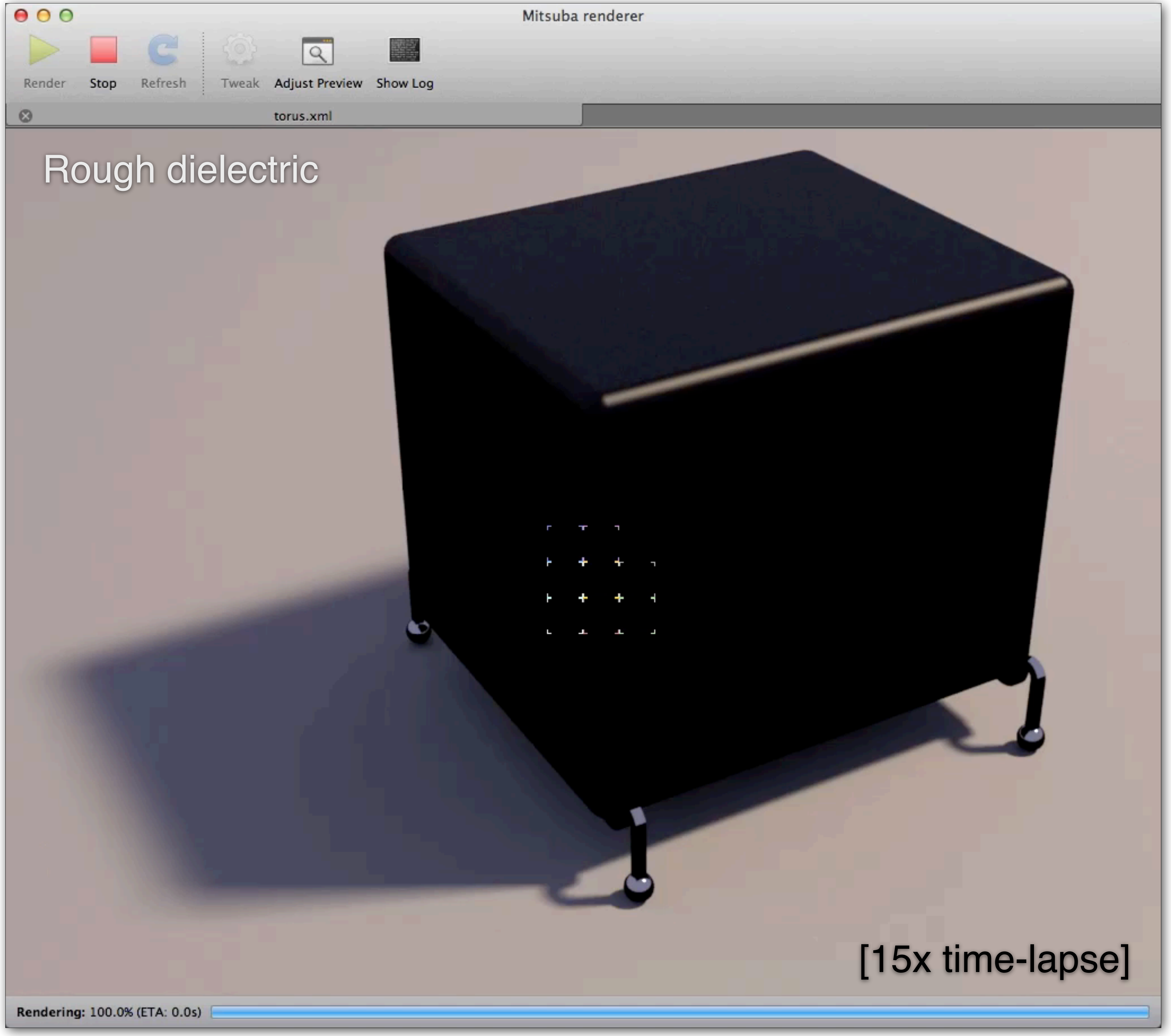
Rendering: 100.0% (ETA: 0.0s)

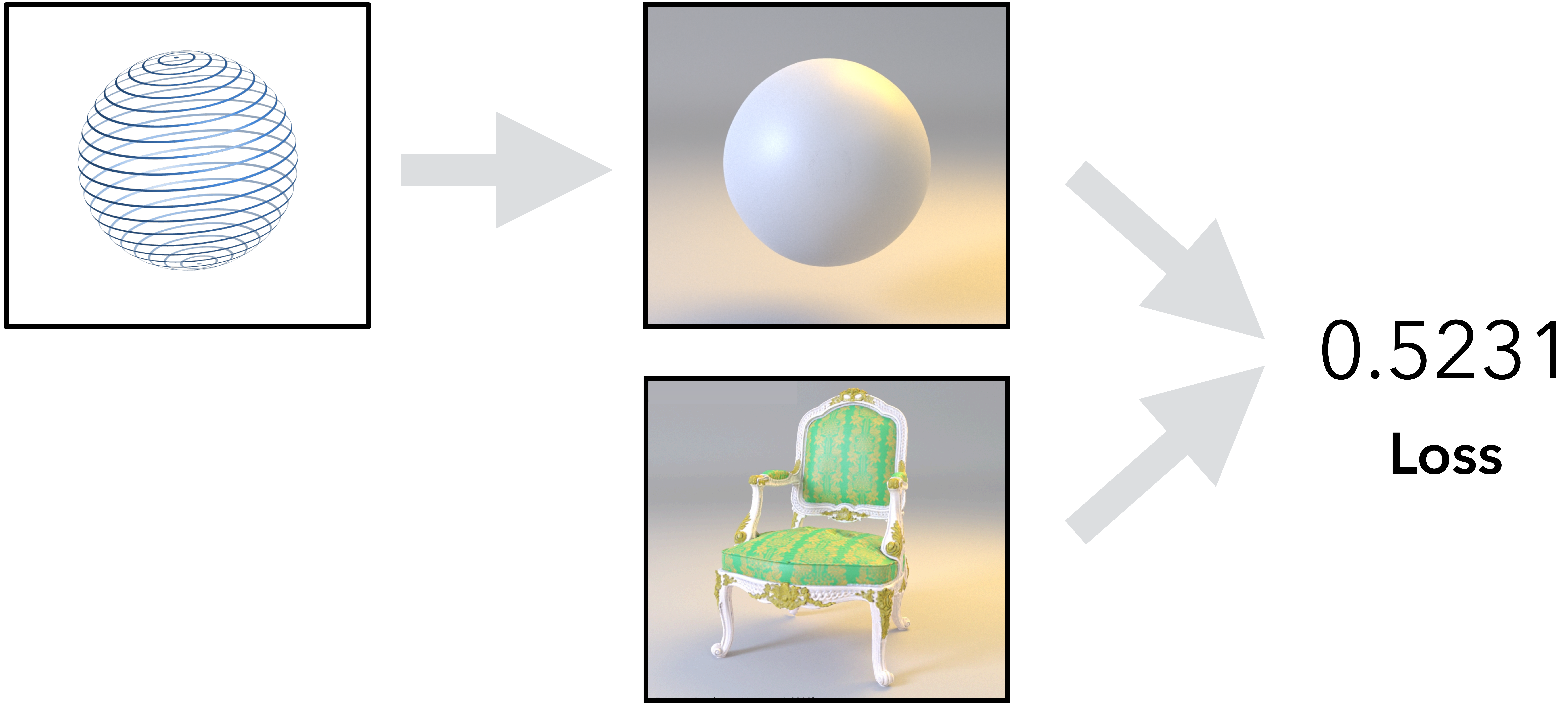


Rendering: 100.0% (ETA: 0.0s)

Manifold Exploration Path Tracing







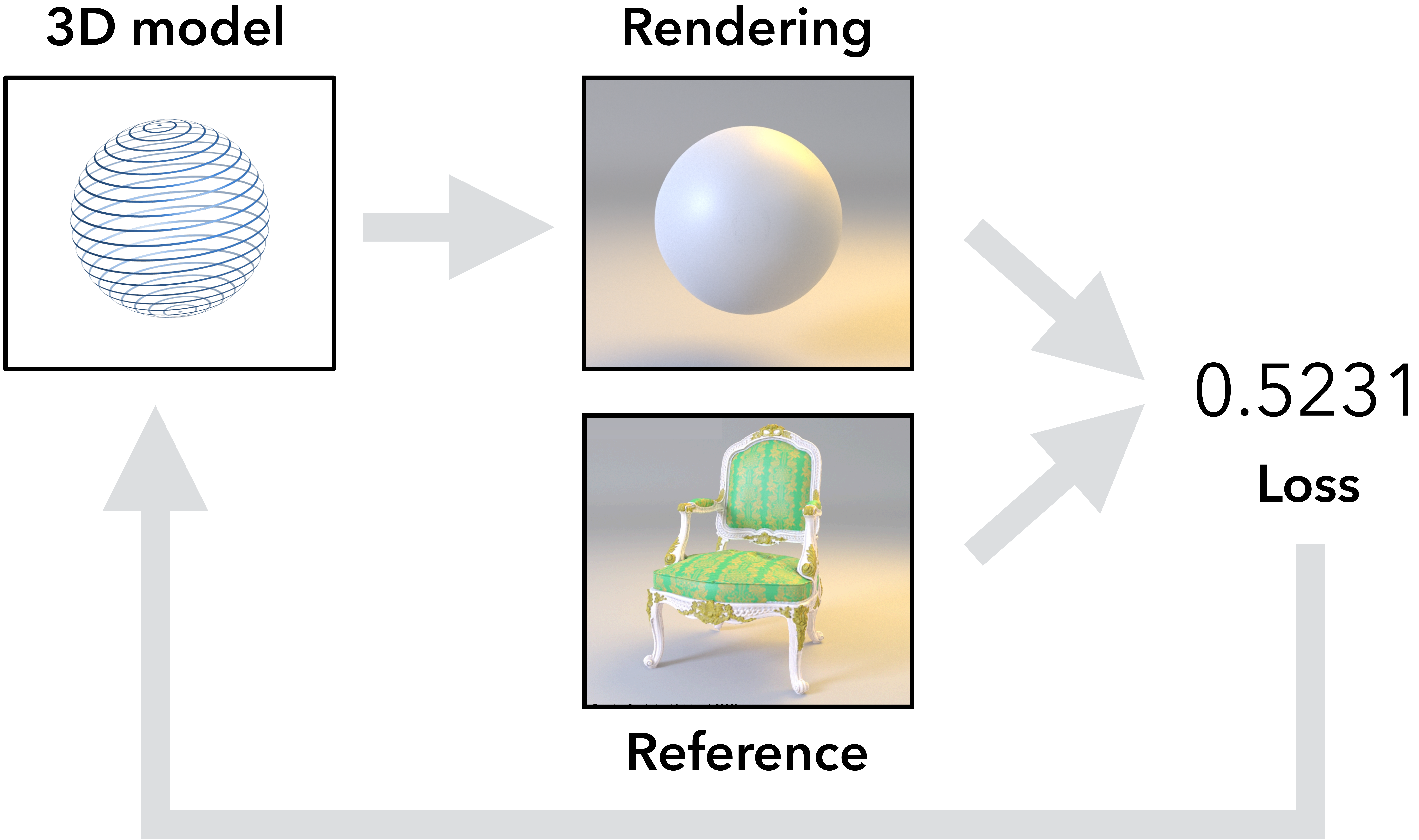
3D model

Rendering

Reference

0.5231

Loss



3D model

Rendering

Reference

0.5231

Loss

MCMC for Inverse Rendering

Markov-Chain Monte Carlo Sampling of Visibility Boundaries for Differentiable Rendering

PEIYU XU, University of California Irvine, United States of America
SAI BANGARU, MIT CSAIL, United States of America
TZU-MAO LI, University of California San Diego, United States of America
SHUANG ZHAO, University of California Irvine, United States of America

Fig. 1. We introduce a Markov-Chain-Monte-Carlo (MCMC) differentiable rendering method to sample boundary paths that are crucial for estimating derivatives with respect to object geometries. This example includes a highly tessellated Nefertiti model (**with 1 million vertices**) lit by one area lights from the bottom left corner (a), casting a shadow on the right wall, and observed both directly and indirectly through a mirror. We compute derivatives with respect to the translation of the Nefertiti model. The highly tessellated mesh and small area emitter make visibility boundary sampling extremely difficult—even with primary-sample-space guiding. At equal sample and lower render time, our technique (d) produces significantly cleaner derivative estimates than the state-of-the-art baseline [Zhang et al. 2023] (e).

Physics-based differentiable rendering requires estimating boundary path integrals emerging from the shift of discontinuities (e.g., visibility boundaries). Previously, although the mathematical formulation of boundary path integrals has been established, efficient and robust estimation of these integrals has remained challenging. Specifically, state-of-the-art boundary sampling methods all rely on primary-sample-space guiding precomputed using sophisticated data structures—whose performance tends to degrade for finely tessellated geometries.

In this paper, we address this problem by introducing a new Markov-Chain-Monte-Carlo (MCMC) method. At the core of our technique is a local perturbation step capable of efficiently exploring highly fragmented primary sample spaces via specifically designed jumping rules. We compare the performance of our technique with several state-of-the-art baselines using synthetic differentiable-rendering and inverse-rendering experiments.

Authors' Contact Information: Peiyu Xu, University of California Irvine, United States of America, peiyux3@uci.edu; Sai Bangaru, MIT CSAIL, United States of America, sbangaru@mit.edu; Tzu-Mao Li, University of California San Diego, United States of America, tzli@ucsd.edu; Shuang Zhao, University of California Irvine, United States of America, shz@ics.uci.edu.

SA Conference Papers '24, December 03–06, 2024, Tokyo, Japan
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1131-2/24/12
https://doi.org/10.1145/3680528.3687622

CCS Concepts: • **Computing methodologies → Rendering**.

Additional Key Words and Phrases: Differentiable rendering, differential path integral, Markov-chain Monte Carlo

ACM Reference Format:
Peiyu Xu, Sai Bangaru, Tzu-Mao Li, and Shuang Zhao. 2024. Markov-Chain Monte Carlo Sampling of Visibility Boundaries for Differentiable Rendering. In *SIGGRAPH Asia 2024 Conference Papers (SA Conference Papers '24), December 03–06, 2024, Tokyo, Japan*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3680528.3687622

1 Introduction

Differentiable rendering computes gradients of detector responses with respect to differential changes of a virtual scene. Being an active research topic in computer graphics, differentiable rendering is a key ingredient for integrating the rendering processes into probabilistic inference and machine learning pipelines, leading to applications in a wide array of areas including computer vision, computational imaging, and computational fabrication.

Recently, great progress has been made in physics-based differentiable rendering theory and algorithms [Li et al. 2018; Zhang et al. 2019, 2020; Bangaru et al. 2020; Xu et al. 2023]. These advances have enabled the capability of differentiating renderings with complex light-transport effects (e.g., interreflection) with respect to arbitrary scene parameters including those controlling global object geometry (e.g., the positions of mesh vertices). Mathematically, it has been

[Xu et al. 2024]

Markov-Chain Monte Carlo Sampling of Visibility Boundaries for Differentiable Rendering

MCMC in Optimization

Intro to Bayesian statistics

Stochastic Gradient Descent (SGD)

Stochastic Gradient Langevin Dynamics

Bayesian statistics

Setup: classification problem

Problem statement

Data

Setup: classification problem

Problem statement

Data

Setup: classification problem

Problem statement

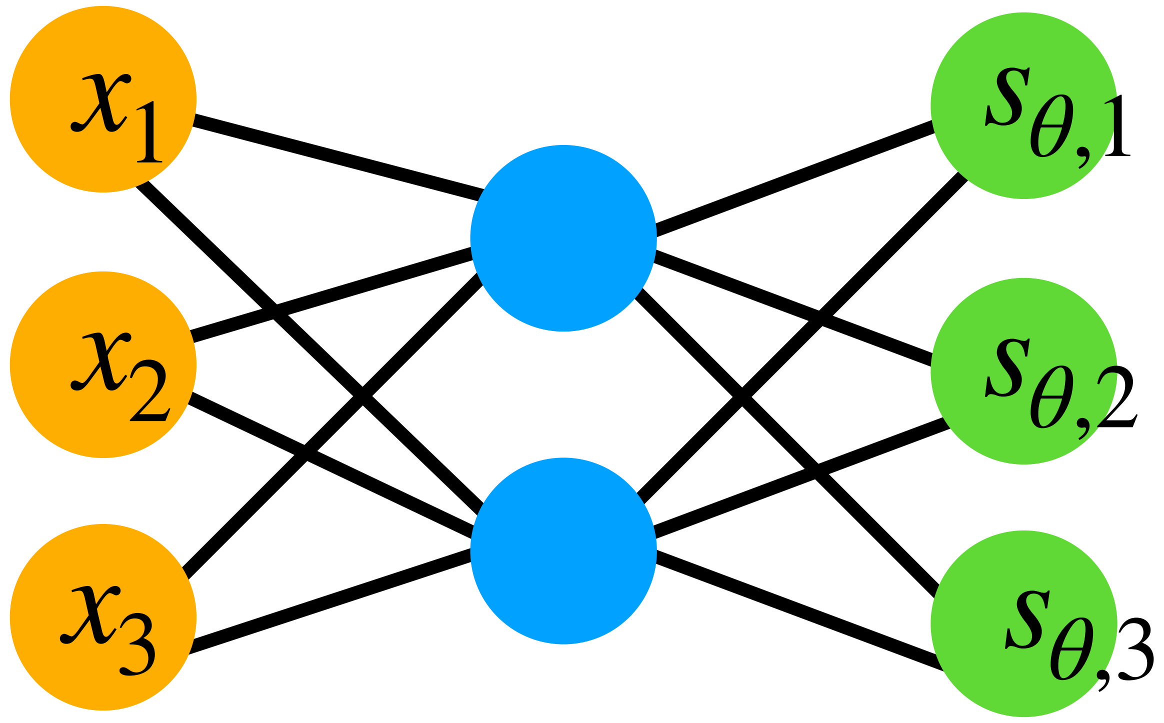
$x_i \sim X$

$i \in \{1,2,3,\cdots\}$

Data

Setup: classification problem

Problem statement



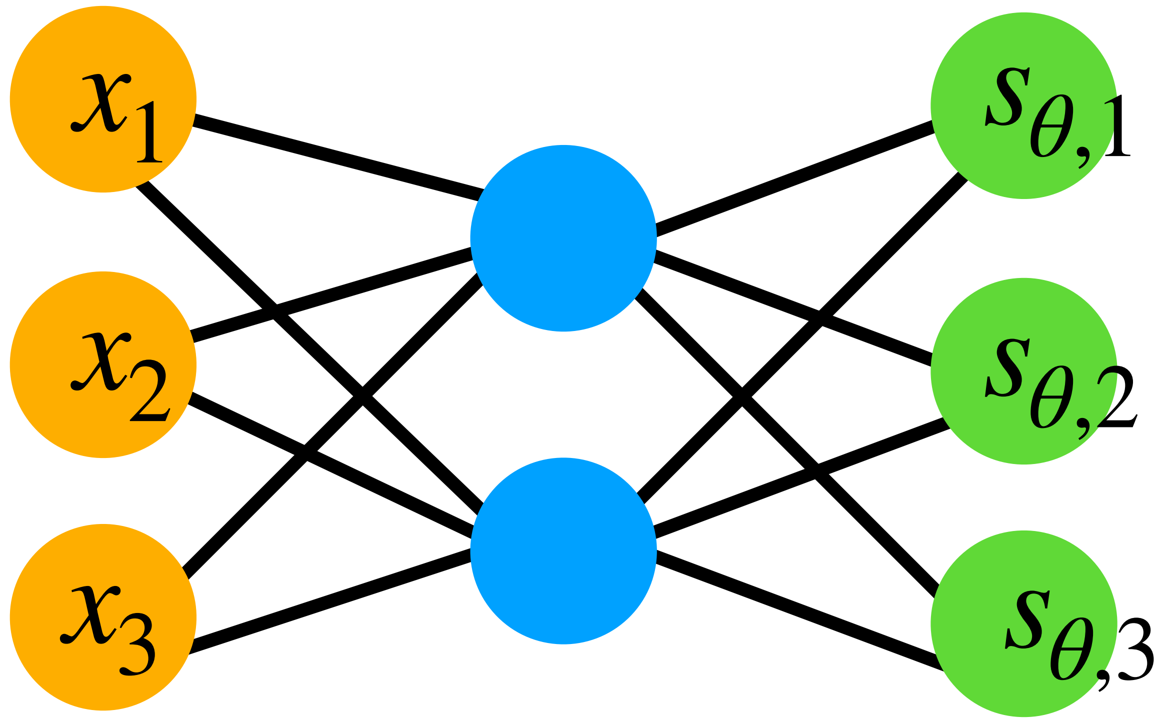
$$x_i \sim X$$

$$i \in \{1,2,3,\cdots\}$$

Data

Setup: classification problem

Problem statement



$$\begin{bmatrix} 0.09 \\ 0.69 \\ 0.14 \end{bmatrix}$$

$x_i \sim X$

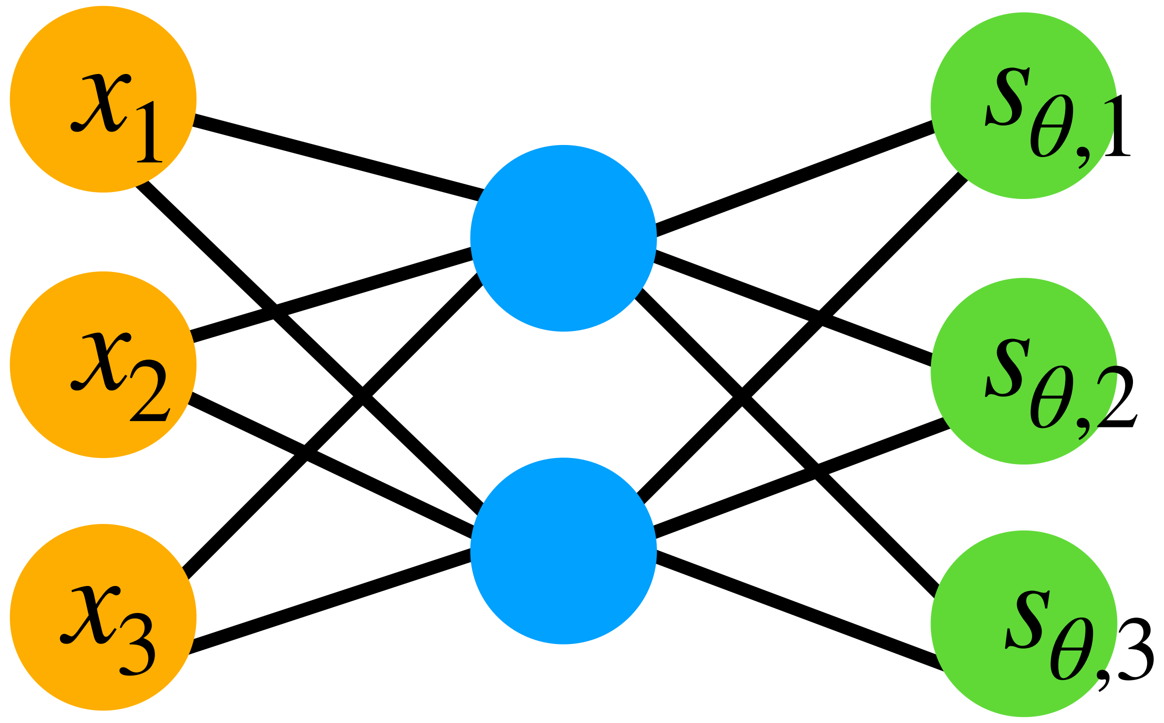
$i \in \{1,2,3,\cdots\}$

Data

Setup: classification problem

Problem statement

Data



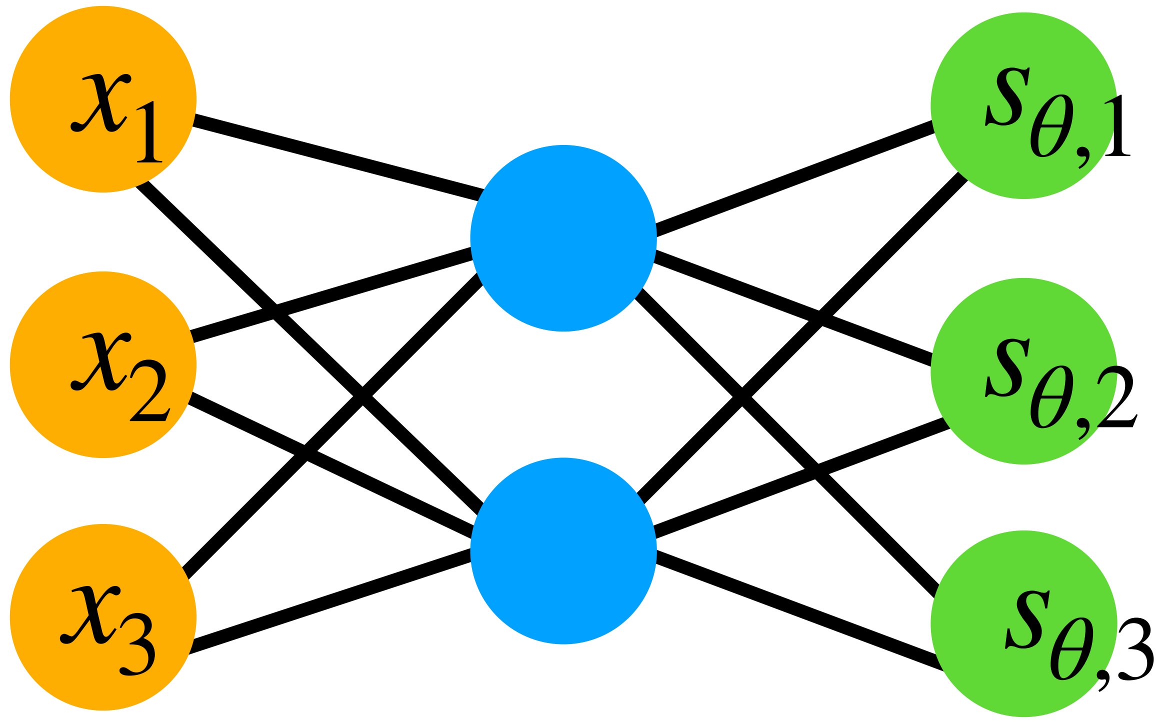
$$\begin{bmatrix}0.09\\0.69\\0.14\end{bmatrix}$$

$x_i \sim X$

$i \in \{1,2,3,\cdots\}$

Setup: classification problem

Problem statement



$$\begin{bmatrix} 0.09 \\ 0.69 \\ 0.14 \end{bmatrix}$$

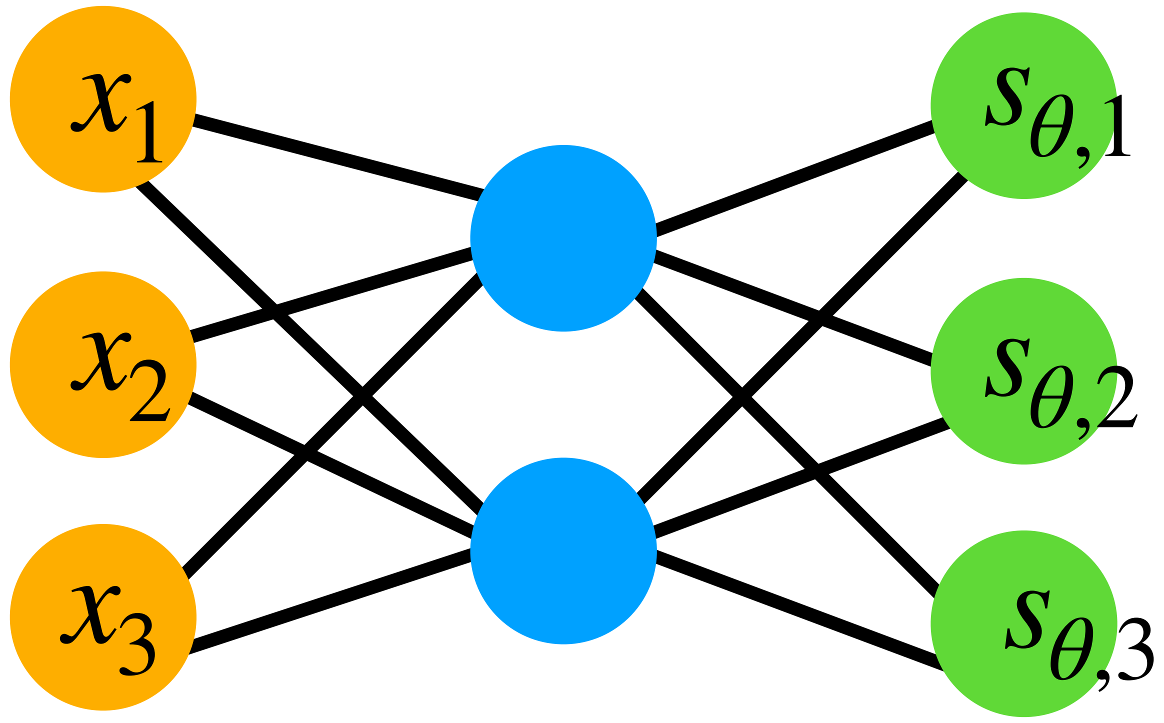
$x_i \sim X$

$i \in \{1,2,3,\cdots\}$

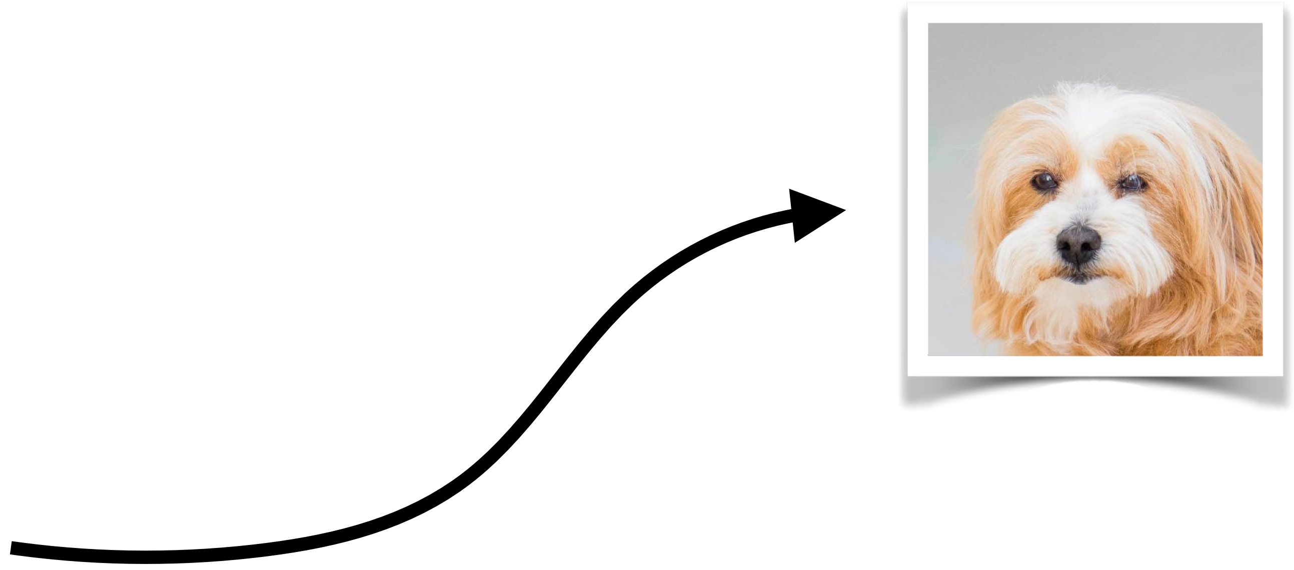
Data

Setup: classification problem

Problem statement



$$\begin{bmatrix}0.09\\ \boxed{0.69}\\ 0.14\end{bmatrix}$$



$x_i \sim X$

$i \in \{1,2,3,\cdots\}$

Data

Bayesian statistics

$$X$$

Bayesian statistics

$$X$$

Random variable

Bayesian statistics

$$X \in [0,1)$$

Random variable

Bayesian statistics

$$X \in [0,1)$$

Bayesian statistics

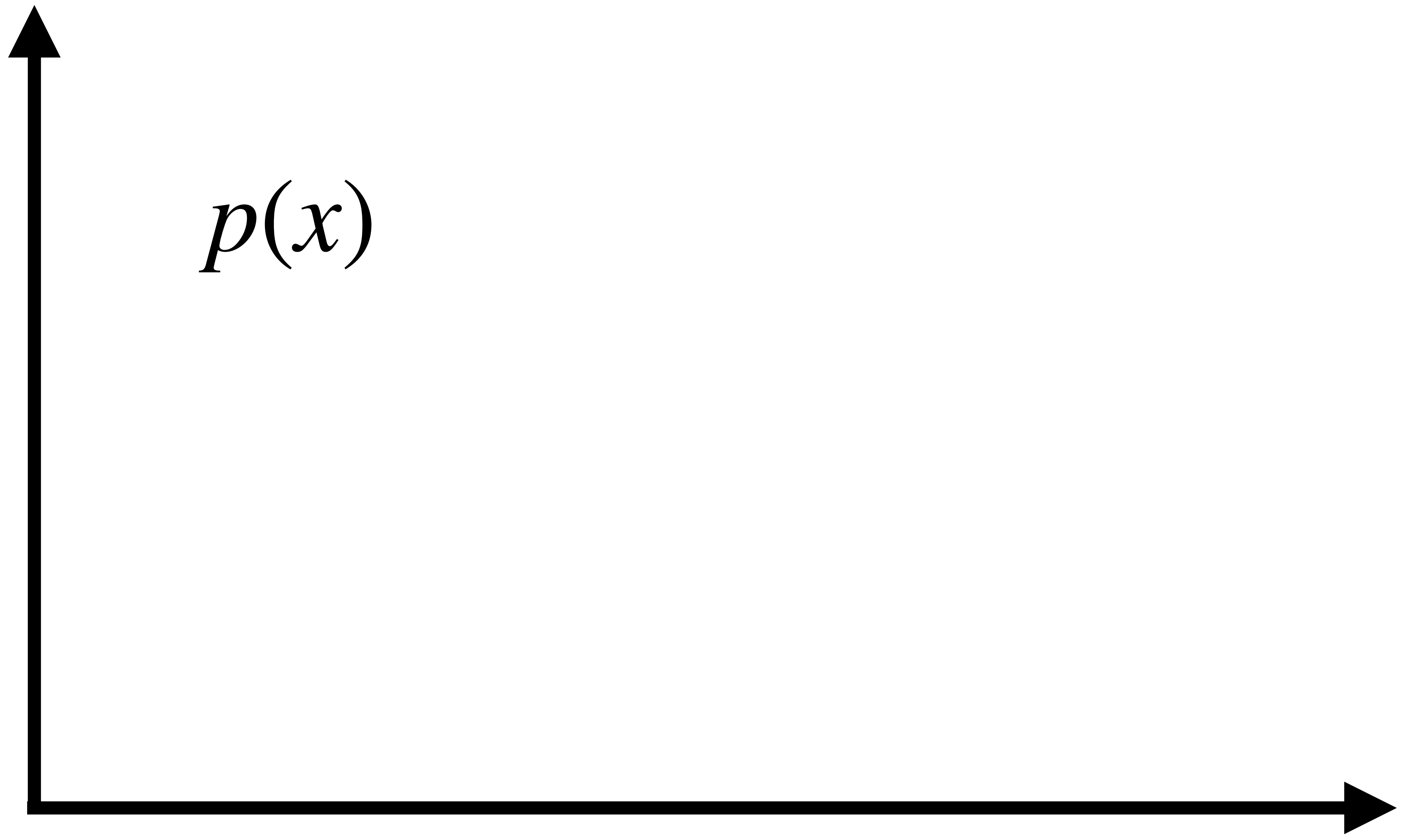
$$x \sim X \quad \in [0,1)$$

Sampling

Probability density function

$x \sim X \quad \in [0,1)$

Sampling

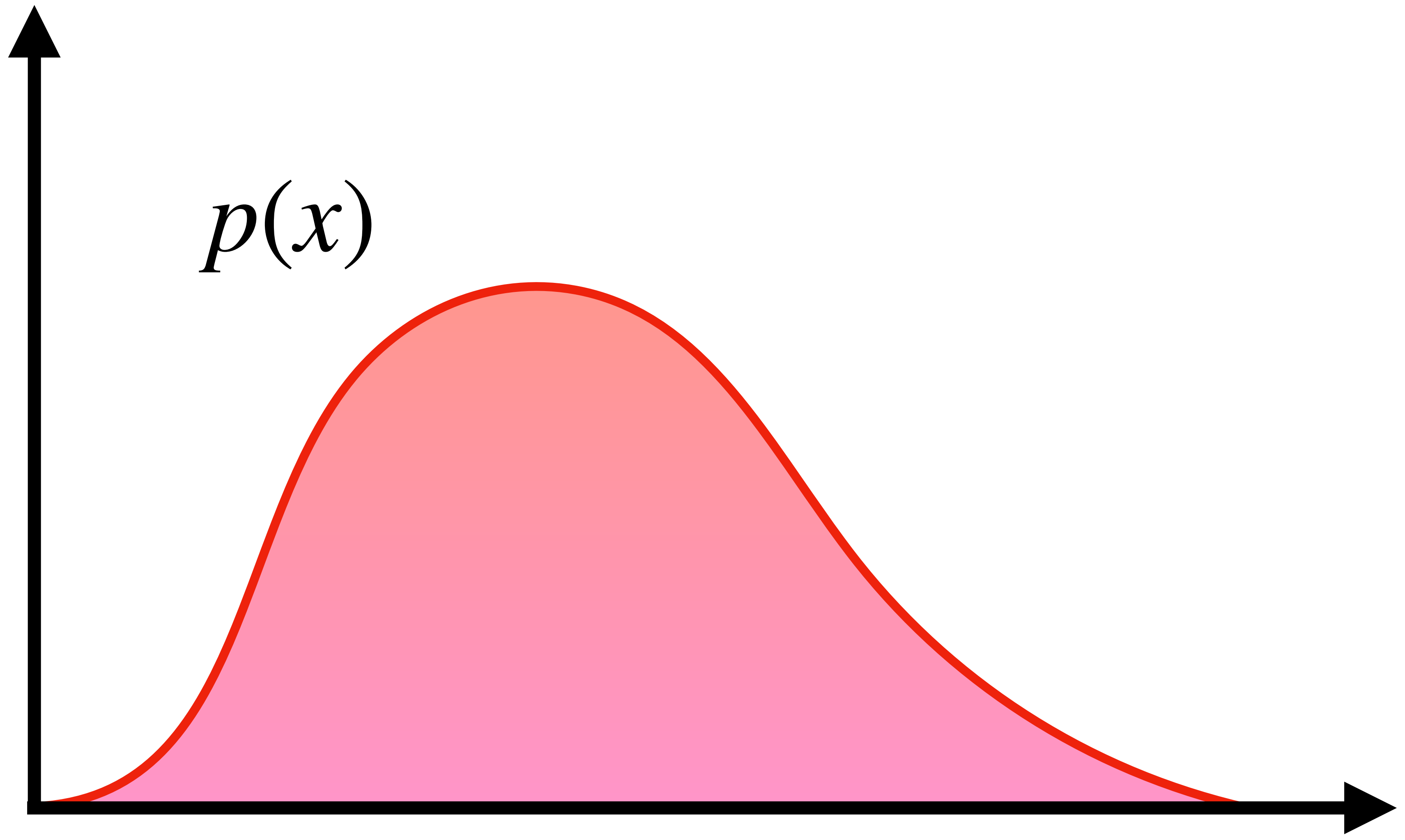


Probability density function

Probability density function

$x \sim X \in [0,1)$

Sampling

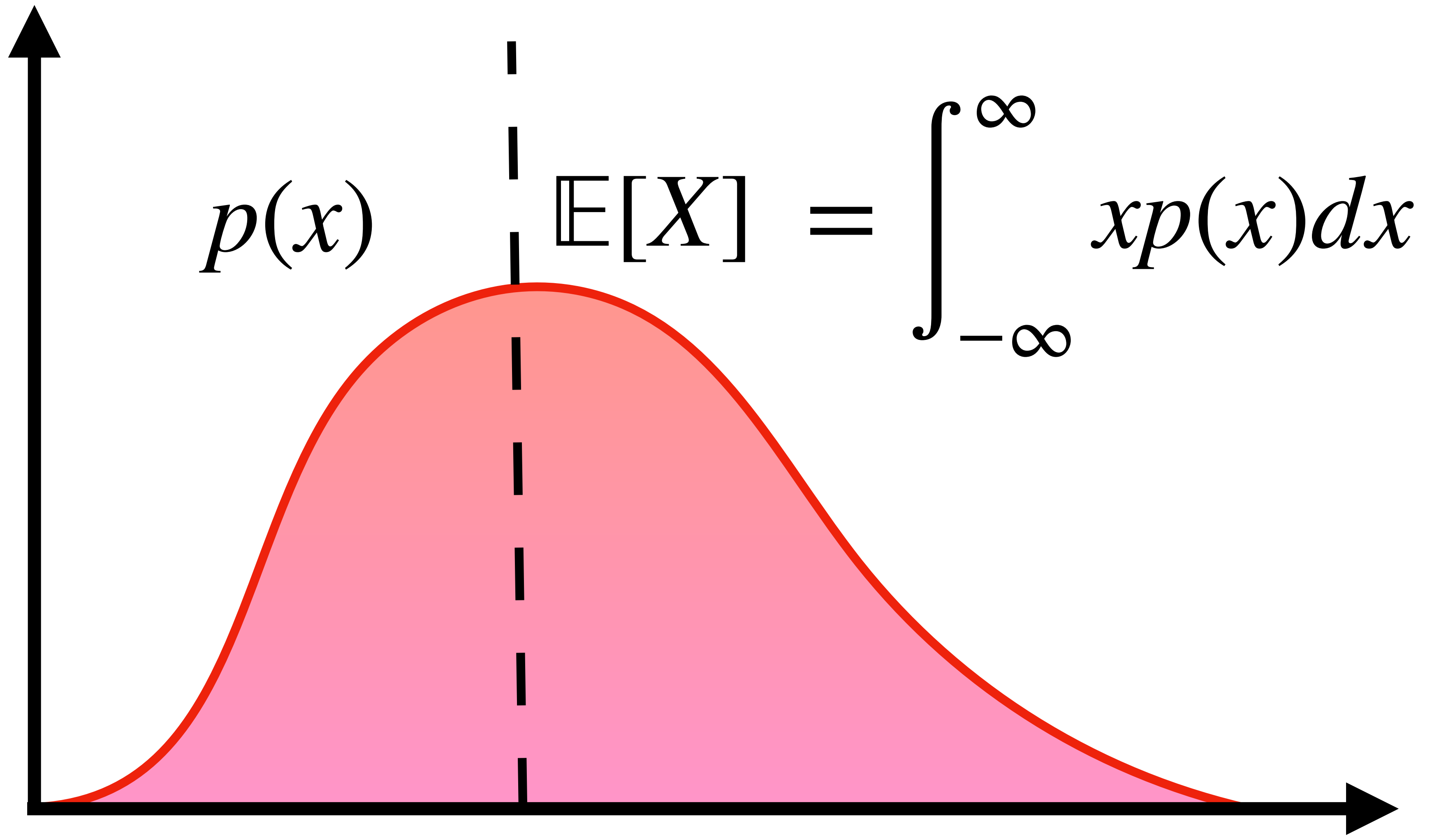


Probability density function

Probability density function

$x \sim X \quad \in [0,1)$

Sampling



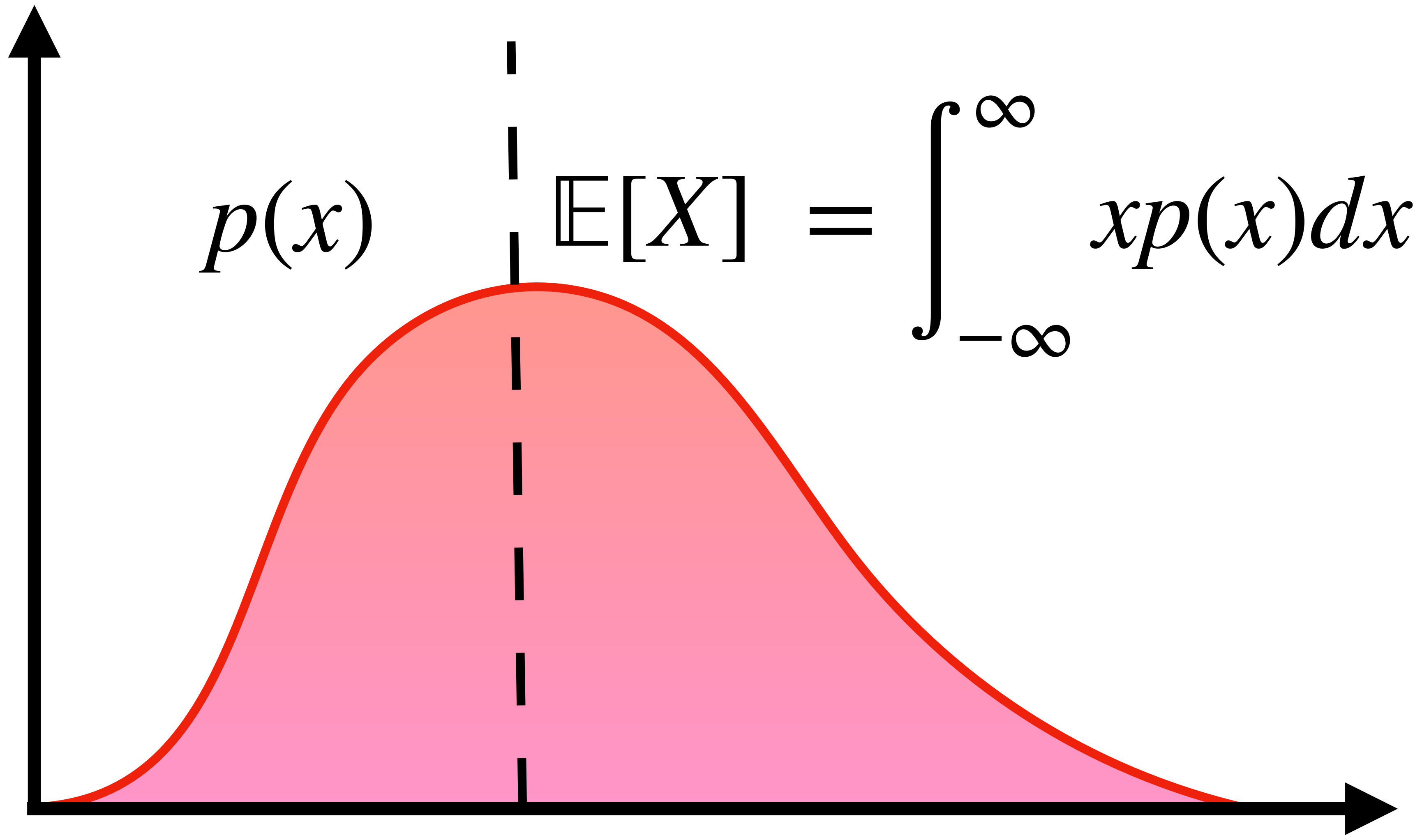
Probability density function

Probability density function

$x \sim X \in [0,1)$

Sampling

Non-negative: $p(x) \geq 0$

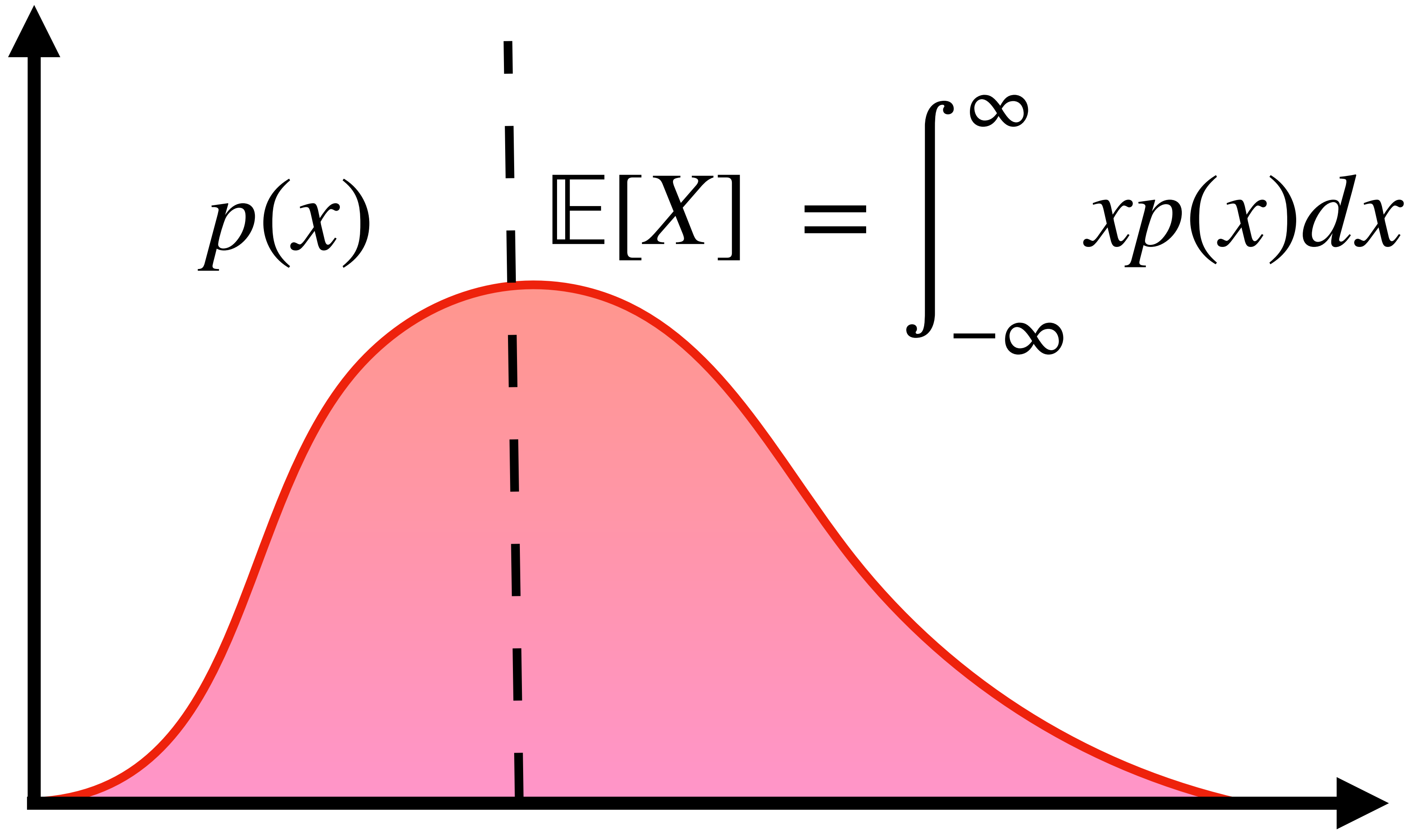


Probability density function

Probability density function

$$x \sim X \quad \in [0,1)$$

Sampling

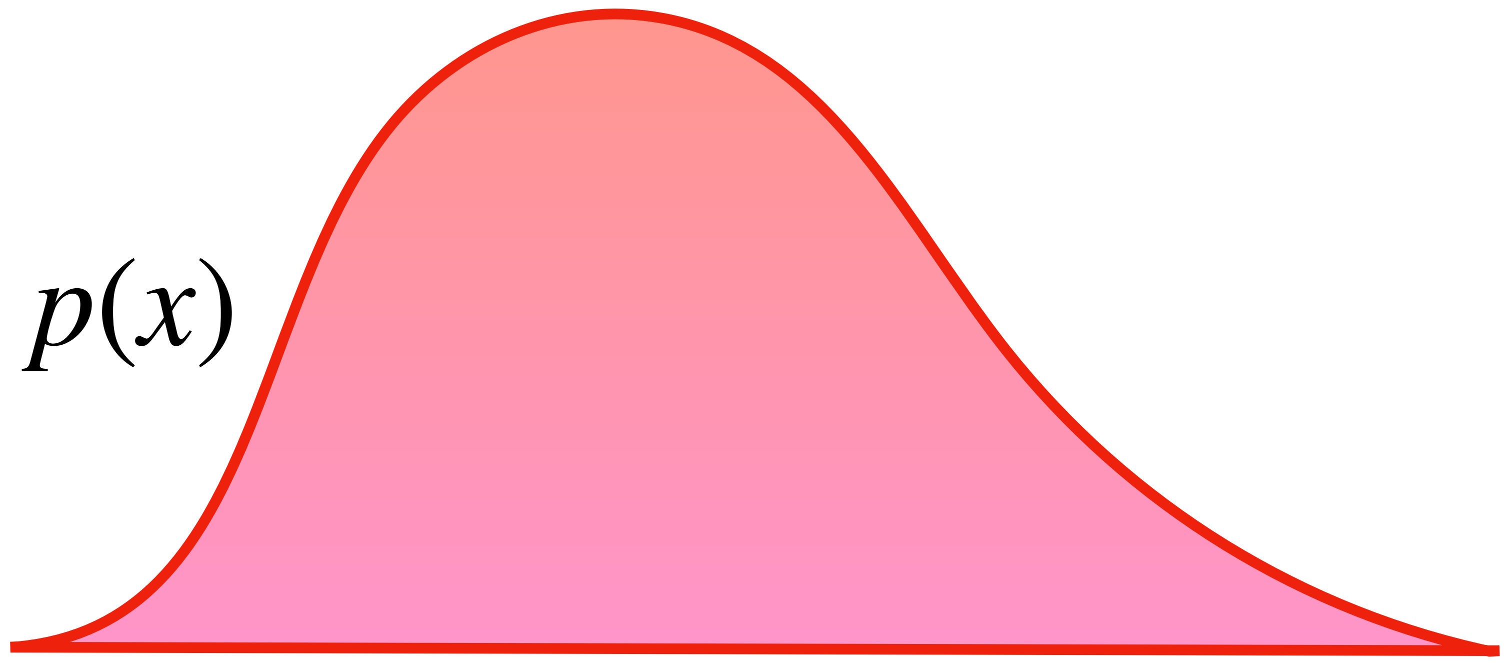


Probability density function

Non-negative: $p(x) \geq 0$

Normalized pdf: $\int p(x)dx = 1$

Joint probability distributions



$x \sim X \quad \in [0,1)$

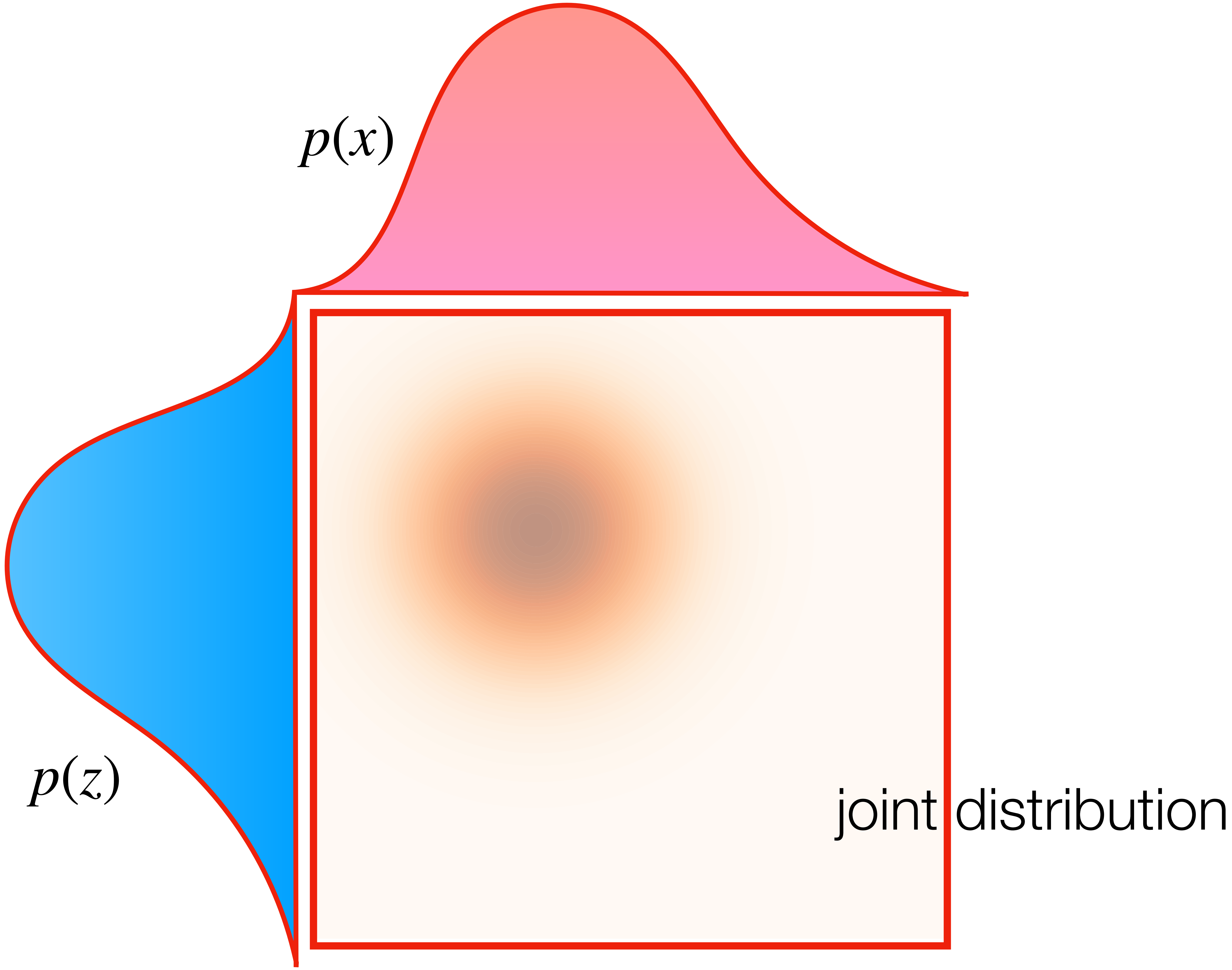
Sampling

$p(z)$

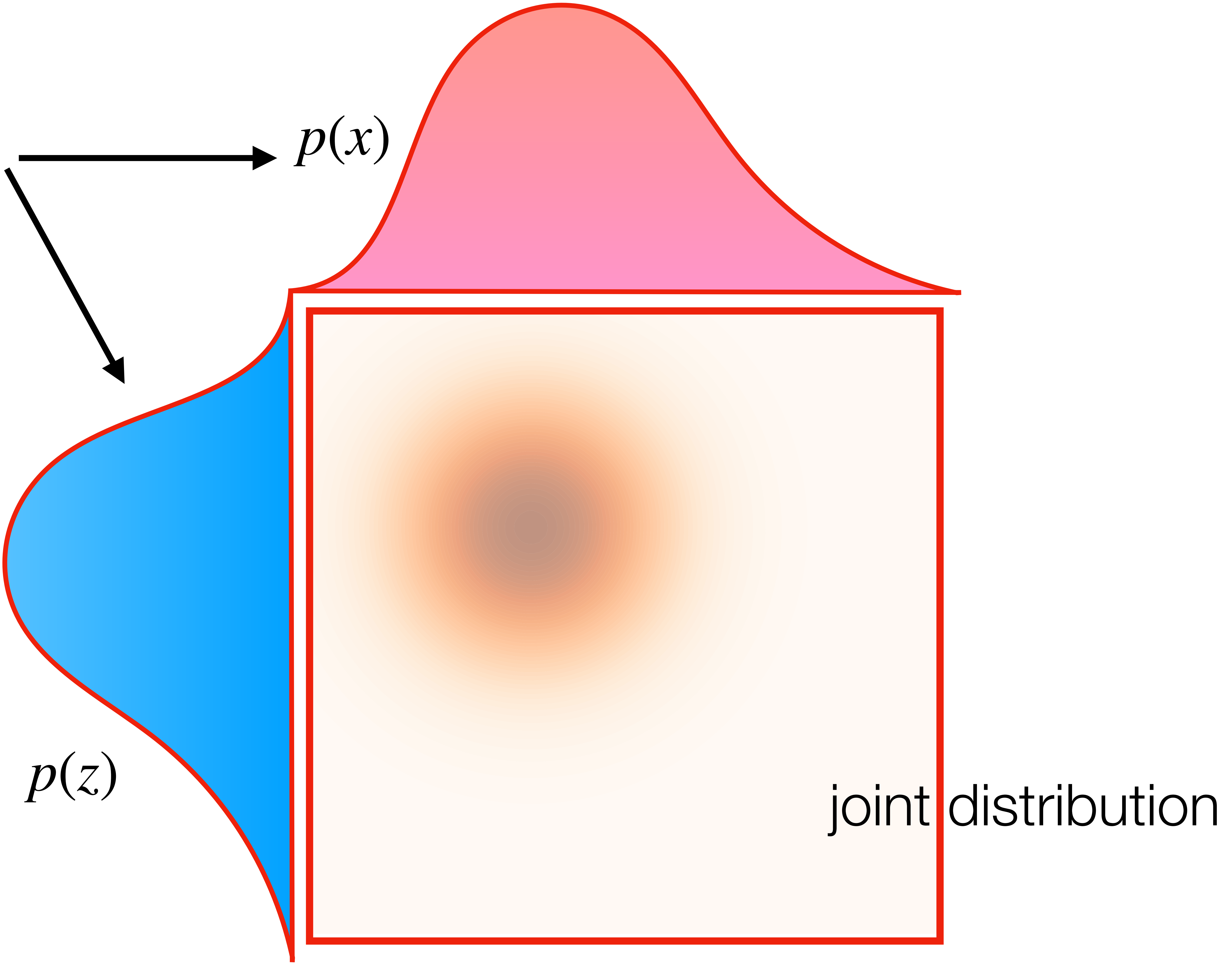
Joint probability distributions

$x \sim X \quad \in [0,1)$

Sampling



Joint probability distributions

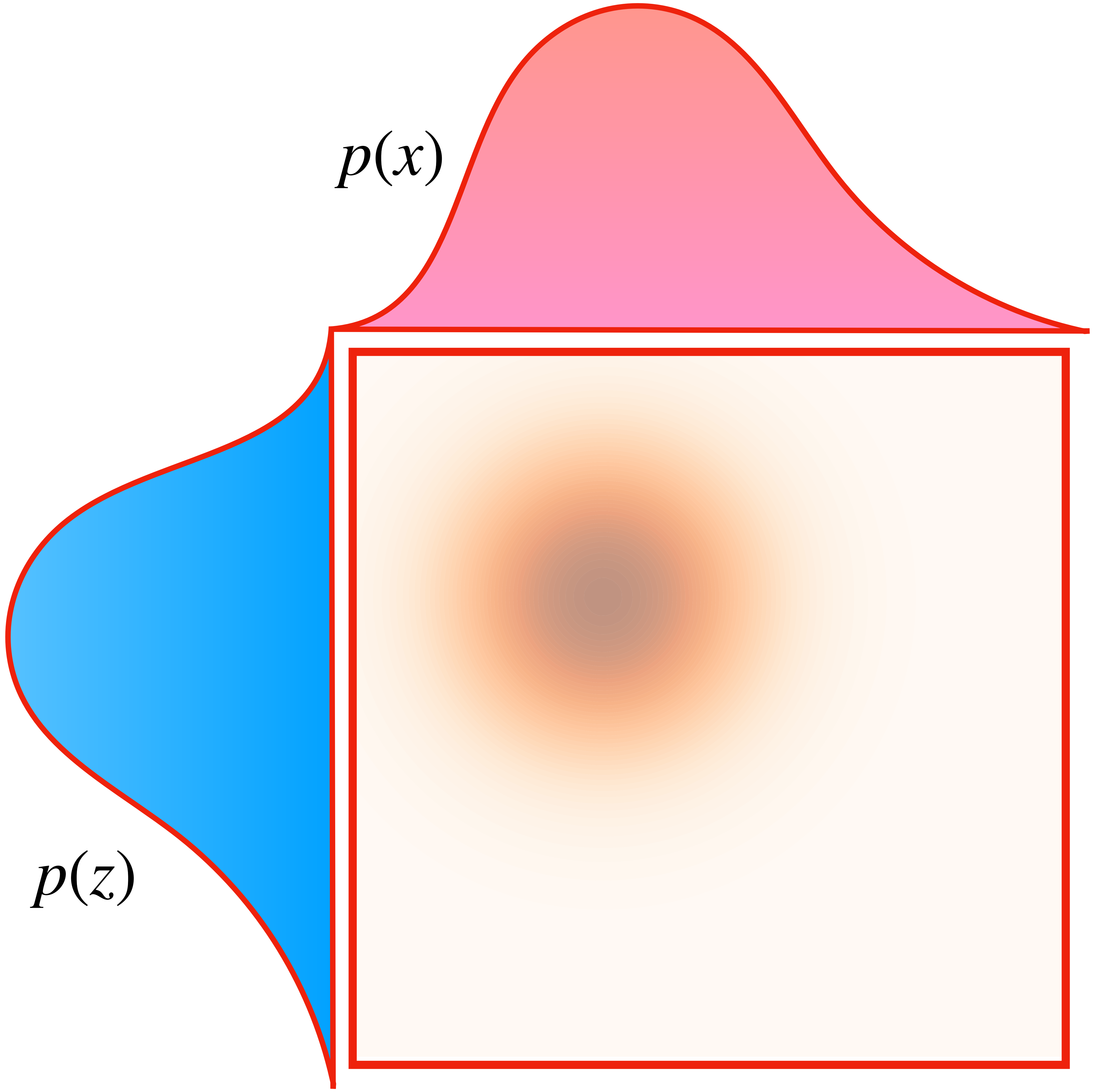


$x \sim X \;\; \in [0,1)$

Sampling

Joint probability distributions: Marginalization

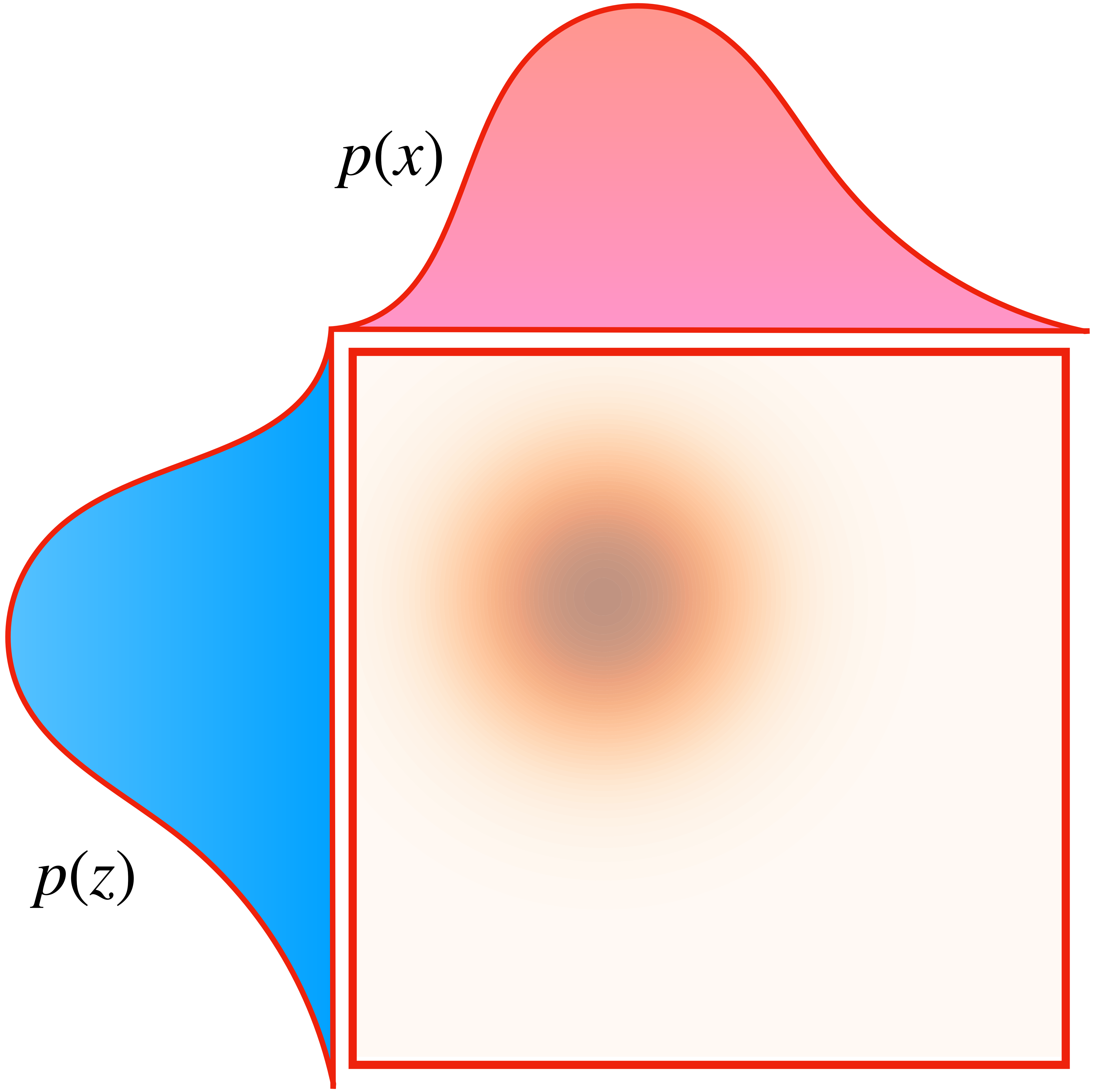
Marginalization



Joint probability distributions: Marginalization

$$p(x) = \int p(x, z)dz$$

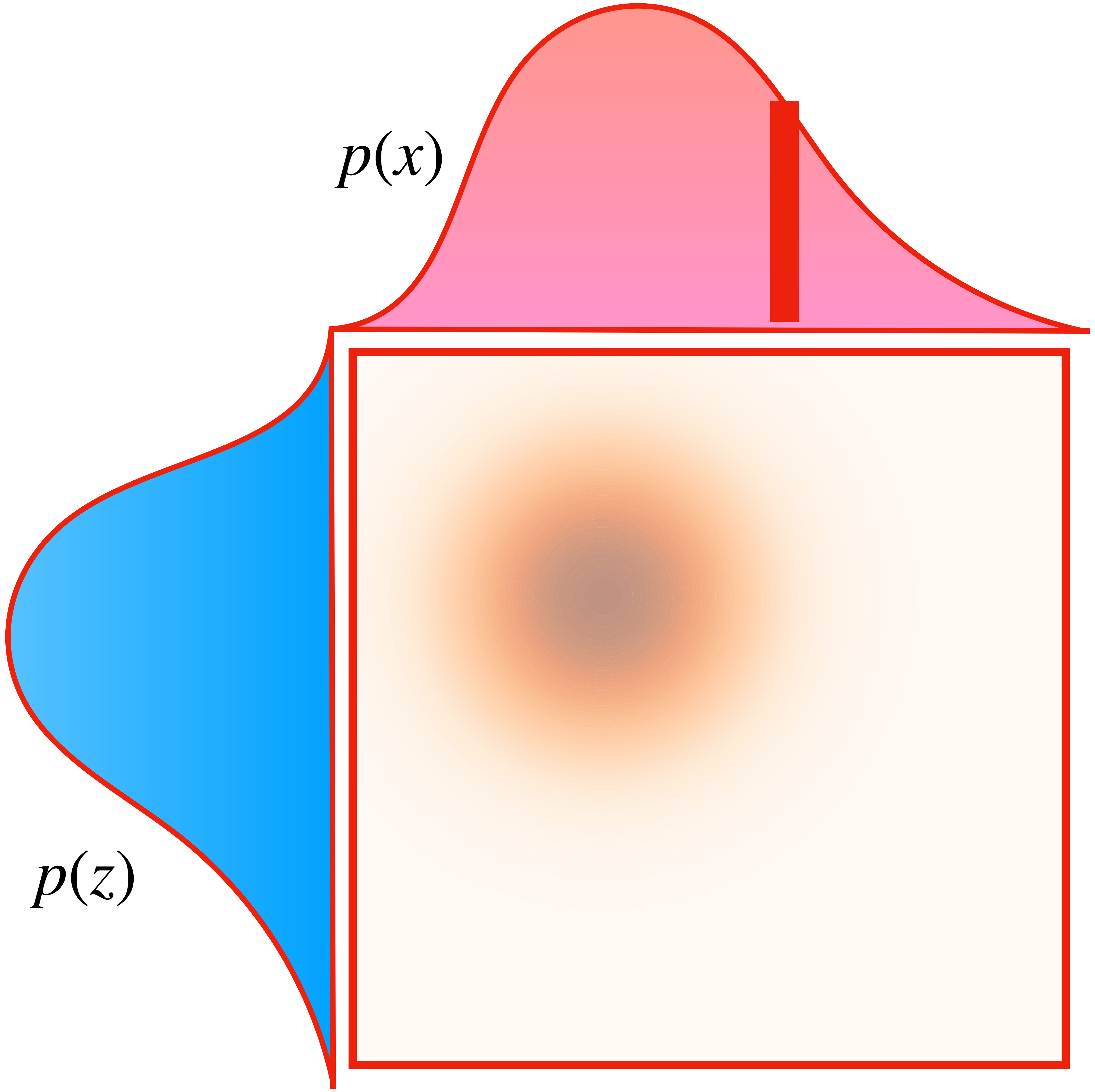
Marginalization



Joint probability distributions: Marginalization

$$p(x) = \int p(x, z) dz$$

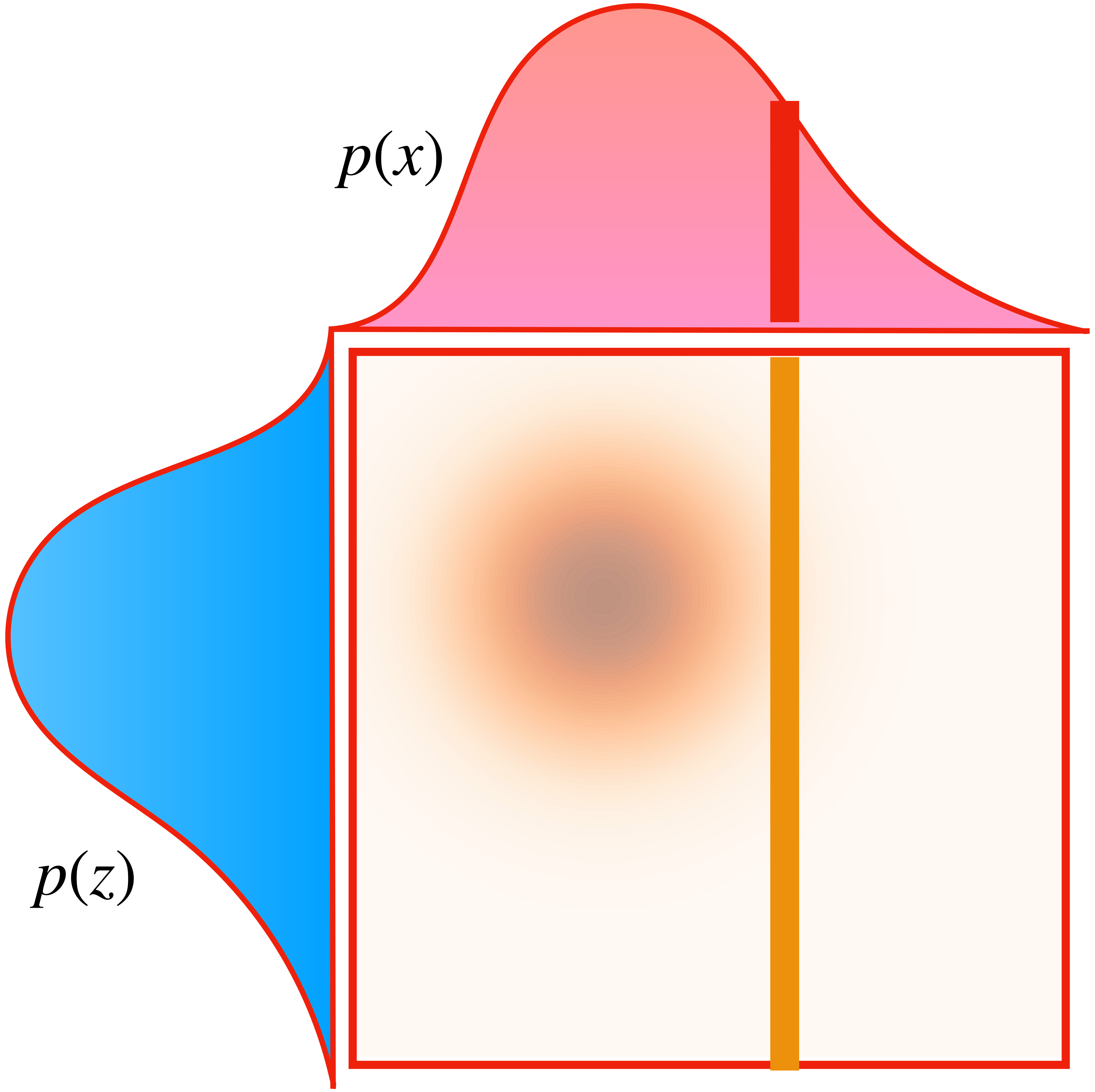
Marginalization



Joint probability distributions: Marginalization

$$p(x) = \int p(x, z)dz$$

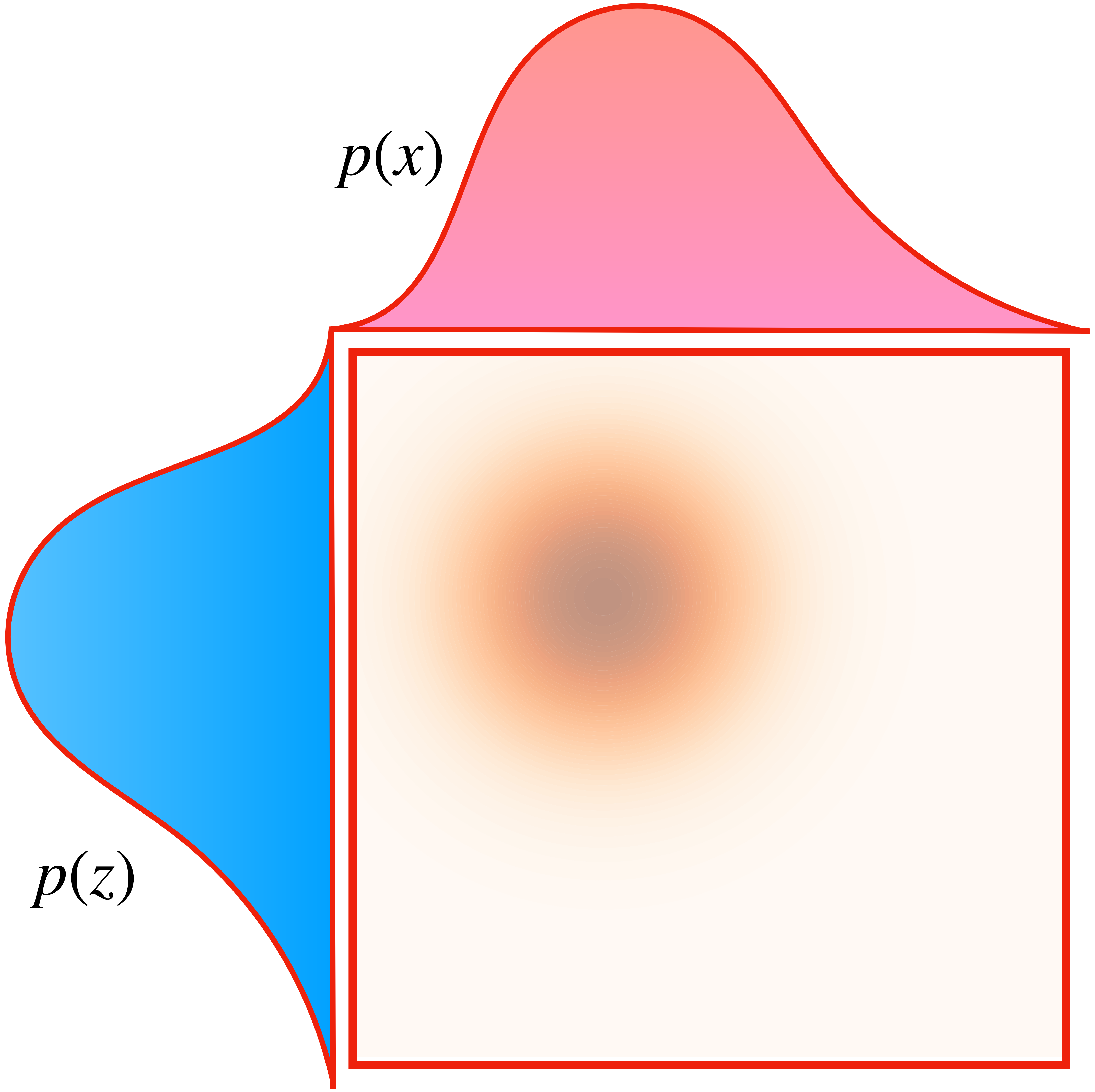
Marginalization



Joint probability distributions: Marginalization

$$p(x) = \int p(x, z) dz$$

Marginalization

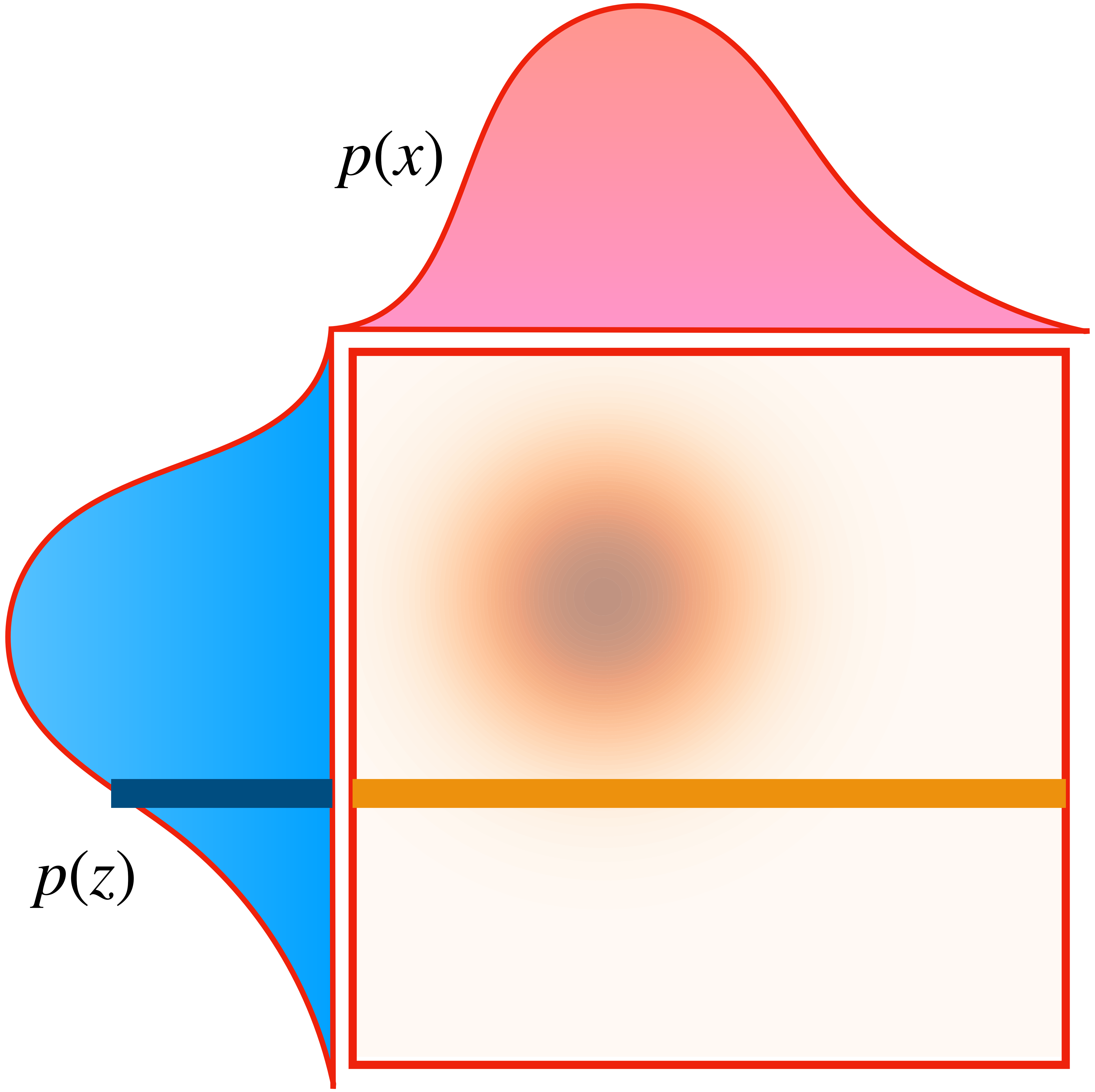


Joint probability distributions: Marginalization

$$p(x) = \int p(x, z)dz$$

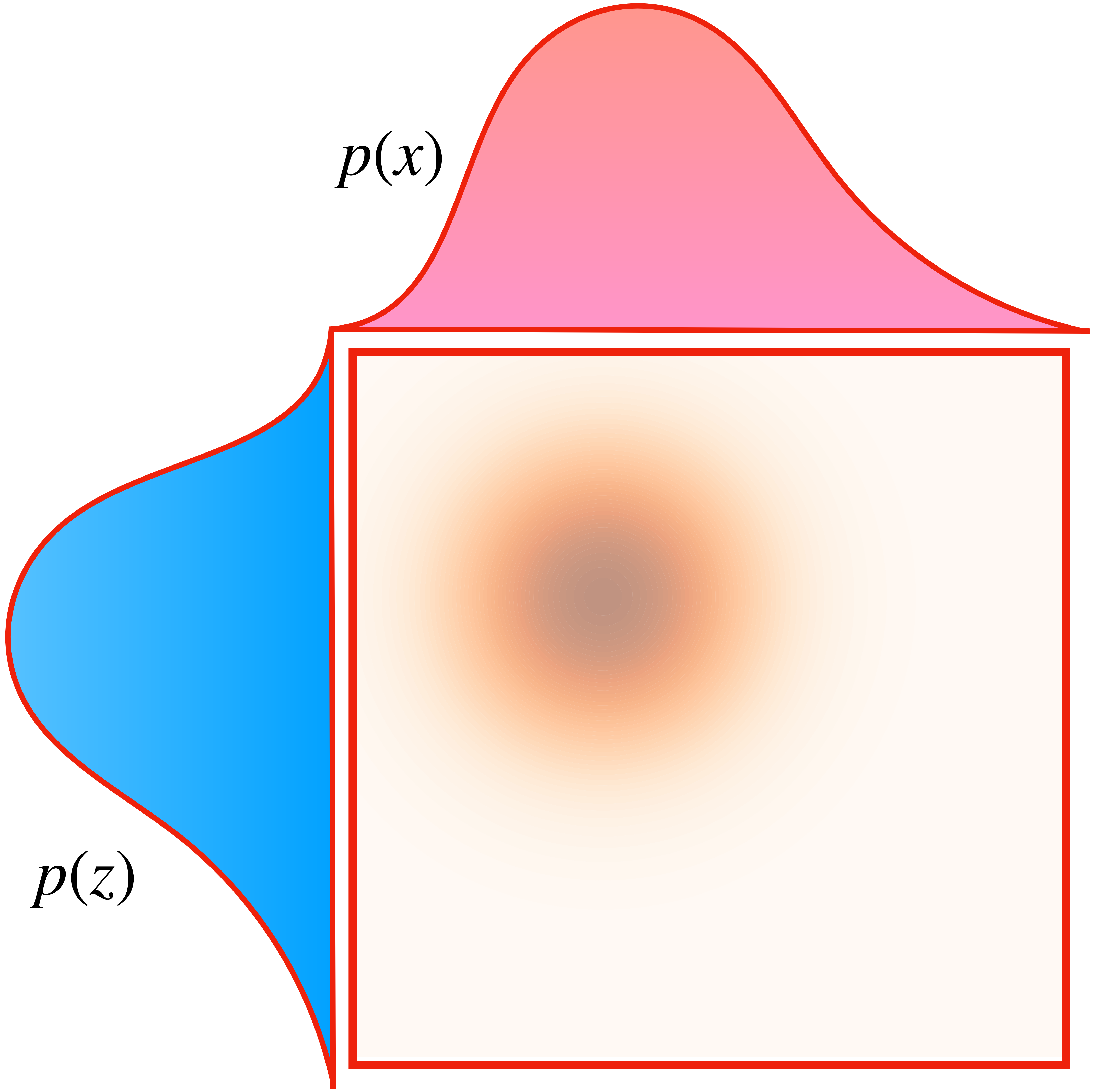
Marginalization

$$p(z) = \int p(x, z)dx$$



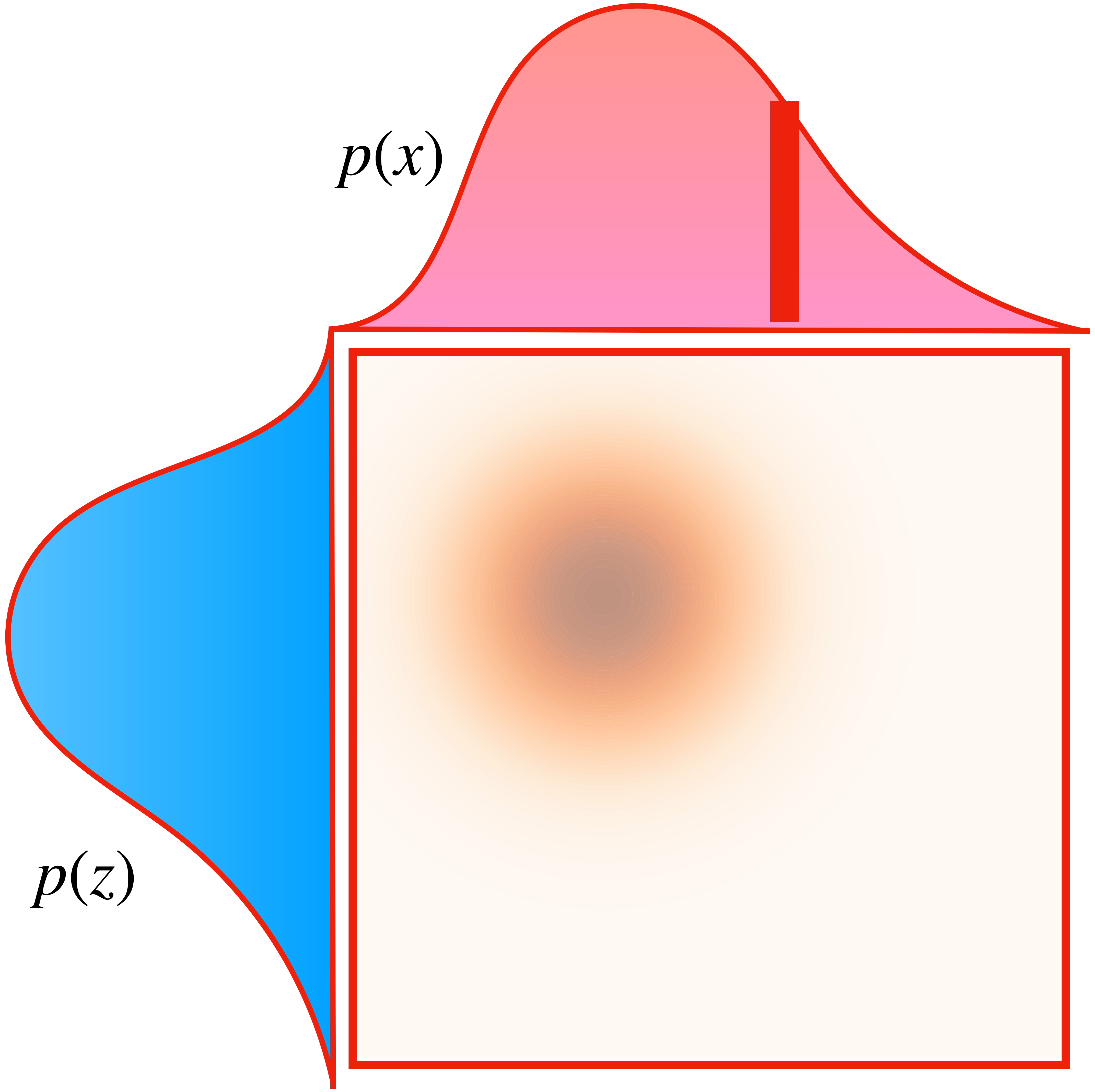
Conditional probability distributions

$$p(x \mid z) = \frac{p(x,z)}{p(z)}$$



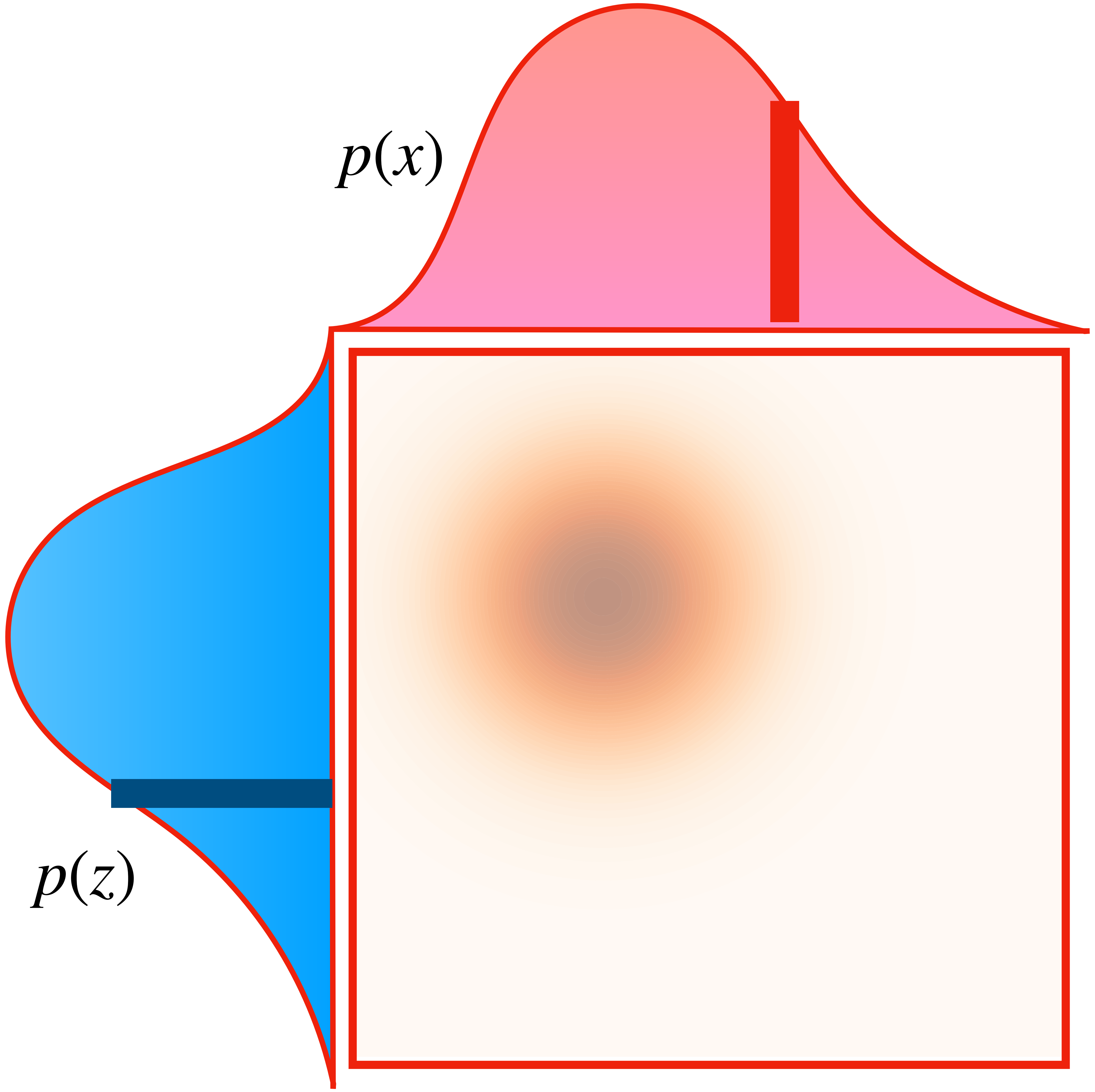
Conditional probability distributions

$$p(x\,|\,z) = \frac{p(x,z)}{p(z)}$$



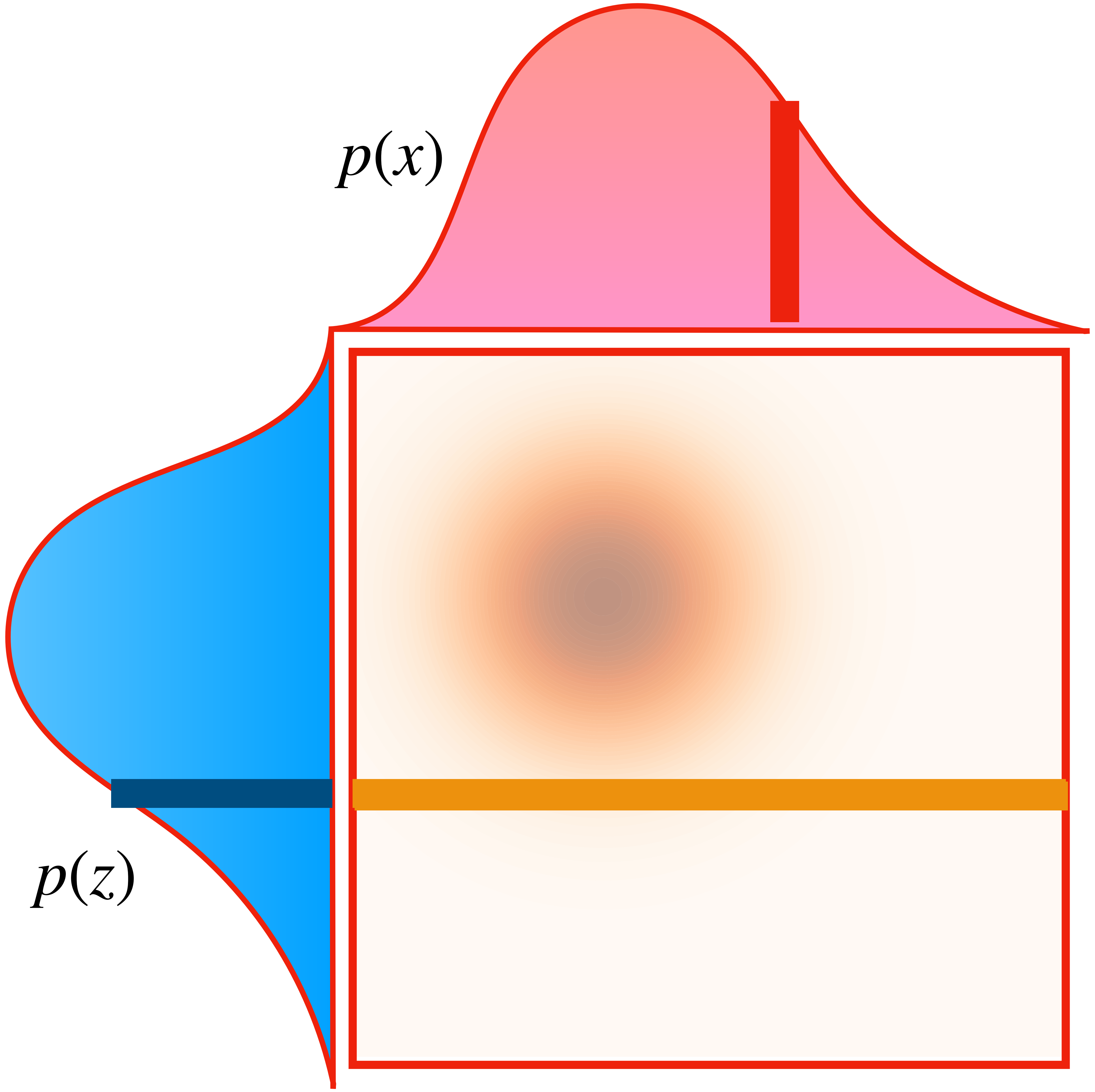
Conditional probability distributions

$$p(x \mid z) = \frac{p(x, z)}{p(z)}$$



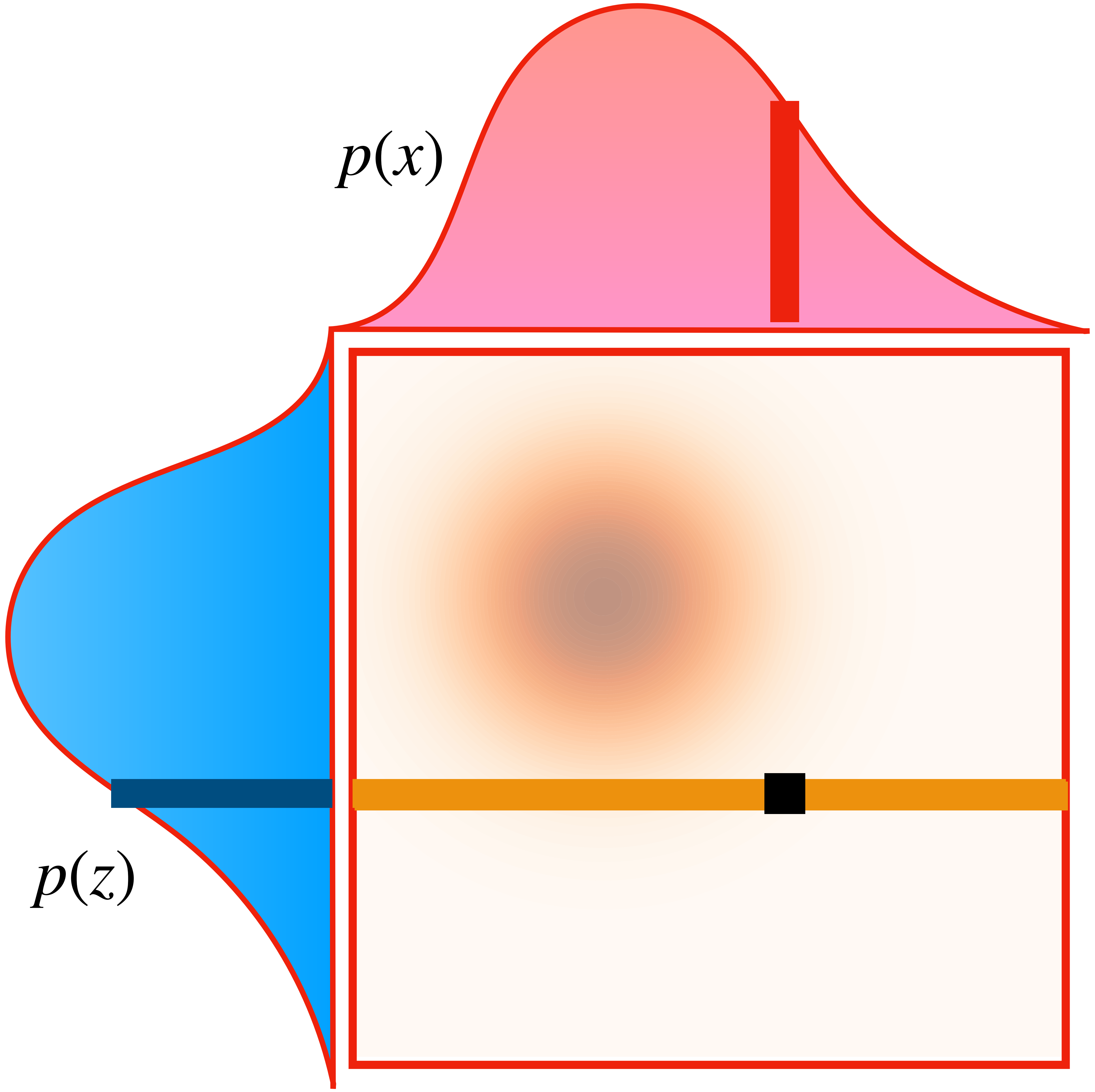
Conditional probability distributions

$$p(x \mid z) = \frac{p(x, z)}{p(z)}$$



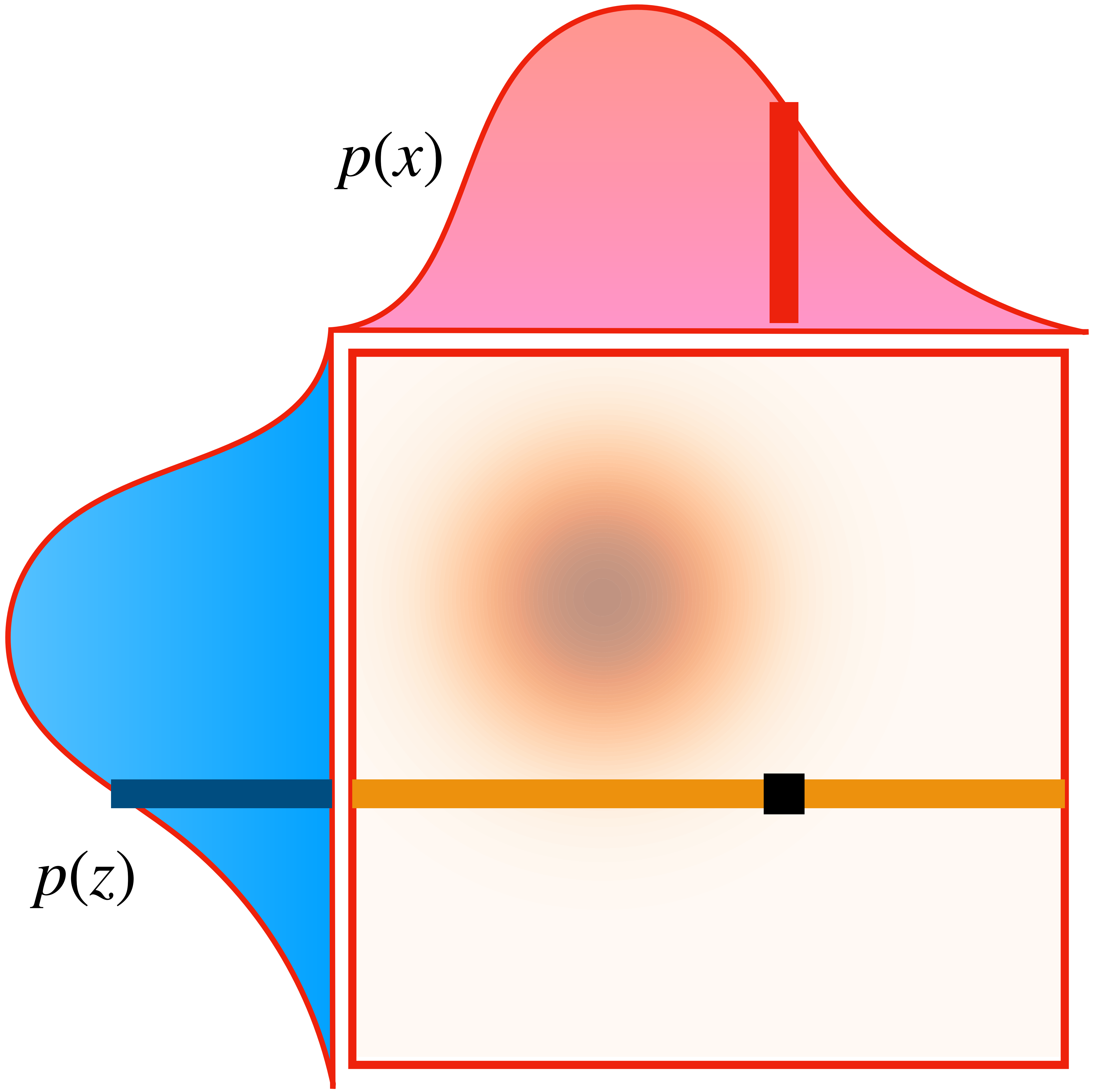
Conditional probability distributions

$$p(x \mid z) = \frac{p(x, z)}{p(z)}$$



Conditional probability distributions

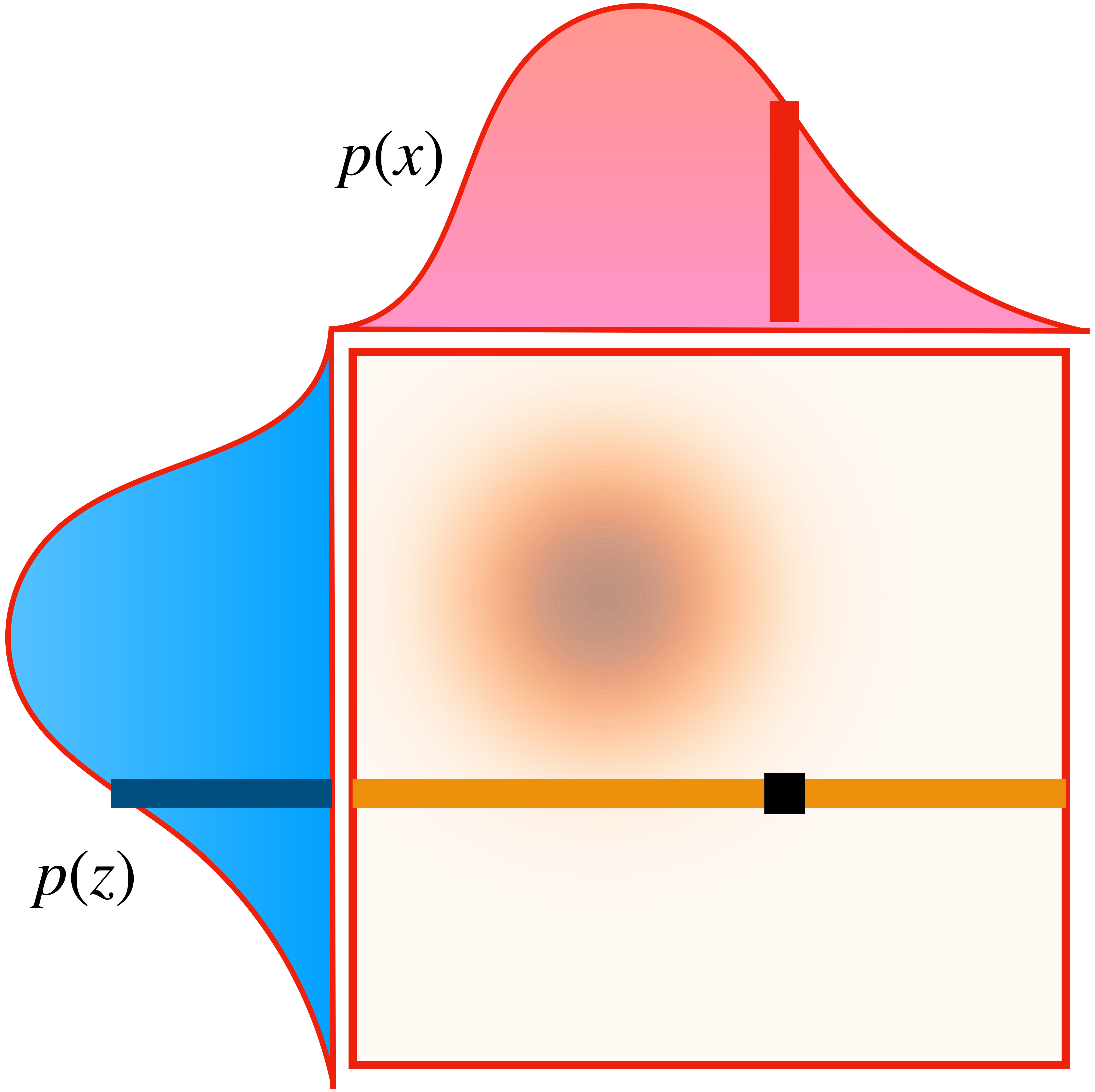
$$p(x \mid z) = \frac{p(x, z)}{p(z)}$$



Conditional probability distributions

$$p(x\,|\,z) = \frac{p(x,z)}{p(z)}$$

Likelihood of the data given the latent variable z

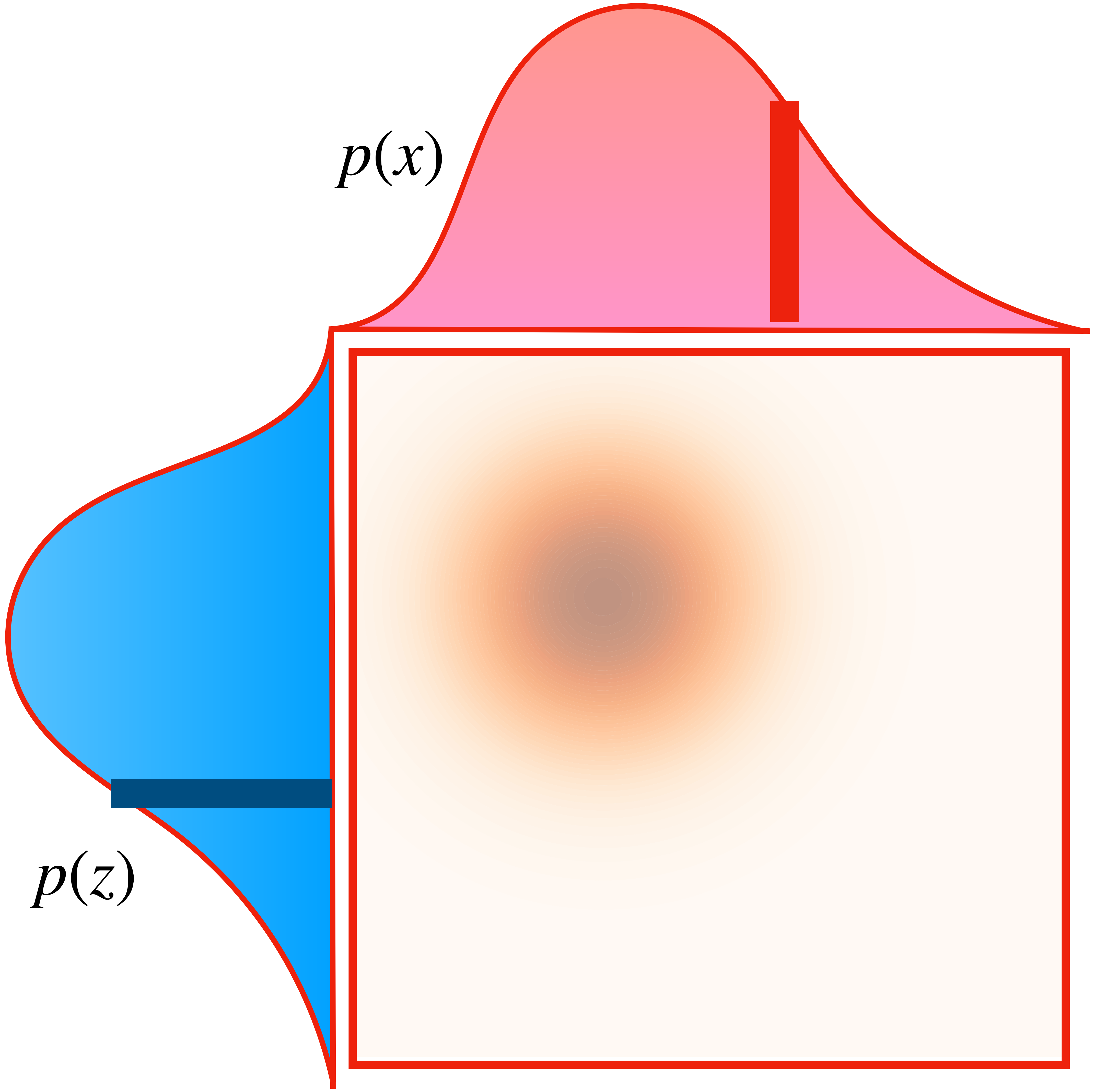


Conditional probability distributions

$$p(x\,|\,z) = \frac{p(x,z)}{p(z)}$$

Likelihood of the data
given the latent variable z

$$p(z\,|\,x) = \frac{p(x,z)}{p(x)}$$

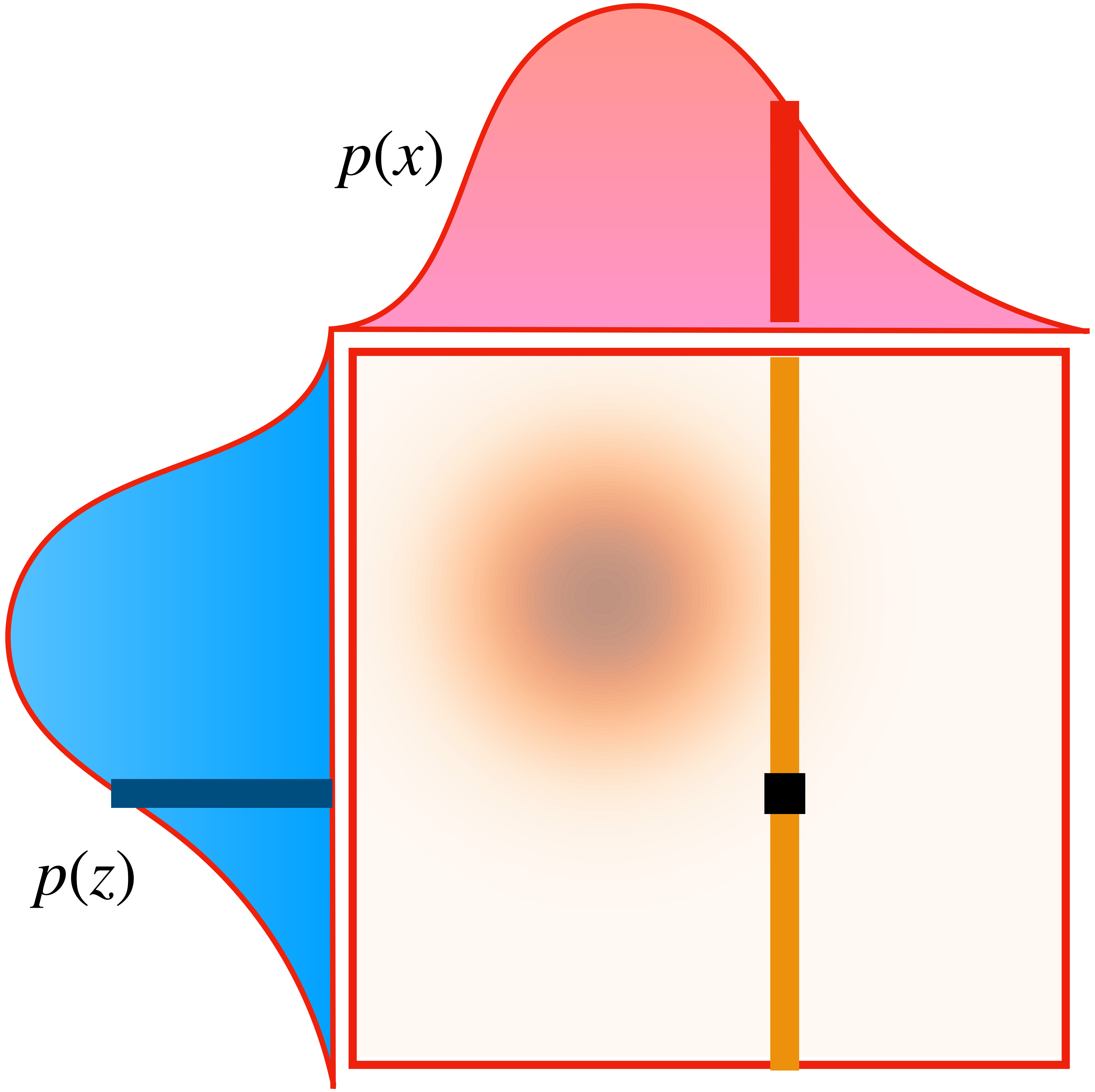


Conditional probability distributions

$$p(x \mid z) = \frac{p(x, z)}{p(z)}$$

Likelihood of the data
given the latent variable z

$$p(z \mid x) = \frac{p(x, z)}{p(x)}$$



Bayes Theorem

Classification

Problem statement

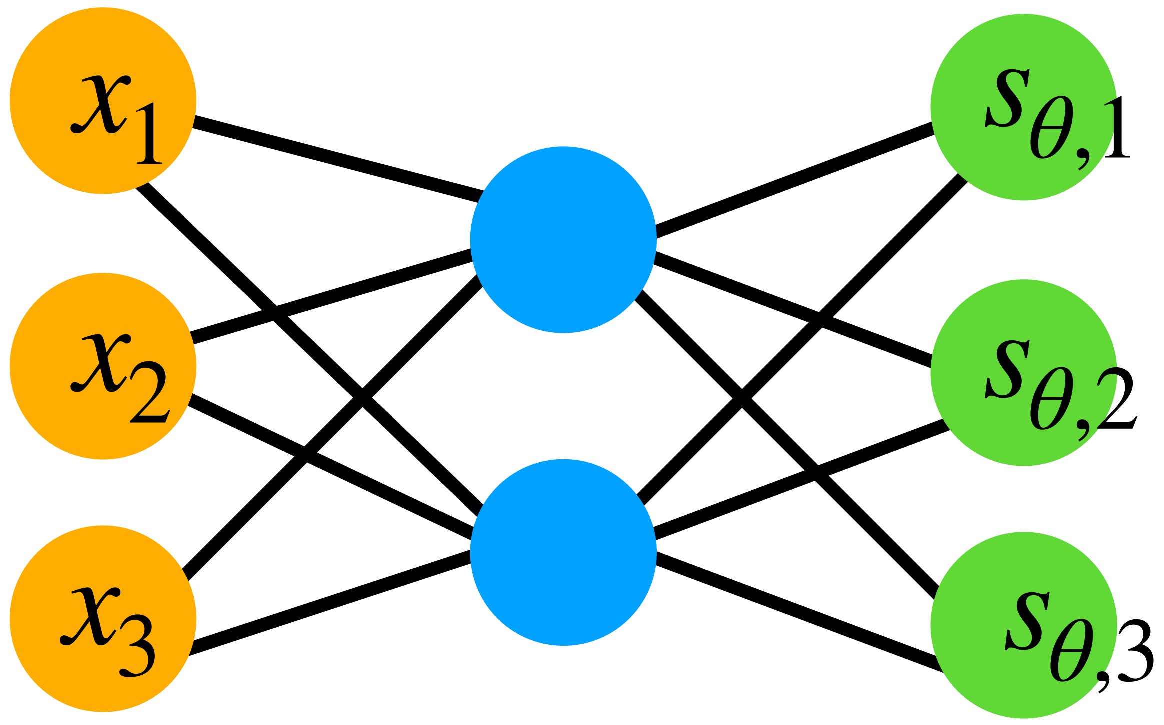
Hypothesis θ

Data

Classification

Problem statement

Data

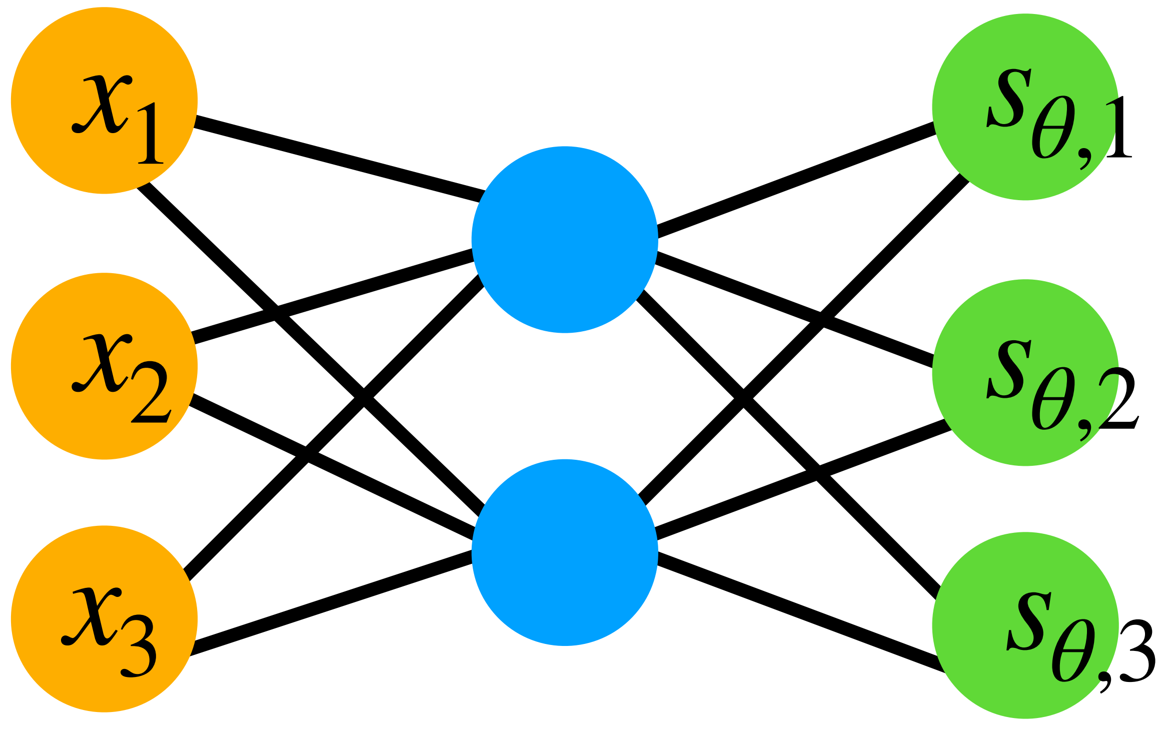


Hypothesis θ

Classification

Problem statement

Data



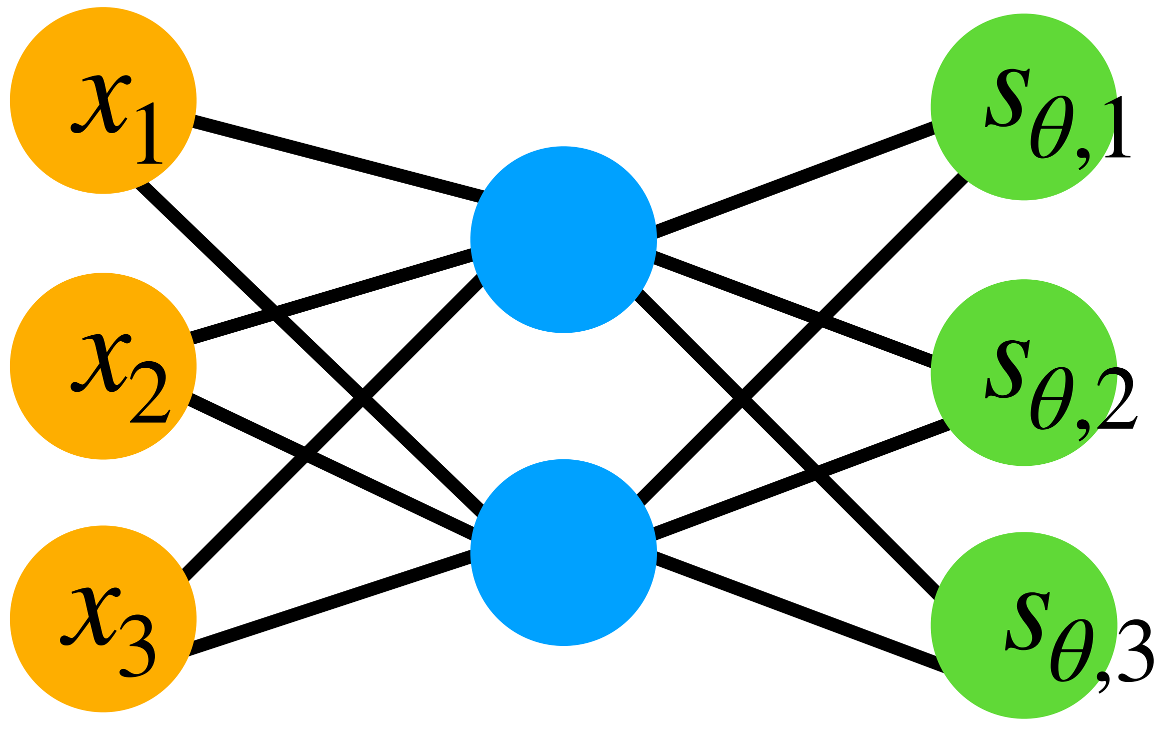
Hypothesis θ

$$\begin{bmatrix} 0.09 \\ 0.69 \\ 0.14 \end{bmatrix}$$

Classification

Problem statement

Data



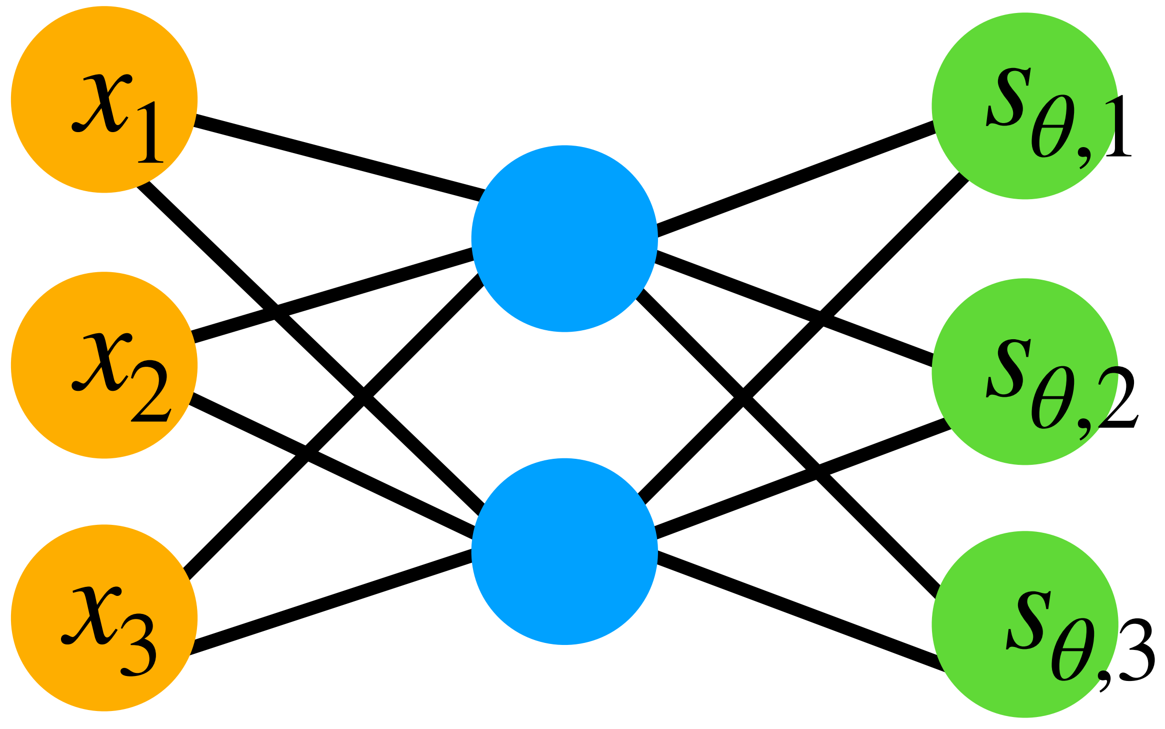
Hypothesis θ

$$\begin{bmatrix}0.09\\0.69\\0.14\end{bmatrix}$$

Classification

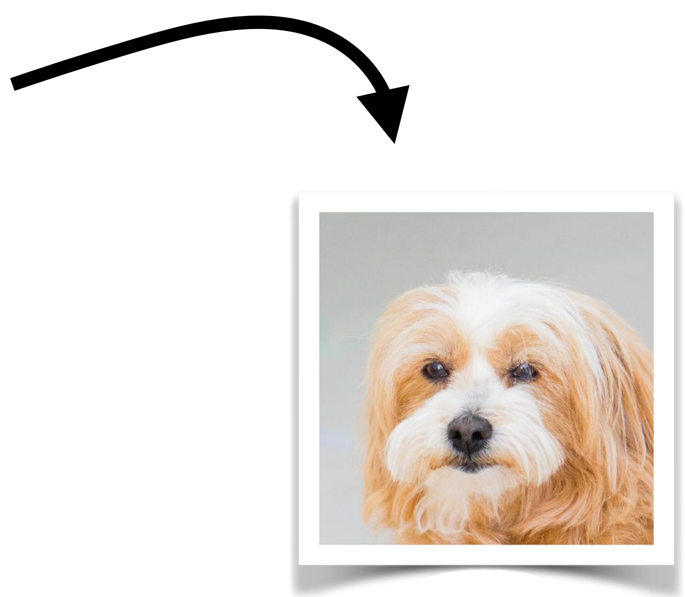
Problem statement

Data



Hypothesis θ

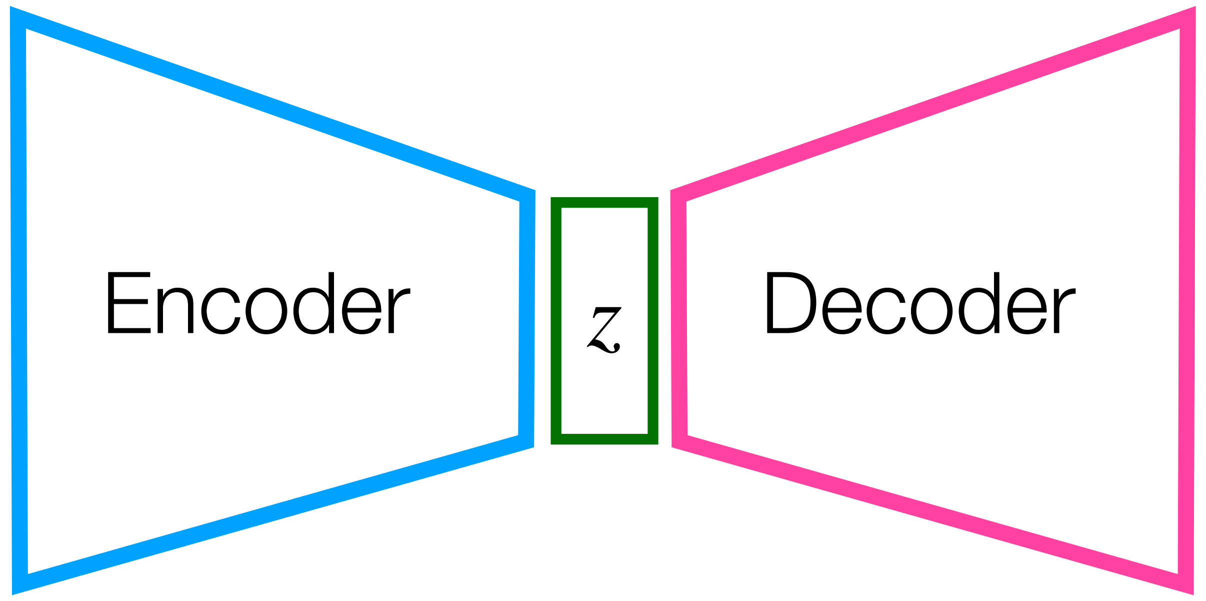
$$\begin{bmatrix} 0.09 \\ 0.69 \\ 0.14 \end{bmatrix}$$



Classification

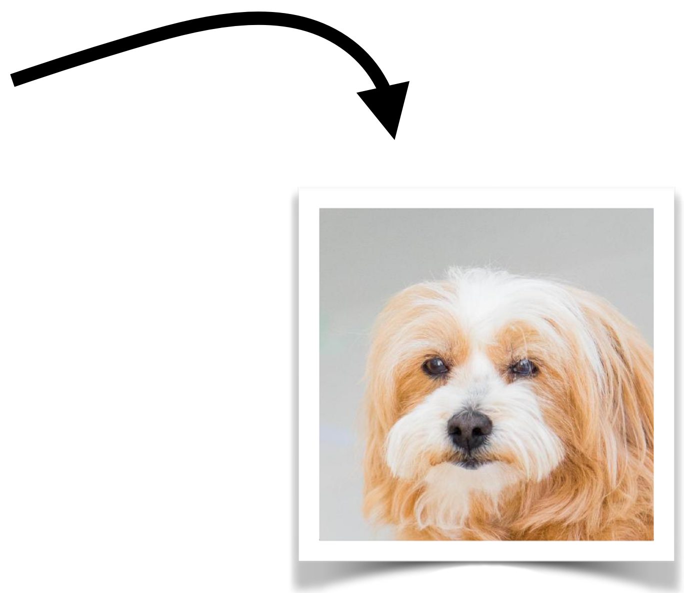
Problem statement

Data



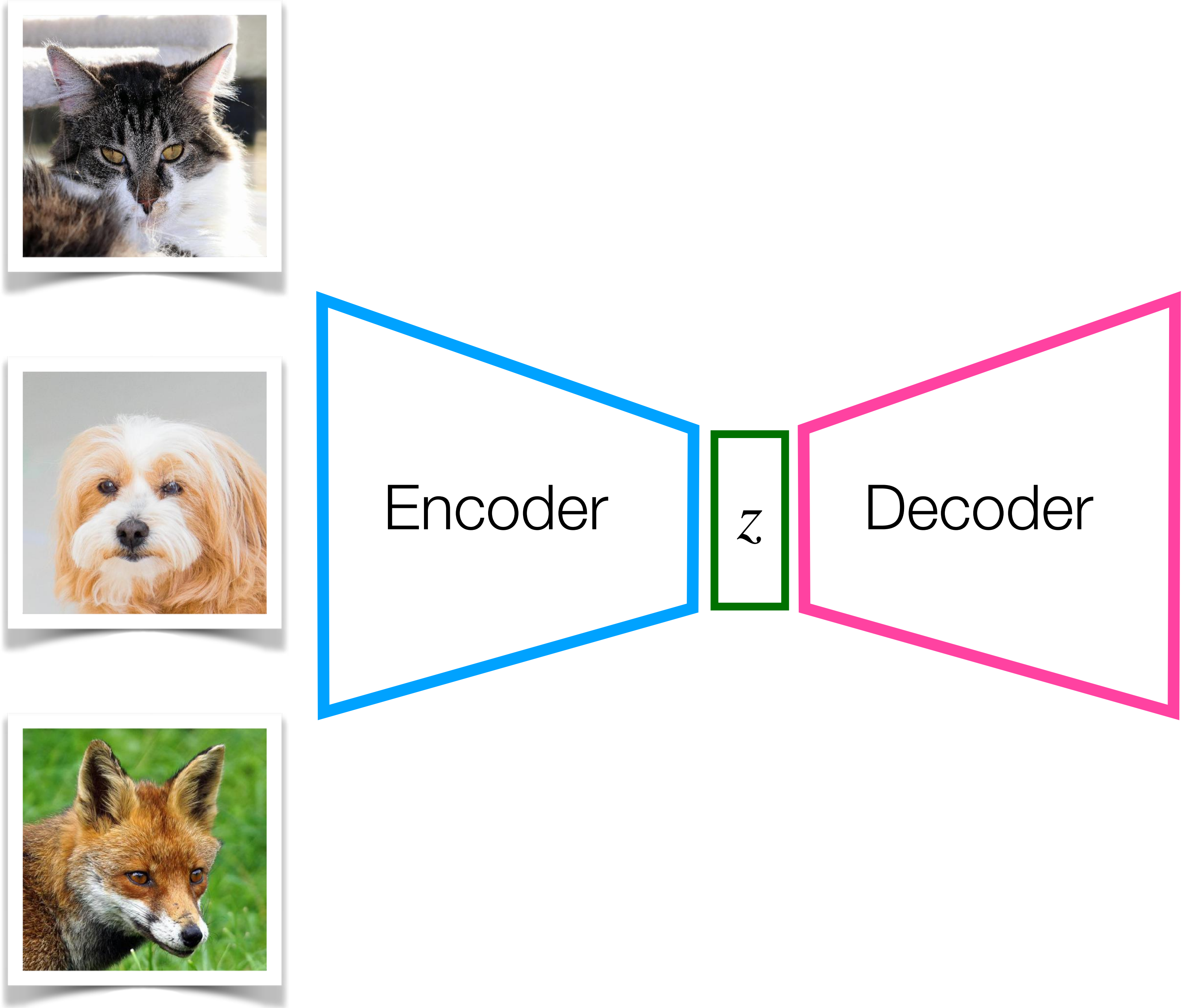
Hypothesis

$$\begin{bmatrix}0.09\\ \boxed{0.69}\\ 0.14\end{bmatrix}$$



Bayes Theorem

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}$$



Encoder

z

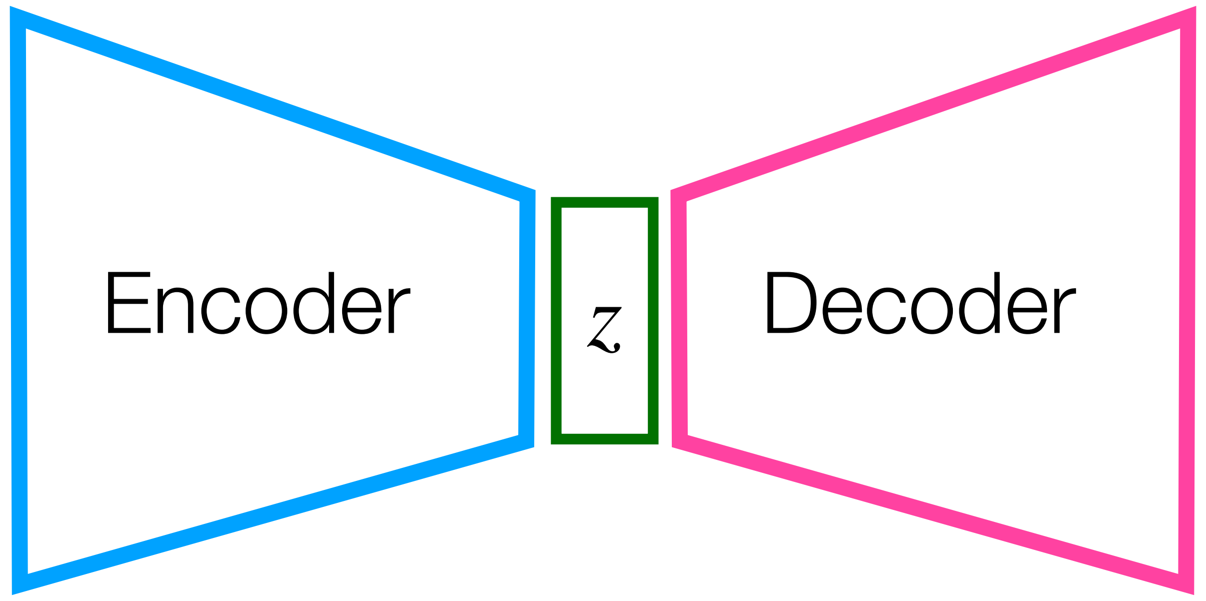
Decoder

Bayes Theorem

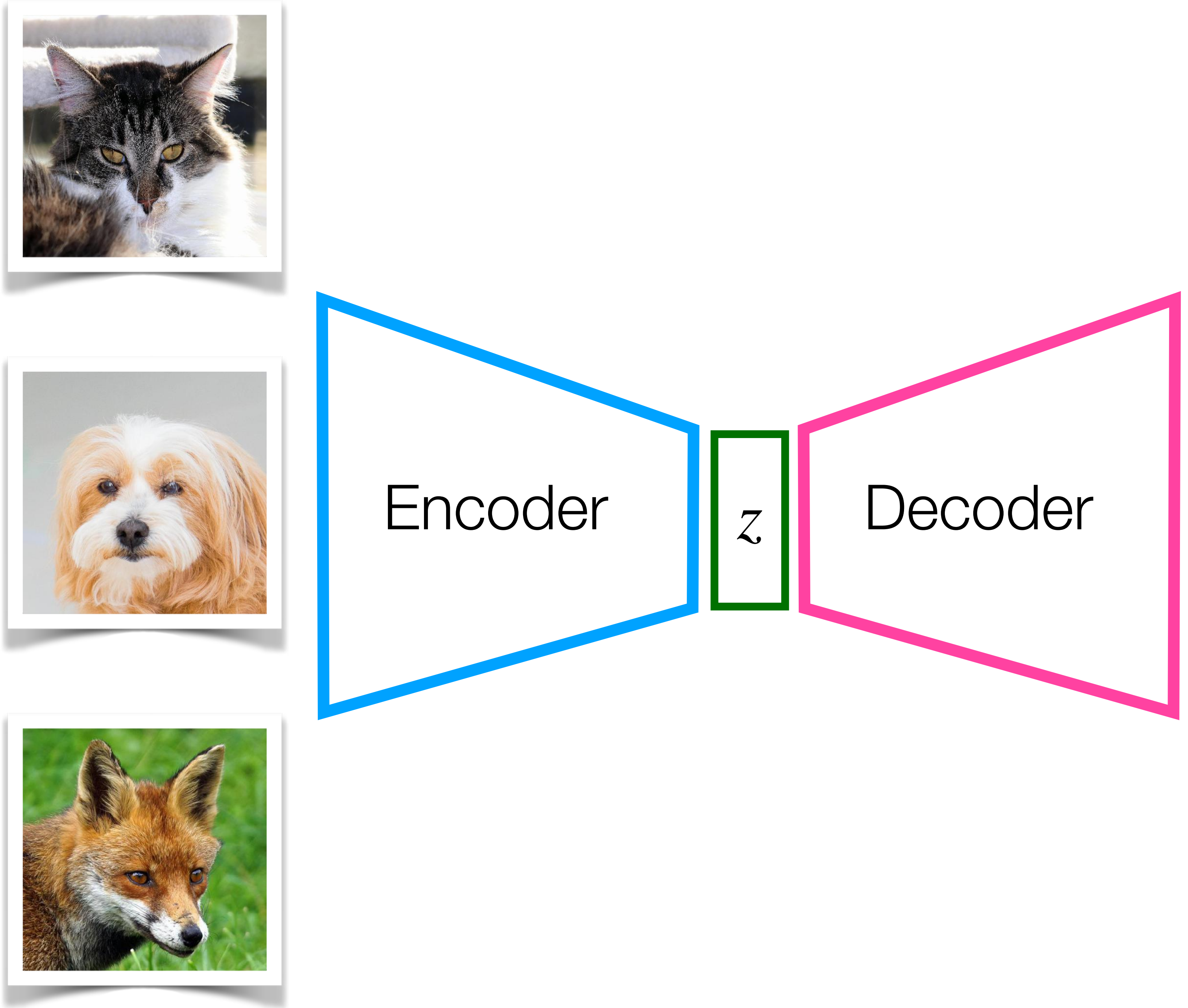
$$p(z \mid x) = \frac{p(x \mid z)p(z)}{p(x)}$$

z : Hypothesis/Latent

x : Data



Bayes Theorem



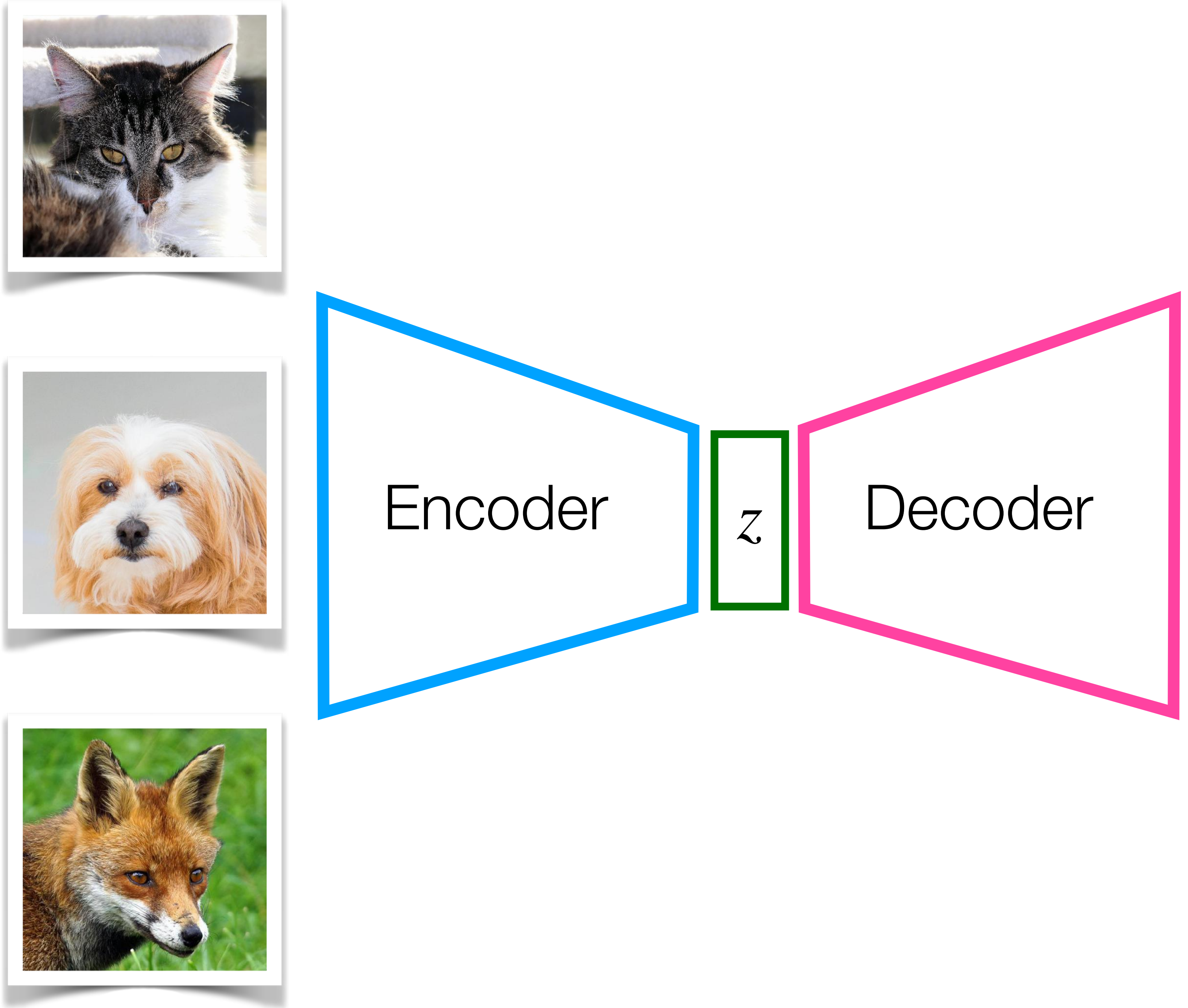
$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}$$

z : Hypothesis/Latent

x : Data

$p(x|z)$

Bayes Theorem

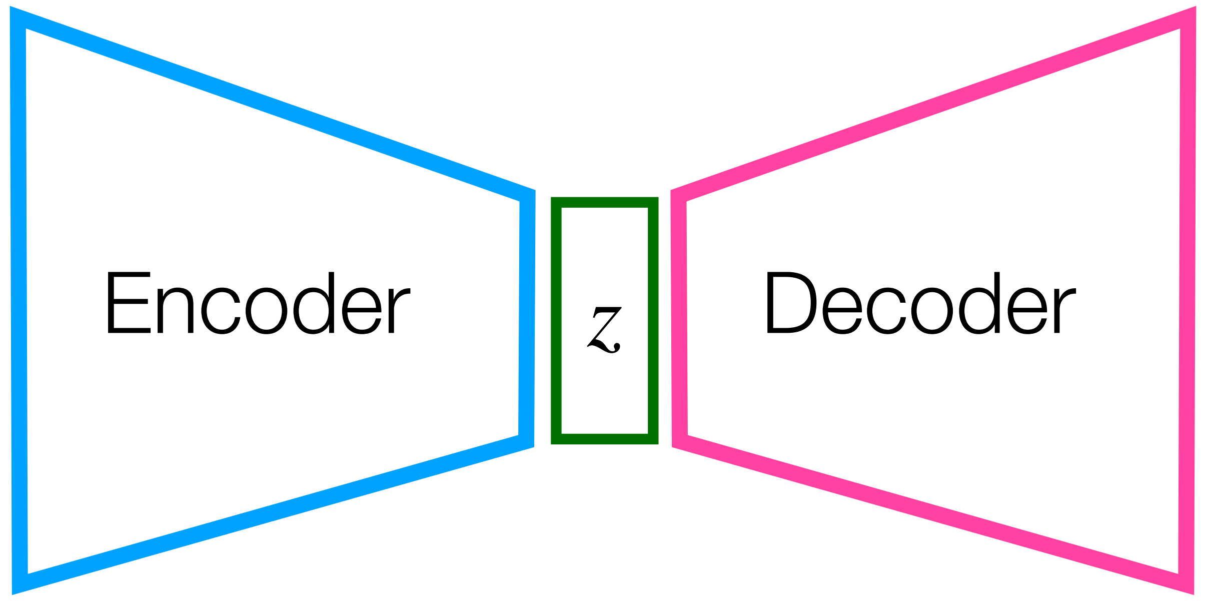


$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}$$

z : Hypothesis/Latent
x : Data

$p(x|z)$: the likelihood (given your hypothesis, what is the probabilty of observing x)

Bayes Theorem



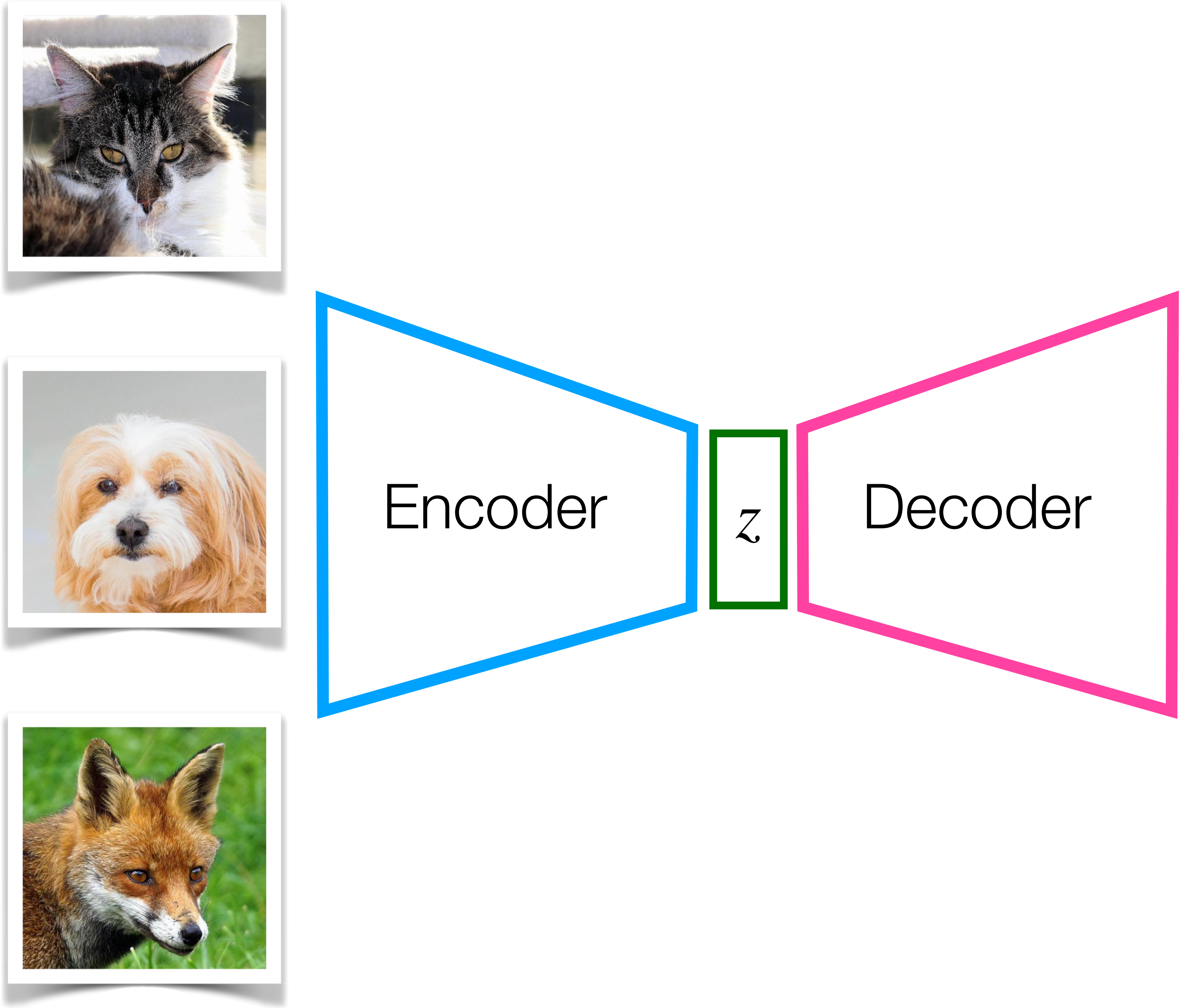
$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}$$

z : Hypothesis/Latent
x : Data

$p(x|z)$: the likelihood (given your hypothesis, what is the probabilty of observing x)

$p(z|x)$

Bayes Theorem



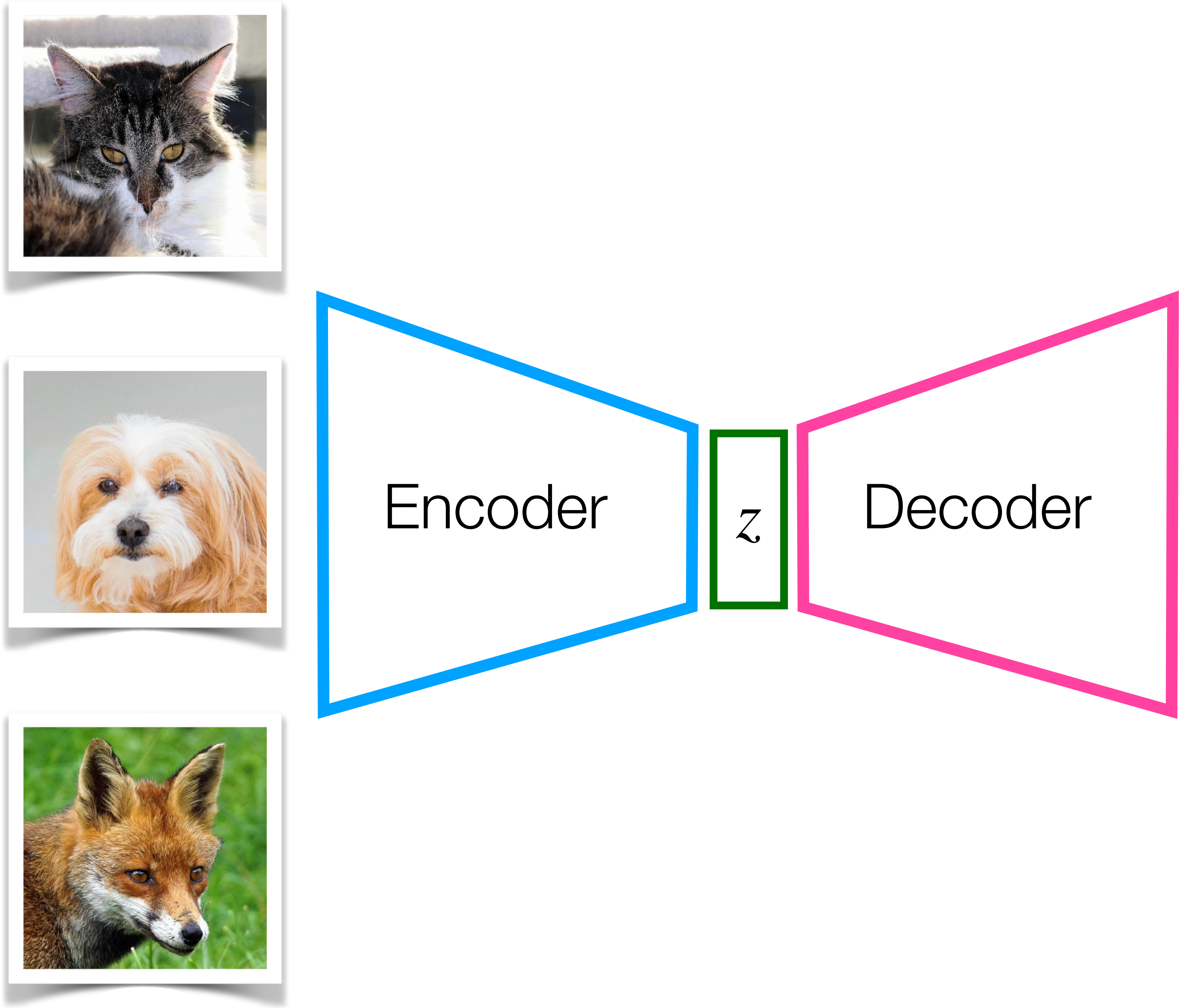
$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}$$

z : Hypothesis/Latent
x : Data

$p(x|z)$: the likelihood (given your hypothesis, what is the probabilty of observing x)

$p(z|x)$: the posterior (updating your belief based the data)

Bayes Theorem



$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}$$

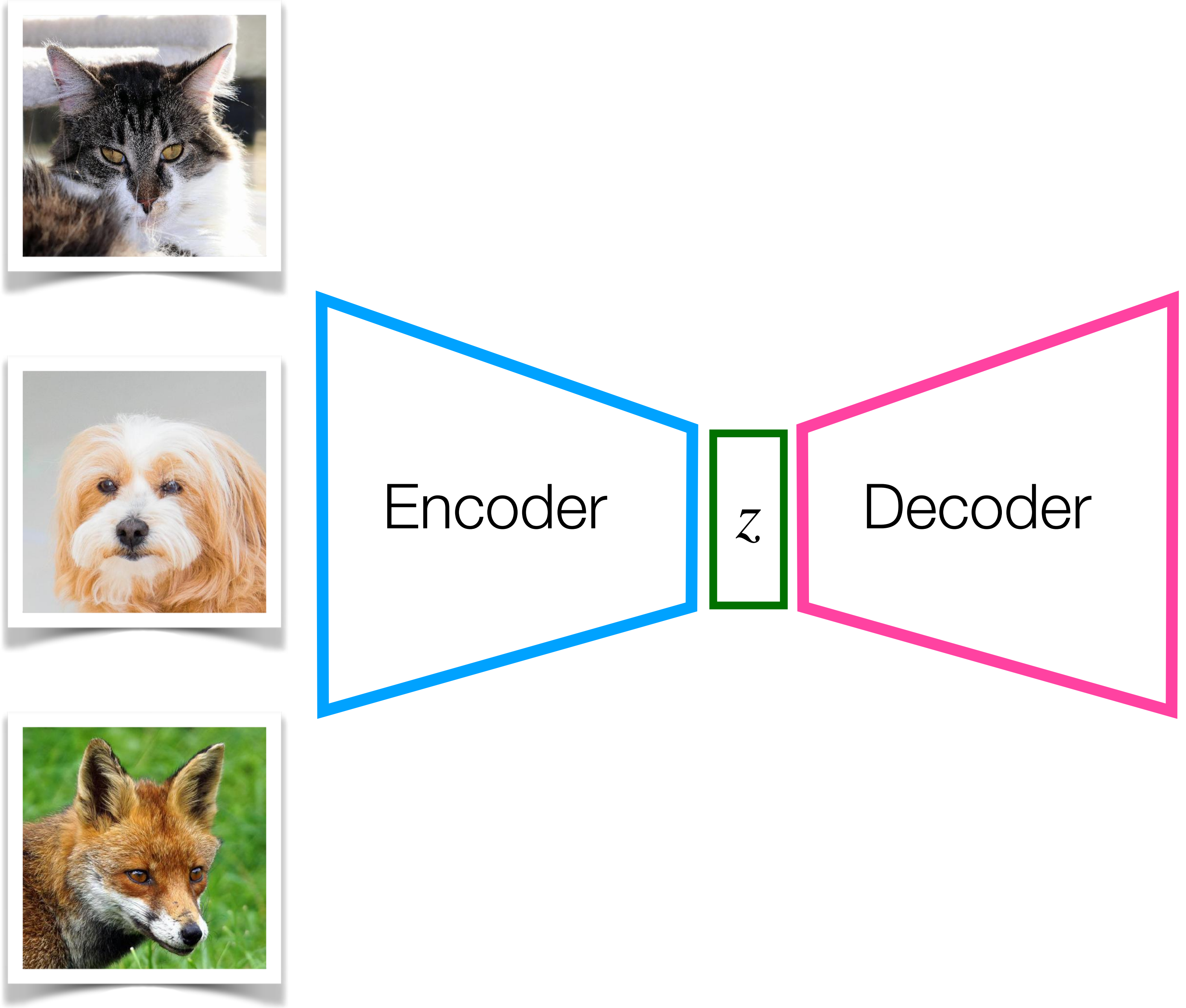
z : Hypothesis/Latent
x : Data

$p(x|z)$: the likelihood (given your hypothesis, what is the probabilty of observing x)

$p(z)$: prior probability

$p(z|x)$: the posterior (updating your belief based the data)

Bayes Theorem



$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}$$

z : Hypothesis/Latent
x : Data

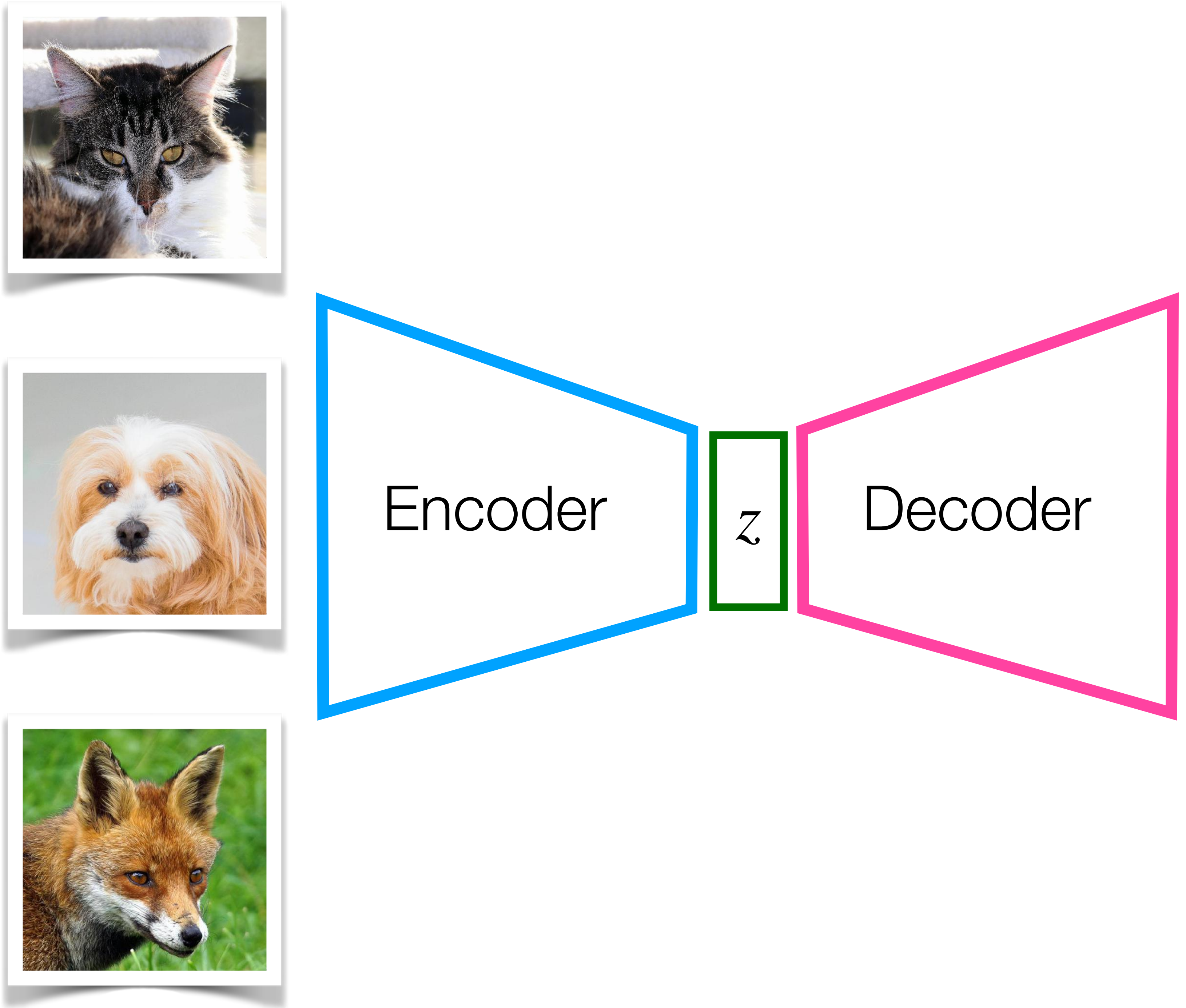
$p(x|z)$: the likelihood (given your hypothesis, what is the probabilty of observing x)

$p(z)$: prior probability

$p(z|x)$: the posterior (updating your belief based the data)

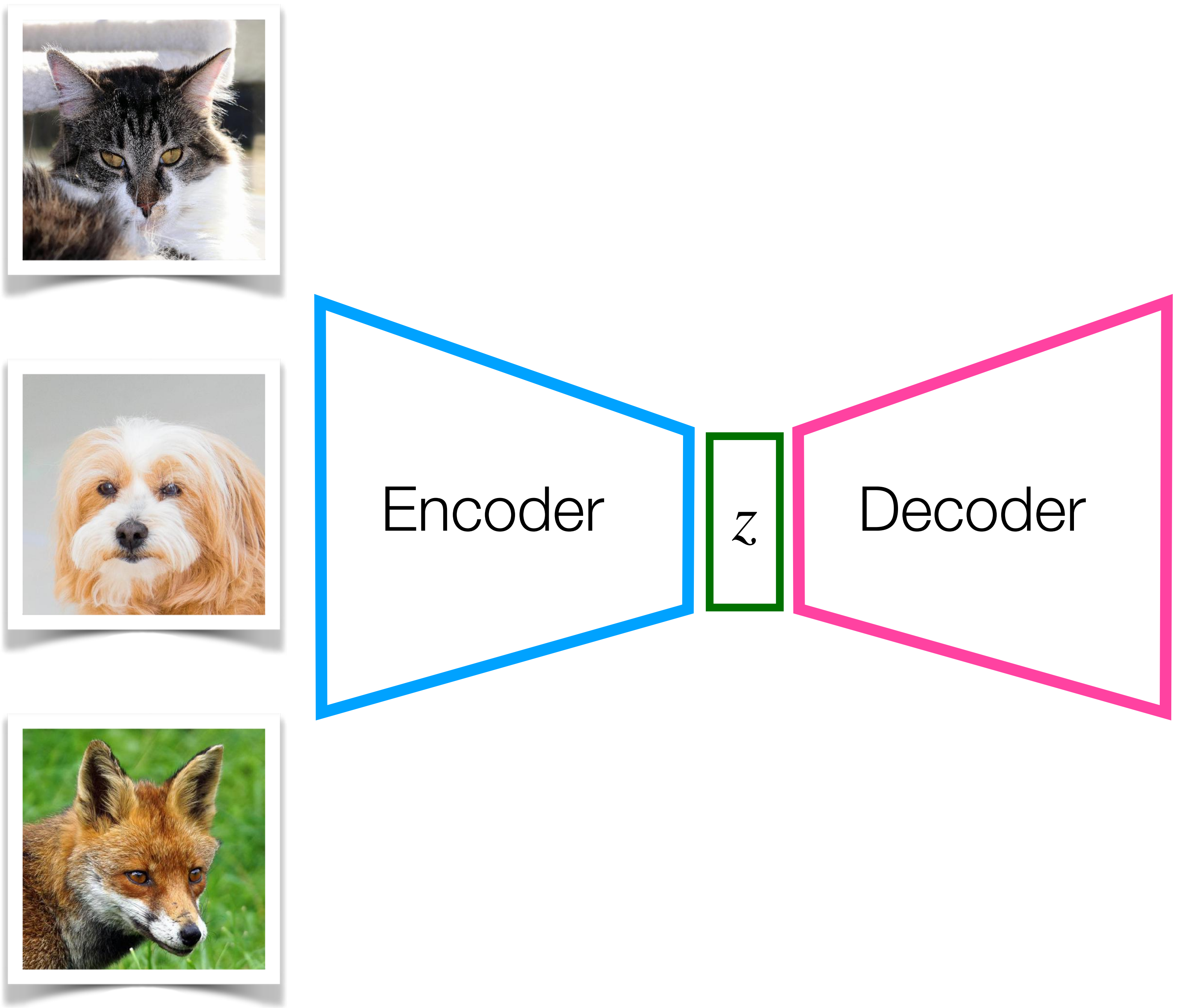
$p(x)$: marginal likelihood

Bayesian Inference



Bayesian Inference

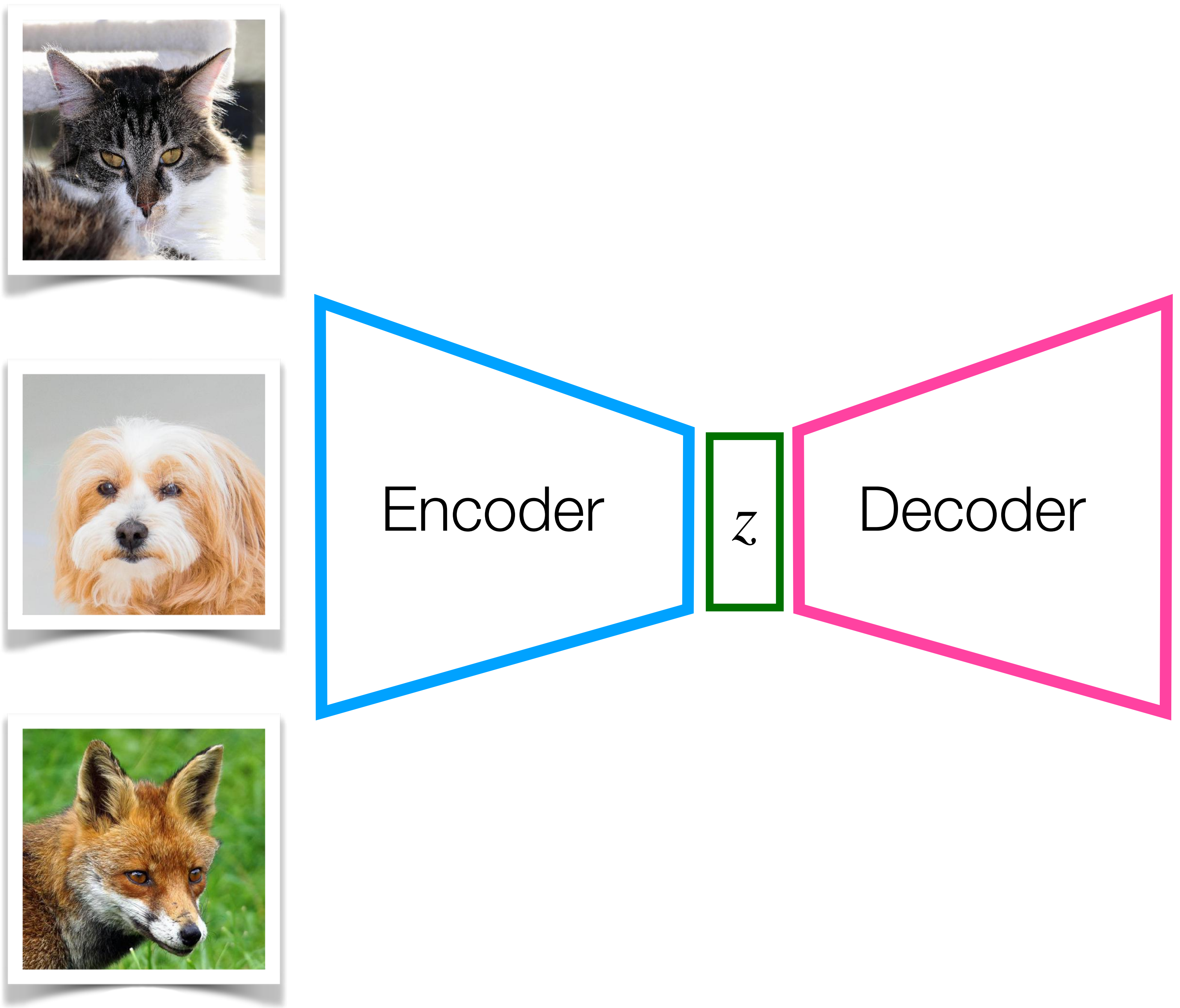
Goal: Compute the posterior distribution $p(z \,|\, x)$ of model parameters (θ or z)



Bayesian Inference

Goal: Compute the posterior distribution $p(z \mid x)$ of model parameters (θ or z)

But computing $p(z \mid x)$ is intractable!

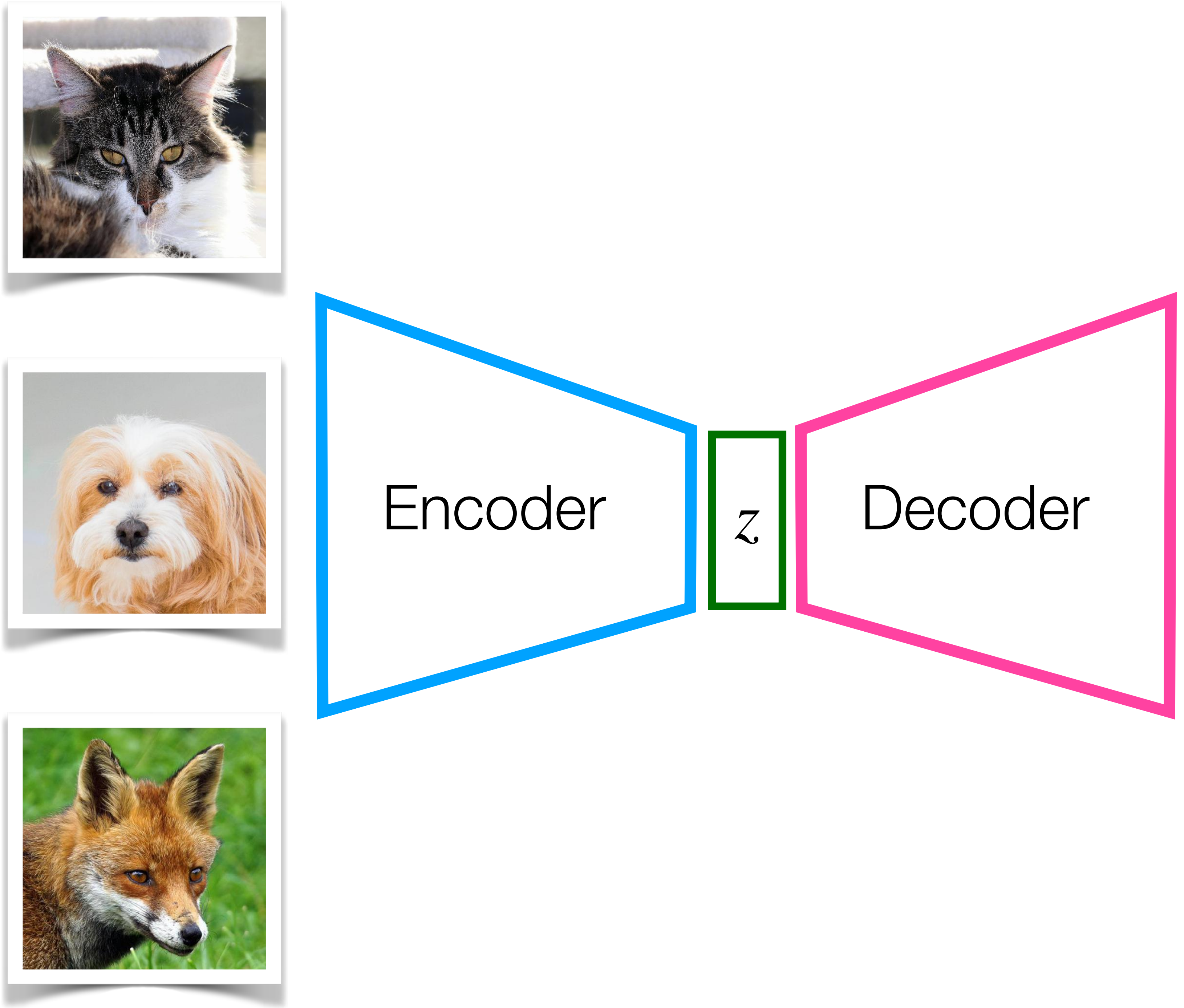


Bayesian Inference

Goal: Compute the posterior distribution $p(z \mid x)$ of model parameters (θ or z)

But computing $p(z \mid x)$ is intractable!

- for complex models
- for large datasets



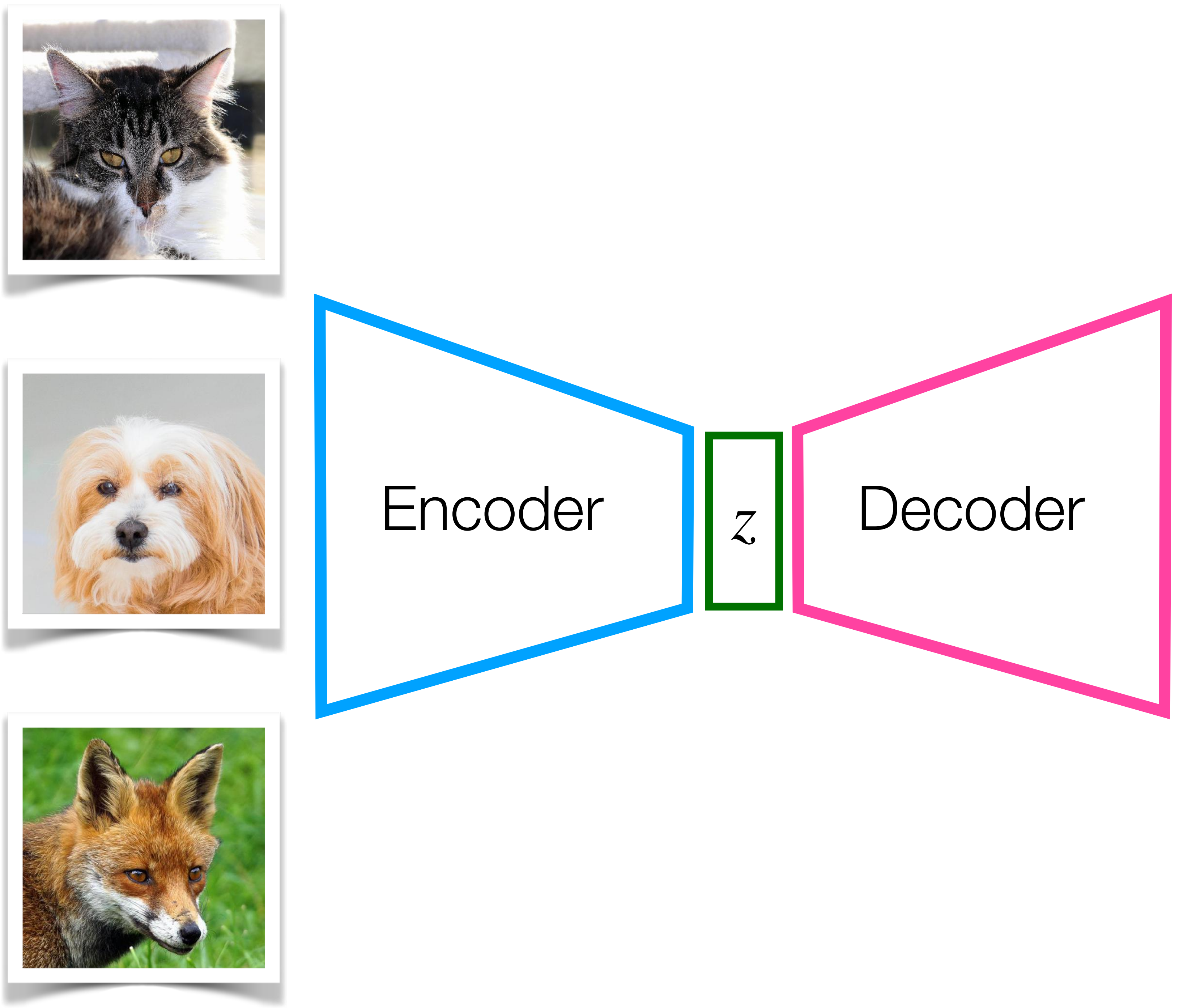
Bayesian Inference

Goal: Compute the posterior distribution $p(z \mid x)$ of model parameters (θ or z)

But computing $p(z \mid x)$ is intractable!

- for complex models
- for large datasets

Use SGD for Bayesian inference



Stochastic Gradient Descent (SGD)

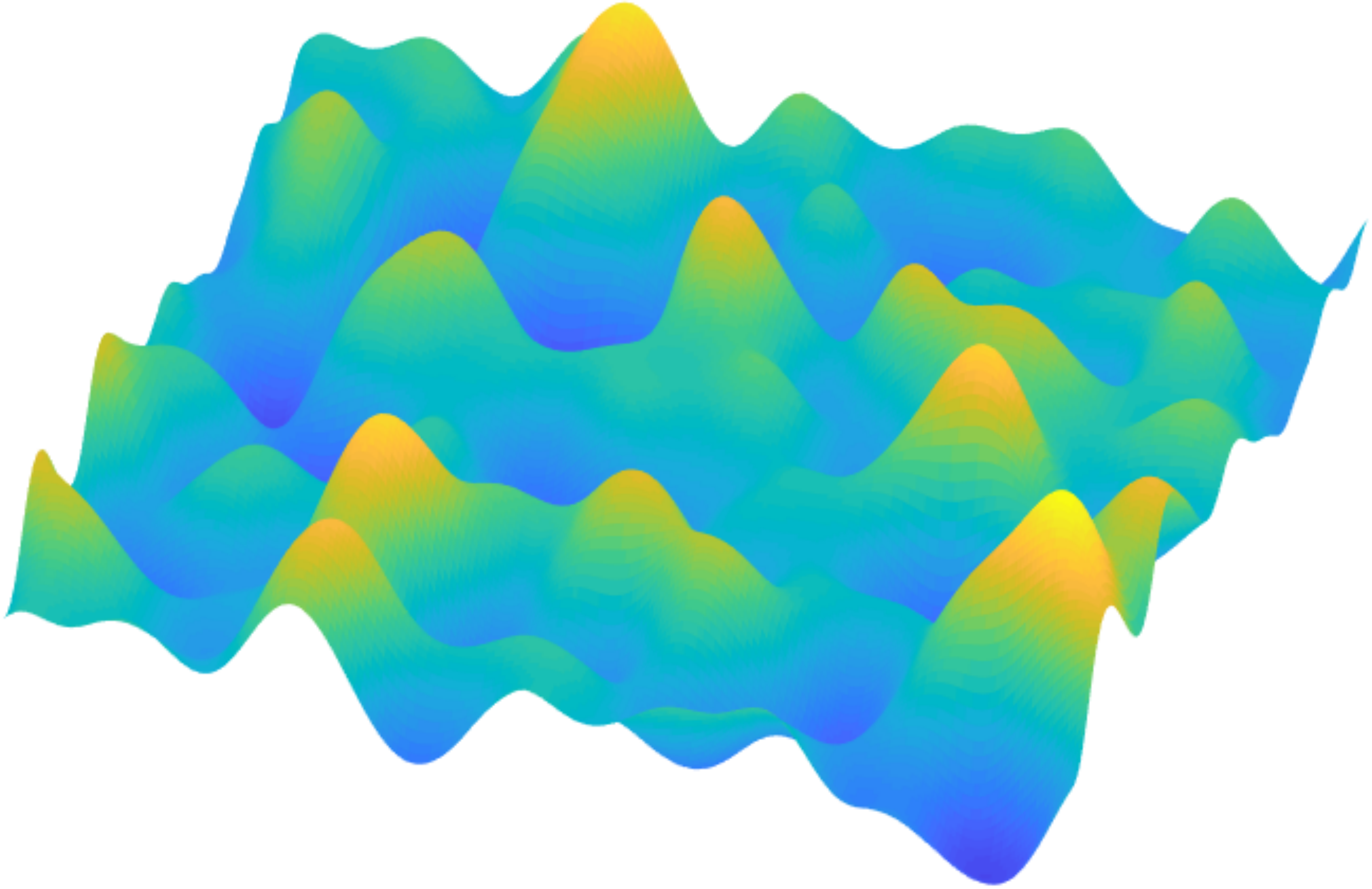
Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD)

Goal: We want to minimize a loss function $\mathscr{L}$

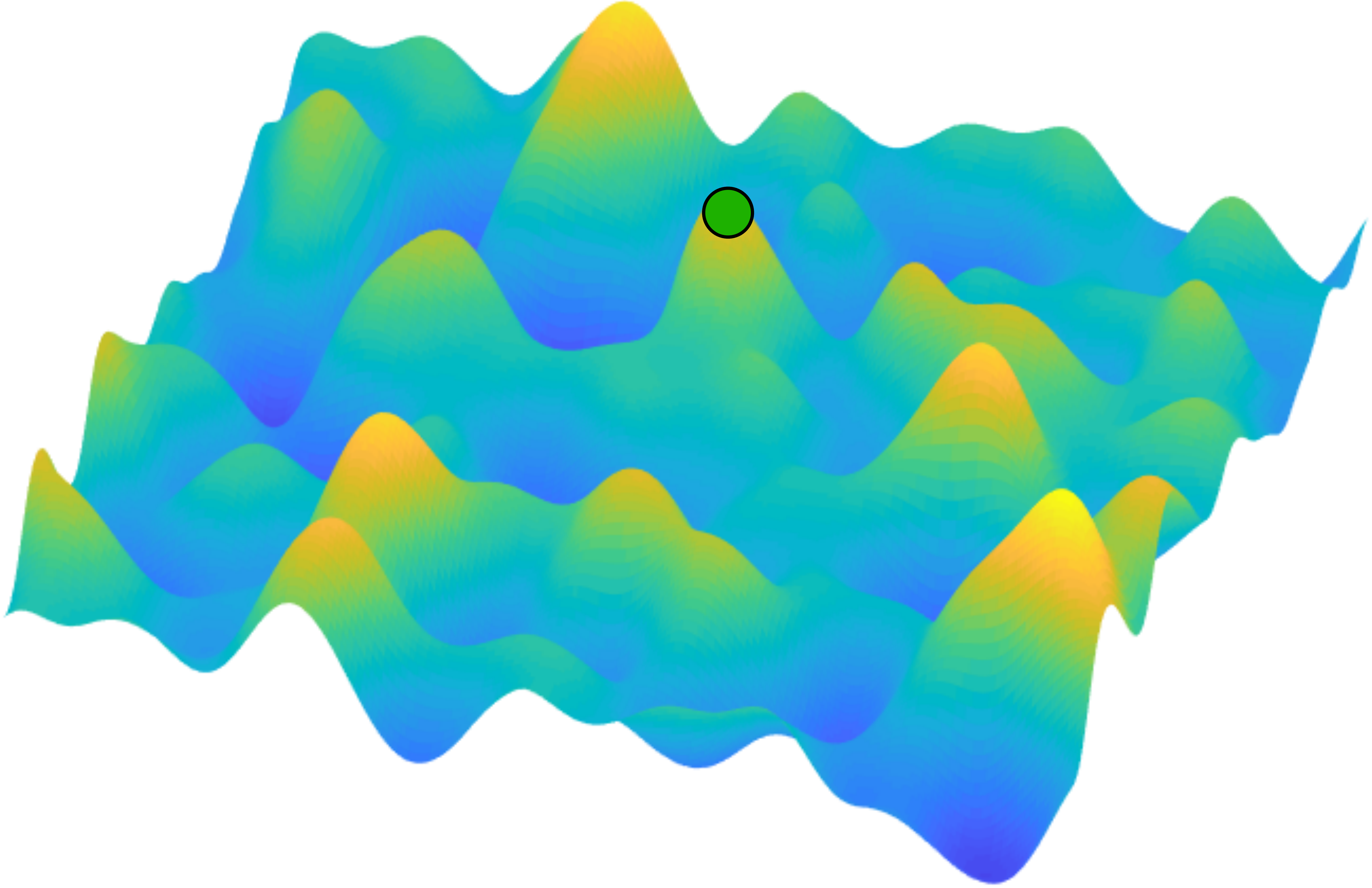
Stochastic Gradient Descent (SGD)

Goal: We want to minimize a loss function $\mathscr{L}$



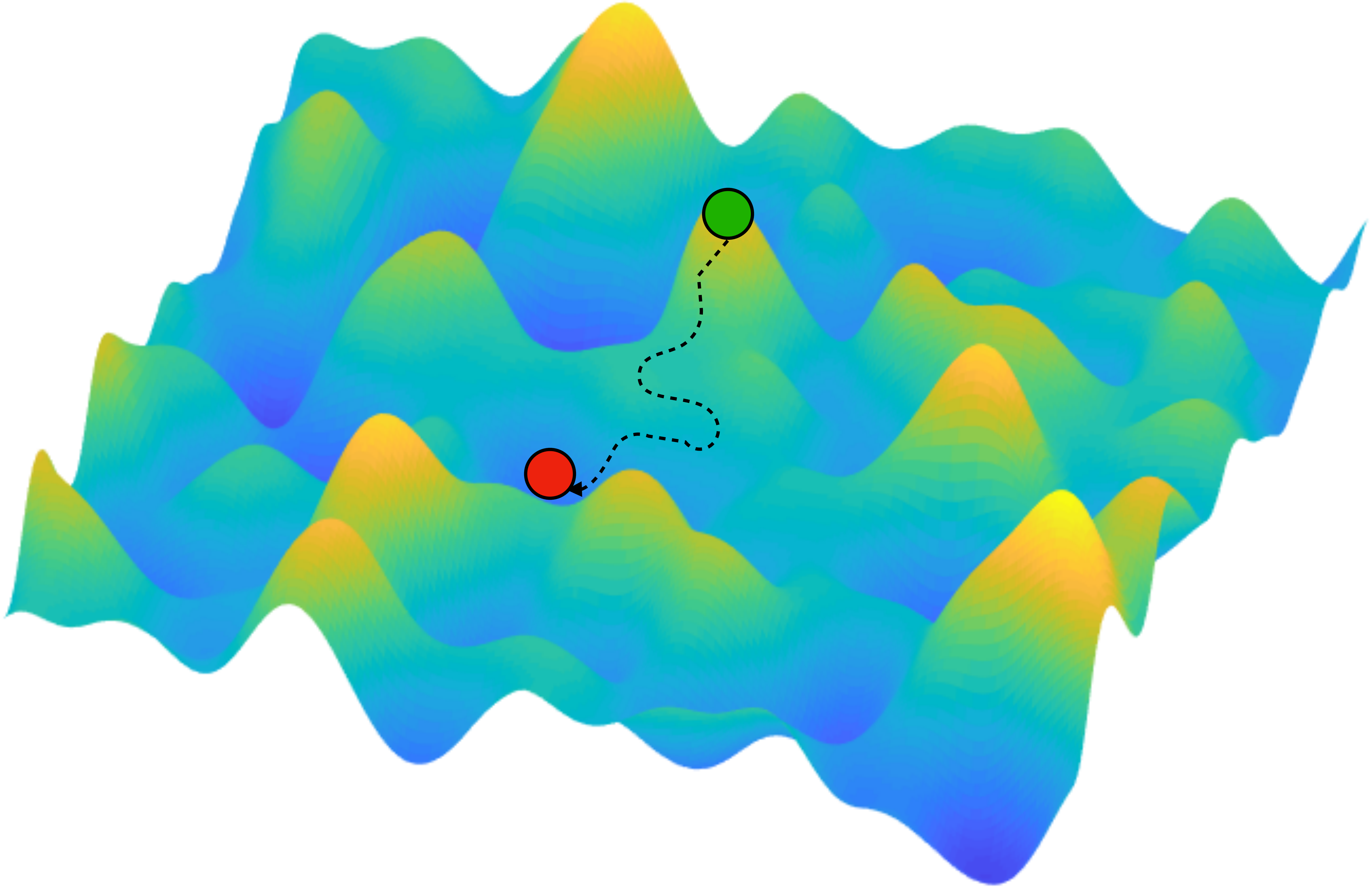
Stochastic Gradient Descent (SGD)

Goal: We want to minimize a loss function $\mathscr{L}$



Stochastic Gradient Descent (SGD)

Goal: We want to minimize a loss function $\mathscr{L}$

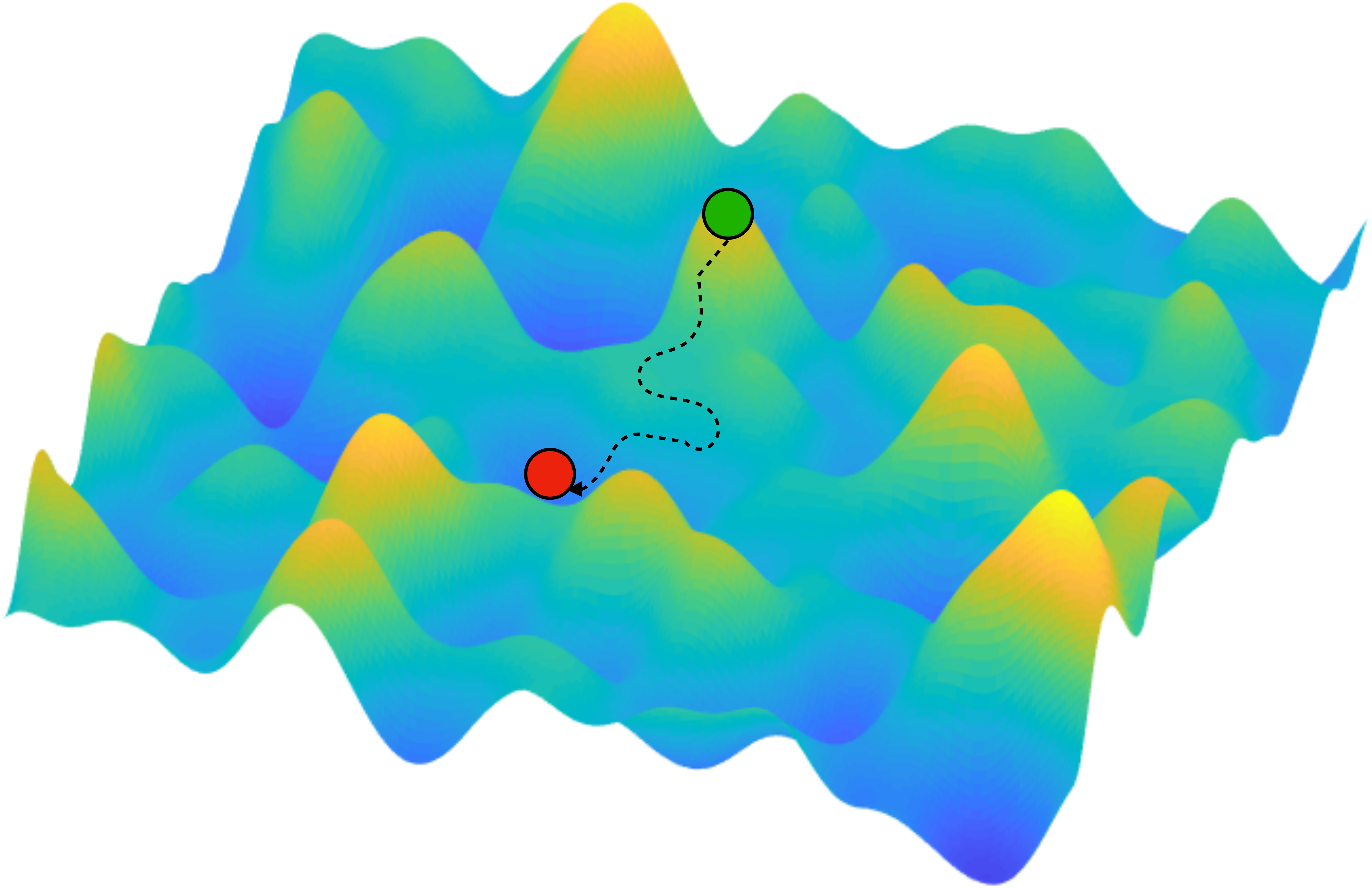


Stochastic Gradient Descent (SGD)

Goal: We want to minimize a loss function $\mathscr{L}$

x_t

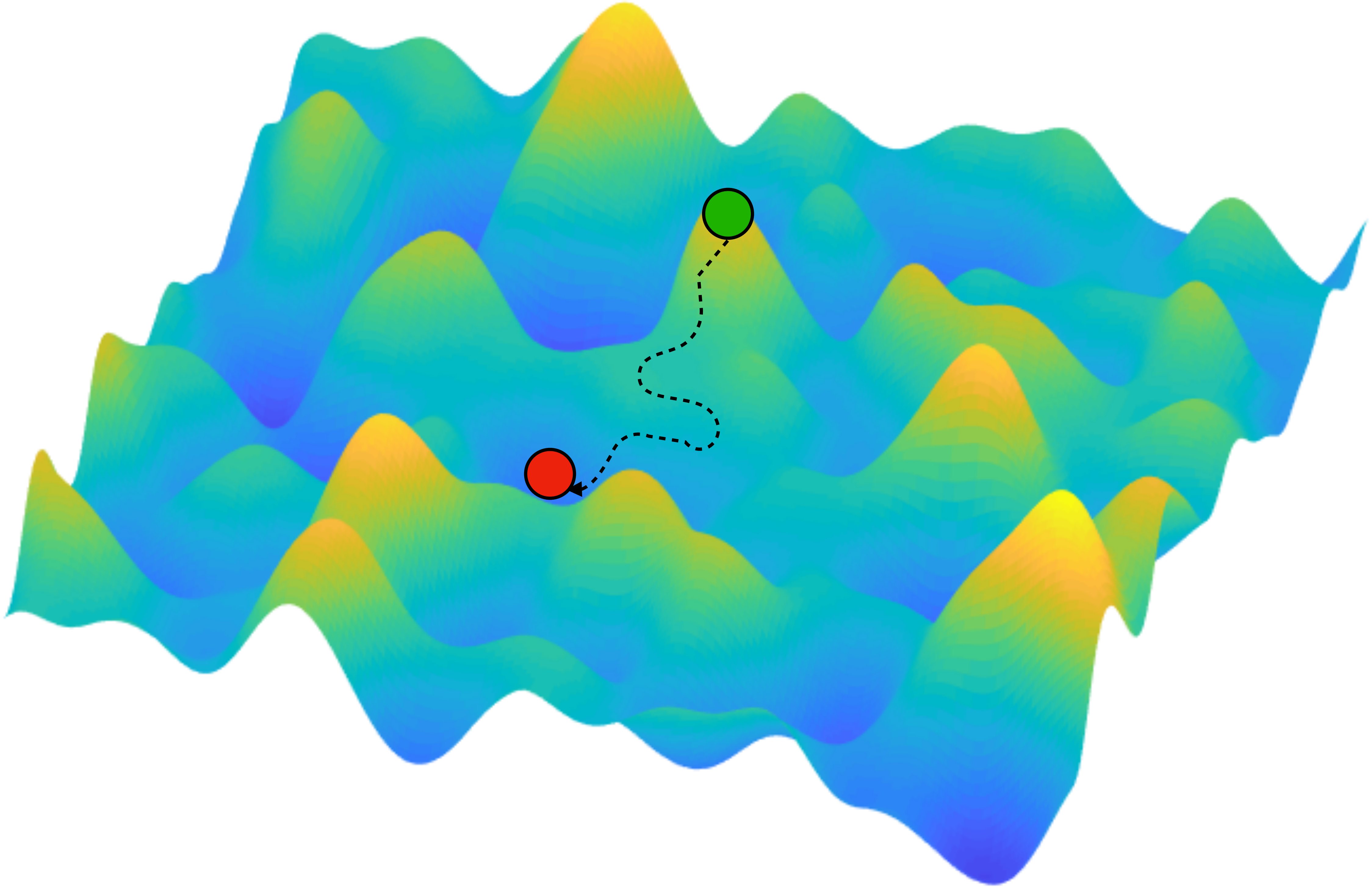
current



Stochastic Gradient Descent (SGD)

Goal: We want to minimize a loss function $\mathscr{L}$

$$\underbrace{x_{t+1}}_{} = \underbrace{x_t}_{\text{current}}$$



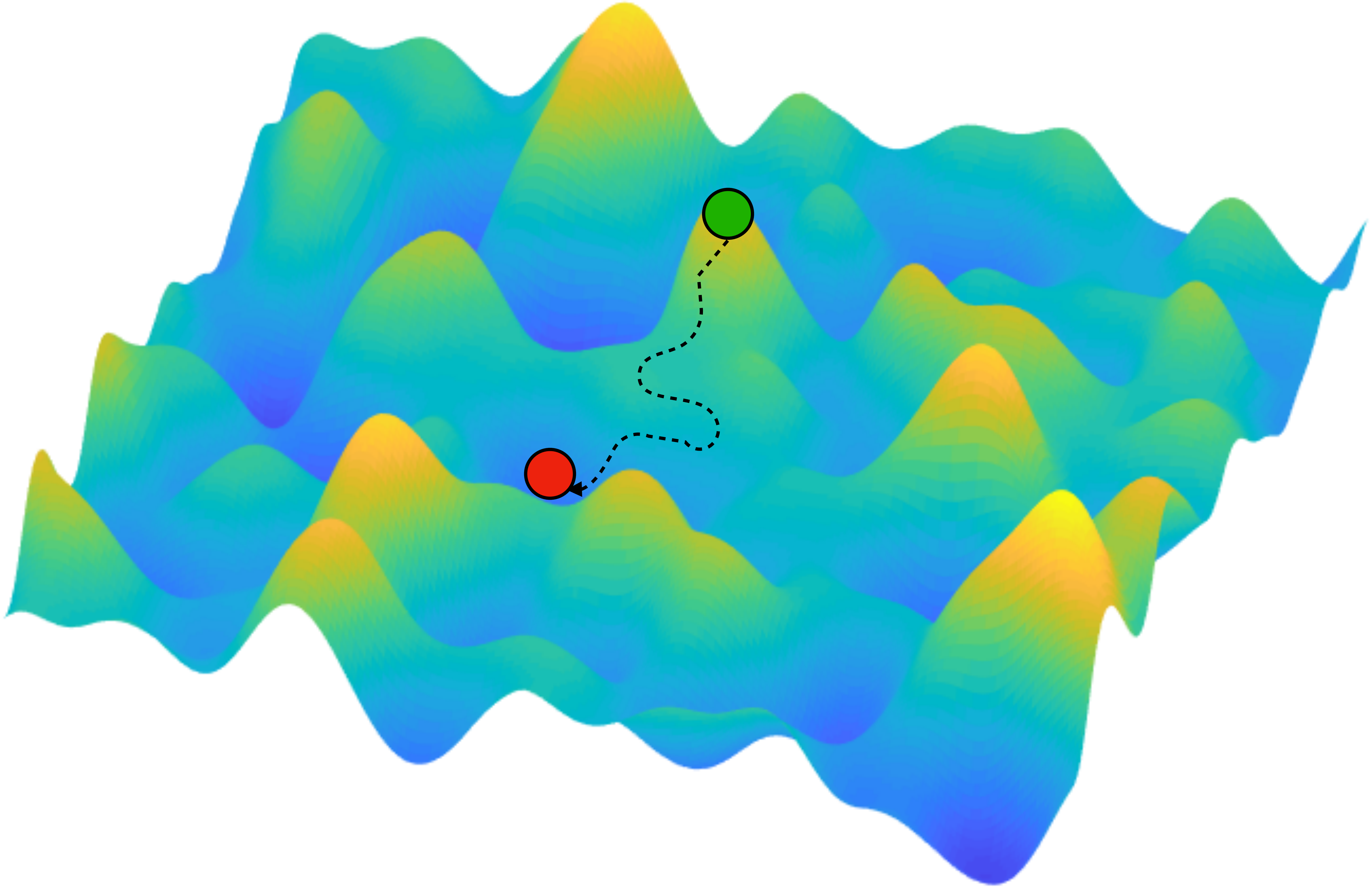
Stochastic Gradient Descent (SGD)

Goal: We want to minimize a loss function $\mathscr{L}$

$$x_{t+1} = x_t - \nabla \mathscr{L}$$

current

Gradient of the loss function



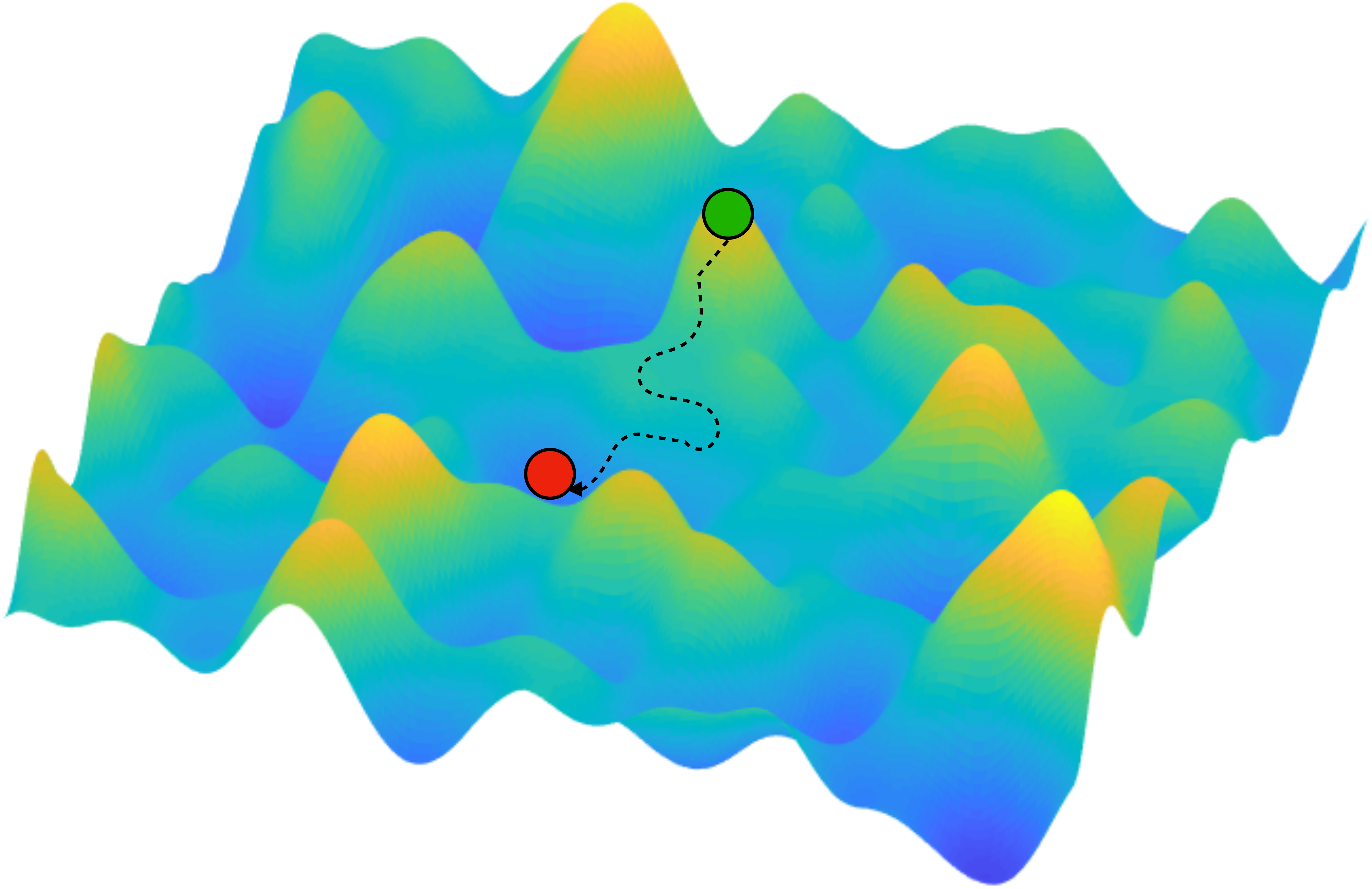
Stochastic Gradient Descent (SGD)

Goal: We want to minimize a loss function $\mathscr{L}$

$$x_{t+1} = x_t - \eta \nabla \mathscr{L}$$

current

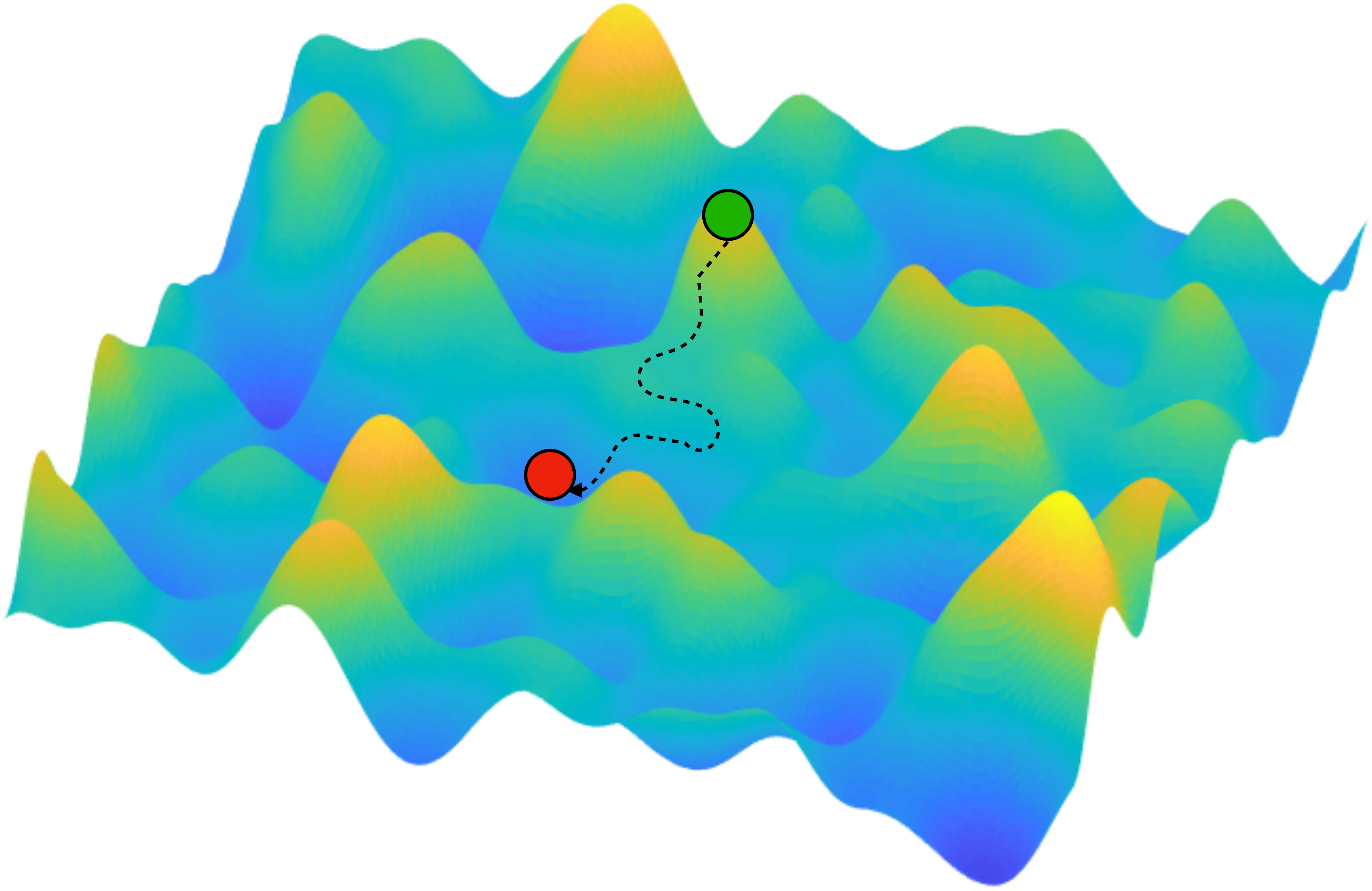
Gradient of the loss function



Stochastic Gradient Descent (SGD)

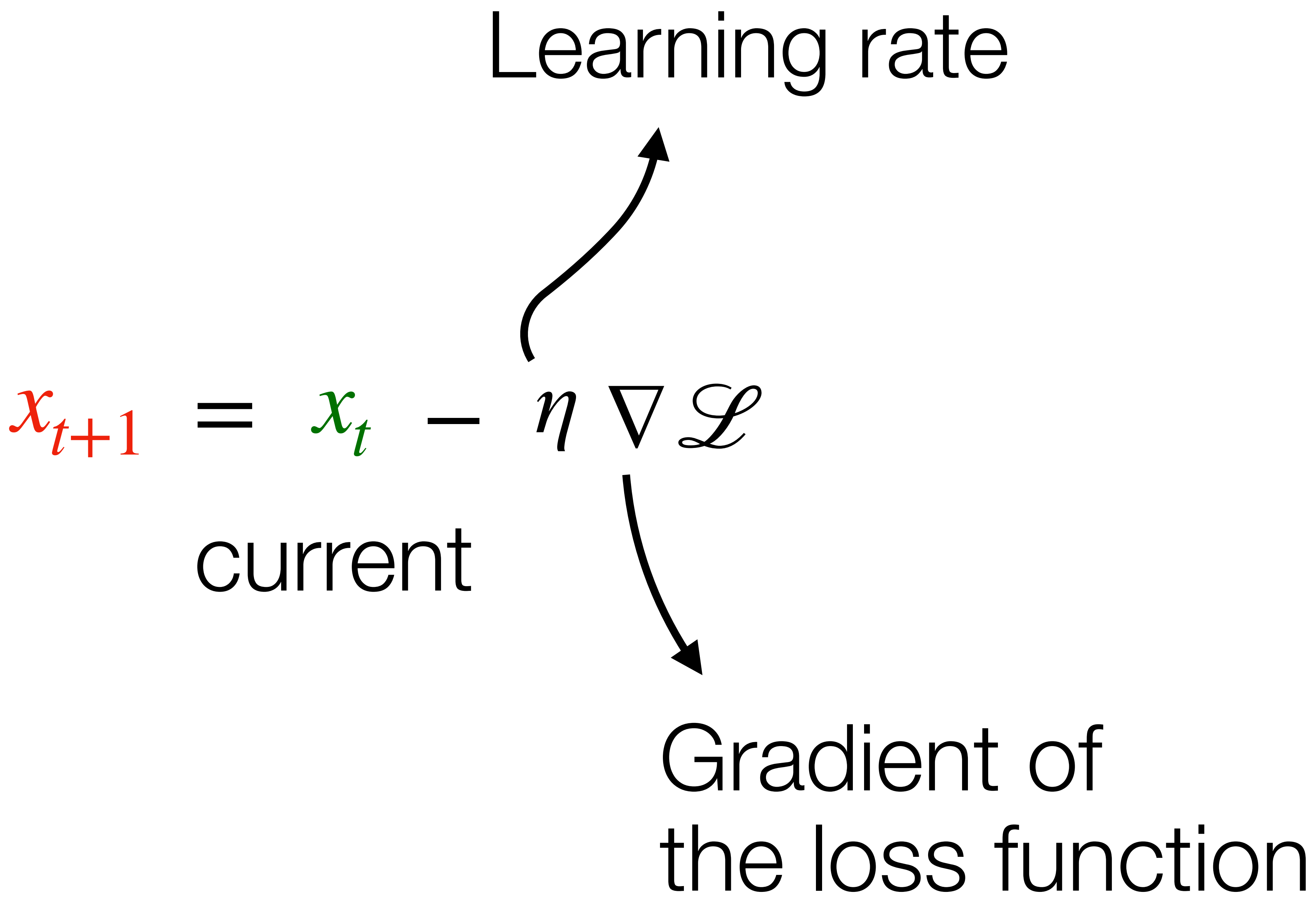
Goal: We want to minimize a loss function $\mathscr{L}$

$$x_{t+1} = \underset{\text{current}}{x_t} - \overset{\text{Learning rate}}{\eta} \underset{\text{Gradient of the loss function}}{\nabla \mathscr{L}}$$



Stochastic Gradient Descent (SGD)

Goal: We want to minimize a loss function $\mathscr{L}$

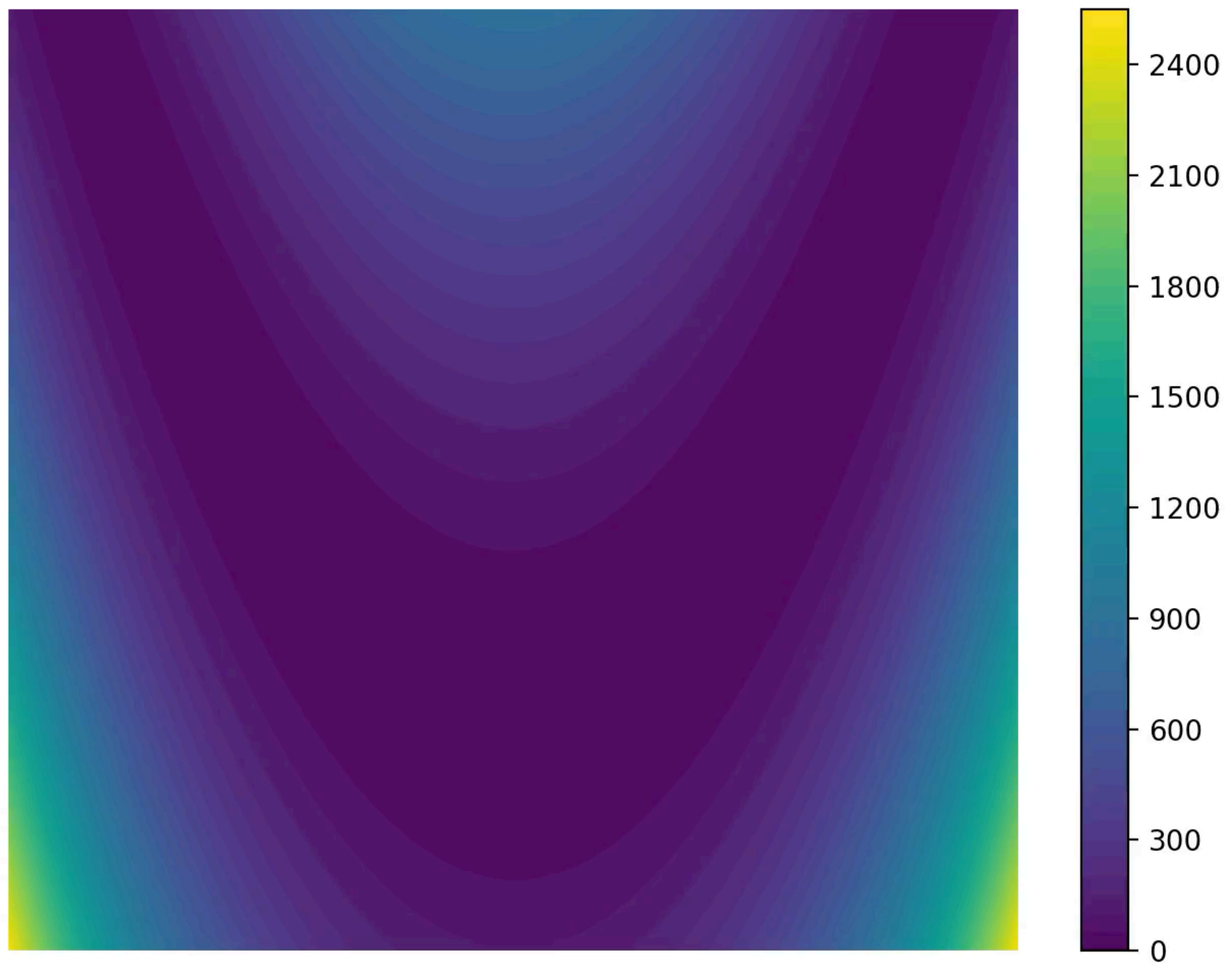


Learning rate

$$x_{t+1} = x_t - \eta \nabla \mathscr{L}$$

current

Gradient of
the loss function



SGD in action

Stochastic Gradient Descent (SGD)

Goal: We want to minimize a loss function $\mathscr{L}$

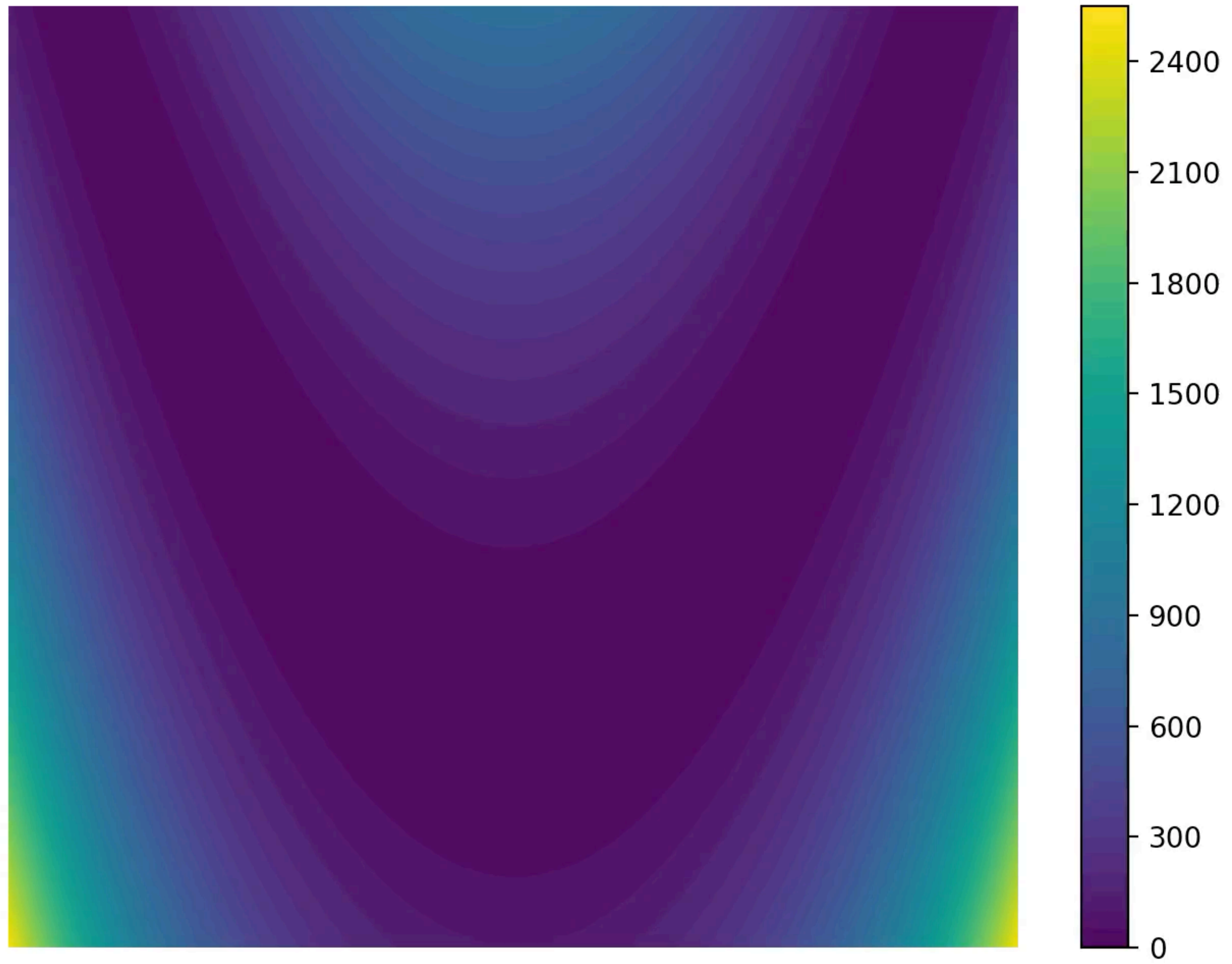
Learning rate

$$x_{t+1} = x_t - \eta \nabla \mathscr{L}$$

current

Gradient of the loss function

$$\mathscr{L}(x, y) = (1 - x)^2 + 100 * (y - x^2)^2$$



SGD in action

Stochastic Gradient Descent (SGD)

Goal: We want to minimize a loss function $\mathscr{L}$

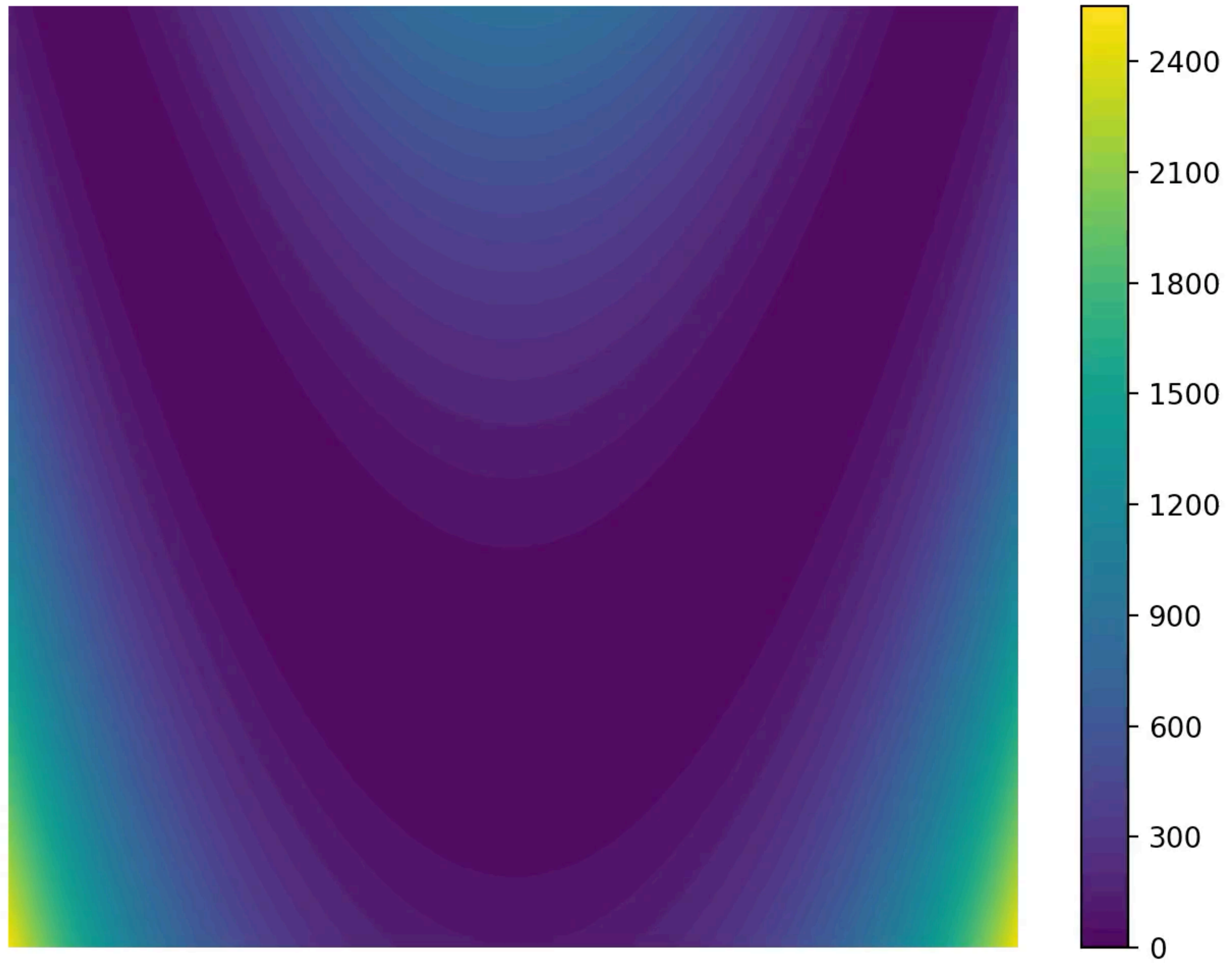
Learning rate

$$x_{t+1} = x_t - \eta \nabla \mathscr{L}$$

current

Gradient of
the loss function

$$\mathscr{L}(x, y) = (1 - x)^2 + 100 * (y - x^2)^2$$



SGD in action

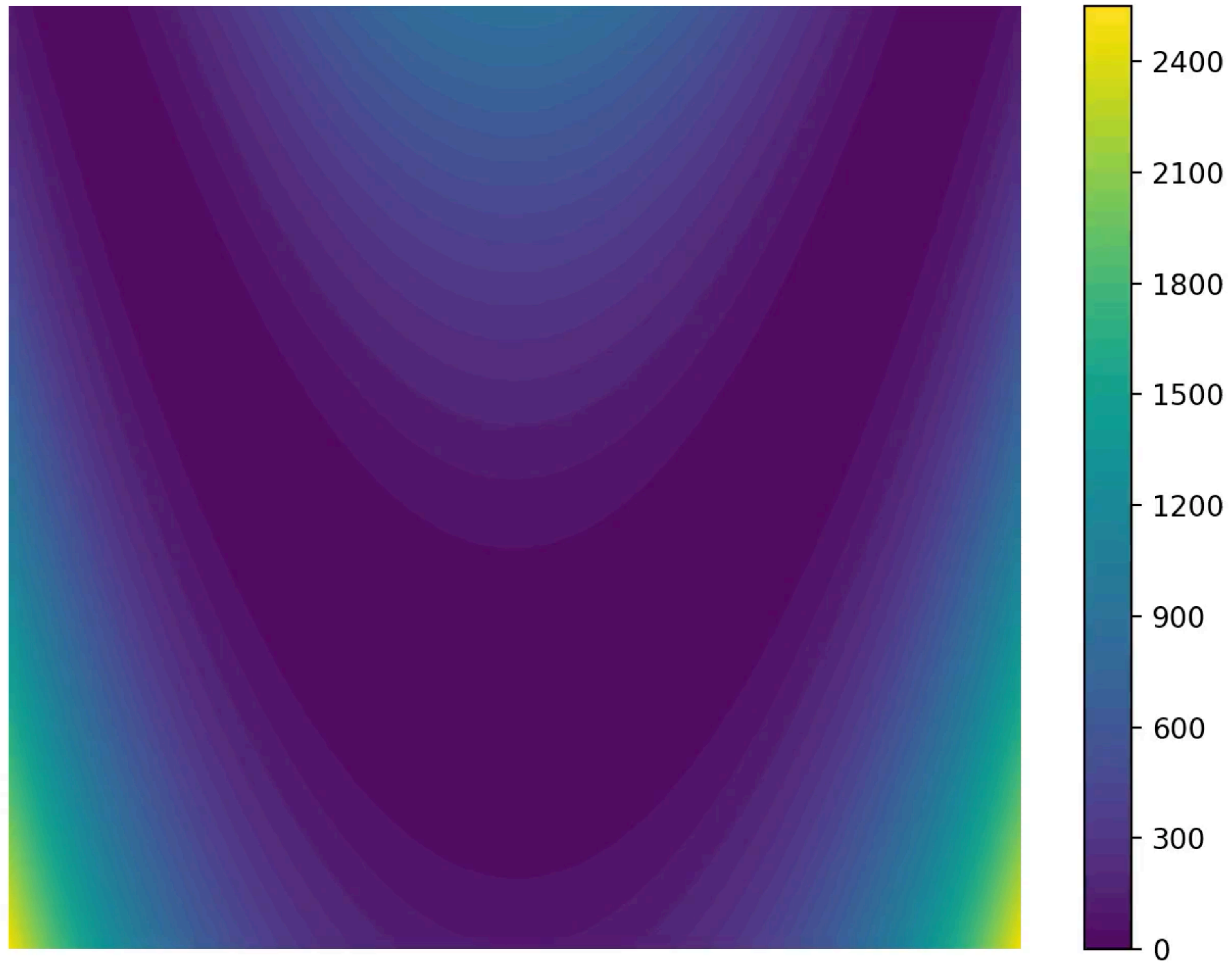
SGD with Noise (Bayesian SGD)

SGD with noise

Goal: We want to minimize a loss function $\mathscr{L}$

$$\mathscr{L}(x, y) = (1 - x)^2 + 100 * (y - x^2)^2$$

$$x_{t+1} = x_t - \eta \nabla \mathscr{L}$$



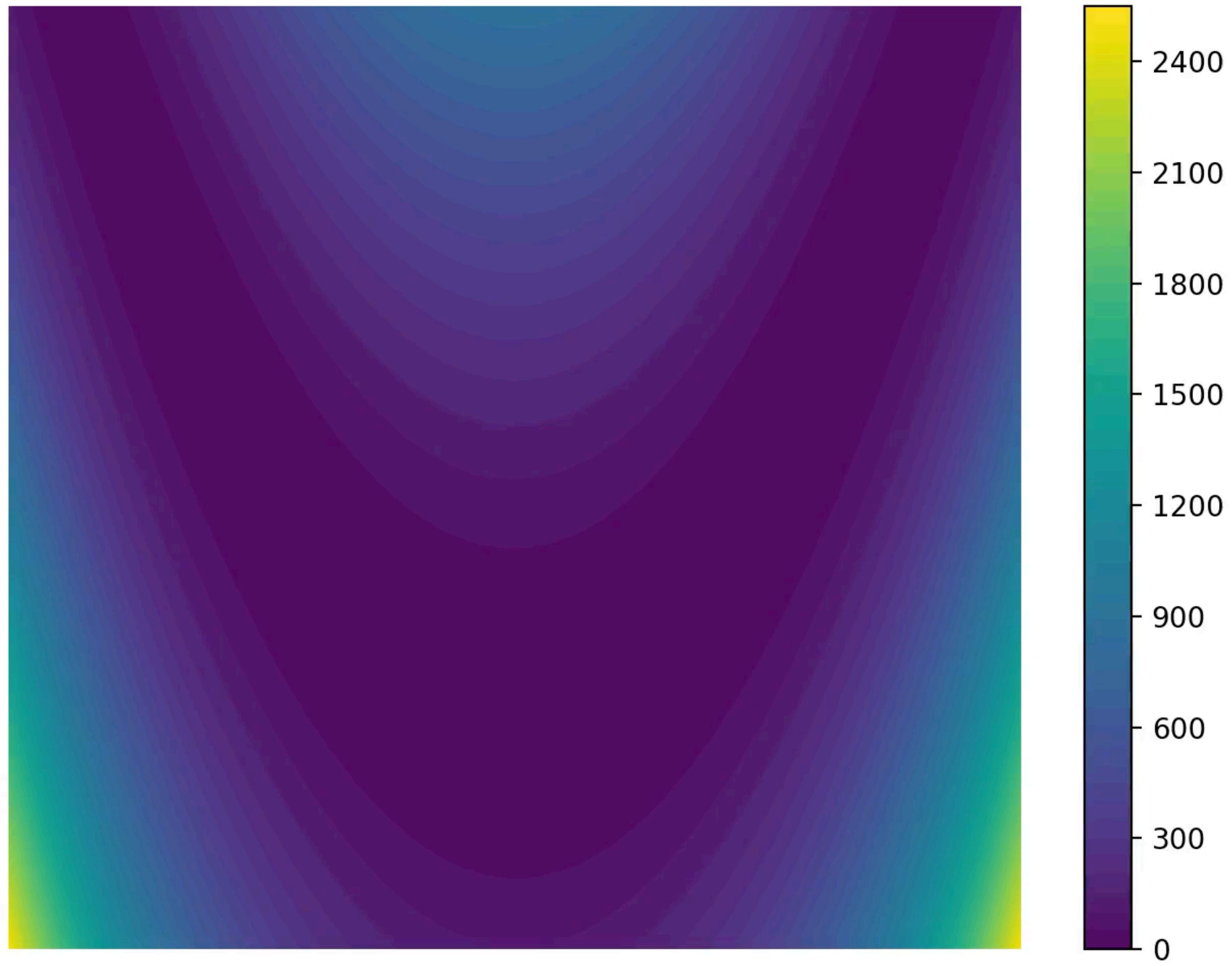
SGD with noise in action

SGD with noise

Goal: We want to minimize a loss function $\mathscr{L}$

$$\mathscr{L}(x, y) = (1 - x)^2 + 100 * (y - x^2)^2$$

$$\color{red}{x_{t+1}} = \color{green}{x_t} - \eta \nabla \mathscr{L} + \epsilon_t$$



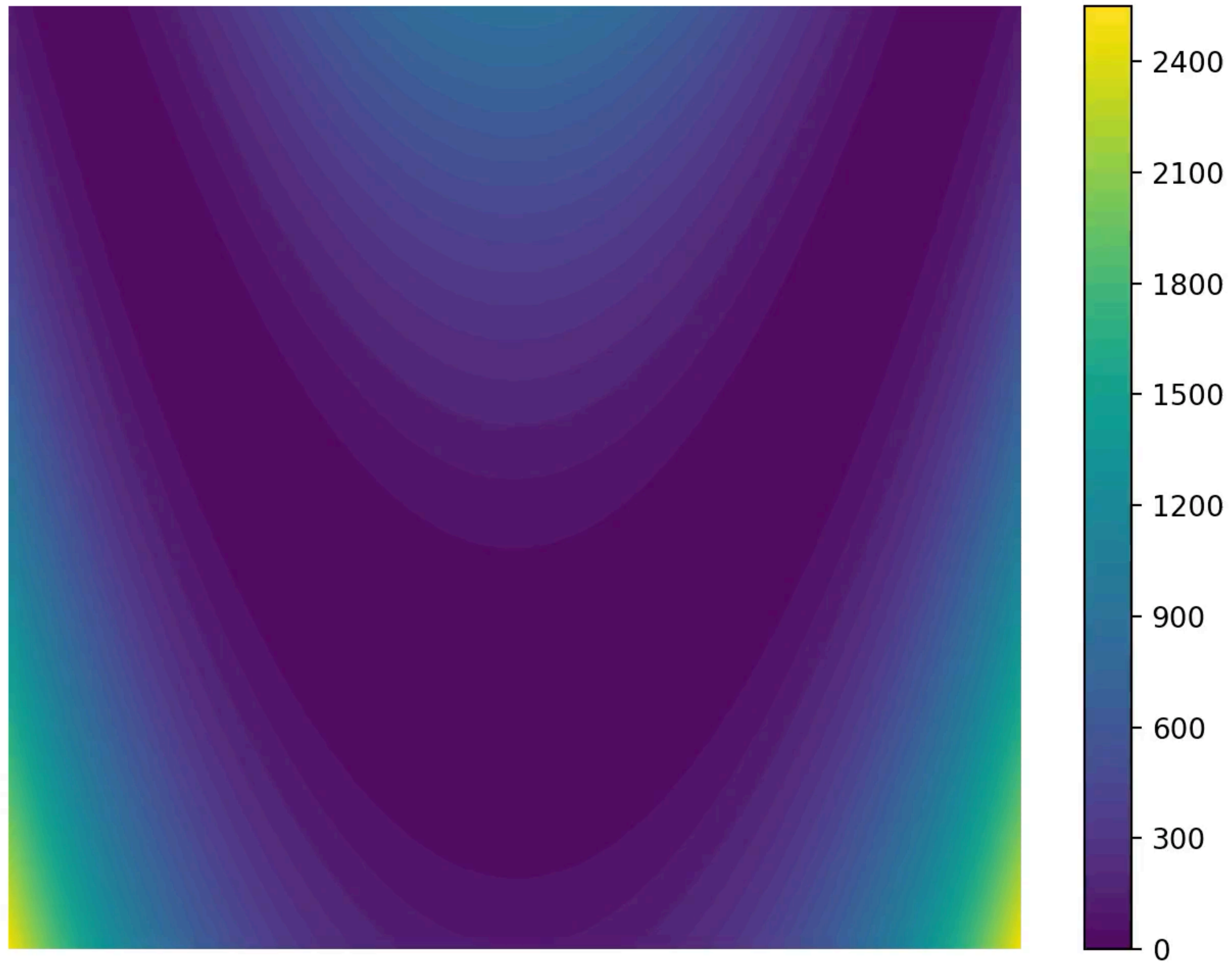
SGD with noise in action

SGD with noise

Goal: We want to minimize a loss function $\mathscr{L}$

$$\mathscr{L}(x, y) = (1 - x)^2 + 100 * (y - x^2)^2$$

$$x_{t+1} = x_t - \eta \nabla \mathscr{L} + \epsilon_t$$

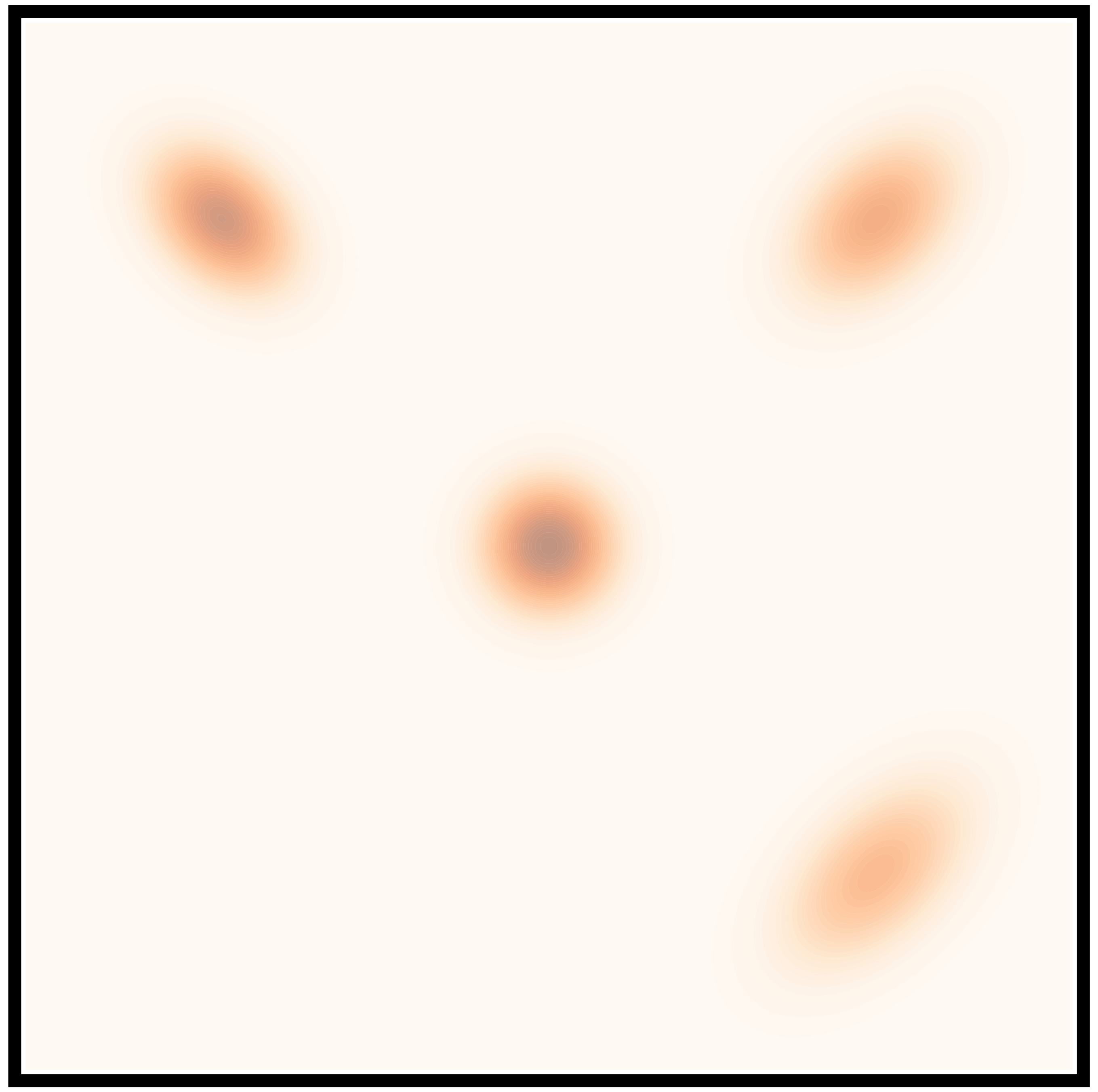


SGD with noise in action

Stochastic Gradient Langevin dynamics

Langevin dynamics

$$d\mathbf{x}_t = \nabla \log p(\mathbf{x}) + \sqrt{2} dW_t$$

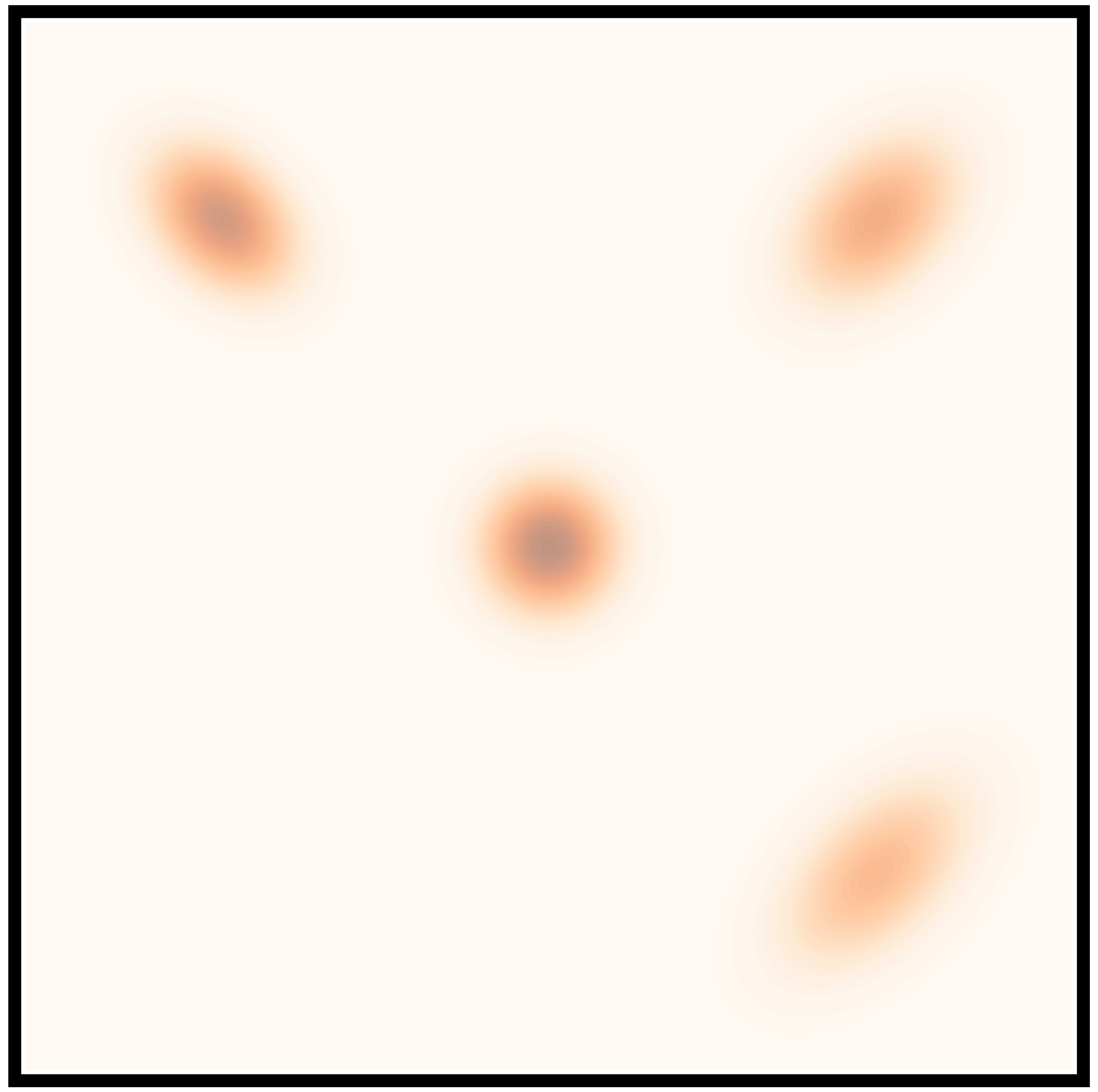


$p(\mathbf{x})$

Langevin dynamics

$$d\mathbf{x}_t = \nabla \log p(\mathbf{x}) + \sqrt{2} dW_t$$

Euler-Maruyama method to simulate Langevin diffusion:



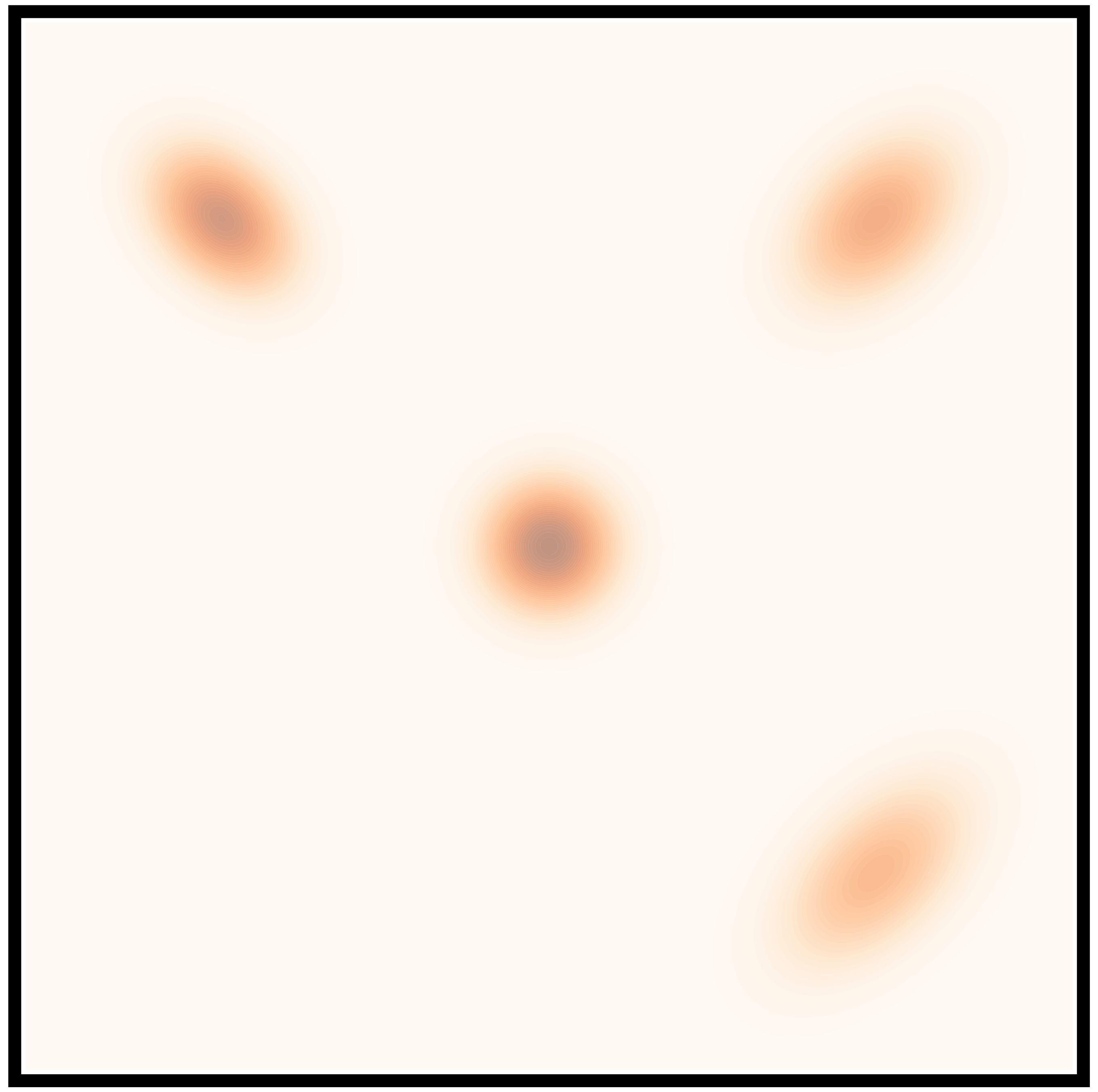
$p(\mathbf{x})$

Langevin dynamics

$$d\mathbf{x}_t = \nabla \log p(\mathbf{x}) + \sqrt{2} dW_t$$

Euler-Maruyama method to simulate Langevin diffusion:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \tau \nabla \log p(\mathbf{x}) + \sqrt{2\tau}\xi$$



$p(\mathbf{x})$

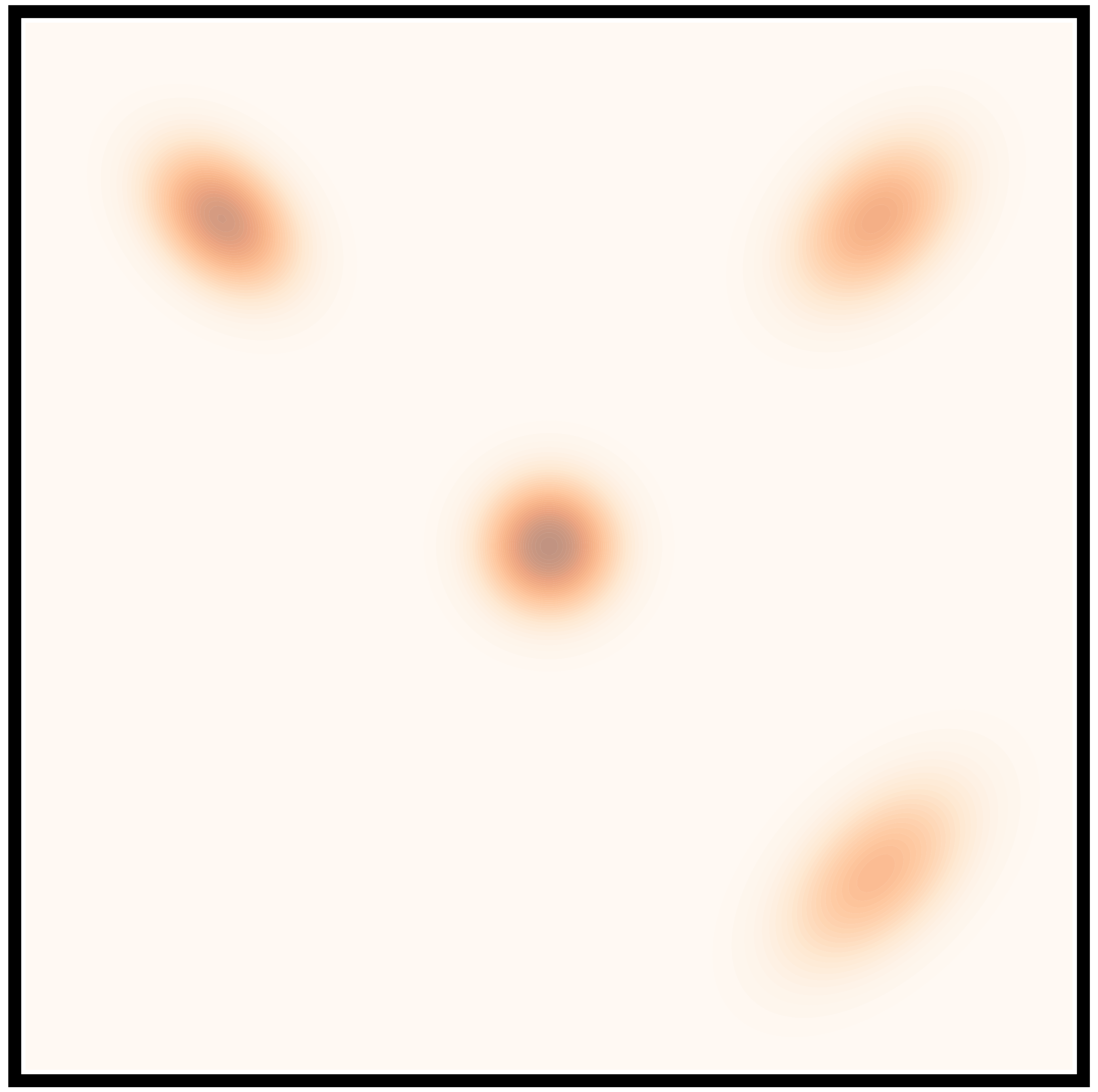
Langevin dynamics

$$d\mathbf{x}_t = \nabla \log p(\mathbf{x}) + \sqrt{2}dW_t$$

Euler-Maruyama method to simulate Langevin diffusion:

Step size

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \tau \nabla \log p(\mathbf{x}) + \sqrt{2\tau}\xi$$



$p(\mathbf{x})$

Langevin dynamics

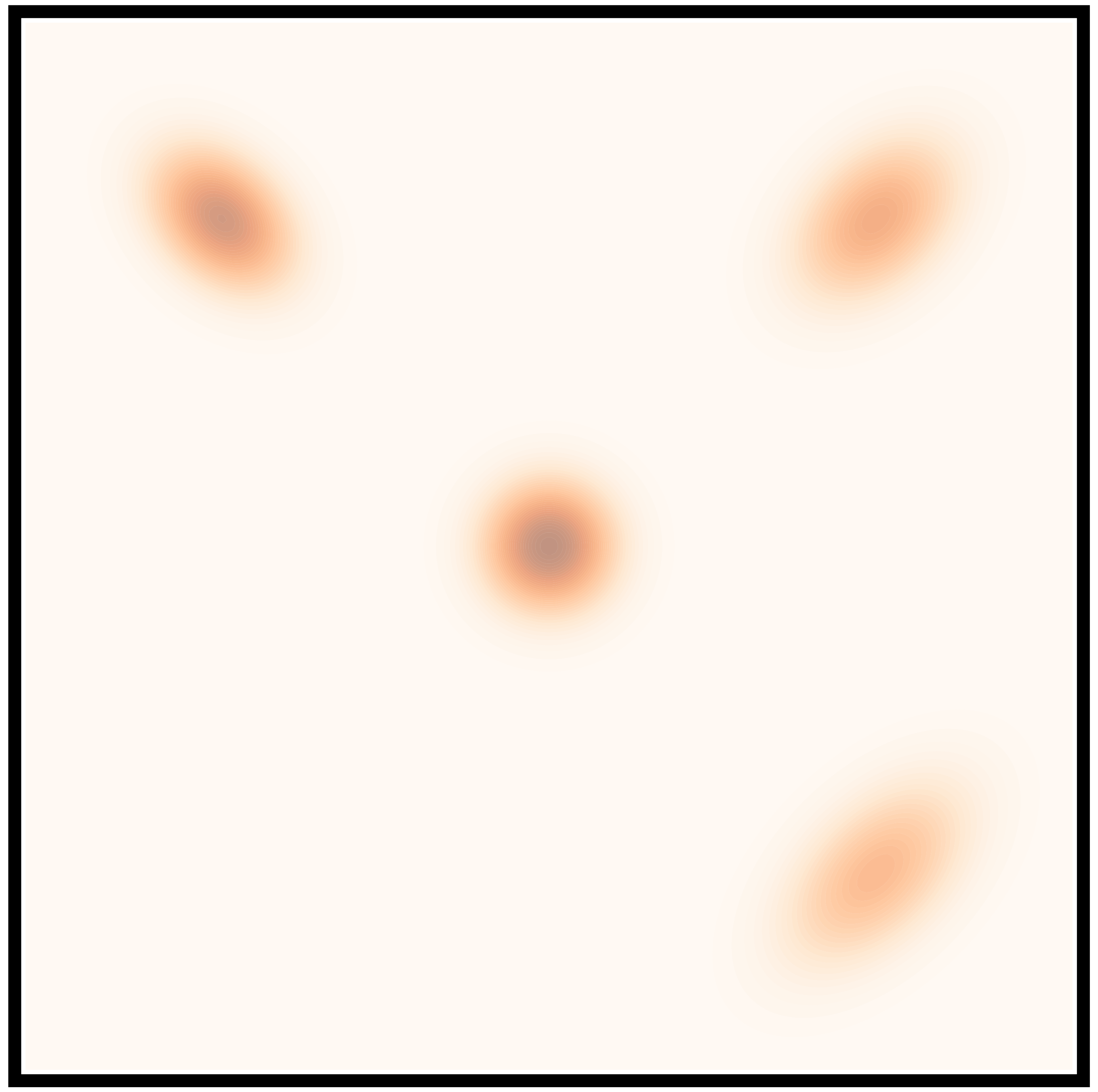
$$d\mathbf{x}_t = \nabla \log p(\mathbf{x}) + \sqrt{2} dW_t$$

Euler-Maruyama method to simulate Langevin diffusion:

Step size

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \tau \nabla \log p(\mathbf{x}) + \sqrt{2\tau}\xi$$

Gaussian noise



$p(\mathbf{x})$

Stochastic Gradient Langevin Dynamics (SGLD)

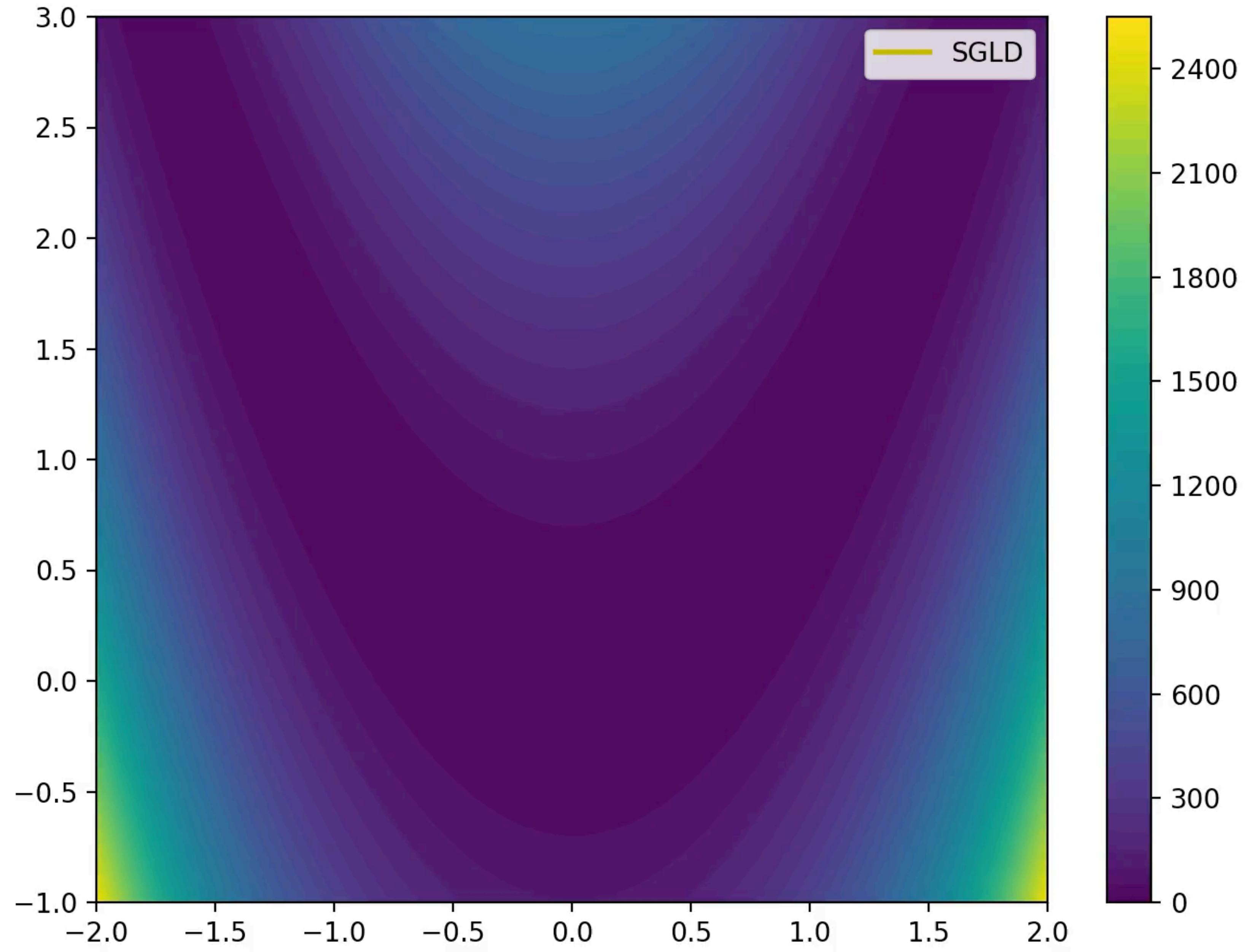
$$x_{t+1} = x_t - \eta \nabla \mathscr{L}$$

Stochastic Gradient Langevin Dynamics (SGLD)

$$x_{t+1} = x_t - \eta \nabla \mathscr{L}$$

Stochastic Gradient Langevin Dynamics (SGLD)

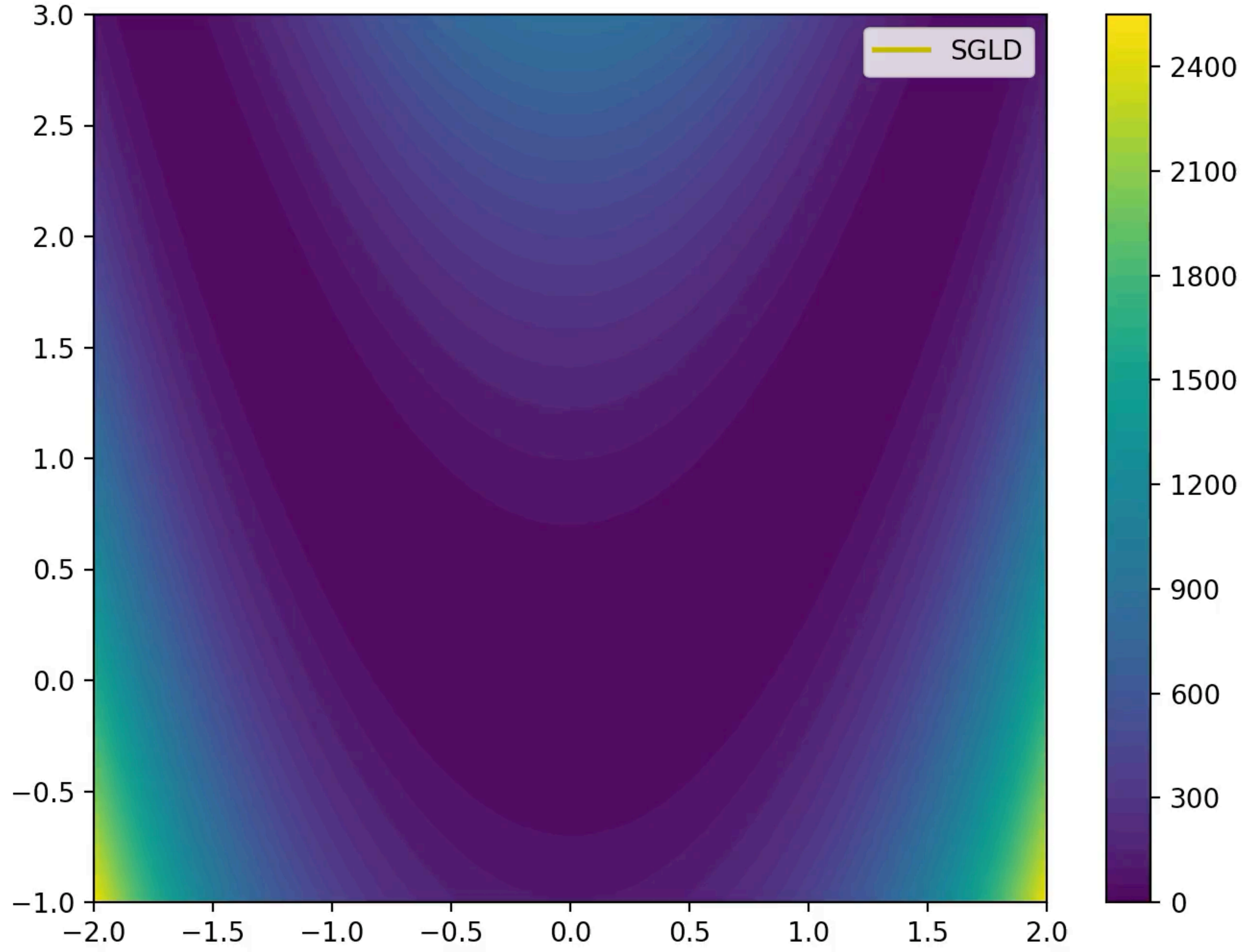
$$x_{t+1} = x_t - \eta \nabla \log \mathscr{L} + \sqrt{2\eta}\epsilon_t$$



SGLD in action

Stochastic Gradient Langevin Dynamics (SGLD)

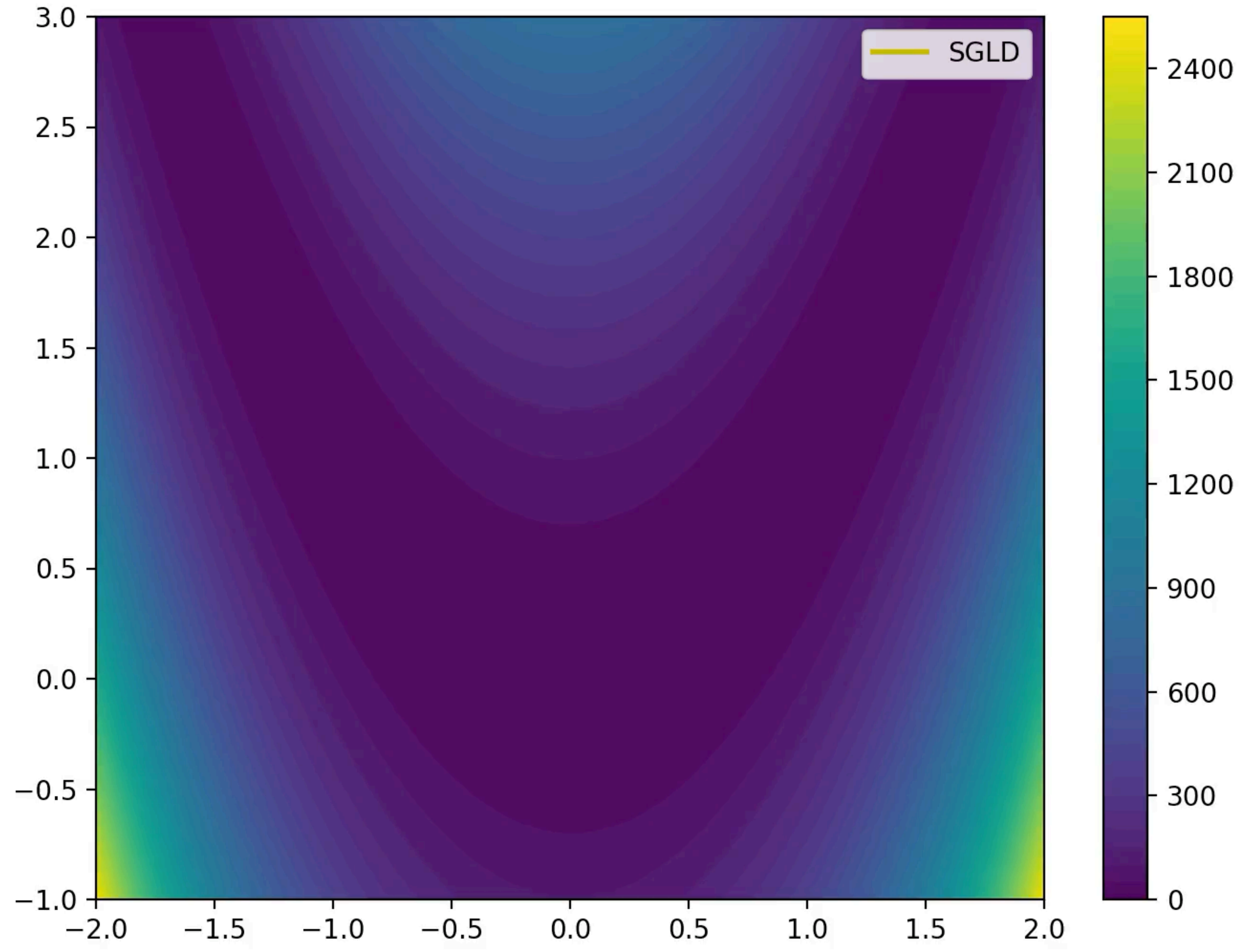
$$x_{t+1} = x_t - \eta \nabla \log \mathscr{L} + \sqrt{2\eta}\epsilon_t$$



SGLD in action

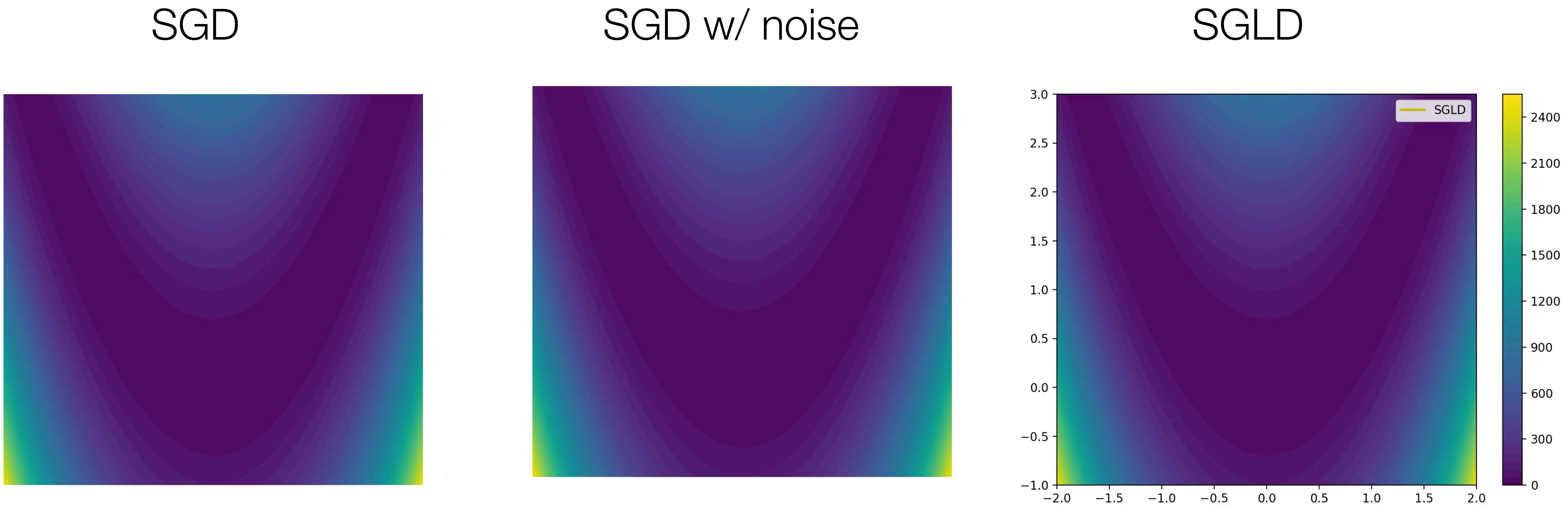
Stochastic Gradient Langevin Dynamics (SGLD)

$$x_{t+1} = x_t - \eta \nabla \log \mathscr{L} + \sqrt{2\eta}\epsilon_t$$



SGLD in action

SGD vs SGLD


SGD

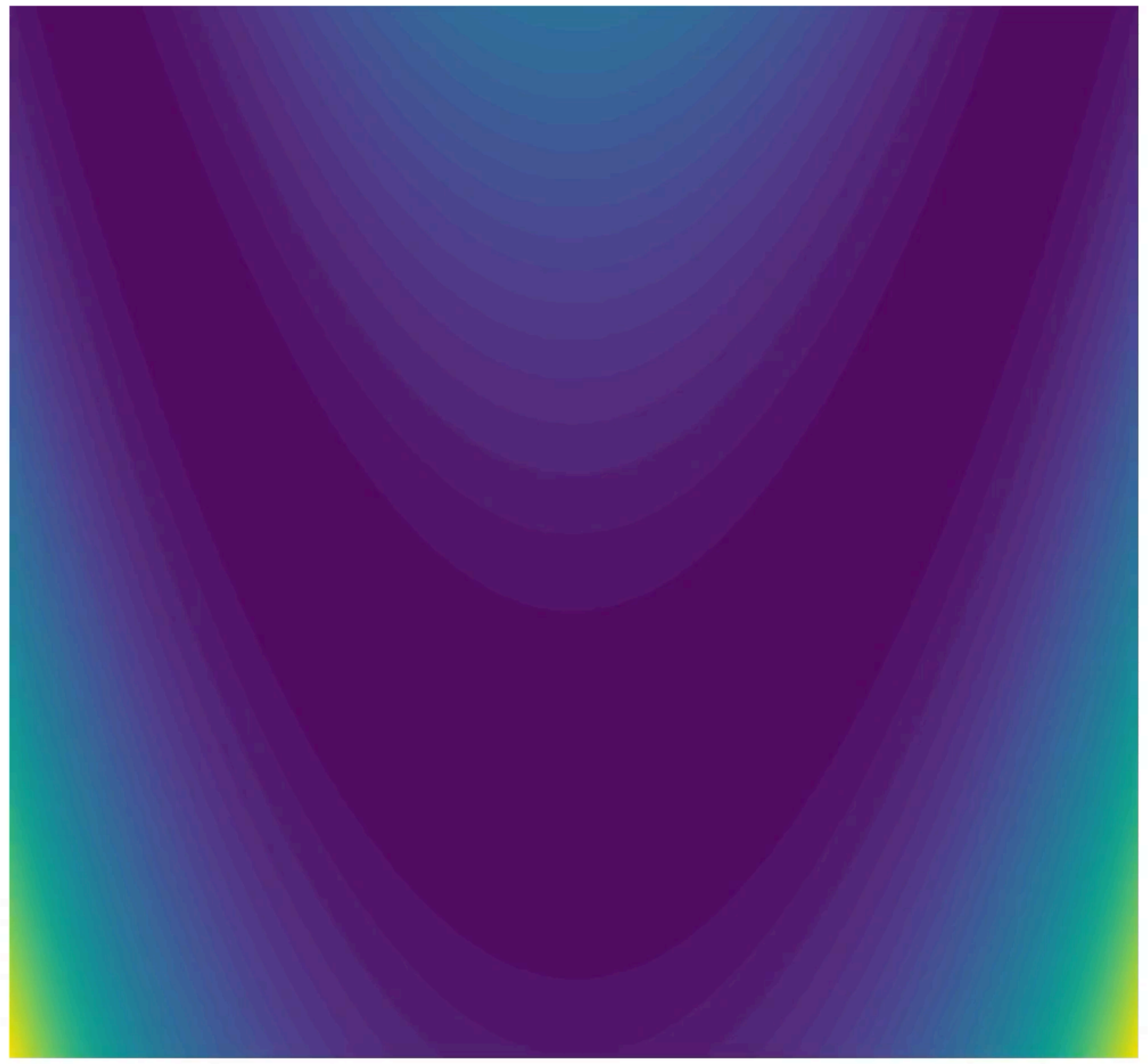
SGD w/ noise

SGLD

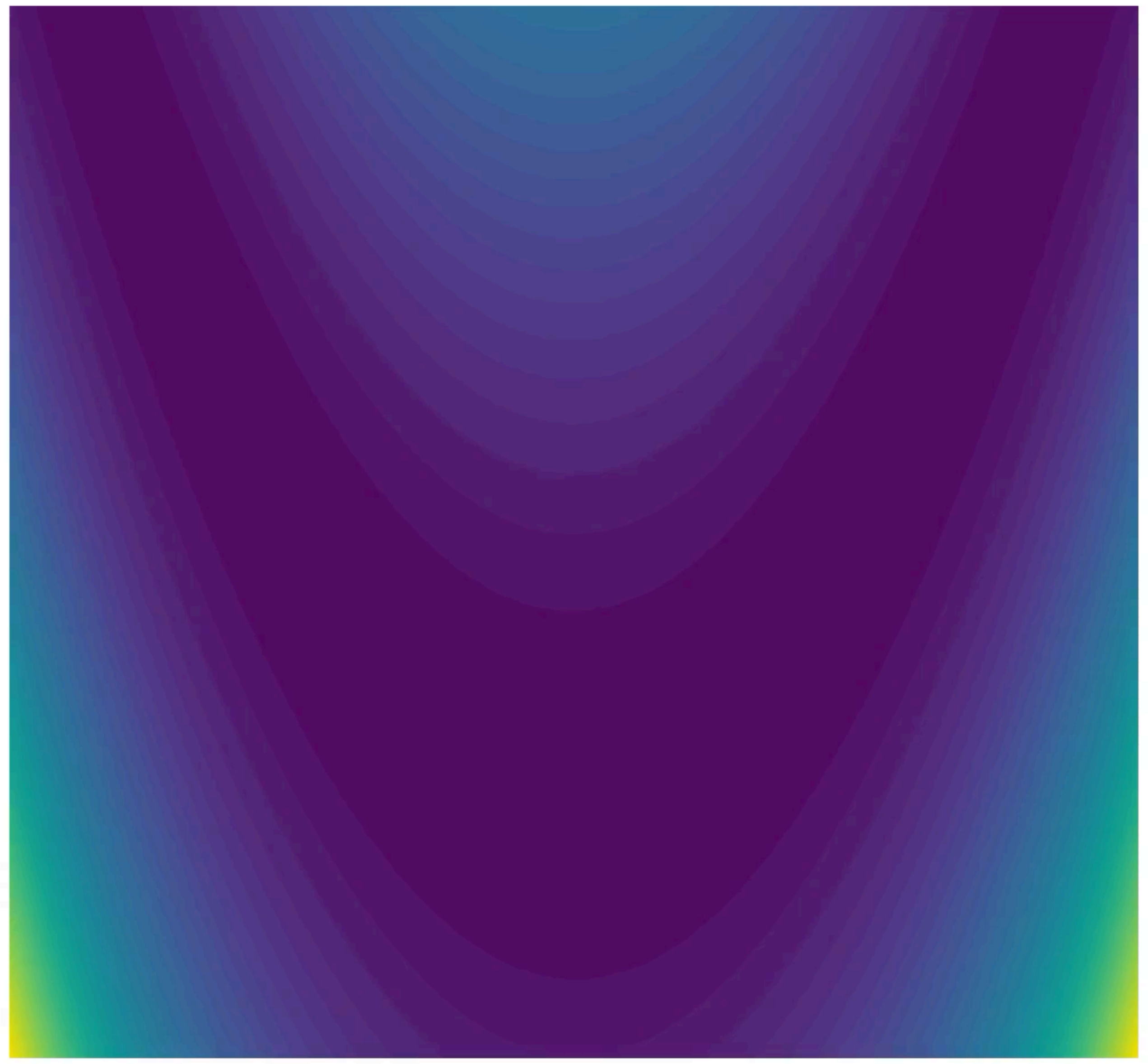
$$\mathscr{L}(x, y) = (1 - x)^2 + 100 * (y - x^2)^2$$

SGD vs SGLD

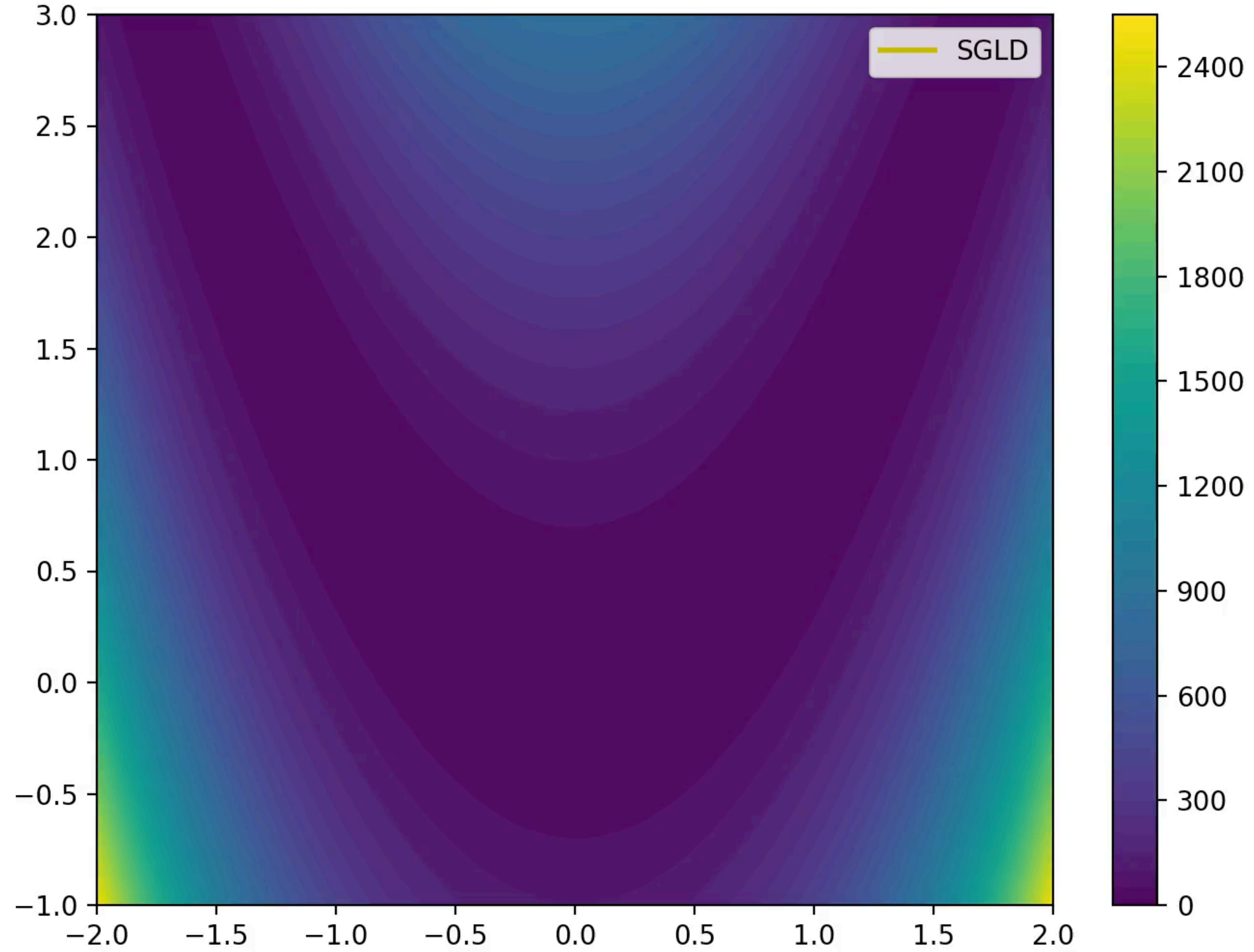
SGD



SGD w/ noise



SGLD

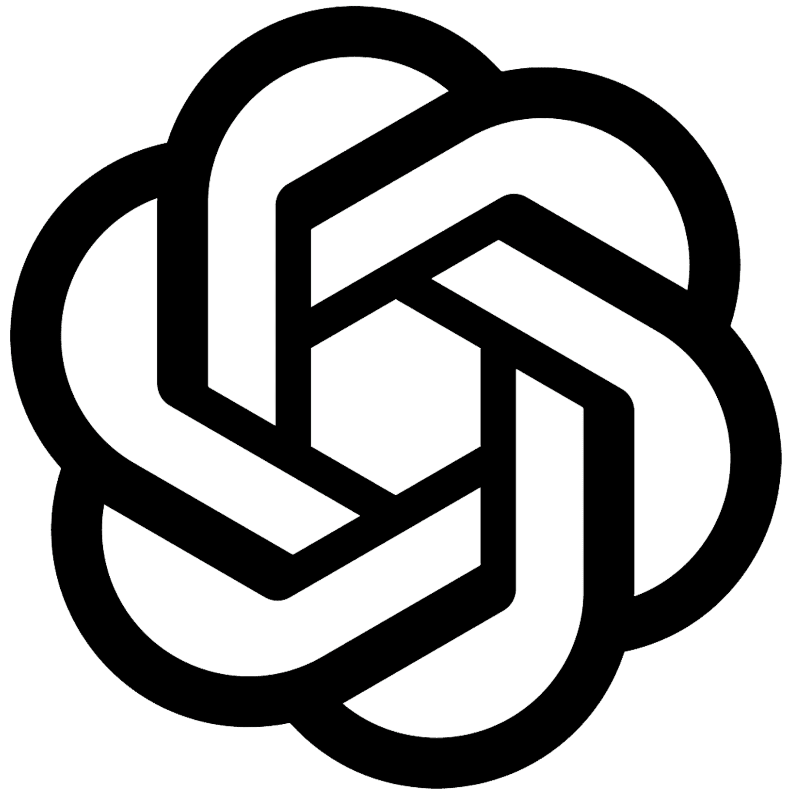


$$\mathcal{L}(x, y) = (1 - x)^2 + 100 * (y - x^2)^2$$

MCMC in Generative AI

What is Generative AI?

Generative AI



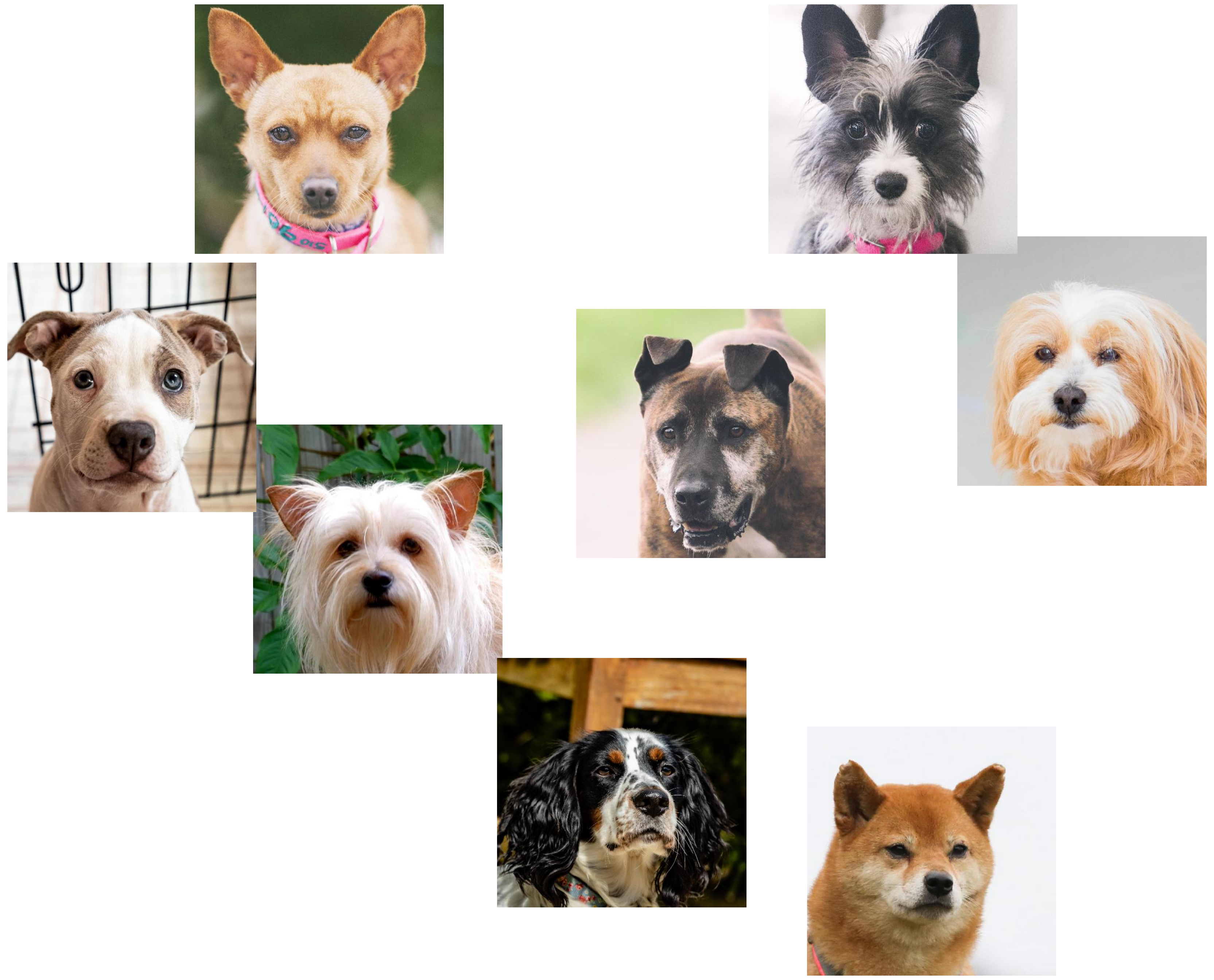
LLAMA 3.2

Generative AI

- Goal: How to learn the underlying distribution of data samples?

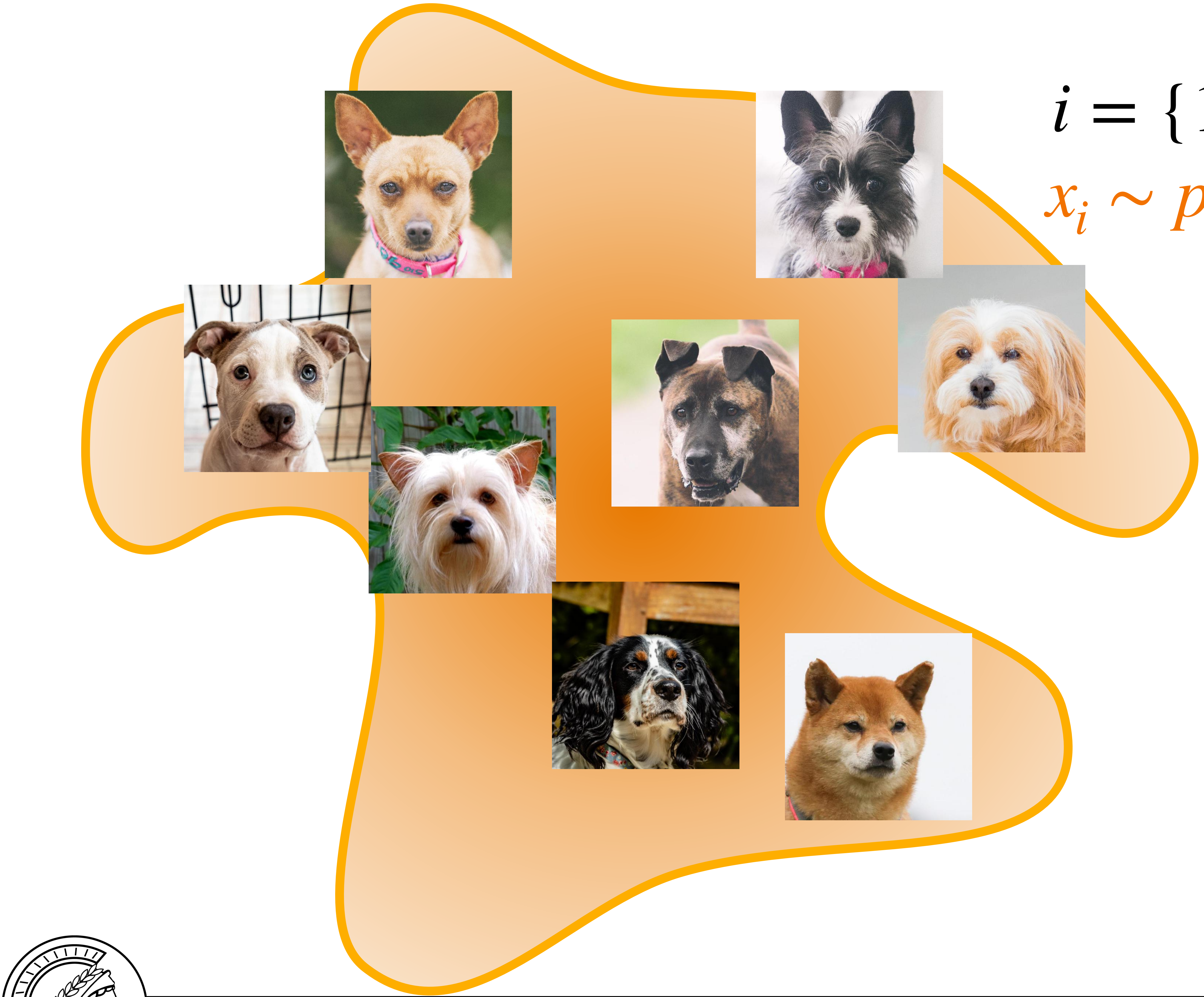
Generative AI

- Goal: How to learn the underlying distribution of data samples?



Generative AI

- Goal: How to learn the underlying distribution of data samples?

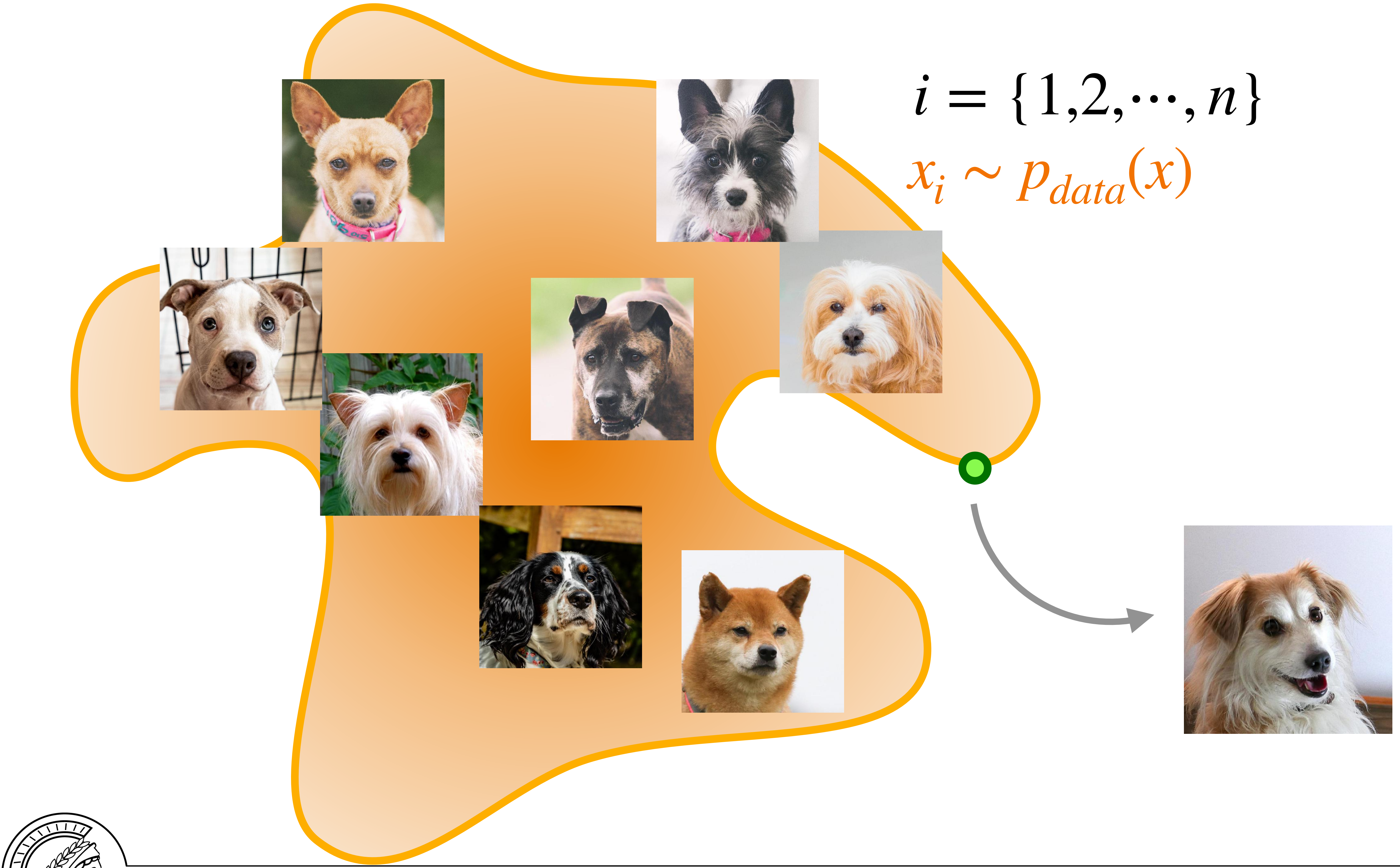


$i = \{1,2,\cdots,n\}$

$x_i \sim p_{data}(x)$

Generative AI

- Goal: How to learn the underlying distribution of data samples?

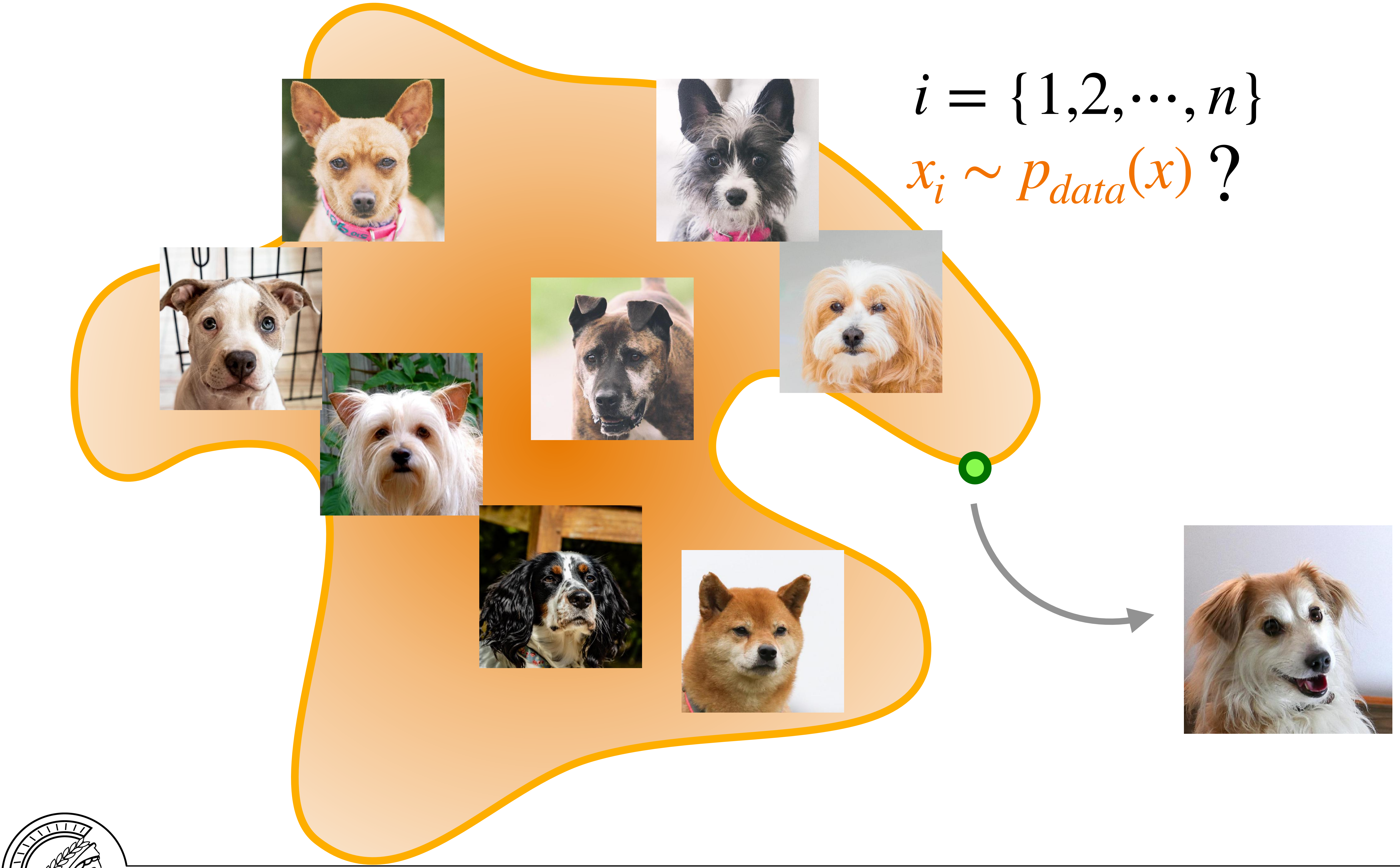


$$i = \{1,2,\cdots,n\}$$

$$x_i \sim p_{data}(x)$$

Generative AI

- Goal: How to learn the underlying distribution of data samples?

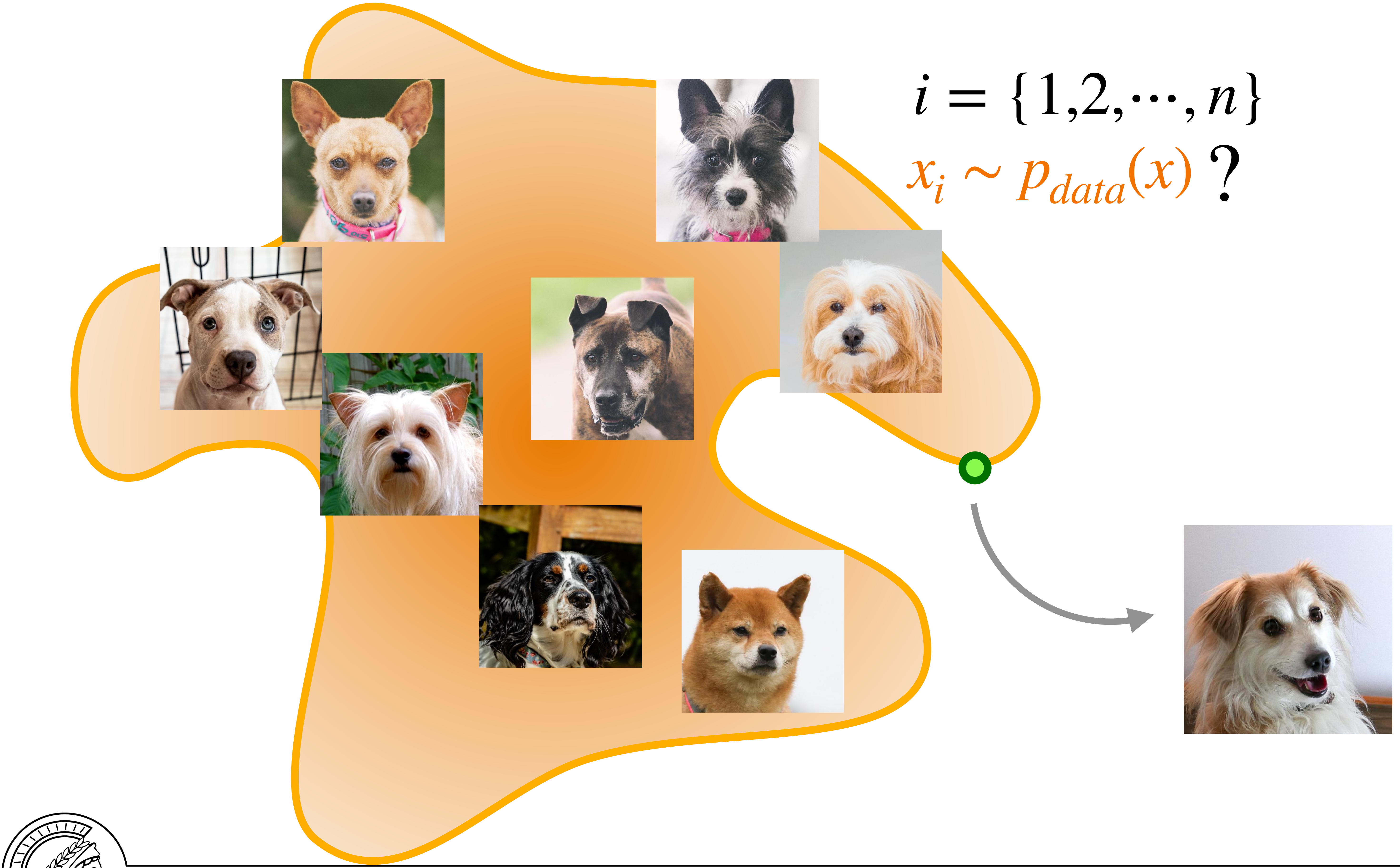


$i = \{1,2,\cdots,n\}$

$x_i \sim p_{data}(x)$?

Generative AI

- Goal: How to learn the underlying distribution of data samples?



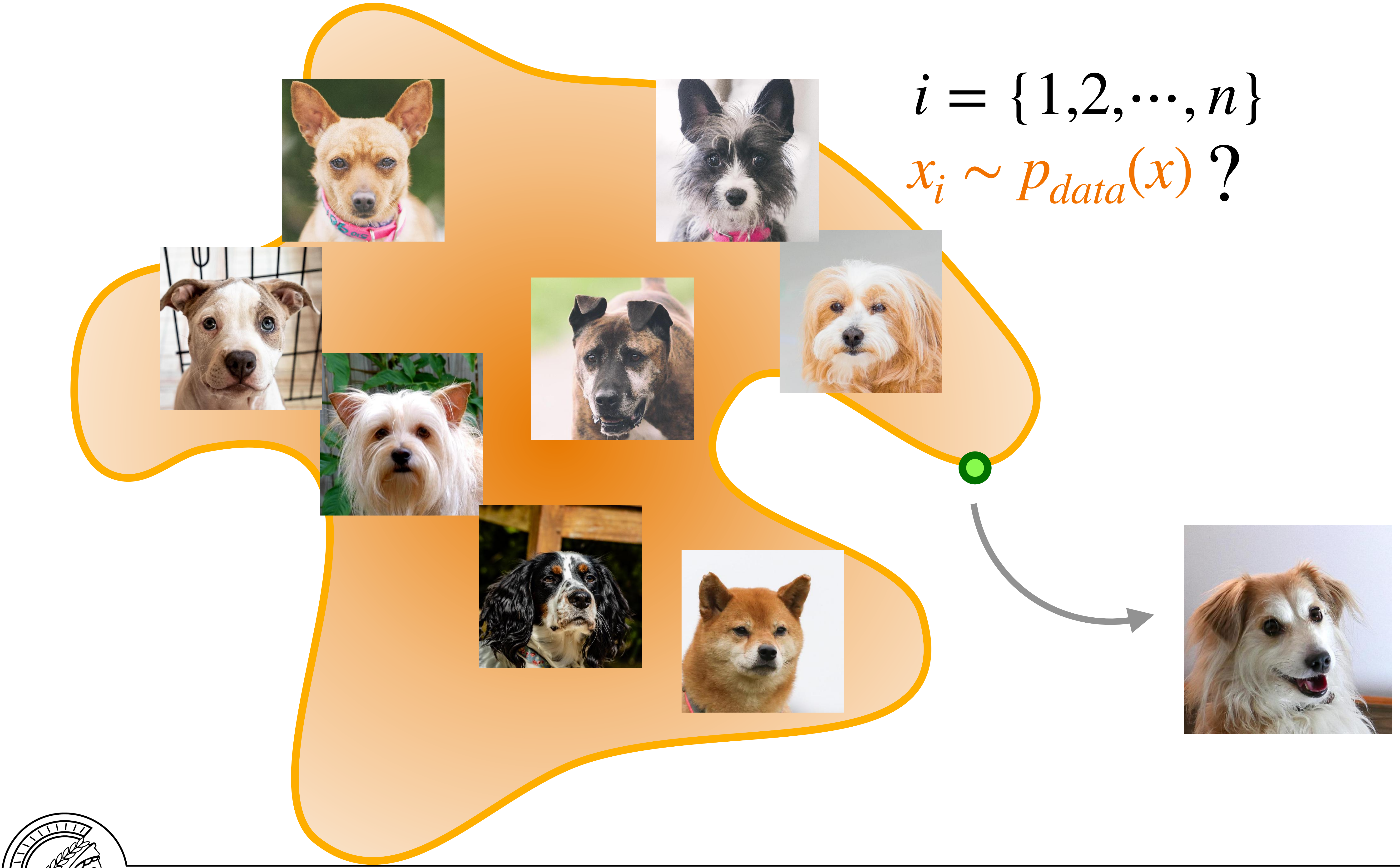
$i = \{1, 2, \cdots, n\}$

$x_i \sim p_{data}(x)$?

How can we sample from an unknown data distribution?

Generative AI

- Goal: How to learn the underlying distribution of data samples?

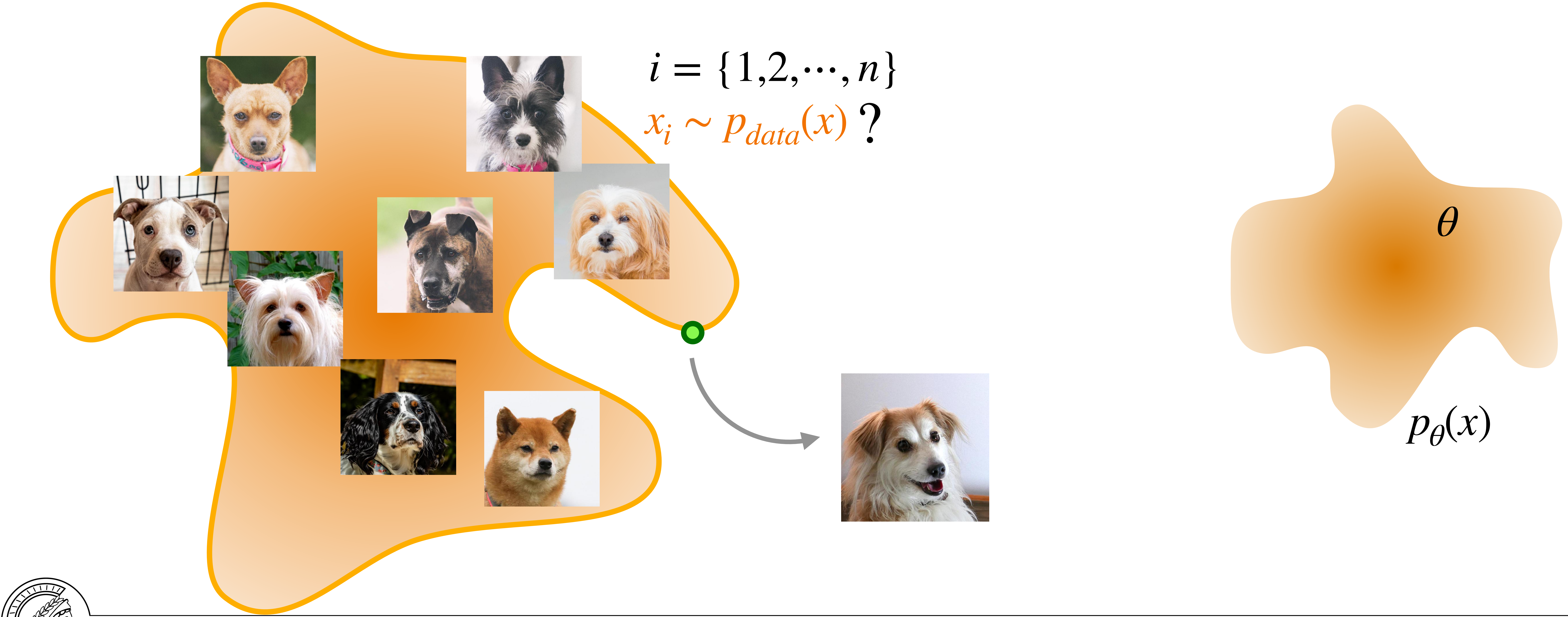


$i = \{1, 2, \cdots, n\}$

$x_i \sim p_{data}(x)$?

Generative AI

- Goal: How to learn the underlying distribution of data samples?

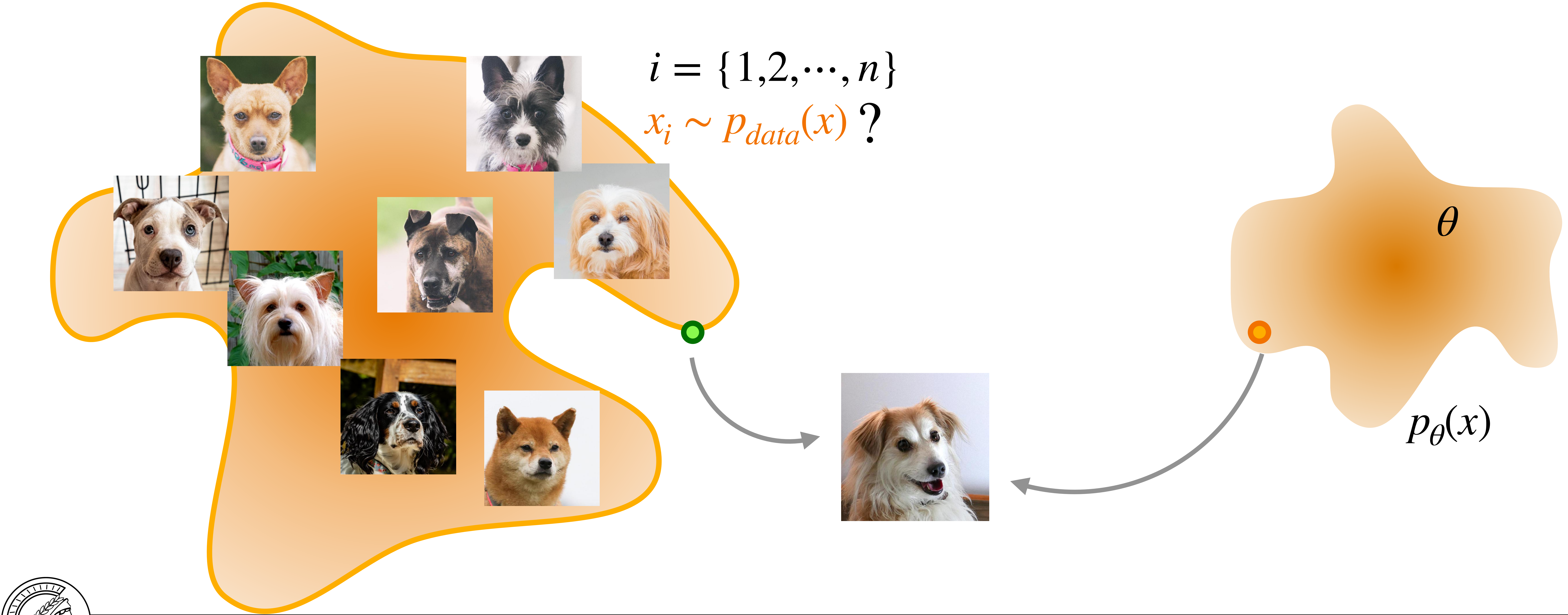


$$i = \{1,2,\cdots,n\}$$

$$x_i \sim p_{data}(x)\ ?$$

Generative AI

- Goal: How to learn the underlying distribution of data samples?



$i = \{1,2,\cdots,n\}$

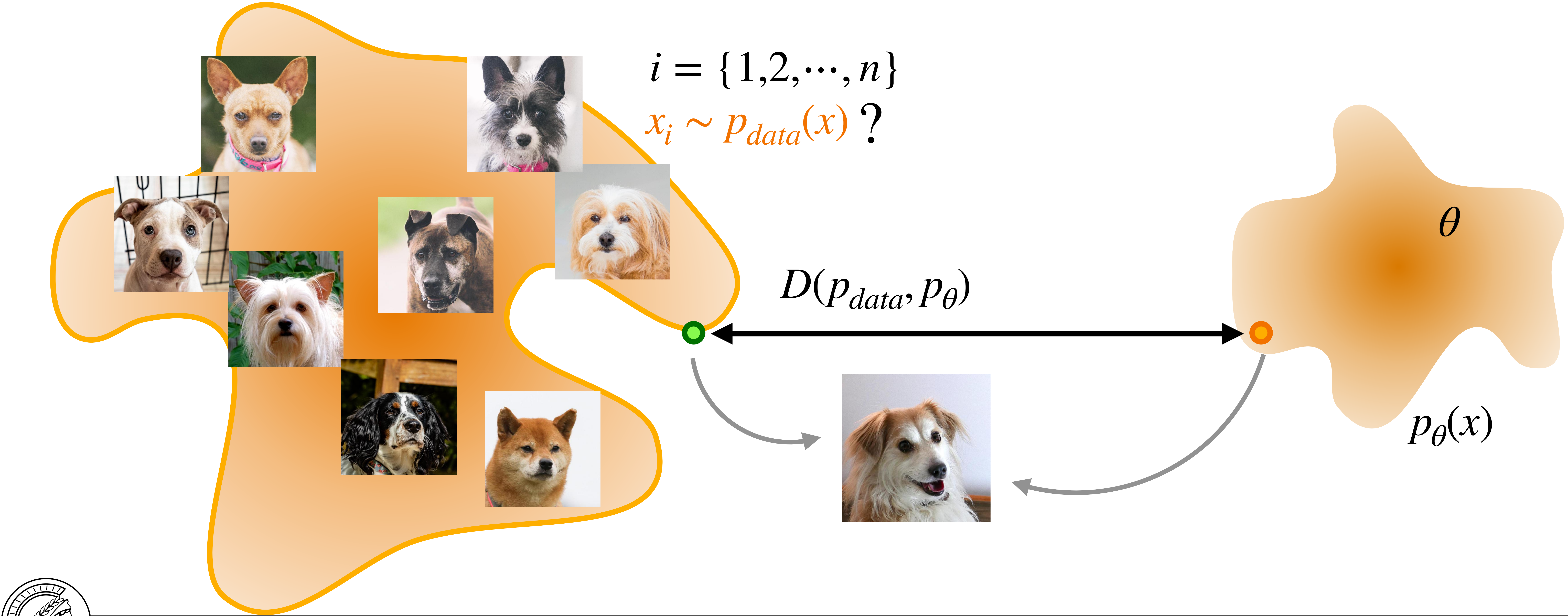
$x_i \sim p_{data}(x)$?

θ

$p_\theta(x)$

Generative AI

- Goal: How to learn the underlying distribution of data samples?



$i = \{1,2,\cdots,n\}$

$x_i \sim p_{data}(x)$?

$D(p_{data}, p_\theta)$

θ

$p_\theta(x)$

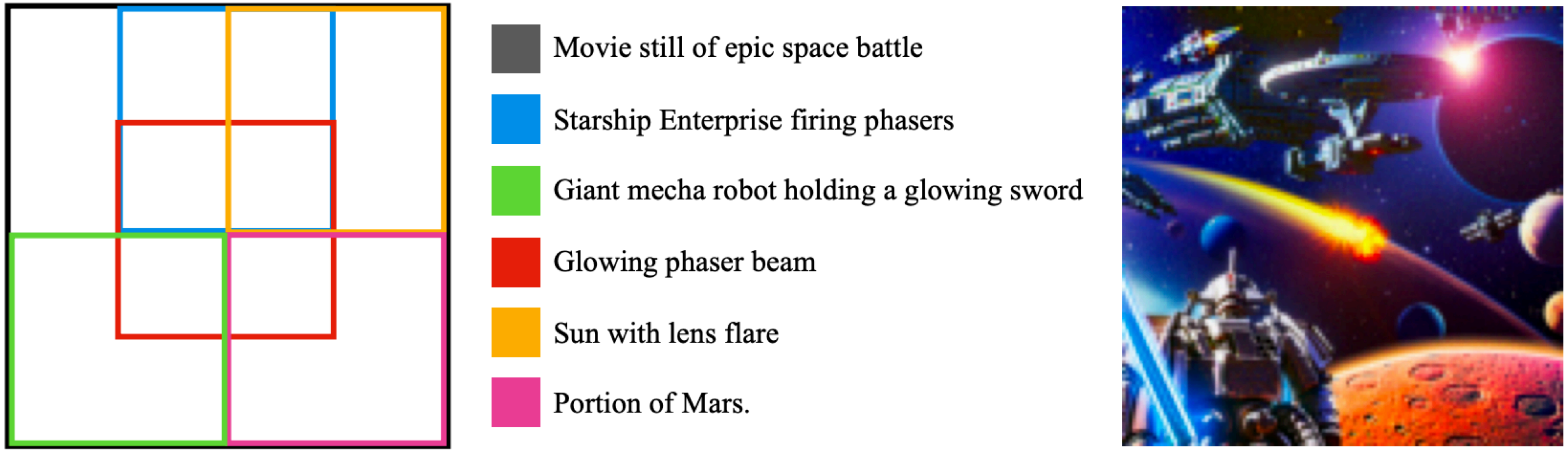
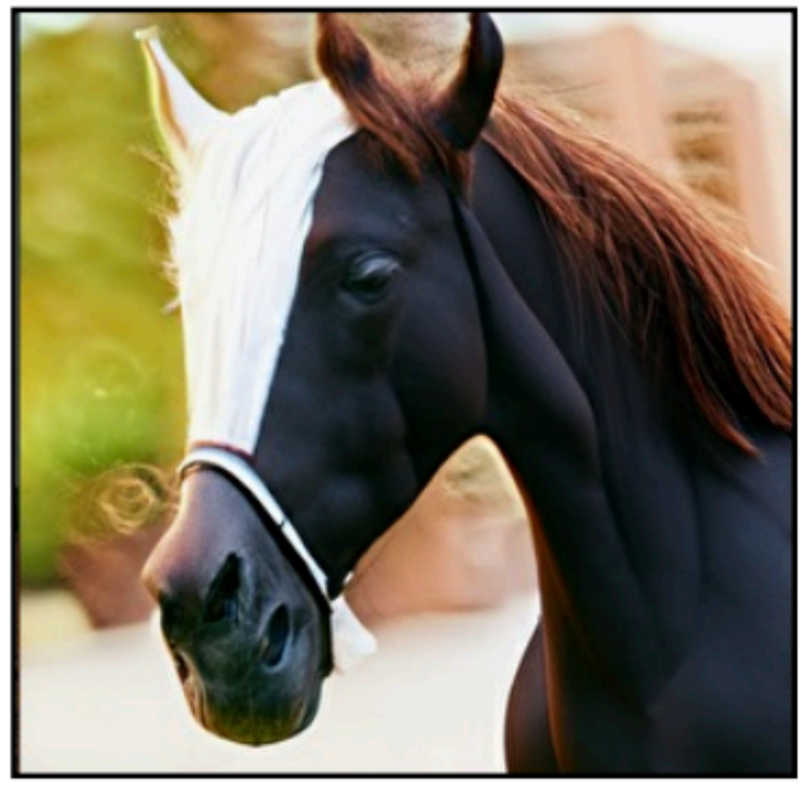

⬛ Movie still of epic space battle
🟦 Starship Enterprise firing phasers
🟩 Giant mecha robot holding a glowing sword
🟥 Glowing phaser beam
🟧 Sun with lens flare
🟪 Portion of Mars.

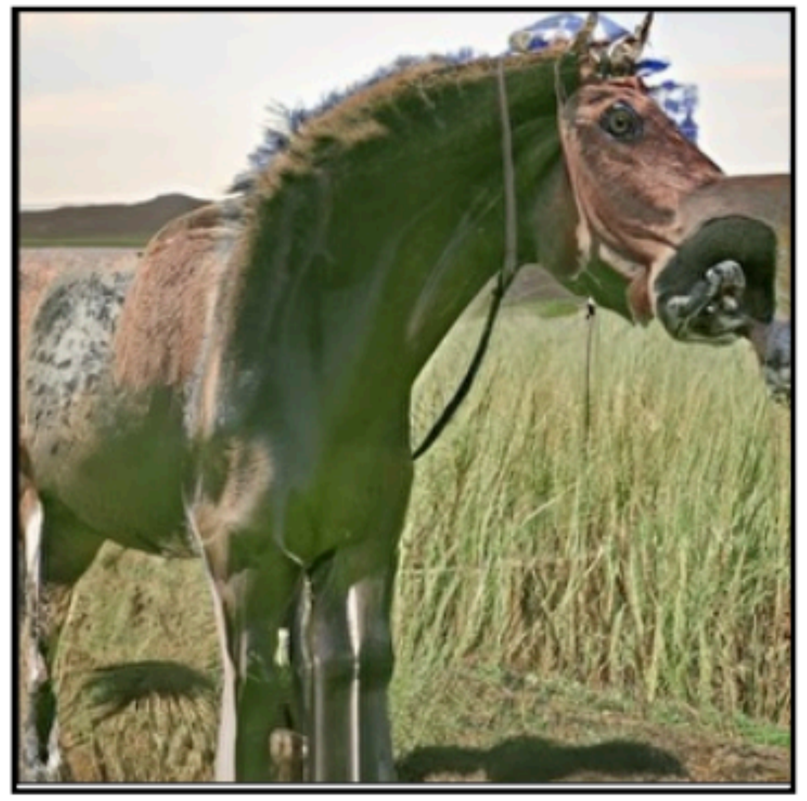
Figure 8: **Composition enables controllable image tapestries.**

Reduce, Reuse, Recycle: Compositional Generation with Energy-Based Diffusion Models and MCMC (Du et al. 2024)

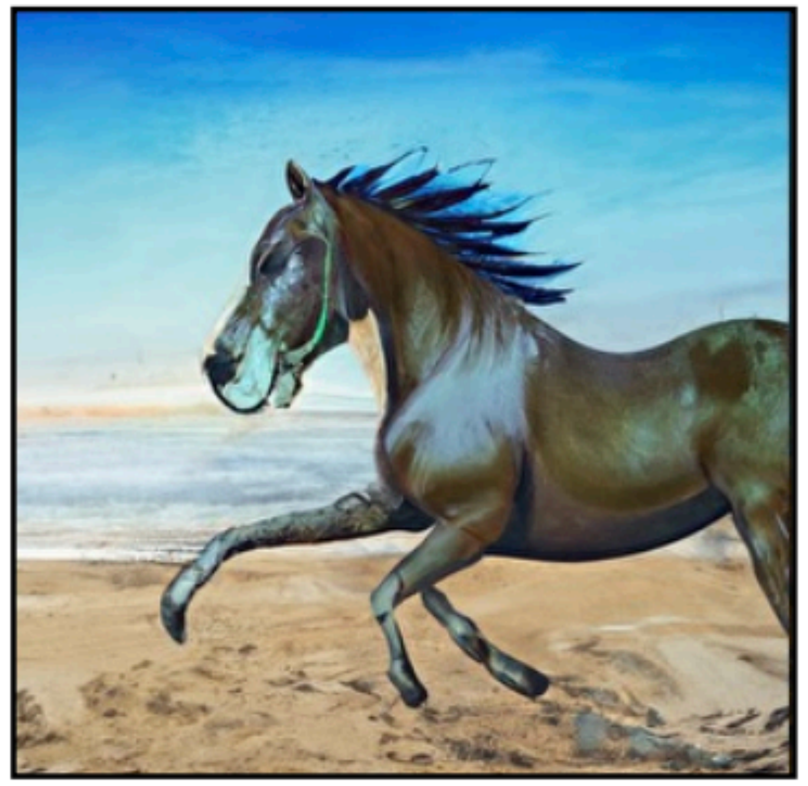
“A horse”



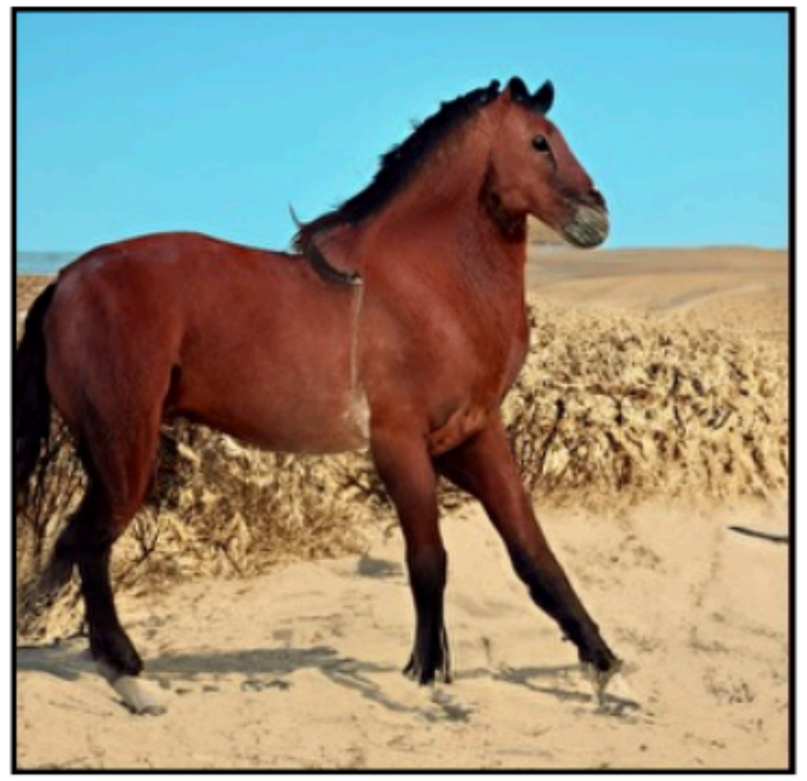
“A horse”
AND
“Grass plains”



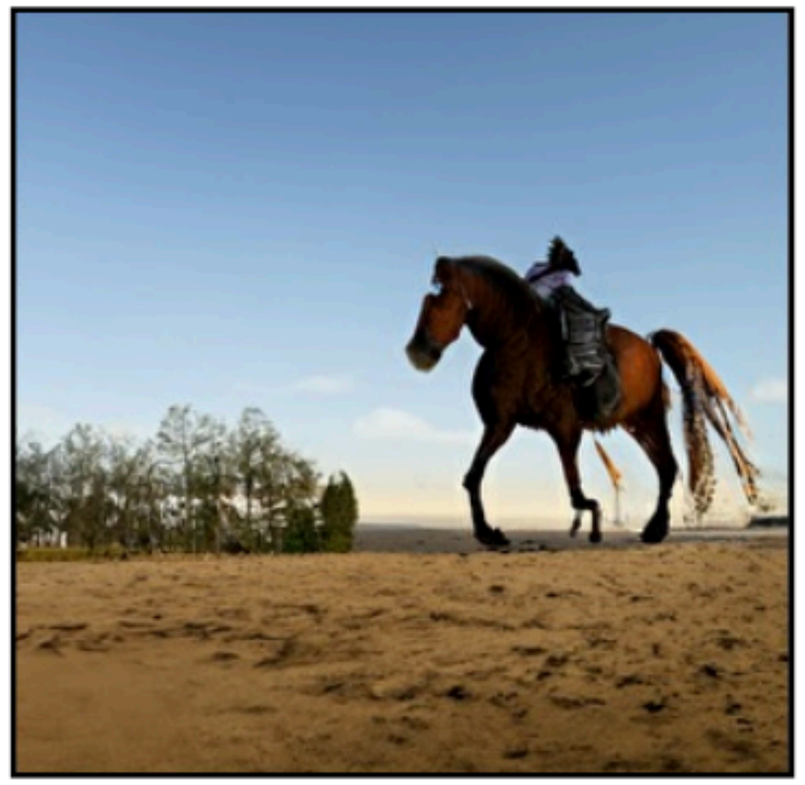
“A horse”
AND
“A sandy beach”



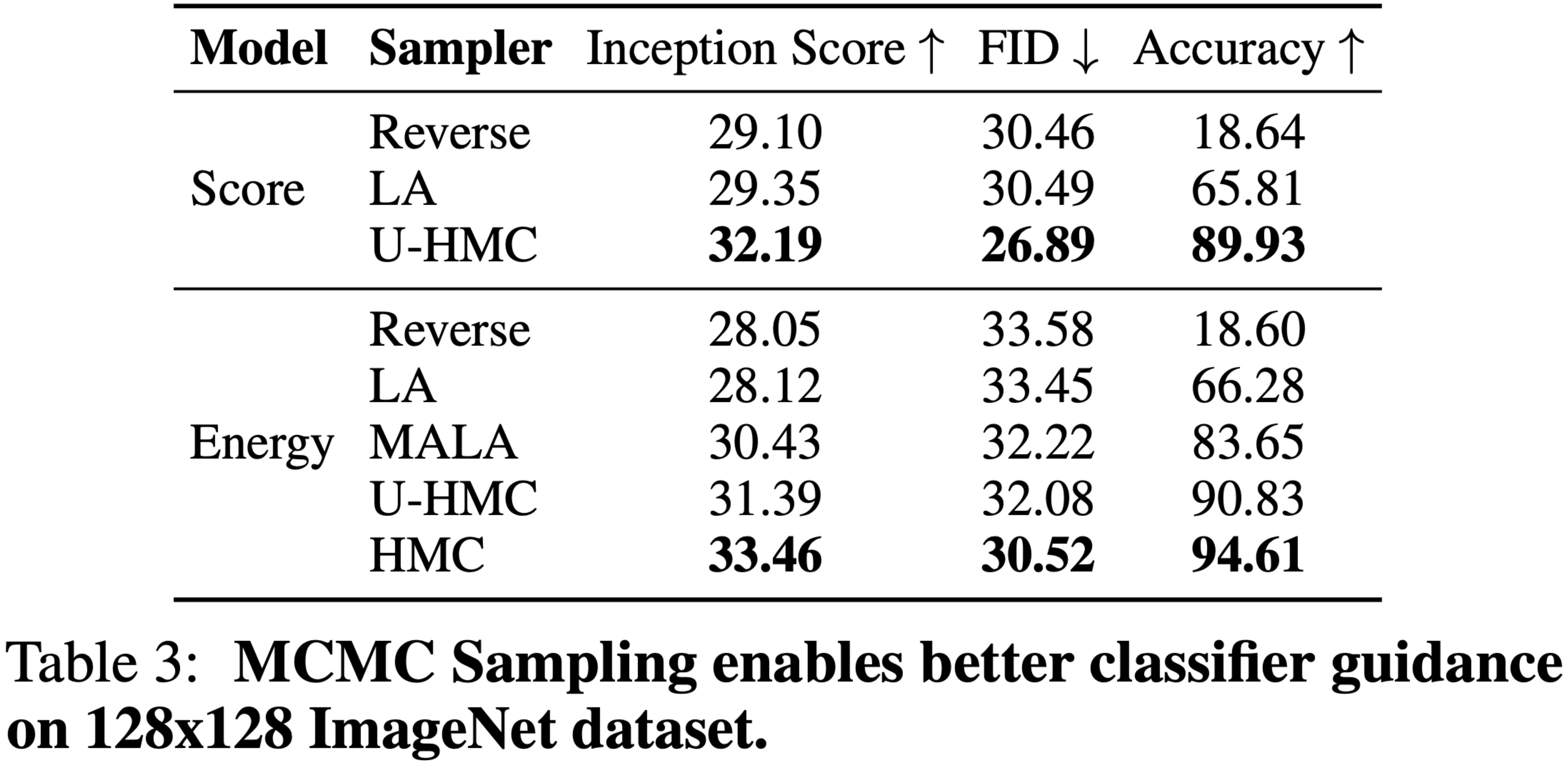
“A horse” AND
(“A sandy beach” OR
“Grass plains”)



“A horse” AND
(“A sandy beach” OR
“Grass plains”)
AND (NOT (“Sunny ”))



Reduce, Reuse, Recycle: Compositional Generation with Energy-Based Diffusion Models and MCMC (Du et al. 2024)



Model	**Sampler**	Inception Score ↑	FID ↓	Accuracy ↑
Score	Reverse	29.10	30.46	18.64
	LA	29.35	30.49	65.81
	U-HMC	**32.19**	**26.89**	**89.93**
Energy	Reverse	28.05	33.58	18.60
	LA	28.12	33.45	66.28
	MALA	30.43	32.22	83.65
	U-HMC	31.39	32.08	90.83
	HMC	**33.46**	**30.52**	**94.61**

Table 3: **MCMC Sampling enables better classifier guidance on 128x128 ImageNet dataset.**

Reduce, Reuse, Recycle: Compositional Generation with Energy-Based Diffusion Models and MCMC (Du et al. 2024)

From VAEs to Diffusion models

From VAEs to Diffusion models

Variational Autoencoders (VAEs)

From VAEs to Diffusion models

Variational Autoencoders (VAEs)

Energy-based models (EBMs)

MCMC methods for EBMs

From VAEs to Diffusion models

Variational Autoencoders (VAEs)

Energy-based models (EBMs)

MCMC methods for EBMs

Score-based Generative models (SBGMs)

MCMC methods for SBGMs

From VAEs to Diffusion models

Variational Autoencoders (VAEs)

Energy-based models (EBMs)

MCMC methods for EBMs

Score-based Generative models (SBGMs)

MCMC methods for SBGMs

SDE-based diffusion models

From VAEs to Diffusion models

Variational Autoencoders (VAEs)

Energy-based models (EBMs)

MCMC methods for EBMs

Score-based Generative models (SBGMs)

MCMC methods for SBGMs

SDE-based diffusion models

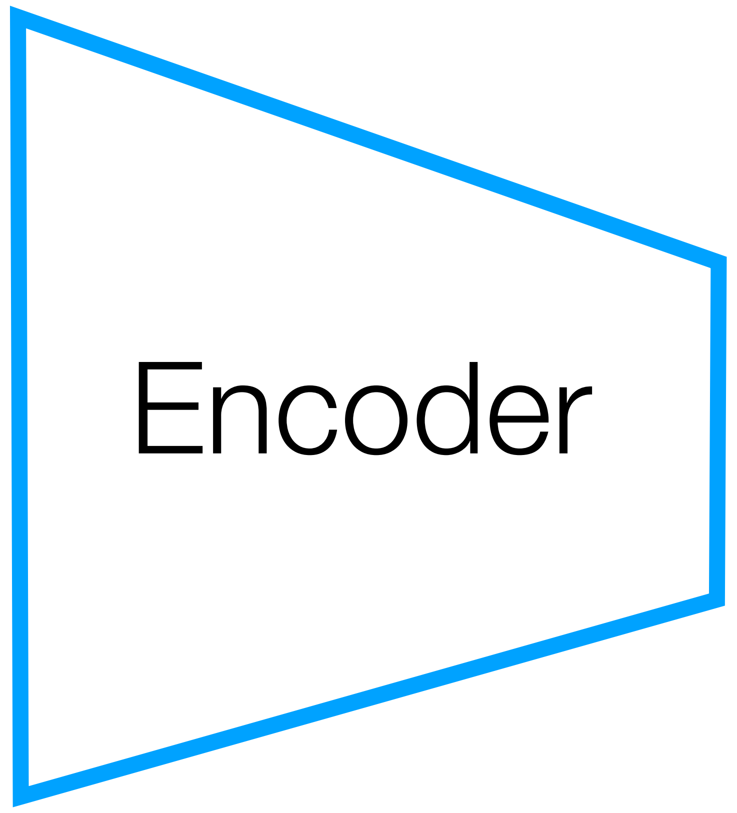
Variational Autoencoders

Variational Autoencoders

VAEs

Variational Autoencoders

VAEs



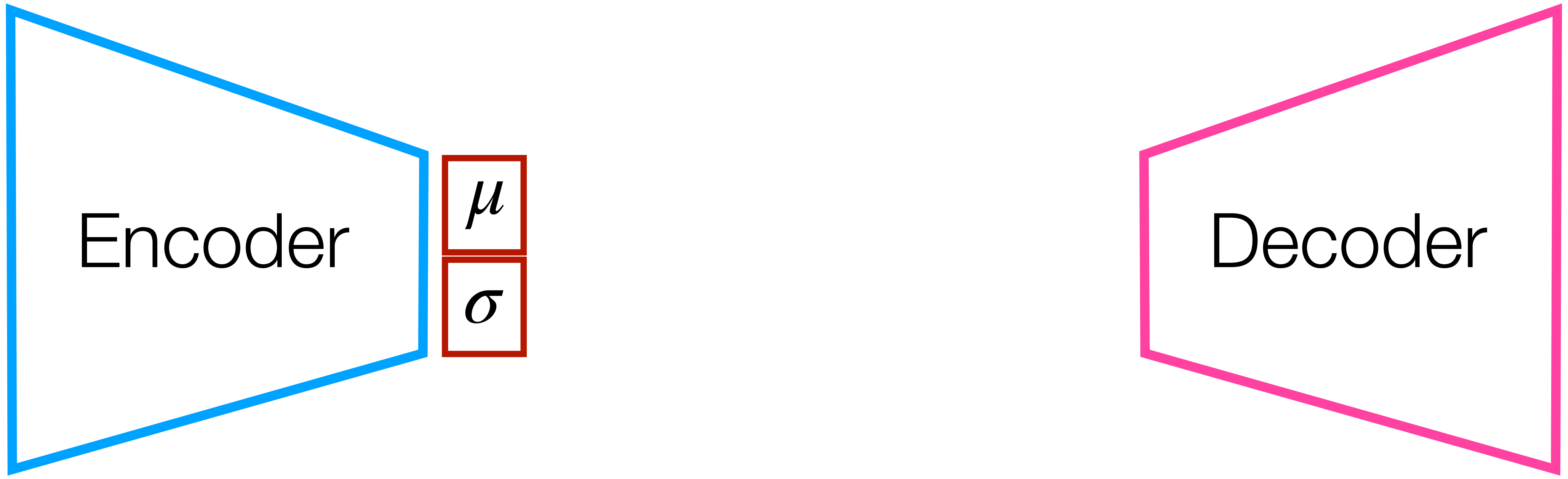
Variational Autoencoders

VAEs

Encoder
Decoder

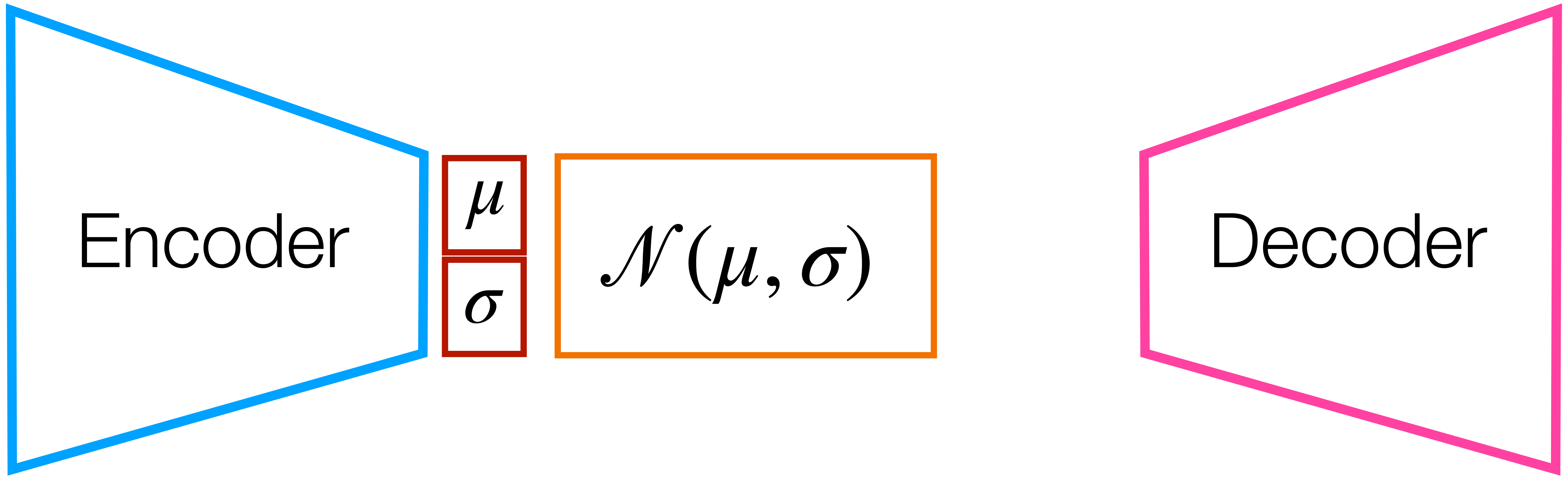
Variational Autoencoders

VAEs



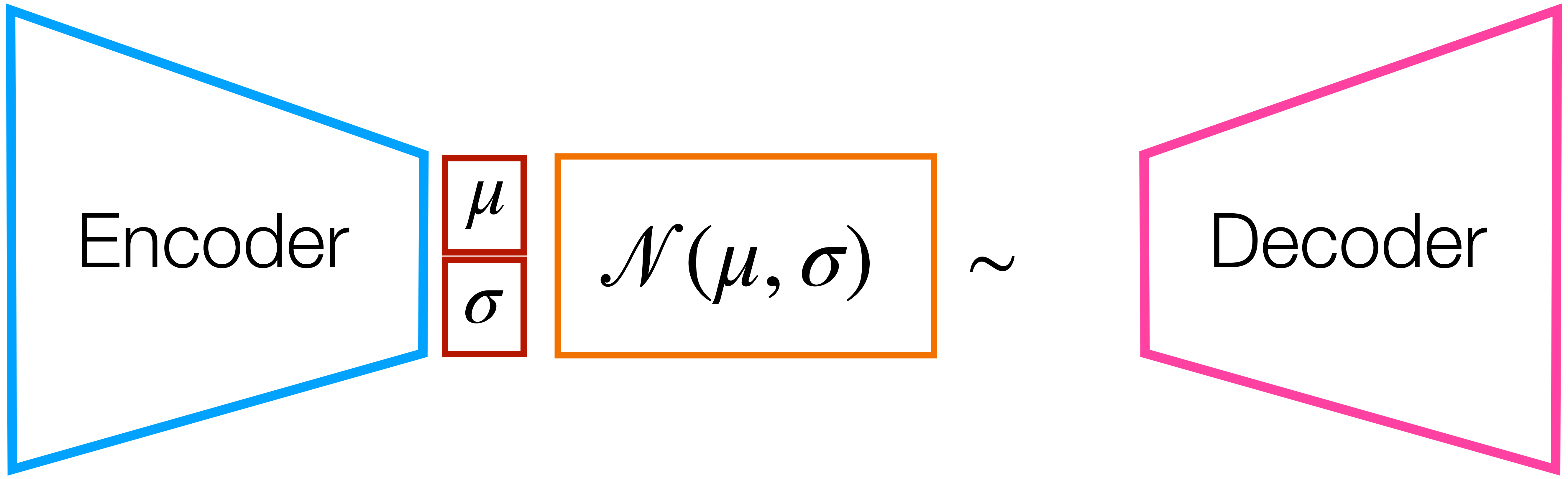
Variational Autoencoders

VAEs



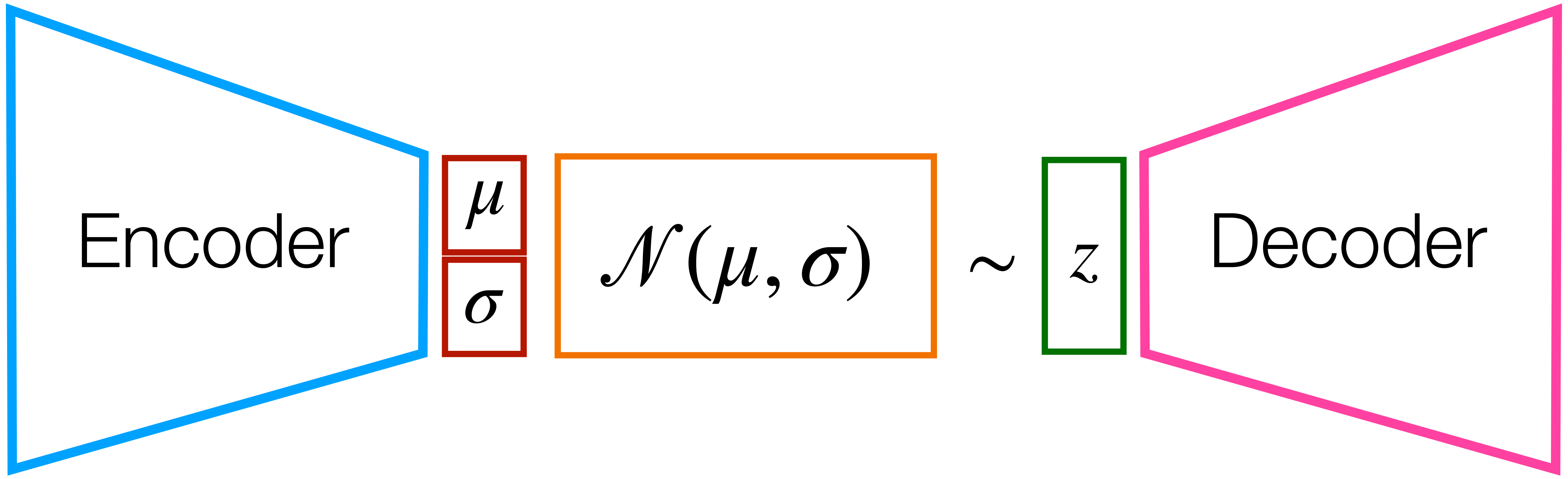
Variational Autoencoders

VAEs



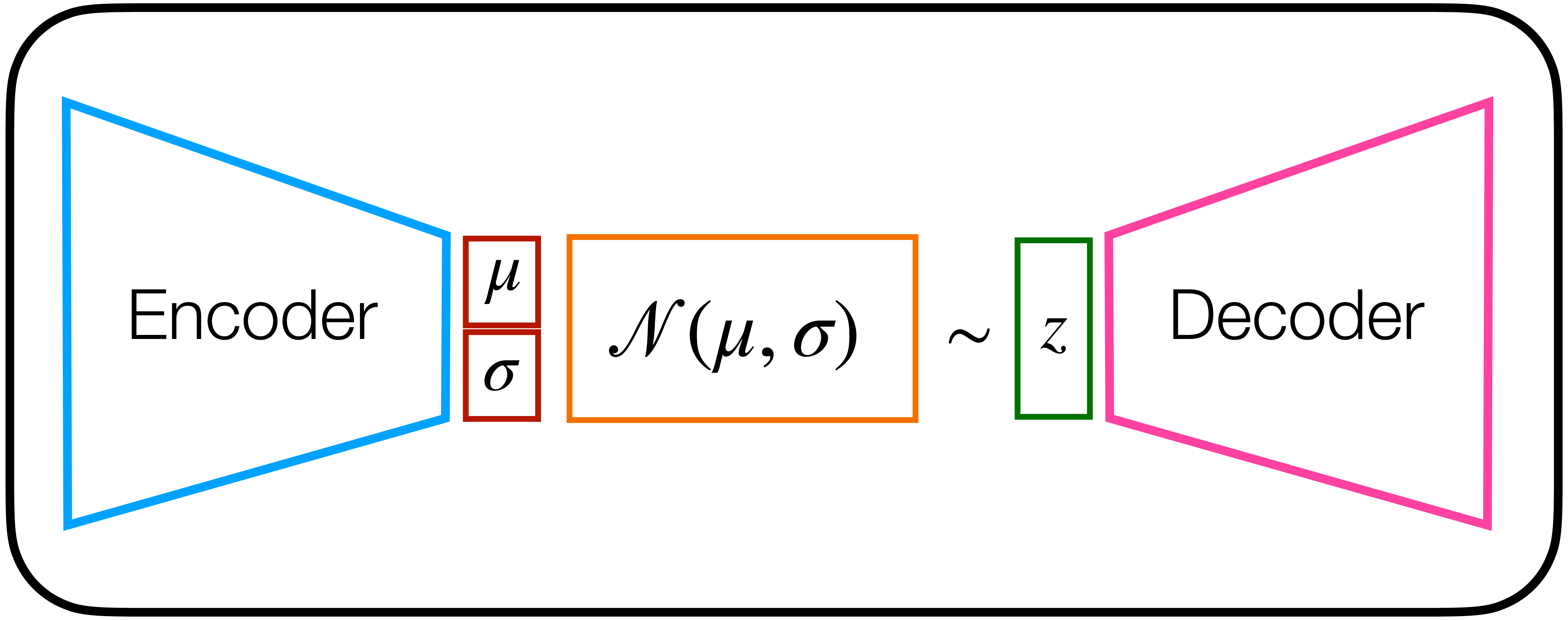
Variational Autoencoders

VAEs



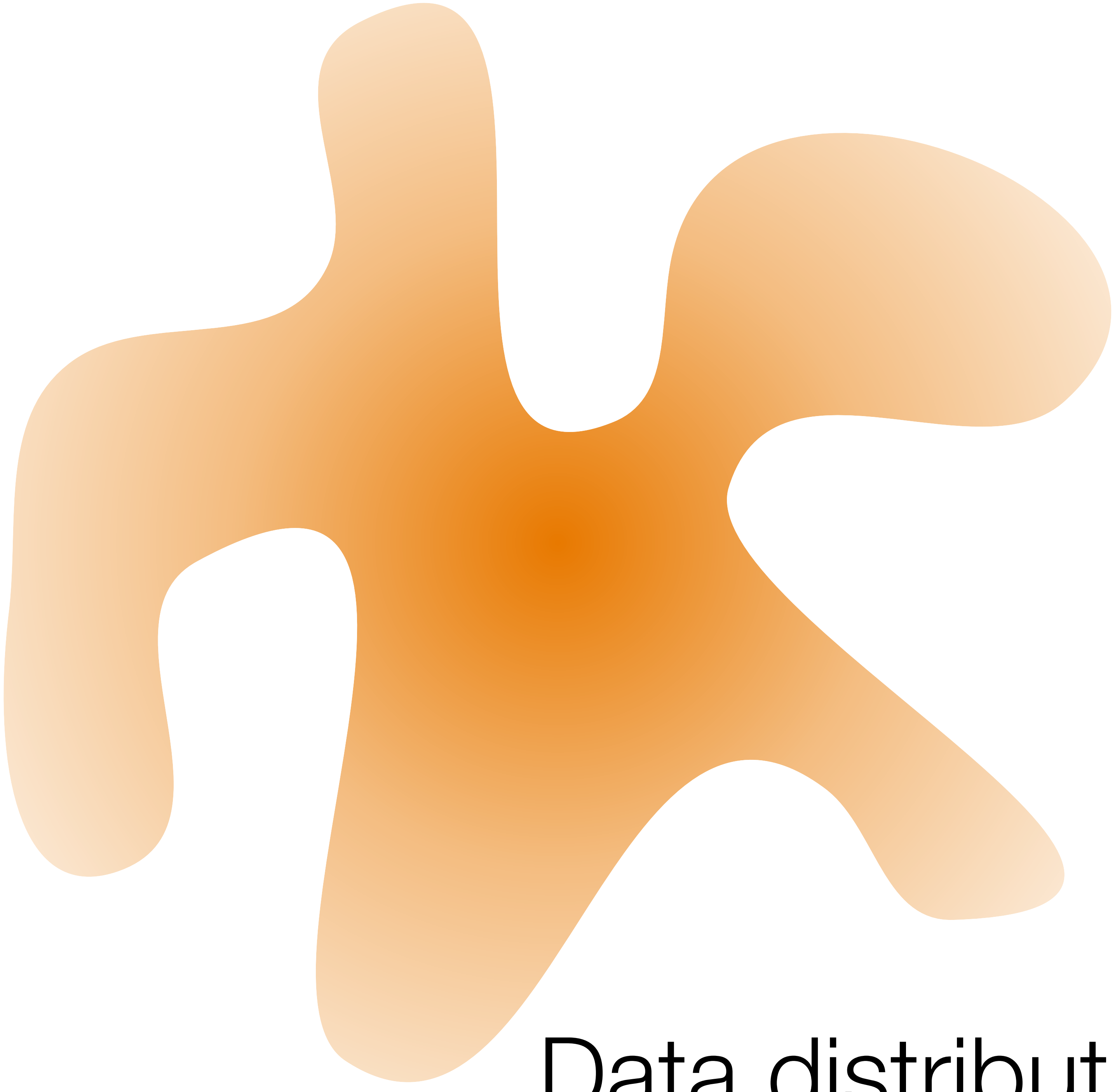
Variational Autoencoders

VAEs



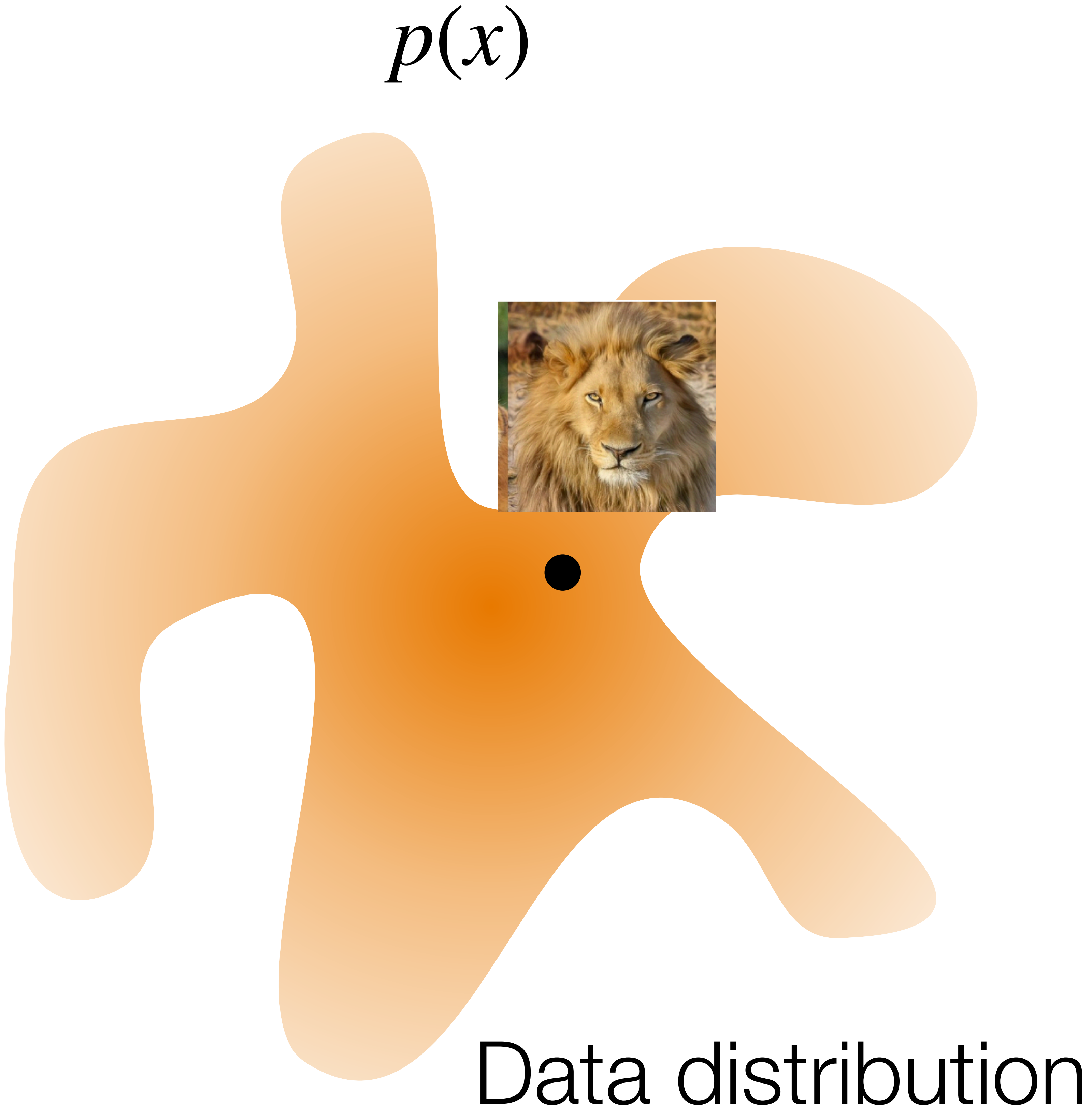
Our goal is to generate samples from an unknown distribution

$p(x)$



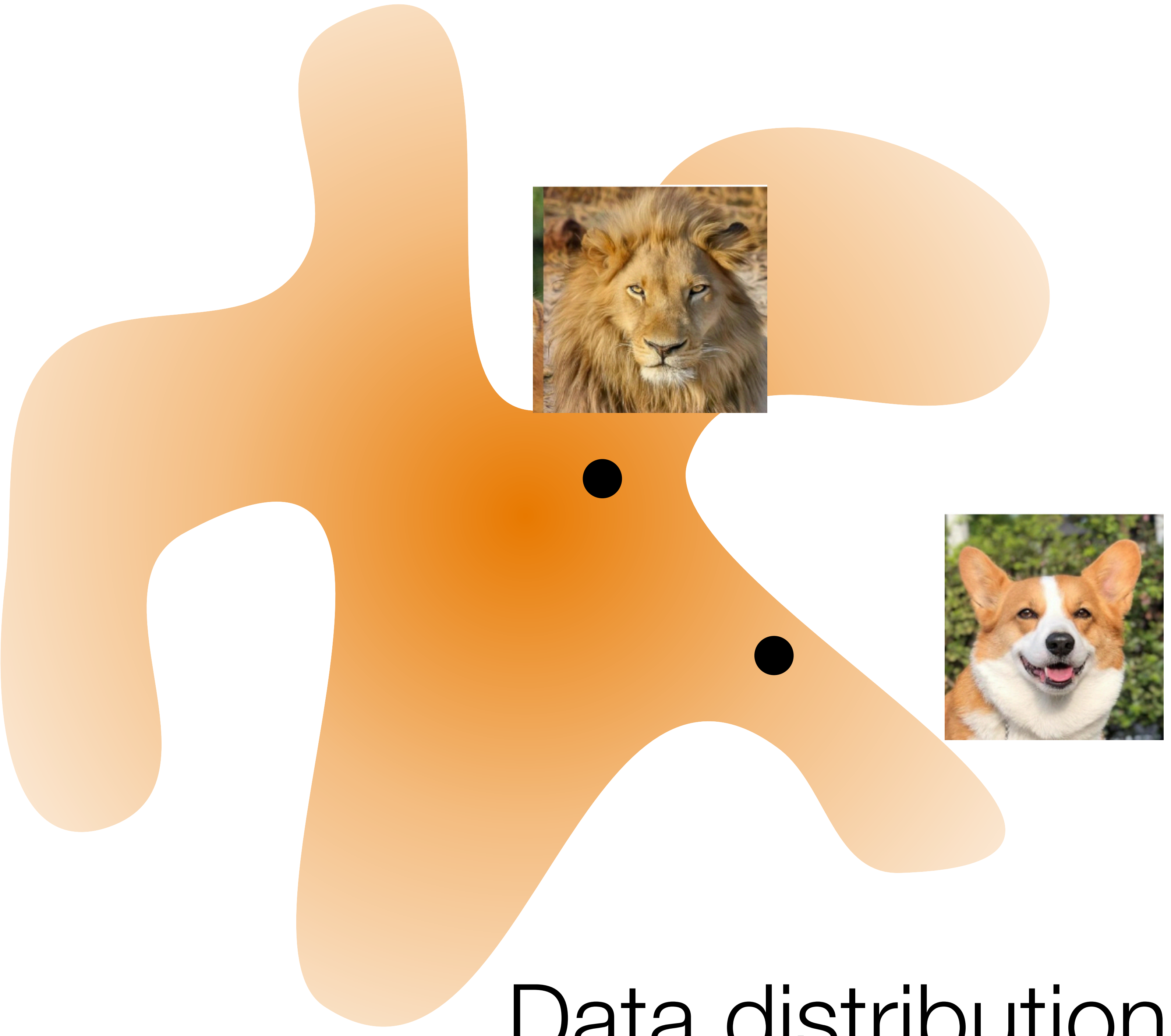
Data distribution

Our goal is to generate samples from an unknown distribution



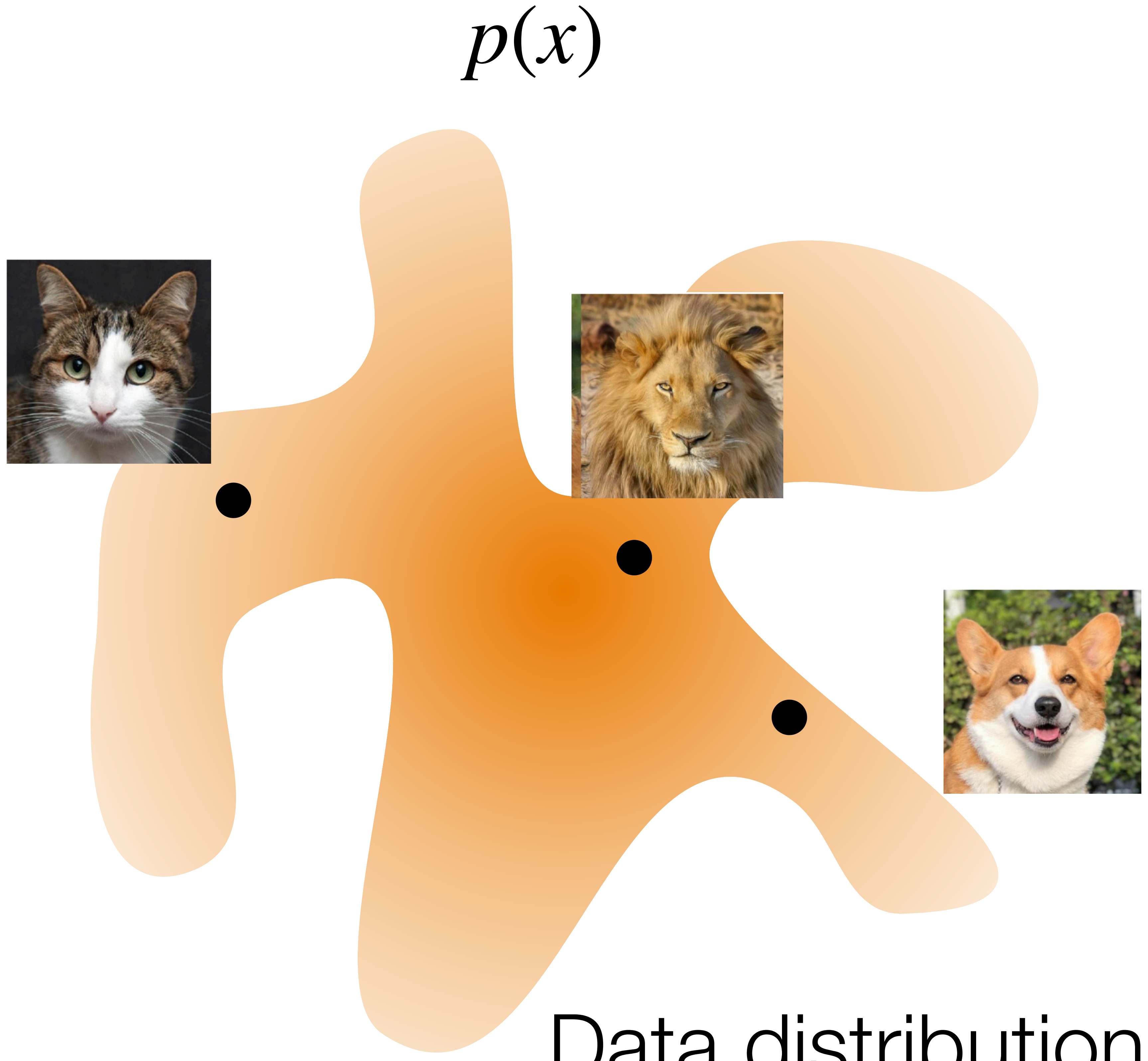
Our goal is to generate samples from an unknown distribution

$p(x)$



Data distribution

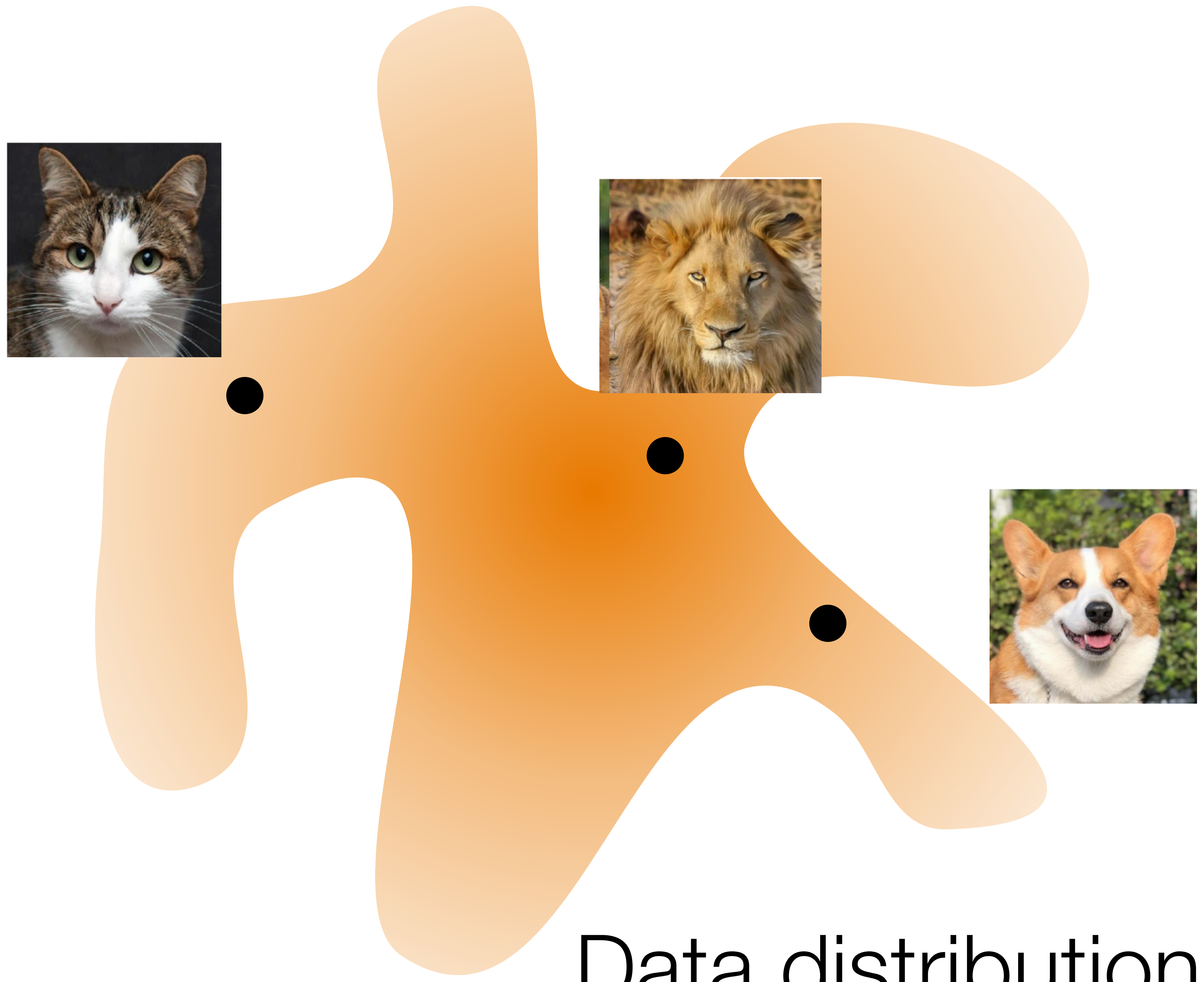
Our goal is to generate samples from an unknown distribution



Data distribution

Our goal is to generate samples from an unknown distribution

$p(x)$

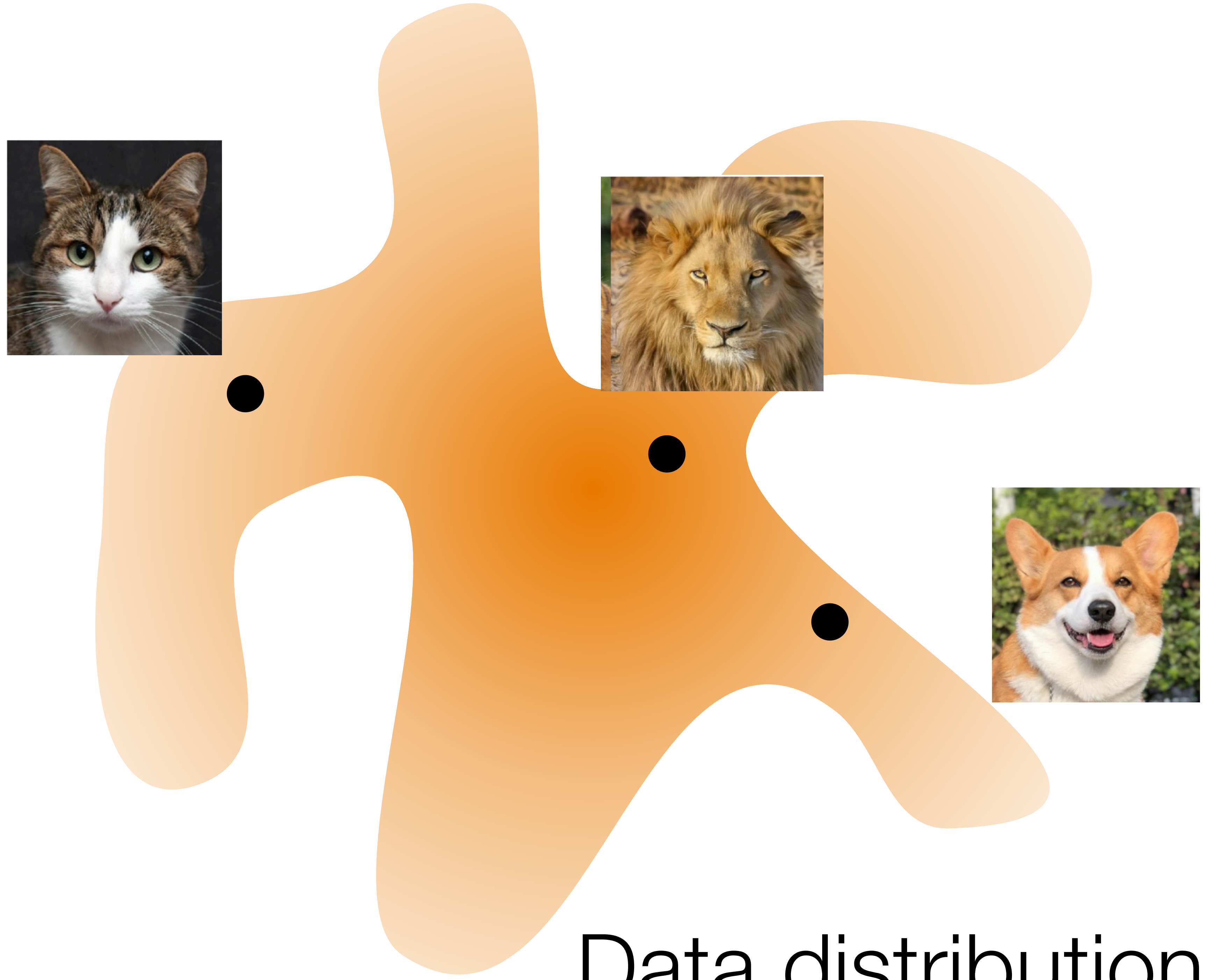


Data distribution

$p(z)$

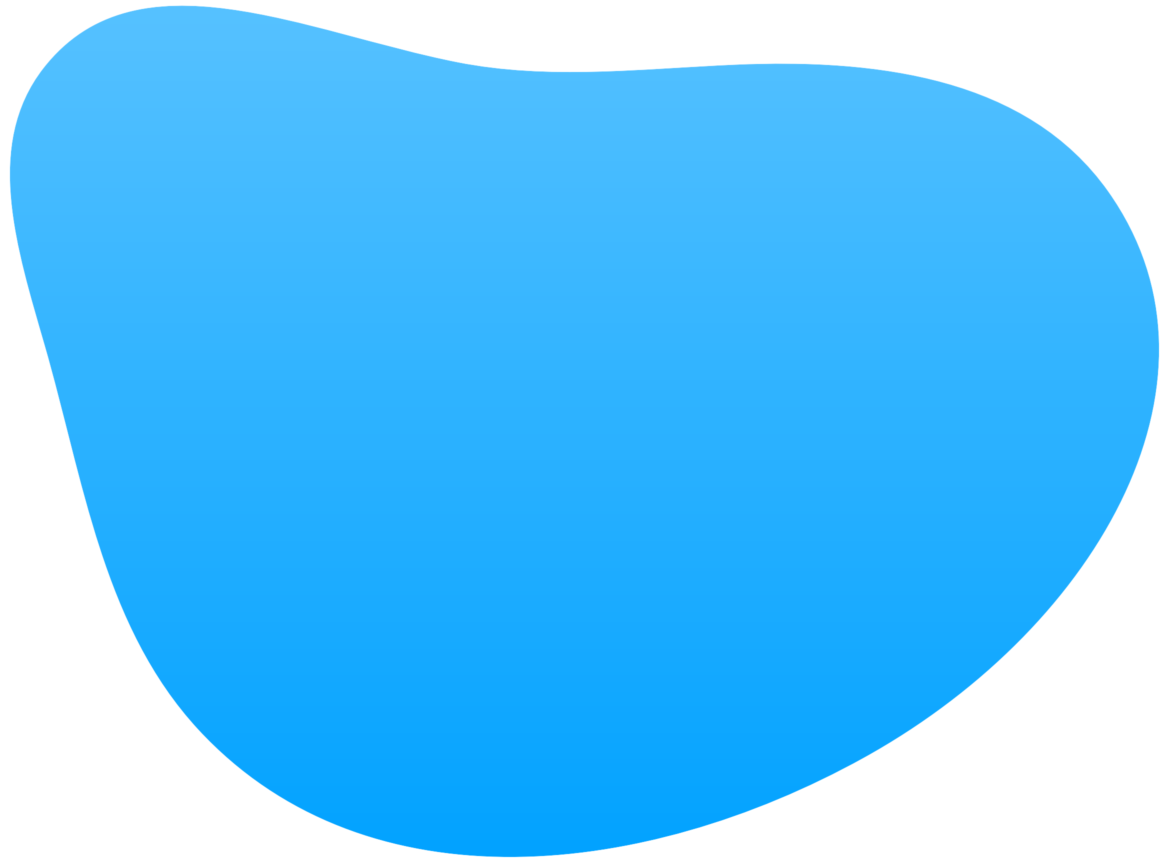
Our goal is to generate samples from an unknown distribution

$p(x)$



Data distribution

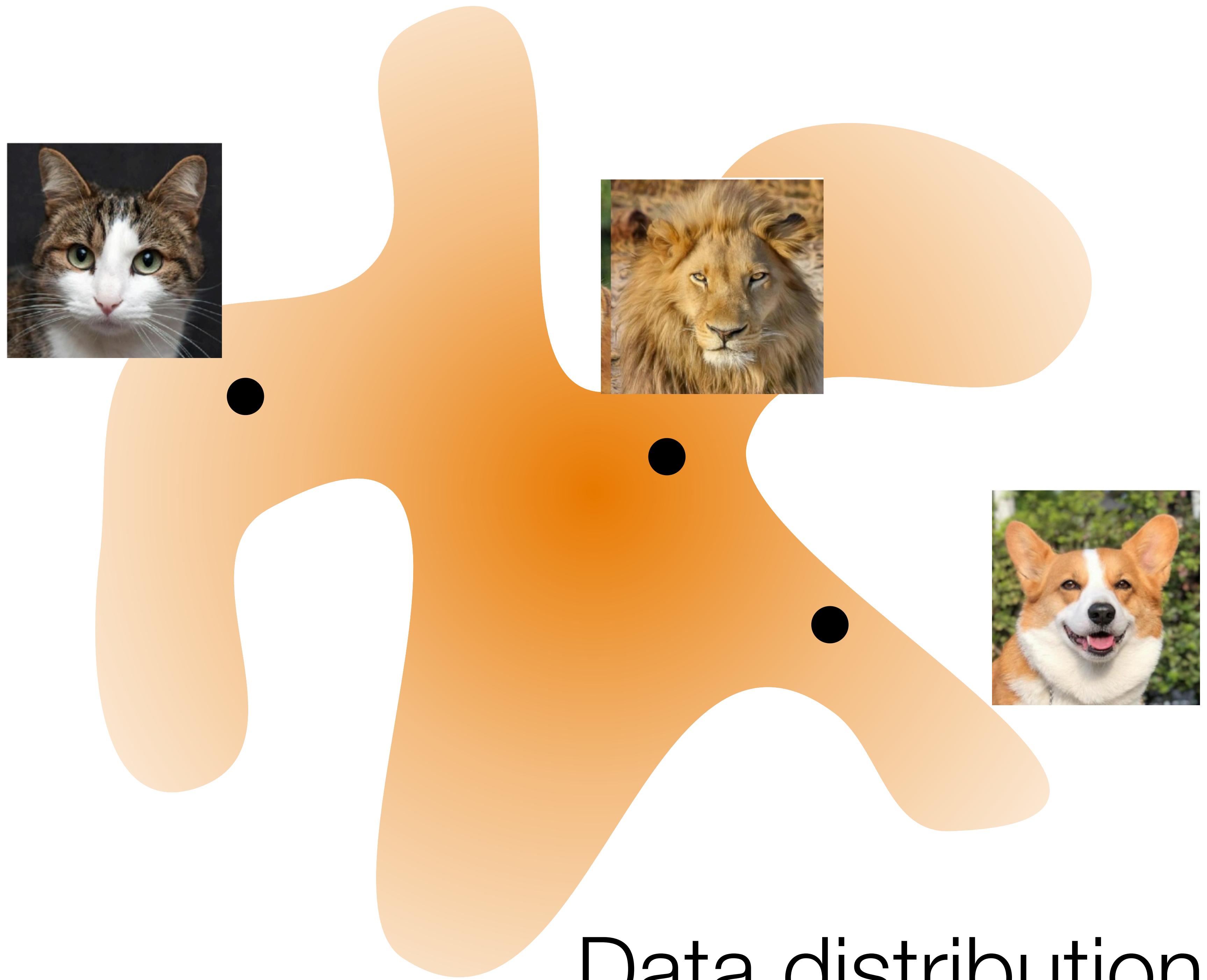
$p(z)$



Latent distribution

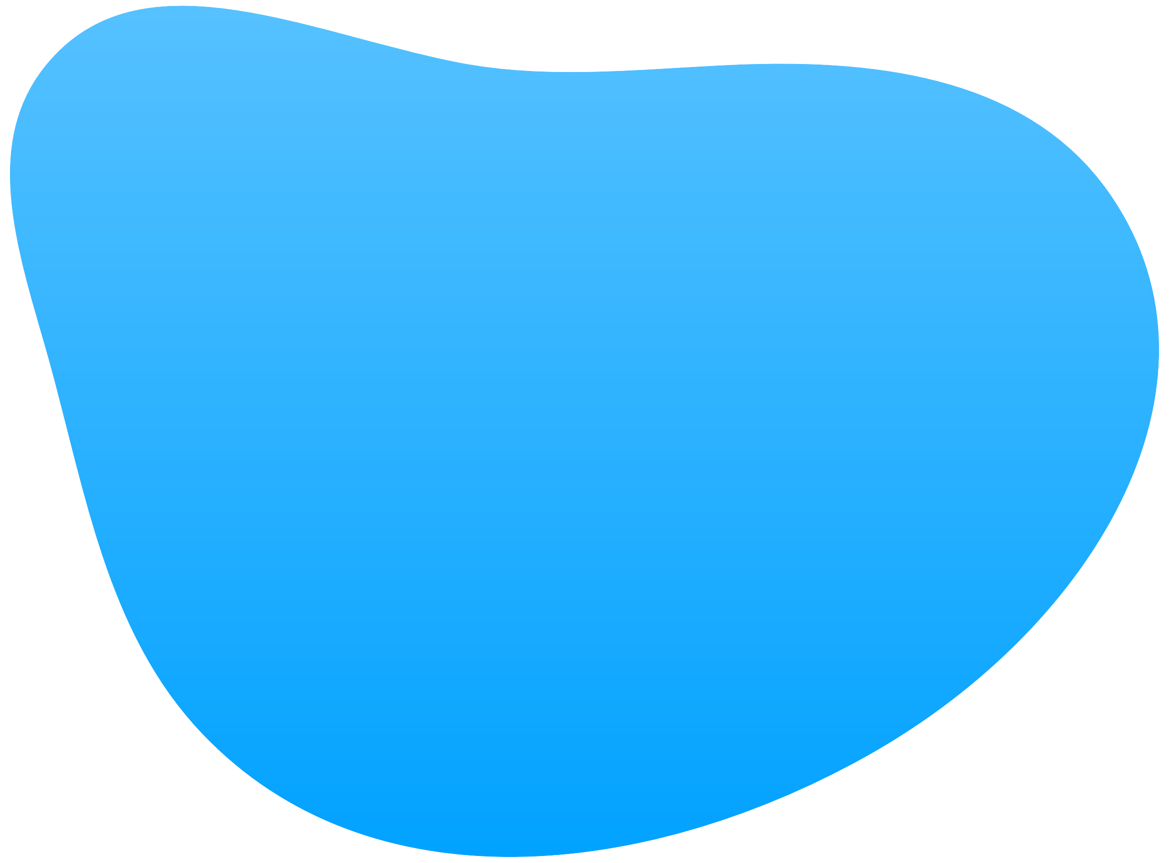
Our goal is to generate samples from an unknown distribution

$p(x)$



Data distribution

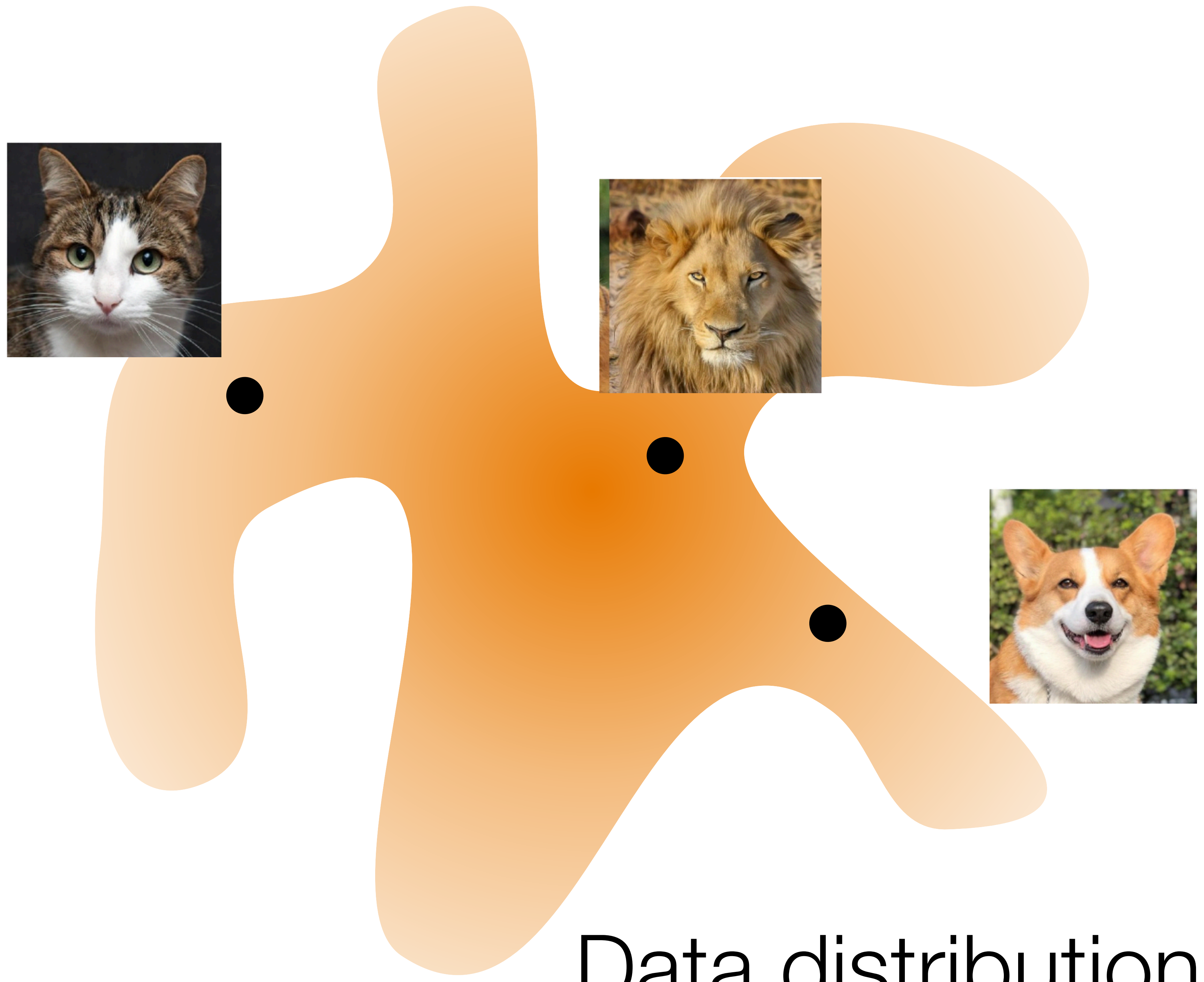
$p(z)$



Latent distribution

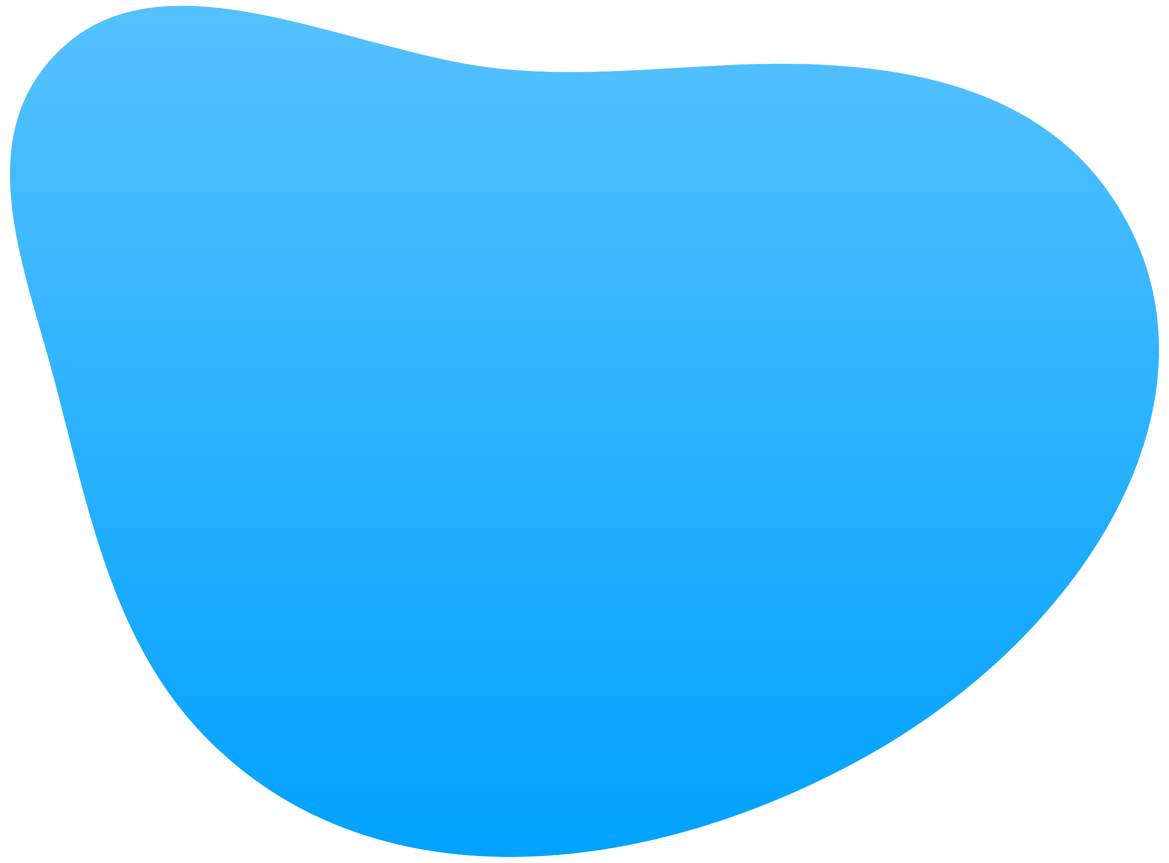
Our goal is to generate samples from an unknown distribution

$p(x)$



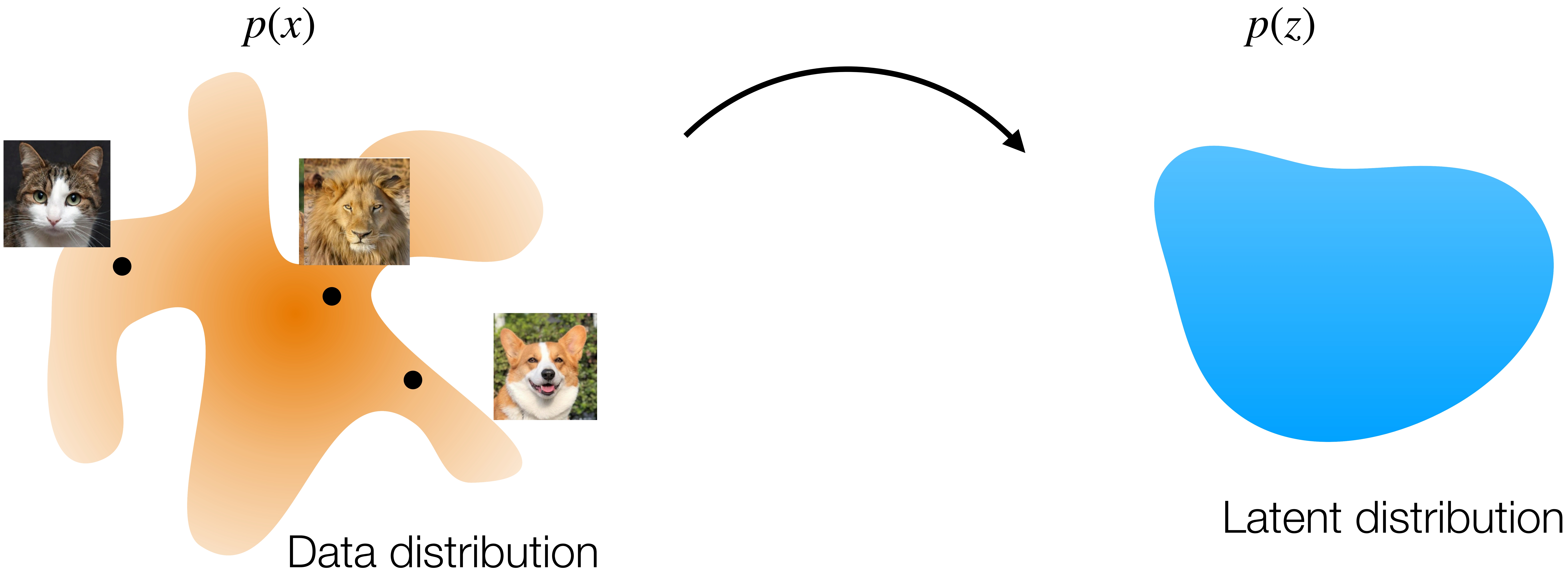
Data distribution

$p(z)$

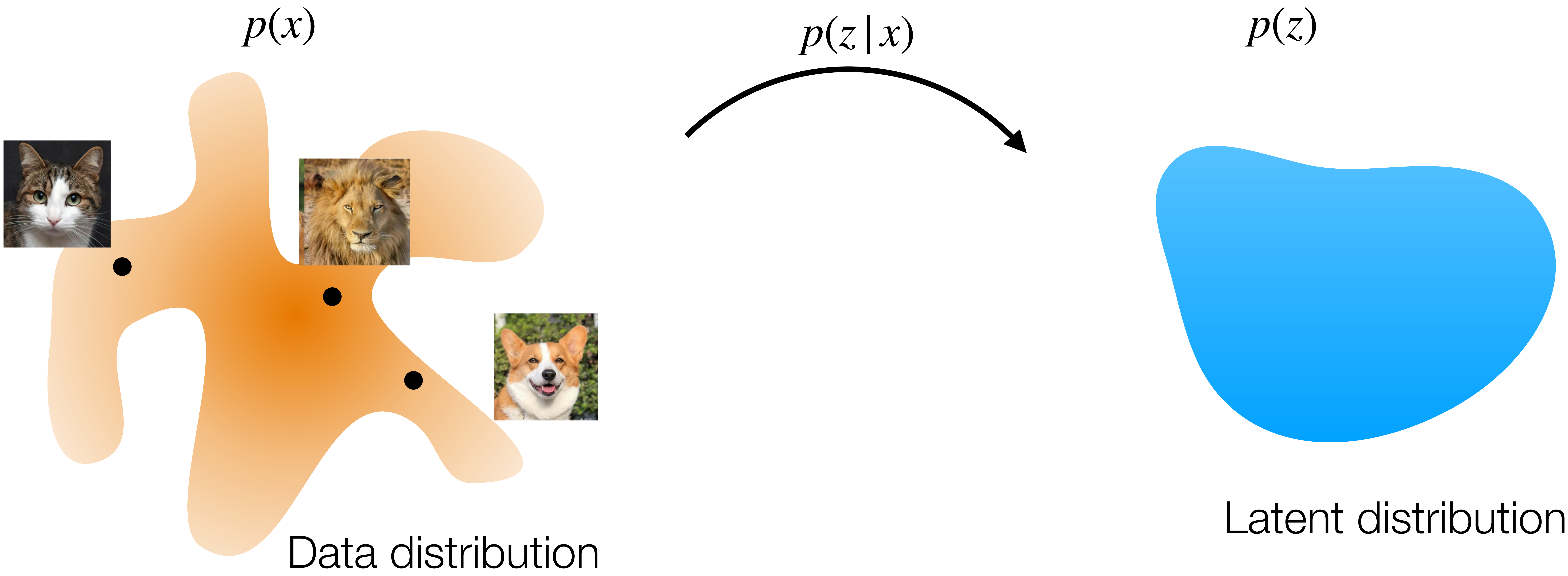


Latent distribution

Our goal is to generate samples from an unknown distribution



Our goal is to generate samples from an unknown distribution



$p(x)$

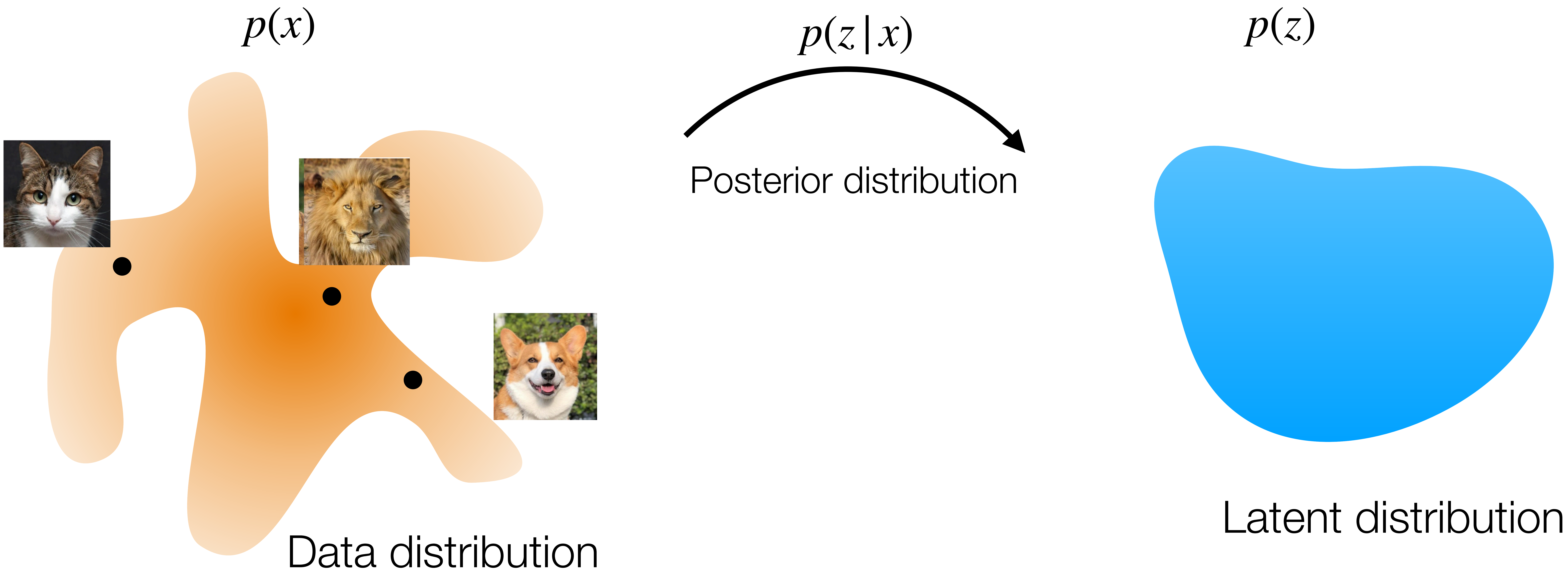
$p(z|x)$

$p(z)$

Data distribution

Latent distribution

Our goal is to generate samples from an unknown distribution



$p(x)$

$p(z \mid x)$

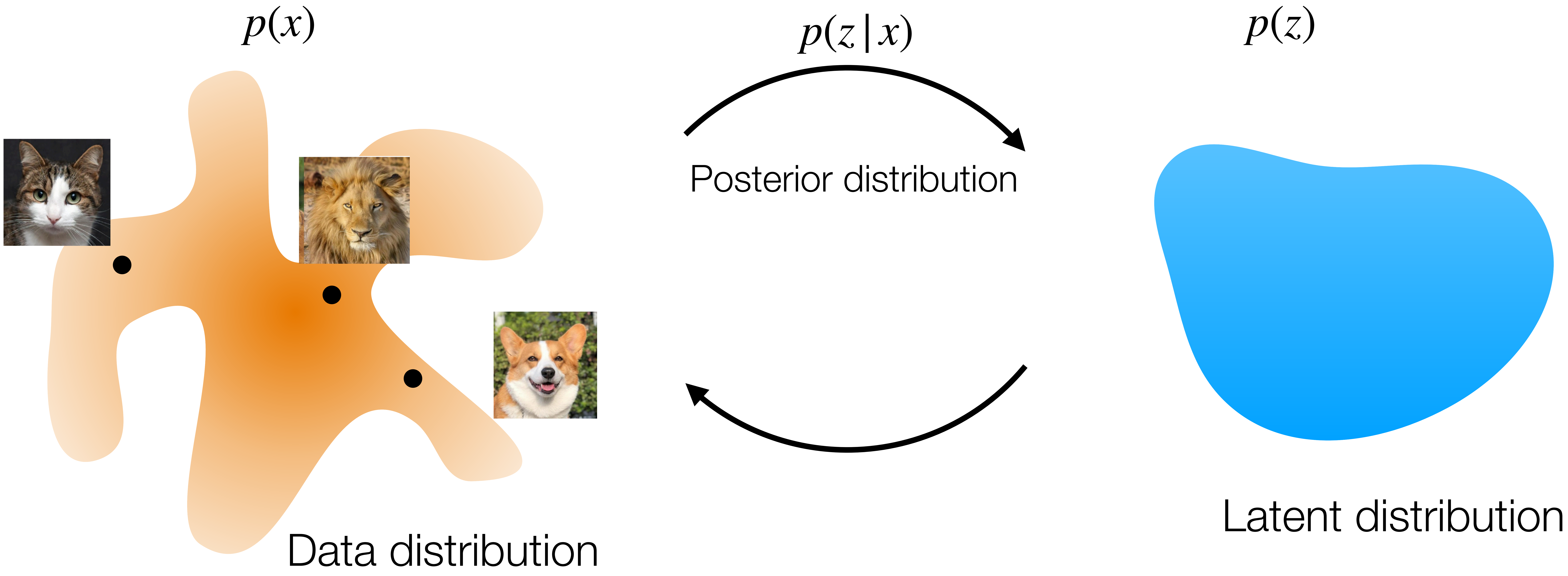
$p(z)$

Posterior distribution

Data distribution

Latent distribution

Our goal is to generate samples from an unknown distribution



$p(x)$

$p(z \mid x)$

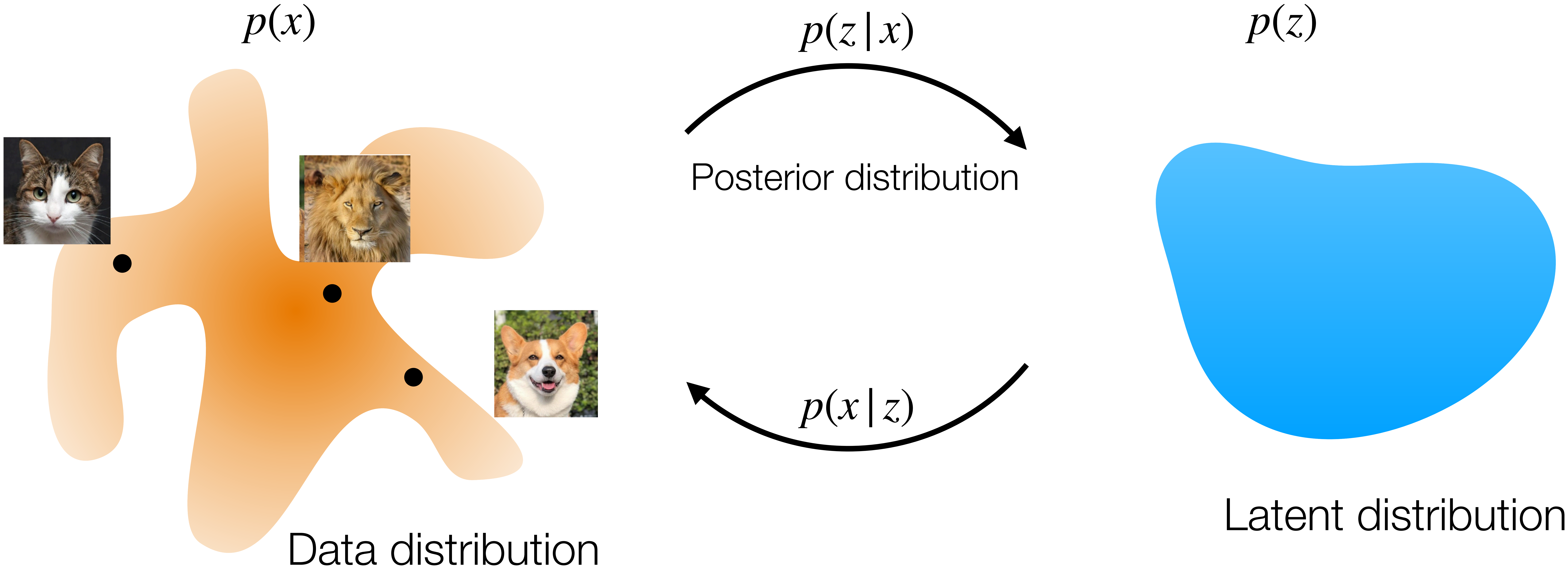
$p(z)$

Posterior distribution

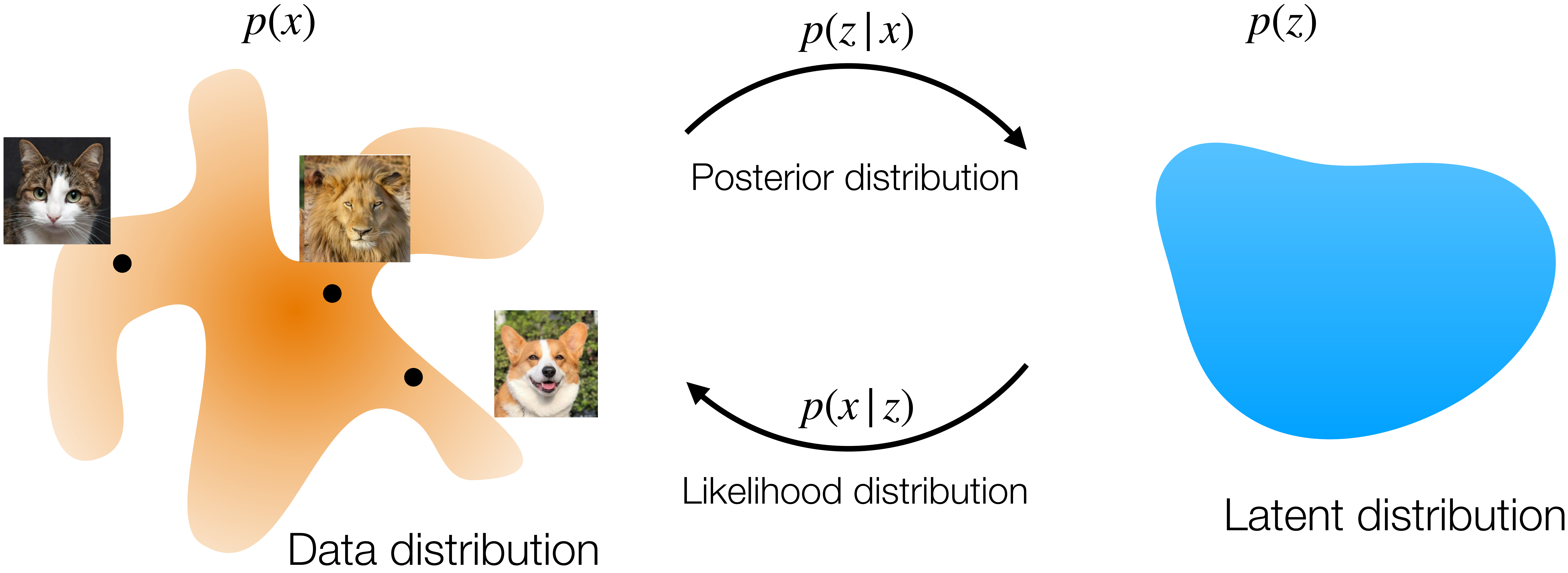
Data distribution

Latent distribution

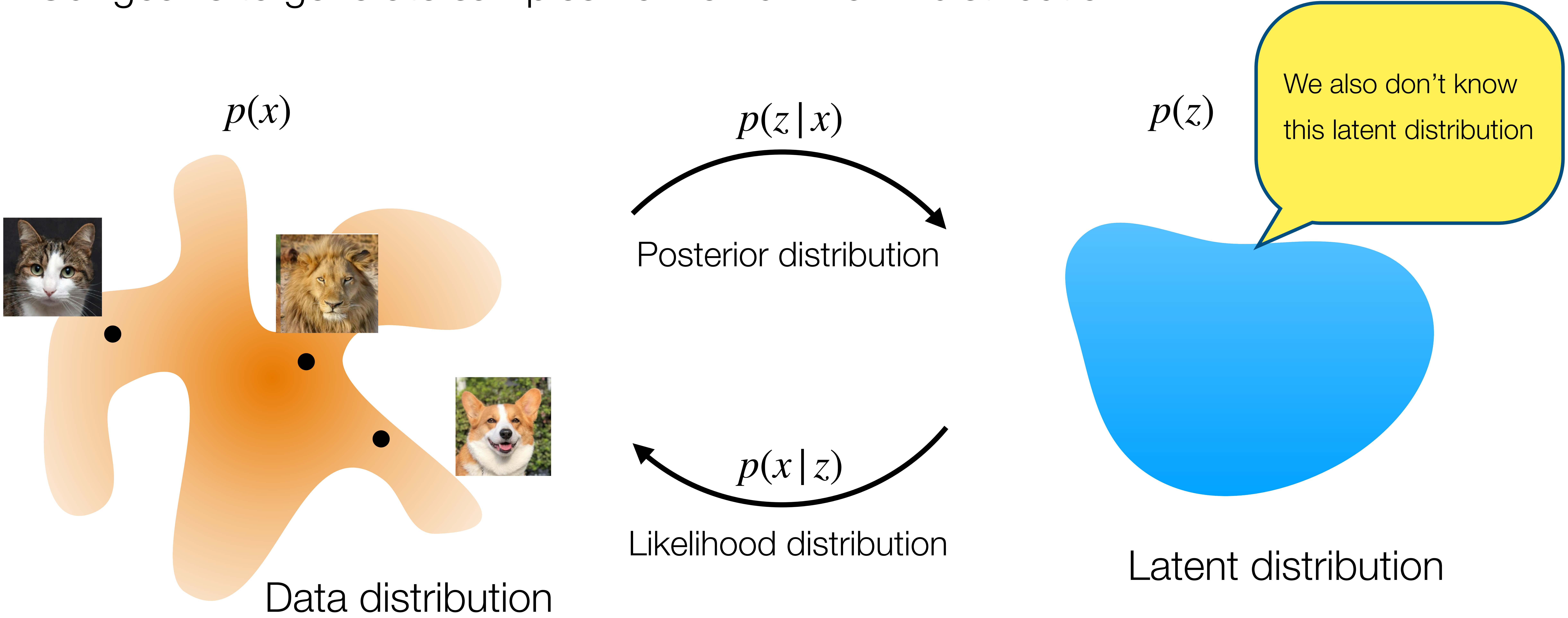
Our goal is to generate samples from an unknown distribution



Our goal is to generate samples from an unknown distribution



Our goal is to generate samples from an unknown distribution



$p(x)$

$p(z \mid x)$

Posterior distribution

$p(z)$

We also don’t know this latent distribution

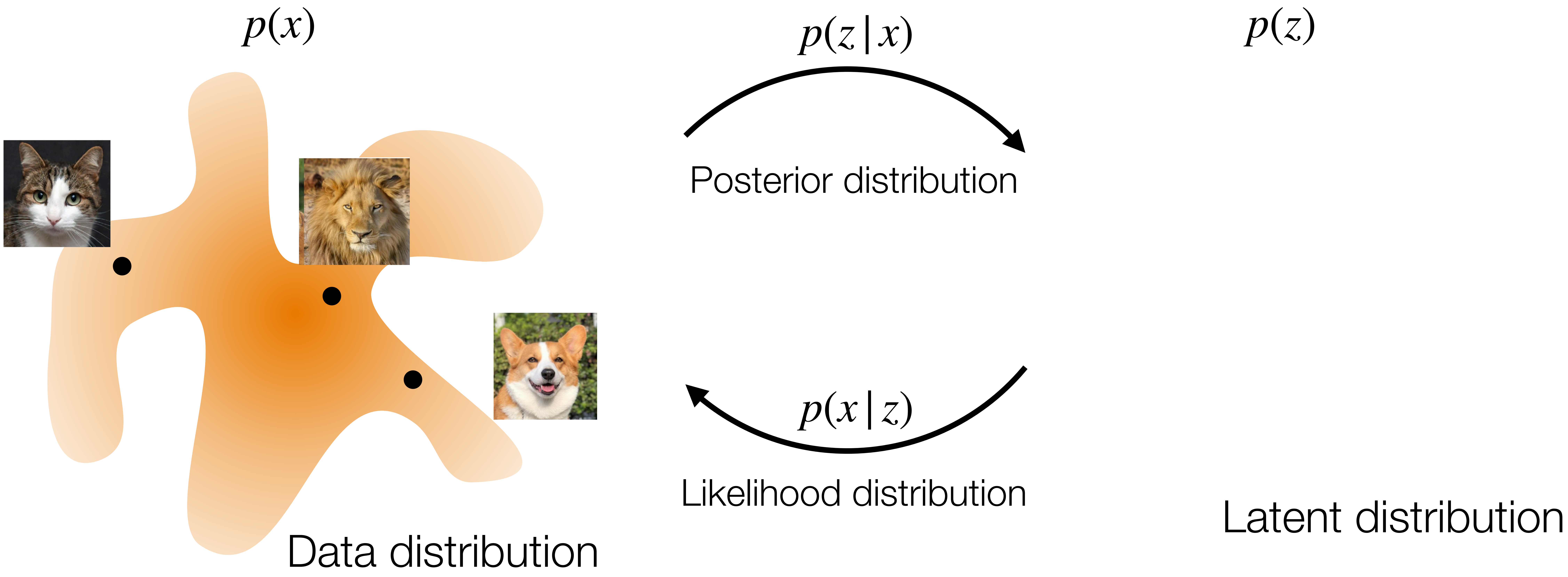
$p(x \mid z)$

Likelihood distribution

Data distribution

Latent distribution

Our goal is to generate samples from an unknown distribution



$p(x)$

Data distribution

$p(z|x)$

Posterior distribution

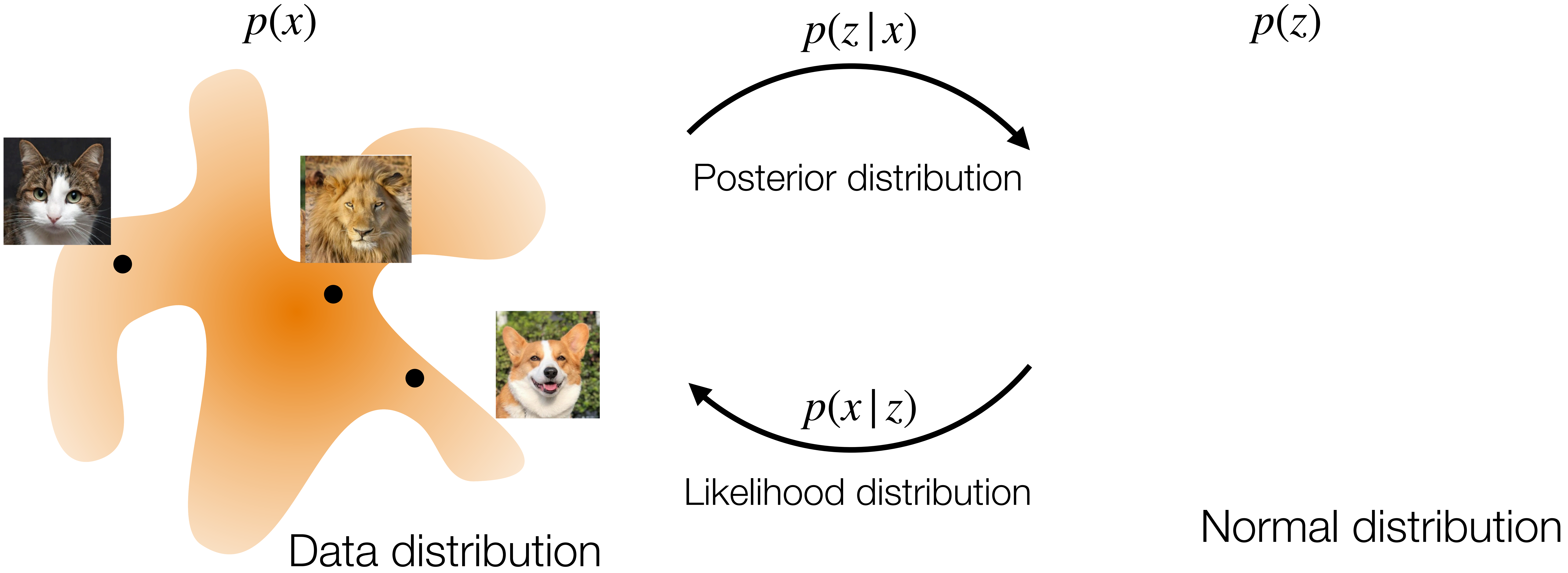
$p(x|z)$

Likelihood distribution

$p(z)$

Latent distribution

Our goal is to generate samples from an unknown distribution



$p(x)$

$p(z|x)$

Posterior distribution

$p(z)$

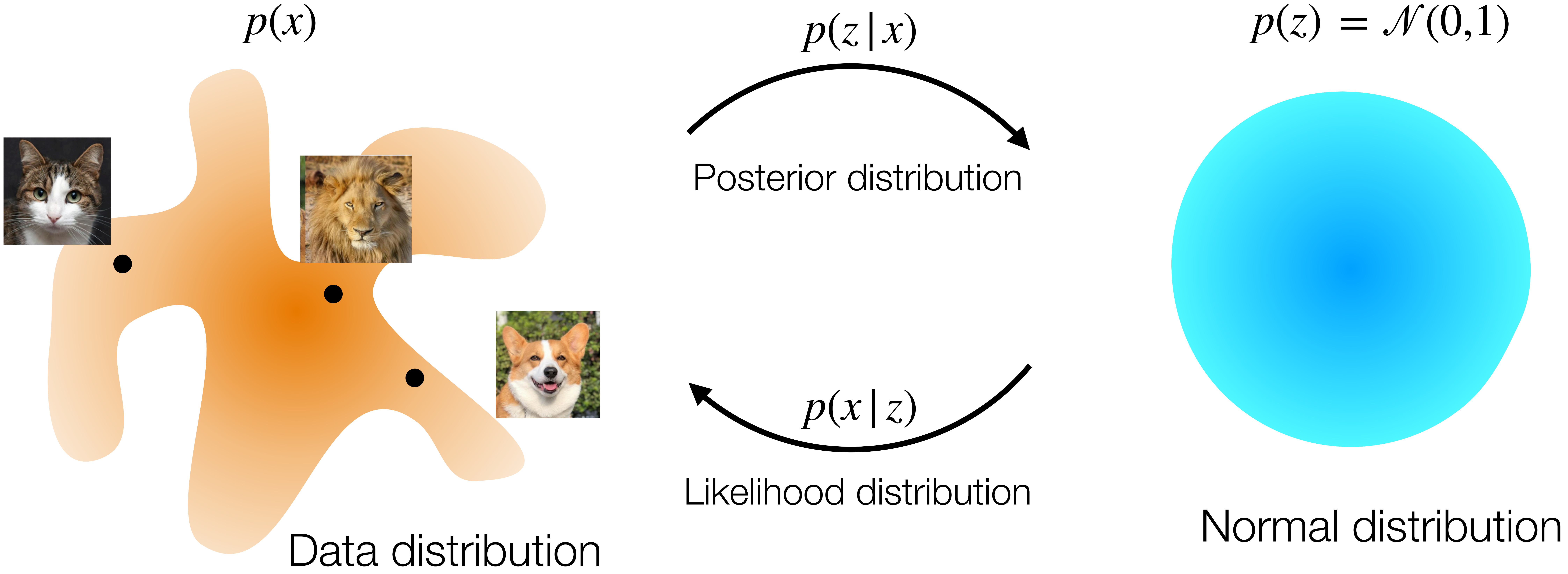
$p(x|z)$

Likelihood distribution

Data distribution

Normal distribution

Our goal is to generate samples from an unknown distribution



$p(x)$

Data distribution

$p(z|x)$

Posterior distribution

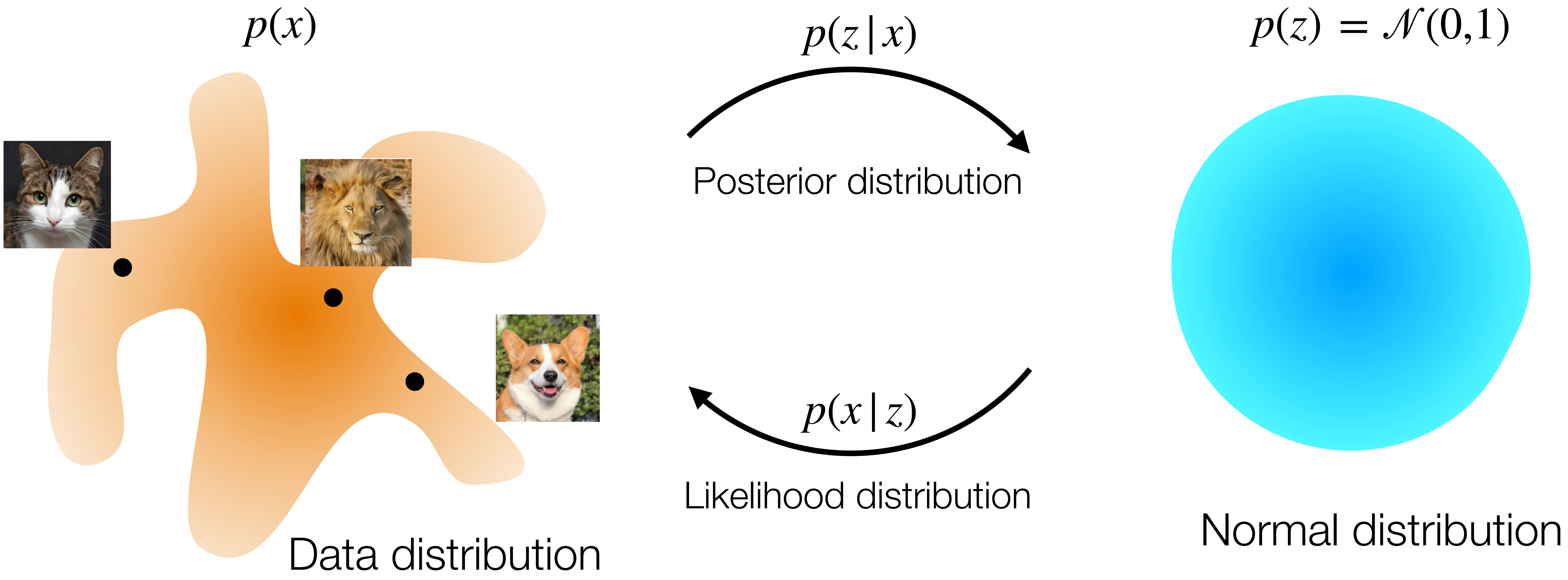
$p(x|z)$

Likelihood distribution

$p(z) = \mathcal{N}(0,1)$

Normal distribution

Our goal is to generate samples from an unknown distribution



$p(x)$

Data distribution

$p(z|x)$

Posterior distribution

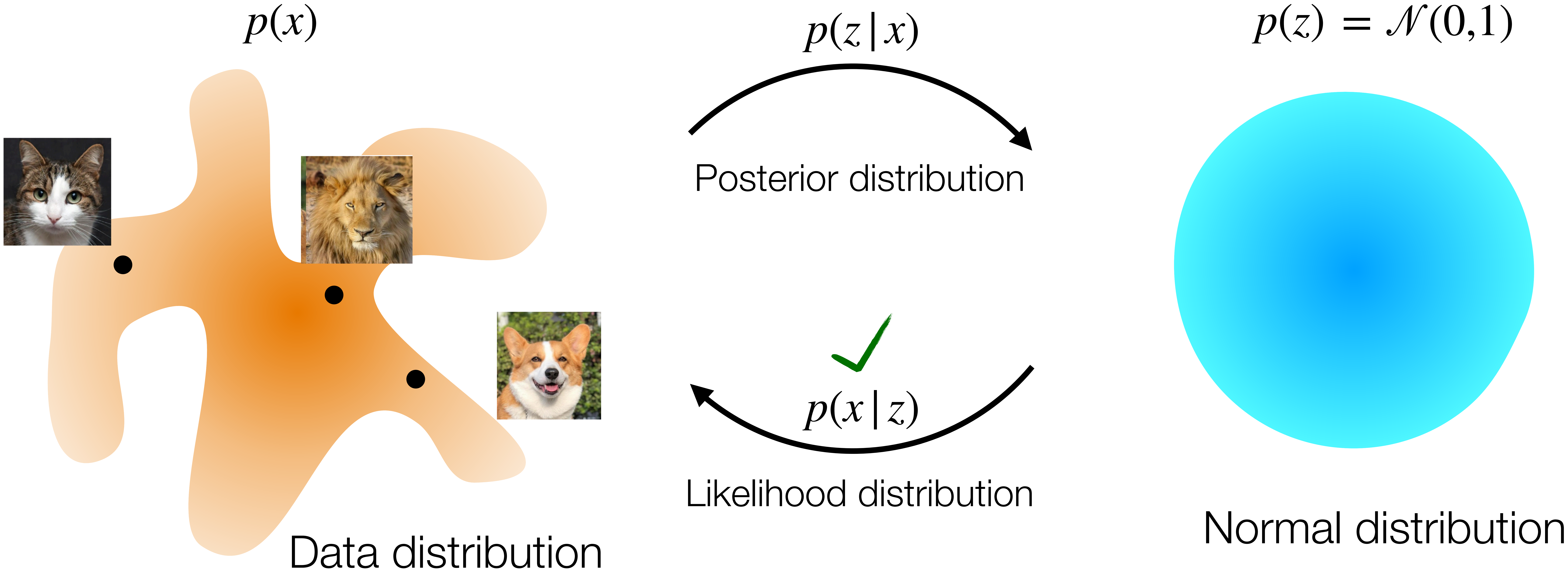
$p(x|z)$

Likelihood distribution

$p(z) = \mathcal{N}(0,1)$

Normal distribution

Our goal is to generate samples from an unknown distribution



$p(x)$

Data distribution

$p(z|x)$

Posterior distribution

✓

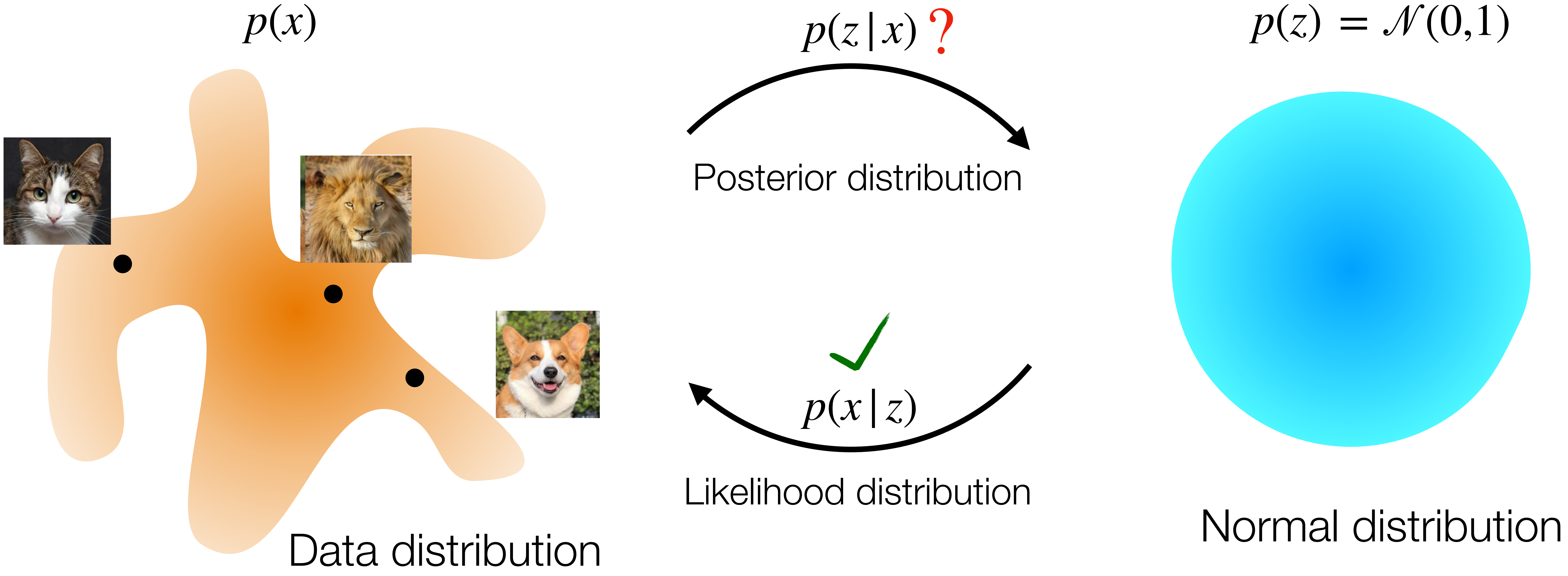
$p(x|z)$

Likelihood distribution

$p(z) = \mathcal{N}(0,1)$

Normal distribution

Our goal is to generate samples from an unknown distribution



$p(x)$

Data distribution

$p(z|x)$?

Posterior distribution

✓

$p(x|z)$

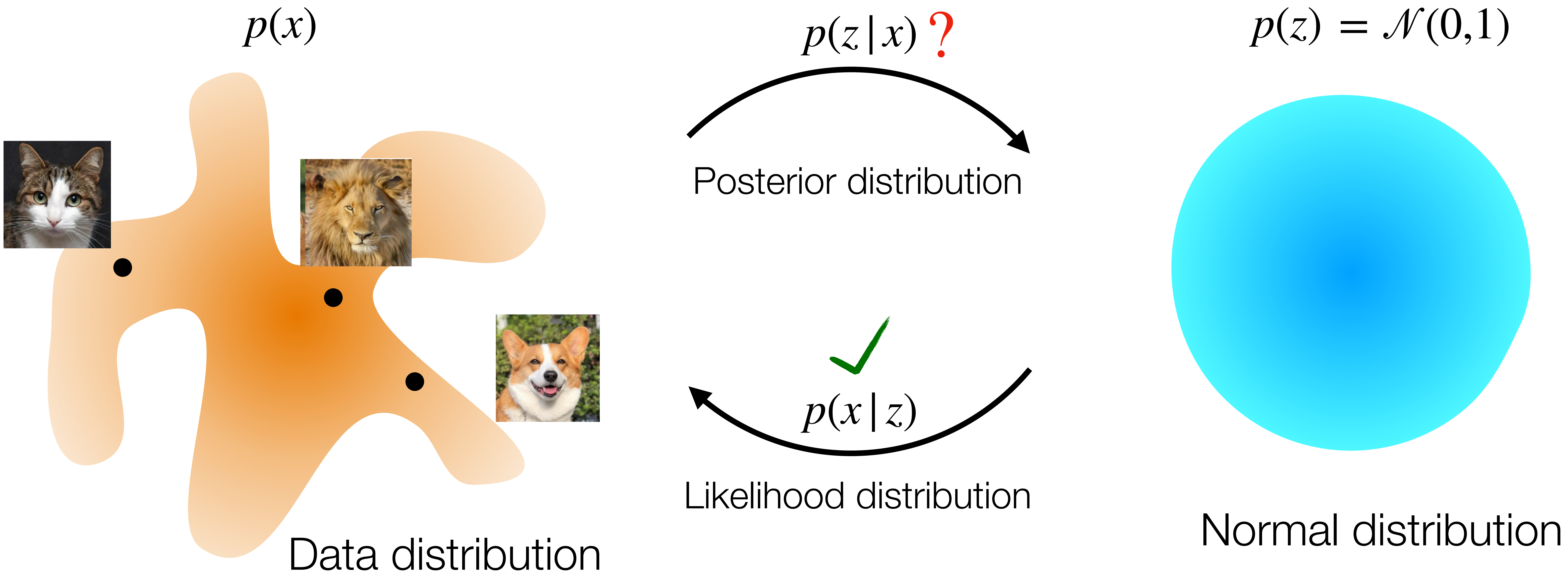
Likelihood distribution

$p(z) = \mathcal{N}(0,1)$

Normal distribution

Variational Bayes

Our goal is to generate samples from an unknown distribution



$p(x)$

$p(z \mid x)$?

Posterior distribution

$p(z) = \mathcal{N}(0,1)$

✓

$p(x \mid z)$

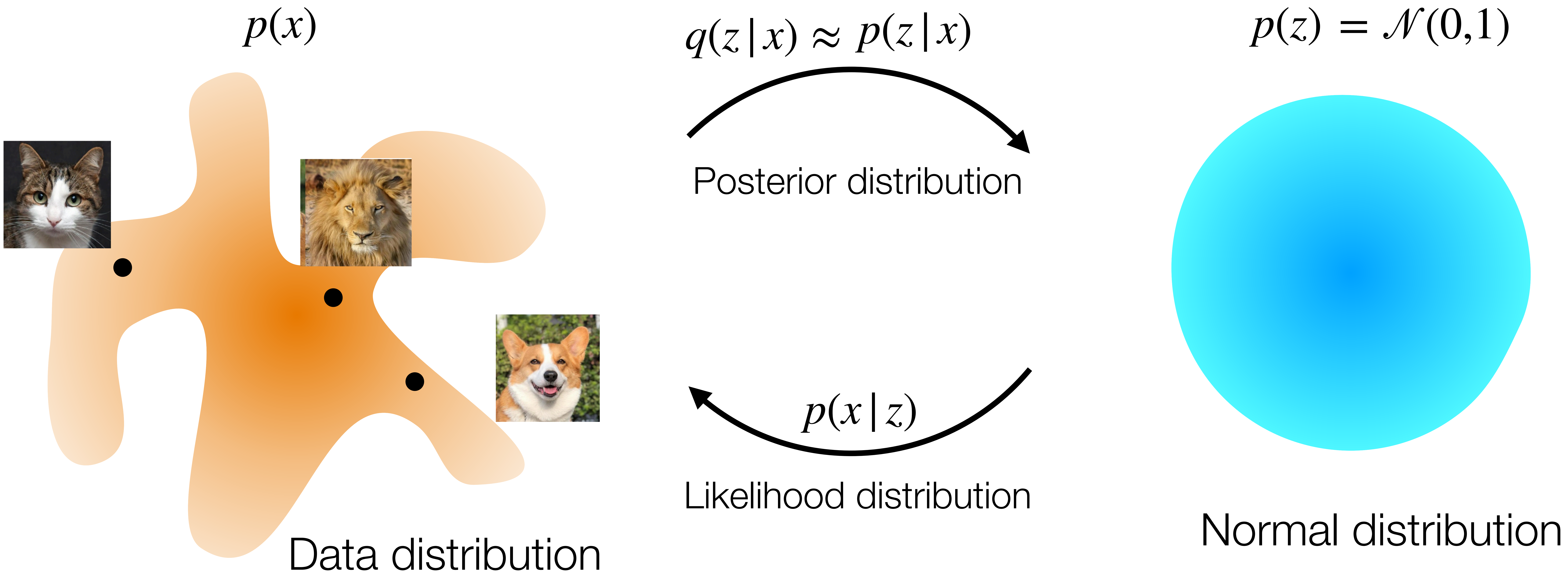
Likelihood distribution

Data distribution

Normal distribution

Variational Bayes

Our goal is to generate samples from an unknown distribution



$p(x)$

Data distribution

$q(z|x) \approx p(z|x)$

Posterior distribution

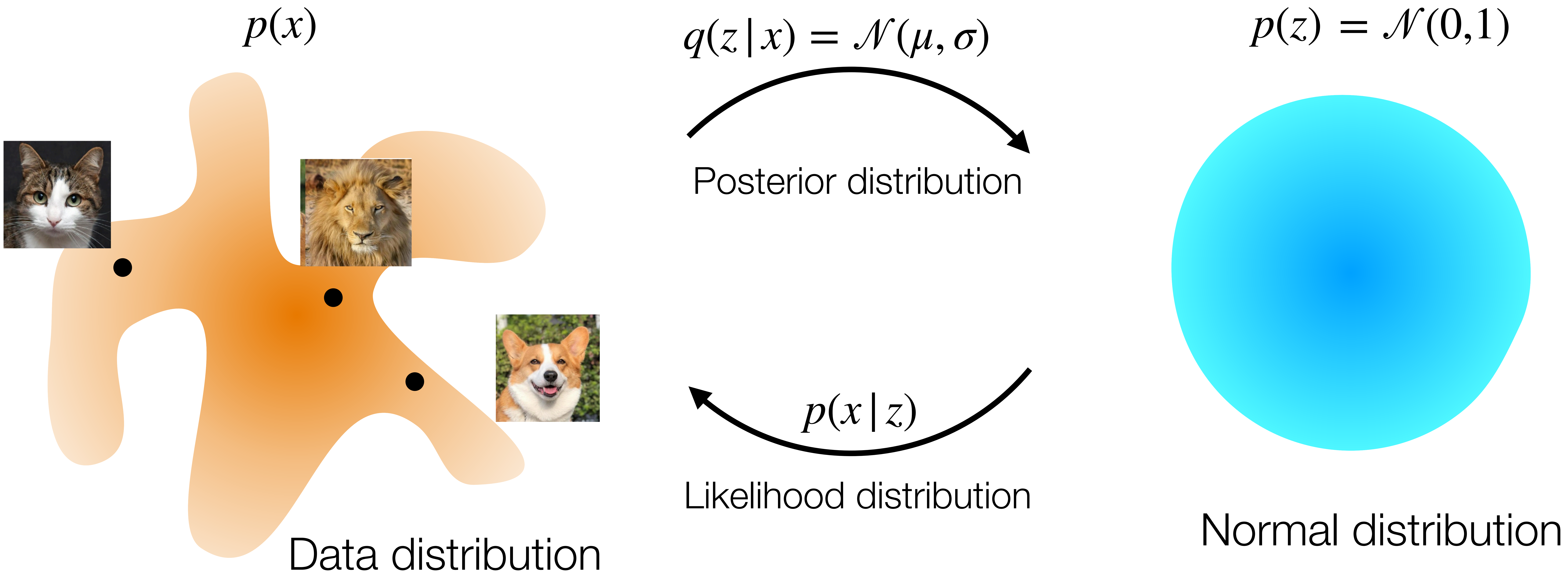
$p(x|z)$

Likelihood distribution

$p(z) = \mathcal{N}(0,1)$

Normal distribution

Variational Bayes



$p(x)$

$q(z\,|\,x) = \mathcal{N}(\mu, \sigma)$

Posterior distribution

$p(z) = \mathcal{N}(0,1)$

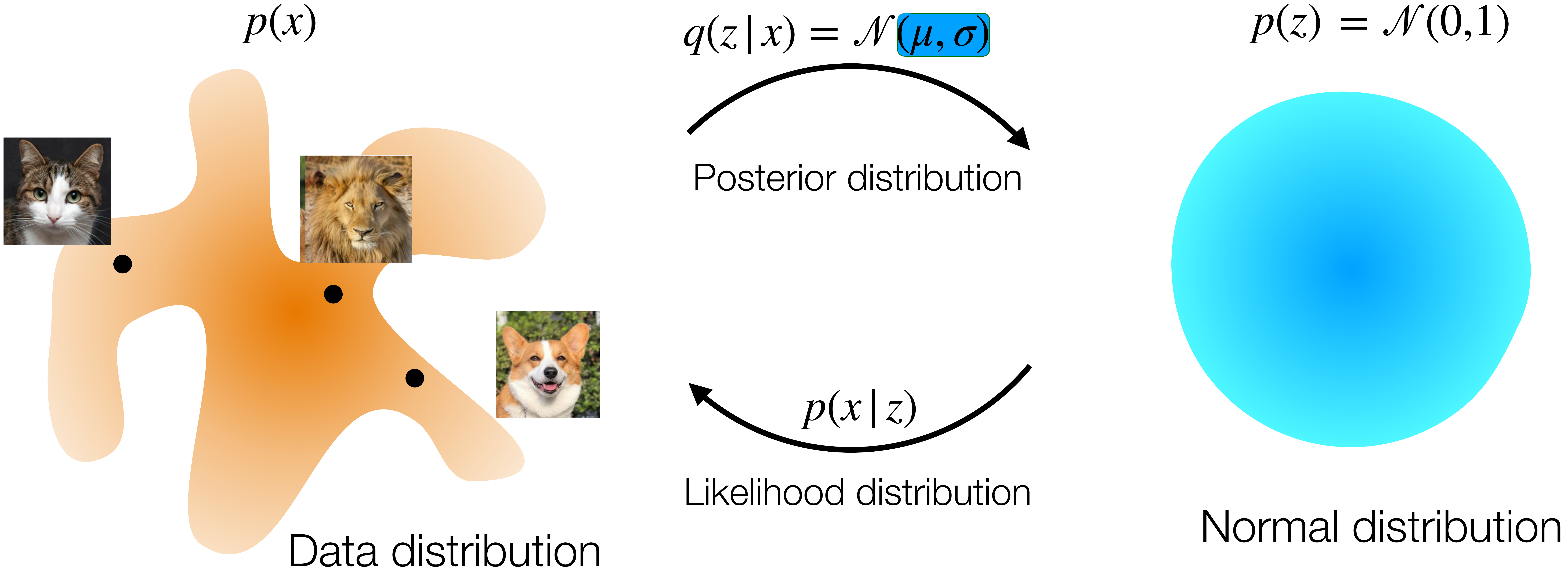
$p(x\,|\,z)$

Likelihood distribution

Data distribution

Normal distribution

Variational Bayes



$p(x)$

$q(z|x) = \mathcal{N}(\mu, \sigma)$

Posterior distribution

$p(x|z)$

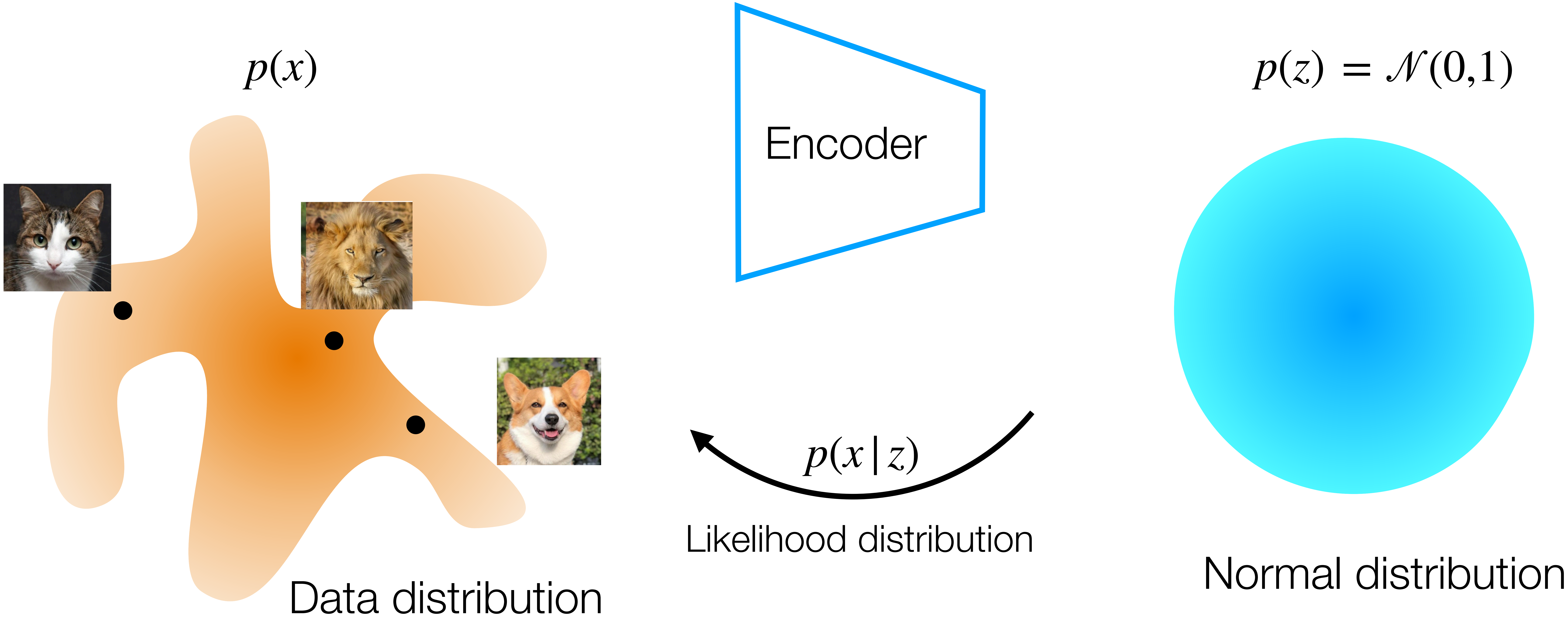
Likelihood distribution

$p(z) = \mathcal{N}(0,1)$

Data distribution

Normal distribution

Autoencoder Variational Bayes



$p(x)$

Encoder

$p(z) = \mathcal{N}(0,1)$

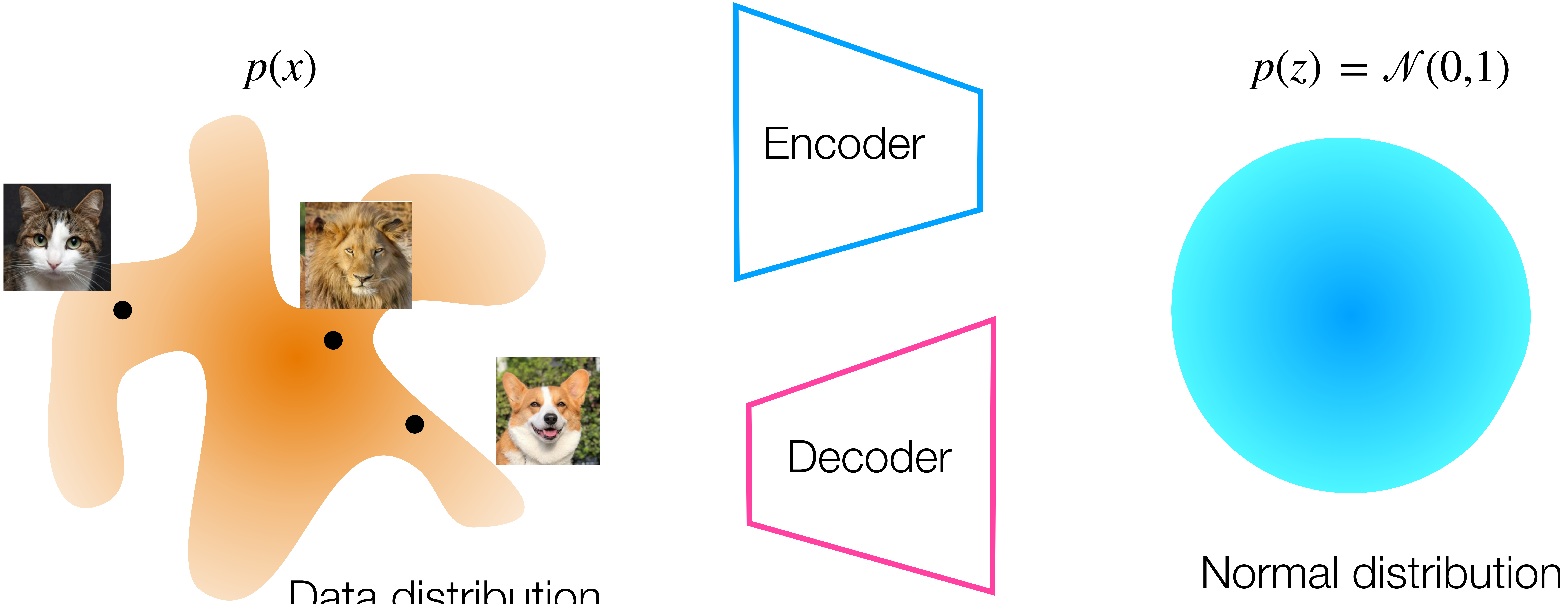
$p(x|z)$

Likelihood distribution

Data distribution

Normal distribution

Autoencoder Variational Bayes



$p(x)$

Encoder

Decoder

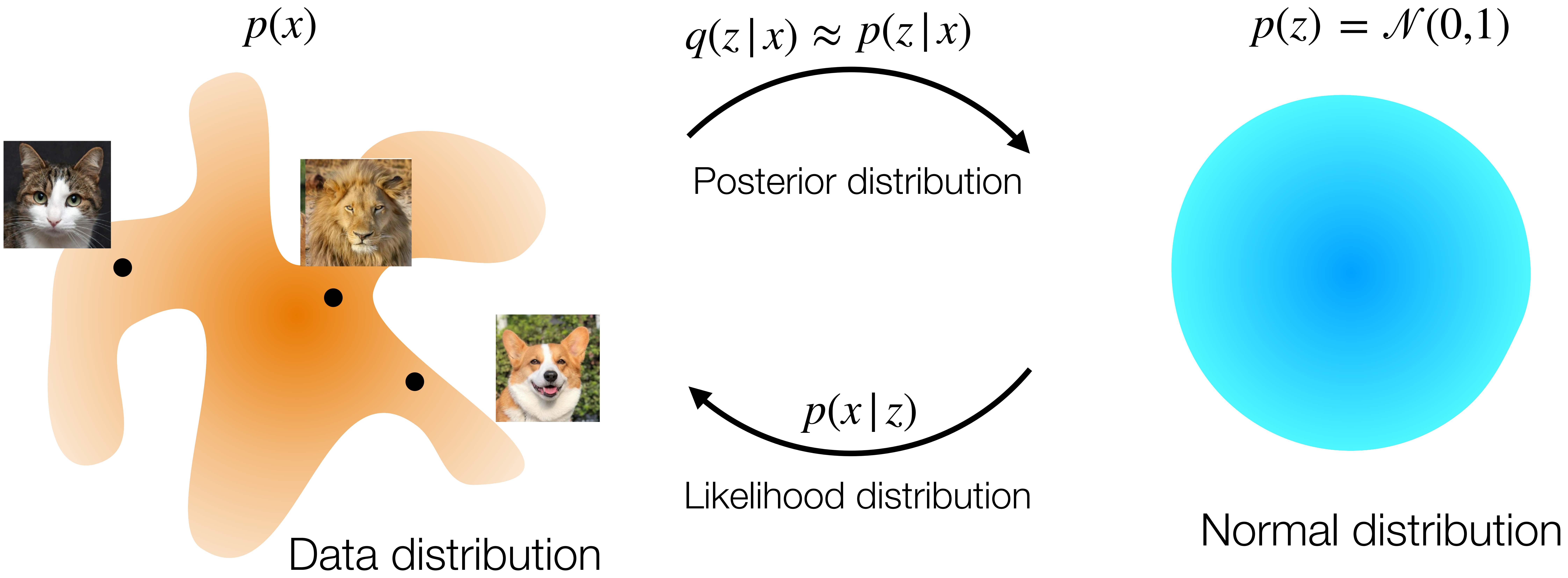
$p(z) = \mathcal{N}(0,1)$

Data distribution

Normal distribution

How do we train this auto encoder?

Our goal is to generate samples from an unknown distribution



$p(x)$

Data distribution

$q(z|x) \approx p(z|x)$

Posterior distribution

$p(x|z)$

Likelihood distribution

$p(z) = \mathcal{N}(0,1)$

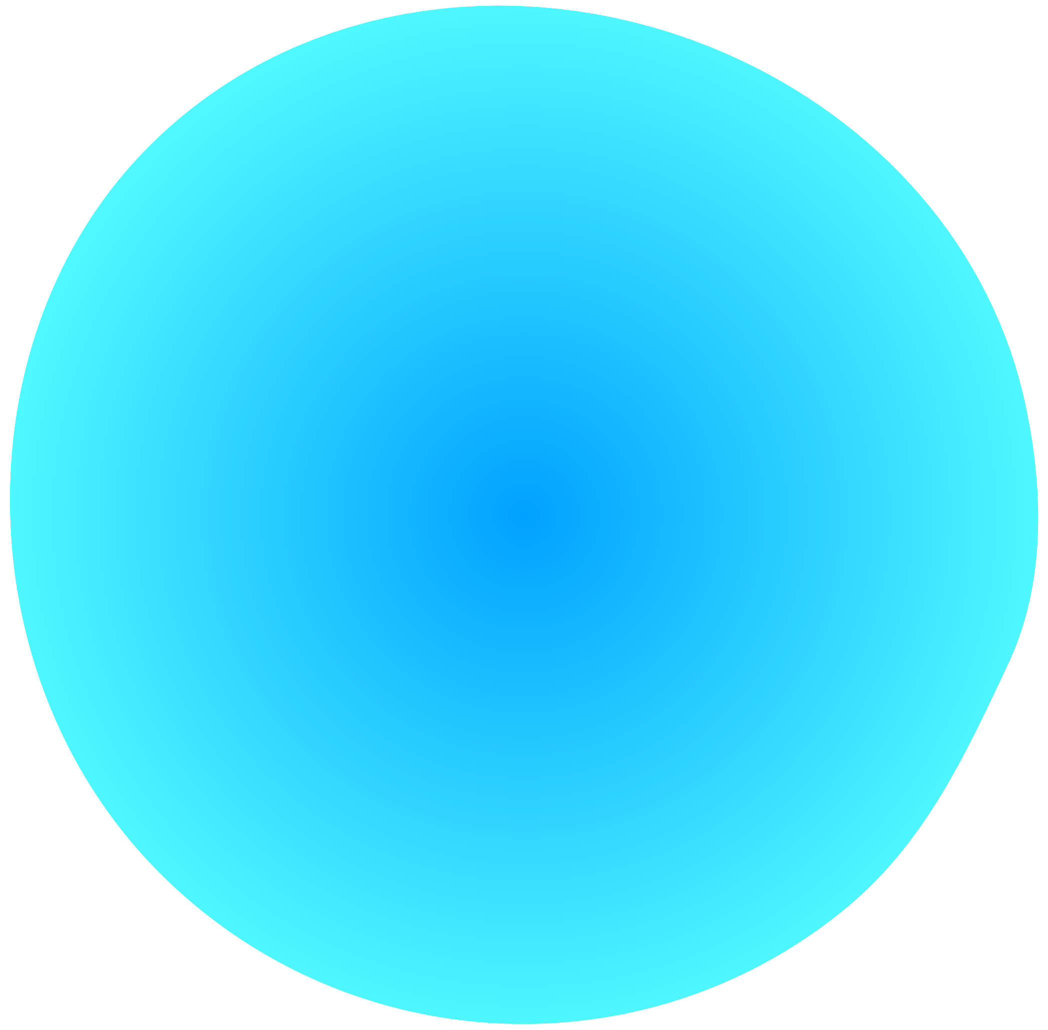
Normal distribution

Evidence lower bound (ELBO)

Likelihood distribution

Posterior distribution

$p(x \mid z)$ $q(z \mid x)$ $p(z)$

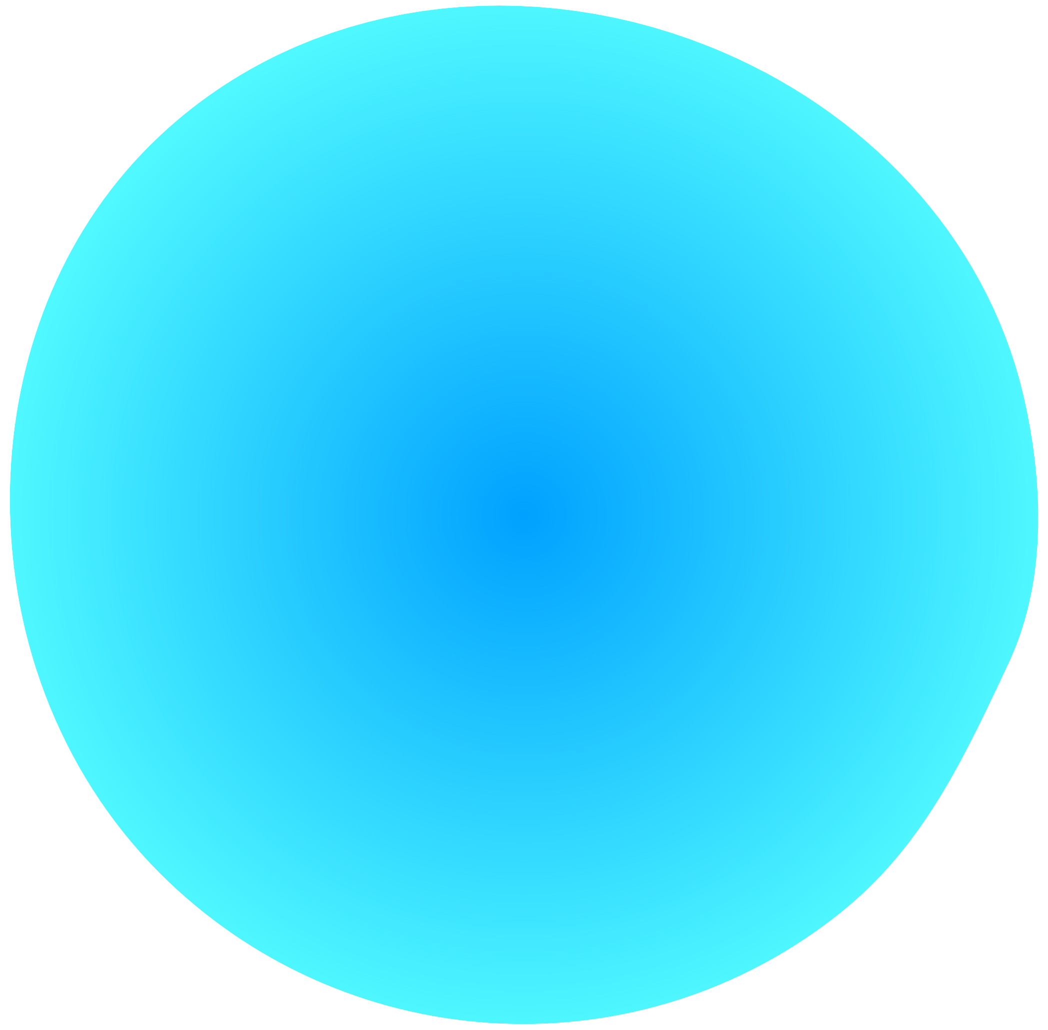


Evidence lower bound (ELBO)

Likelihood distribution

Posterior distribution

$$\mathcal{L}(x) = \mathbb{E}_{q(z|x)}[\log p(x\,|\,z)] - KL(q(z\,|\,x)\,|\,p(z))$$

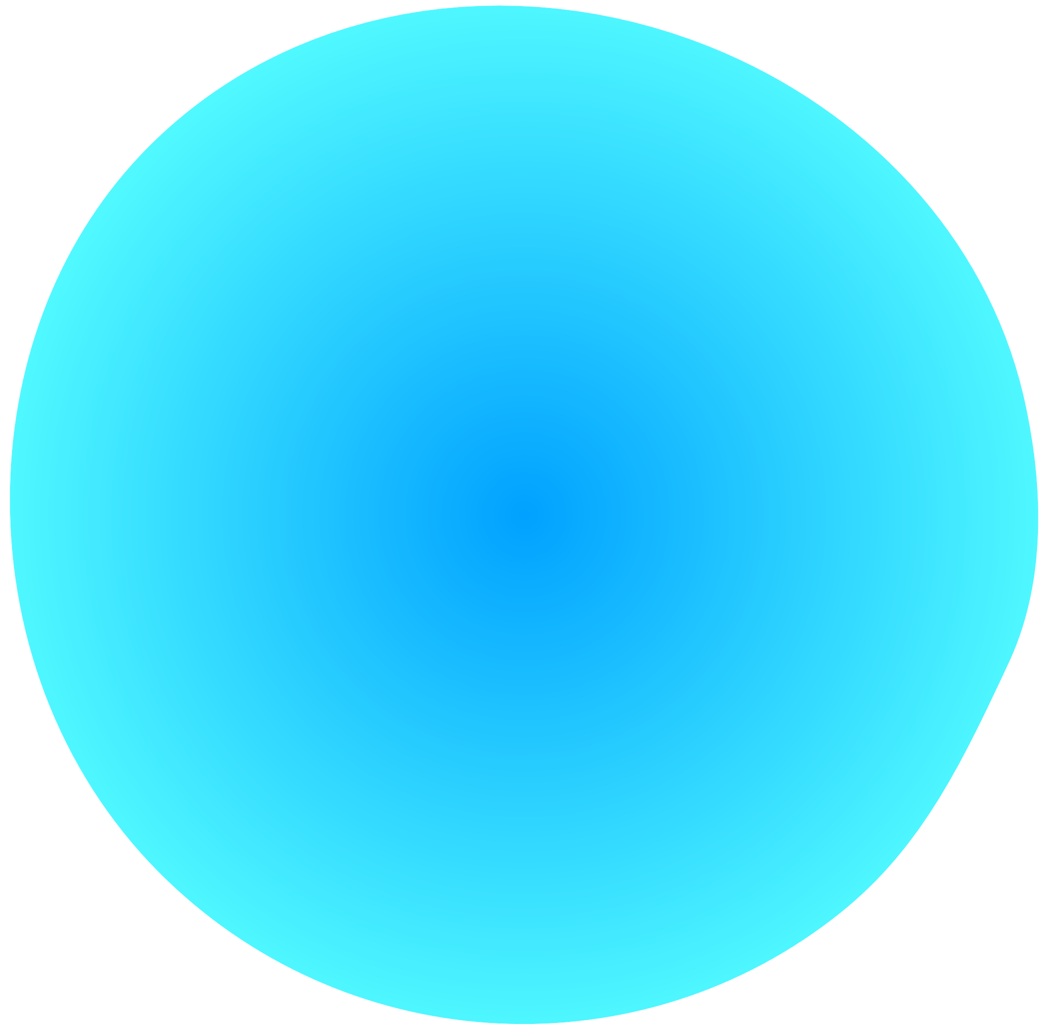


Evidence lower bound (ELBO)

Likelihood distribution

Posterior distribution

$$\mathcal{L}(x) = \underbrace{\mathbb{E}_{q(z|x)}[\log p(x\,|\,z)]}_{\text{Data consistency}} - KL(q(z\,|\,x)\,|\,p(z))$$

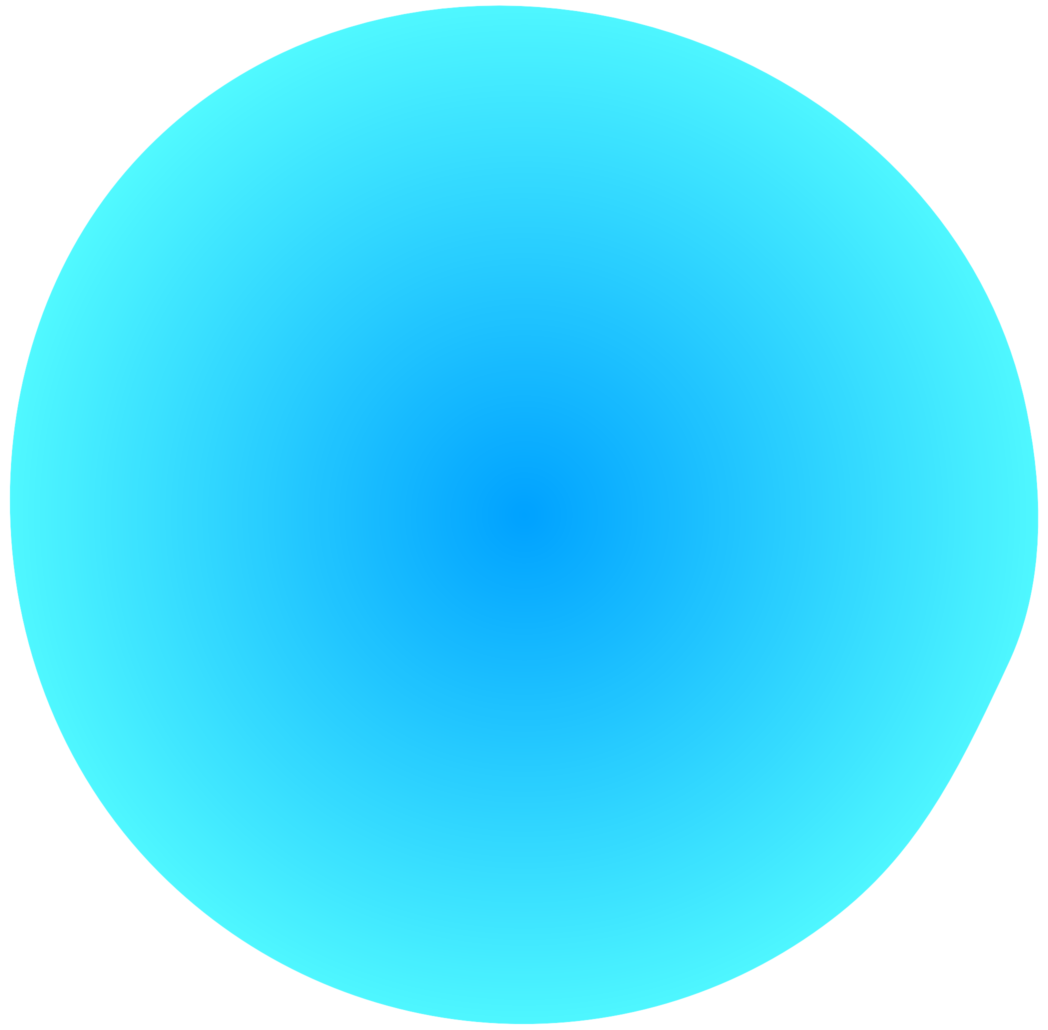


Evidence lower bound (ELBO)

Likelihood distribution

Posterior distribution

$$\mathscr{L}(x) = \underbrace{\mathbb{E}_{q(z|x)}[\log p(x\,|\,z)]}_{\text{Data consistency}} - \underbrace{KL(q(z\,|\,x)\,|\,p(z))}_{\text{Regularization}}$$

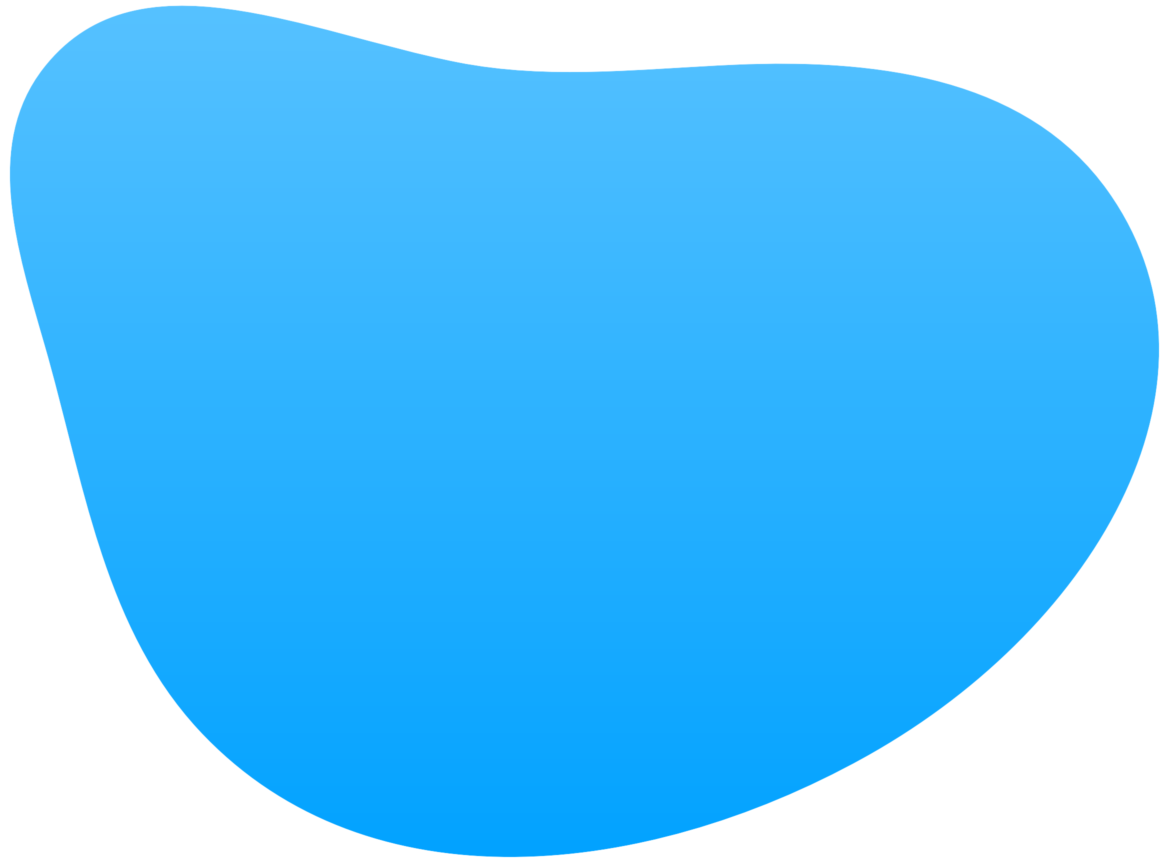


ELBO: Likelihood as an L_2 term

$$\mathscr{L}(x) = \mathbb{E}_{q(z|x)}[\log p(x \,|\, z)] - KL(q(z \,|\, x) \,|\, p(z))$$

Data consistency

Regularization

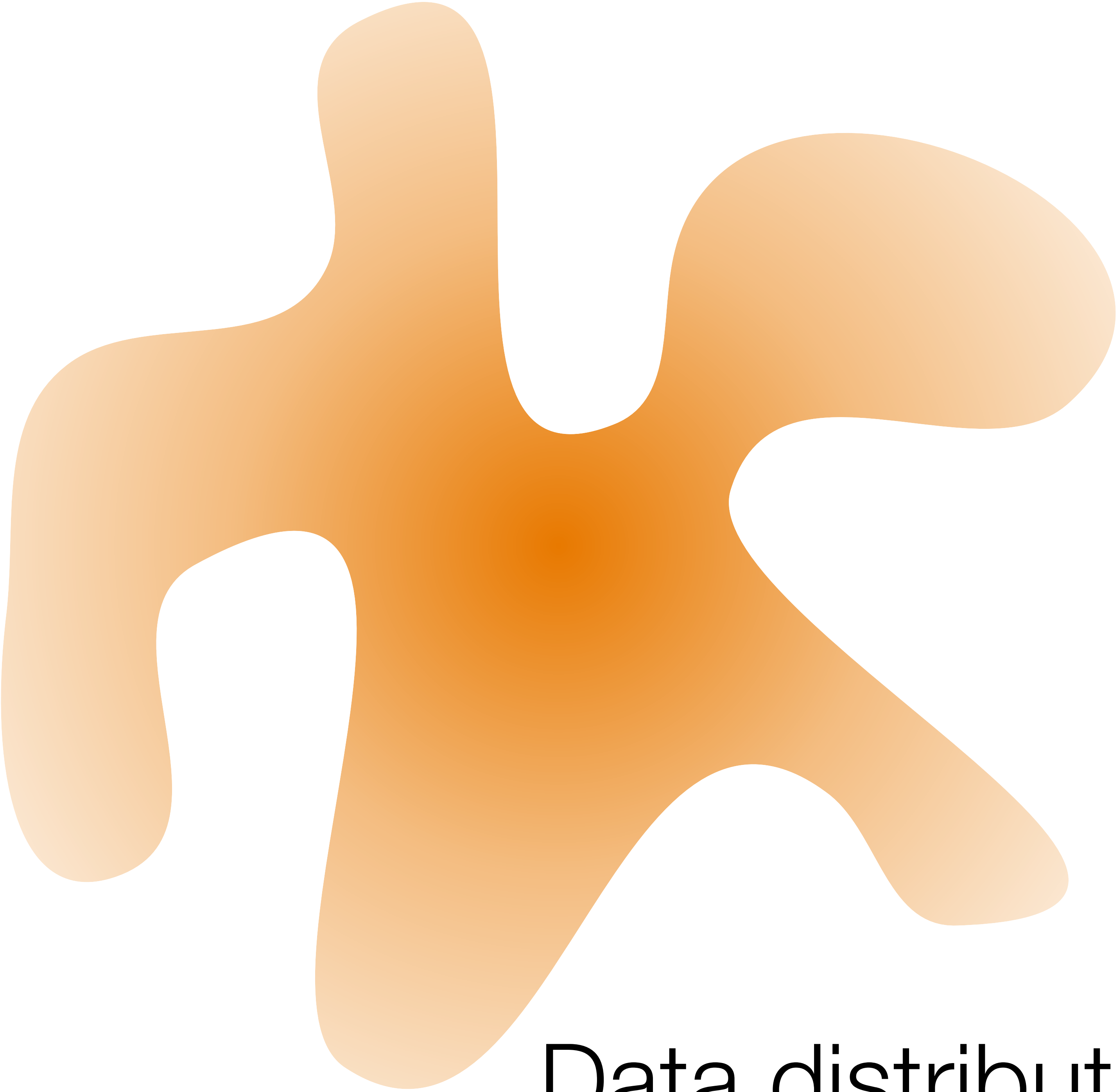


Data distribution

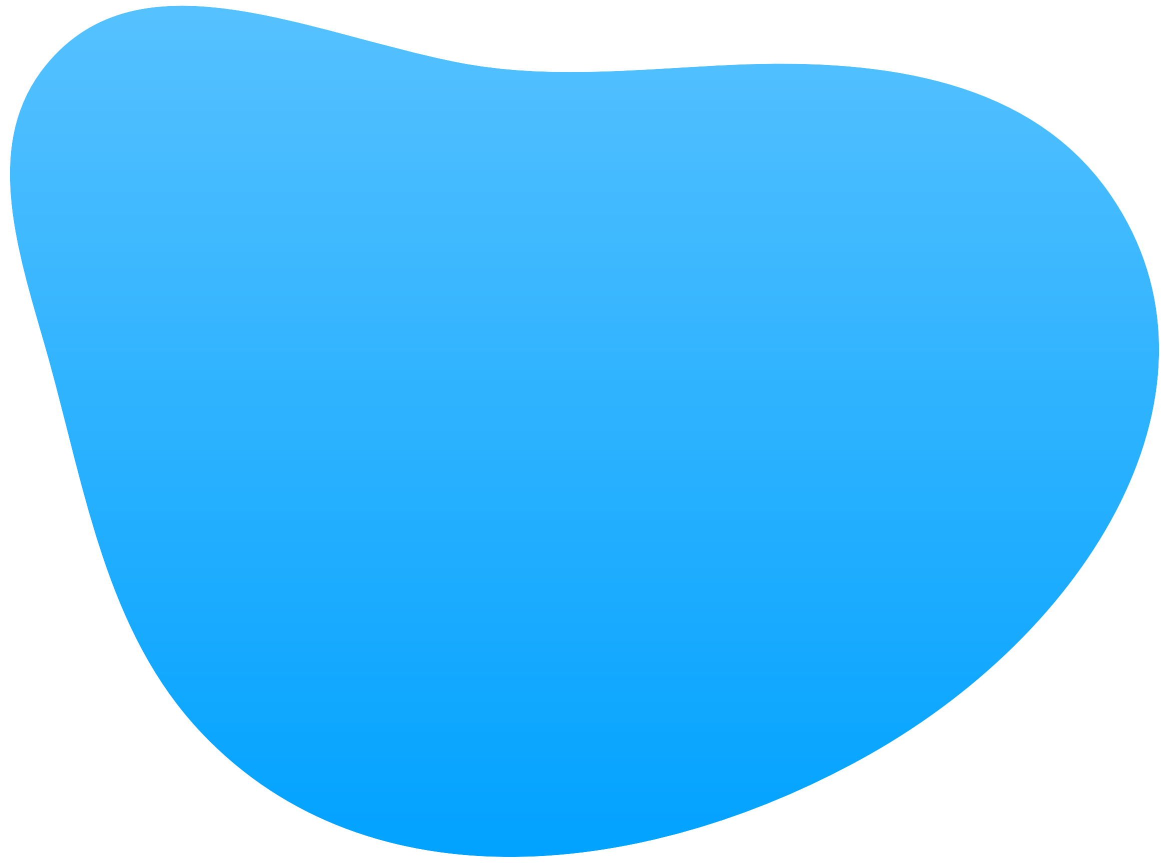
ELBO: Likelihood as an L_2 term

$$\mathscr{L}(x) = \mathbb{E}_{q(z|x)}[\log p(x\,|\,z)] - KL(q(z\,|\,x)\,|\,p(z))$$

Data consistency Regularization

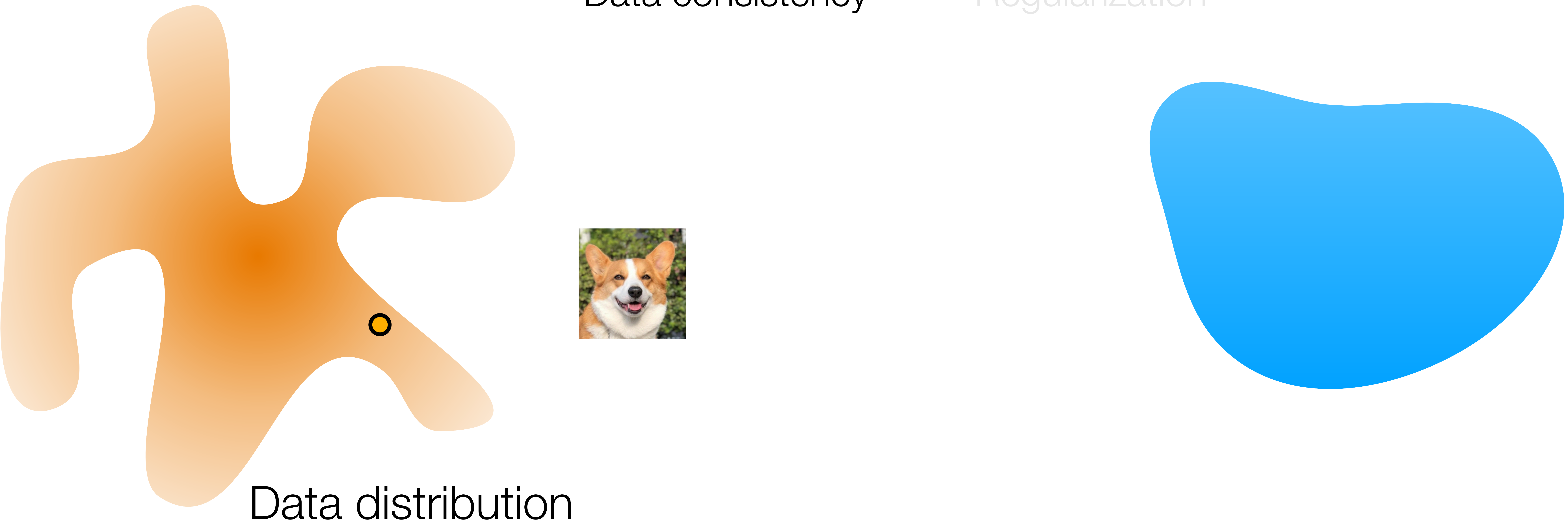


Data distribution



ELBO: Likelihood as an L_2 term

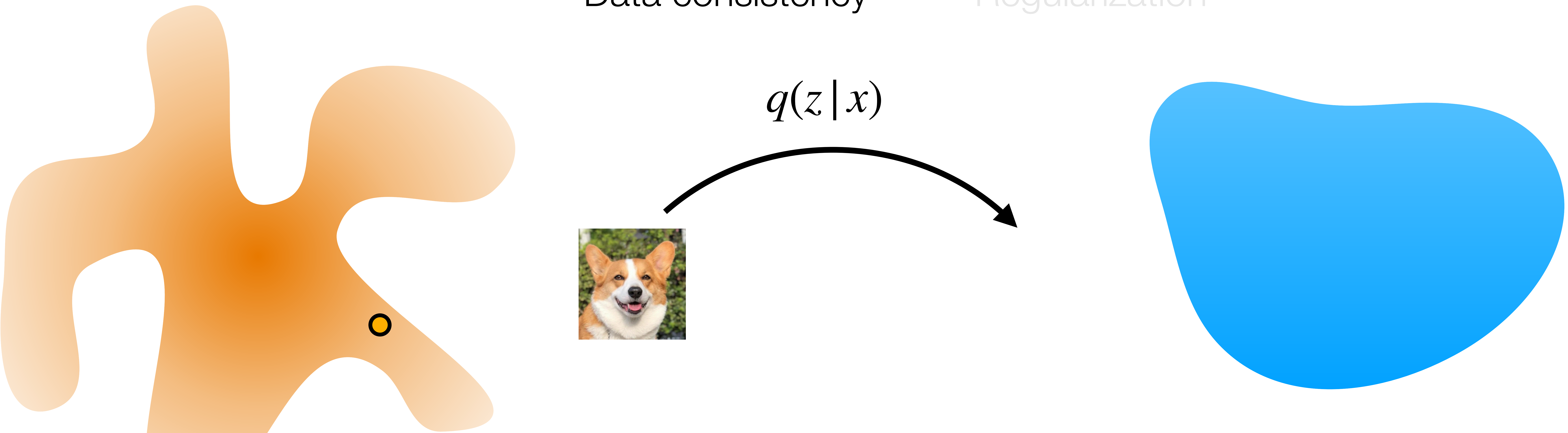
$$\mathscr{L}(x) = \underbrace{\mathbb{E}_{q(z|x)}[\log p(x\,|\,z)]}_{\text{Data consistency}} - \underbrace{KL(q(z\,|\,x)\,|\,p(z))}_{\text{Regularization}}$$



ELBO: Likelihood as an L_2 term

$$\mathscr{L}(x) = \mathbb{E}_{q(z|x)}[\log p(x\,|\,z)] - KL(q(z\,|\,x)\,|\,p(z))$$

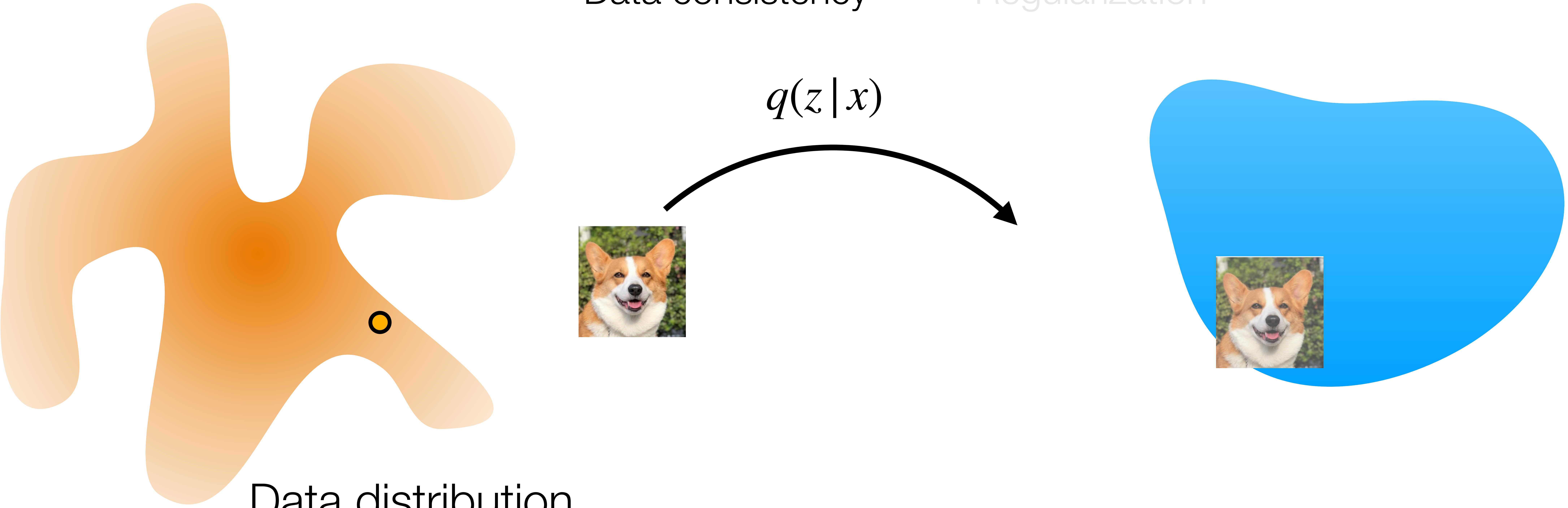
Data consistency Regularization



ELBO: Likelihood as an L_2 term

$$\mathcal{L}(x) = \mathbb{E}_{q(z|x)}[\log p(x\,|\,z)] - KL(q(z\,|\,x)\,|\,p(z))$$

Data consistency Regularization

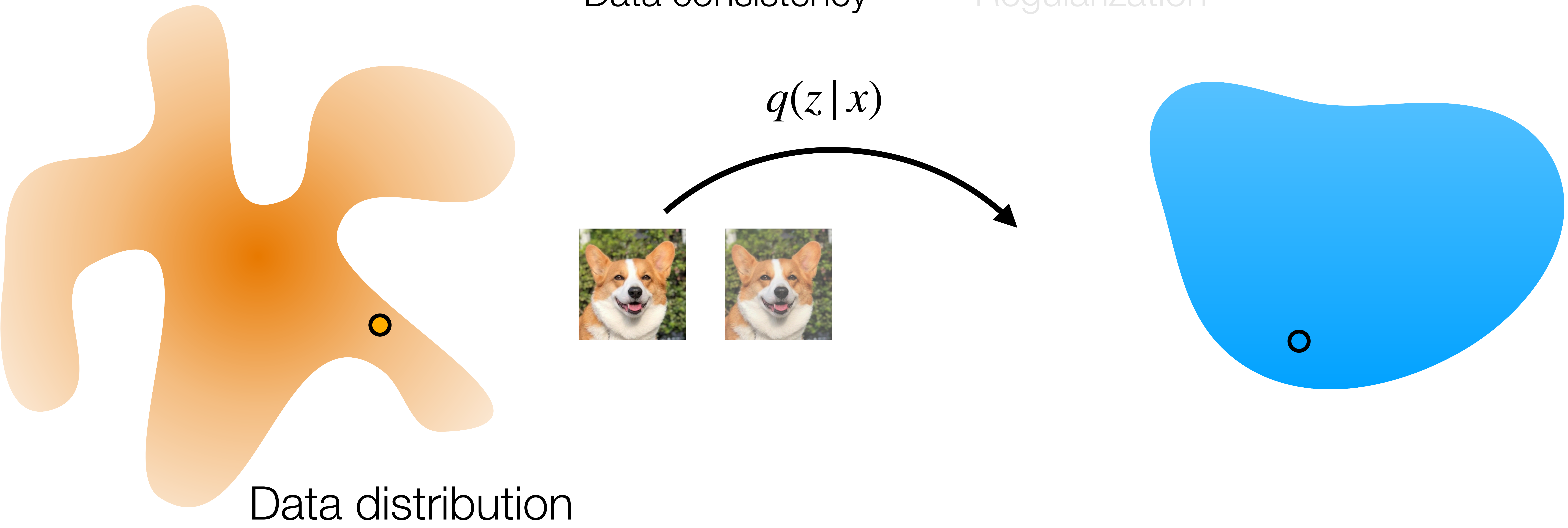


Data distribution

ELBO: Likelihood as an L_2 term

$$\mathscr{L}(x) = \mathbb{E}_{q(z|x)}[\log p(x\,|\,z)] - KL(q(z\,|\,x)\,|\,p(z))$$

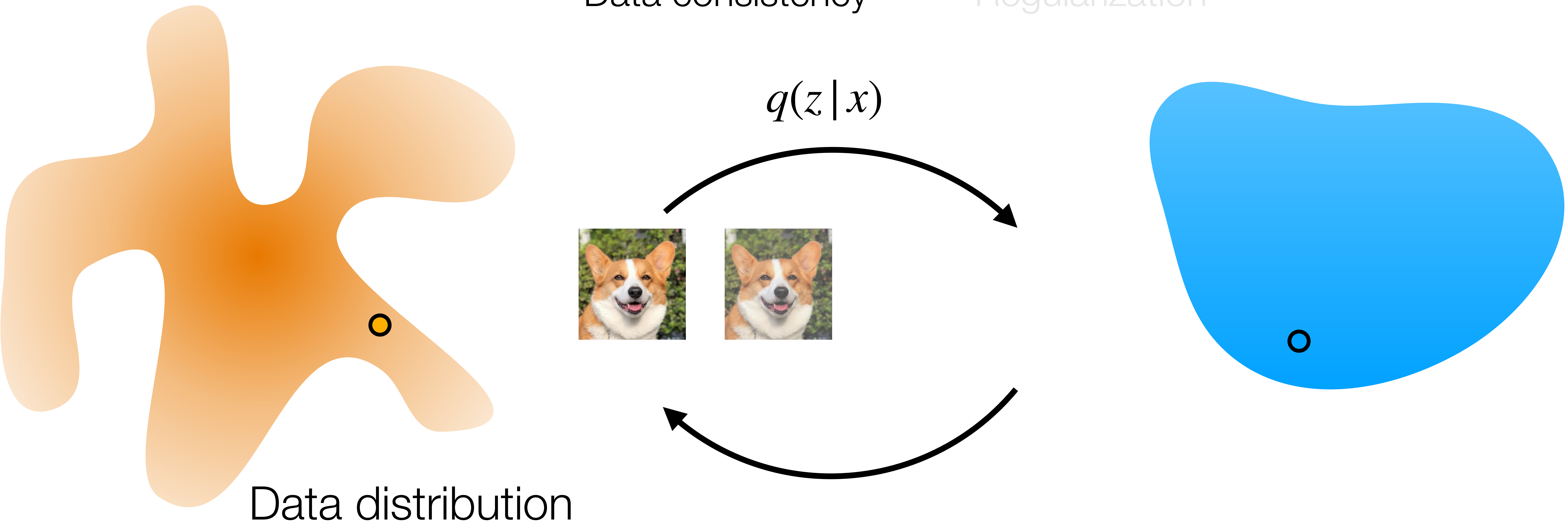
Data consistency Regularization



ELBO: Likelihood as an L_2 term

$$\mathscr{L}(x) = \mathbb{E}_{q(z|x)}[\log p(x|z)] - KL(q(z|x)\,|\,p(z))$$

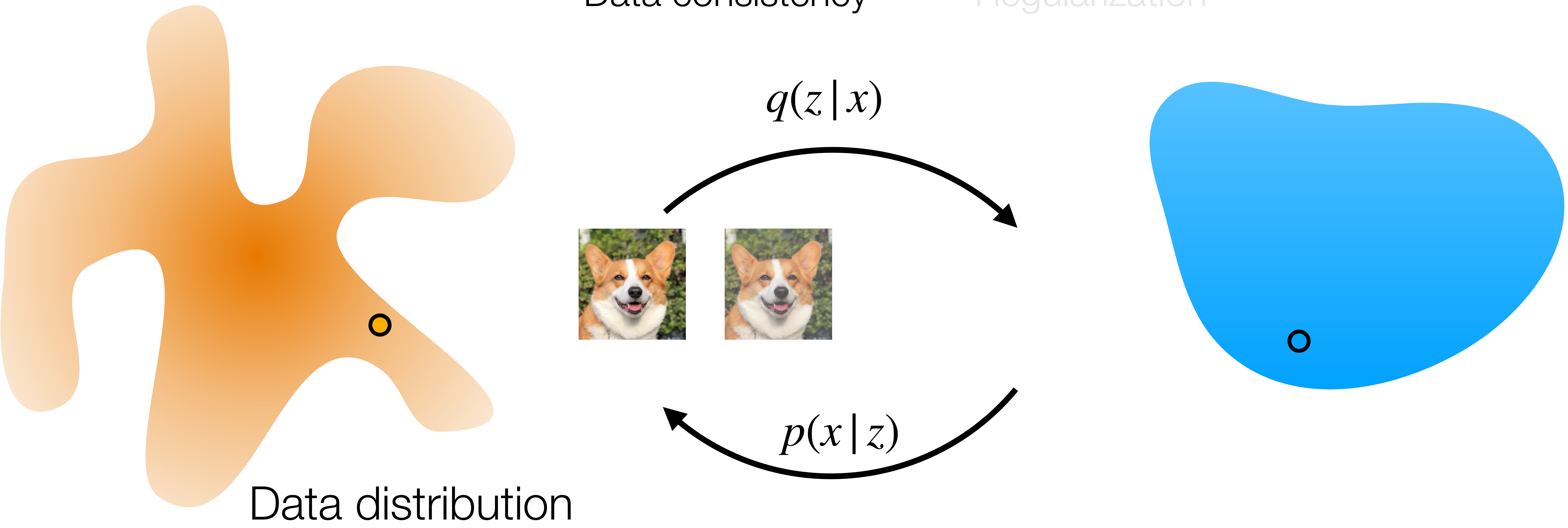
Data consistency Regularization



ELBO: Likelihood as an L_2 term

$$\mathscr{L}(x) = \mathbb{E}_{q(z|x)}[\log p(x|z)] - KL(q(z|x)\,|\,p(z))$$

Data consistency Regularization

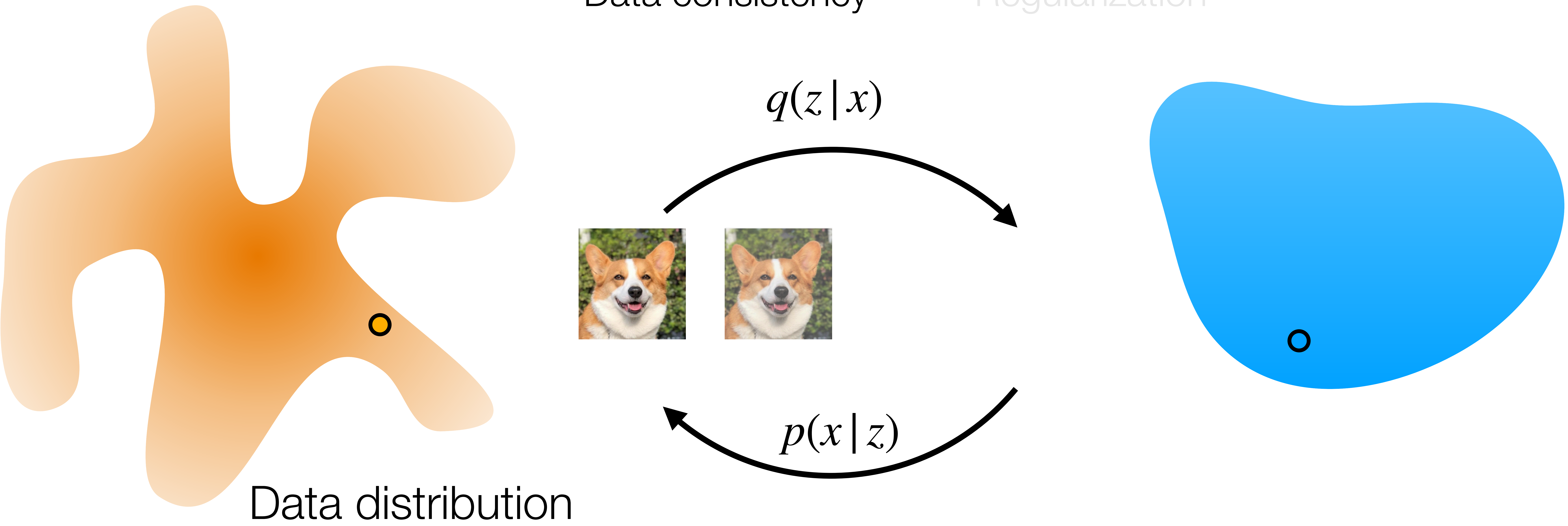


ELBO: Likelihood as an L_2 term

$$\mathbb{E}_{q(z|x)}[\log p(x|z)] = L_2$$

$$\mathcal{L}(x) = \mathbb{E}_{q(z|x)}[\log p(x|z)] - KL(q(z|x)\,|\,p(z))$$

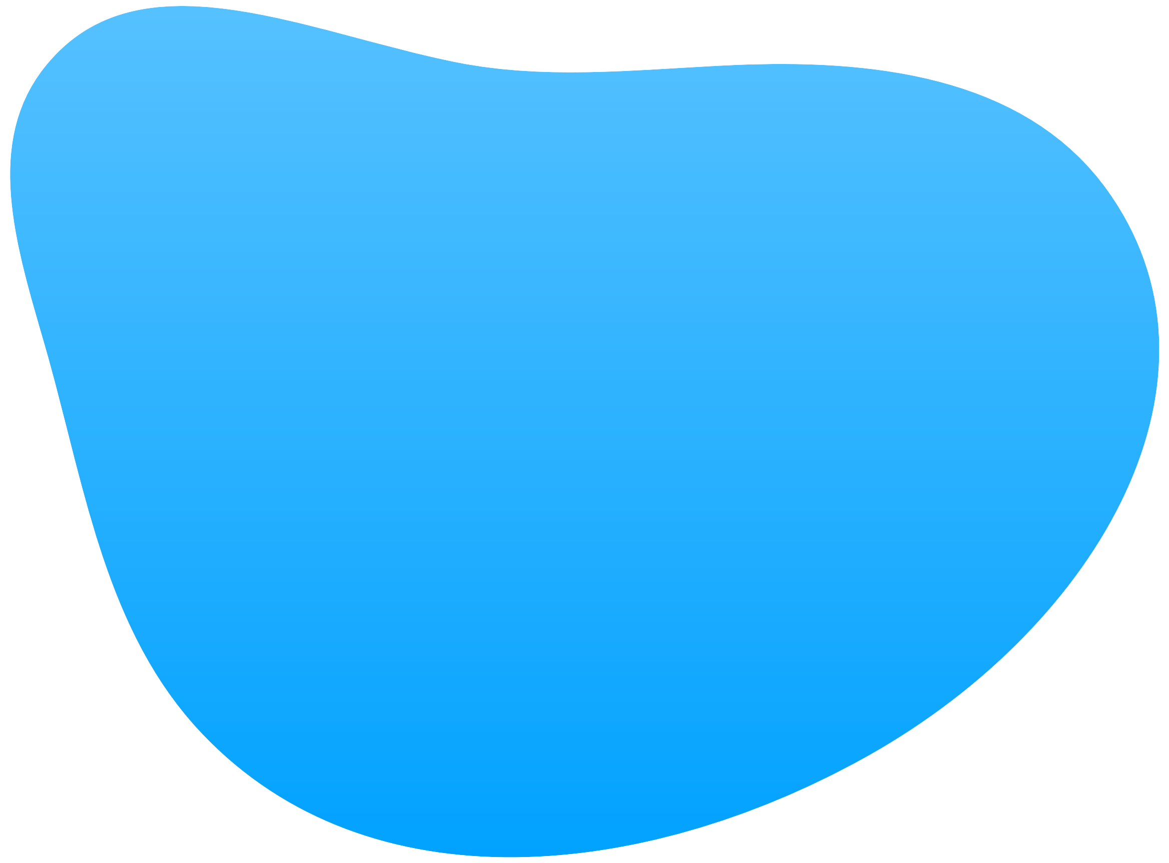
Data consistency Regularization



ELBO: KL divergence term

Data consistency Regularization

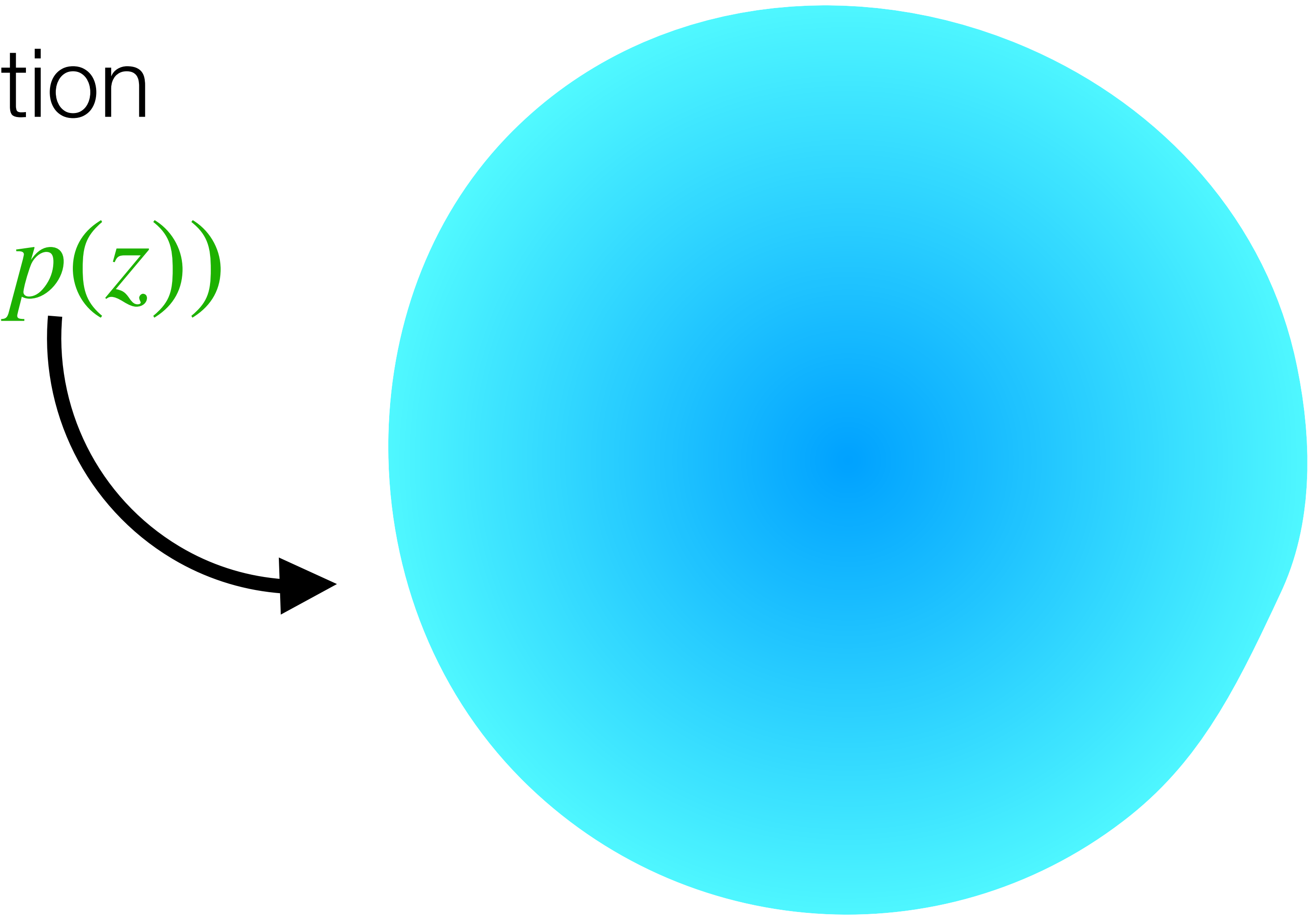
$$\mathscr{L}(x) = \mathbb{E}_{q(z|x)}[\log p(x|z)] - KL(q(z|x)|p(z))$$

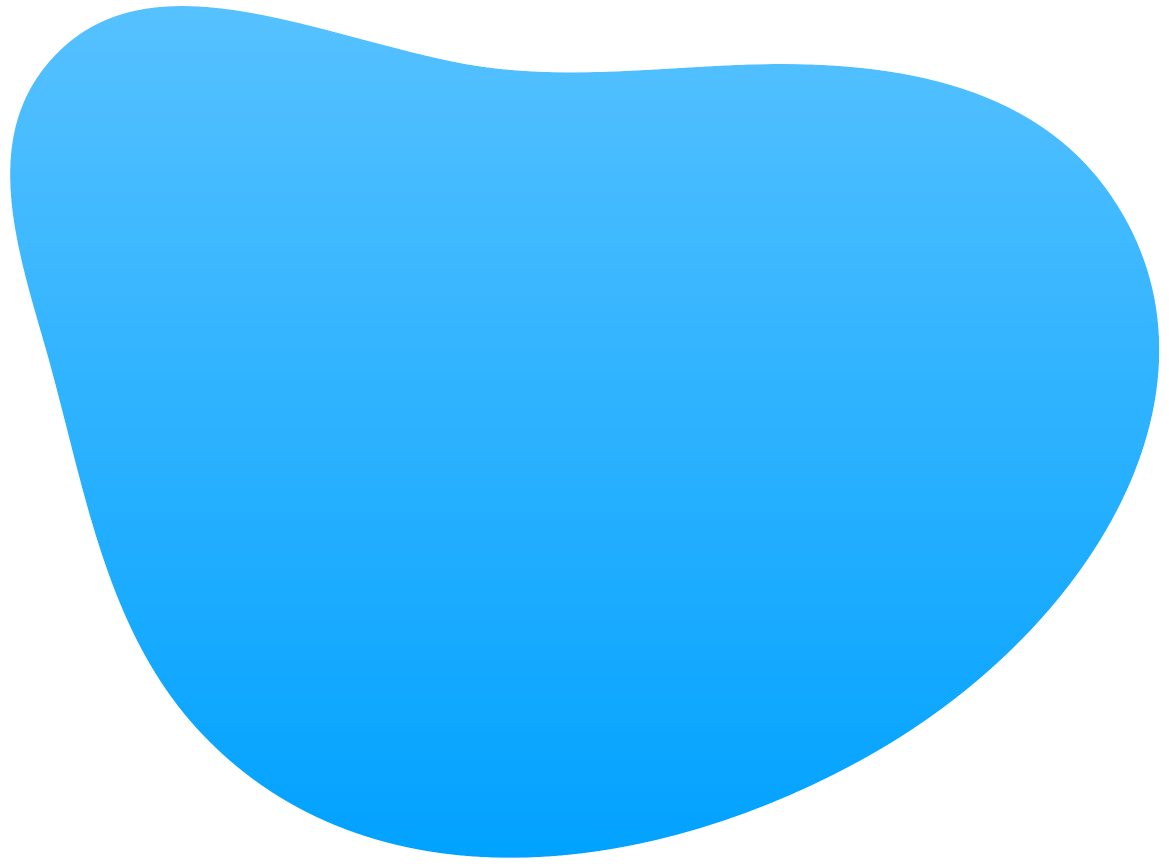


ELBO: KL divergence term

Data consistency Regularization

$$\mathscr{L}(x) = \mathbb{E}_{q(z|x)}[\log p(x|z)] - KL(q(z|x)\,|\,p(z))$$

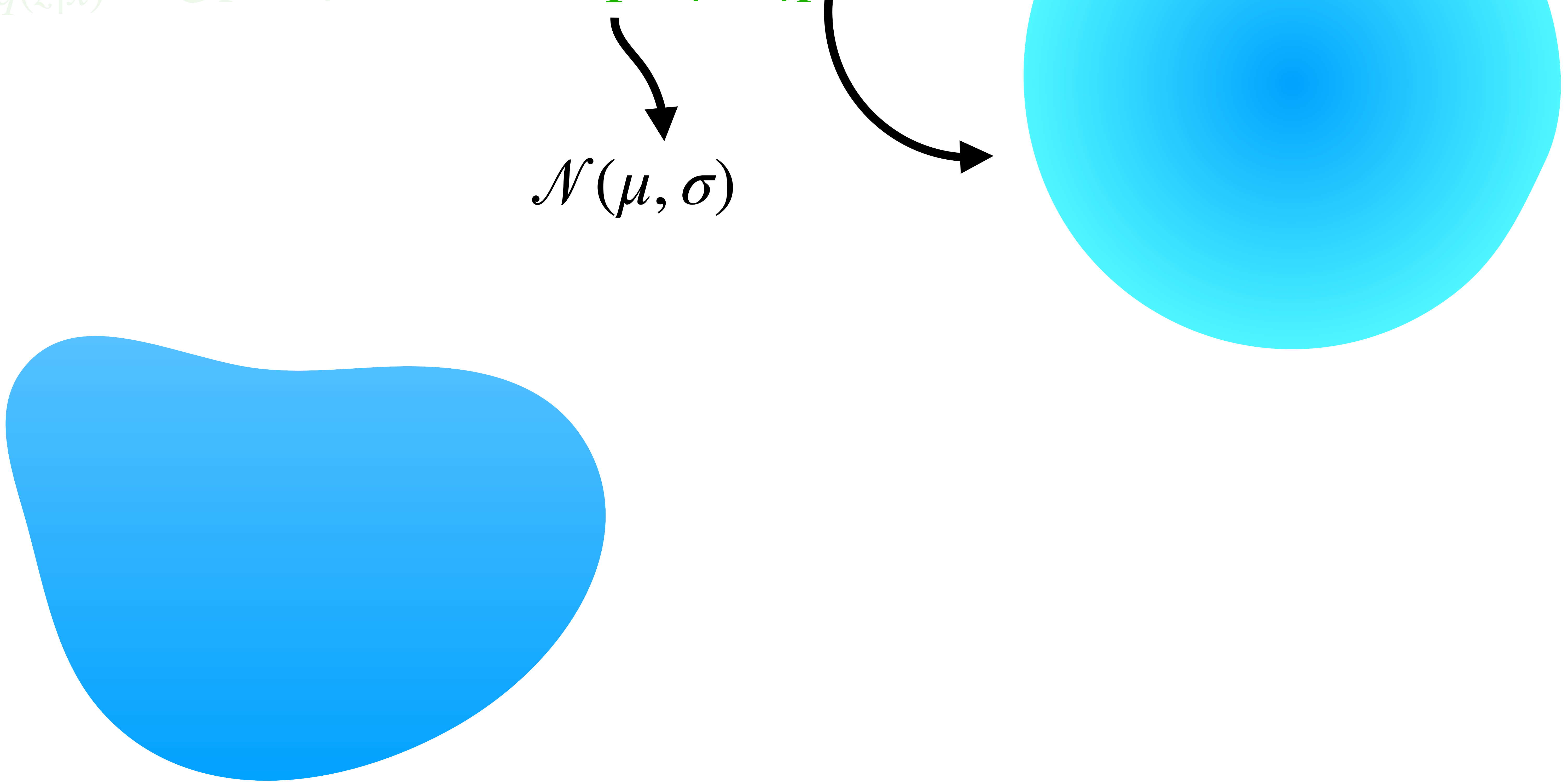




ELBO: KL divergence term

Data consistency Regularization

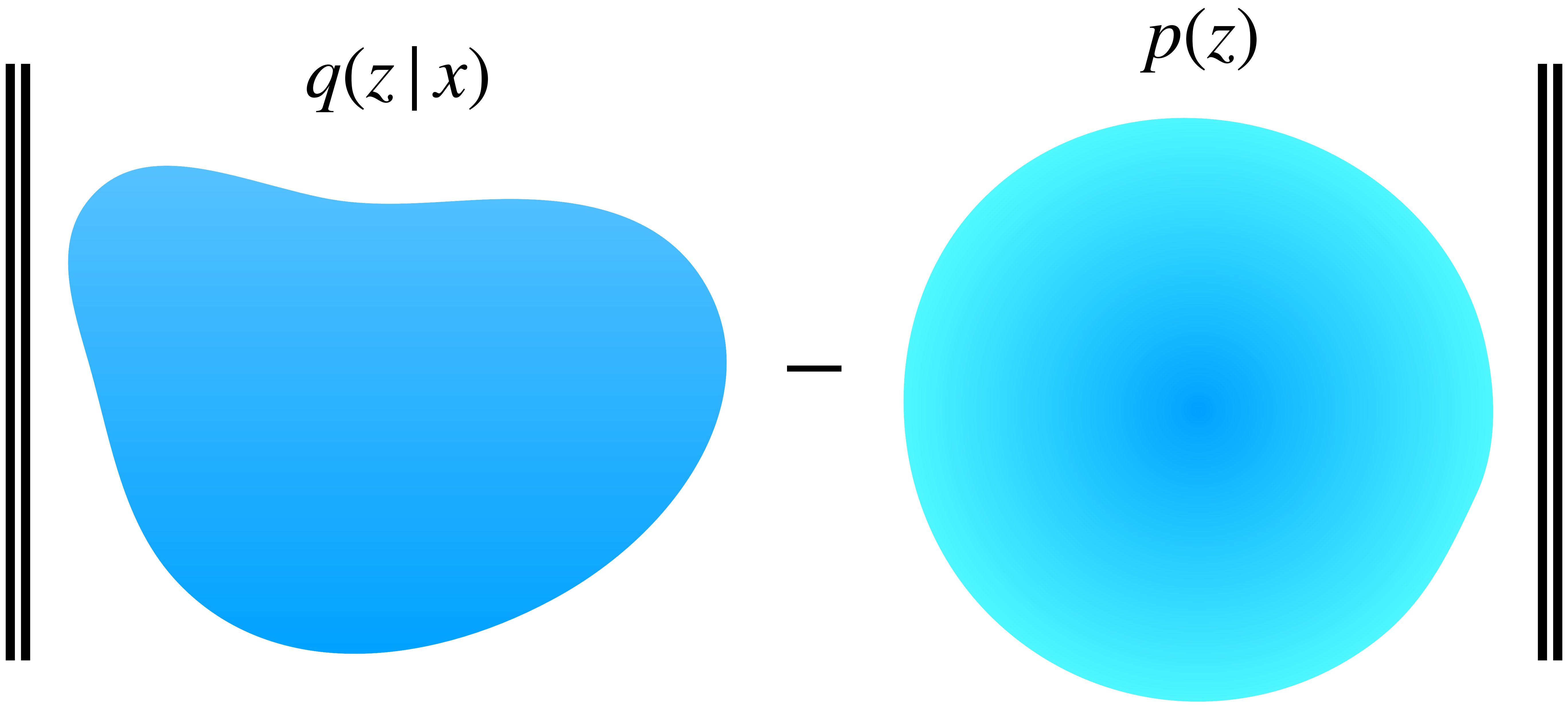
$$\mathscr{L}(x) = \mathbb{E}_{q(z|x)}[\log p(x\,|\,z)] - KL(q(z\,|\,x)\,|\,p(z))$$



$\mathcal{N}(\mu, \sigma)$

Data consistency Regularization

$$\mathscr{L}(x) = \mathbb{E}_{q(z|x)}[\log p(x|z)] - KL(q(z|x)|p(z))$$



Evidence lower bound (ELBO)

$$\mathscr{L}(x) = \mathbb{E}_{q(z|x)}[\log p(x\,|\,z)] - KL(q(z\,|\,x)\,|\,p(z))$$

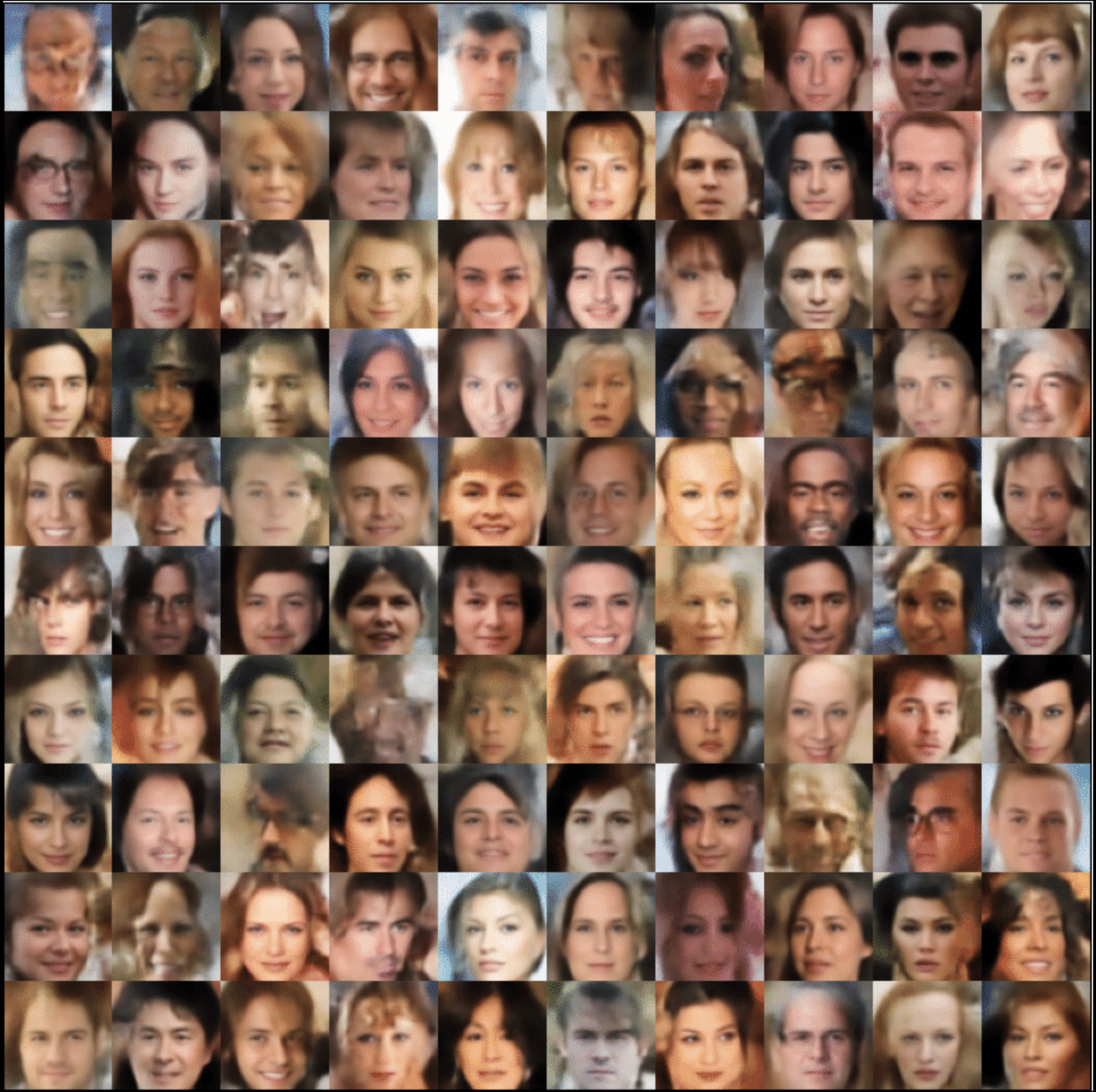
Evidence lower bound (ELBO)

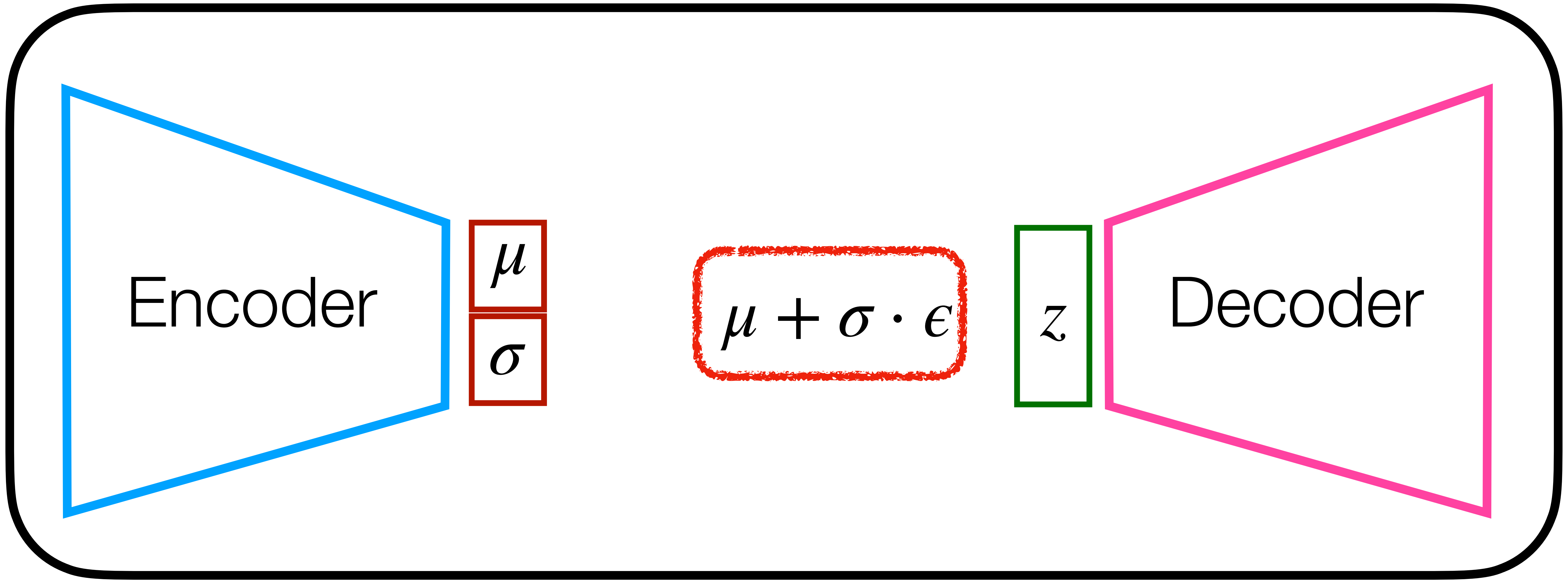
$$\mathscr{L}(x) = \underbrace{\mathbb{E}_{q(z|x)}[\log p(x\,|\,z)]}_{L_2} - KL(q(z\,|\,x)\,|\,p(z))$$

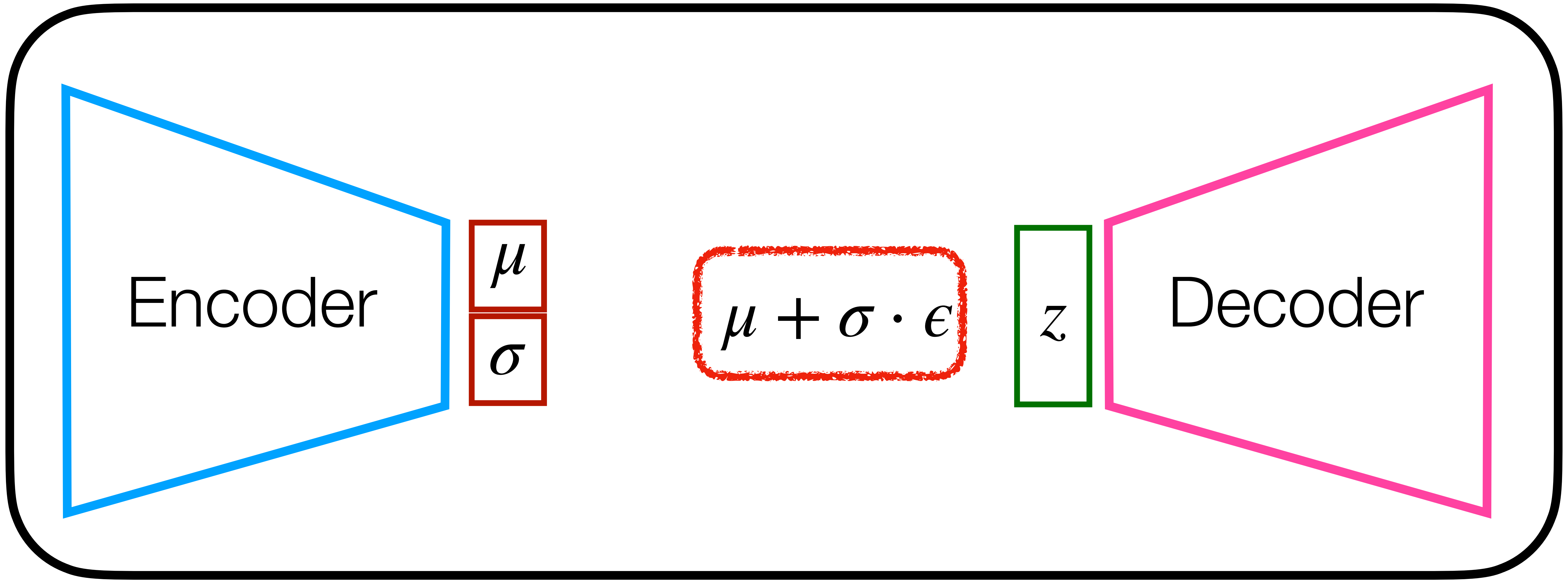
Evidence lower bound (ELBO)

$$\mathscr{L}(x) = \underbrace{\mathbb{E}_{q(z|x)}[\log p(x\,|\,z)]}_{L_2} - \underbrace{KL(q(z\,|\,x)\,|\,p(z))}_{\text{Latent space regularization}}$$

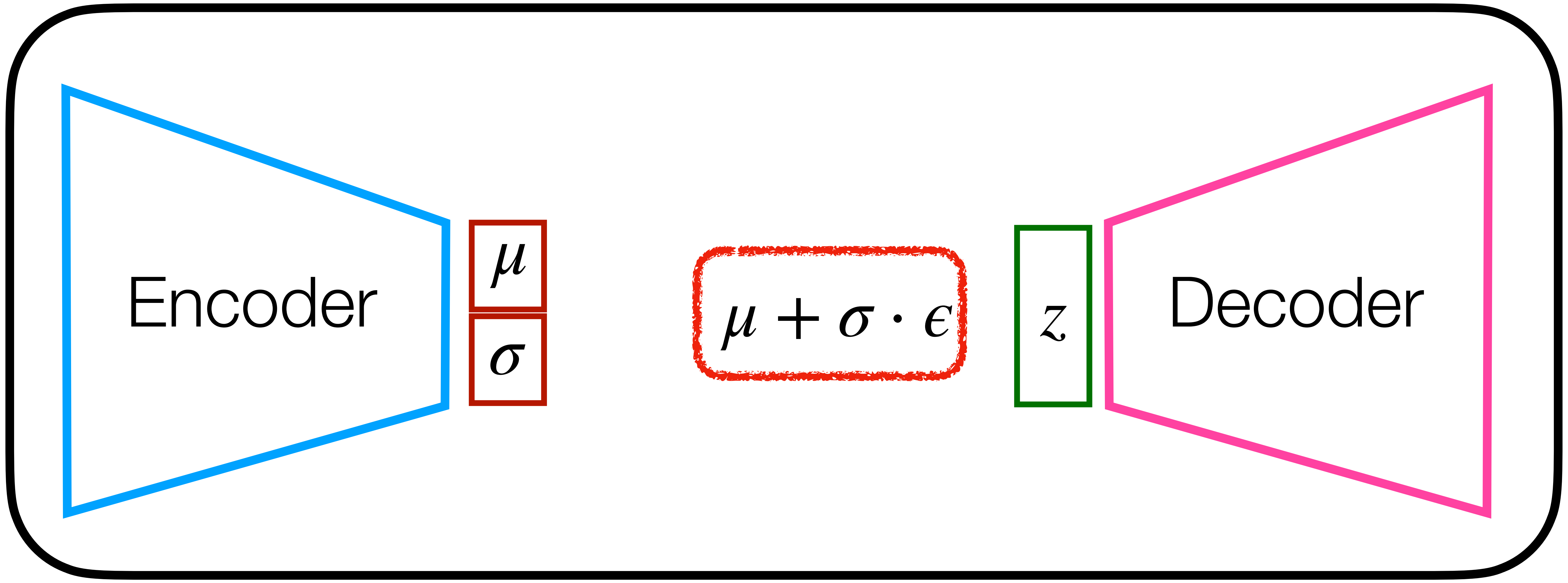
With VAEs:
Results are good but slightly blurred




Encoder
μ
σ
$\mu + \sigma \cdot \epsilon$
z
Decoder

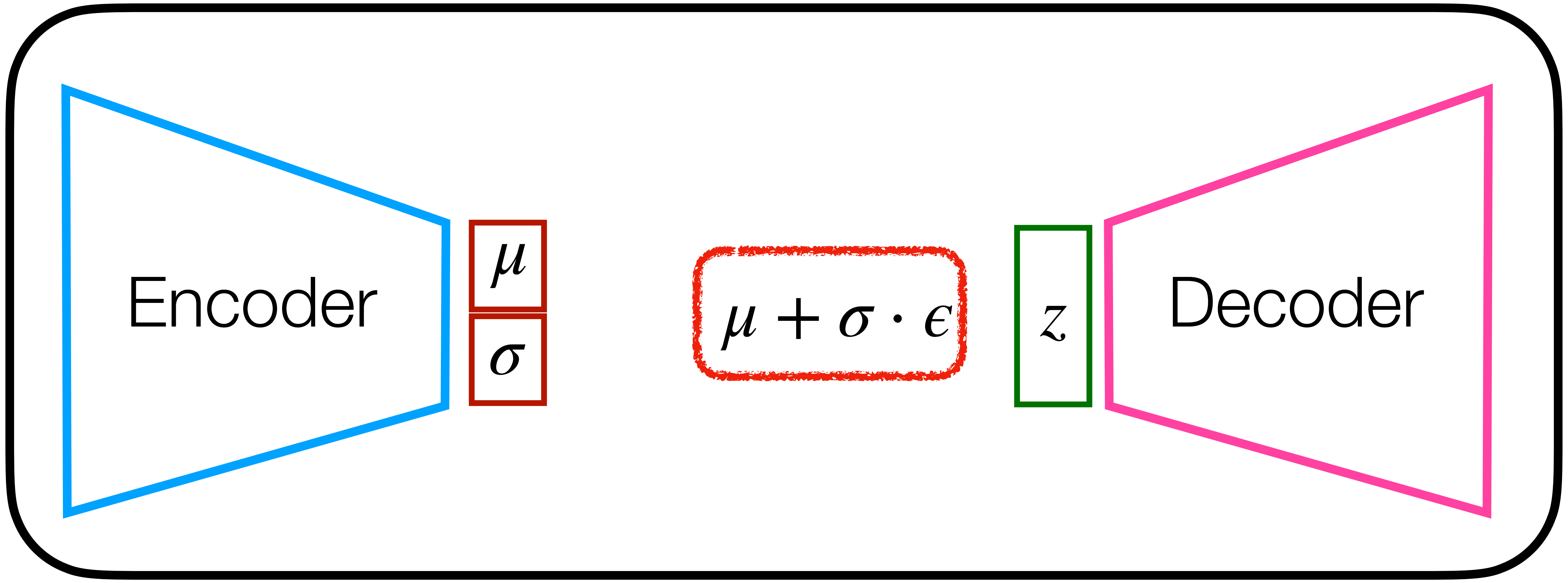


Cons:



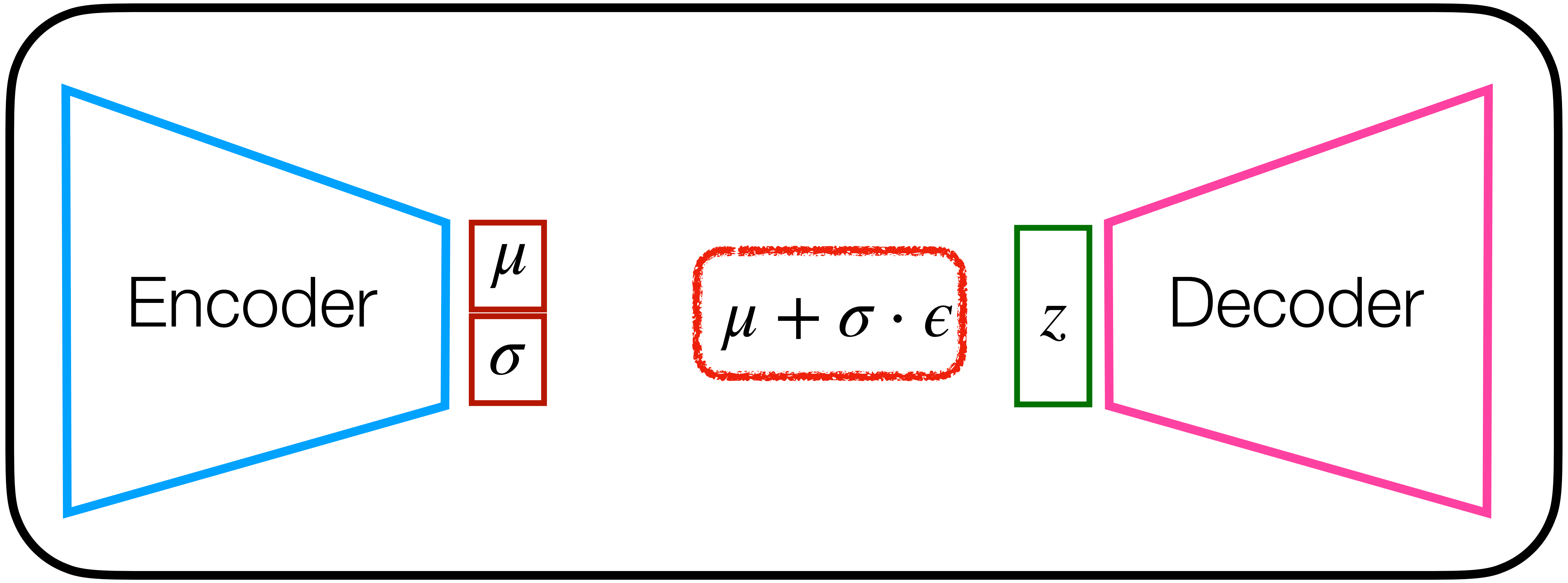
Cons:

- Model architectures are restricted



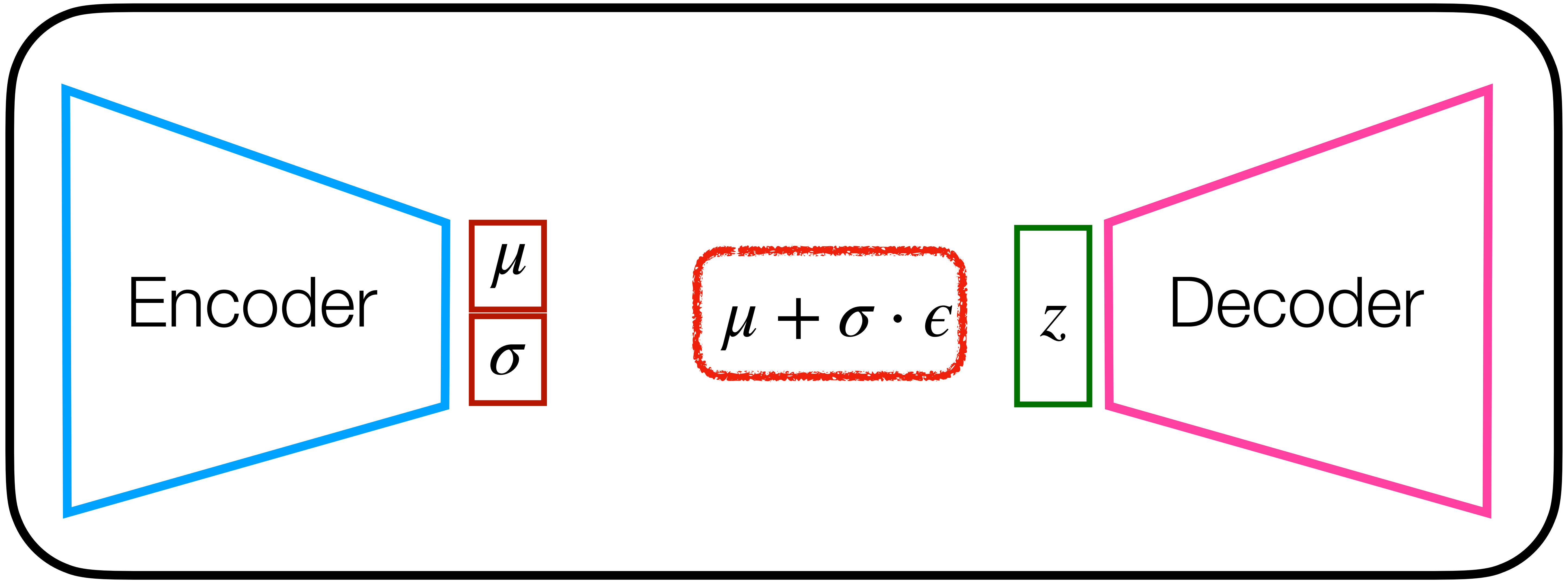
Cons:

- Model architectures are restricted
- We cannot pick an arbitrary neural network that can take as input data samples and output a scalar.



Cons:

- Model architectures are restricted
- We cannot pick an arbitrary neural network that can take as input data samples and output a scalar.
- It has to be a valid PDF



Cons:

- Model architectures are restricted
- We cannot pick an arbitrary neural network that can take as input data samples and output a scalar.
- It has to be a valid PDF
- In VAEs, we use approximations to circumvent these issues

From VAEs to Diffusion models

Variational Autoencoders (VAEs)

Energy-based models (EBMs)

MCMC methods for EBMs

Score-based Generative models (SBGMs)

MCMC methods for SBGMs

SDE-based diffusion models

Energy-based models

Energy-based models

- Very flexible model architectures

Energy-based models

- Very flexible model architectures
- Stable training

Energy-based models

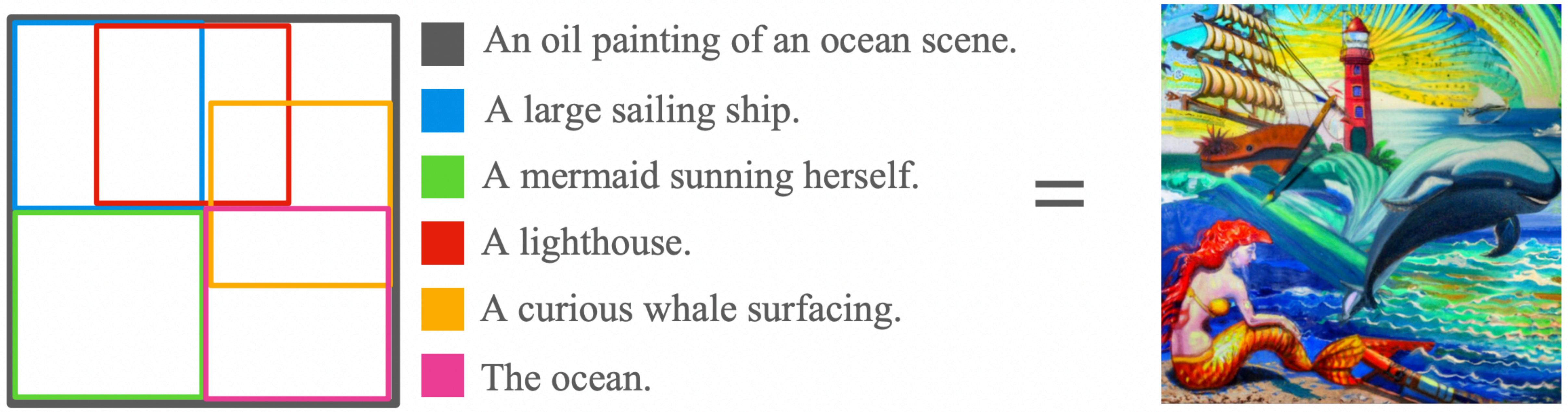
- Very flexible model architectures
- Stable training
- Relatively high sample quality

Energy-based models

- Very flexible model architectures
- Stable training
- Relatively high sample quality
- Very close to diffusion models

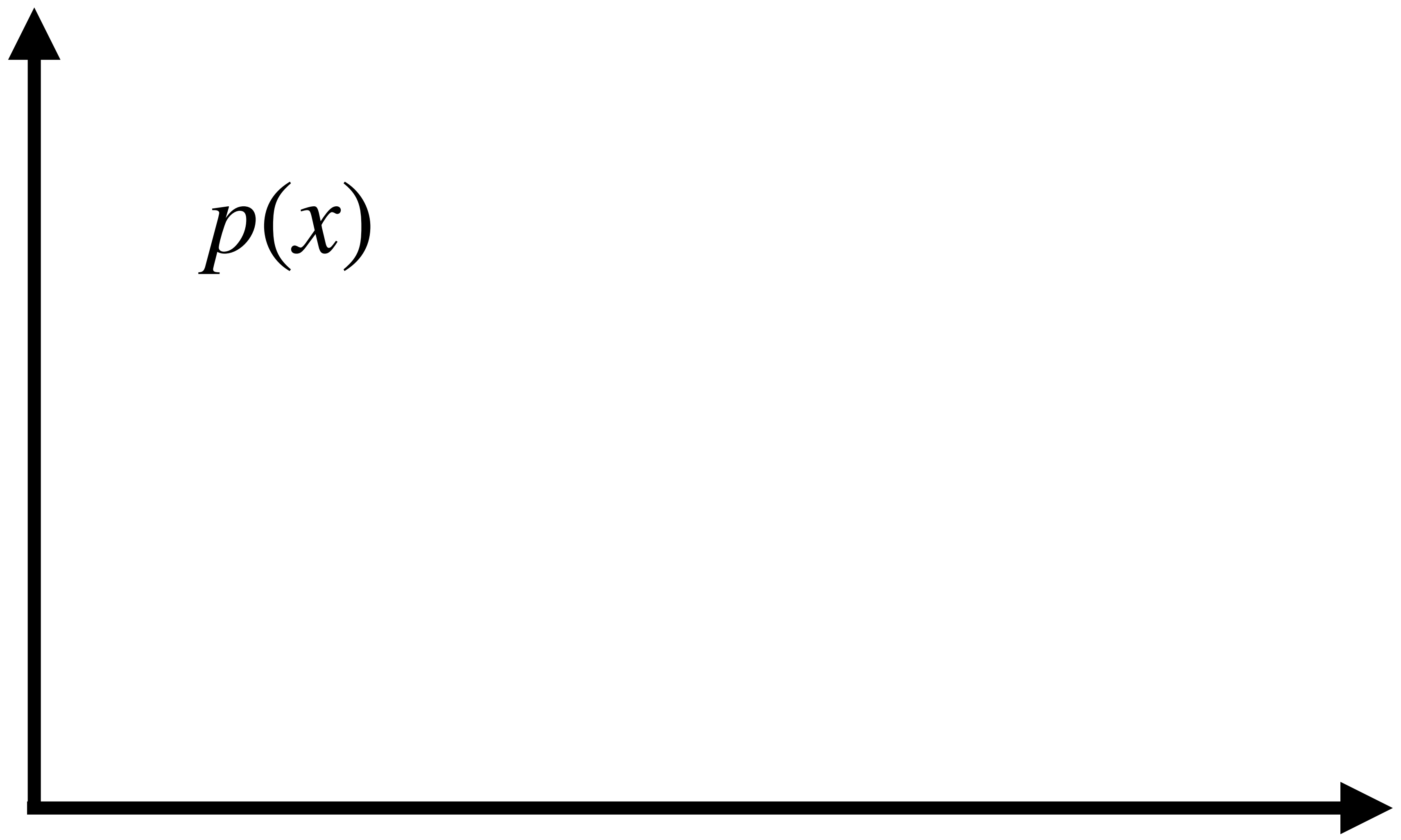
Energy-based models

- Very flexible model architectures
- Stable training
- Relatively high sample quality
- Very close to diffusion models
- Flexible composition


⬛ An oil painting of an ocean scene.
🟦 A large sailing ship.
🟩 A mermaid sunning herself.
🟥 A lighthouse.
🟧 A curious whale surfacing.
🟪 The ocean.
=

Bayesian statistics: Probability density function

- PDFs are the key building blocks in generative modeling



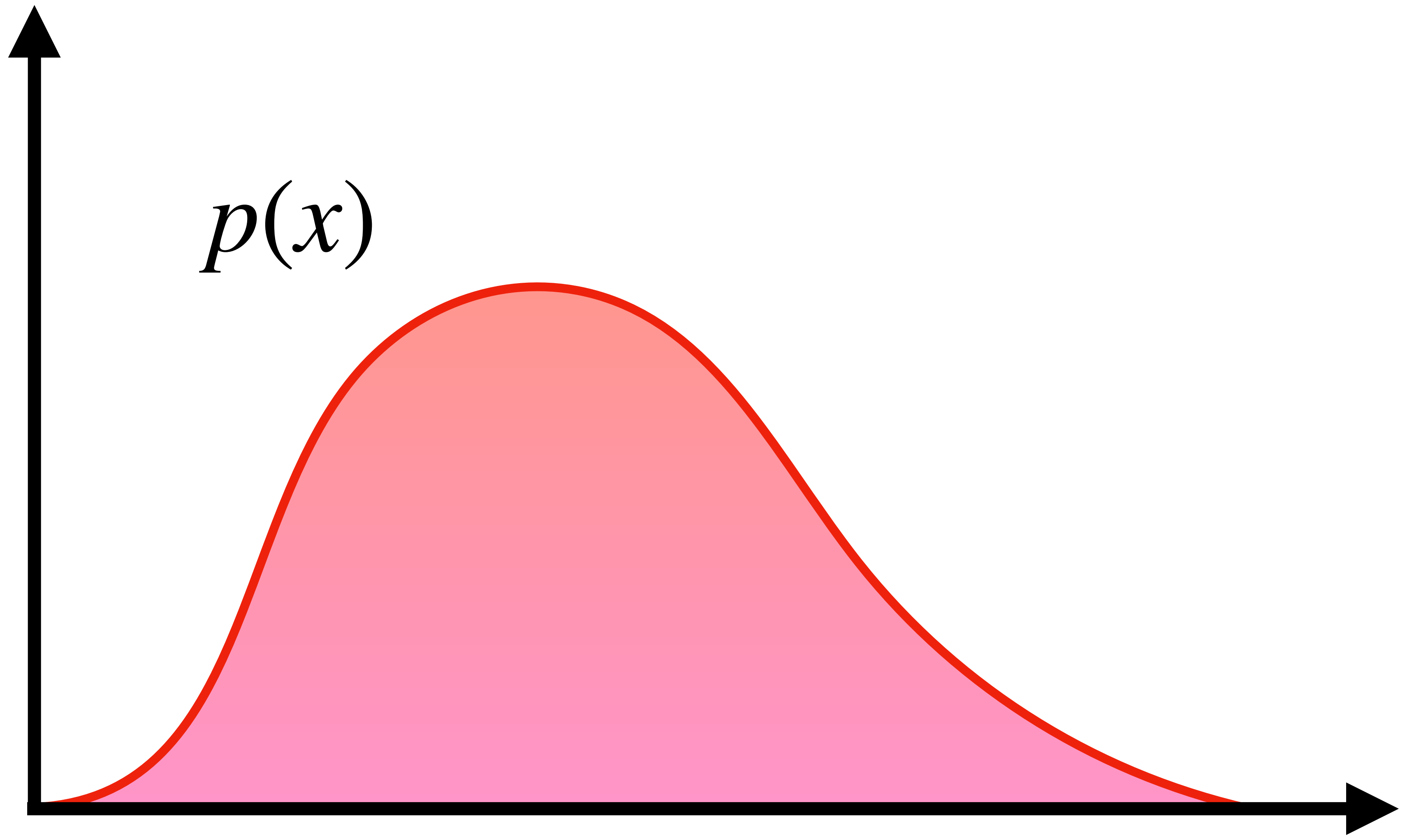
Probability density function

$$x \sim X$$

Sampling

Bayesian statistics: Probability density function

- PDFs are the key building blocks in generative modeling



Probability density function

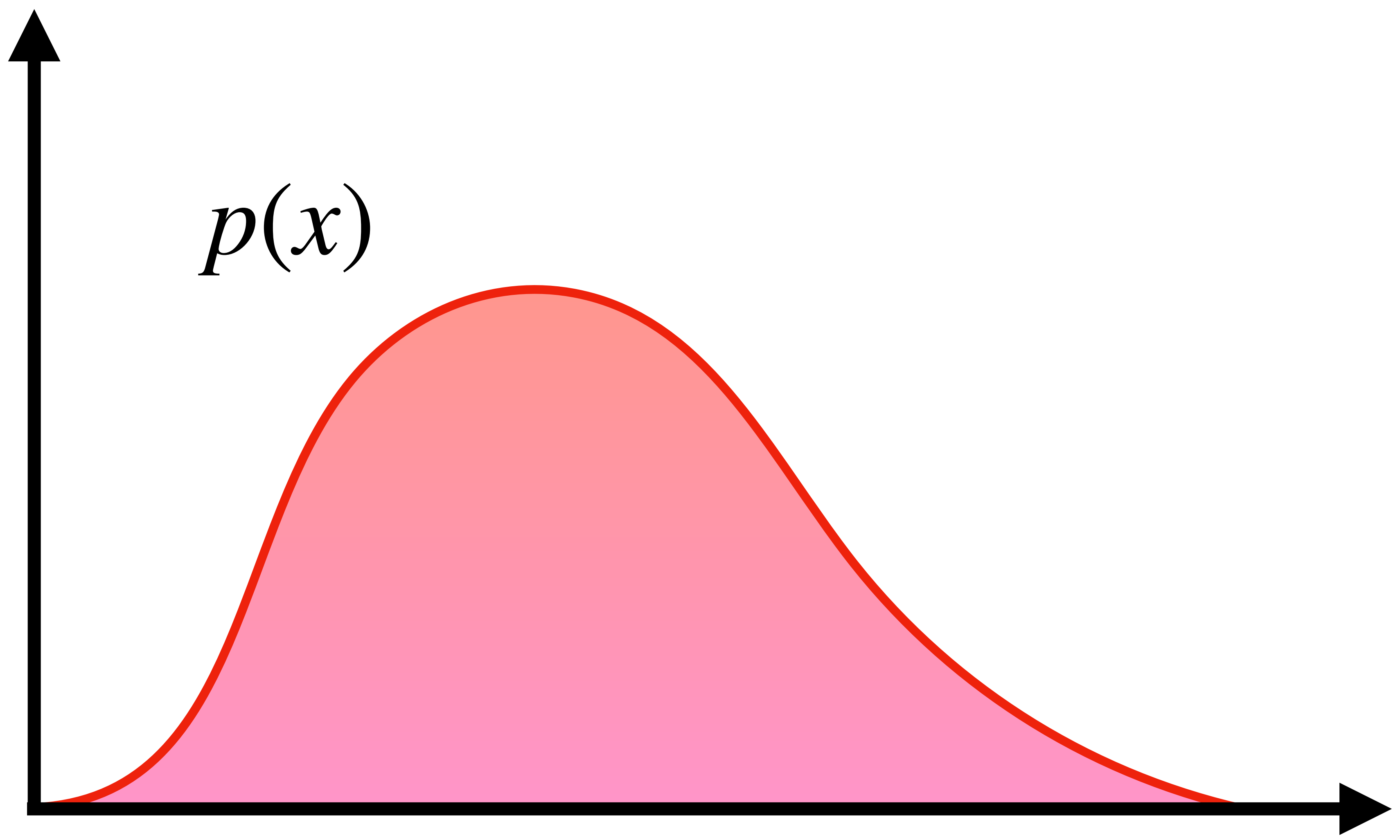
$$x \sim X$$

Sampling

Bayesian statistics: Probability density function

- PDFs are the key building blocks in generative modeling

Non-negative: $p(x) \geq 0$



Probability density function

$$x \sim X$$

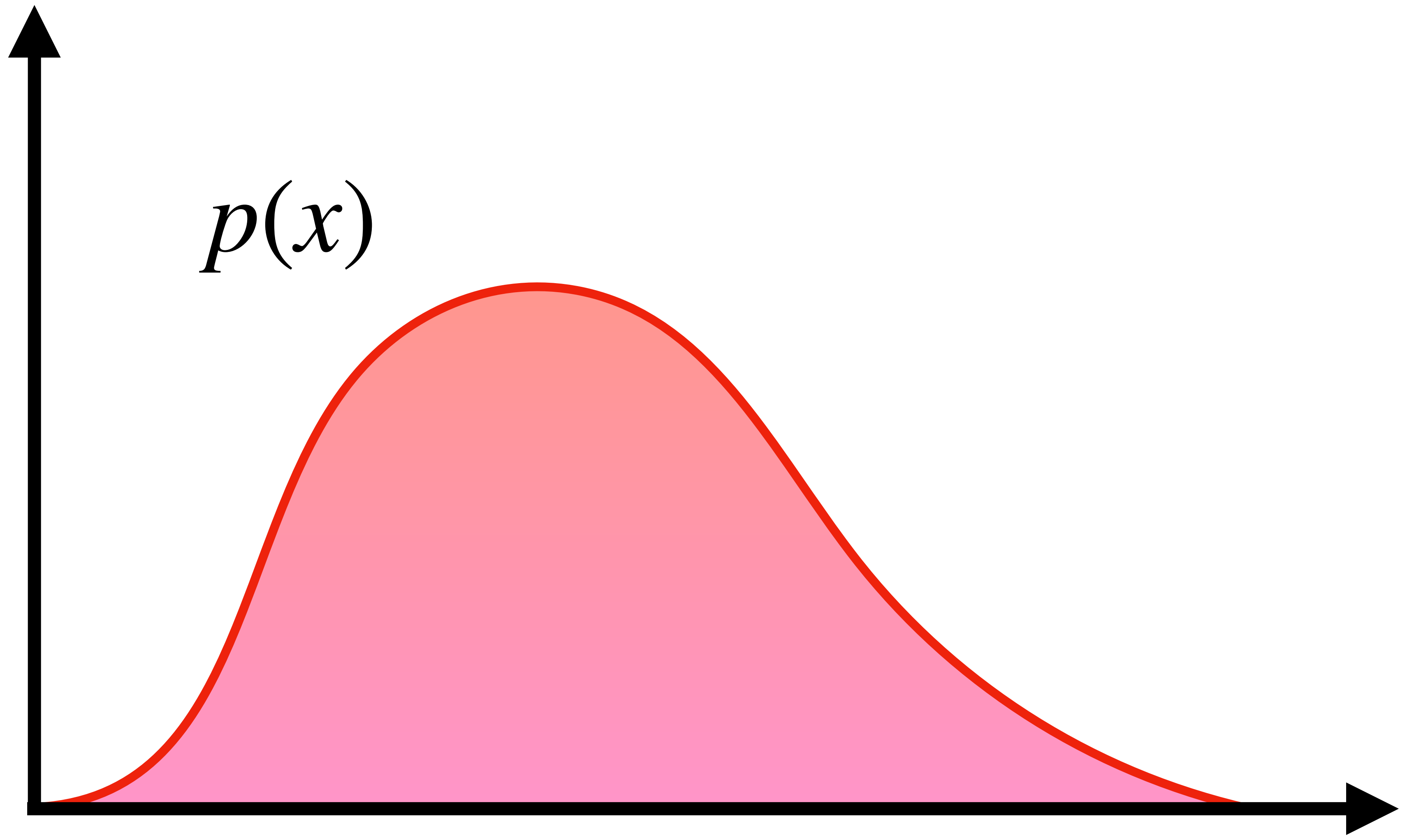
Sampling

Bayesian statistics: Probability density function

- PDFs are the key building blocks in generative modeling

Non-negative: $p(x) \geq 0$

Normalized pdf: $\int p(x)dx = 1$



Probability density function

$$x \sim X$$

Sampling

Parameterizing probability distributions

Non-negative: $p(x) \geq 0$

Coming up with a non-negative function $p_\theta(x)$ is not hard

Parameterizing probability distributions

Non-negative: $p(x) \geq 0$

Coming up with a non-negative function $p_\theta(x)$ is not hard

Given any function or an arbitrary neural network $f_\theta(x)$, we can chose:

Parameterizing probability distributions

Non-negative: $p(x) \geq 0$

Coming up with a non-negative function $p_\theta(x)$ is not hard

Given any function or an arbitrary neural network $f_\theta(x)$, we can chose:

$g_\theta(x) = f_\theta(x)^2$

Parameterizing probability distributions

Non-negative: $p(x) \geq 0$

Coming up with a non-negative function $p_\theta(x)$ is not hard

Given any function or an arbitrary neural network $f_\theta(x)$, we can chose:

$$g_\theta(x) = f_\theta(x)^2$$

$$g_\theta(x) = \exp(f_\theta(x))$$

Parameterizing probability distributions

Non-negative: $p(x) \geq 0$

Coming up with a non-negative function $p_\theta(x)$ is not hard

Given any function or an arbitrary neural network $f_\theta(x)$, we can chose:

$$g_\theta(x) = f_\theta(x)^2$$

$$g_\theta(x) = \exp(f_\theta(x))$$

$$g_\theta(x) = |f_\theta(x)|$$

Parameterizing probability distributions

Parameterizing probability distributions

Normalized pdf: $\int p(x)dx = 1$

Parameterizing probability distributions

Normalized pdf: $\int p(x)dx = 1$

This property ensures that total volume is fixed: i.e. increasing $p_\theta(x_{train})$ guarantees that x_{train} becomes more likely (compared to the rest).

Parameterizing probability distributions

Parameterizing probability distributions

Problem: $g_\theta(x) \geq 0$ is easy, but $g_\theta(x)$ might not be normalized

Parameterizing probability distributions

Problem: $g_\theta(x) \geq 0$ is easy, but $g_\theta(x)$ might not be normalized

For example: Energy-based model

- we assume the following form of $g_\theta(x) = \exp f_\theta(x)$

Energy-based model

$$p_\theta(x) = \frac{1}{Z(\theta)} \exp(f_\theta(x))$$

$$Z(\theta) = \int \exp(f_\theta(x))dx$$

Energy-based model

$$p_\theta(x) = \frac{1}{Z(\theta)} \exp(f_\theta(x))$$

$$Z(\theta) = \int \exp(f_\theta(x)) dx$$

The normalization constant is also called a partition function

Energy-based model

$$p_\theta(x) = \frac{1}{Z(\theta)} \exp(f_\theta(x))$$

$$Z(\theta) = \int \exp(f_\theta(x)) dx$$

The normalization constant is also called a partition function

Why exponential of $f_\theta(x)$?

Energy-based model

$$p_\theta(x) = \frac{1}{Z(\theta)}\exp(f_\theta(x))$$

$$Z(\theta) = \int \exp(f_\theta(x))dx$$

The normalization constant is also called a partition function

Why exponential of $f_\theta(x)$?

Want to capture big variations in the probability, log-space is a natural choice

Energy-based model

$$p_\theta(x) = \frac{1}{Z(\theta)} \exp(f_\theta(x))$$

$$Z(\theta) = \int \exp(f_\theta(x))dx$$

The normalization constant is also called a partition function

Why exponential of $f_\theta(x)$?

Want to capture big variations in the probability, log-space is a natural choice

In statistical physics, these distributions arise under fairly general assumptions

Energy-based model

$$p_\theta(x) = \frac{1}{Z(\theta)} \exp(f_\theta(x))$$

$$Z(\theta) = \int \exp(f_\theta(x)) dx$$

The normalization constant is also called a partition function

Why exponential of $f_\theta(x)$?

Want to capture big variations in the probability, log-space is a natural choice

In statistical physics, these distributions arise under fairly general assumptions

- $-f_\theta(x)$ is called the energy, hence the name.

Energy-based model

$$p_\theta(x) = \frac{1}{Z(\theta)} \exp(f_\theta(x))$$

$$Z(\theta) = \int \exp(f_\theta(x)) dx$$

Energy-based model

$$p_\theta(x) = \frac{1}{Z(\theta)} \exp(f_\theta(x))$$

$$Z(\theta) = \int \exp(f_\theta(x)) dx$$

$$\log p_\theta(x) = \log \left[\frac{1}{Z(\theta)} \exp(f_\theta(x)) \right]$$

Energy-based model

$$p_\theta(x) = \frac{1}{Z(\theta)} \exp(f_\theta(x))$$

$$Z(\theta) = \int \exp(f_\theta(x))dx$$

$$\log p_\theta(x) = \log \left[\frac{1}{Z(\theta)} \exp(f_\theta(x)) \right]$$

$$\log p_\theta(x) = \log \exp(f_\theta(x)) - \log Z(\theta)$$

Energy-based model

$$p_\theta(x) = \frac{1}{Z(\theta)}\exp(f_\theta(x))$$

$$Z(\theta) = \int \exp(f_\theta(x))dx$$

$$\log p_\theta(x) = \log\left[\frac{1}{Z(\theta)}\exp(f_\theta(x))\right]$$

$$\log p_\theta(x) = \log\exp(f_\theta(x)) - \log Z(\theta)$$

$$\log p_\theta(x) = f_\theta(x) - \log Z(\theta)$$

Energy-based model

$$p_\theta(x) = \frac{1}{Z(\theta)} \exp(f_\theta(x)) \qquad Z(\theta) = \int \exp(f_\theta(x))dx$$

$$\log p_\theta(x) = \log \left[\frac{1}{Z(\theta)} \exp(f_\theta(x)) \right]$$

$$\log p_\theta(x) = \log \exp(f_\theta(x)) - \log Z(\theta)$$

$$\log p_\theta(x) = f_\theta(x) - \log Z(\theta)$$

This term is called the log likelihood

Energy-based model

$$\log p_\theta(x) = f_\theta(x) - \log Z(\theta) \qquad\qquad Z(\theta) = \int \exp(f_\theta(x))dx$$

Energy-based model

$$\log p_\theta(x) = f_\theta(x) - \log Z(\theta) \qquad Z(\theta) = \int \exp(f_\theta(x))dx$$

To train the model, we want to maximize the log-likelihood:

Energy-based model

$$\log p_\theta(x) = f_\theta(x) - \log Z(\theta) \qquad Z(\theta) = \int \exp(f_\theta(x))dx$$

To train the model, we want to maximize the log-likelihood:

$$\max_\theta (f_\theta(x_{train}) - \log Z(\theta))$$

Energy-based model

$$\log p_\theta(x) = f_\theta(x) - \log Z(\theta) \qquad\qquad Z(\theta) = \int \exp(f_\theta(x))dx$$

To train the model, we want to maximize the log-likelihood:

$$\max_\theta (f_\theta(x_{train}) - \log Z(\theta))$$

We need to compute the gradient of the log-likelihood:

Energy-based model

$$\log p_\theta(x) = f_\theta(x) - \log Z(\theta) \qquad Z(\theta) = \int \exp(f_\theta(x))dx$$

To train the model, we want to maximize the log-likelihood:

$$\max_\theta (f_\theta(x_{train}) - \log Z(\theta))$$

We need to compute the gradient of the log-likelihood:

$$\nabla_\theta f_\theta(x_{train}) - \nabla_\theta \log Z(\theta)$$

$$Z(\theta) = \int \exp(f_\theta(x))dx$$

Energy-based model

$$\nabla_\theta f_\theta(x_{train}) - \nabla_\theta \log Z(\theta)$$

Gradient of the log-likelihood

Energy-based model

$$Z(\theta) = \int \exp(f_\theta(x))dx$$

$$\nabla_\theta f_\theta(x_{train}) - \nabla_\theta \log Z(\theta)$$

Gradient of the log-likelihood

$$= \nabla_\theta f_\theta(x_{train}) - \frac{\nabla_\theta Z(\theta)}{Z(\theta)}$$

differentiating the log term

Energy-based model

$$Z(\theta) = \int \exp(f_\theta(x))dx$$

$$\nabla_\theta f_\theta(x_{train}) - \nabla_\theta \log Z(\theta)$$

Gradient of the log-likelihood

$$= \nabla_\theta f_\theta(x_{train}) - \frac{\nabla_\theta Z(\theta)}{Z(\theta)}$$

differentiating the log term

$$= \nabla_\theta f_\theta(x_{train}) - \frac{\int \nabla_\theta \exp(f_\theta(x))dx}{Z(\theta)}$$

using the definition of $Z(\theta)$

Energy-based model

$$Z(\theta) = \int \exp(f_\theta(x))dx$$

$$\nabla_\theta f_\theta(x_{train}) - \nabla_\theta \log Z(\theta)$$

Gradient of the log-likelihood

$$= \nabla_\theta f_\theta(x_{train}) - \frac{\nabla_\theta Z(\theta)}{Z(\theta)}$$

differentiating the log term

$$= \nabla_\theta f_\theta(x_{train}) - \frac{\int \nabla_\theta \exp(f_\theta(x))dx}{Z(\theta)}$$

using the definition of $Z(\theta)$

$$= \nabla_\theta f_\theta(x_{train}) - \frac{\int \exp(f_\theta(x))\, \nabla_\theta f_\theta(x)dx}{Z(\theta)}$$

differentiating the exponent term

Energy-based model

$$Z(\theta) = \int \exp(f_\theta(x))dx$$

$$\nabla_\theta f_\theta(x_{train}) - \nabla_\theta \log Z(\theta)$$

Gradient of the log-likelihood

$$= \nabla_\theta f_\theta(x_{train}) - \frac{\nabla_\theta Z(\theta)}{Z(\theta)}$$

differentiating the log term

$$= \nabla_\theta f_\theta(x_{train}) - \frac{\int \nabla_\theta \exp(f_\theta(x))dx}{Z(\theta)}$$

using the definition of $Z(\theta)$

$$= \nabla_\theta f_\theta(x_{train}) - \frac{\int \exp(f_\theta(x))\,\nabla_\theta f_\theta(x)dx}{Z(\theta)}$$

differentiating the exponent term

$$= \nabla_\theta f_\theta(x_{train}) - \int \frac{\exp(f_\theta(x))}{Z(\theta)} \nabla_\theta f_\theta(x)dx$$

rearranging the terms

Energy-based model

$$Z(\theta) = \int \exp(f_\theta(x))dx$$

$$\nabla_\theta f_\theta(x_{train}) - \nabla_\theta \log Z(\theta)$$ Gradient of the log-likelihood

$$= \nabla_\theta f_\theta(x_{train}) - \frac{\nabla_\theta Z(\theta)}{Z(\theta)}$$ differentiating the log term

$$= \nabla_\theta f_\theta(x_{train}) - \frac{\int \nabla_\theta \exp(f_\theta(x))dx}{Z(\theta)}$$ using the definition of $Z(\theta)$

$$= \nabla_\theta f_\theta(x_{train}) - \frac{\int \exp(f_\theta(x))\, \nabla_\theta f_\theta(x)dx}{Z(\theta)}$$ differentiating the exponent term

$$= \nabla_\theta f_\theta(x_{train}) - \int \boxed{\frac{\exp(f_\theta(x))}{Z(\theta)}} \nabla_\theta f_\theta(x)dx$$ rearranging the terms

$p_\theta(x)$

Energy-based model

$$Z(\theta) = \int \exp(f_\theta(x))dx$$

$$\nabla_\theta f_\theta(x_{train}) - \nabla_\theta \log Z(\theta)$$

Gradient of the log-likelihood

$$= \nabla_\theta f_\theta(x_{train}) - \frac{\nabla_\theta Z(\theta)}{Z(\theta)}$$

differentiating the log term

$$= \nabla_\theta f_\theta(x_{train}) - \frac{\int \nabla_\theta \exp(f_\theta(x))dx}{Z(\theta)}$$

using the definition of $Z(\theta)$

$$= \nabla_\theta f_\theta(x_{train}) - \frac{\int \exp(f_\theta(x))\, \nabla_\theta f_\theta(x)dx}{Z(\theta)}$$

differentiating the exponent term

$$= \nabla_\theta f_\theta(x_{train}) - \int \underbrace{\boxed{\frac{\exp(f_\theta(x))}{Z(\theta)}}}_{p_\theta(x)} \nabla_\theta f_\theta(x)dx$$

rearranging the terms

$$= \nabla_\theta f_\theta(x_{train}) - \mathbb{E}_{x_{sample}}[\nabla_\theta f_\theta(x_{sample})]$$

Energy-based model

Gradient of the log-likelihood

$$\nabla_\theta f_\theta(x_{train}) - \nabla_\theta \log Z(\theta) \quad = \nabla_\theta f_\theta(x_{train}) - \mathbb{E}_{x_{sample}}[\nabla_\theta f_\theta(x_{sample})]$$

Energy-based model

Gradient of the log-likelihood

$$\nabla_\theta f_\theta(x_{train}) - \nabla_\theta \log Z(\theta) \quad = \nabla_\theta f_\theta(x_{train}) - \mathbb{E}_{x_{sample}}[\nabla_\theta f_\theta(x_{sample})]$$

$$\approx \nabla_\theta f_\theta(x_{train}) - \nabla_\theta f_\theta(x_{sample})$$

Energy-based model

Gradient of the log-likelihood

$$\nabla_\theta f_\theta(x_{train}) - \nabla_\theta \log Z(\theta) \quad = \nabla_\theta f_\theta(x_{train}) - \mathbb{E}_{x_{sample}}[\nabla_\theta f_\theta(x_{sample})]$$

$$\approx \nabla_\theta f_\theta(x_{train}) - \nabla_\theta f_\theta(x_{sample})$$

which is a 1-sample Monte Carlo estimator

Energy-based model

Gradient of the log-likelihood

$$\nabla_\theta f_\theta(x_{train}) - \nabla_\theta \log Z(\theta) \quad = \nabla_\theta f_\theta(x_{train}) - \mathbb{E}_{x_{sample}}[\nabla_\theta f_\theta(x_{sample})]$$

$$\approx \nabla_\theta f_\theta(x_{train}) - \nabla_\theta f_\theta(x_{sample})$$

which is a 1-sample Monte Carlo estimator

$$x_{sample} \sim \frac{\exp(f_\theta(x))}{Z(\theta)}$$

Energy-based model

Gradient of the log-likelihood

$$\nabla_\theta f_\theta(x_{train}) - \nabla_\theta \log Z(\theta) \quad = \nabla_\theta f_\theta(x_{train}) - \mathbb{E}_{x_{sample}}[\nabla_\theta f_\theta(x_{sample})]$$

$$\approx \nabla_\theta f_\theta(x_{train}) - \nabla_\theta f_\theta(x_{sample})$$

which is a 1-sample Monte Carlo estimator

$$x_{sample} \sim \frac{\exp(f_\theta(x))}{Z(\theta)}$$

This is an unbiased estimator of a true gradient.

Energy-based model

Contrastive-Divergence method

Gradient of the log-likelihood

$$\nabla_\theta f_\theta(x_{train}) - \nabla_\theta \log Z(\theta) \quad = \nabla_\theta f_\theta(x_{train}) - \mathbb{E}_{x_{sample}}[\nabla_\theta f_\theta(x_{sample})]$$

$$\approx \nabla_\theta f_\theta(x_{train}) - \nabla_\theta f_\theta(x_{sample})$$

which is a 1-sample Monte Carlo estimator

$$x_{sample} \sim \frac{\exp(f_\theta(x))}{Z(\theta)}$$

This is an unbiased estimator of a true gradient.

Energy-based model

Contrastive-Divergence method

Gradient of the log-likelihood

$$\nabla_\theta f_\theta(x_{train}) - \nabla_\theta \log Z(\theta) \quad = \nabla_\theta f_\theta(x_{train}) - \mathbb{E}_{x_{sample}}[\nabla_\theta f_\theta(x_{sample})]$$

$$\approx \nabla_\theta f_\theta(x_{train}) - \nabla_\theta f_\theta(x_{sample})$$

which is a 1-sample Monte Carlo estimator

How to sample?

$$x_{sample} \sim \frac{\exp(f_\theta(x))}{Z(\theta)}$$

This is an unbiased estimator of a true gradient.

Sampling from Energy-based models

How to sample?

Sampling from Energy-based models

How to sample?

Use an iterative approach called **Metropolis-Hastings MCMC**:

Sampling from Energy-based models

How to sample?

Use an iterative approach called **Metropolis-Hastings MCMC**:

- Initialize $x_0 \sim \pi(x)$ randomly at $t = 0$

Sampling from Energy-based models

How to sample?

Use an iterative approach called **Metropolis-Hastings MCMC**:

- Initialize $x_0 \sim \pi(x)$ randomly at $t = 0$
- Repeat for $t = 0,1,2,\cdots,T-1$:

Sampling from Energy-based models

How to sample?

Use an iterative approach called **Metropolis-Hastings MCMC**:

- Initialize $x_0 \sim \pi(x)$ randomly at $t = 0$
- Repeat for $t = 0,1,2,\cdots,T-1$:
    - $x' = x_t + \text{noise}$

Sampling from Energy-based models

How to sample?

Use an iterative approach called **Metropolis-Hastings MCMC**:

- Initialize $x_0 \sim \pi(x)$ randomly at $t = 0$
- Repeat for $t = 0,1,2,\cdots,T-1$:
    - $x' = x_t + \text{noise}$
    - If $f_\theta(x') > f_\theta(x_t)$ let $x_{t+1} = x'$

Sampling from Energy-based models

How to sample?

Use an iterative approach called **Metropolis-Hastings MCMC**:

- Initialize $x_0 \sim \pi(x)$ randomly at $t = 0$
- Repeat for $t = 0,1,2,\cdots,T-1$:
  - $x' = x_t +$ noise
  - If $f_\theta(x') > f_\theta(x_t)$ let $x_{t+1} = x'$
  - Otherwise let $x_{t+1} = x'$ with probability $\exp(f_\theta(x') - f_\theta(x_t))$

occasionally take downhill moves

Sampling from Energy-based models

How to sample?

Use an iterative approach called **Metropolis-Hastings MCMC**:

- Initialize $x_0 \sim \pi(x)$ randomly at $t = 0$
- Repeat for $t = 0,1,2,\cdots,T-1$:
    - $x' = x_t +$ noise
    - If $f_\theta(x') > f_\theta(x_t)$ let $x_{t+1} = x'$
    - Otherwise let $x_{t+1} = x'$ with probability $\exp(f_\theta(x') - f_\theta(x_t))$

occasionally take downhill moves

Works in theory, but can take very long to converge.

Using Contrastive Divergence to train an EBM requires sampling even during the training phase, not just the inference phase.

Using Contrastive Divergence to train an EBM requires sampling even during the training phase, not just the inference phase.

Even if you have EBM trained, generating samples is very expensive

Using Contrastive Divergence to train an EBM requires sampling even during the training phase, not just the inference phase.

Even if you have EBM trained, generating samples is very expensive

Can we do better?

Unadjusted Langevin MCMC

Sampling from EBMs: Unadjusted Langevin MCMC

Sampling from EBMs: Unadjusted Langevin MCMC

- Initialize x_0 randomly at $t = 0$

Sampling from EBMs: Unadjusted Langevin MCMC

- Initialize x_0 randomly at $t = 0$
- Repeat for $t = 0,1,2,\cdots T - 1$:

Sampling from EBMs: Unadjusted Langevin MCMC

- Initialize x_0 randomly at $t = 0$
- Repeat for $t = 0,1,2,\cdots T - 1$:
    - $\epsilon_t \sim \mathcal{N}(0,1)$

Sampling from EBMs: Unadjusted Langevin MCMC

- Initialize x_0 randomly at $t = 0$
- Repeat for $t = 0,1,2,\cdots T-1$:
  - $\epsilon_t \sim \mathcal{N}(0,1)$
  - $x_{t+1} = x_t + \tau \nabla_x \log p_\theta(x)\,|_{x=x_t} + \sqrt{2\tau}\epsilon_t$

Sampling from EBMs: Unadjusted Langevin MCMC

- Initialize x_0 randomly at $t = 0$
- Repeat for $t = 0,1,2,\cdots T-1$:
    - $\epsilon_t \sim \mathcal{N}(0,1)$
    - $x_{t+1} = x_t + \tau \underline{\nabla_x \log p_\theta(x)} |_{x=x_t} + \underline{\sqrt{2\tau}\epsilon_t}$

Sampling from EBMs: Unadjusted Langevin MCMC

- Initialize x_0 randomly at $t = 0$
- Repeat for $t = 0,1,2,\cdots T-1$:
  - $\epsilon_t \sim \mathcal{N}(0,1)$
  - $x_{t+1} = x_t + \tau \underline{\nabla_x \log p_\theta(x)}\,|_{x=x_t} + \underline{\sqrt{2\tau}\epsilon_t}$

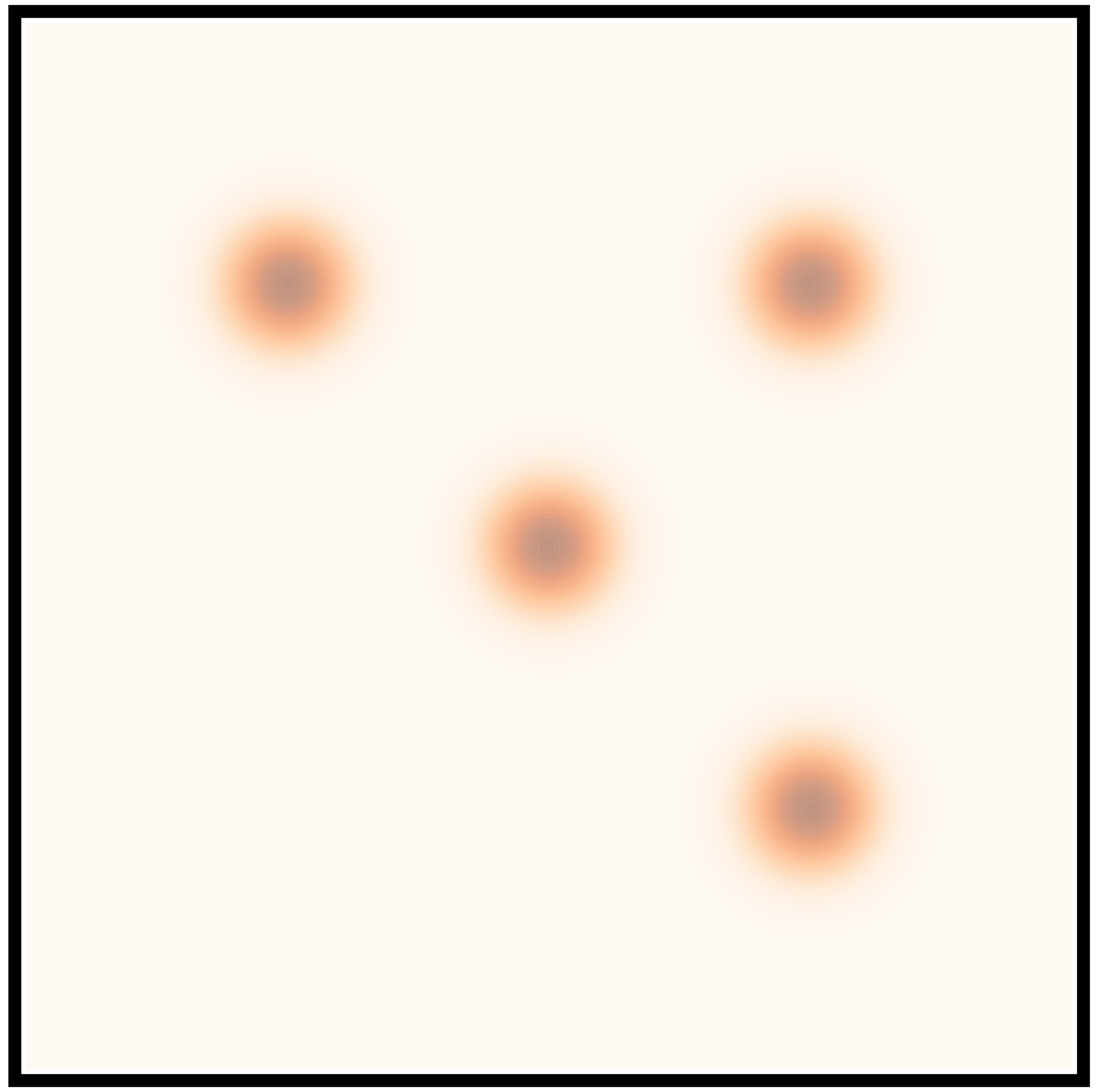
Properties:

Sampling from EBMs: Unadjusted Langevin MCMC

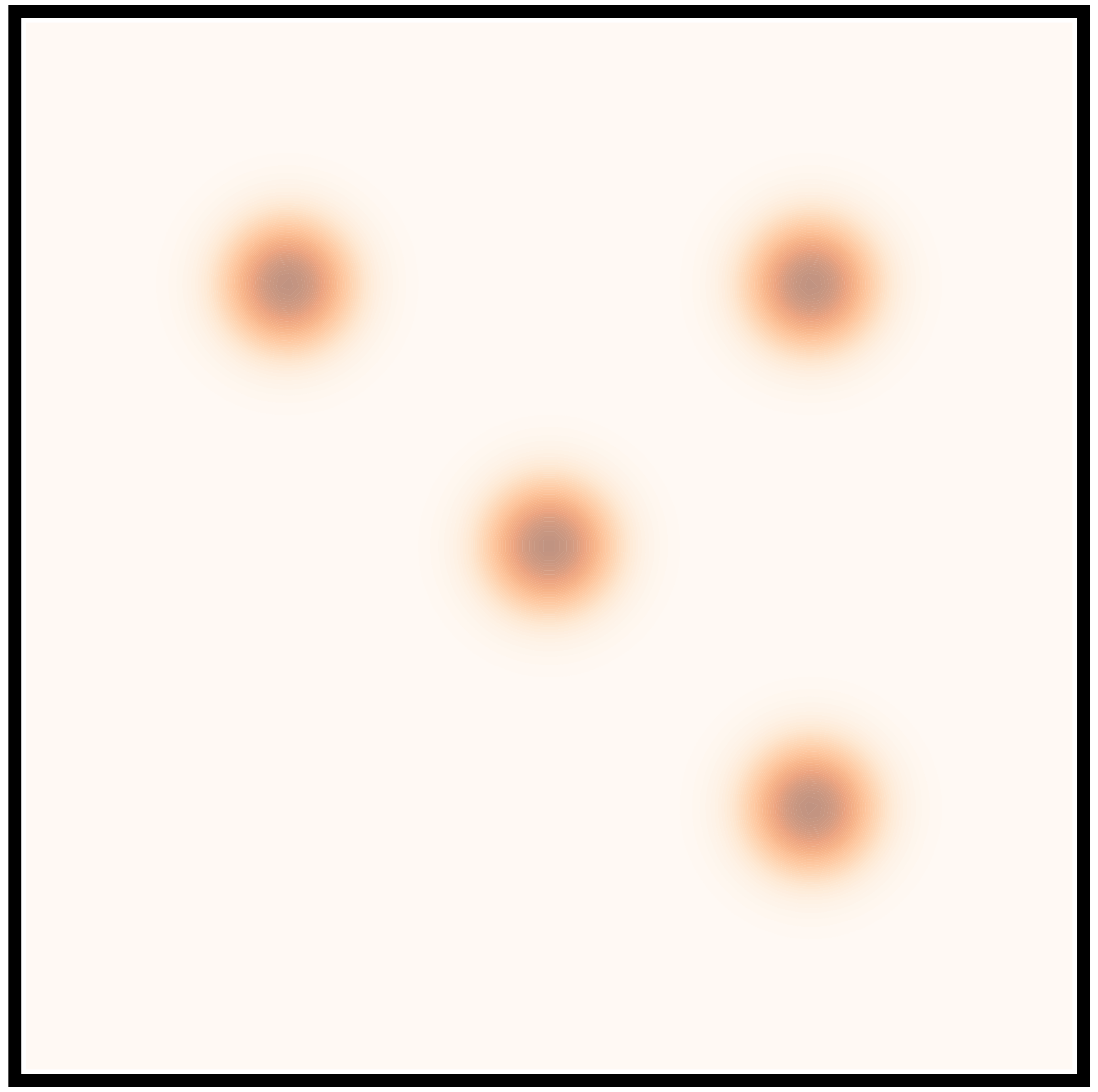
- Initialize x_0 randomly at $t = 0$
- Repeat for $t = 0,1,2,\cdots T-1$:
    - $\epsilon_t \sim \mathcal{N}(0,1)$
    - $x_{t+1} = x_t + \tau \underline{\nabla_x \log p_\theta(x)}\,|_{x=x_t} + \underline{\sqrt{2\tau}\epsilon_t}$

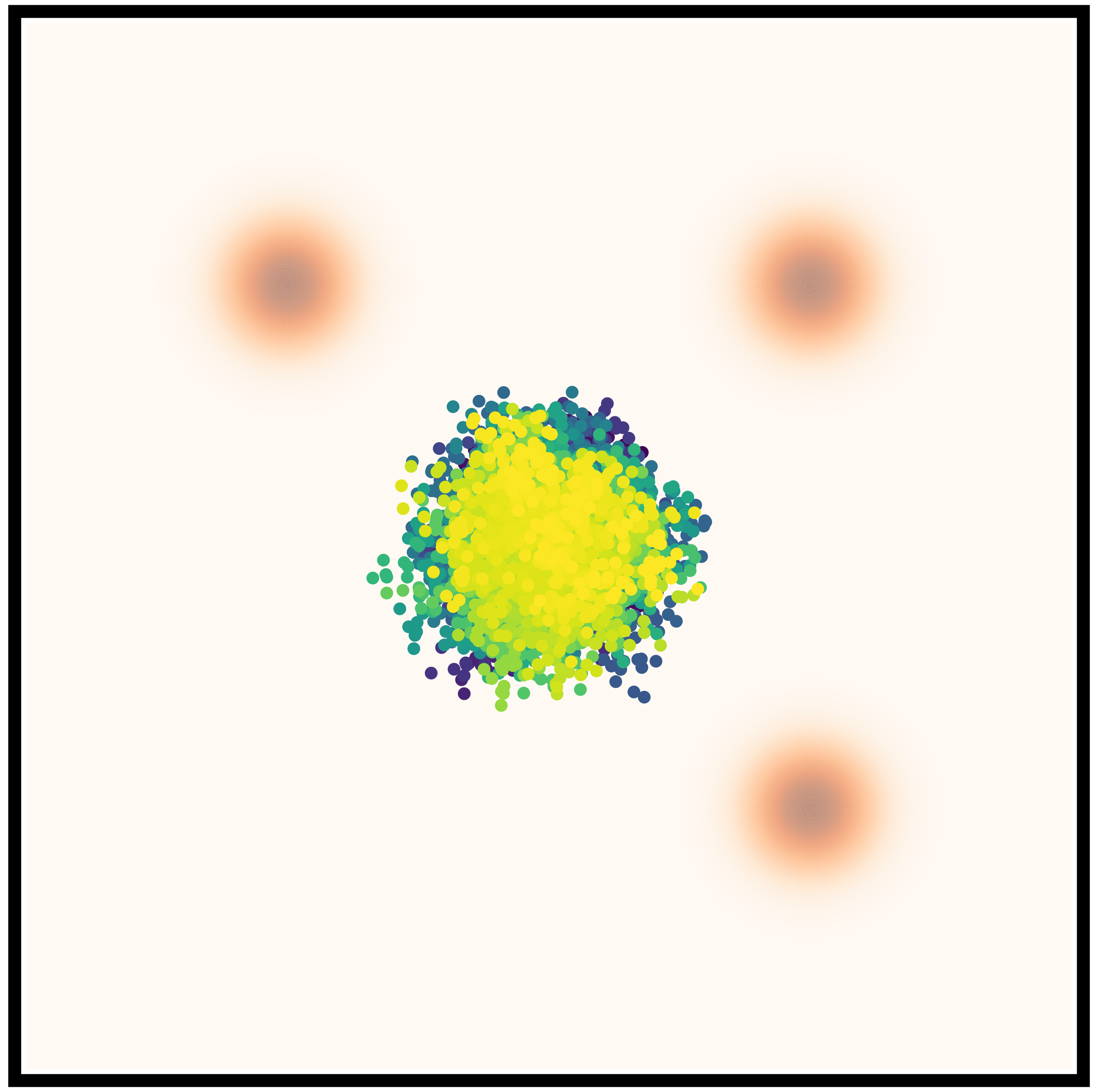
Properties: No rejection involved but x_t converges to a sample from $p_\theta(x)$ when $T \to \infty$ and $\tau \to 0$

Unadjusted Langevin dynamics: Examples



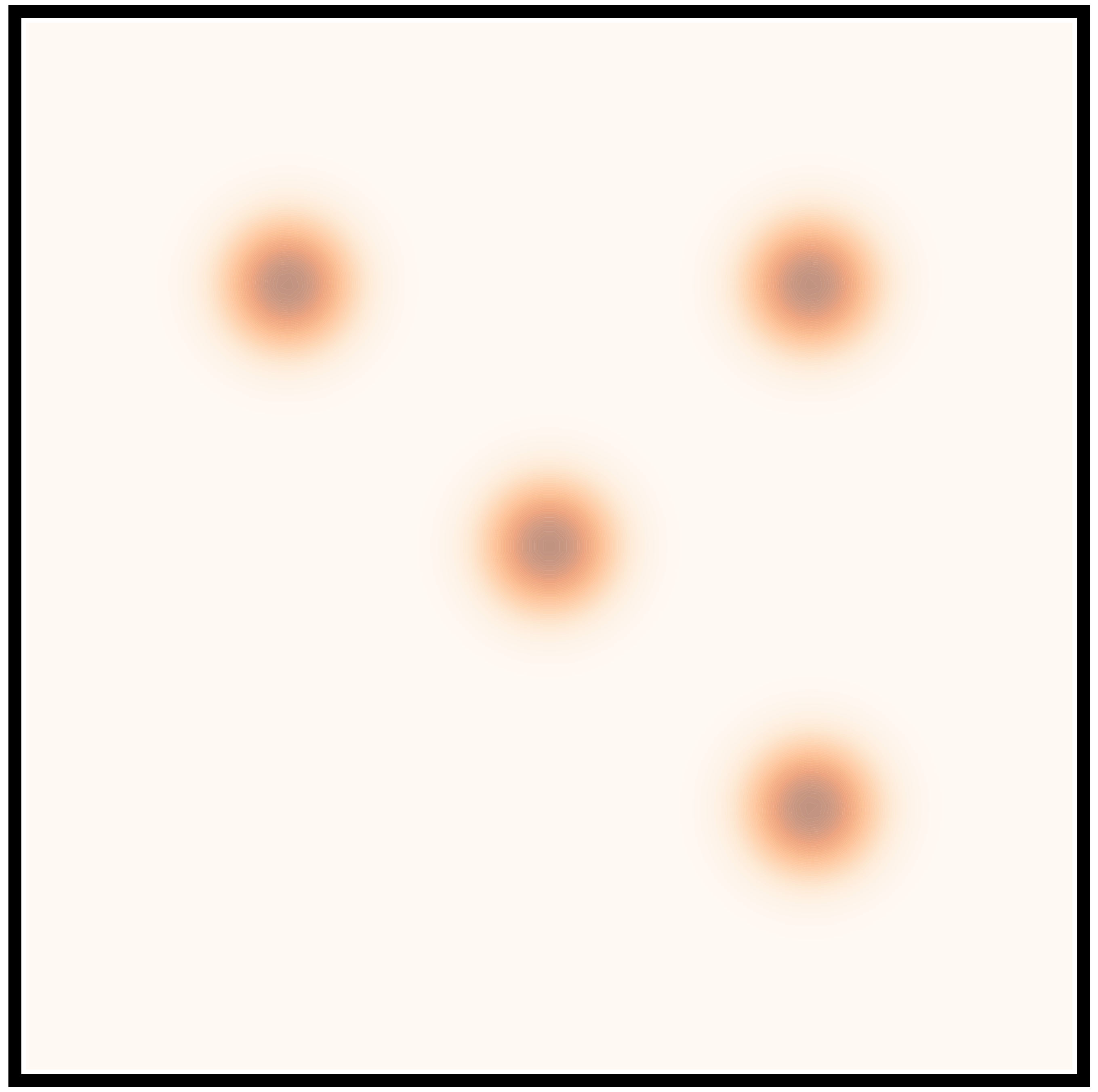
Unadjusted Langevin dynamics: Examples

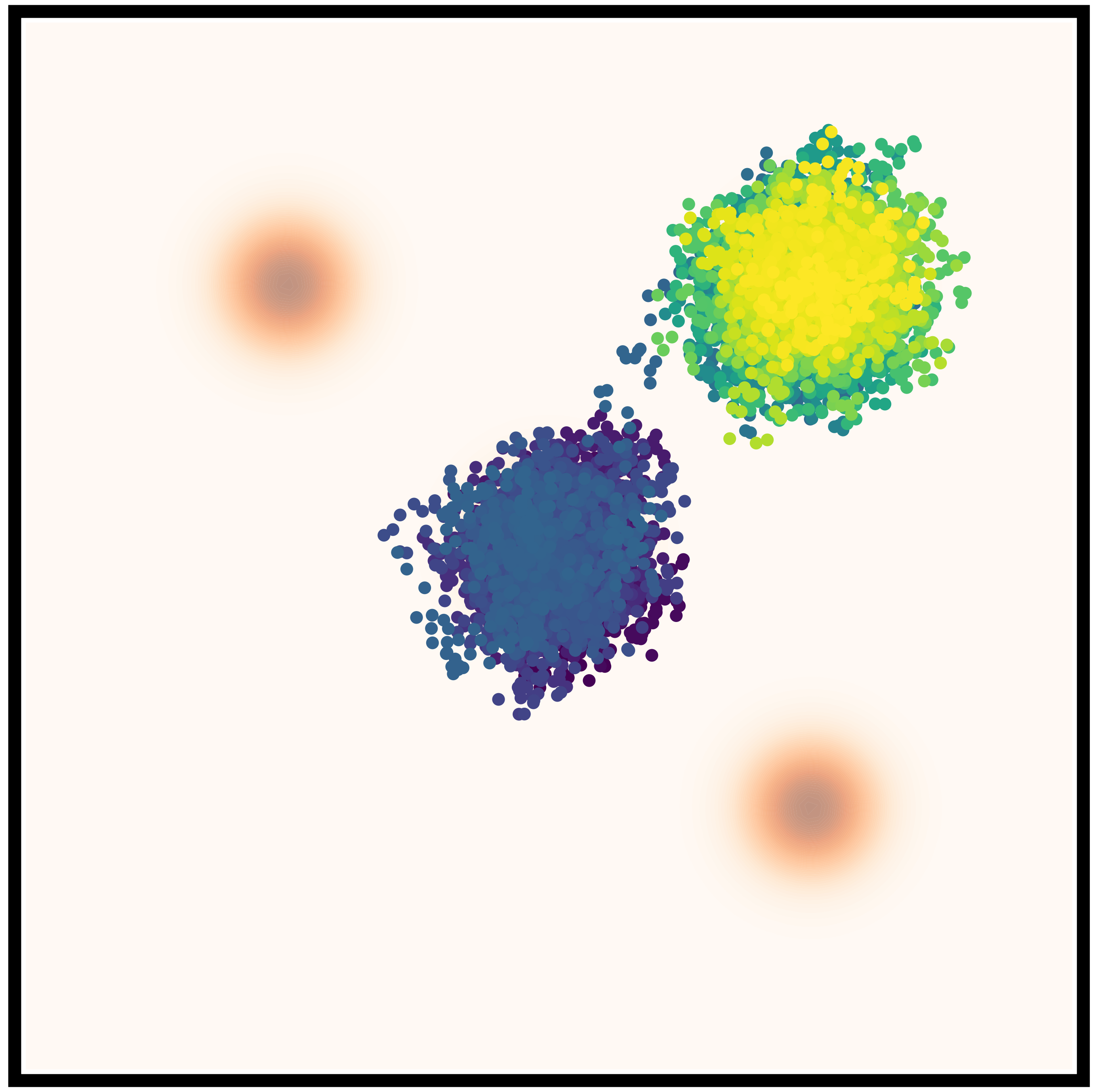




Step size τ=0.1

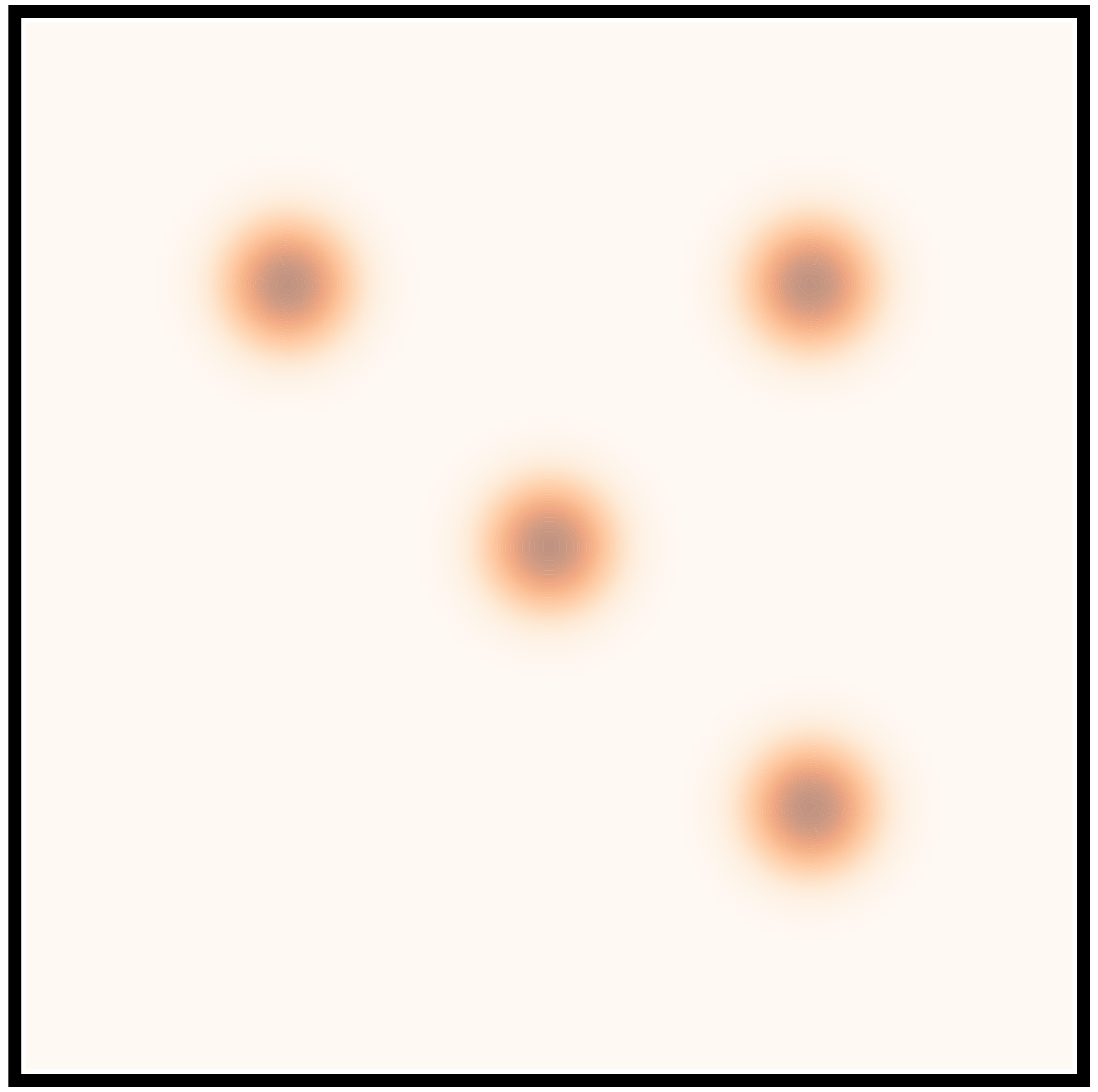
Unadjusted Langevin dynamics: Examples

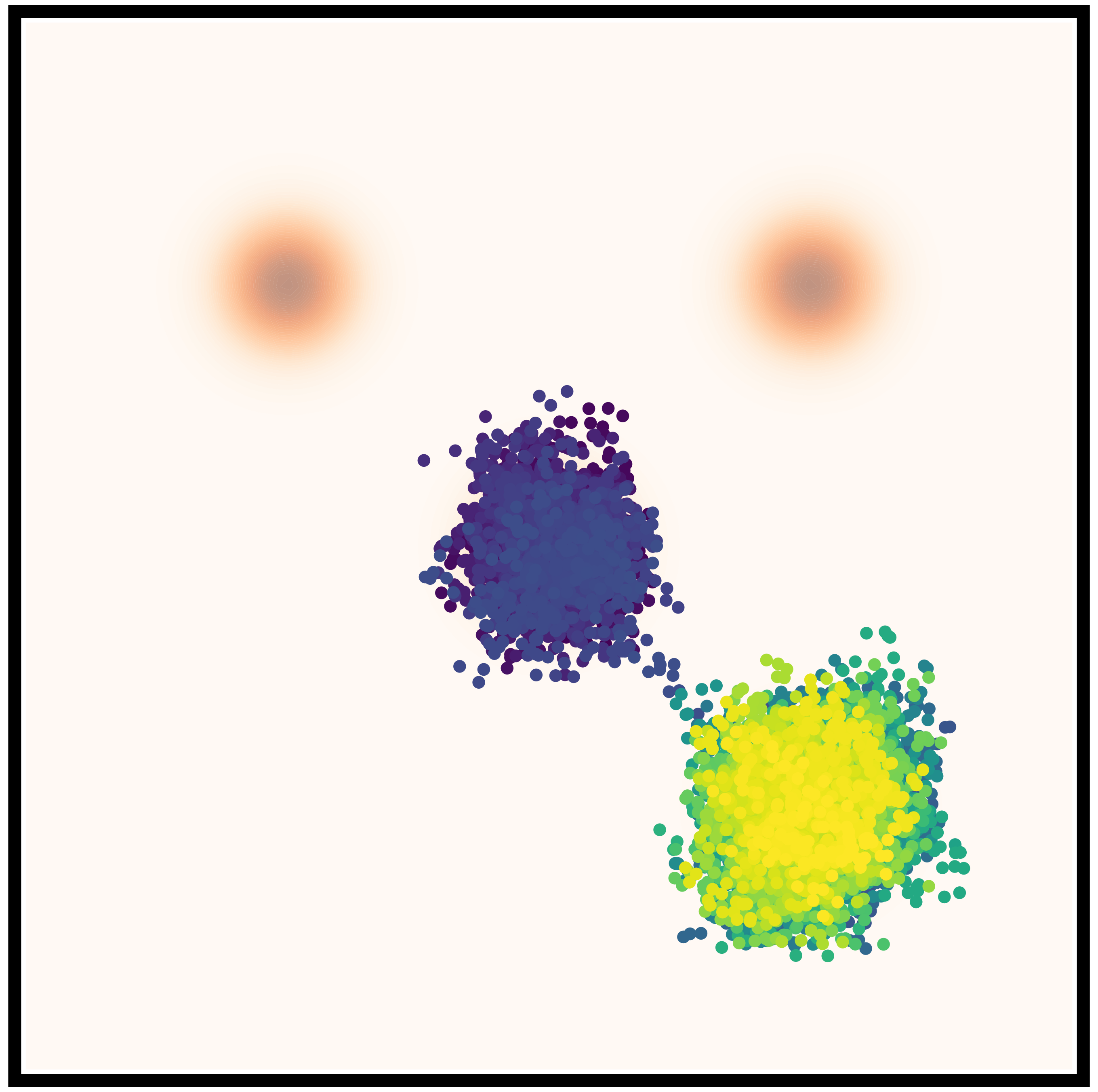




Step size τ=0.1

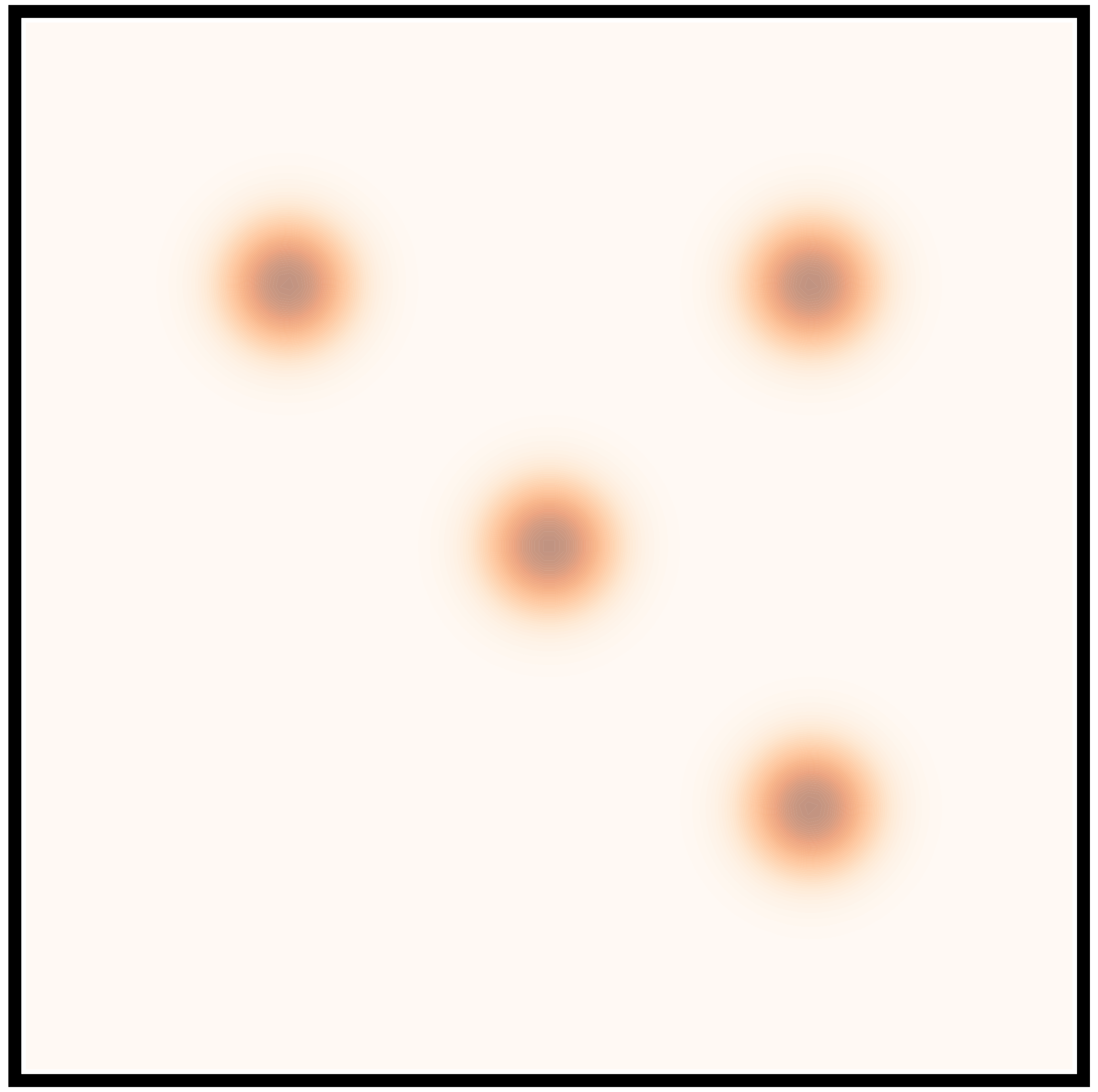
Unadjusted Langevin dynamics: Examples

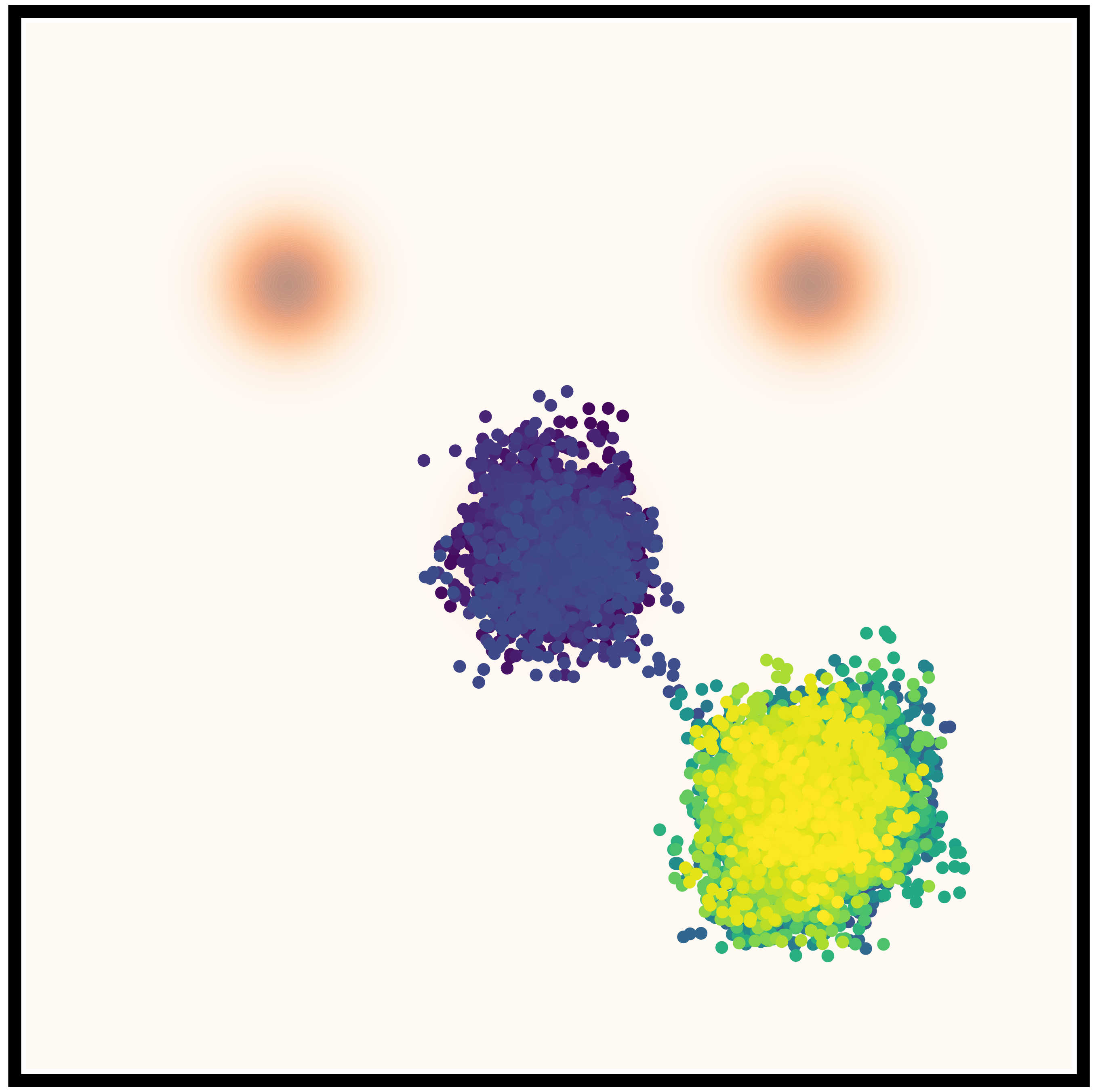




Step size τ=0.1

Unadjusted Langevin dynamics: Examples

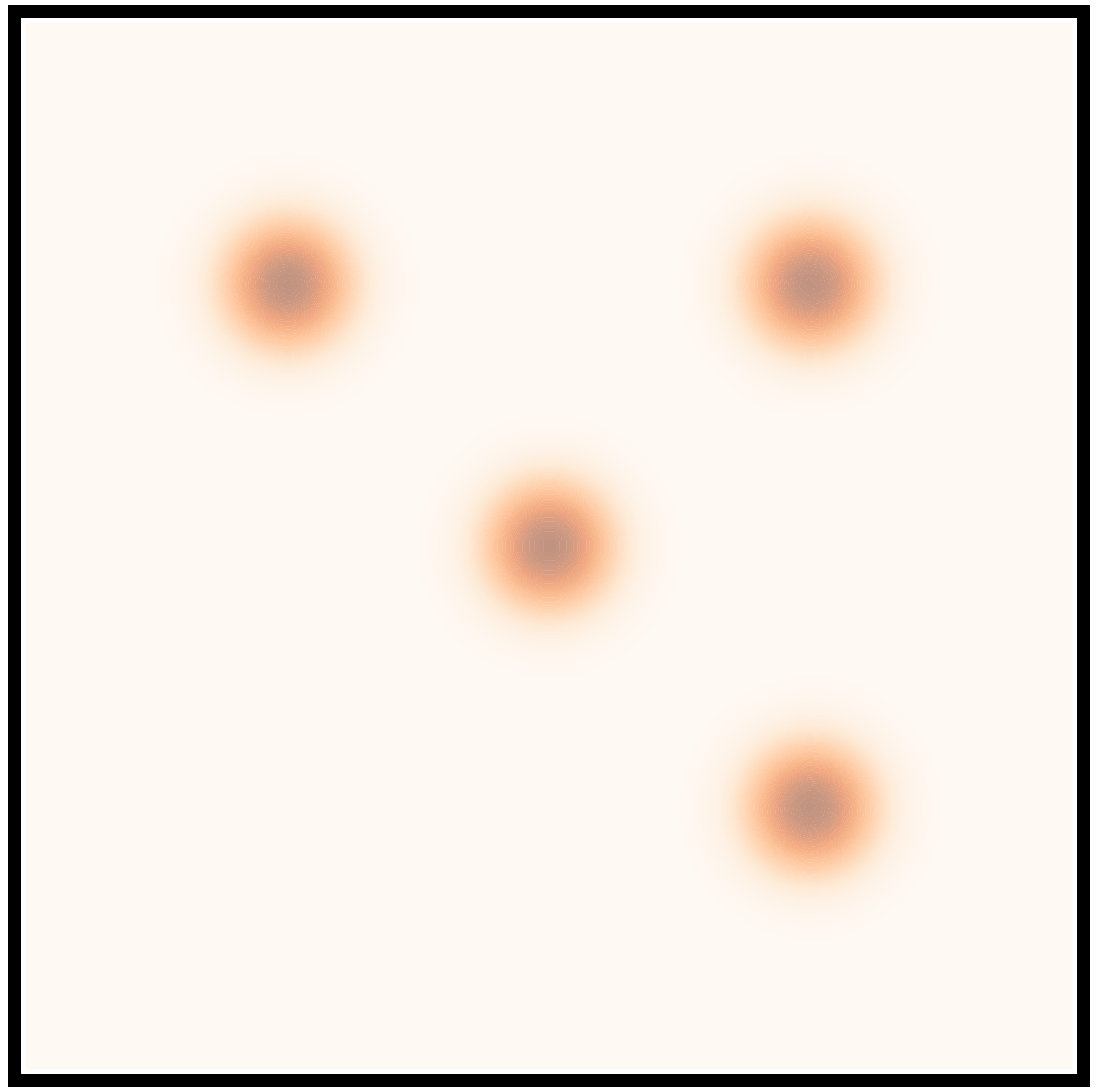


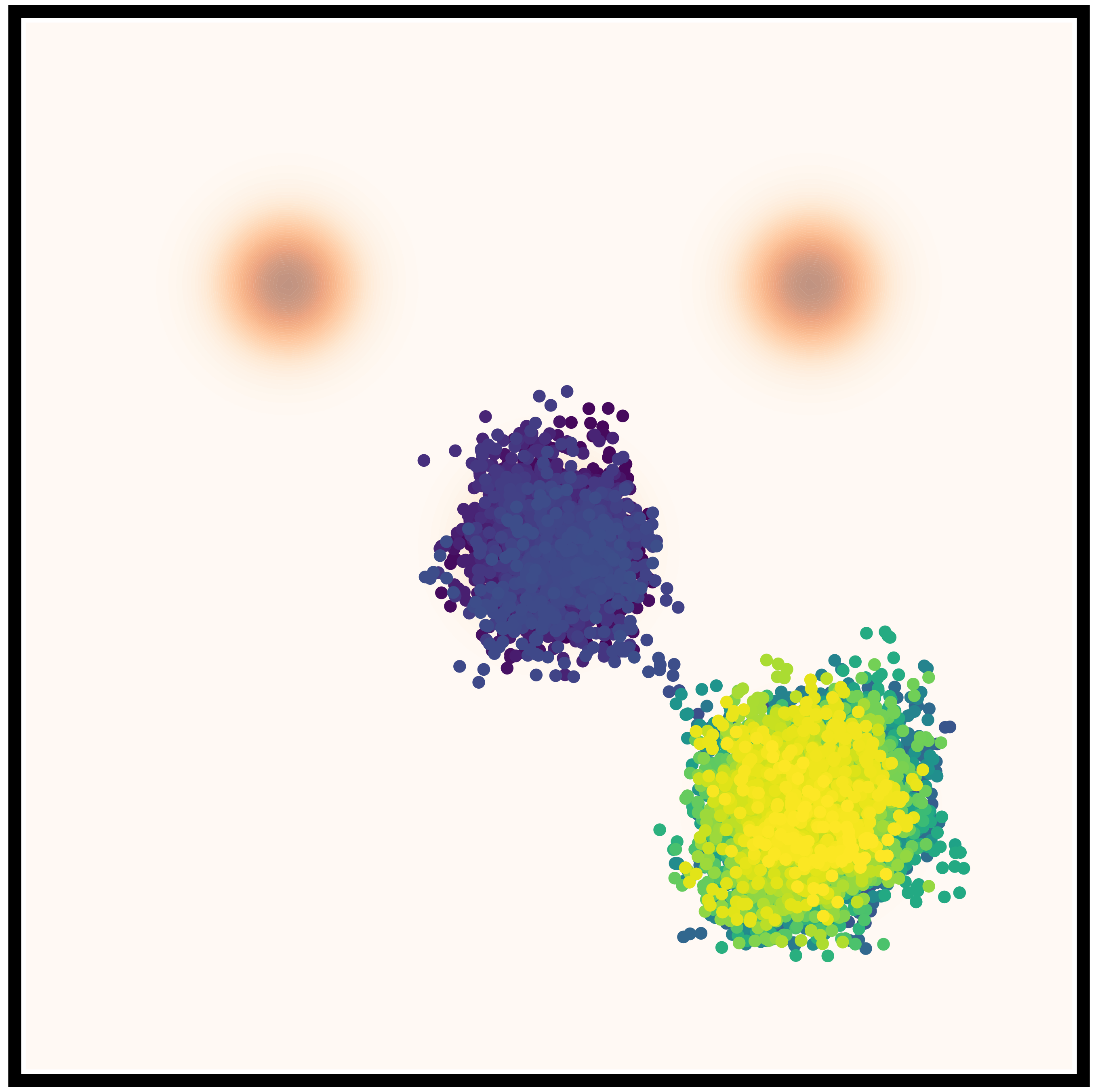


Step size τ=0.1

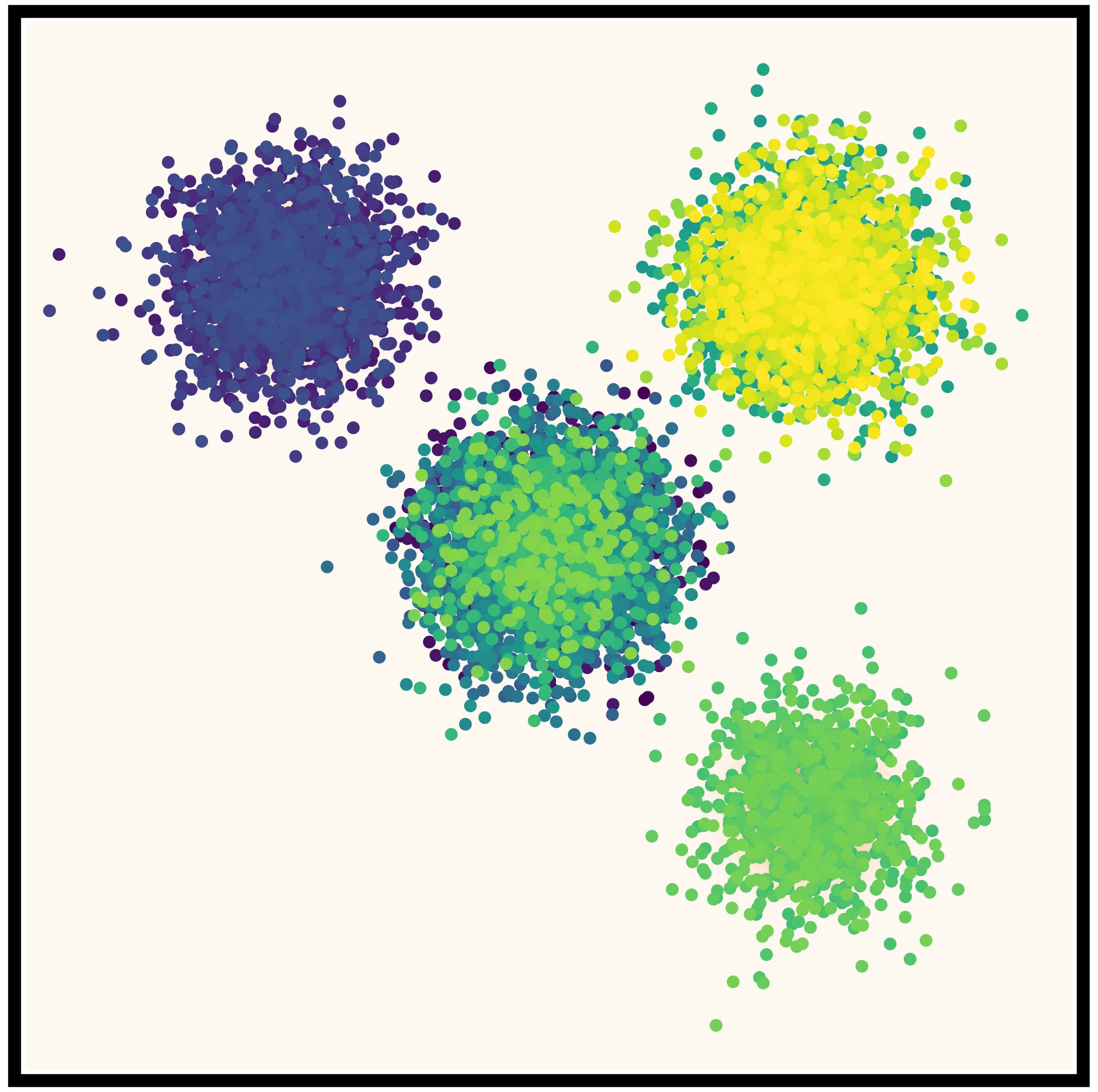
Step size τ=1

Unadjusted Langevin dynamics: Examples





Step size τ=0.1



Step size τ=1

Sampling from EBMs: Unadjusted Langevin MCMC

- Initialize x_0 randomly at $t = 0$
- Repeat for $t = 0,1,2,\cdots T-1$:
    - $\epsilon_t \sim \mathcal{N}(0,1)$
    - $x_{t+1} = x_t + \tau \nabla_x \log p_\theta(x) |_{x=x_t} + \sqrt{2\tau}\epsilon_t$

Sampling from EBMs: Unadjusted Langevin MCMC

- Initialize x_0 randomly at $t = 0$
- Repeat for $t = 0,1,2,\cdots T - 1$:
  - $\epsilon_t \sim \mathcal{N}(0,1)$
  - $x_{t+1} = x_t + \tau \underline{\nabla_x \log p_\theta(x)} \,|_{x=x_t} + \sqrt{2\tau}\epsilon_t$

EBMs: Computing the gradient of log-likelihood

- $x_{t+1} = x_t + \tau \nabla_x \log p_\theta(x) \,|_{x=x_t} + \sqrt{2\tau}\epsilon_t$

$$\log p_\theta(x) = f_\theta(x) - \log Z(\theta)$$

EBMs: Computing the gradient of log-likelihood

- $x_{t+1} = x_t + \tau \nabla_x \log p_\theta(x) \,|_{x=x_t} + \sqrt{2\tau}\epsilon_t$

$$p_\theta(x) = \frac{1}{Z(\theta)} \exp(f_\theta(x))$$

$$Z(\theta) = \int \exp(f_\theta(x))dx$$

$$\log p_\theta(x) = f_\theta(x) - \log Z(\theta)$$

EBMs: Computing the gradient of log-likelihood

- $x_{t+1} = x_t + \tau \nabla_x \log p_\theta(x) \,|_{x=x_t} + \sqrt{2\tau}\epsilon_t$

$$p_\theta(x) = \frac{1}{Z(\theta)} \exp(f_\theta(x))$$

$$Z(\theta) = \int \exp(f_\theta(x))dx$$

$$\log p_\theta(x) = f_\theta(x) - \log Z(\theta)$$

$$\nabla_x \log p_\theta(x) = \nabla_x (f_\theta(x) - \log Z(\theta))$$

EBMs: Computing the gradient of log-likelihood

- $x_{t+1} = x_t + \tau \nabla_x \log p_\theta(x)\,|_{x=x_t} + \sqrt{2\tau}\epsilon_t$

$$p_\theta(x) = \frac{1}{Z(\theta)}\exp(f_\theta(x))$$

$$\log p_\theta(x) = f_\theta(x) - \log Z(\theta)$$

$$Z(\theta) = \int \exp(f_\theta(x))dx$$

$$\nabla_x \log p_\theta(x) = \nabla_x (f_\theta(x) - \log Z(\theta))$$

Gradient is wrt x and not θ

EBMs: Computing the gradient of log-likelihood

- $x_{t+1} = x_t + \tau \nabla_x \log p_\theta(x) \,|_{x=x_t} + \sqrt{2\tau}\epsilon_t$

$$p_\theta(x) = \frac{1}{Z(\theta)} \exp(f_\theta(x))$$

$$Z(\theta) = \int \exp(f_\theta(x)) dx$$

$$\log p_\theta(x) = f_\theta(x) - \log Z(\theta)$$

$$\nabla_x \log p_\theta(x) = \nabla_x (f_\theta(x) - \log Z(\theta))$$

Gradient is wrt x and not θ

$$\nabla_x \log p_\theta(x) = \nabla_x f_\theta(x) - \nabla_x \log Z(\theta)$$

EBMs: Computing the gradient of log-likelihood

- $x_{t+1} = x_t + \tau \nabla_x \log p_\theta(x) \,|_{x=x_t} + \sqrt{2\tau}\epsilon_t$

$$p_\theta(x) = \frac{1}{Z(\theta)}\exp(f_\theta(x))$$

$$\log p_\theta(x) = f_\theta(x) - \log Z(\theta)$$

$$Z(\theta) = \int \exp(f_\theta(x))dx$$

$$\nabla_x \log p_\theta(x) = \nabla_x (f_\theta(x) - \log Z(\theta))$$

Gradient is wrt x and not θ

$$\nabla_x \log p_\theta(x) = \nabla_x f_\theta(x) - \nabla_x \log Z(\theta)$$

is zero

EBMs: Computing the gradient of log-likelihood

- $x_{t+1} = x_t + \tau \nabla_x \log p_\theta(x) \,|_{x=x_t} + \sqrt{2\tau}\epsilon_t$

$$p_\theta(x) = \frac{1}{Z(\theta)} \exp(f_\theta(x))$$

$$Z(\theta) = \int \exp(f_\theta(x)) dx$$

$$\log p_\theta(x) = f_\theta(x) - \log Z(\theta)$$

$$\nabla_x \log p_\theta(x) = \nabla_x (f_\theta(x) - \log Z(\theta))$$

Gradient is wrt x and not θ

$$\nabla_x \log p_\theta(x) = \nabla_x f_\theta(x) - \nabla_x \log Z(\theta)$$

is zero

$$\nabla_x \log p_\theta(x) = \nabla_x f_\theta(x)$$

Pros & Cons of unadjusted Langevin MCMC

- In practice, the number of steps are lesser than MH approach

Pros & Cons of unadjusted Langevin MCMC

- In practice, the number of steps are lesser than MH approach
- However, convergence slows down as dimensionality grows

Can we train EBMs without sampling?

From VAEs to Diffusion models

Variational Autoencoders (VAEs)

Energy-based models (EBMs)

MCMC methods for EBMs

Score-based Generative models (SBGMs)

MCMC methods for SBGMs

SDE-based diffusion models

Score-based EBMs

Energy-based model

$$p_\theta(x) = \frac{\exp(f_\theta(x))}{Z(\theta)}$$

Energy-based model

$$p_\theta(x) = \frac{\exp(f_\theta(x))}{Z(\theta)}$$

$$\log p_\theta(x) = \log \exp(f_\theta(x)) - \log Z(\theta)$$

Energy-based model

$$p_\theta(x) = \frac{\exp(f_\theta(x))}{Z(\theta)}$$

$$\log p_\theta(x) = \log \exp(f_\theta(x)) - \log Z(\theta)$$

$$\log p_\theta(x) = \log f_\theta(x) - \log Z(\theta)$$

Energy-based model

$$p_\theta(x) = \frac{\exp(f_\theta(x))}{Z(\theta)}$$

$$\log p_\theta(x) = \log \exp(f_\theta(x)) - \log Z(\theta)$$

$$\log p_\theta(x) = \log f_\theta(x) - \log Z(\theta)$$

$$s_\theta(x) = \nabla_x \log p_\theta(x)$$ Score function for EBMs

Score function: how is it different from a PDF?

Score function: how is it different from a PDF?

$$s_\theta(x) = \nabla_x \log p_\theta(x)$$

Score function: how is it different from a PDF?

$$s_\theta(x) = \nabla_x \log p_\theta(x)$$

Gaussian distribution:

Score function: how is it different from a PDF?

$$s_\theta(x) = \nabla_x \log p_\theta(x)$$

Gaussian distribution:

$$p_\theta(x) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(x-\mu)}{\sigma^2}\right)$$

Score function: how is it different from a PDF?

$$s_\theta(x) = \nabla_x \log p_\theta(x)$$

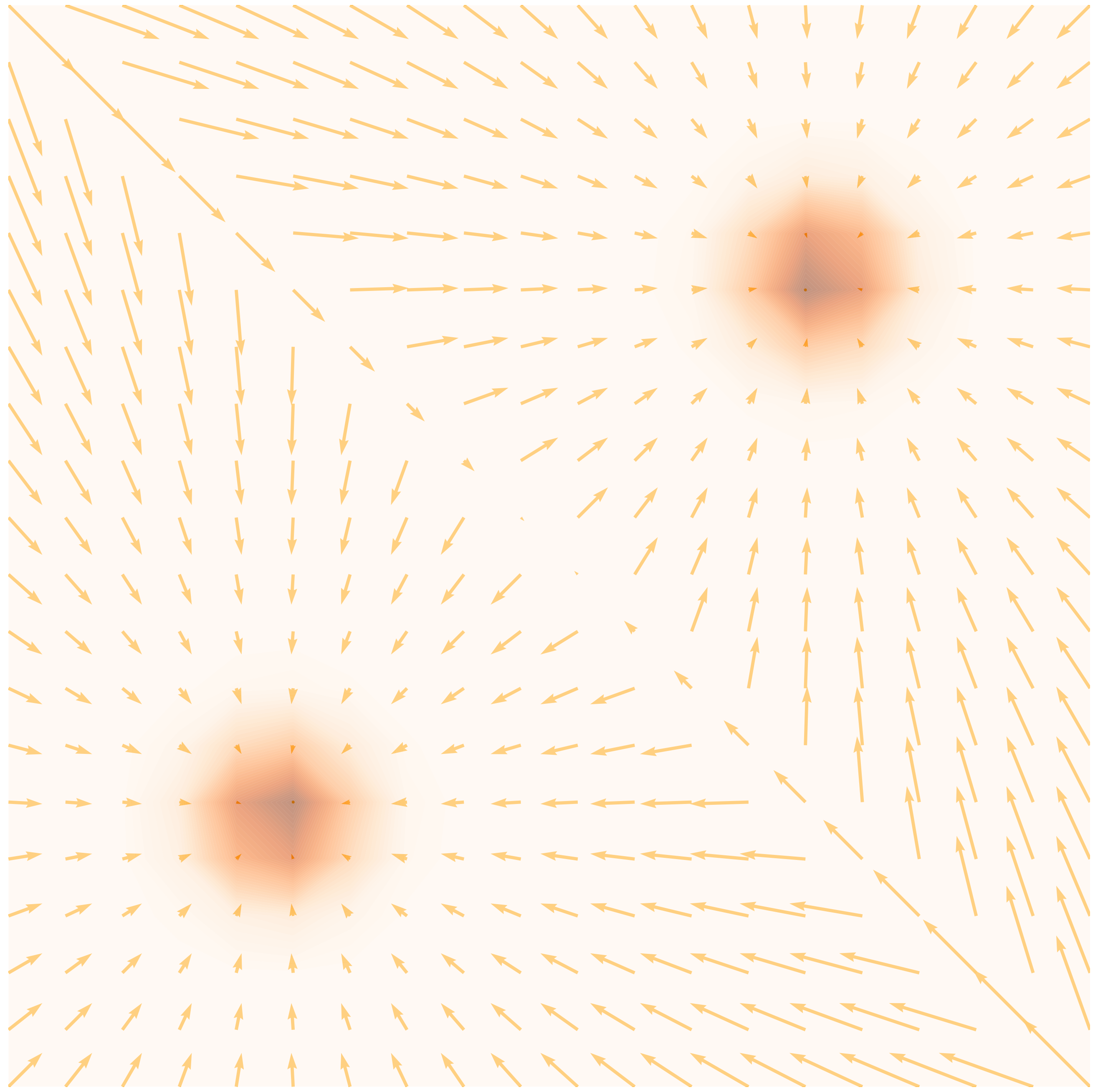
Gaussian distribution:

$$p_\theta(x) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(x-\mu)}{\sigma^2}\right)$$

$$s_\theta(x) = -\frac{(x-\mu)}{\sigma^2}$$

Score function: how is it different from a PDF?

$$s_\theta(x) = \nabla_x \log p_\theta(x)$$



Gaussian distribution:

$$p_\theta(x) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(x-\mu)}{\sigma^2}\right)$$

$$s_\theta(x) = -\frac{(x-\mu)}{\sigma^2}$$

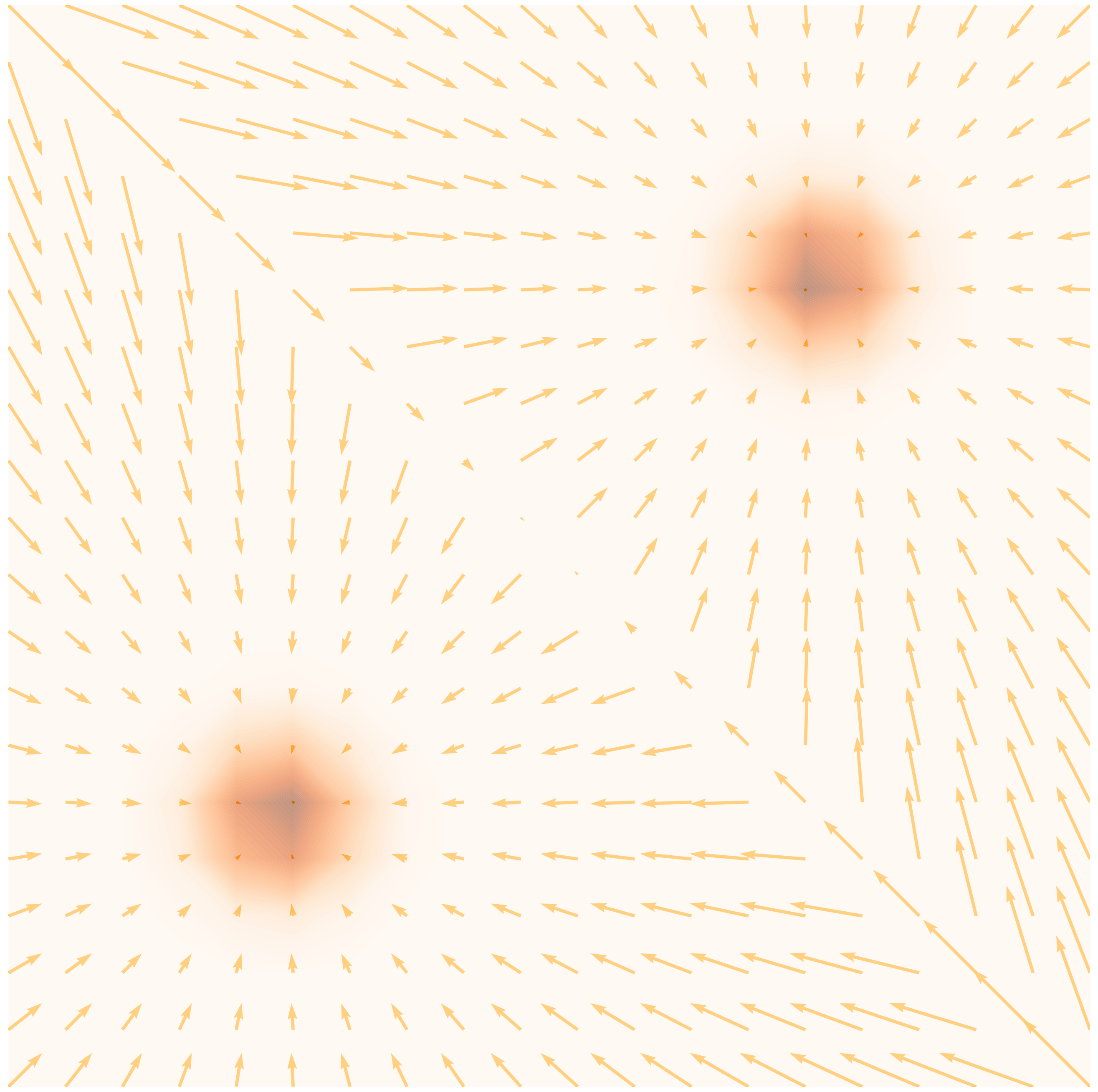
Score function: how is it different from a PDF?

$$s_\theta(x) = \nabla_x \log p_\theta(x)$$

Gaussian distribution:

$$p_\theta(x) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(x-\mu)}{\sigma^2}\right)$$

$$s_\theta(x) = -\frac{(x-\mu)}{\sigma^2}$$



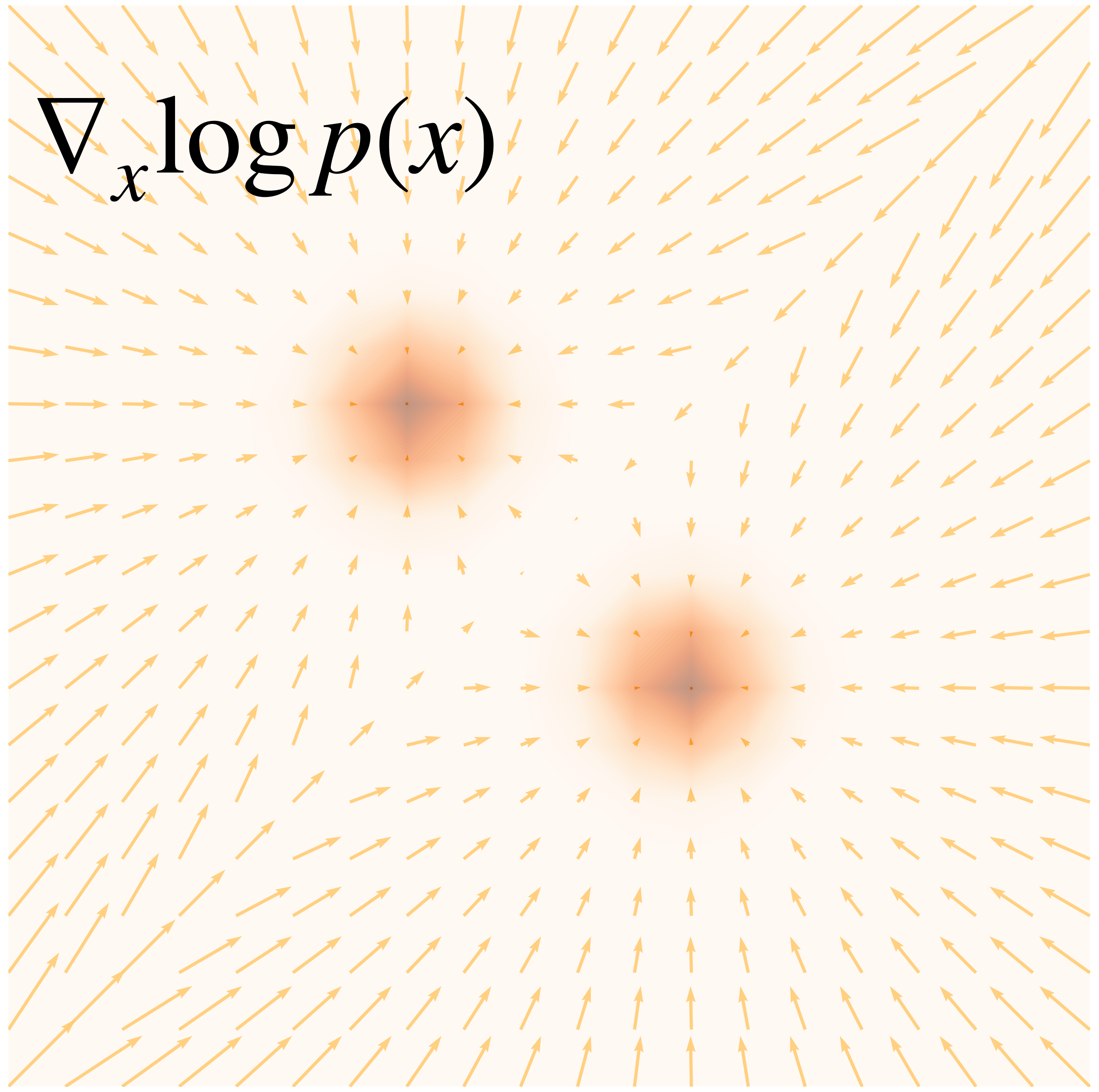
Score: vector field

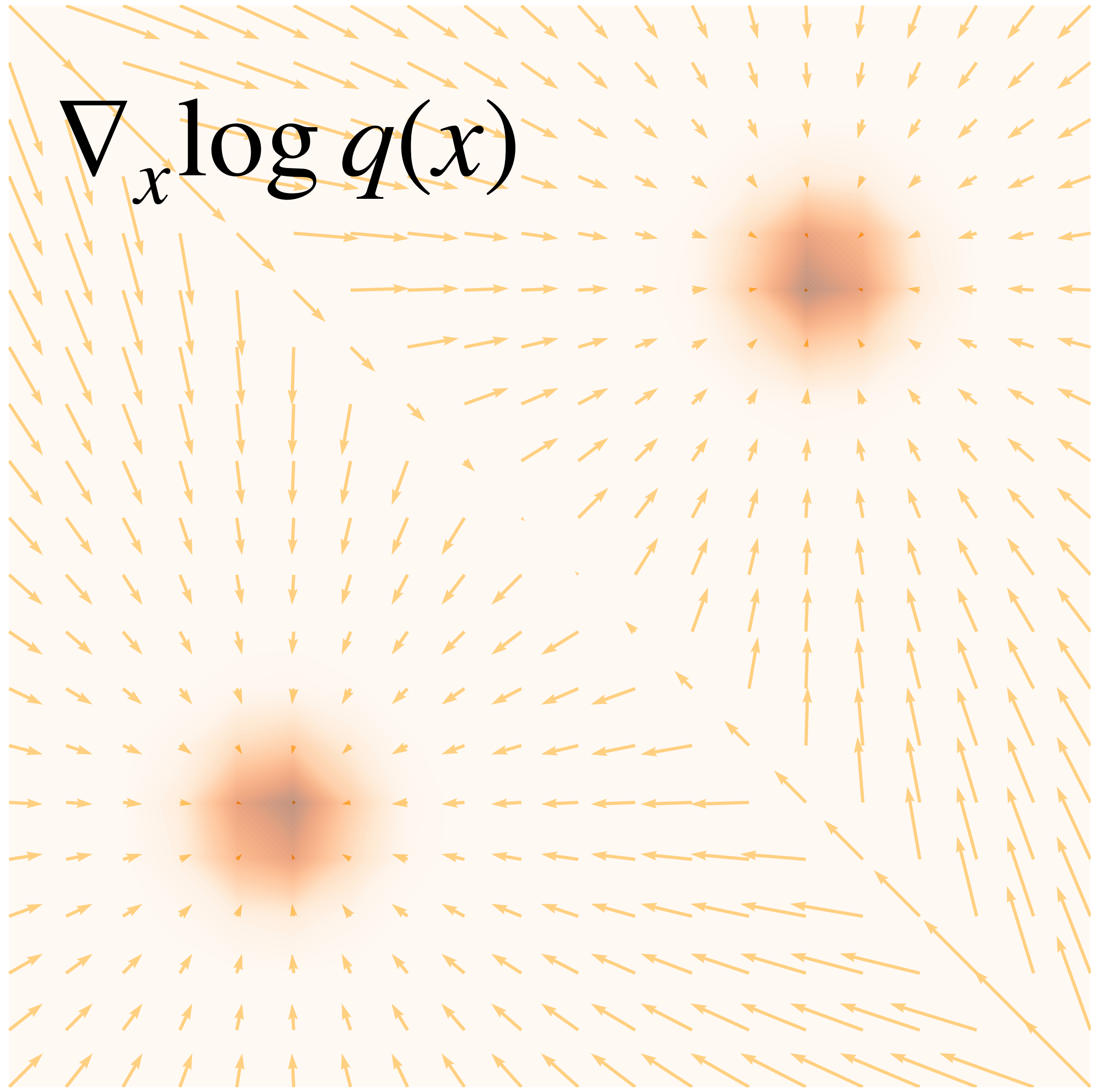
vs

PDF: scalar value

Score matching

$$s_\theta(x) = \nabla_x \log p_\theta(x)$$

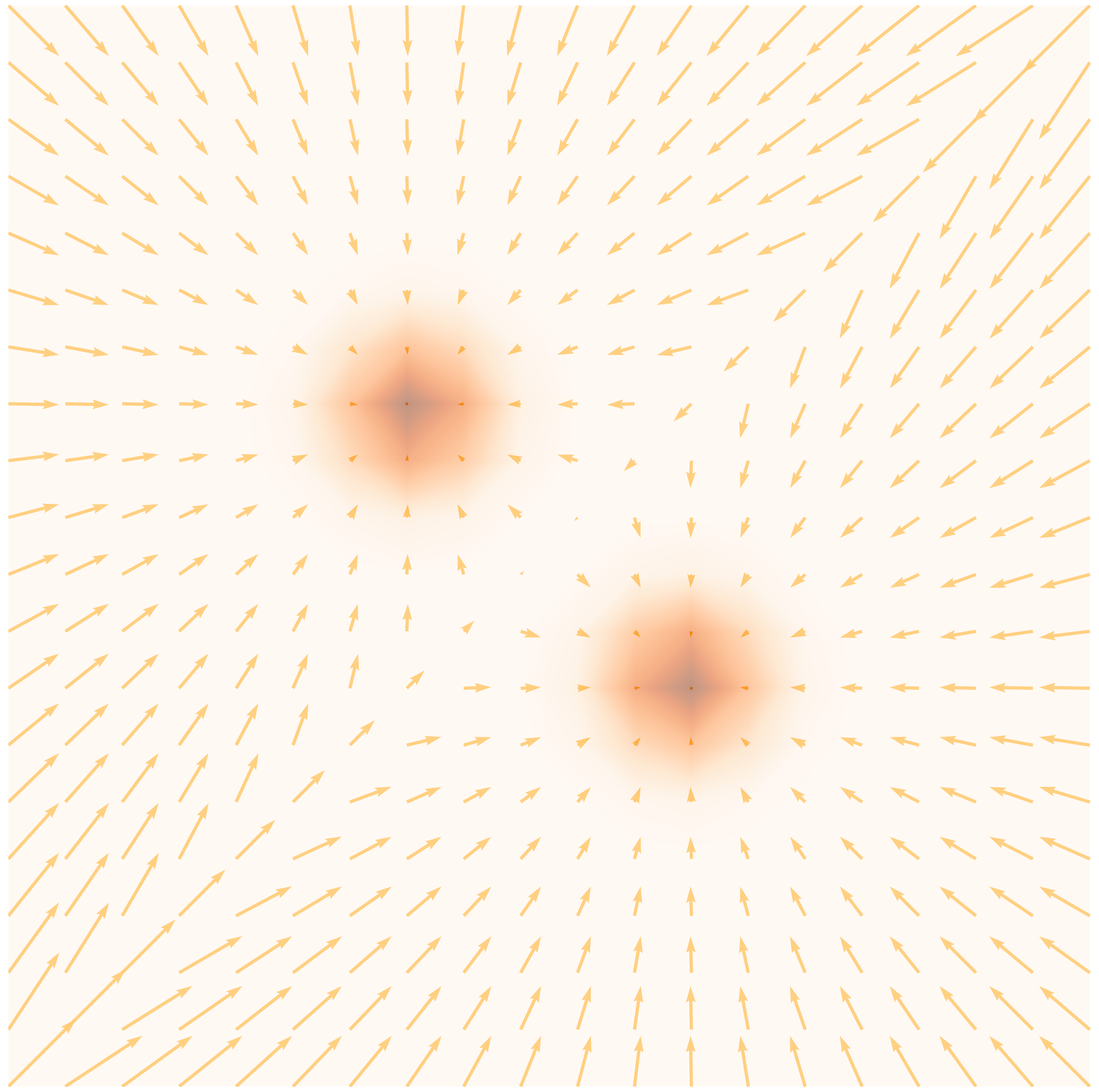




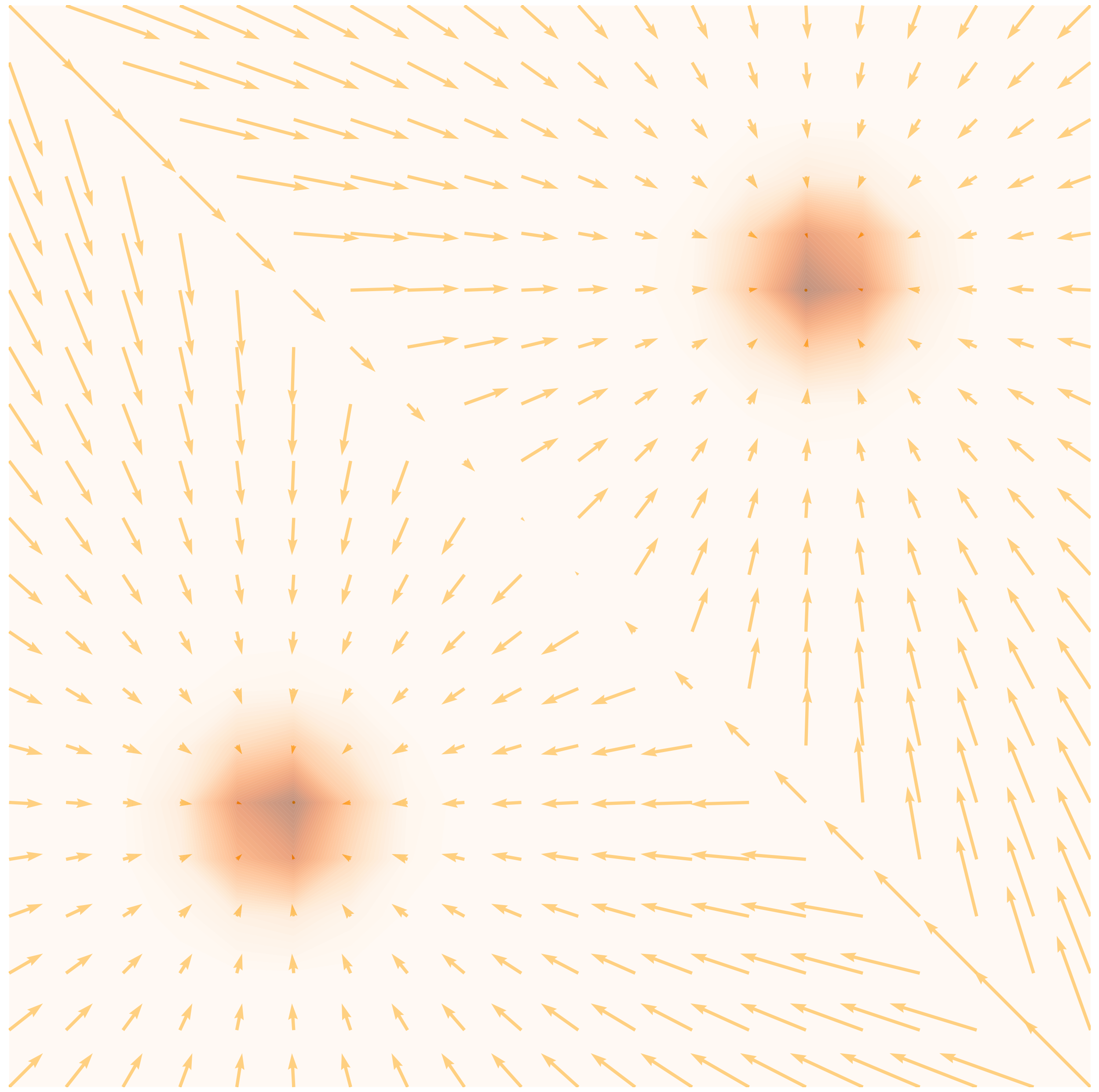
Score matching

$$s_\theta(x) = \nabla_x \log p_\theta(x)$$

$\nabla_x \log p(x)$



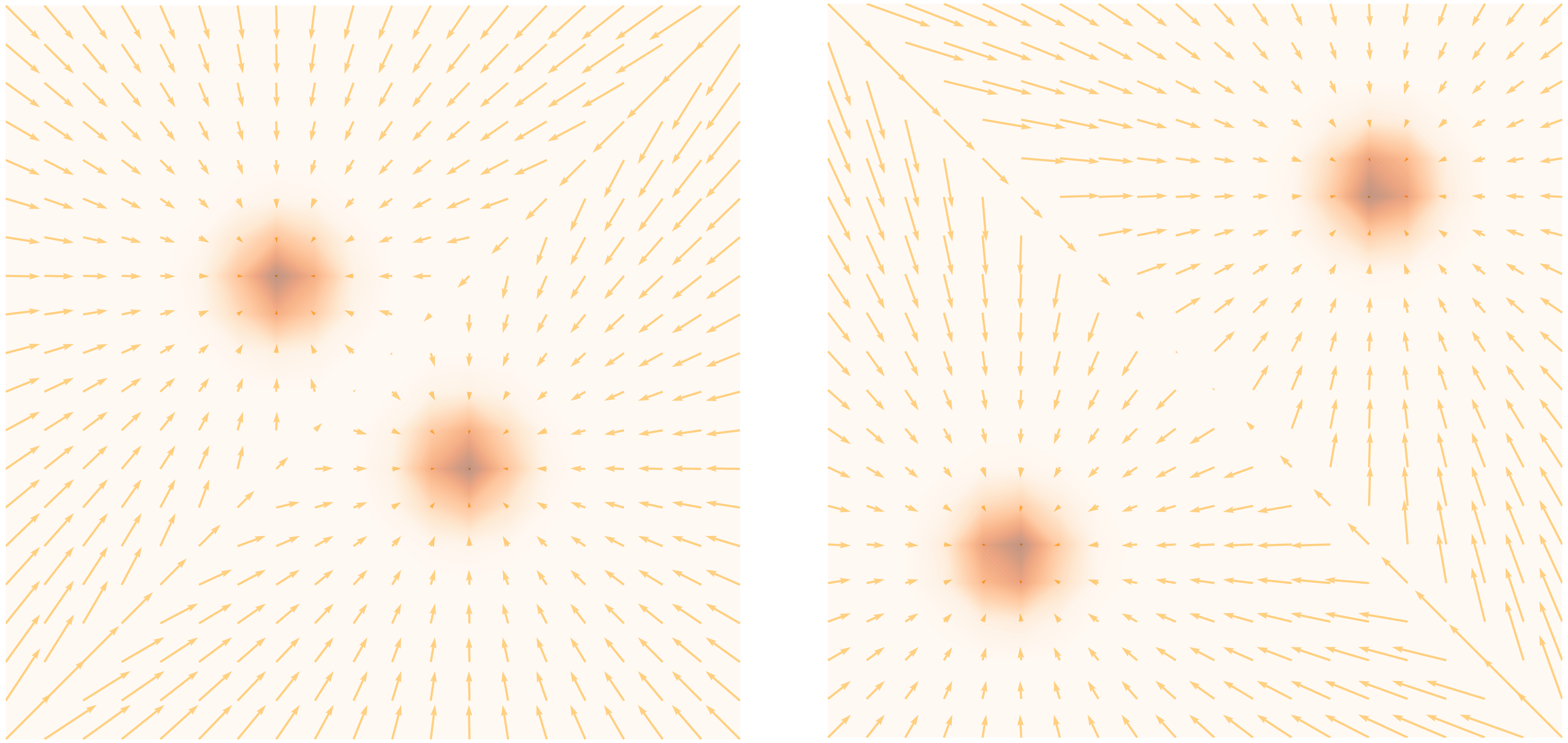
$\nabla_x \log q(x)$



Score matching

$$s_\theta(x) = \nabla_x \log p_\theta(x)$$

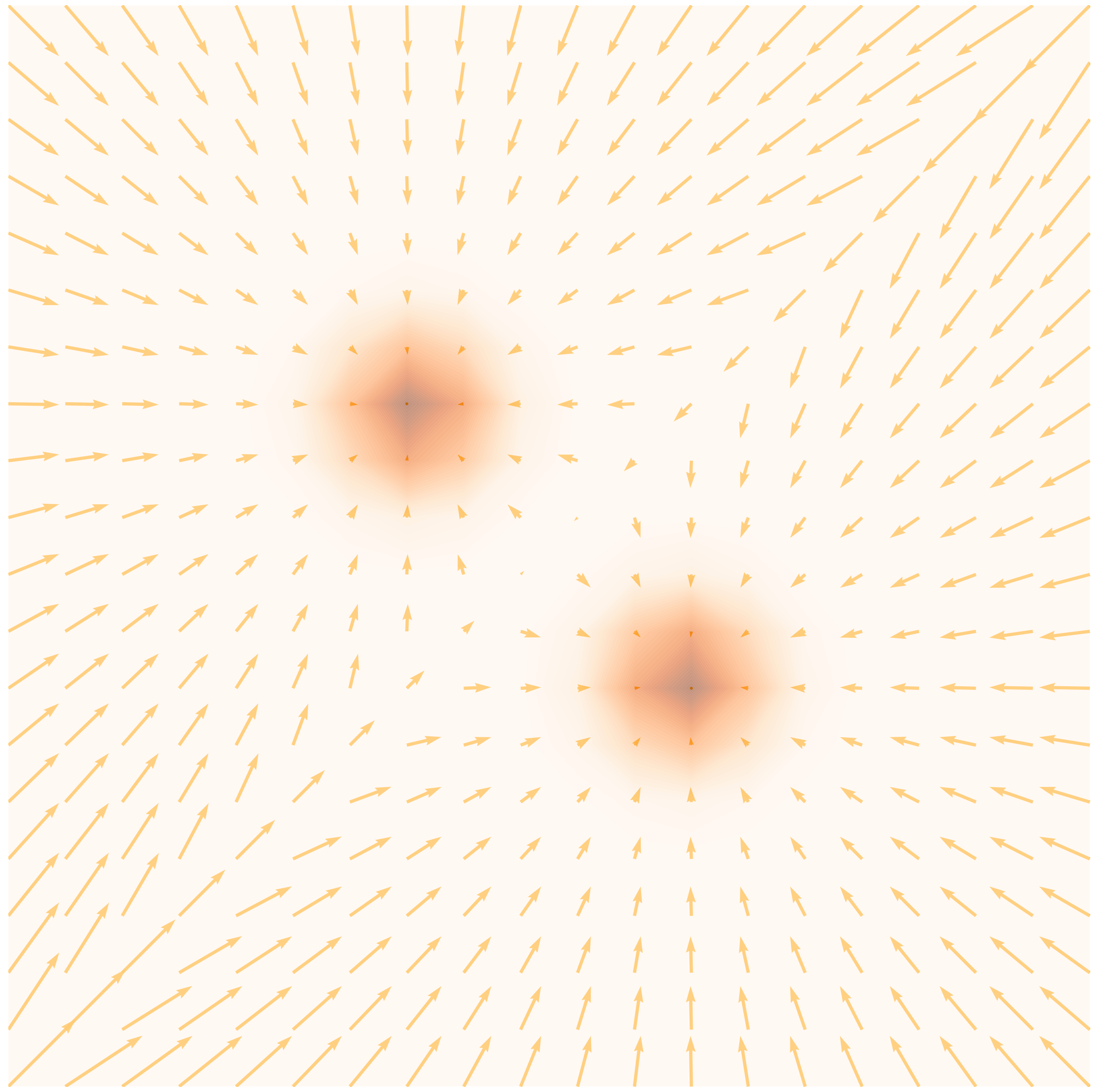
$$D_F(p, q) = \frac{1}{2}\mathbb{E}_{x\sim p}\left[|| \nabla_x \log p(x) - \nabla_x \log q(x) ||_2^2 \right]$$

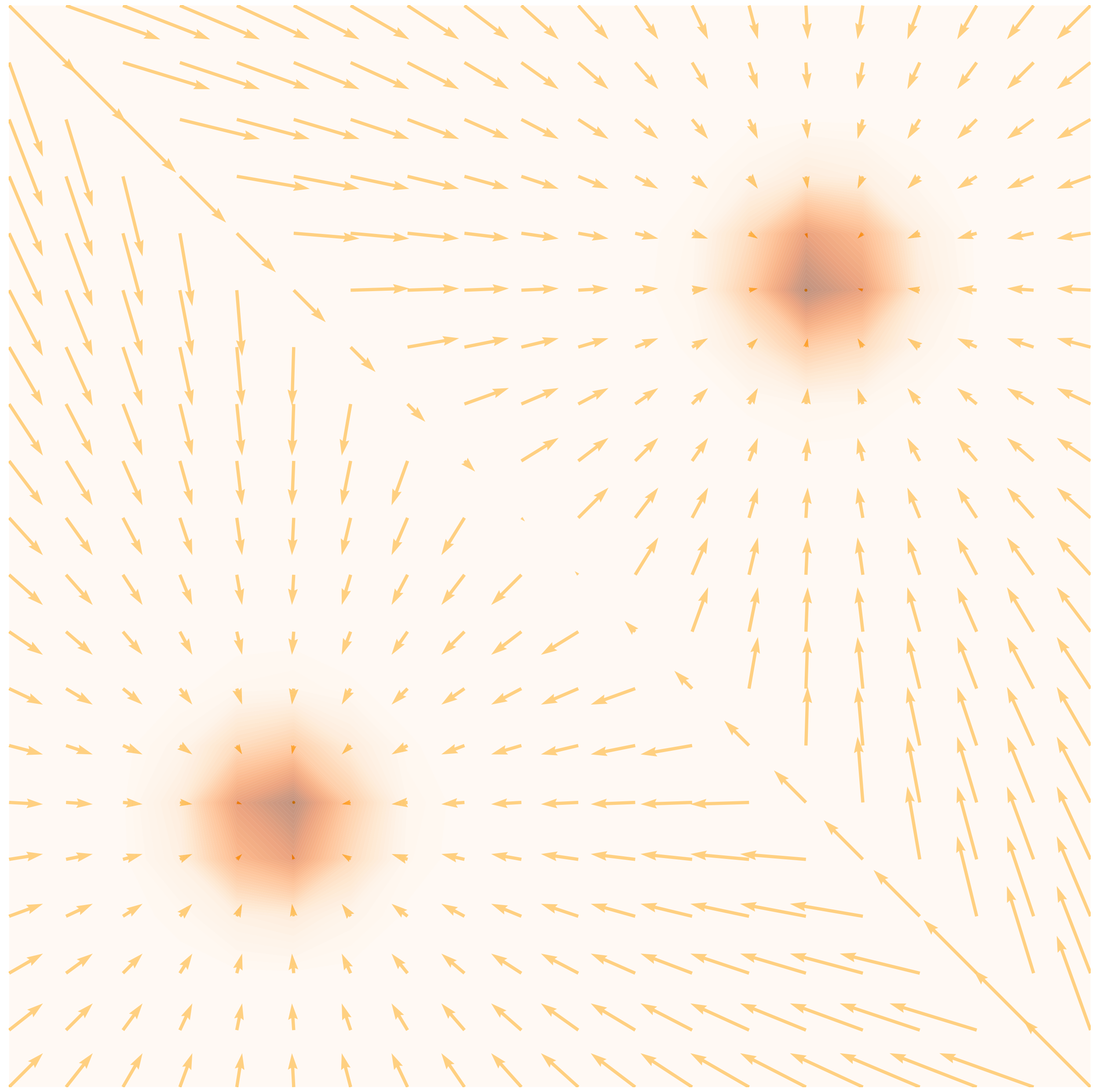


Score matching

$$s_\theta(x) = \nabla_x \log p_\theta(x)$$

Fischer divergence: If two PDFs $p(x)$ and $q(x)$ are similar, their score vector field should be similar: $D_F(p, q) = \frac{1}{2}\mathbb{E}_{x\sim p}\left[||\nabla_x \log p(x) - \nabla_x \log q(x)||_2^2\right]$





Score matching

$$s_\theta(x) = \nabla_x \log p_\theta(x)$$

Fischer divergence: If two PDFs $p(x)$ and $q(x)$ are similar, their score vector field should be similar: $D_F(p, q) = \frac{1}{2}\mathbb{E}_{x\sim p}\left[||\nabla_x \log p(x) - \nabla_x \log q(x)||_2^2\right]$

Score matching

$$s_\theta(x) = \nabla_x \log p_\theta(x)$$

Fischer divergence: If two PDFs $p(x)$ and $q(x)$ are similar, their score vector field should be similar: $D_F(p,q) = \frac{1}{2}\mathbb{E}_{x\sim p}\left[||\nabla_x \log p(x) - \nabla_x \log q(x)||_2^2\right]$

Score matching minimizes the Fischer divergence between $p_{data}(x)$ and the EBM $p_\theta(x) \propto \exp(f_\theta(x))$

Score matching

$$s_\theta(x) = \nabla_x \log p_\theta(x)$$

$$D_F(p, q) = \frac{1}{2}\mathbb{E}_{x \sim p}\left[|| \nabla_x \log p(x) - \nabla_x \log q(x) ||_2^2 \right]$$

$$=$$

Score matching minimizes the Fischer divergence between $p_{data}(x)$ and the EBM

$$p_\theta(x) \propto \exp(f_\theta(x))$$

Score matching

$$s_\theta(x) = \nabla_x \log p_\theta(x)$$

$$D_F(p, q) = \frac{1}{2}\mathbb{E}_{x \sim p}\left[|| \nabla_x \log p(x) - \nabla_x \log q(x) ||_2^2 \right]$$

$$= \frac{1}{2}\mathbb{E}_{x \sim p}\left[|| \nabla_x \log p_{data}(x) - s_\theta(x) ||_2^2 \right]$$

Score matching minimizes the Fischer divergence between $p_{data}(x)$ and the EBM

$p_\theta(x) \propto \exp(f_\theta(x))$

Score matching

$$s_\theta(x) = \nabla_x \log p_\theta(x)$$

$$\frac{1}{2}\mathbb{E}_{x\sim p}\left[|| \nabla_x \log p_{data}(x) - s_\theta(x) ||_2^2 \right]$$

Score matching

$$s_\theta(x) = \nabla_x \log p_\theta(x)$$

$$\frac{1}{2}\mathbb{E}_{x\sim p}\left[|| \nabla_x \log p_{data}(x) - s_\theta(x) ||_2^2 \right]$$

How to deal with $\nabla_x \log p_{data}(x)$ given only samples? Use integration by parts!

Score matching

$$s_\theta(x) = \nabla_x \log p_\theta(x)$$

$$\frac{1}{2}\mathbb{E}_{x\sim p}\left[|| \nabla_x \log p_{data}(x) - s_\theta(x) ||_2^2 \right]$$

How to deal with $\nabla_x \log p_{data}(x)$ given only samples? Use integration by parts!

$$\mathbb{E}_{x\sim p}\left[\frac{1}{2} || \nabla_x \log p_\theta(x) ||_2^2 + tr(\nabla_x^2 \log p_\theta(x)) \right]$$

Score matching

$$s_\theta(x) = \nabla_x \log p_\theta(x)$$

$$\frac{1}{2}\mathbb{E}_{x\sim p}\left[|| \nabla_x \log p_{data}(x) - s_\theta(x) ||_2^2 \right]$$

How to deal with $\nabla_x \log p_{data}(x)$ given only samples? Use integration by parts!

$$\mathbb{E}_{x\sim p}\left[\frac{1}{2} || \nabla_x \log p_\theta(x) ||_2^2 + tr(\nabla_x^2 \log p_\theta(x)) \right]$$

Score matching

$$\mathbb{E}_{x \sim p}\left[\frac{1}{2}||\nabla_x \log p_\theta(x)||_2^2 + tr(\nabla_x^2 \log p_\theta(x))\right]$$

Score matching

$$\mathbb{E}_{x \sim p}\left[\frac{1}{2}||\nabla_x \log p_\theta(x)||_2^2 + tr(\nabla_x^2 \log p_\theta(x))\right]$$

Sample from a mini-batch of datapoints $\{x_1, x_2, \cdots, x_n\}$

Score matching

$$\mathbb{E}_{x \sim p}\left[\frac{1}{2}||\nabla_x \log p_\theta(x)||_2^2 + tr(\nabla_x^2 \log p_\theta(x))\right]$$

Sample from a mini-batch of datapoints $\{x_1, x_2, \cdots, x_n\}$

Estimate the score matching loss with empirical mean over all data points

Score matching

$$\mathbb{E}_{x \sim p}\left[\frac{1}{2}||\nabla_x \log p_\theta(x)||_2^2 + tr(\nabla_x^2 \log p_\theta(x))\right]$$

Sample from a mini-batch of datapoints $\{x_1, x_2, \cdots, x_n\}$

Estimate the score matching loss with empirical mean over all data points

$$\frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{x \sim p_{data}}\left[\frac{1}{2}||\nabla_x \log p_\theta(x_i)||_2^2 + tr(\nabla_x^2 \log p_\theta(x_i))\right]$$

Score matching

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{x\sim p_{data}}\left[\frac{1}{2}||\nabla_x \log p_\theta(x_i)||_2^2 + tr(\nabla_x^2 \log p_\theta(x_i))\right]$$

Sample from a mini-batch of datapoints $\{x_1, x_2, \cdots, x_n\}$

Estimate the score matching loss with empirical mean over all data points

Score matching

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{x\sim p_{data}}\left[\frac{1}{2}||\nabla_x \log p_\theta(x_i)||_2^2 + tr(\nabla_x^2 \log p_\theta(x_i))\right]$$

Sample from a mini-batch of datapoints $\{x_1, x_2, \cdots, x_n\}$

Estimate the score matching loss with empirical mean over all data points

Perform stochastic gradient descent (SGD)

Score matching

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{x\sim p_{data}}\left[\frac{1}{2}||\nabla_x \log p_\theta(x_i)||_2^2 + tr(\nabla_x^2 \log p_\theta(x_i))\right]$$

Sample from a mini-batch of datapoints $\{x_1, x_2, \cdots, x_n\}$

Estimate the score matching loss with empirical mean over all data points

Perform stochastic gradient descent (SGD)

No need to sample from the EBMs!

Score matching for EBMs

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{x\sim p_{data}}\left[\frac{1}{2}\,||\,\nabla_x \log p_\theta(x_i)\,||_2^2 + tr(\,\nabla_x^2 \log p_\theta(x_i))\right]$$

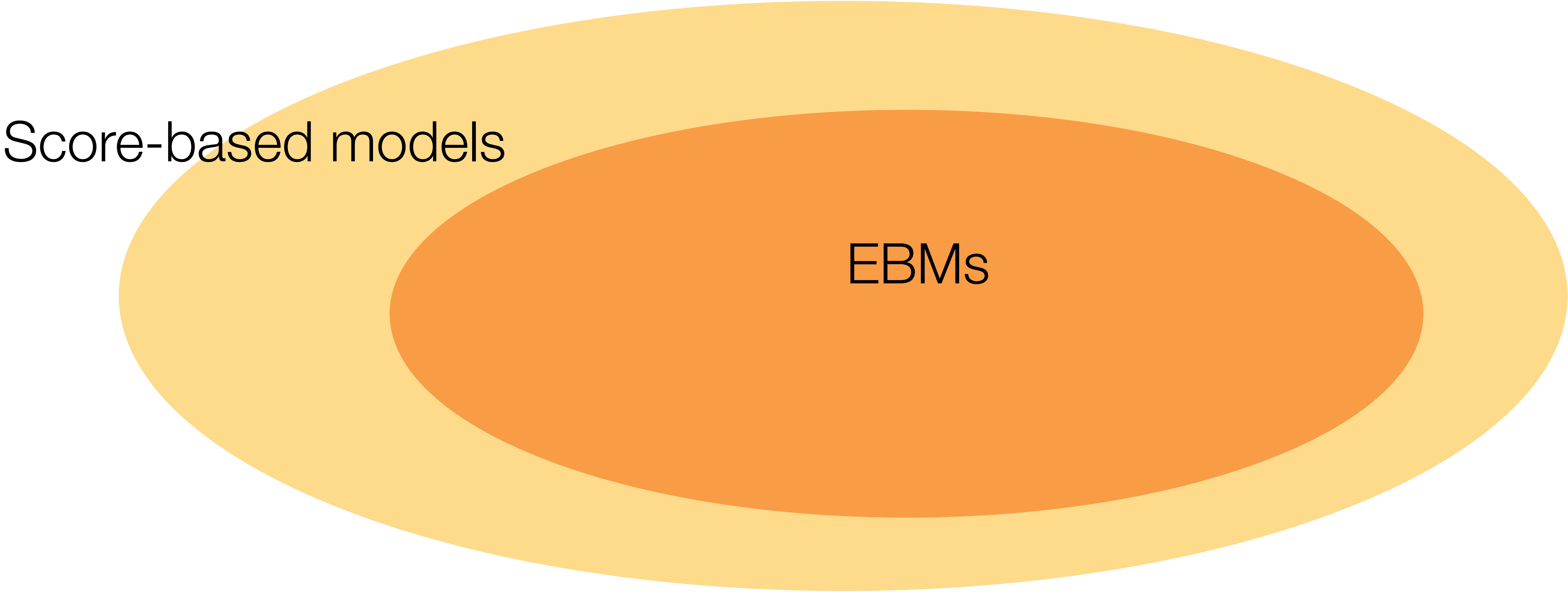
Score matching for EBMs

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{x\sim p_{data}}\left[\frac{1}{2}||\nabla_x \log p_\theta(x_i)||_2^2 + tr(\nabla_x^2 \log p_\theta(x_i))\right]$$

Caveat: The Hessian $tr(\nabla_x^2 \log p_\theta(x))$ term is computationally very expensive for large models.

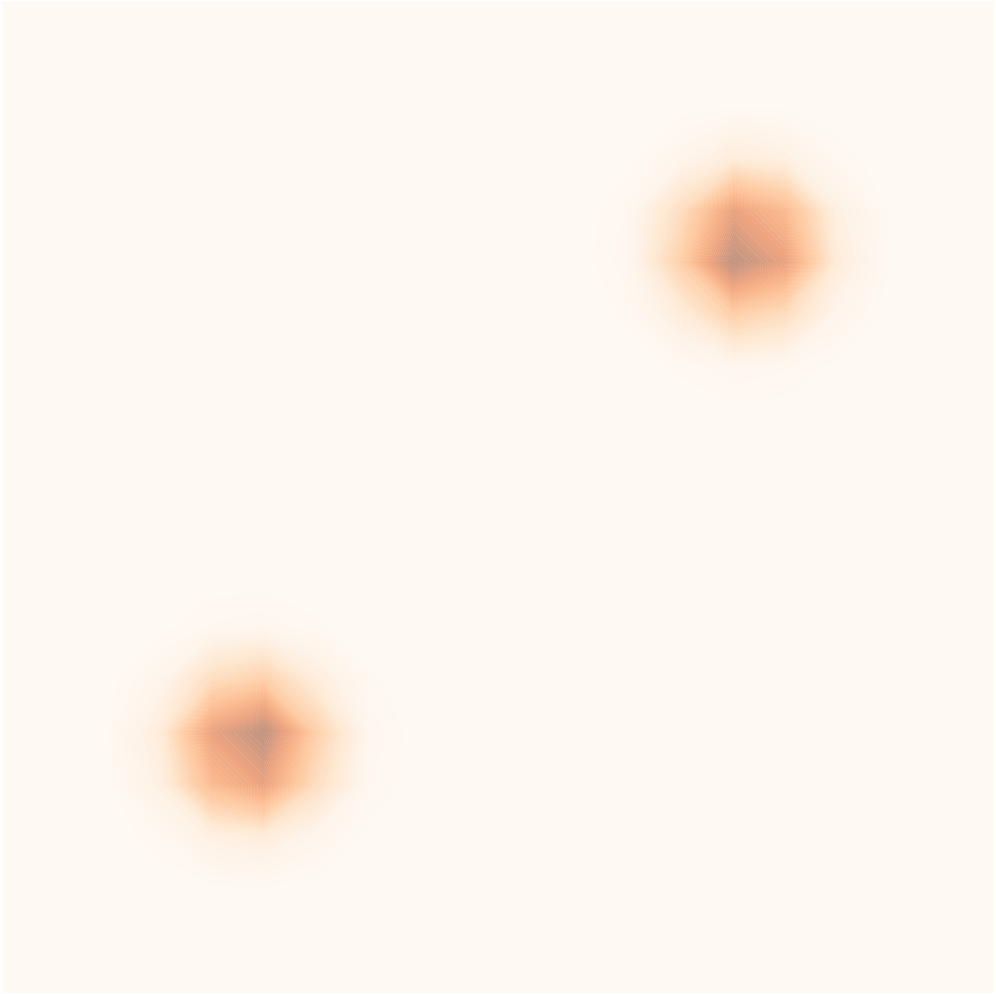
Score-based generative models

Score-based generative models



Score estimation by training score-based models

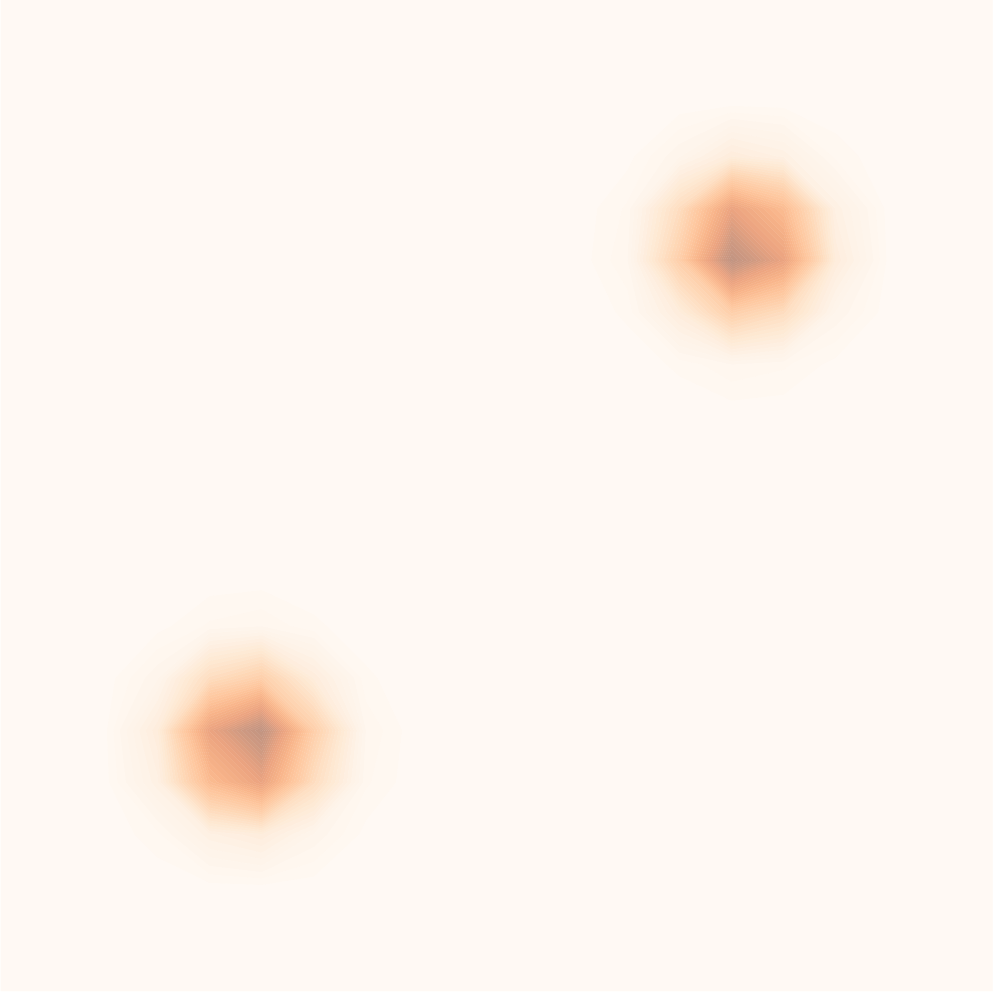
$p_{data}(x)$



PDF

Score estimation by training score-based models

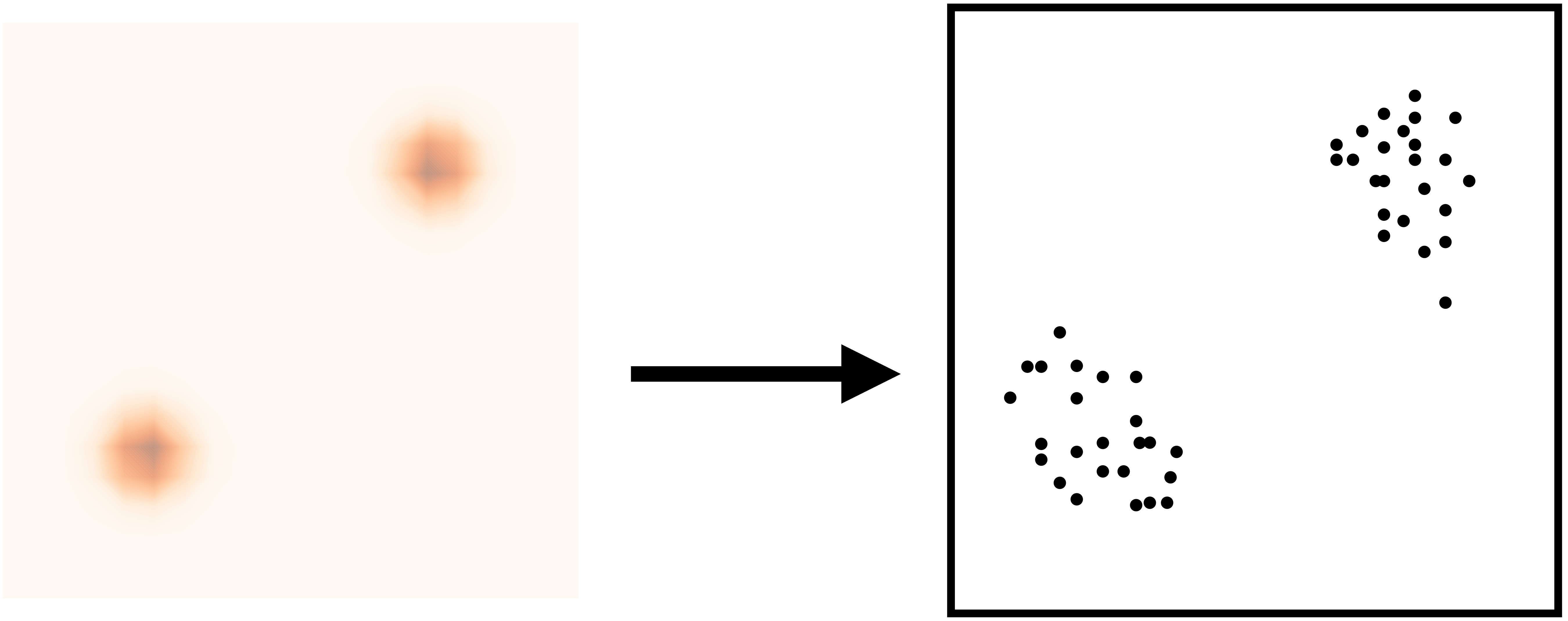
$p_{data}(x)$



PDF

Score estimation by training score-based models

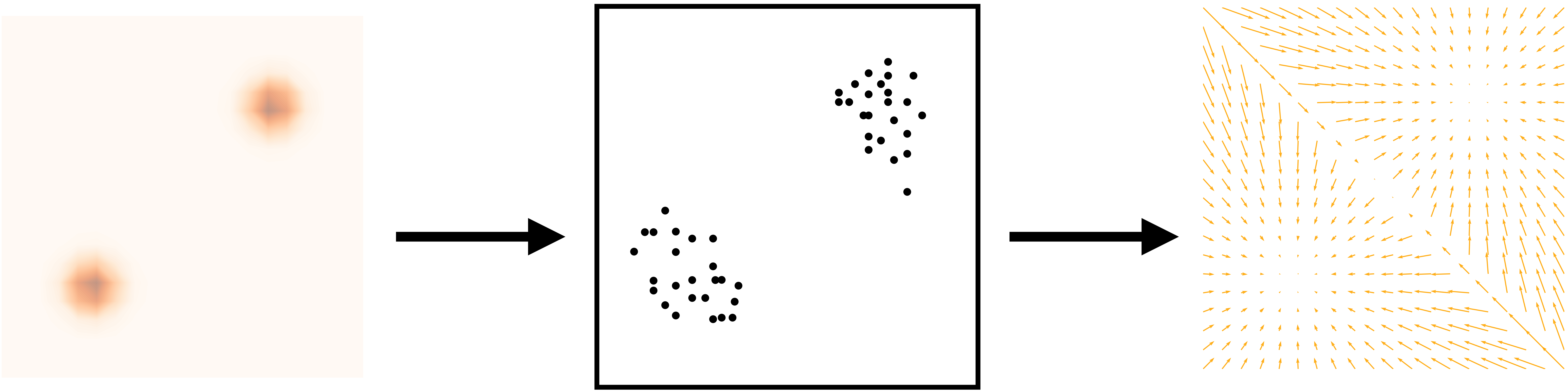
$p_{data}(x)$



PDF

Data samples

Score estimation by training score-based models



$p_{data}(x)$

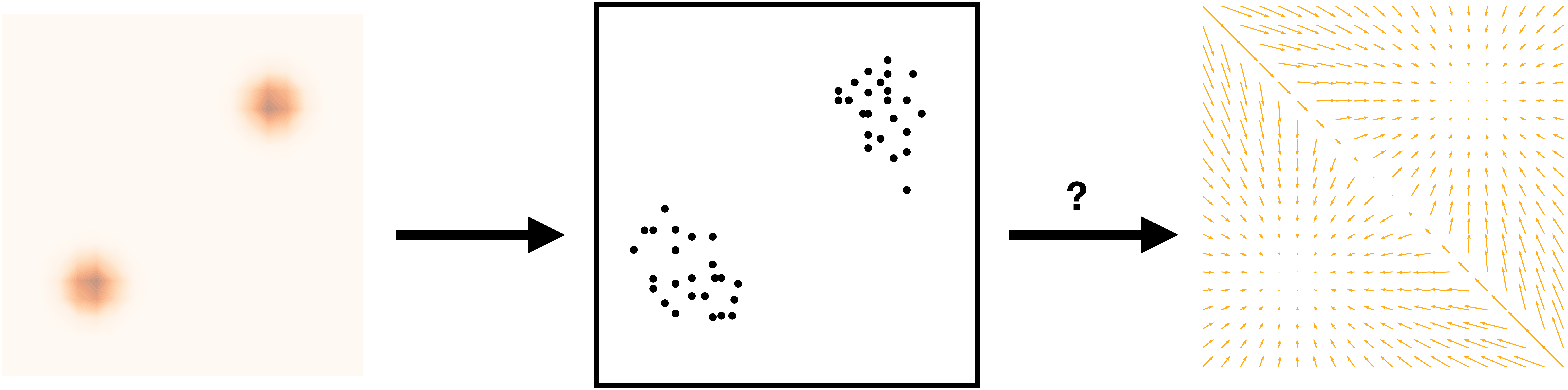
PDF

Data samples

$s_\theta(x) \approx \nabla_x \log p_{data}(x)$

Score function

Score estimation by training score-based models



$p_{data}(x)$

$s_\theta(x) \approx \nabla_x \log p_{data}(x)$

PDF

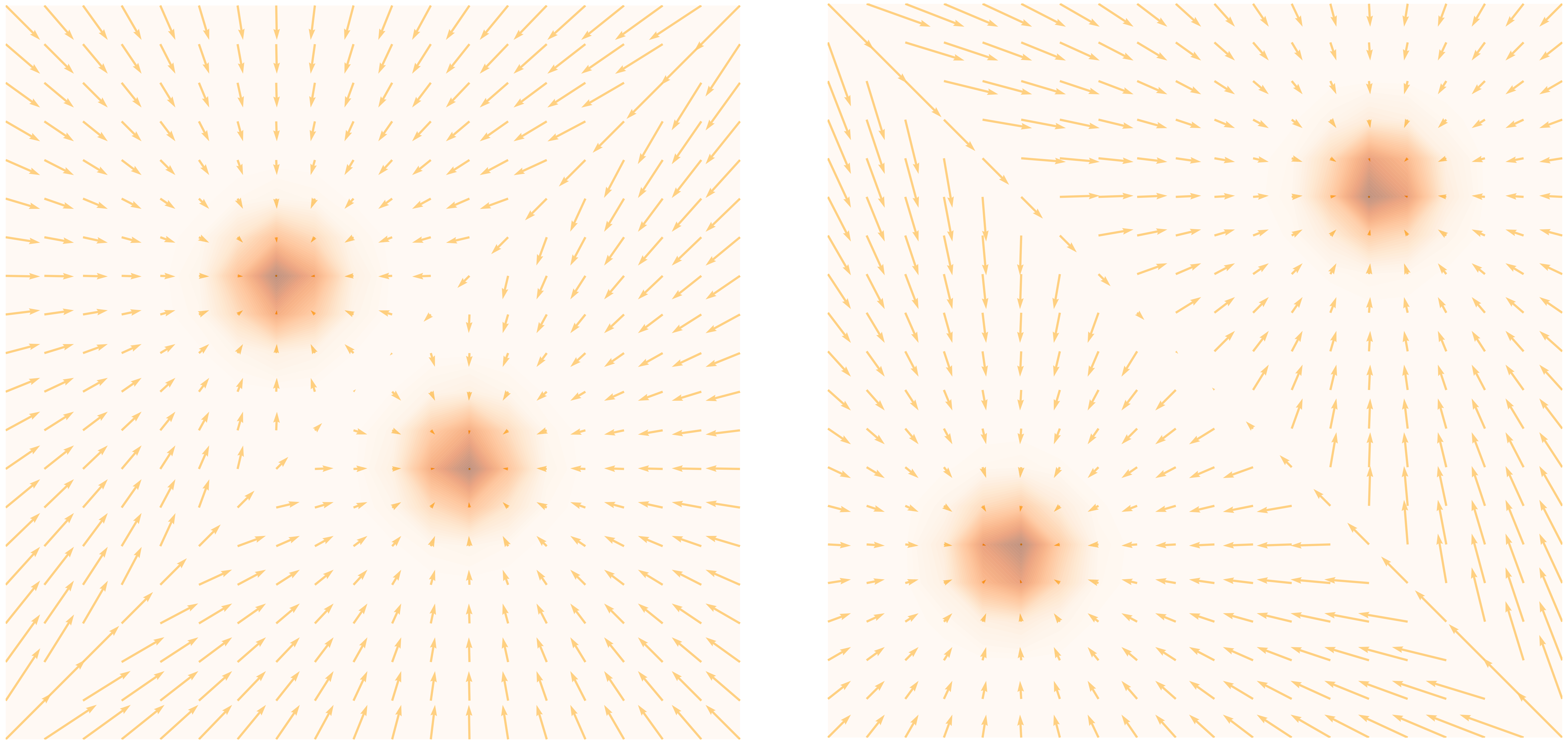
Data samples

Score function

Score estimation by training score-based models

$$s_\theta(x) \approx \nabla_x \log p_{data}(x)$$

Score matching:

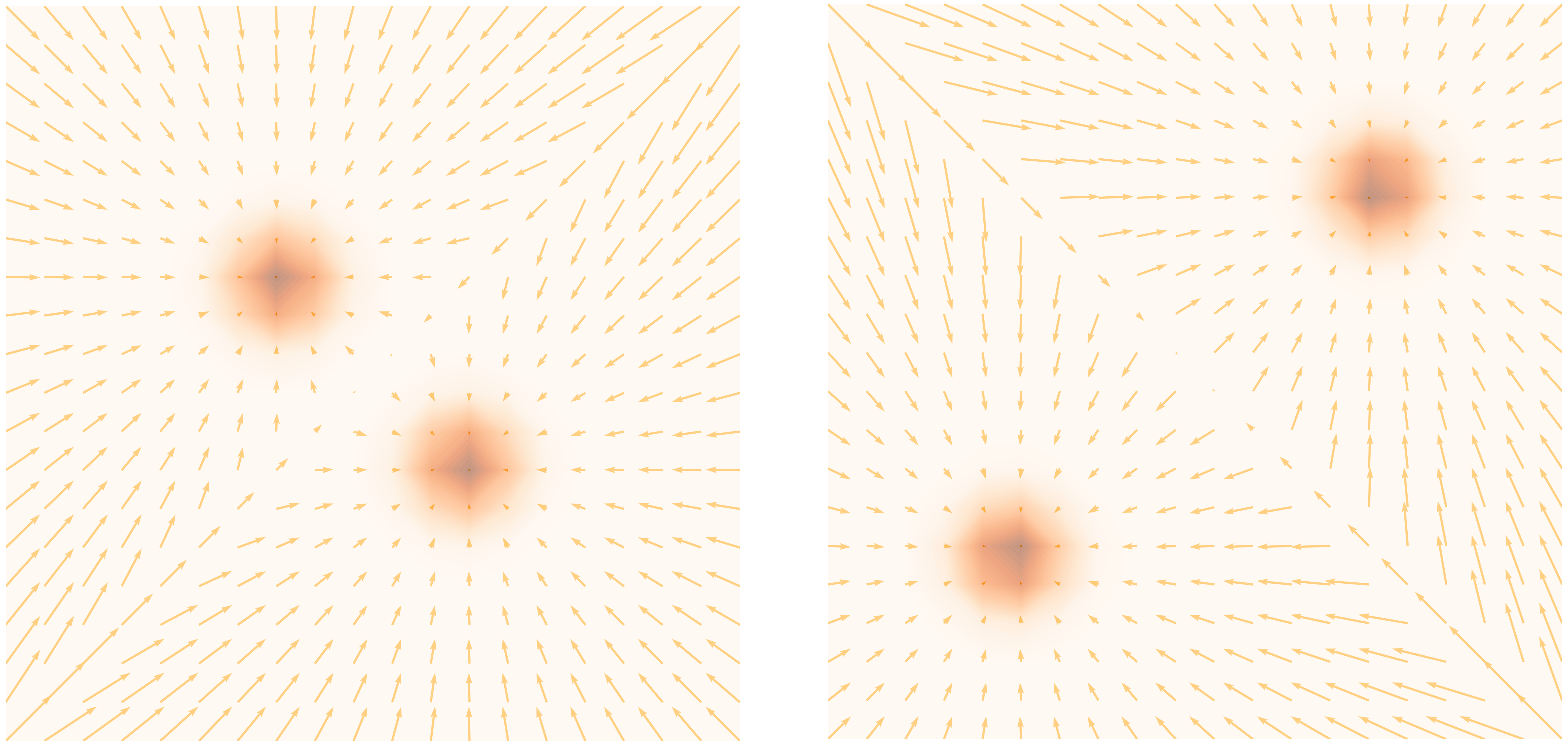


Score estimation by training score-based models

$s_\theta(x) \approx \nabla_x \log p_{data}(x)$

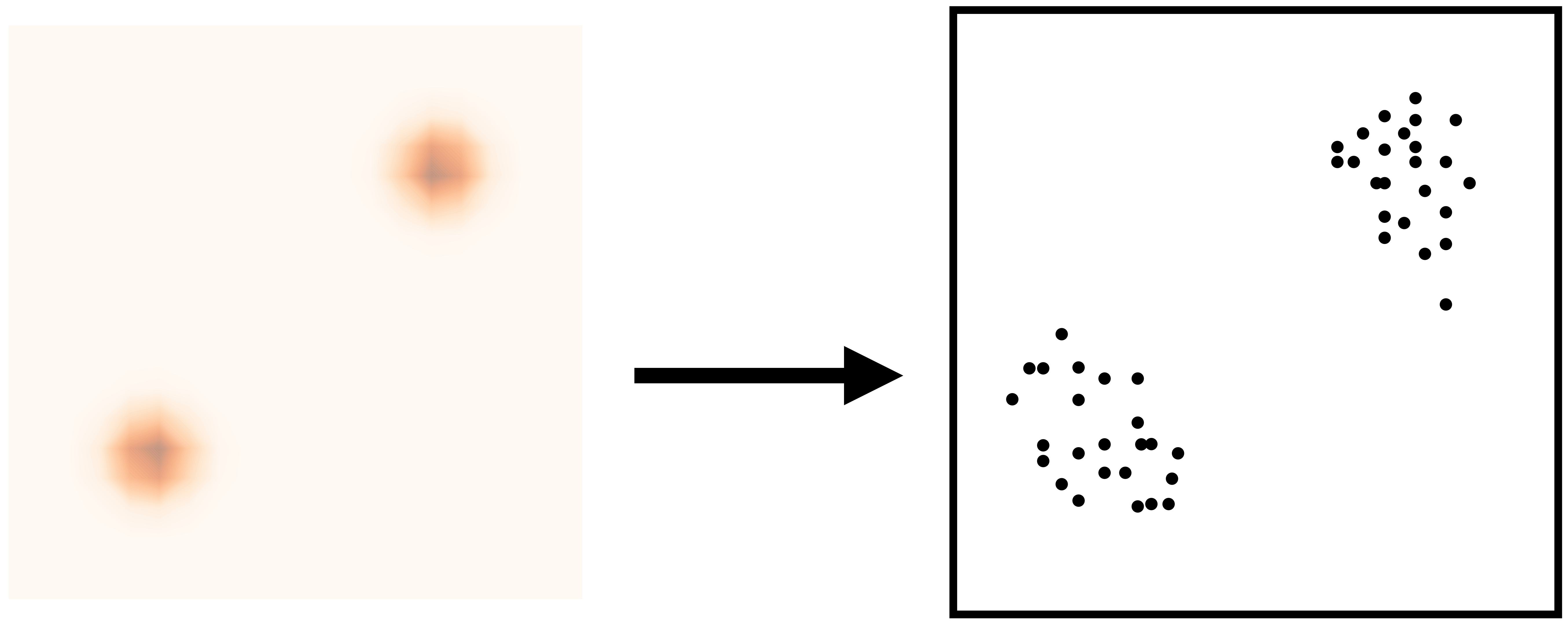
Objective: Minimize the difference between a predicted score vector field wrt the ground truth

Score matching:



Score estimation by training score-based models

$p_{data}(x)$

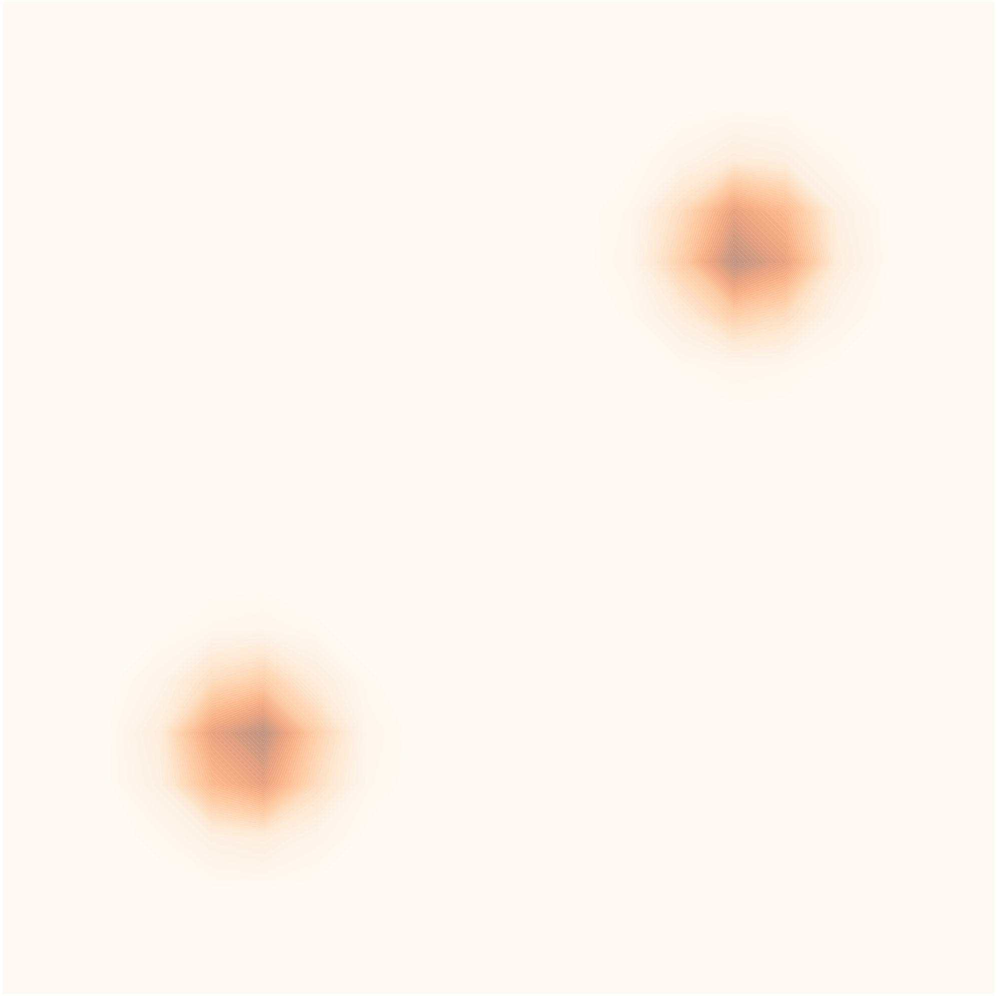


PDF

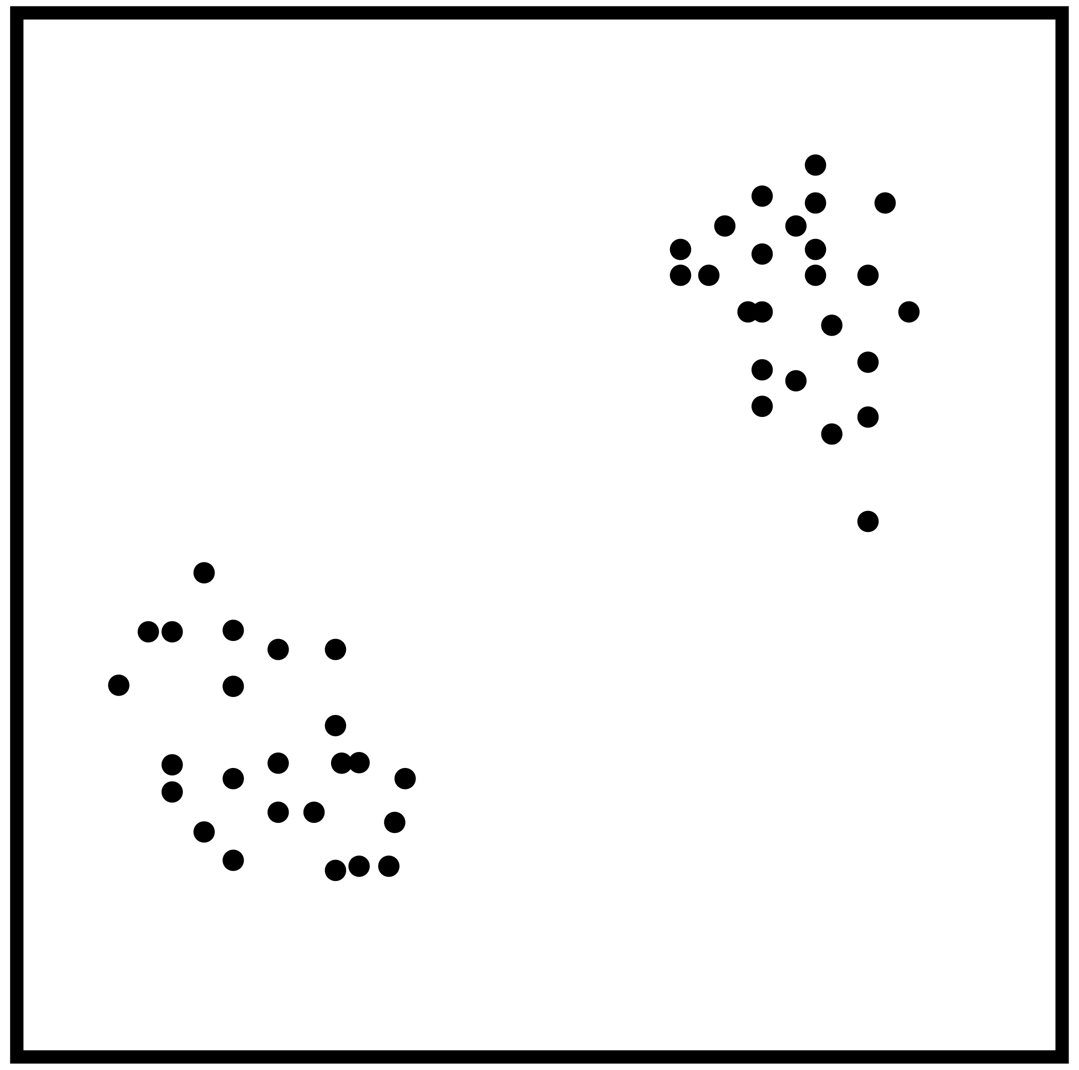
Data samples

Score estimation by training score-based models

$p_{data}(x)$

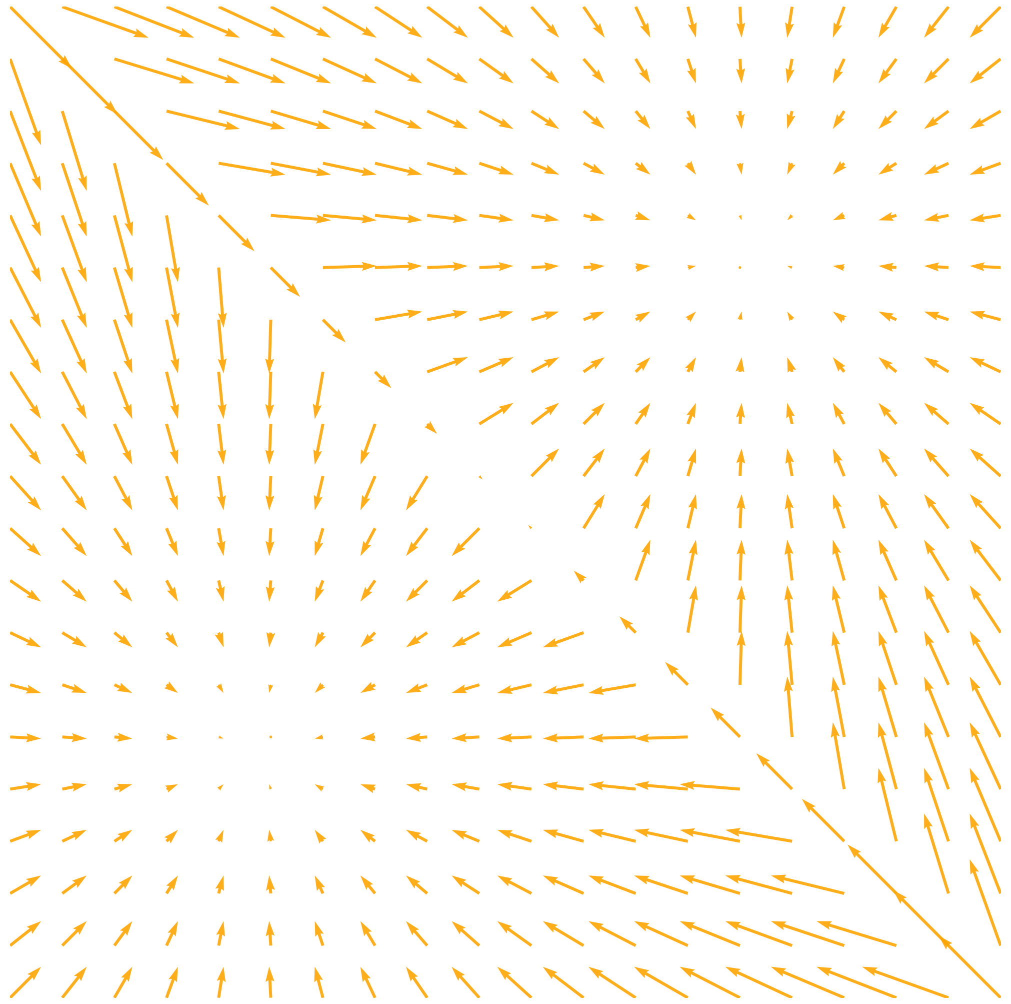


PDF



Data samples

$s_\theta(x) \approx \nabla_x \log p_{data}(x)$

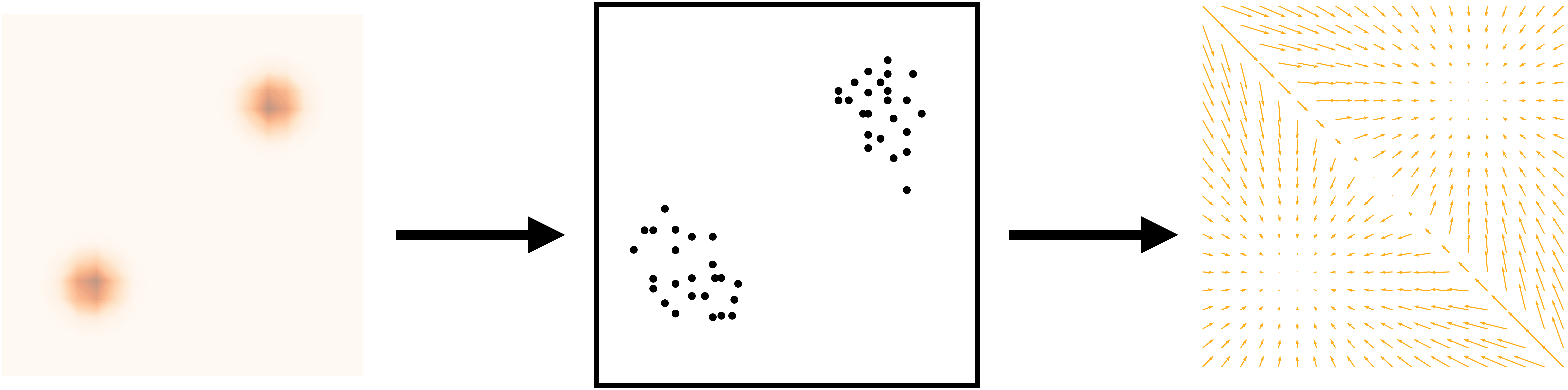


Score function

Score estimation by training score-based models

$p_{data}(x)$

$s_\theta(x) \approx \nabla_x \log p_{data}(x)$

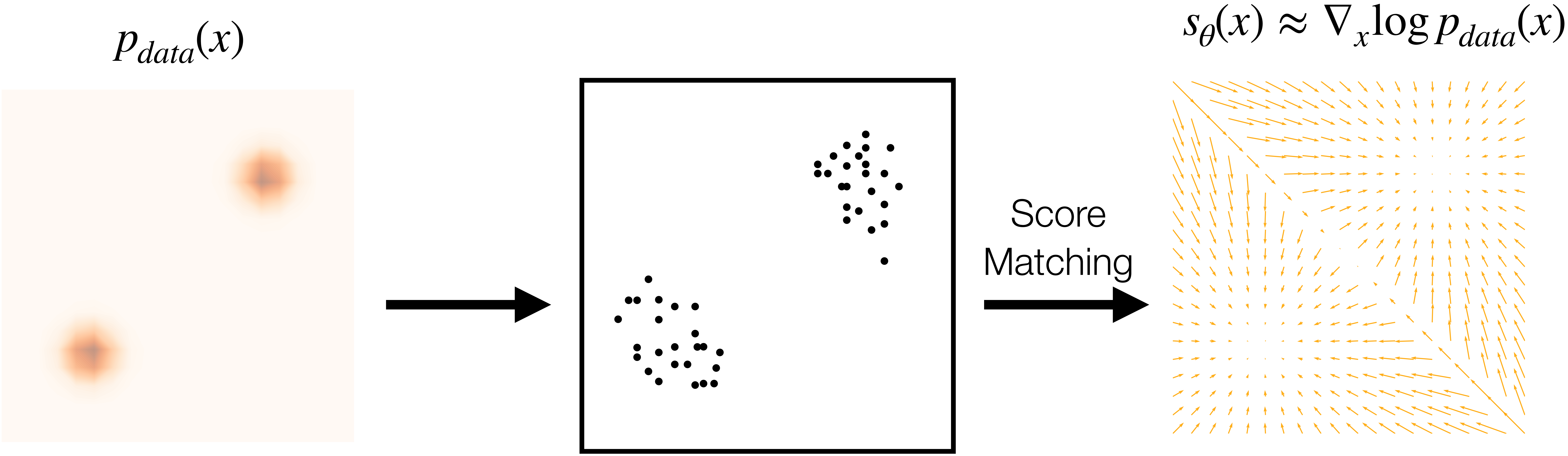


PDF

Data samples

Score function

Score estimation by training score-based models



$p_{data}(x)$

$s_\theta(x) \approx \nabla_x \log p_{data}(x)$

Score Matching

PDF

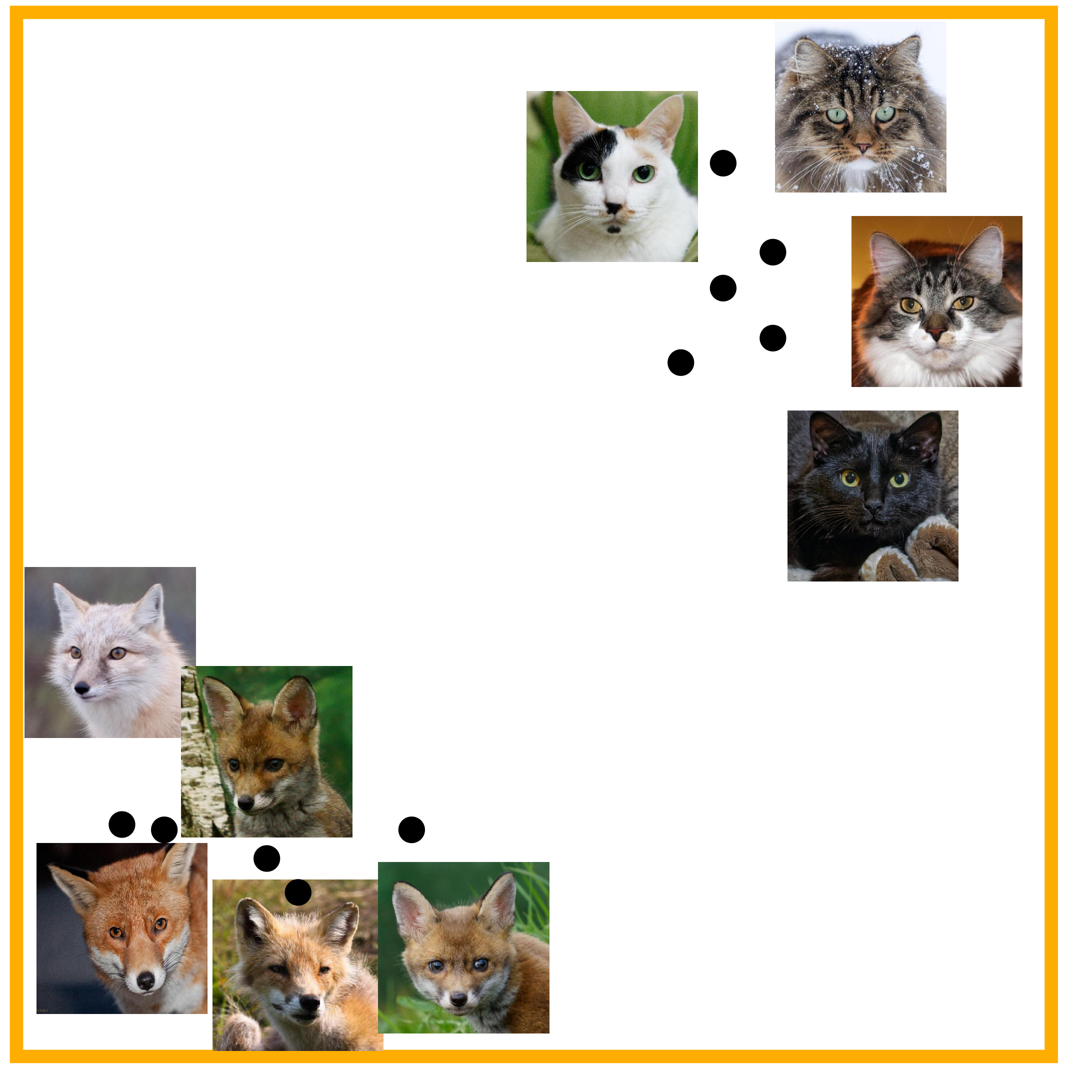
Data samples

Score function

How do we generate samples?

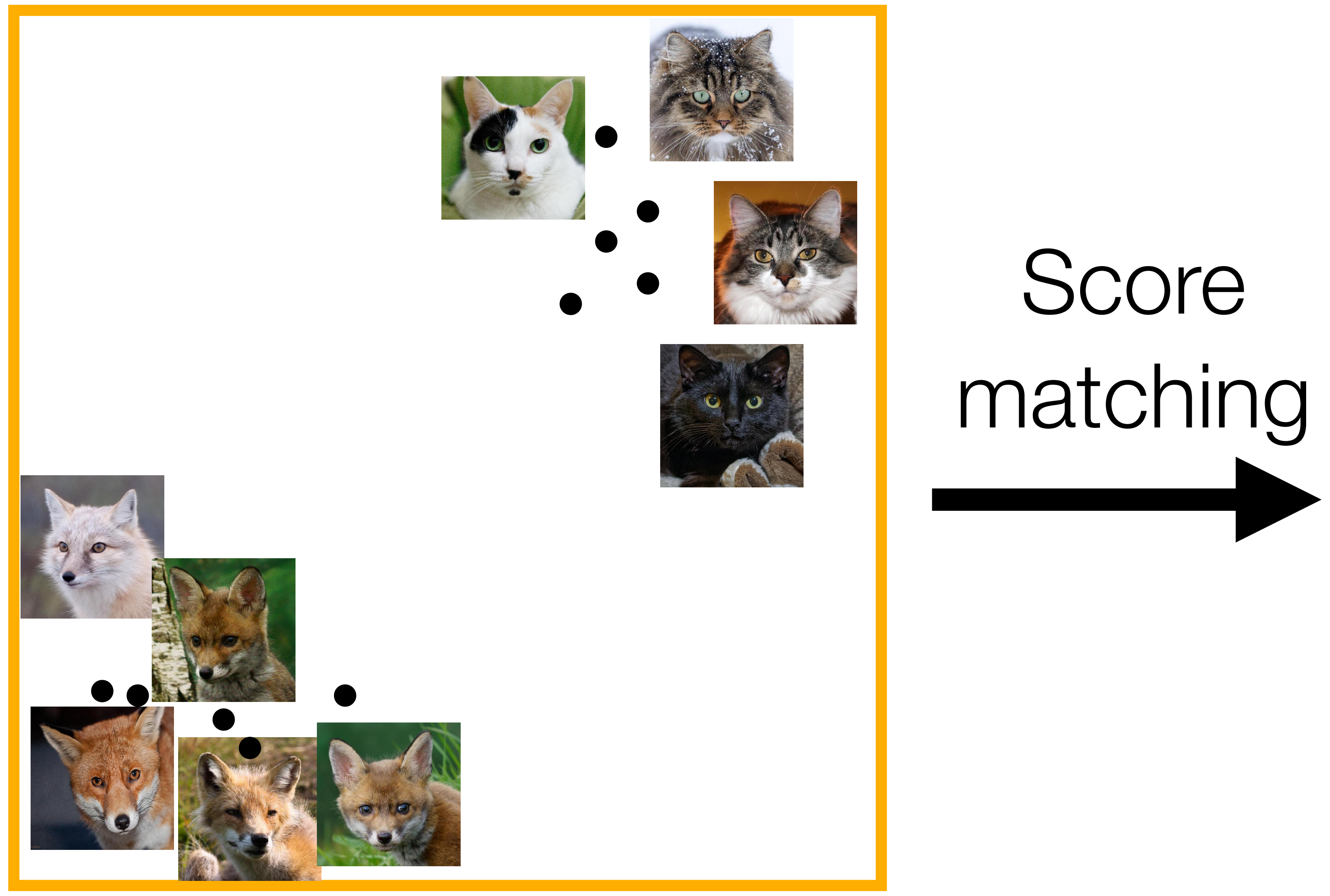
Role of MCMC in Score-based Models

Sampling in score-based generative models



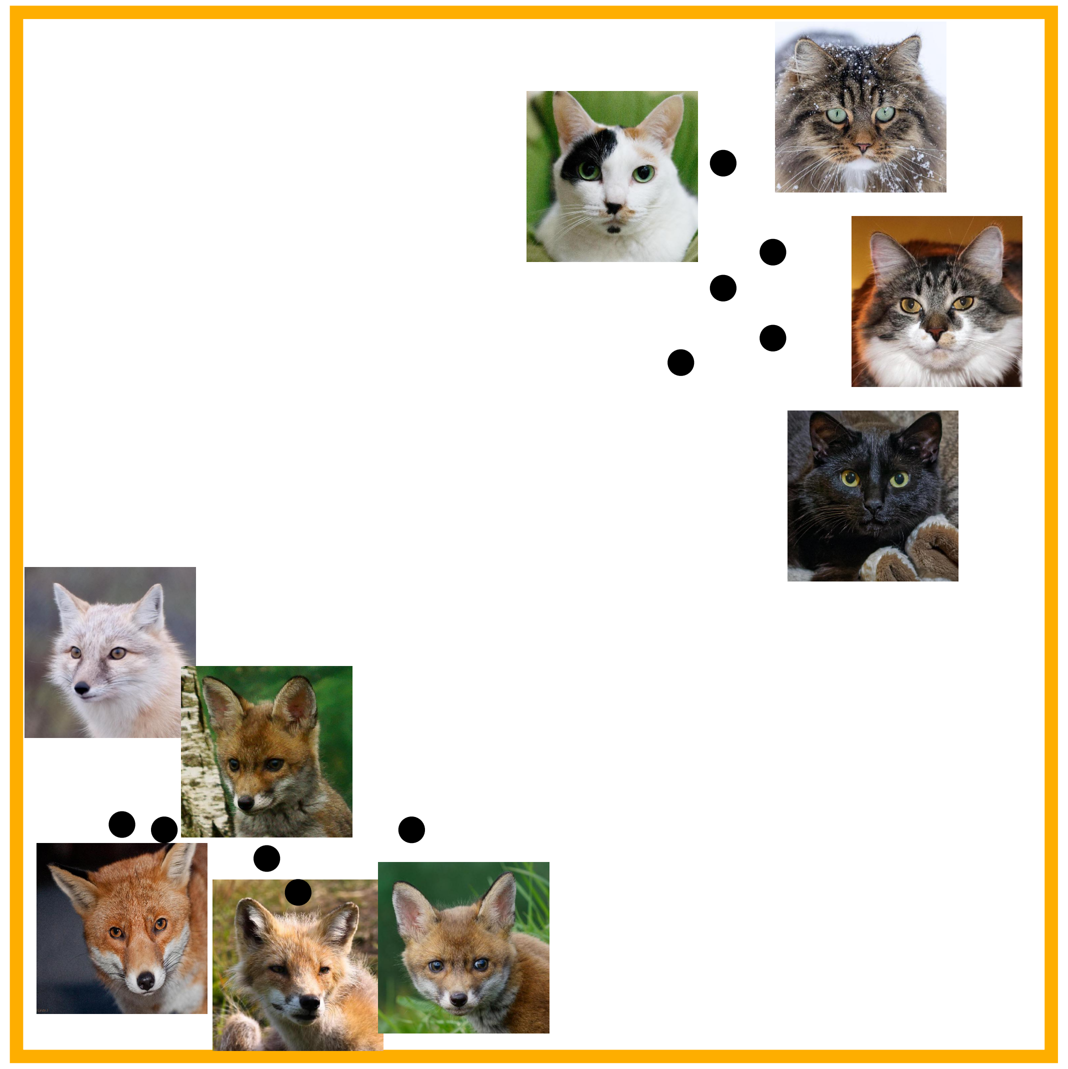
Data samples

Sampling in score-based generative models

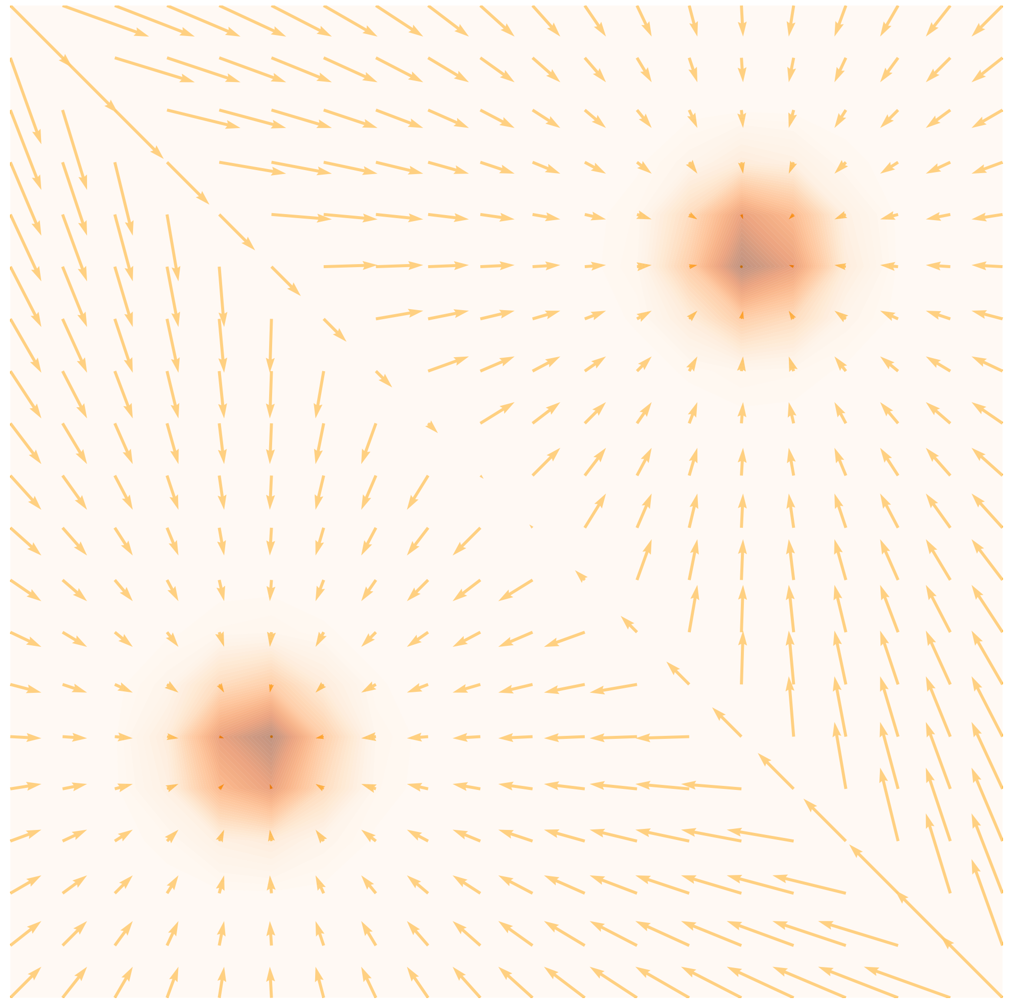


Data samples

Sampling in score-based generative models



Score matching

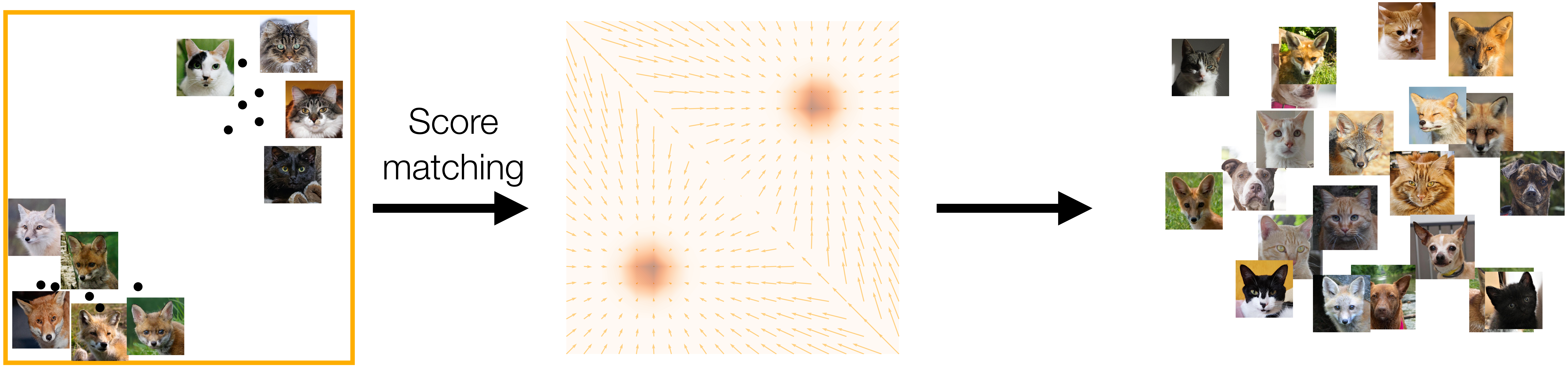


Data samples

Scores

$$s_\theta(x) \approx \nabla_x \log p_{data}(x)$$

Sampling in score-based generative models



Score matching

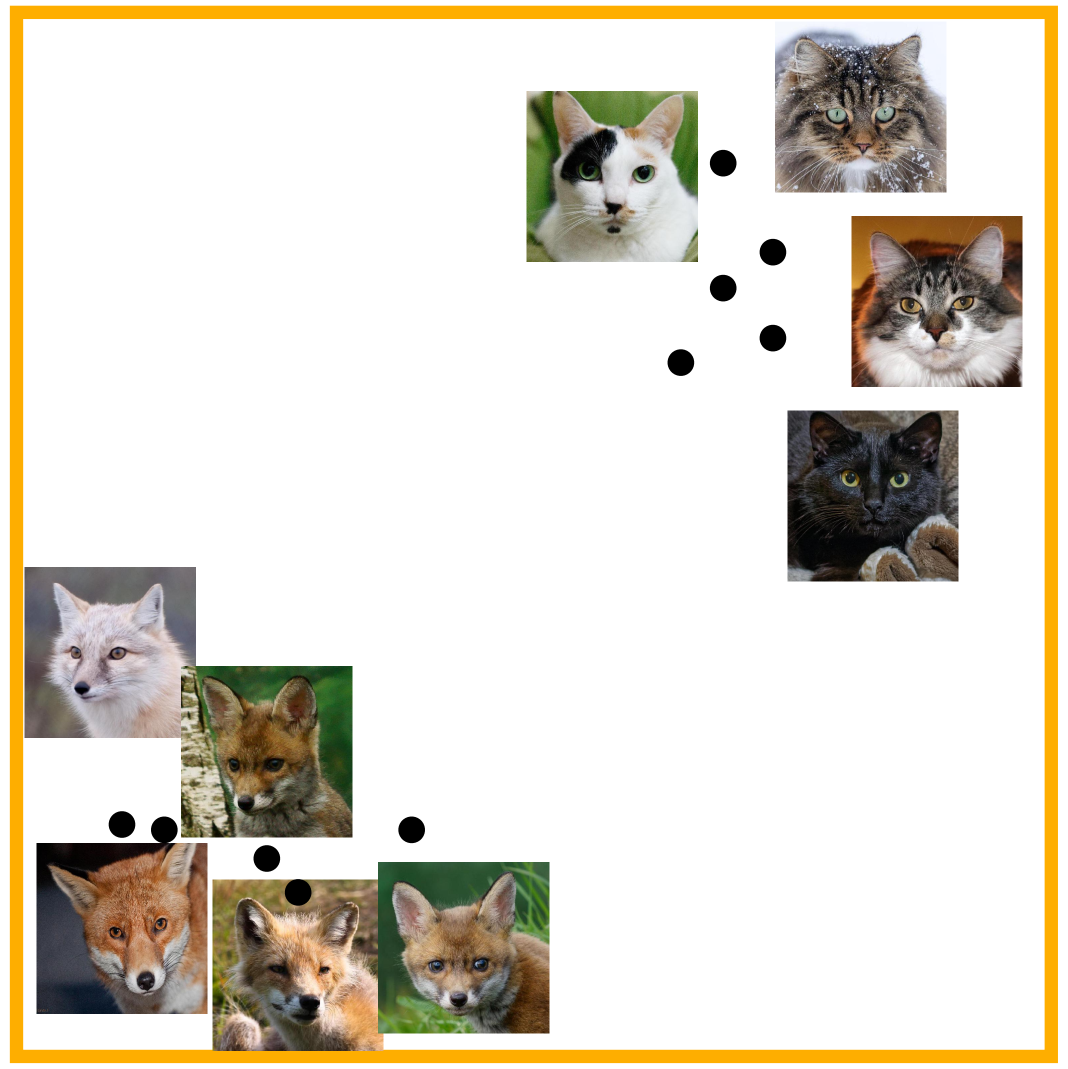
Data samples

Scores

$$s_\theta(x) \approx \nabla_x \log p_{data}(x)$$

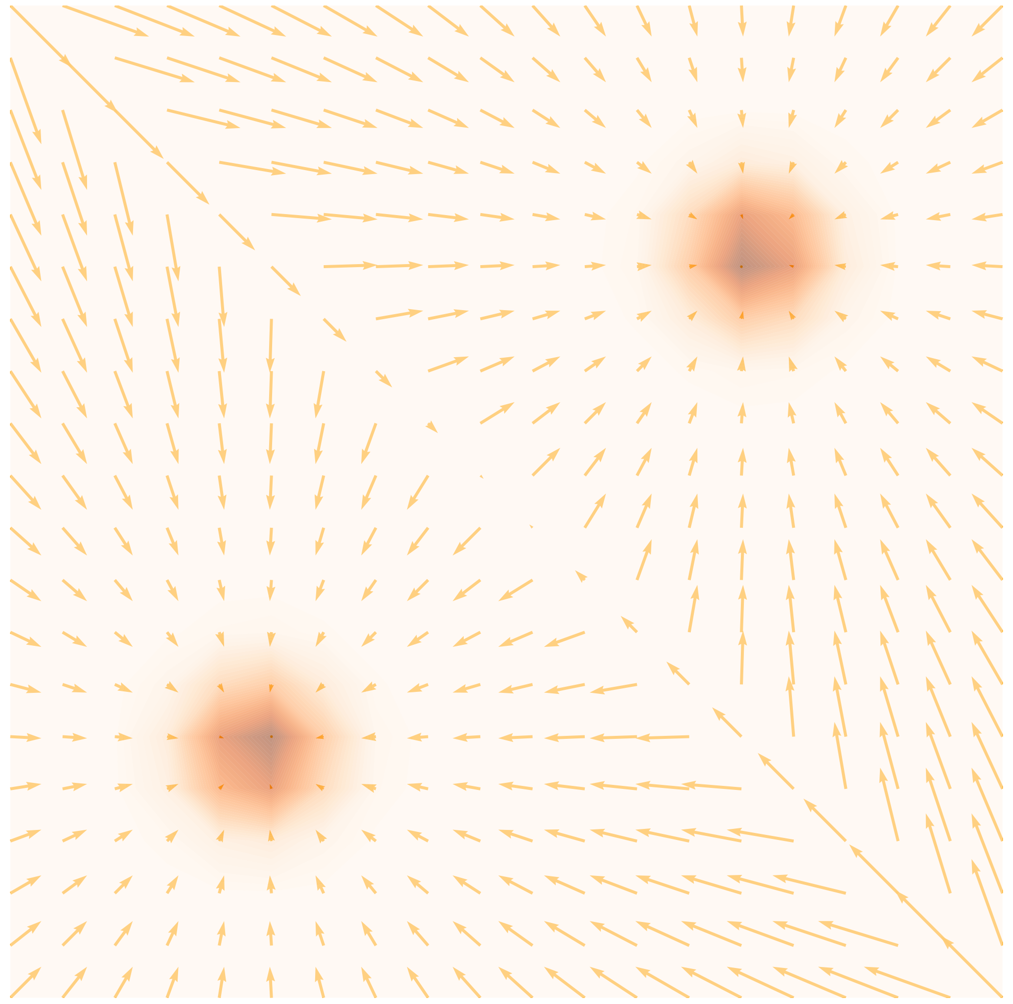
New samples

Sampling in score-based generative models



Data samples

Score matching



Scores

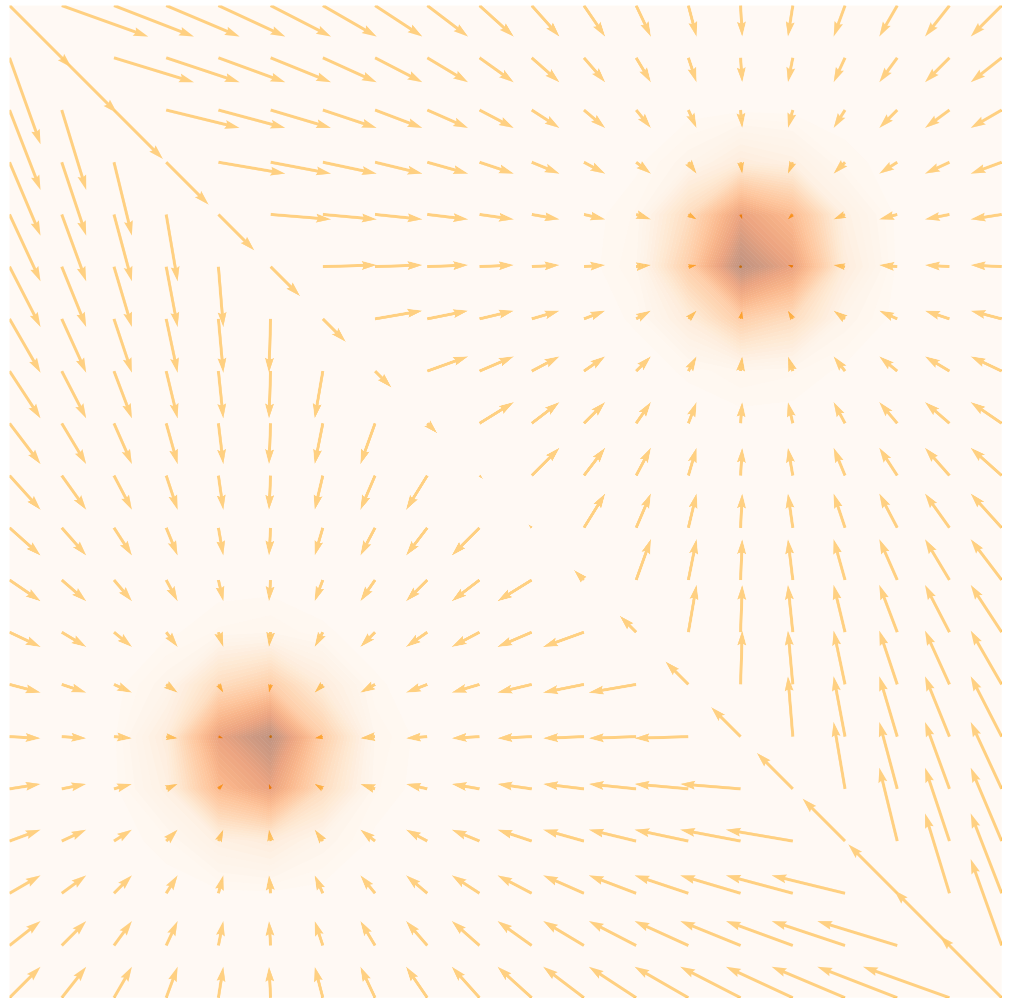
$s_\theta(x) \approx \nabla_x \log p_{data}(x)$

?

New samples

From scores to samples: Langevin MCMC

Score matching

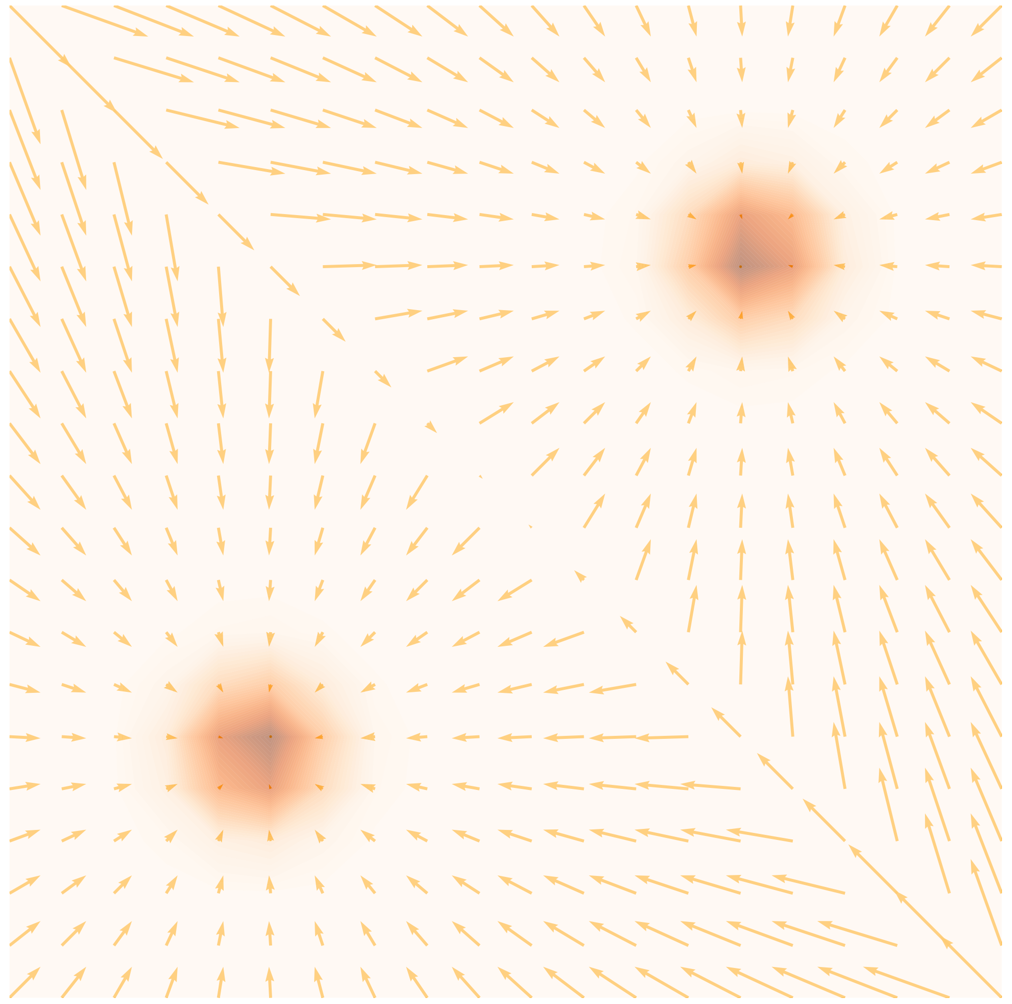


Scores

$s_\theta(x) \approx \nabla_x \log p_{data}(x)$

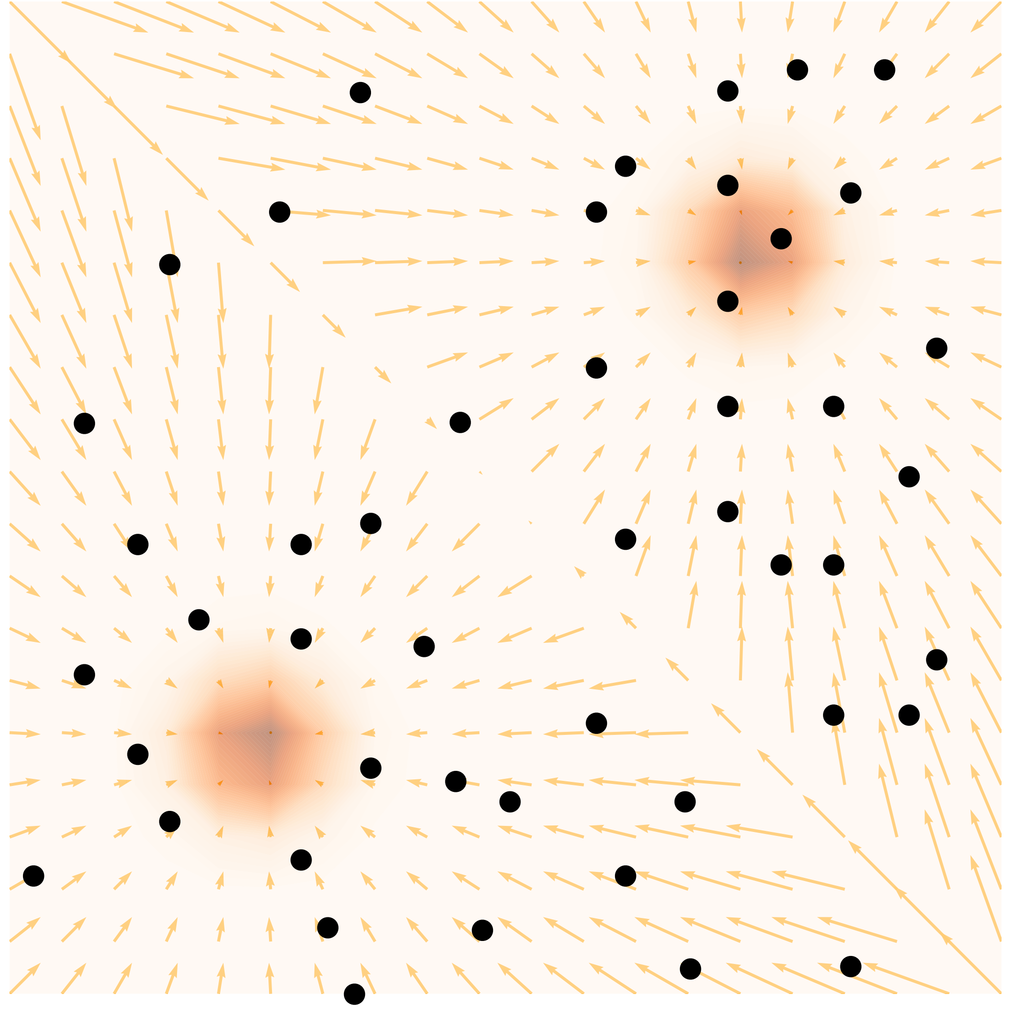
From scores to samples: Langevin MCMC

Score matching →



Scores

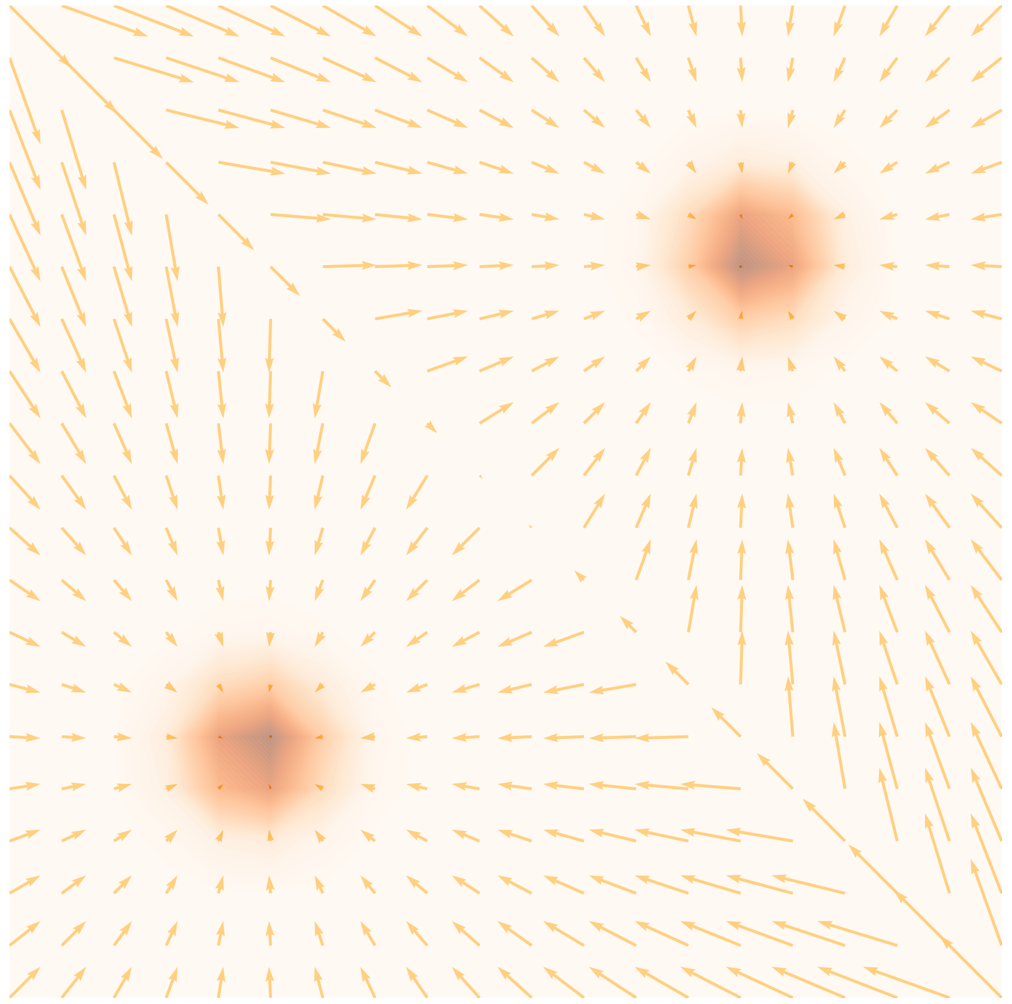
$s_\theta(x) \approx \nabla_x \log p_{data}(x)$

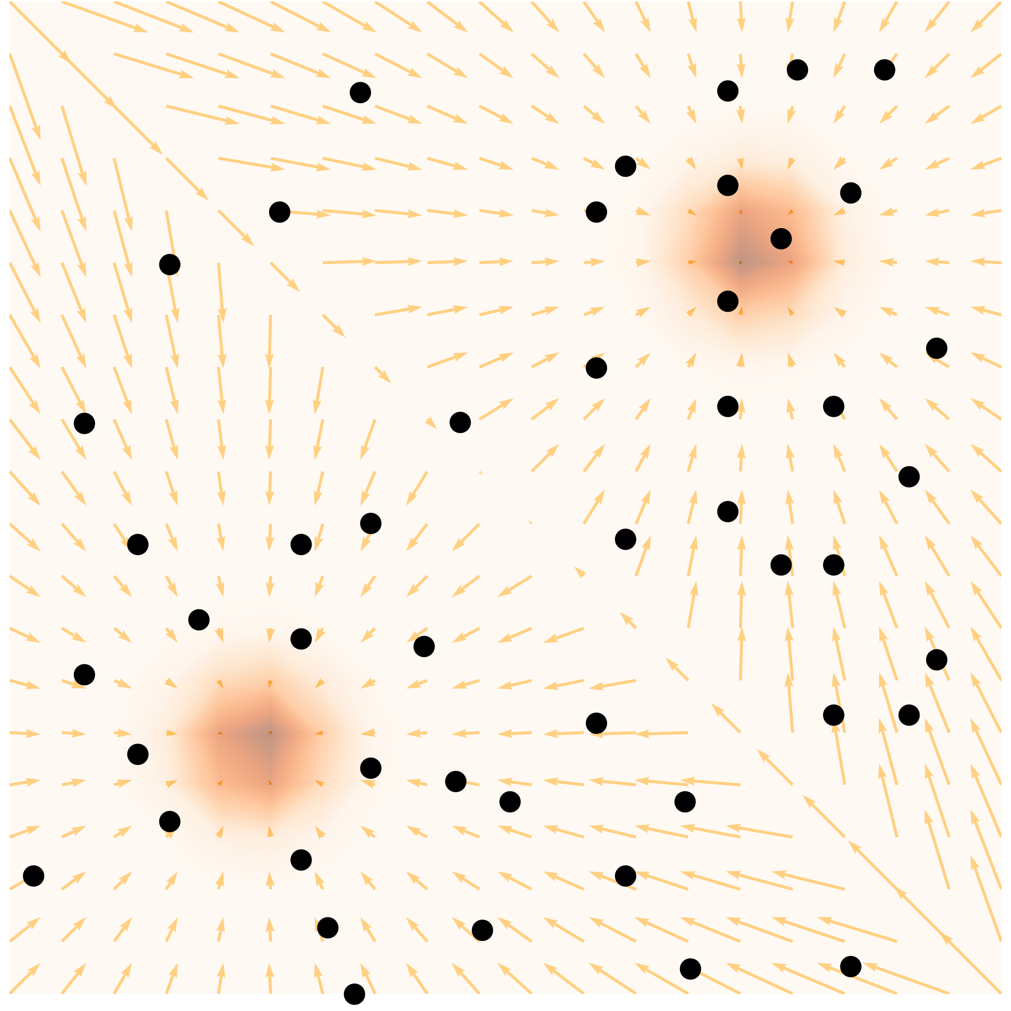


Follow the score

From scores to samples: Langevin MCMC

Score matching





Scores

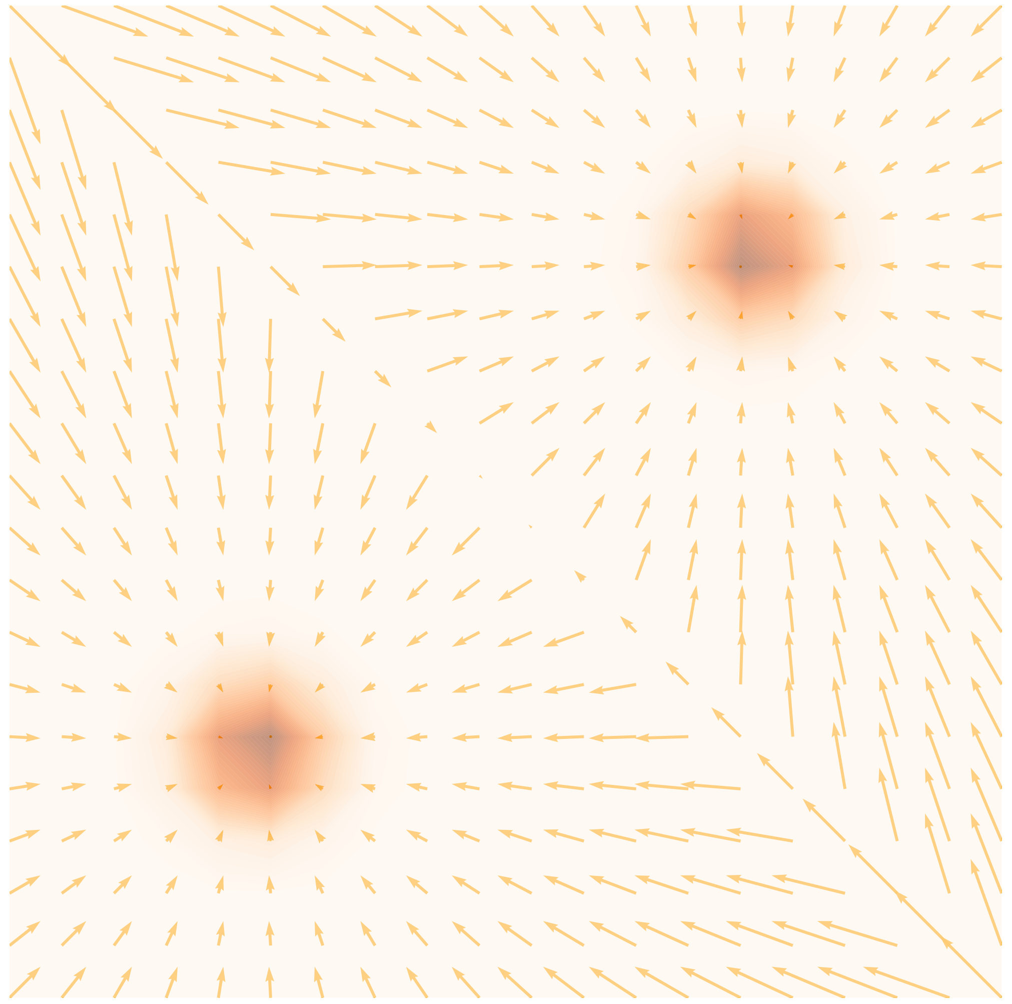
$s_\theta(x) \approx \nabla_x \log p_{data}(x)$

Follow the score

$$\tilde{x}_{t+1} = \tilde{x}_t + \frac{\tau}{2} s_\theta(\tilde{x}_t)$$

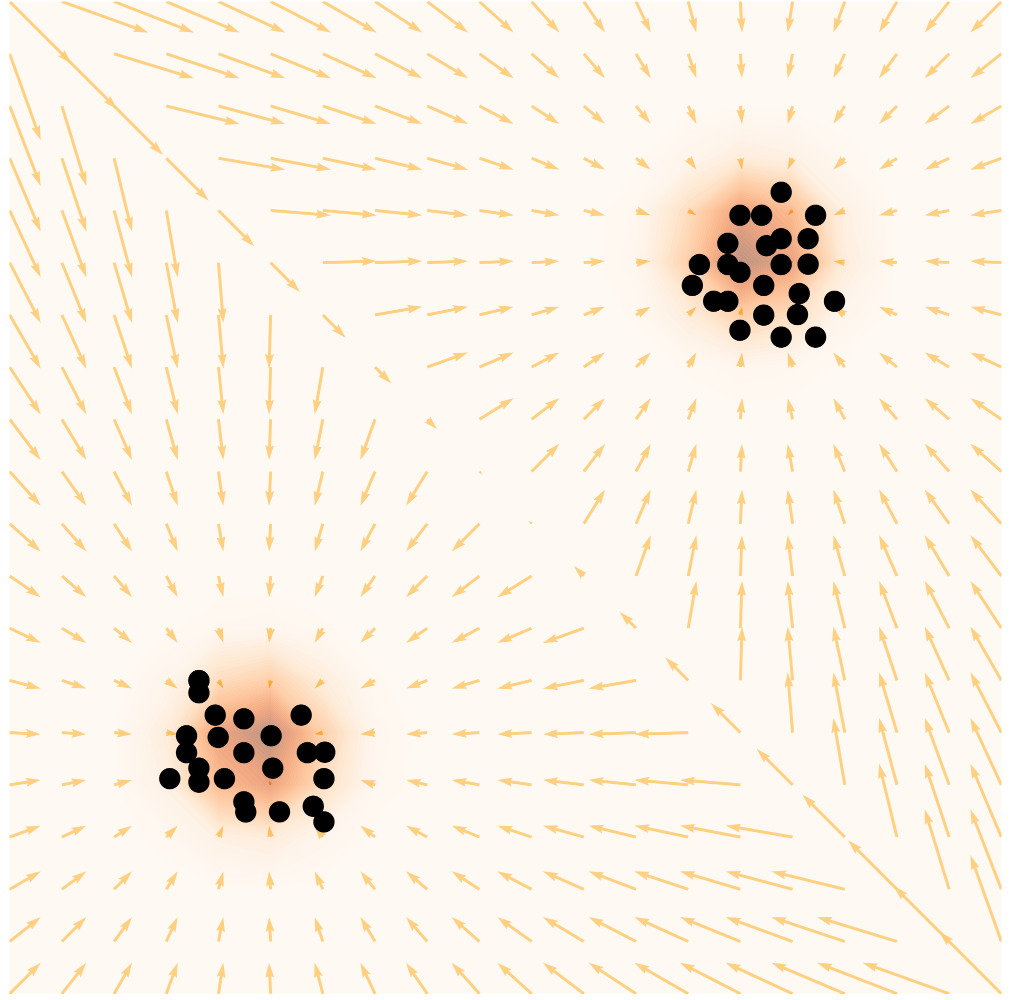
From scores to samples: Langevin MCMC

Score matching



Scores

$s_\theta(x) \approx \nabla_x \log p_{data}(x)$

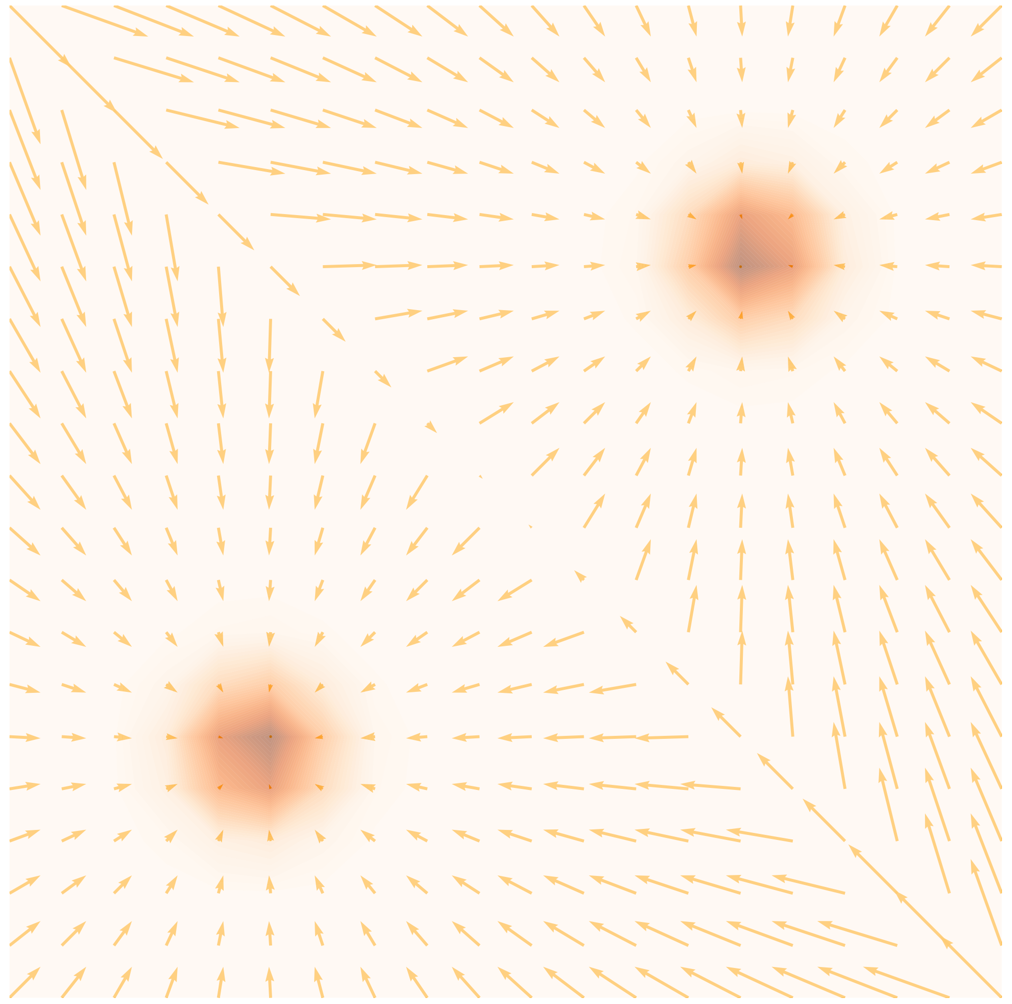


Follow the score

$$\tilde{x}_{t+1} = \tilde{x}_t + \frac{\tau}{2} s_\theta(\tilde{x}_t)$$

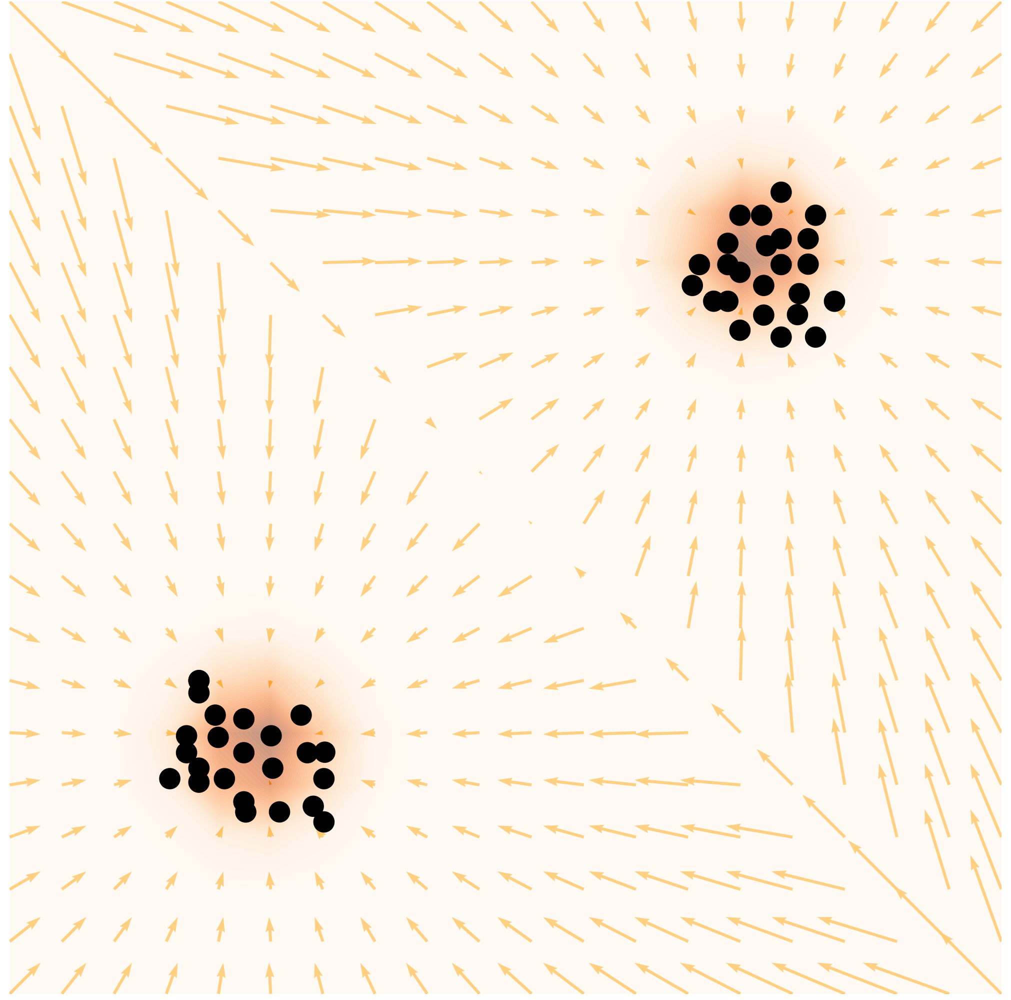
From scores to samples: Langevin MCMC

Score matching



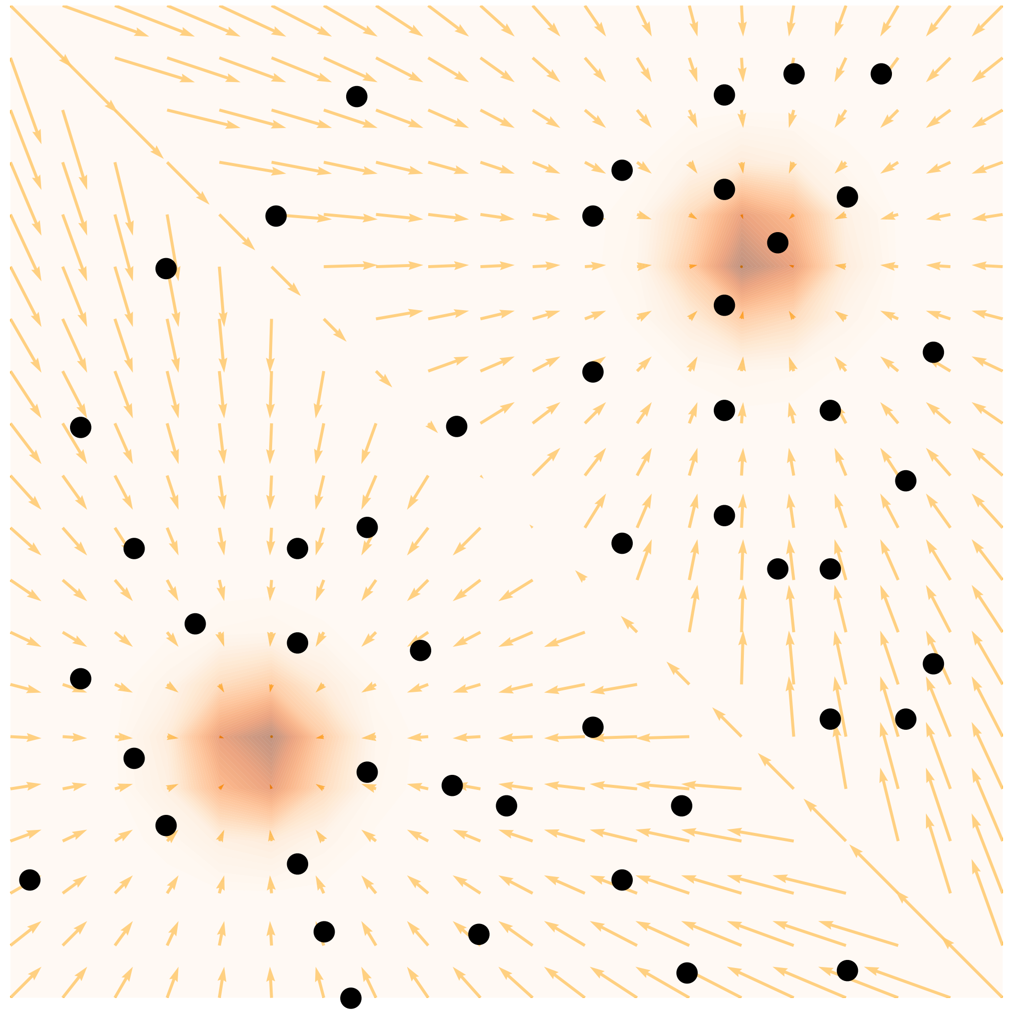
Scores

$s_\theta(x) \approx \nabla_x \log p_{data}(x)$



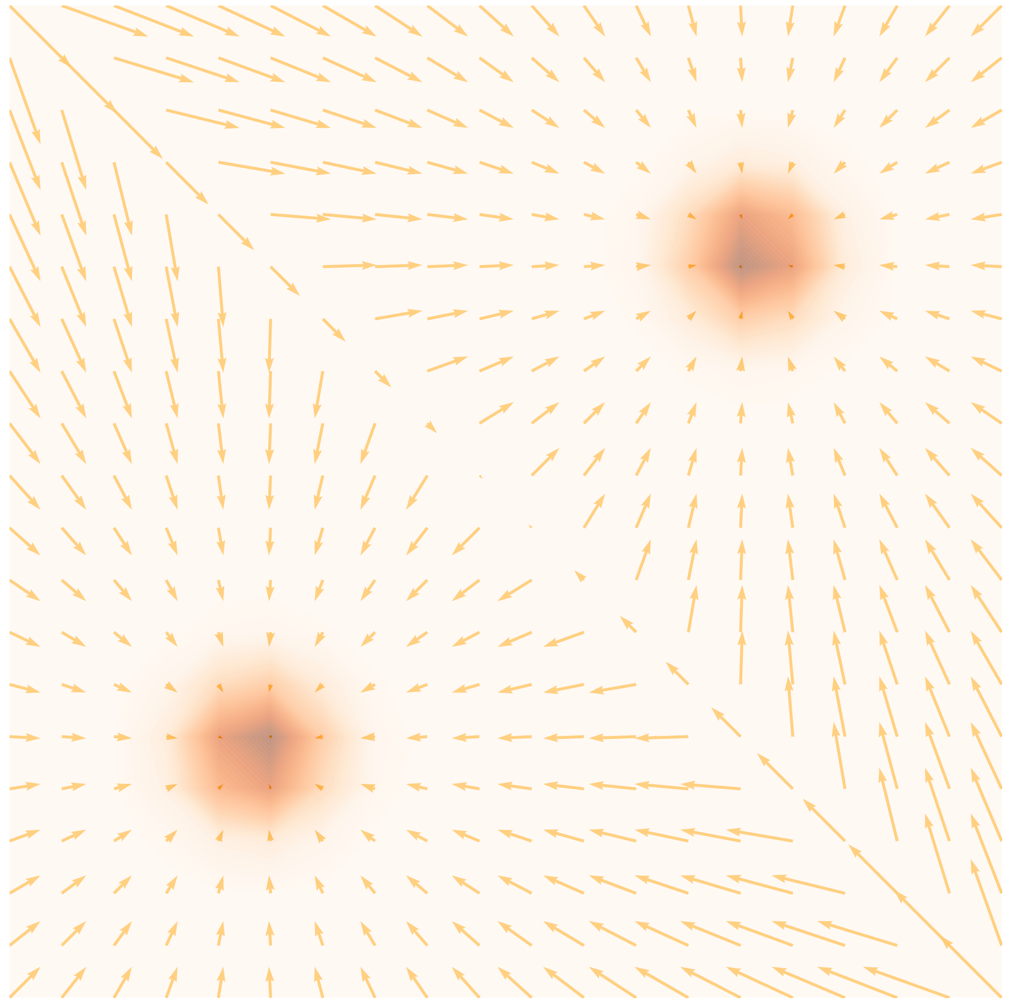
Follow the score

$$\tilde{x}_{t+1} = \tilde{x}_t + \frac{\tau}{2} s_\theta(\tilde{x}_t)$$



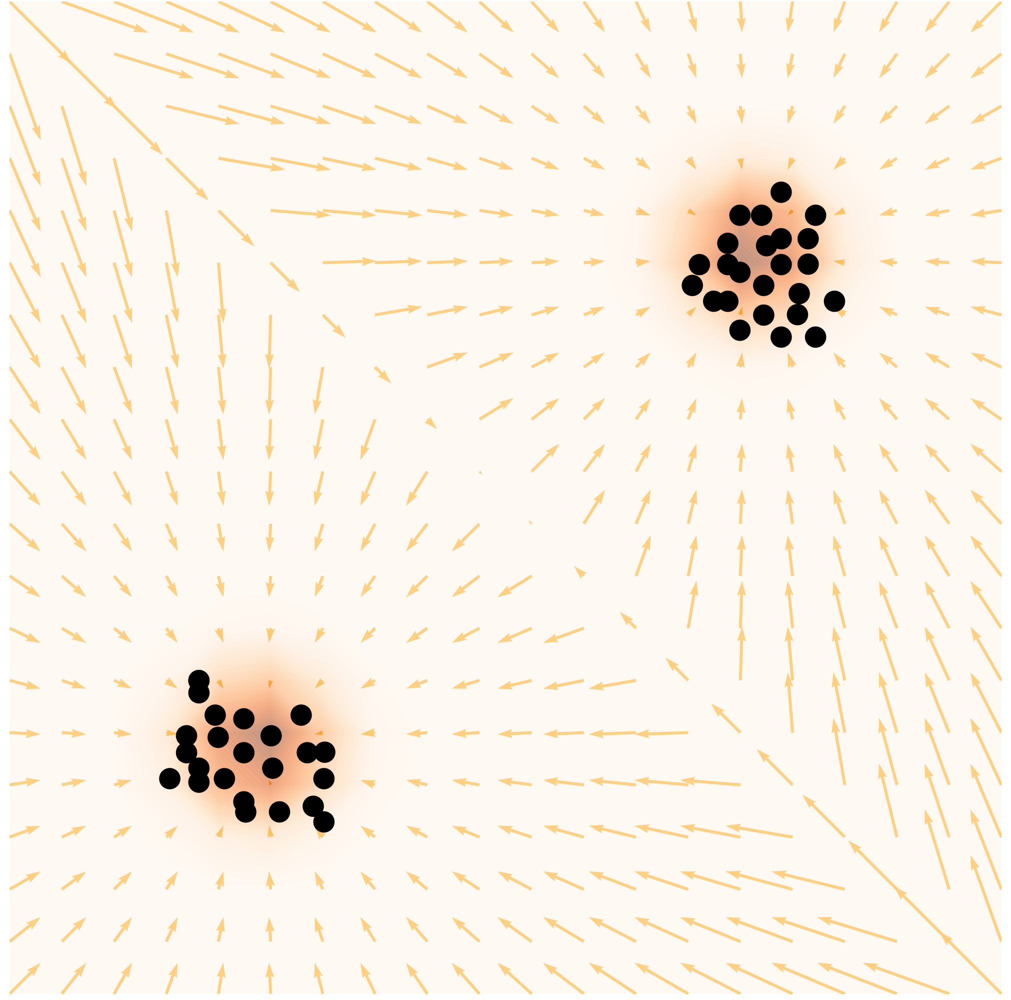
From scores to samples: Langevin MCMC

Score matching



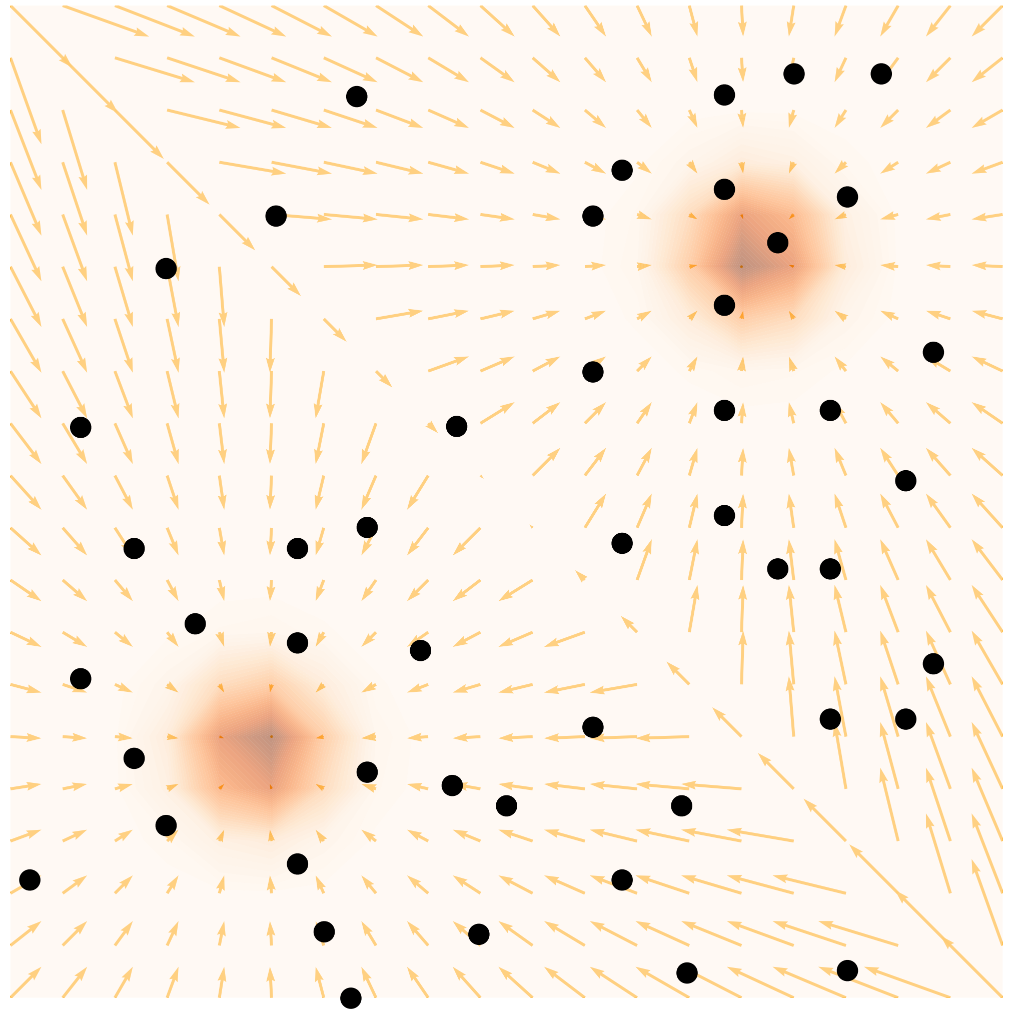
Scores

$$s_\theta(x) \approx \nabla_x \log p_{data}(x)$$



Follow the score

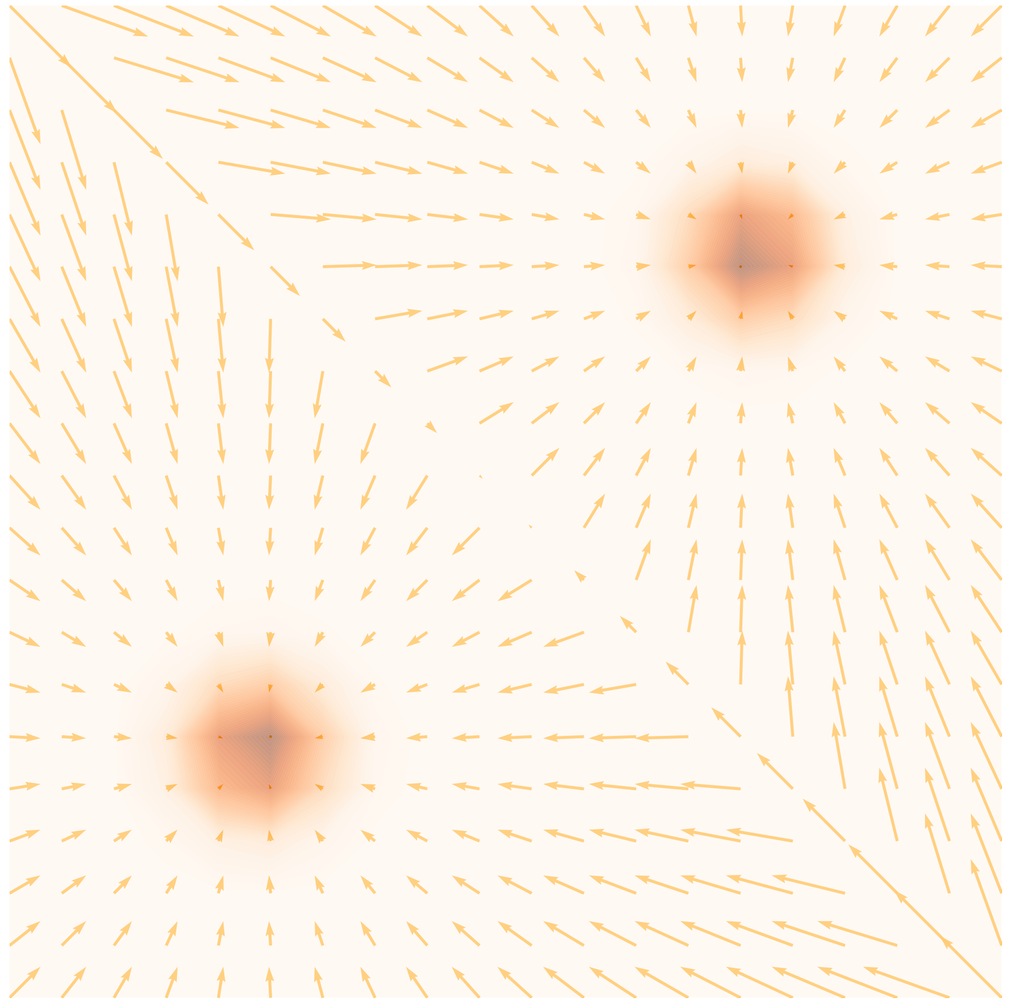
$$\tilde{x}_{t+1} = \tilde{x}_t + \frac{\tau}{2} s_\theta(\tilde{x}_t)$$



Follow the noisy score

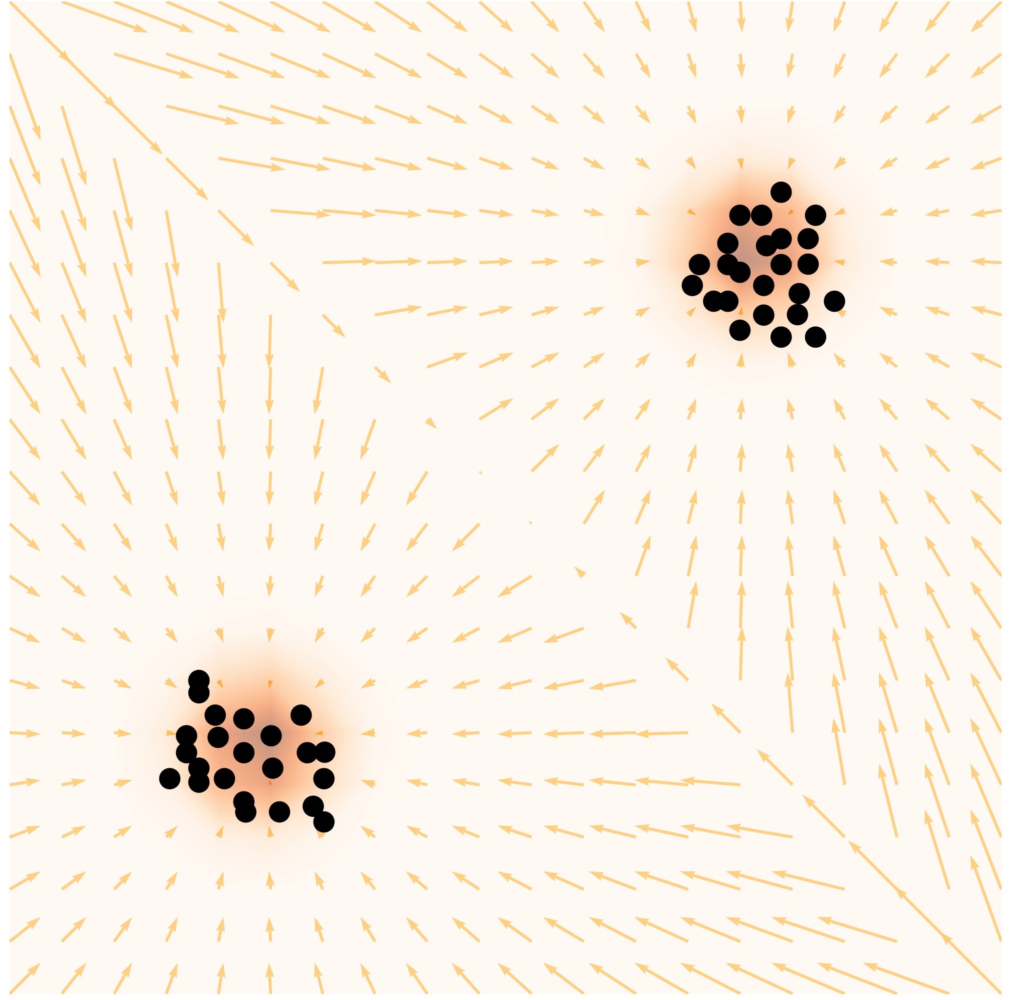
From scores to samples: Langevin MCMC

Score matching →



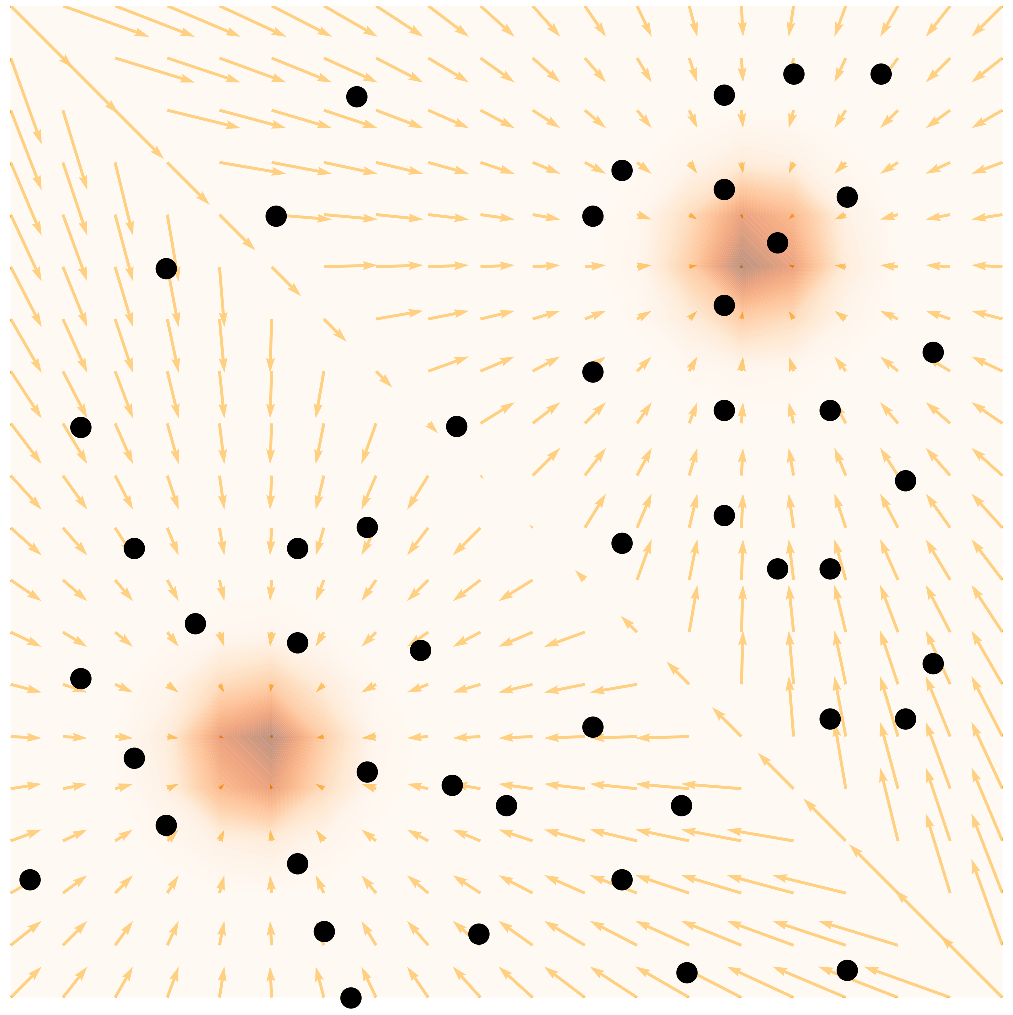
Scores

$s_\theta(x) \approx \nabla_x \log p_{data}(x)$



Follow the score

$\tilde{x}_{t+1} = \tilde{x}_t + \frac{\tau}{2} s_\theta(\tilde{x}_t)$

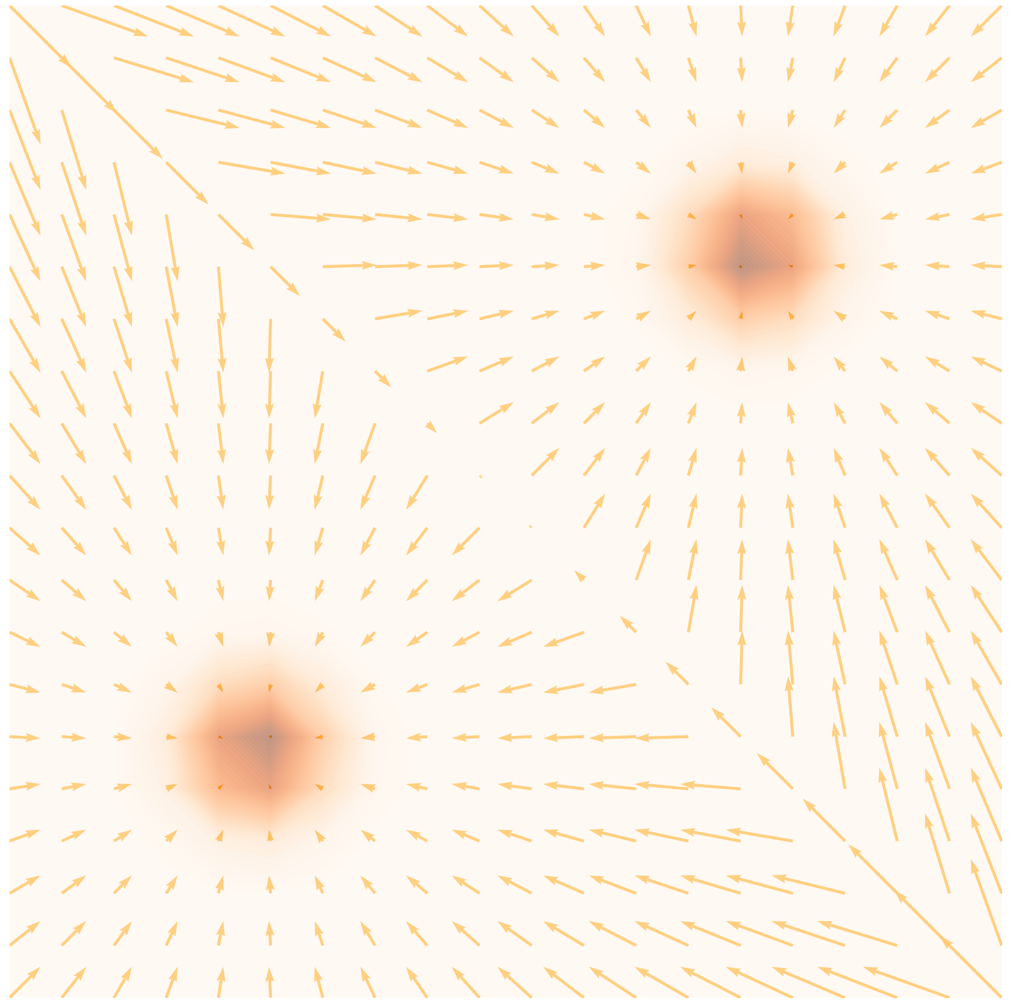


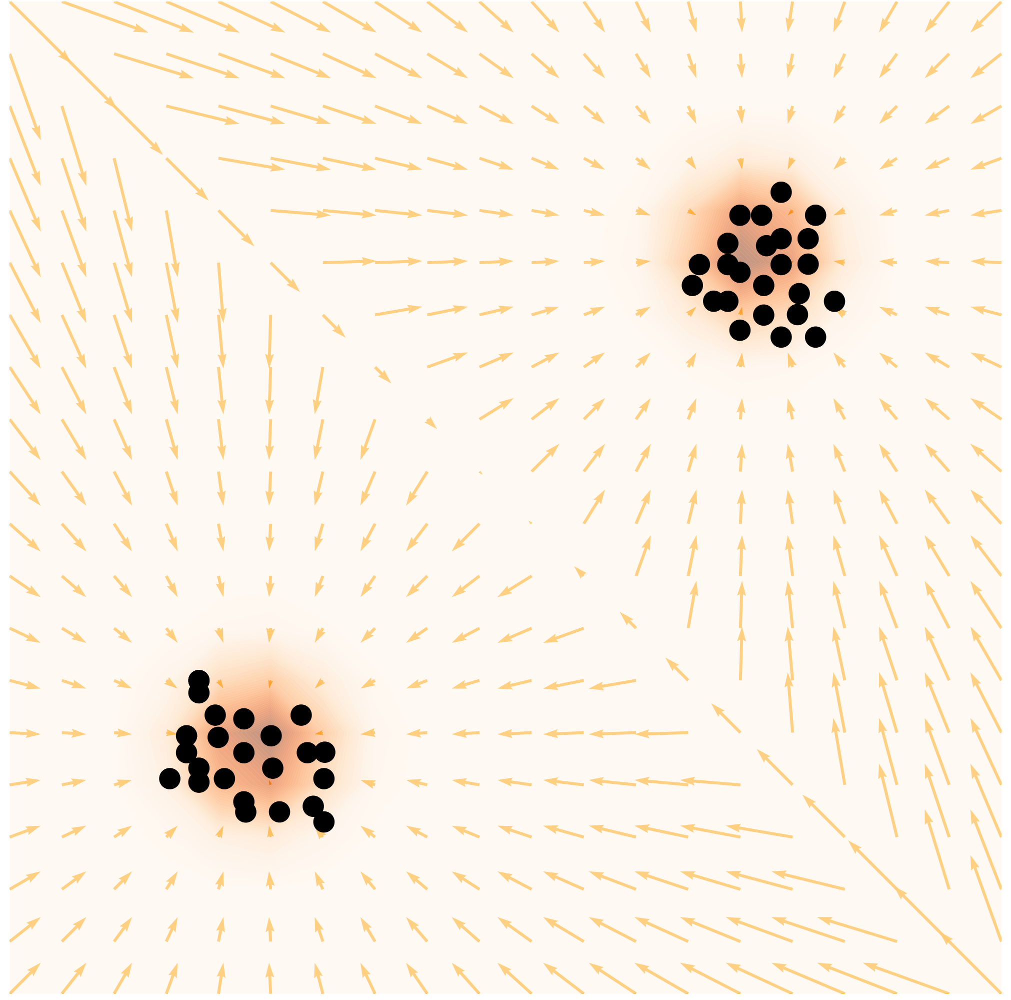
Follow the noisy score

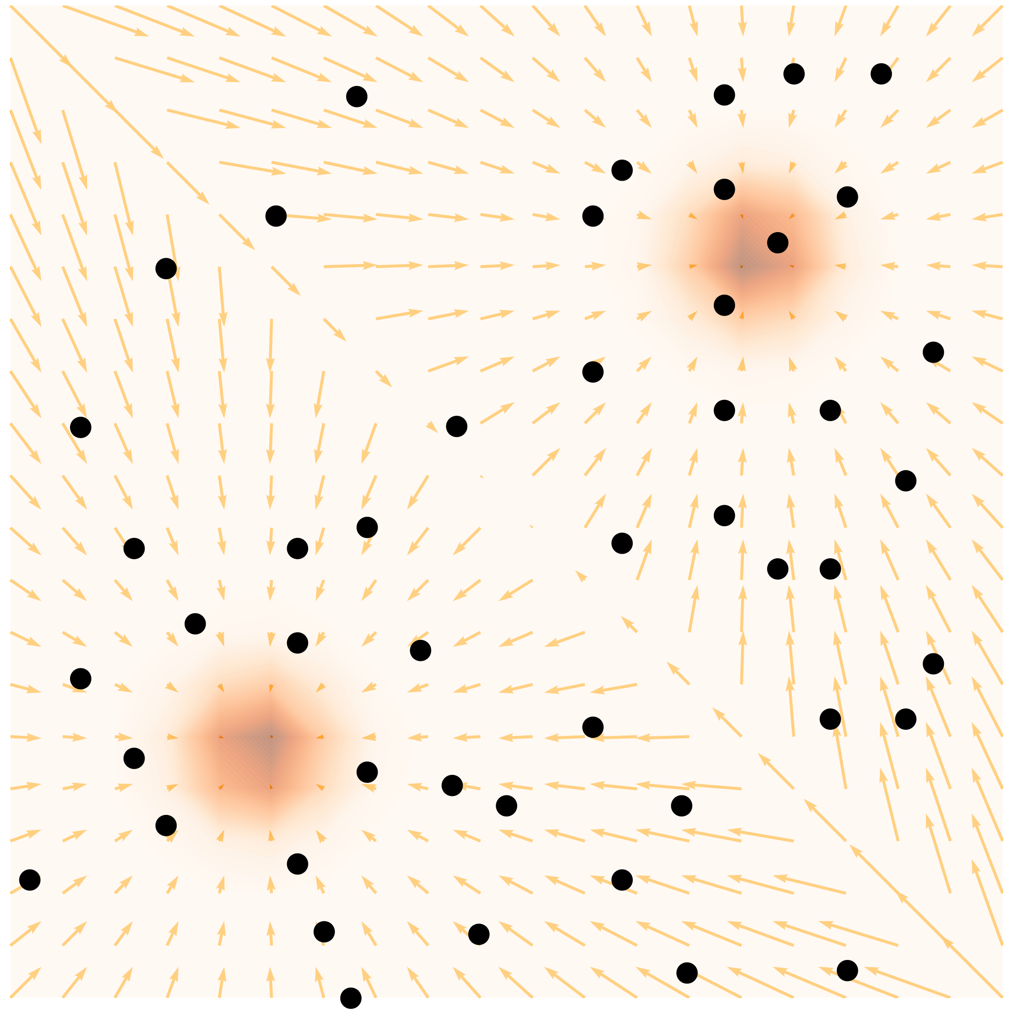
$\tilde{x}_{t+1} = \tilde{x}_t + \frac{\tau}{2} s_\theta(\tilde{x}_t) + \sqrt{2\tau}\epsilon$

From scores to samples: Langevin MCMC

Score matching







Scores

$s_\theta(x) \approx \nabla_x \log p_{data}(x)$

Follow the score

$$\tilde{x}_{t+1} = \tilde{x}_t + \frac{\tau}{2} s_\theta(\tilde{x}_t)$$

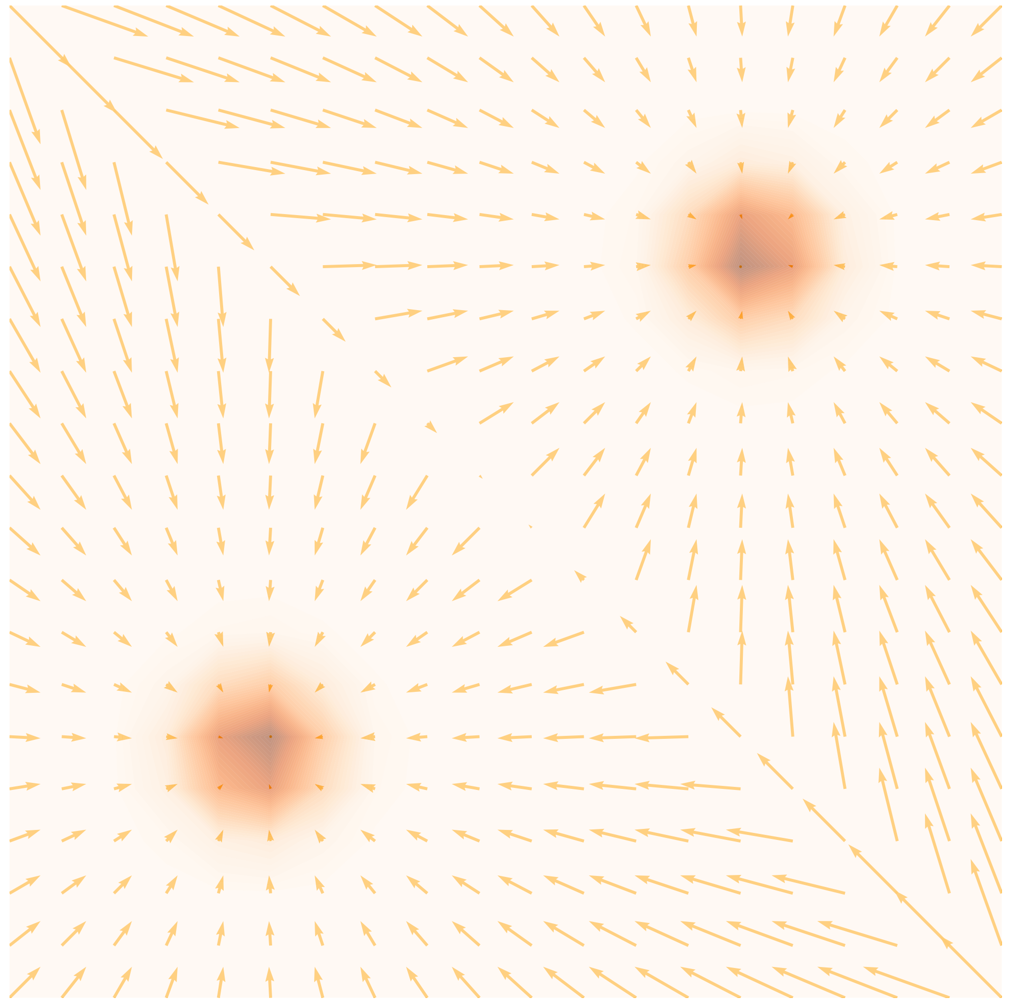
Follow the noisy score

$$\tilde{x}_{t+1} = \tilde{x}_t + \frac{\tau}{2} s_\theta(\tilde{x}_t) + \sqrt{2\tau}\epsilon$$

$$\epsilon \sim \mathcal{N}(0,1)$$

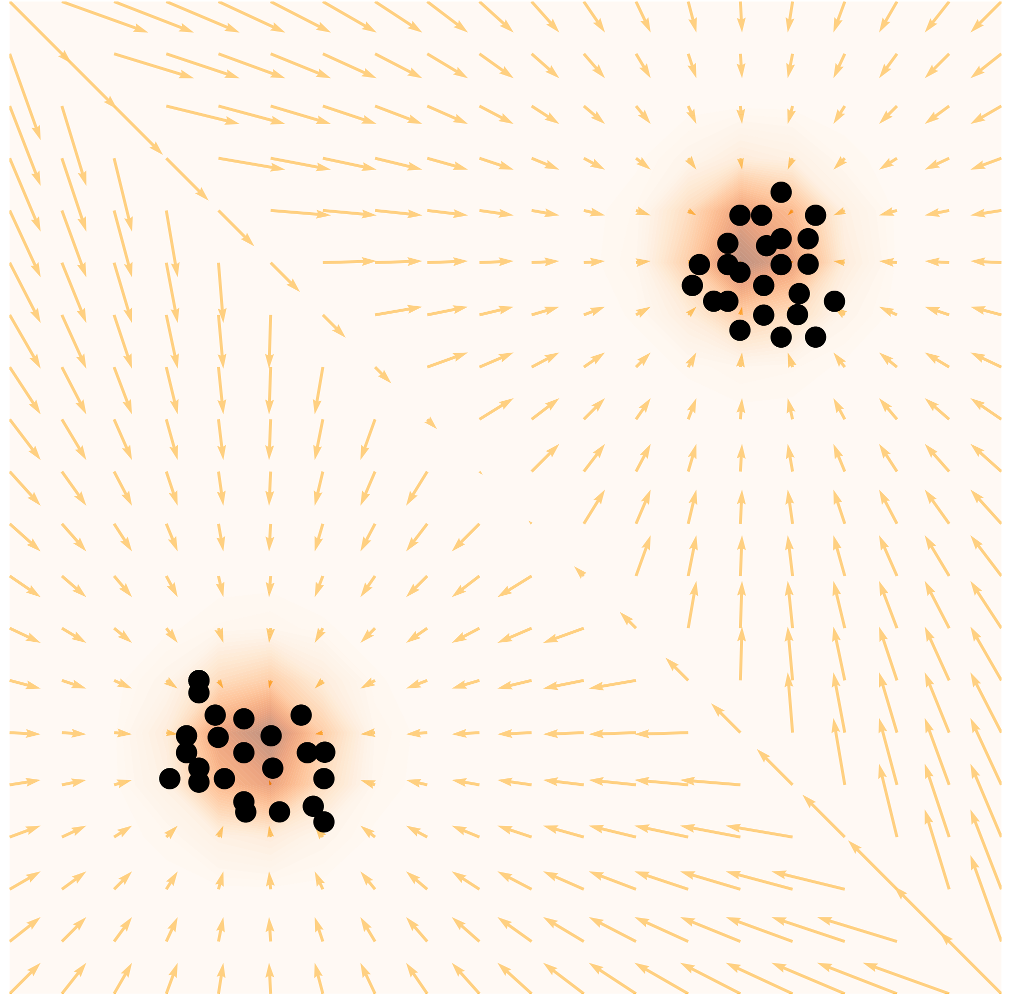
From scores to samples: Langevin MCMC

Score matching



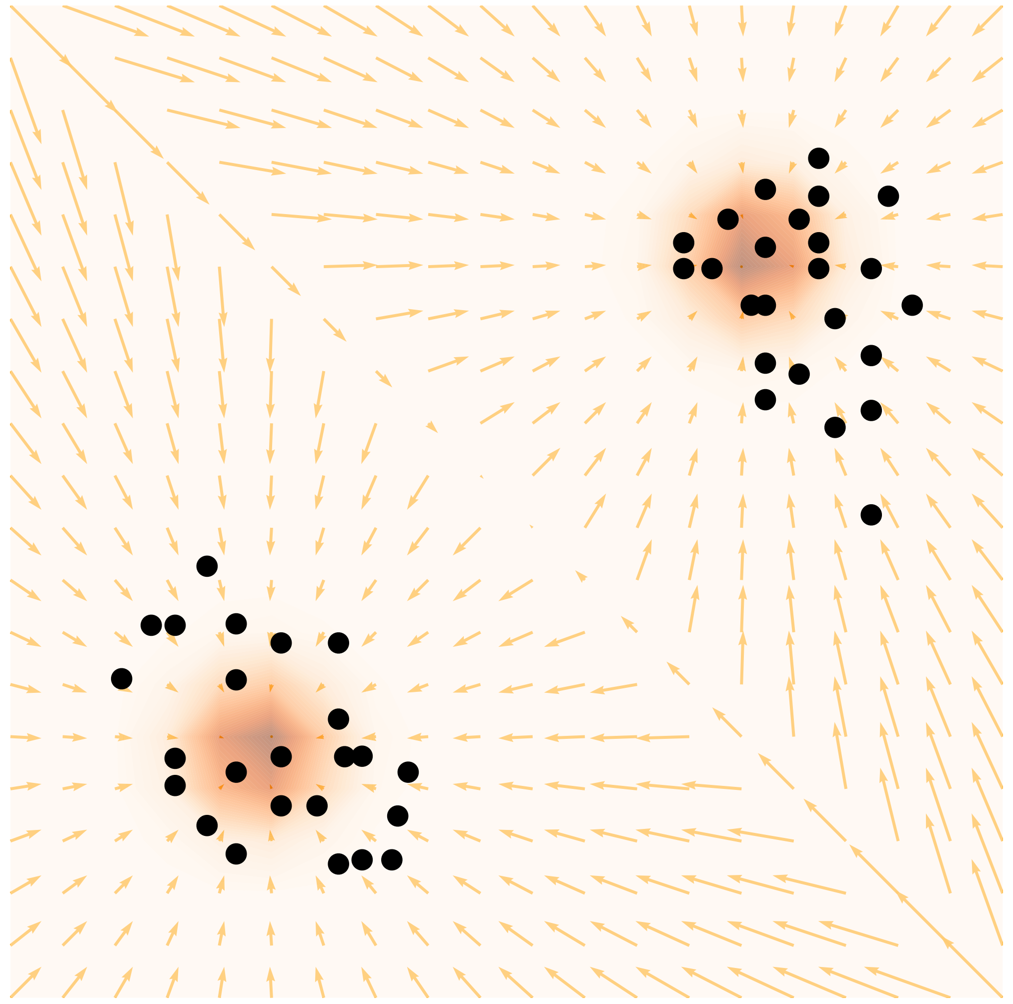
Scores

$$s_\theta(x) \approx \nabla_x \log p_{data}(x)$$



Follow the score

$$\tilde{x}_{t+1} = \tilde{x}_t + \frac{\tau}{2} s_\theta(\tilde{x}_t)$$

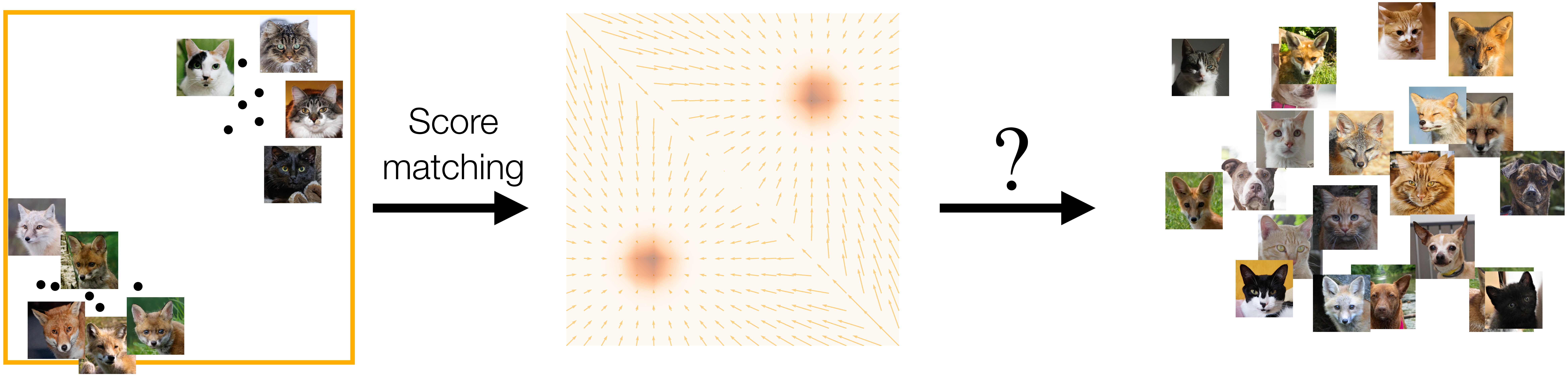


Follow the noisy score

$$\tilde{x}_{t+1} = \tilde{x}_t + \frac{\tau}{2} s_\theta(\tilde{x}_t) \boxed{+ \sqrt{2\tau}\epsilon}$$

$$\epsilon \sim \mathcal{N}(0,1)$$

Sampling in score-based generative models



Score matching

?

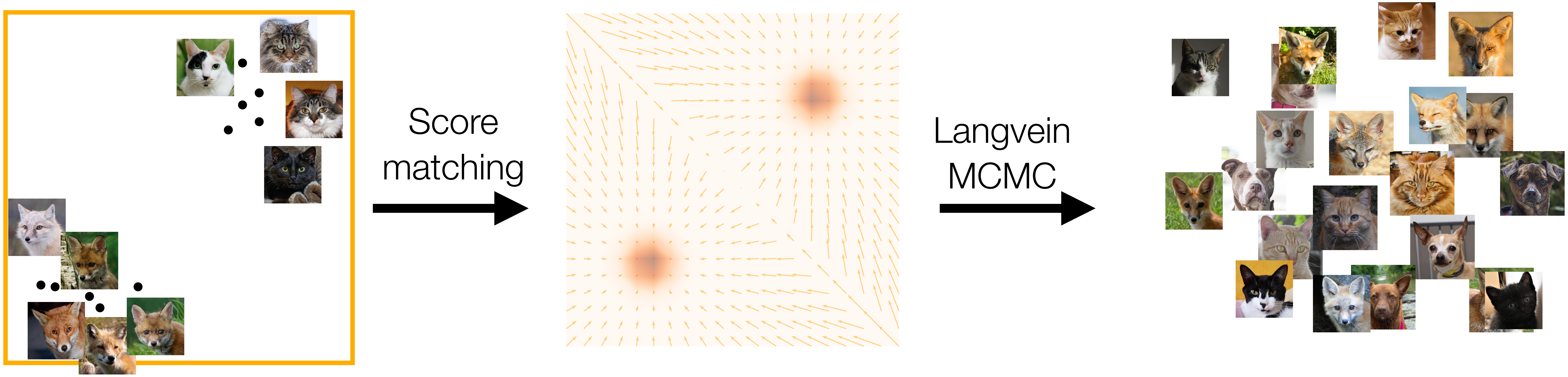
Data samples

Scores

$s_\theta(x) \approx \nabla_x \log p_{data}(x)$

New samples

Sampling in score-based generative models



Score matching

Langvein MCMC

Data samples

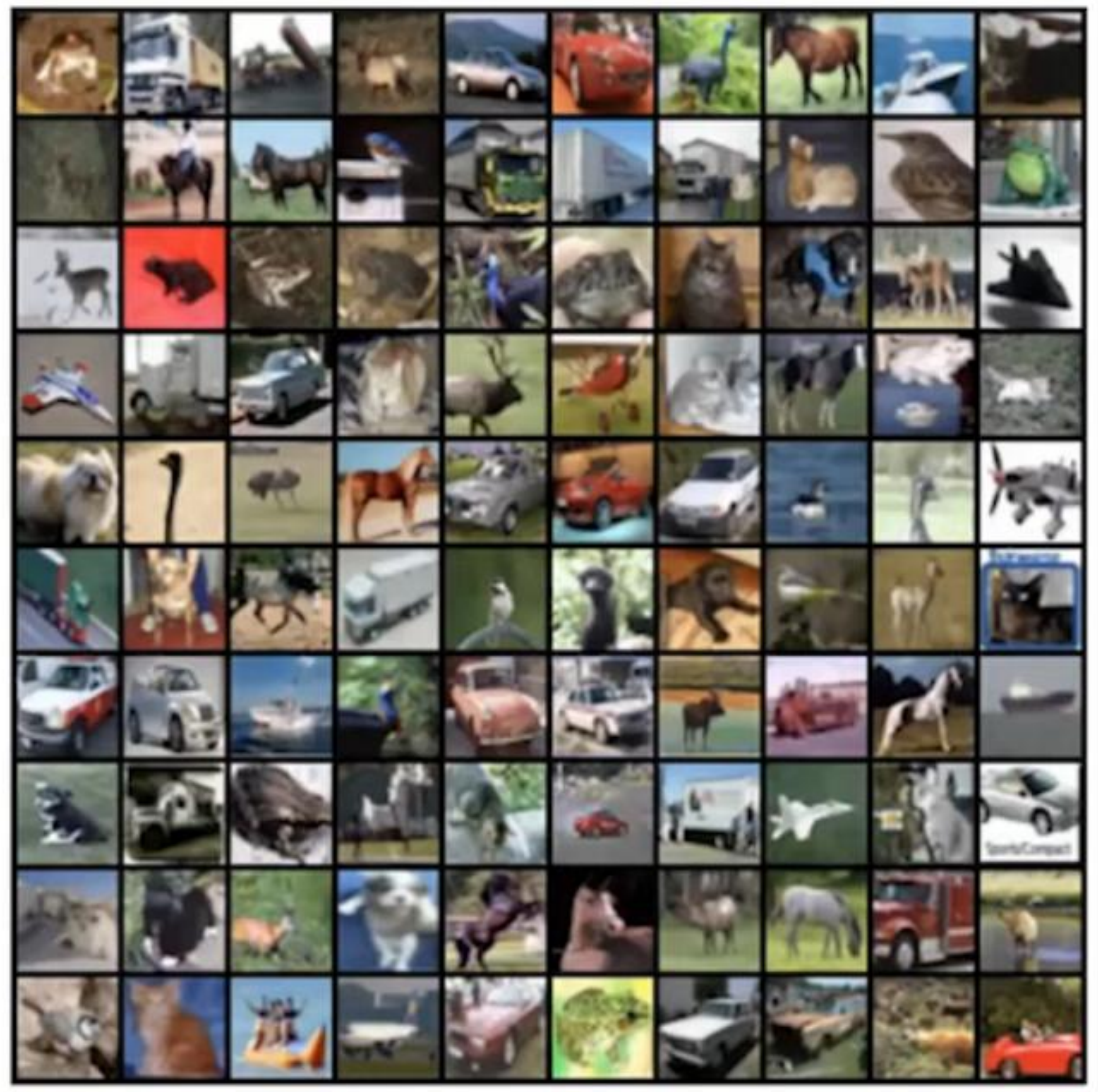
Scores

$$s_\theta(x) \approx \nabla_x \log p_{data}(x)$$

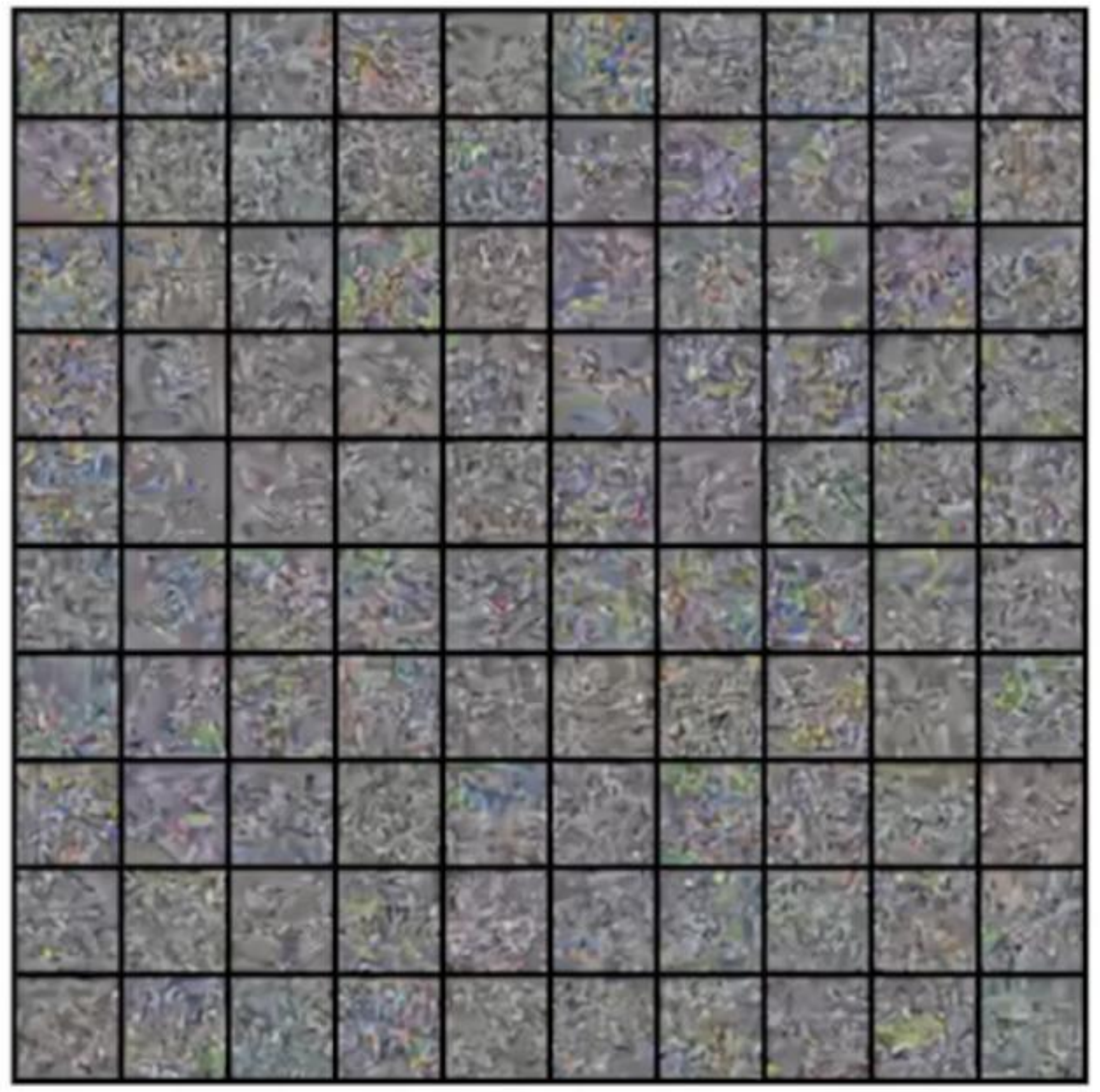
New samples

Issues with score-based generative modeling

- Langevin MCMC process does not work
- We only get noise, and the optimization process get stuck in some local minima



CIFAR-10 data

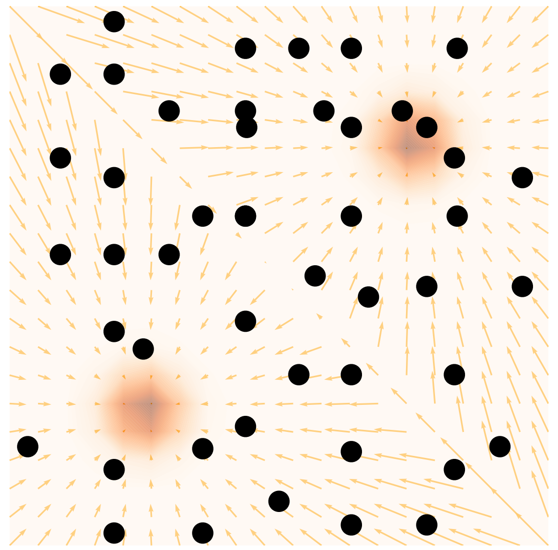


Model samples

How to fix these issues?

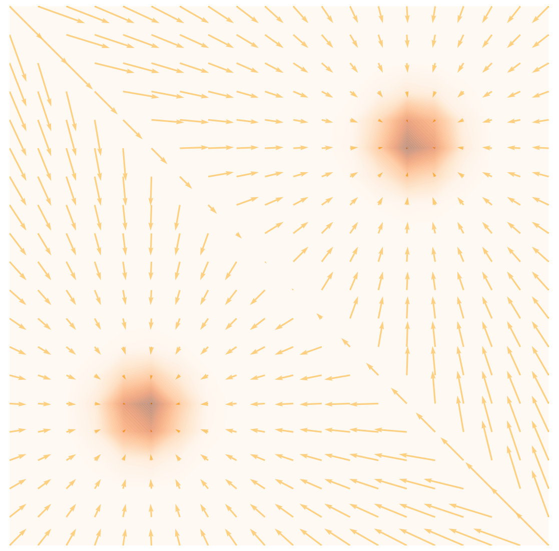
Path to diffusion models

Annealed Langevin dynamics to generate samples

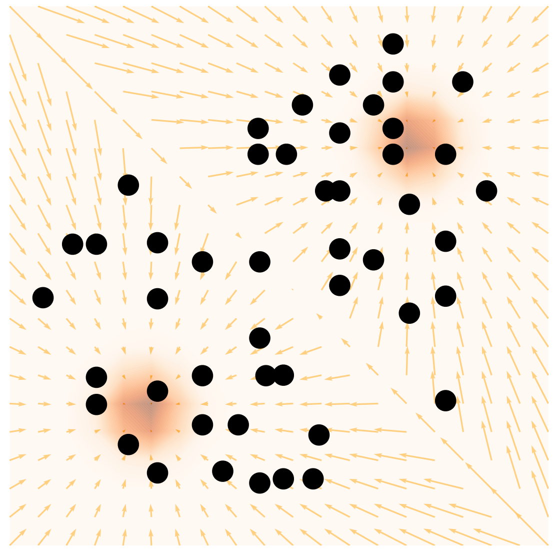


σ_1

Annealed Langevin dynamics to generate samples

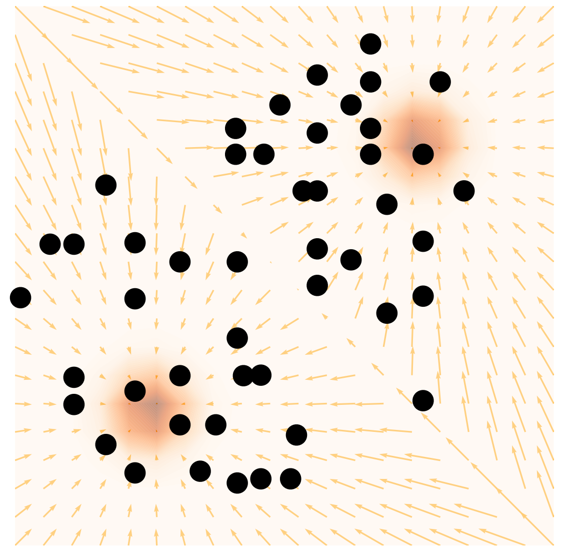


σ_2

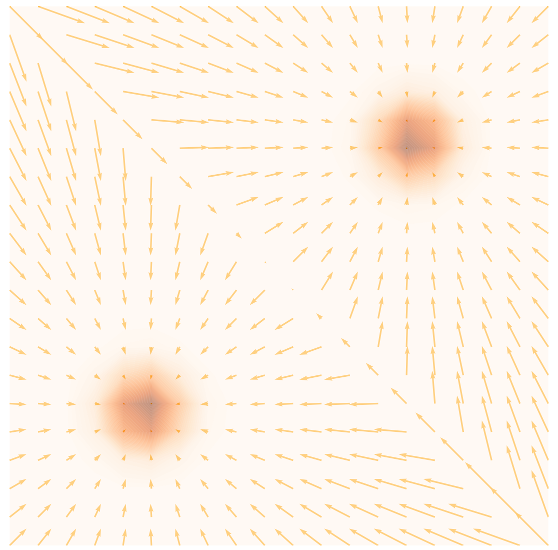


σ_1

Annealed Langevin dynamics to generate samples

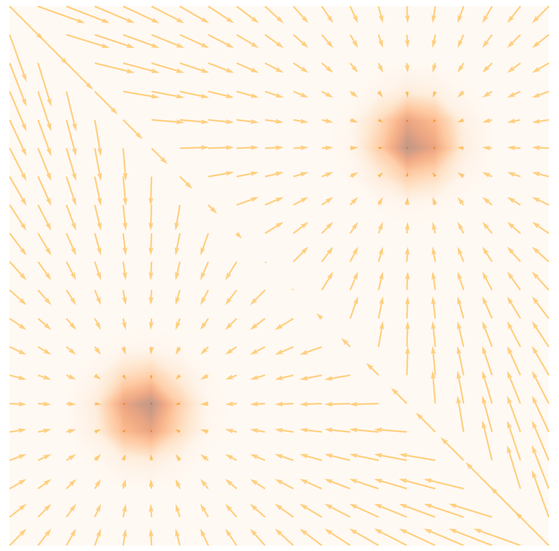


σ_2

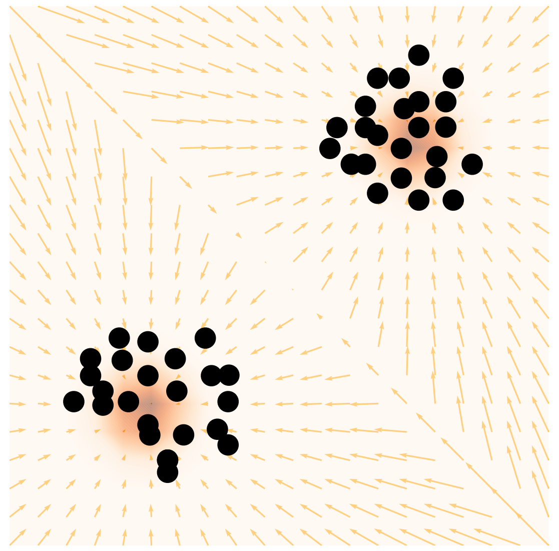


σ_1

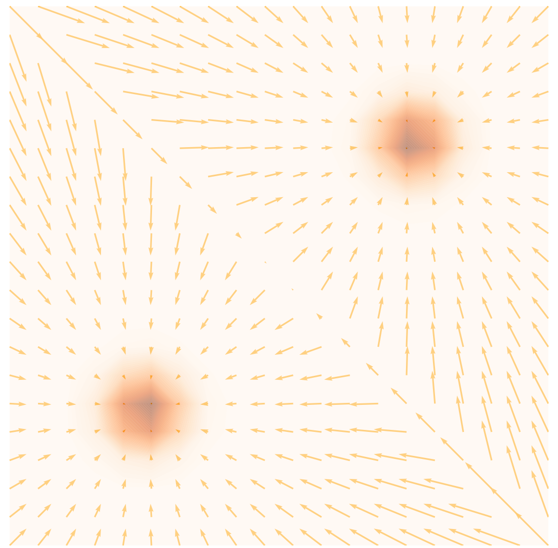
Annealed Langevin dynamics to generate samples



σ_3

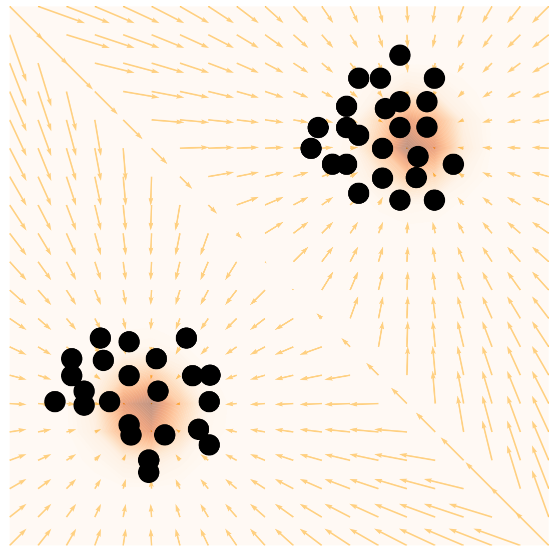


σ_2

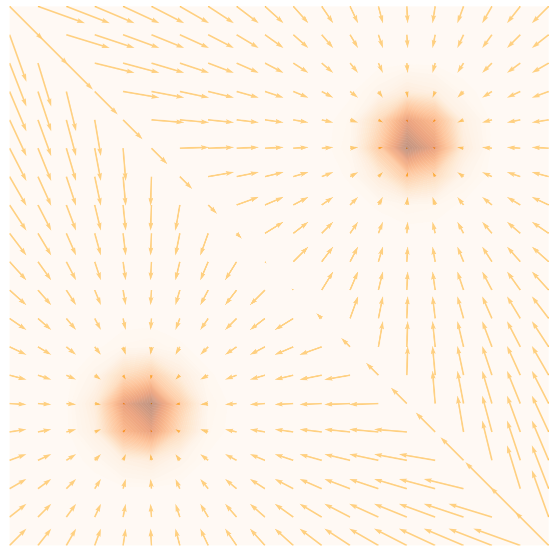


σ_1

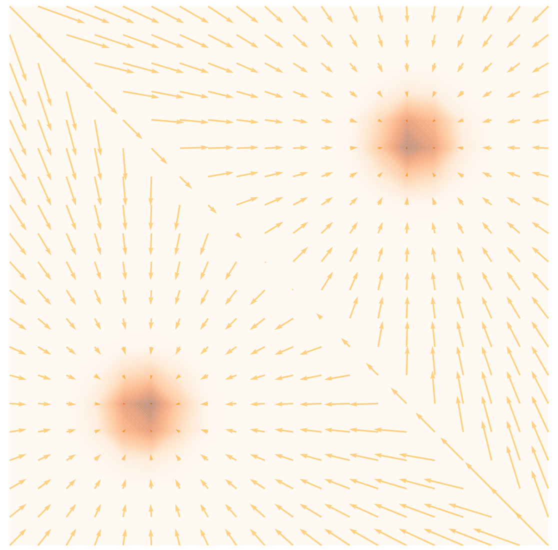
Annealed Langevin dynamics to generate samples



σ_3



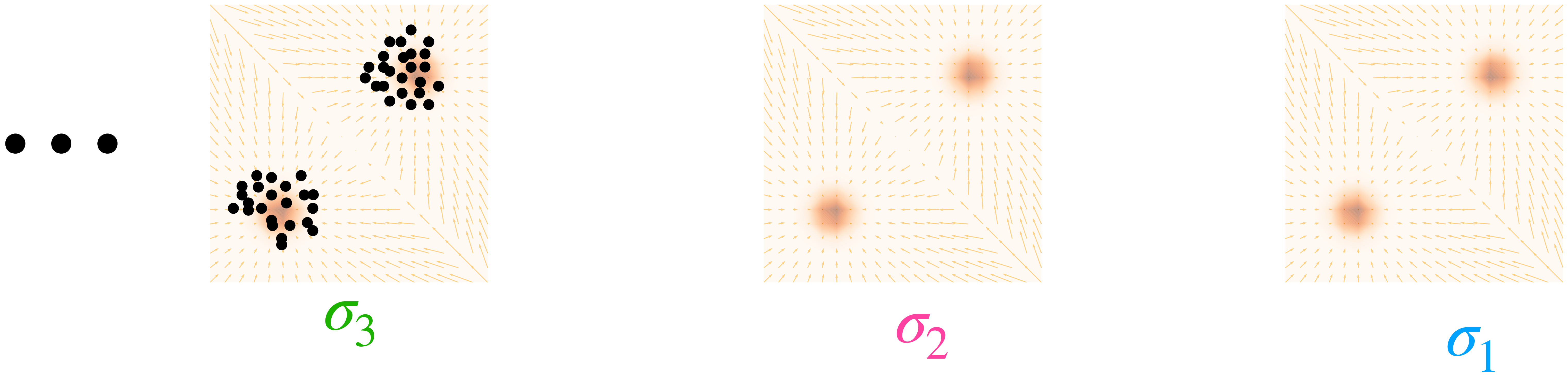
σ_2



σ_1

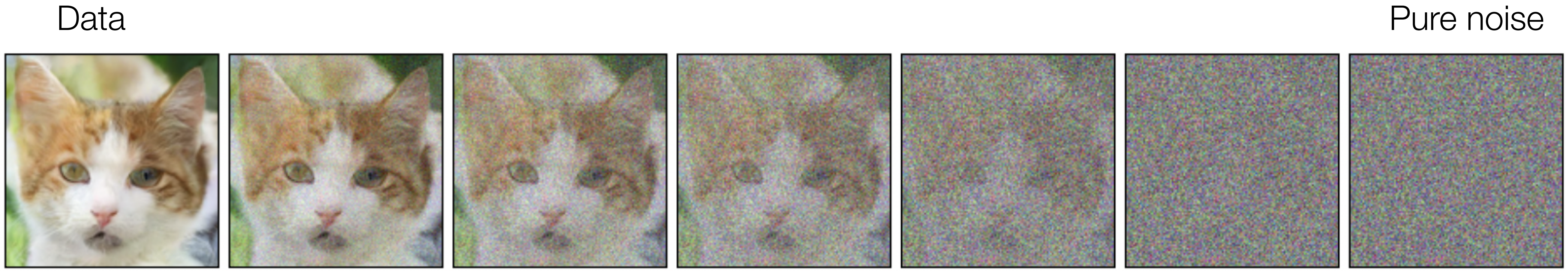
Using multiple noise levels

Annealed Langevin dynamics to generate samples



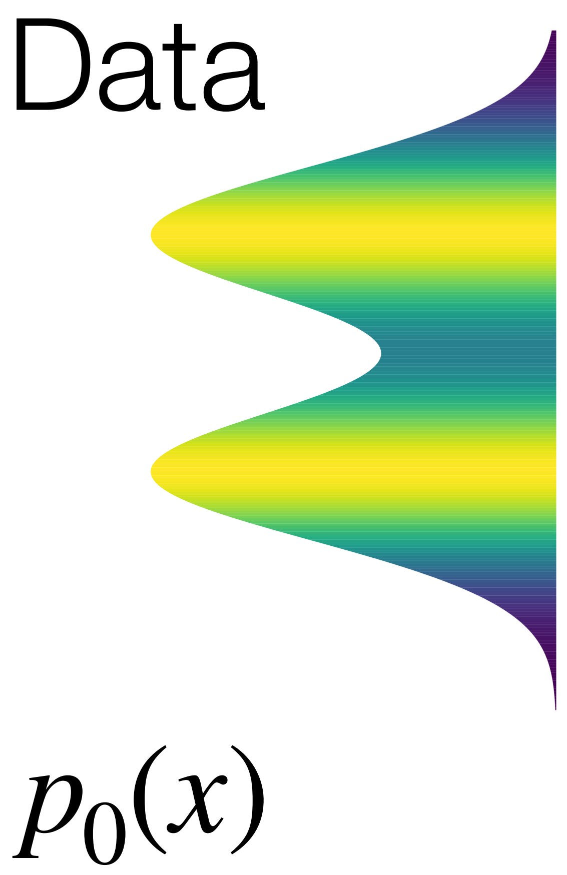
Using multiple noise levels

Path to Diffusion models



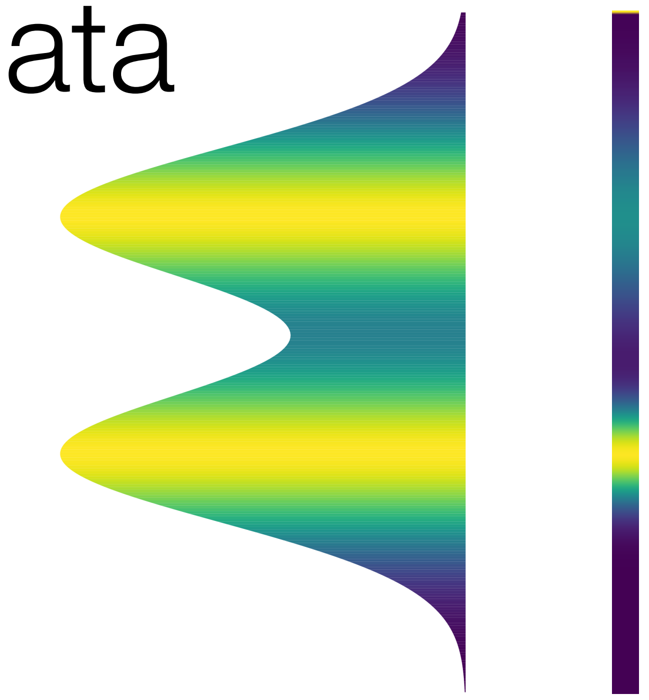
Using multiple noise levels

Using multiple noise levels



Using multiple noise levels

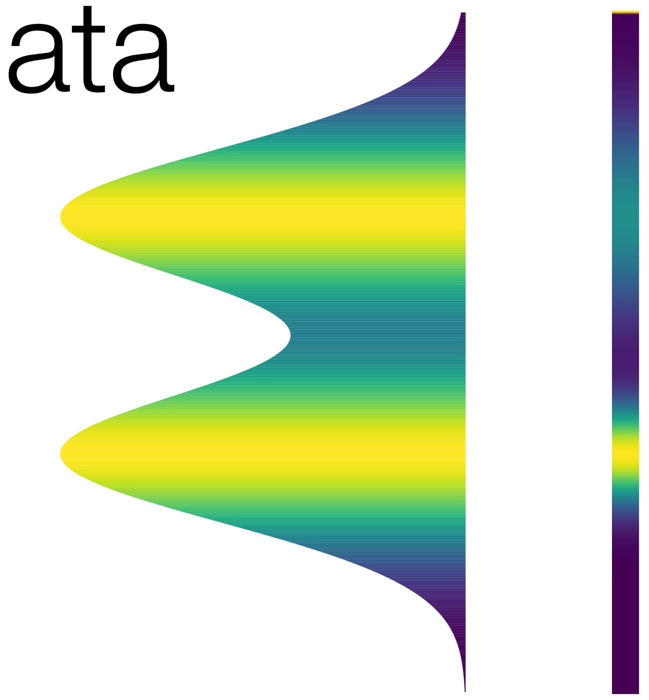
Data



$p_0(x)$

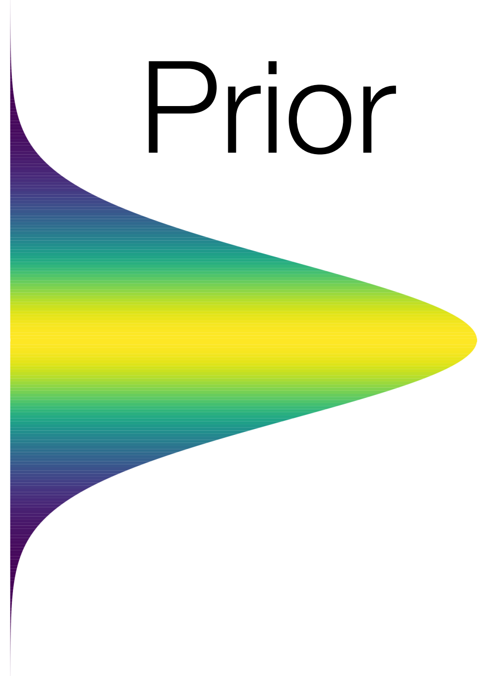
Using multiple noise levels

Data



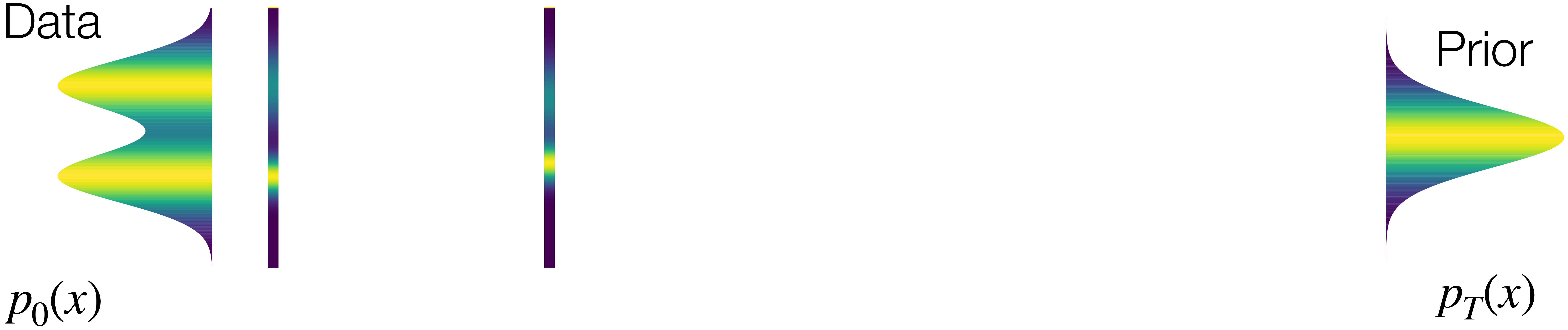
$p_0(x)$

Prior



$p_T(x)$

Using multiple noise levels



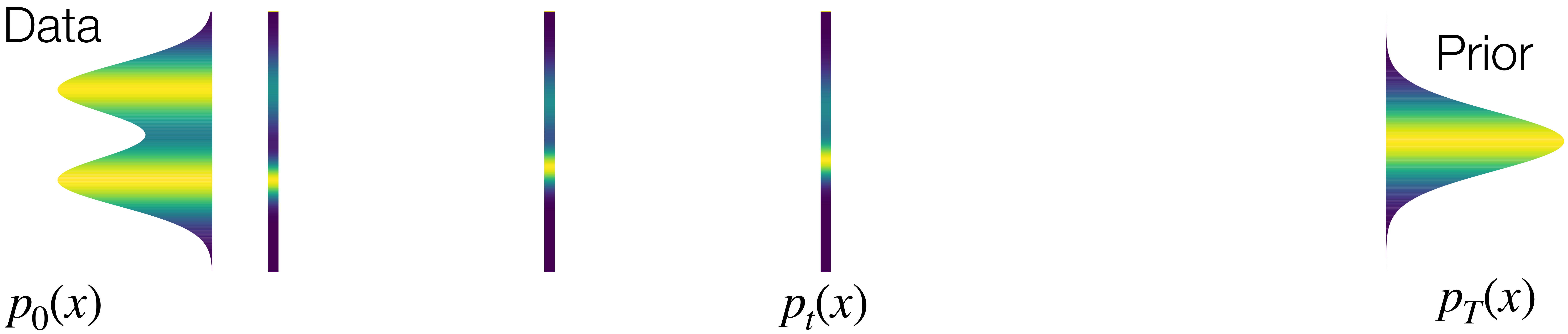
Data

$p_0(x)$

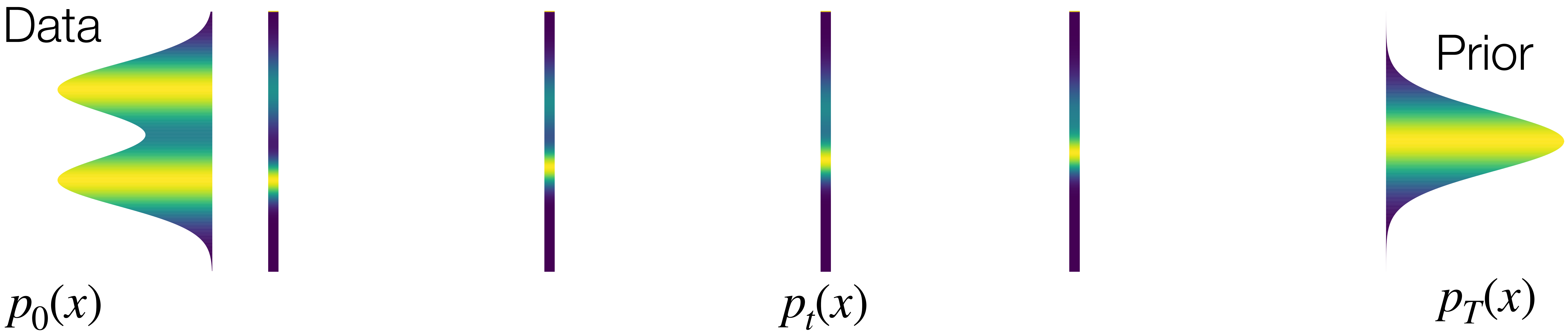
Prior

$p_T(x)$

Using multiple noise levels



Using multiple noise levels



Data

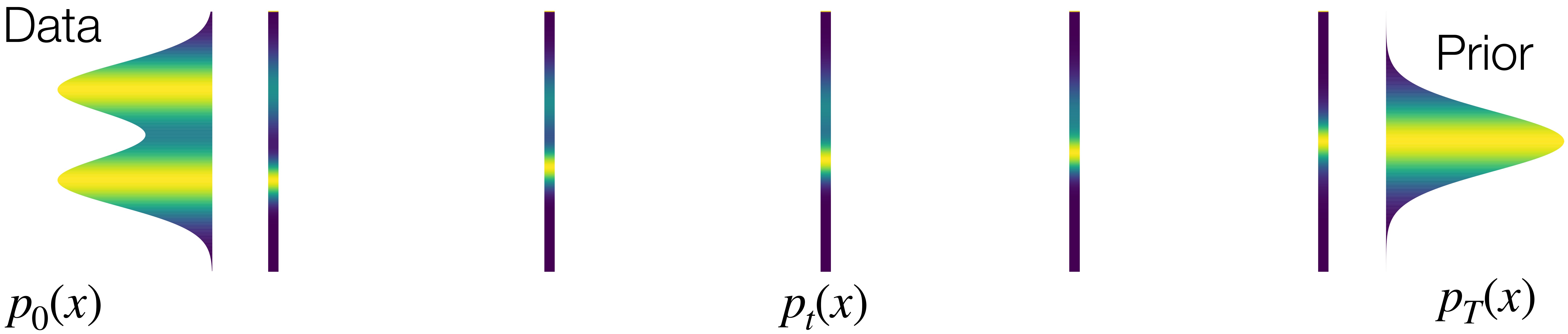
Prior

$p_0(x)$

$p_t(x)$

$p_T(x)$

Using multiple noise levels



What happens when we have infinite noise levels?

From VAEs to Diffusion models

Variational Autoencoders (VAEs)

Energy-based models (EBMs)

MCMC methods for EBMs

Score-based Generative models (SBGMs)

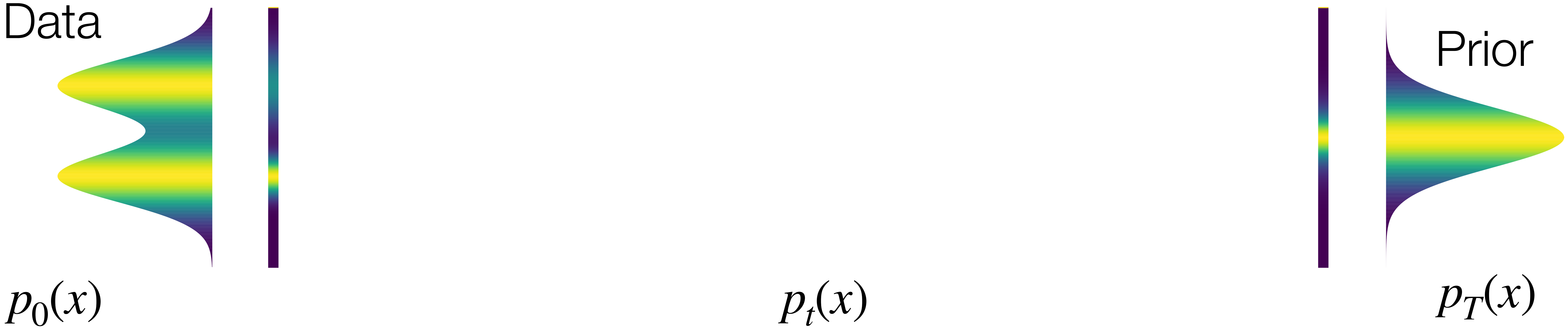
MCMC methods for SBGMs

SDE-based diffusion models

SDE-based diffusion models

Perturbing data with stochastic processes

Perturbing data with stochastic processes



Data

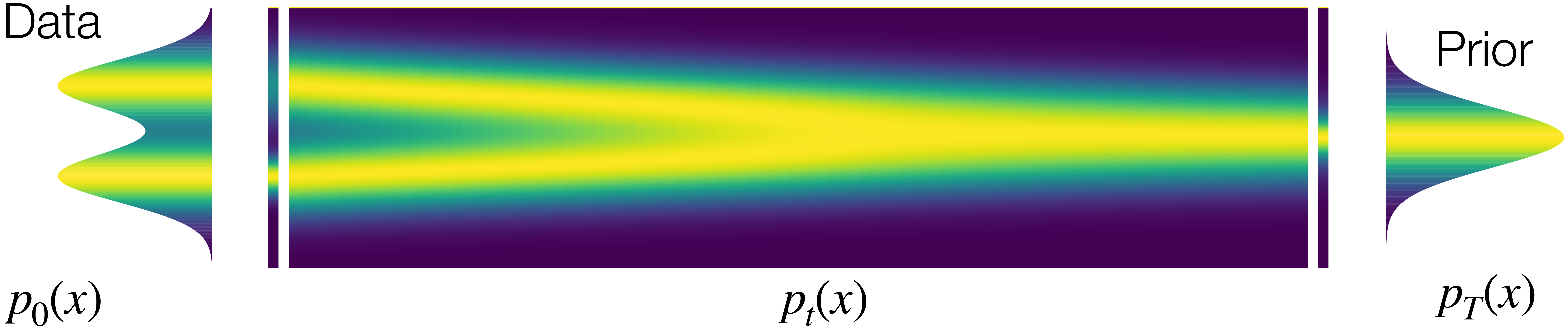
Prior

$p_0(x)$

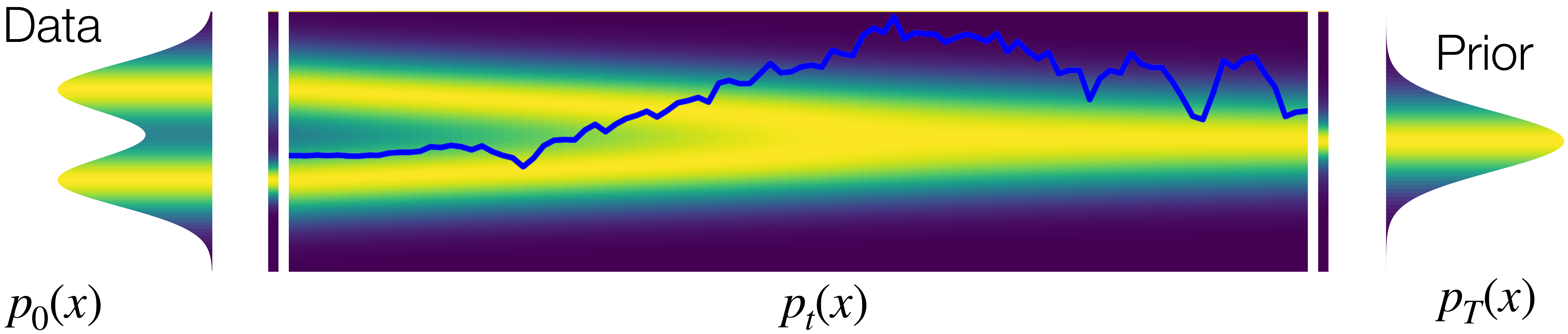
$p_t(x)$

$p_T(x)$

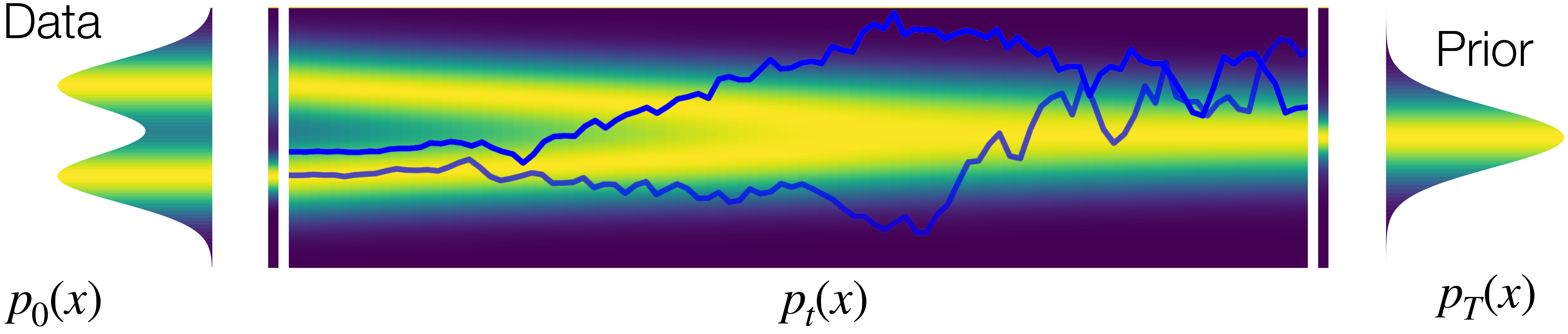
Perturbing data with stochastic processes



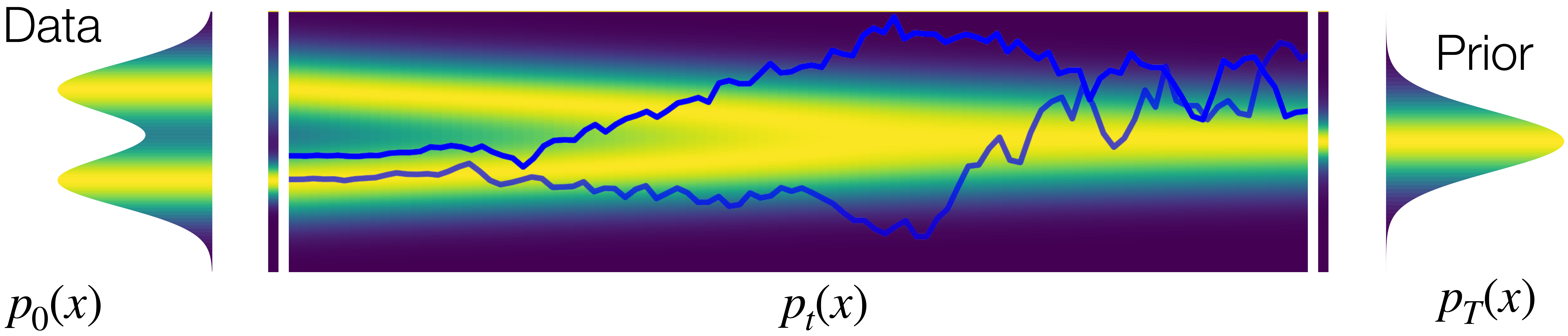
Perturbing data with stochastic processes



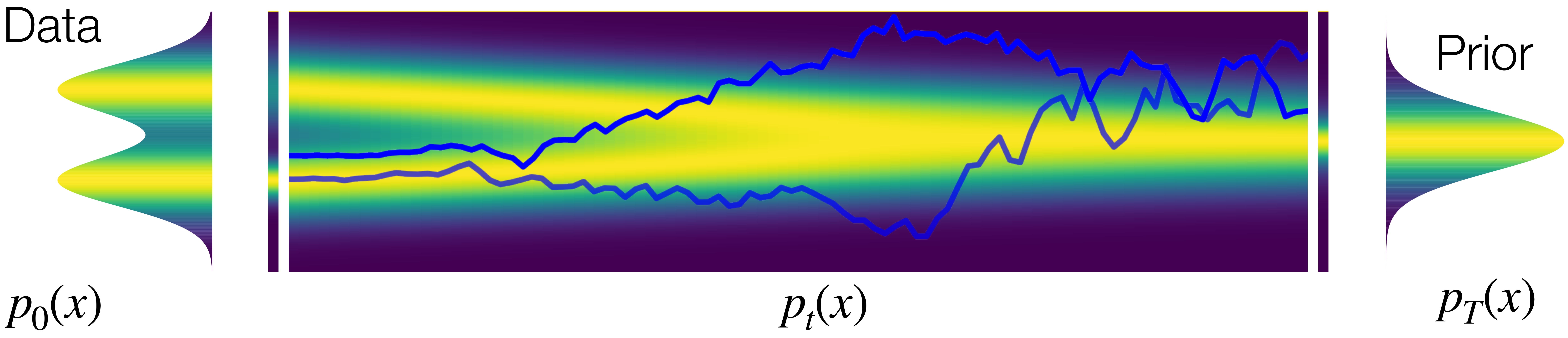
Perturbing data with stochastic processes



Perturbing data with stochastic processes

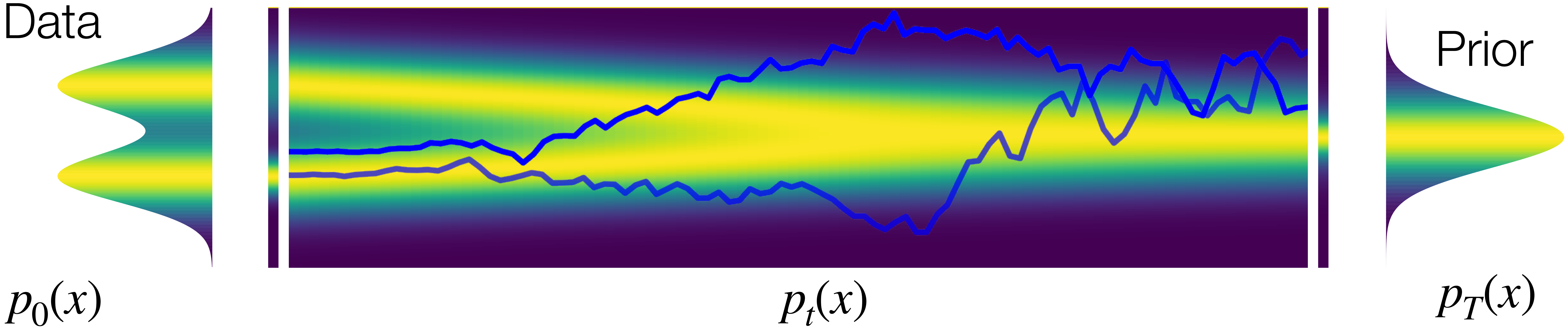


Perturbing data with stochastic processes



Stochastic processes

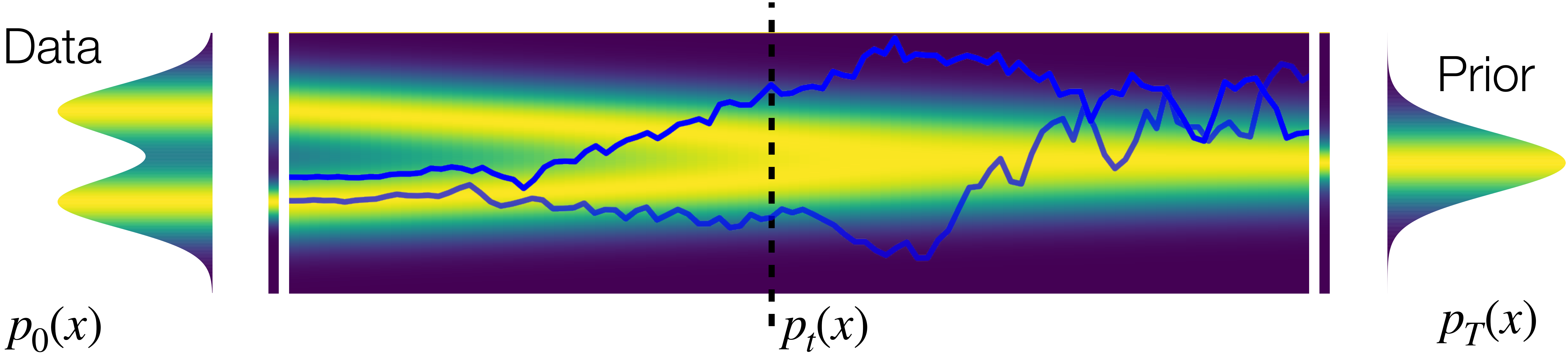
Perturbing data with stochastic processes



Stochastic processes

$$\{\mathbf{x}_t\}_{t\in[0,T]}$$

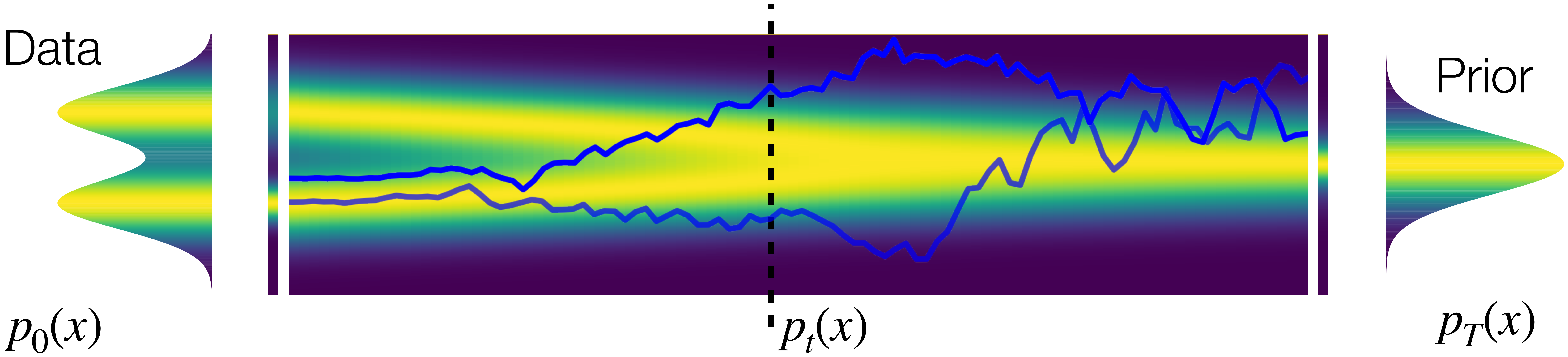
Perturbing data with stochastic processes



Stochastic processes

$$\{\mathbf{x}_t\}_{t\in[0,T]}$$

Perturbing data with stochastic processes

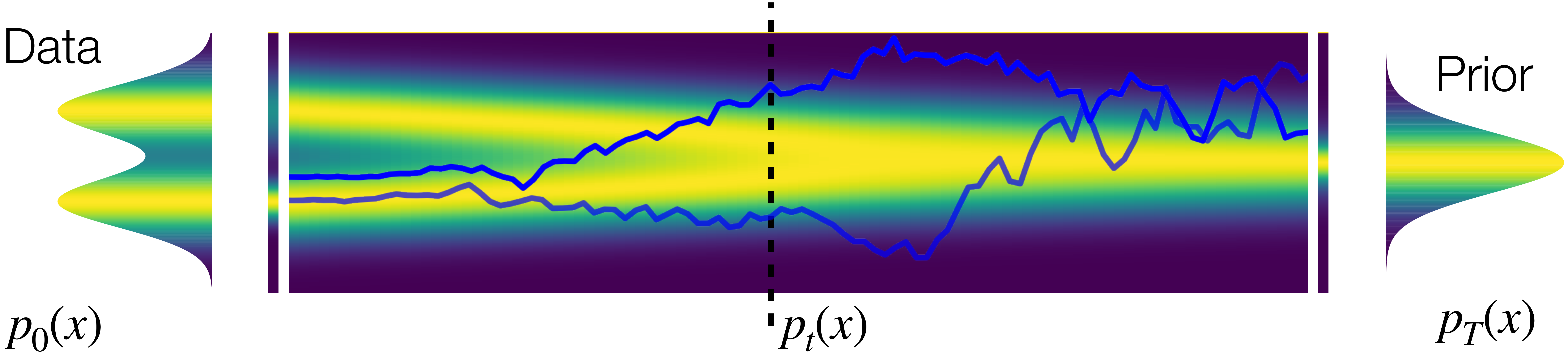


Stochastic processes

$$\{\mathbf{x}_t\}_{t\in[0,T]}$$

$$\downarrow$$

Perturbing data with stochastic processes



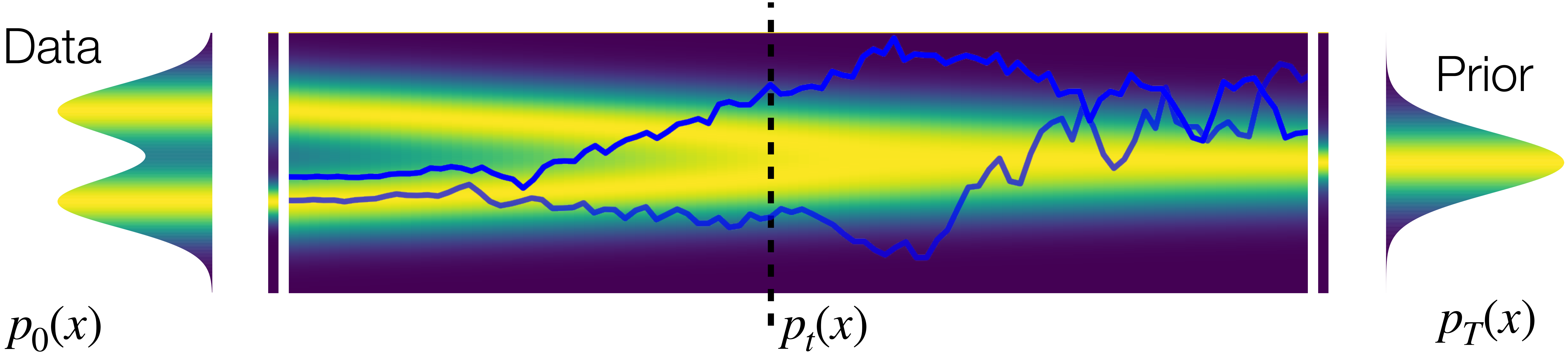
Stochastic processes

$$\{\mathbf{x}_t\}_{t\in[0,T]}$$

$$\downarrow$$

Probability densities

Perturbing data with stochastic processes



Stochastic processes

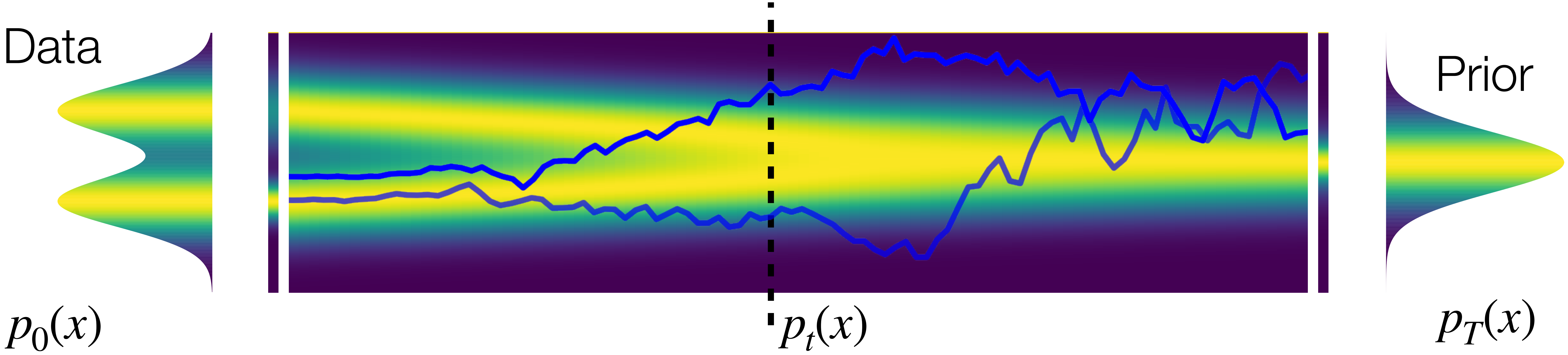
$$\{\mathbf{x}_t\}_{t\in[0,T]}$$

↓

Probability densities

$$\{p_t(\mathbf{x})\}_{t\in[0,T]}$$

Perturbing data with stochastic processes



Stochastic processes

$$\{\mathbf{x}_t\}_{t\in[0,T]}$$

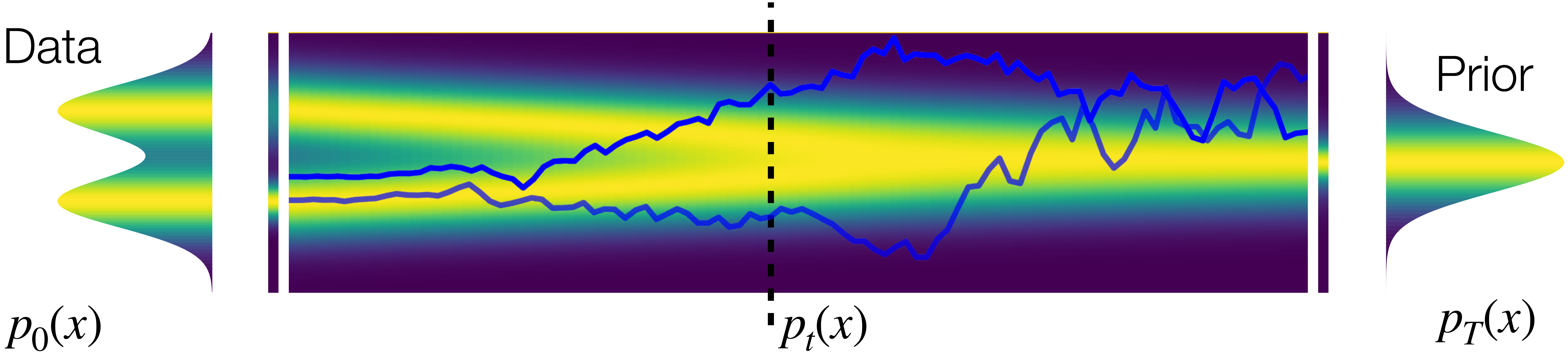
Probability densities

$$\{p_t(\mathbf{x})\}_{t\in[0,T]}$$

Stochastic differential equation (SDE)

$$d\mathbf{x}_t = \mu(\mathbf{x}_t, t)dt + \sigma(\mathbf{x}_t, t)dW_t$$

Perturbing data with stochastic processes



Stochastic processes

$$\{\mathbf{x}_t\}_{t\in[0,T]}$$

$$\Big\downarrow$$

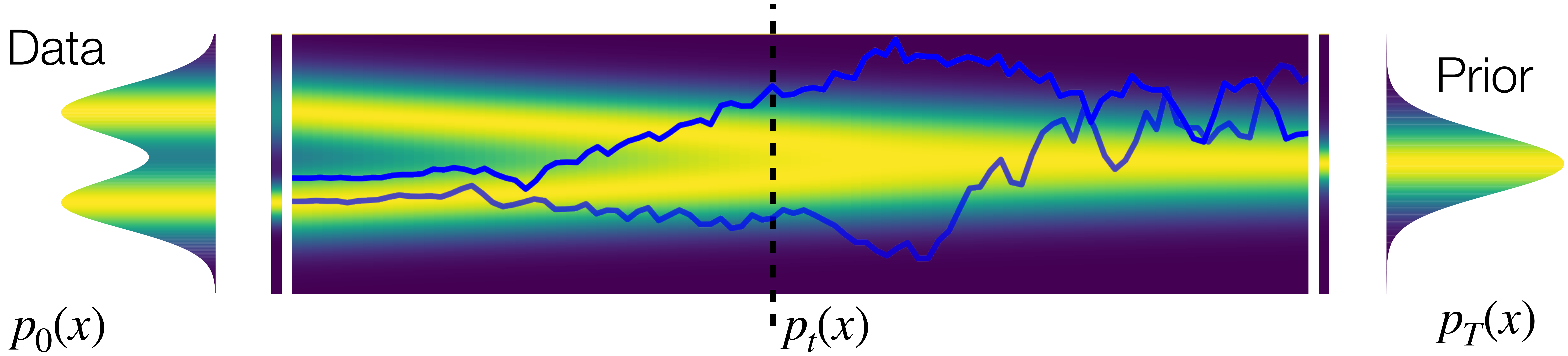
Probability densities

$$\{p_t(\mathbf{x})\}_{t\in[0,T]}$$

Stochastic differential equation (SDE)

$$d\mathbf{x}_t = \underbrace{\mu(\mathbf{x}_t, t)dt}_{\text{drift}} + \sigma(\mathbf{x}_t, t)dW_t$$

Perturbing data with stochastic processes



Stochastic processes

$$\{\mathbf{x}_t\}_{t\in[0,T]}$$

↓

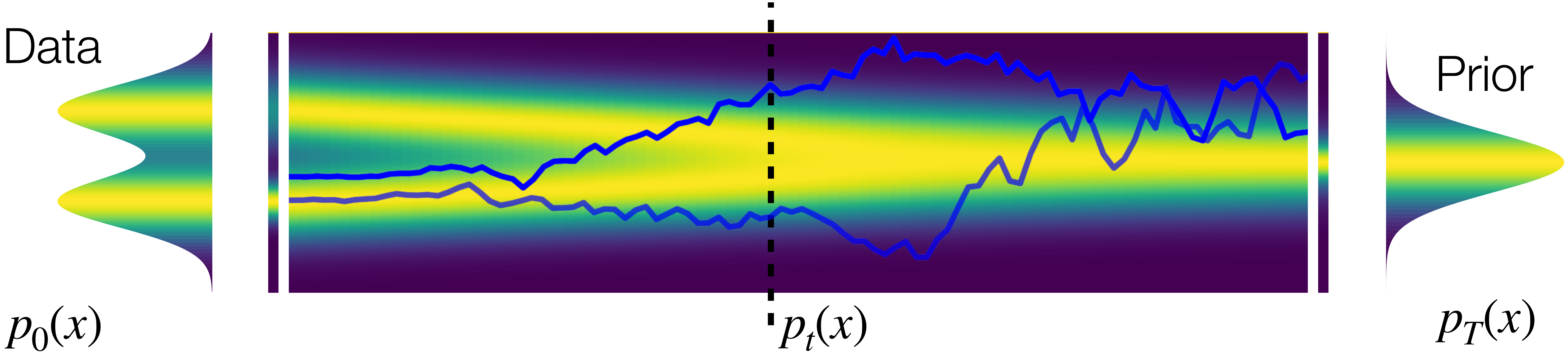
Probability densities

$$\{p_t(\mathbf{x})\}_{t\in[0,T]}$$

Stochastic differential equation (SDE)

$$d\mathbf{x}_t = \underbrace{\mu(\mathbf{x}_t, t)dt}_{\text{drift}} + \underbrace{\sigma(\mathbf{x}_t, t)dW_t}_{\text{randomness}}$$

Perturbing data with stochastic processes



Stochastic processes

$$\{\mathbf{x}_t\}_{t\in[0,T]}$$

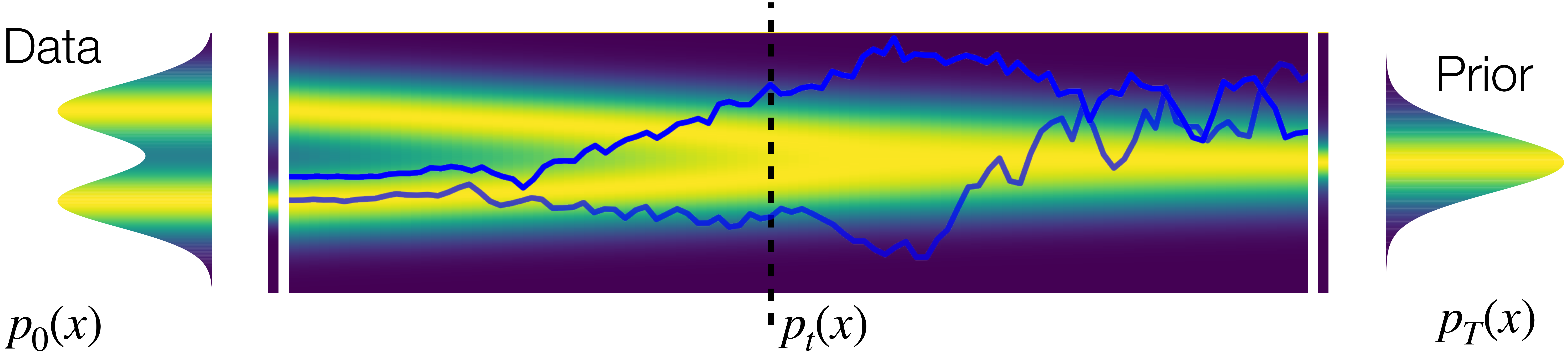
Probability densities

$$\{p_t(\mathbf{x})\}_{t\in[0,T]}$$

Stochastic differential equation (SDE)

$$d\mathbf{x}_t = \underbrace{\mu(\mathbf{x}_t, t)dt}_{\text{drift}} + \underbrace{\sigma(\mathbf{x}_t, t)dW_t}_{\text{randomness}}$$

Perturbing data with stochastic processes



Stochastic processes

$$\{\mathbf{x}_t\}_{t\in[0,T]}$$

←

Stochastic differential equation (SDE)

$$d\mathbf{x}_t = \underbrace{\mu(\mathbf{x}_t, t)dt}_{\text{drift}} + \underbrace{\sigma(\mathbf{x}_t, t)dW_t}_{\text{randomness}}$$

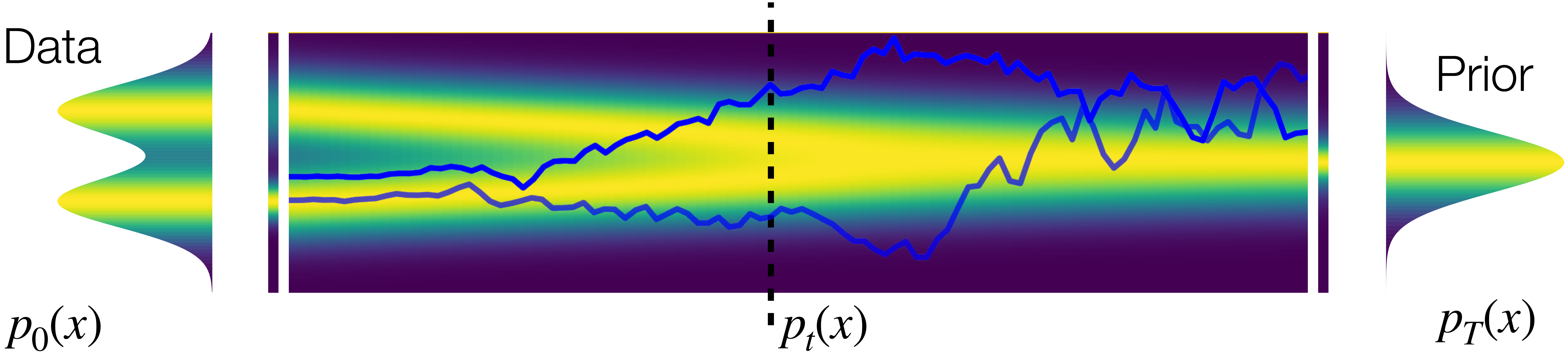
↓

Probability densities

$$\{p_t(\mathbf{x})\}_{t\in[0,T]}$$

Toy SDE: $d\mathbf{x}_t = \sigma(\mathbf{x}_t, t)dW_t$

Perturbing data with stochastic processes



Stochastic processes

$$\{\mathbf{x}_t\}_{t\in[0,T]}$$

Stochastic differential equation (SDE)

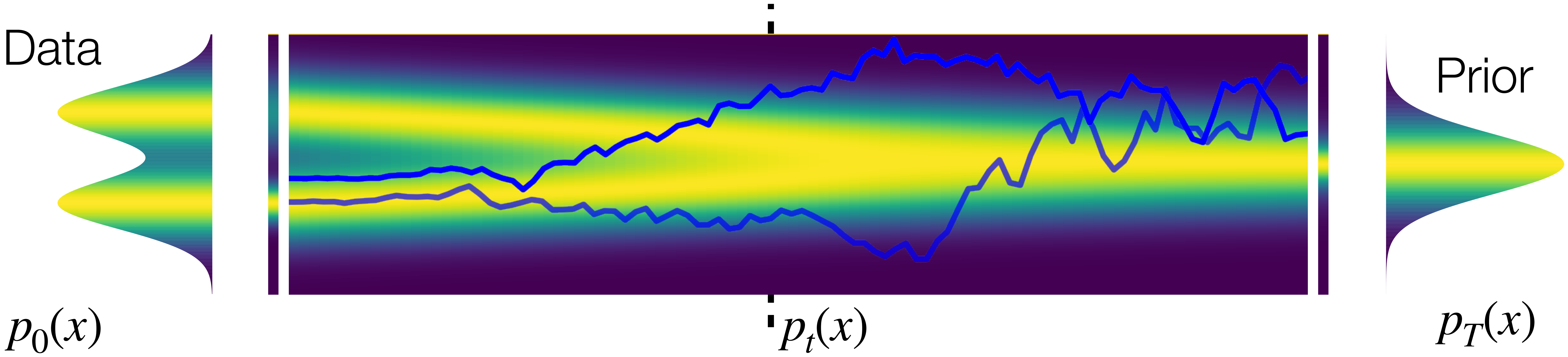
$$d\mathbf{x}_t = \underbrace{\mu(\mathbf{x}_t, t)dt}_{\text{drift}} + \underbrace{\sigma(\mathbf{x}_t, t)dW_t}_{\text{randomness}}$$

Probability densities

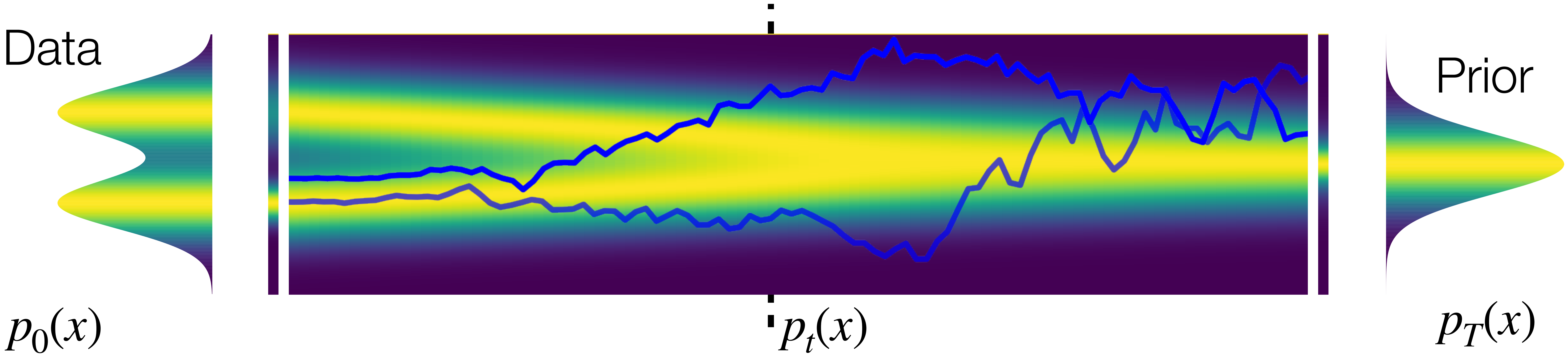
$$\{p_t(\mathbf{x})\}_{t\in[0,T]}$$

Toy SDE: $d\mathbf{x}_t = \sigma(\mathbf{x}_t, t)dW_t$

Generation via reverse stochastic process

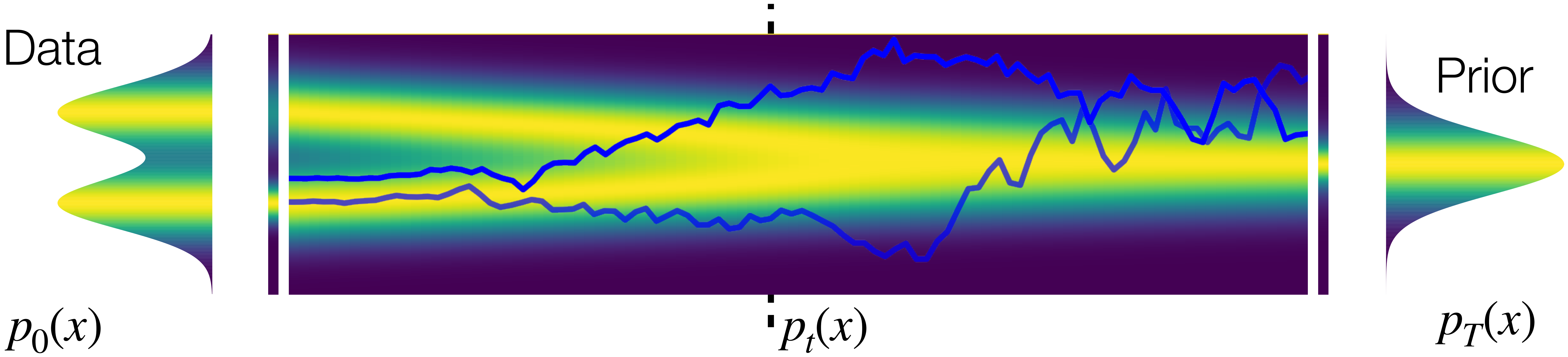


Generation via reverse stochastic process



Forward SDE: $(t : 0 \rightarrow T)$

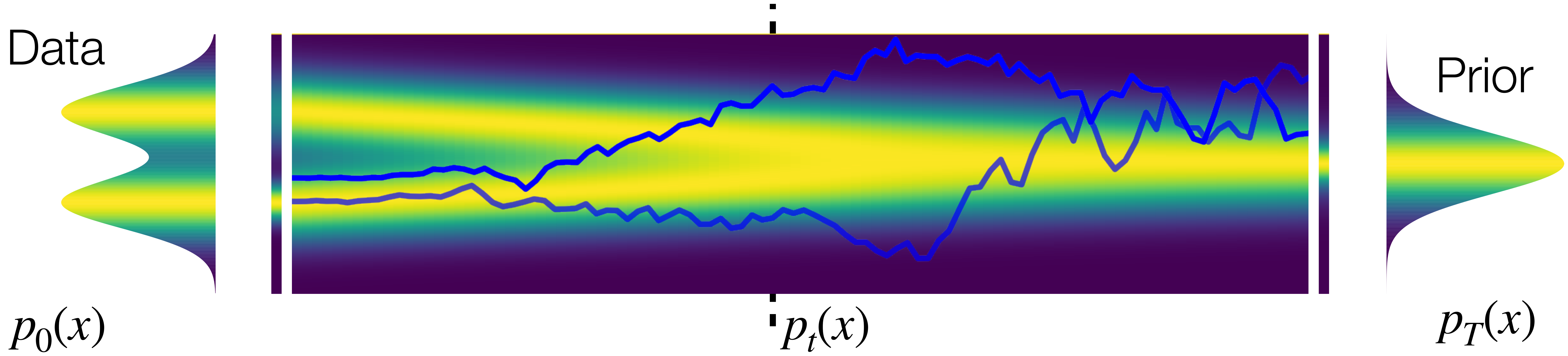
Generation via reverse stochastic process



Forward SDE: $(t : 0 \rightarrow T)$

$$d\mathbf{x}_t = \sigma(\mathbf{x}_t, t)dW_t$$

Generation via reverse stochastic process

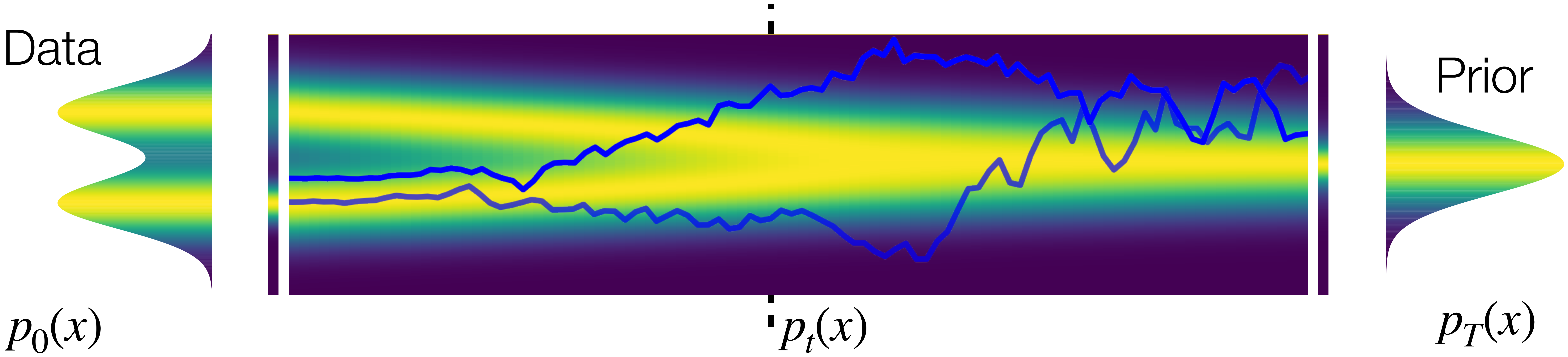


Forward SDE: $(t : 0 \rightarrow T)$

$$d\mathbf{x}_t = \sigma(\mathbf{x}_t, t)dW_t$$

Reverse SDE: $(t : T \rightarrow 0)$

Generation via reverse stochastic process



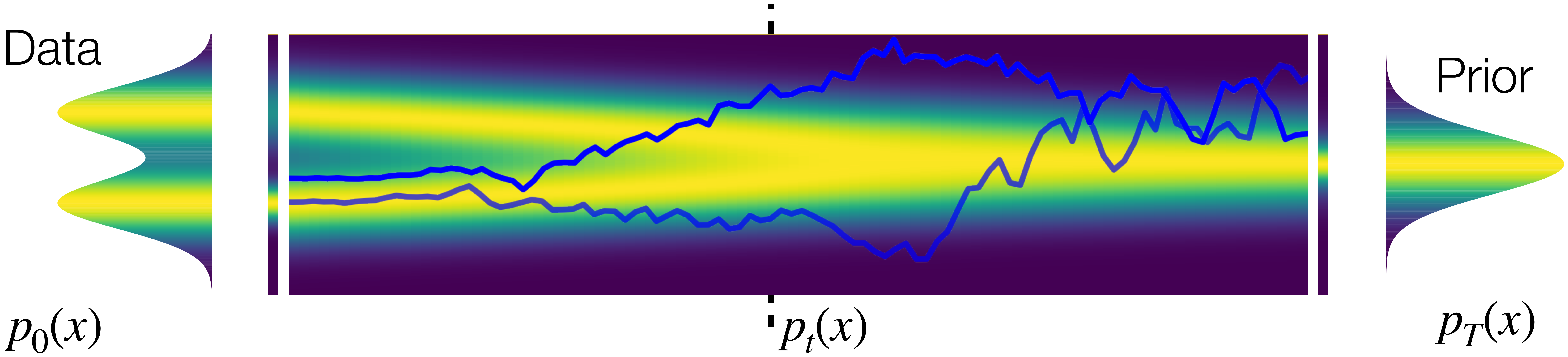
Forward SDE: $(t : 0 \rightarrow T)$

$$d\mathbf{x}_t = \sigma(\mathbf{x}_t, t) dW_t$$

Reverse SDE: $(t : T \rightarrow 0)$

$$d\mathbf{x}_t = -\sigma(t)^2 \nabla_x \log p_t(\mathbf{x}_t) dt + \sigma(t) dW_t$$

Generation via reverse stochastic process



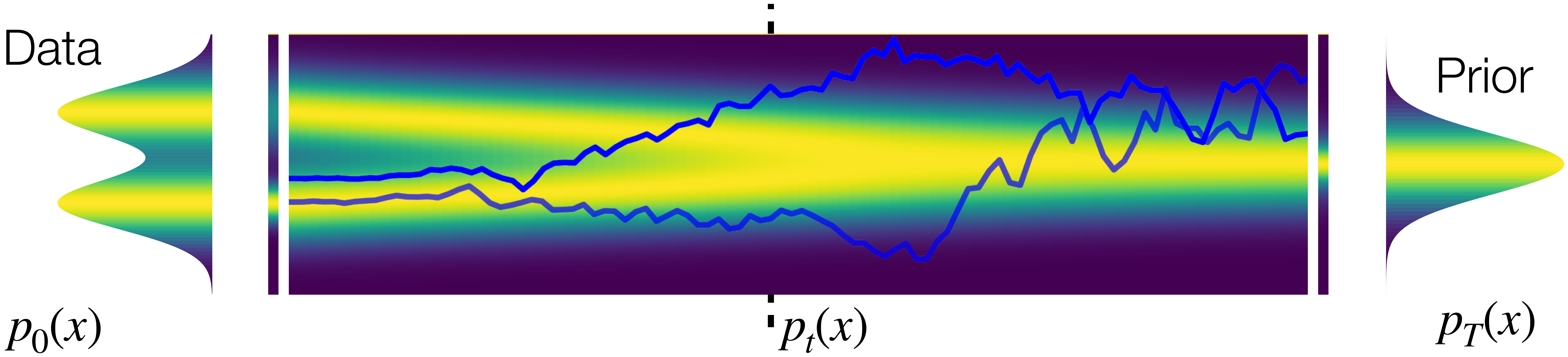
Forward SDE: $(t : 0 \rightarrow T)$

$$d\mathbf{x}_t = \sigma(\mathbf{x}_t, t)dW_t$$

Reverse SDE: $(t : T \rightarrow 0)$

$$d\mathbf{x}_t = -\sigma(t)^2 \underbrace{\nabla_x \log p_t(\mathbf{x}_t)}^{\text{score function}} dt + \sigma(t)dW_t$$

Generation via reverse stochastic process



Forward SDE: $(t: 0 \rightarrow T)$

$$d\mathbf{x}_t = \sigma(\mathbf{x}_t, t)dW_t$$

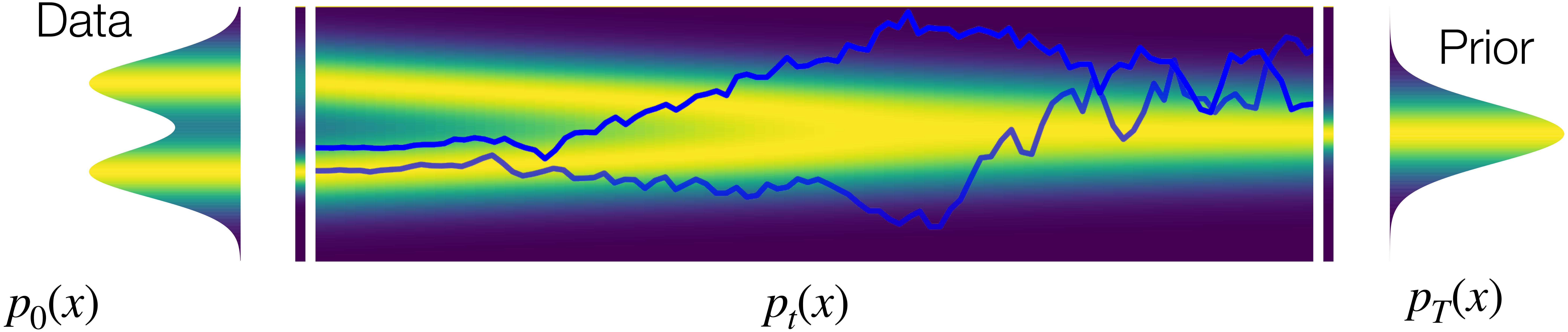
Infinitesimal noise in the reverse time direction

Reverse SDE: $(t: T \rightarrow 0)$

score function

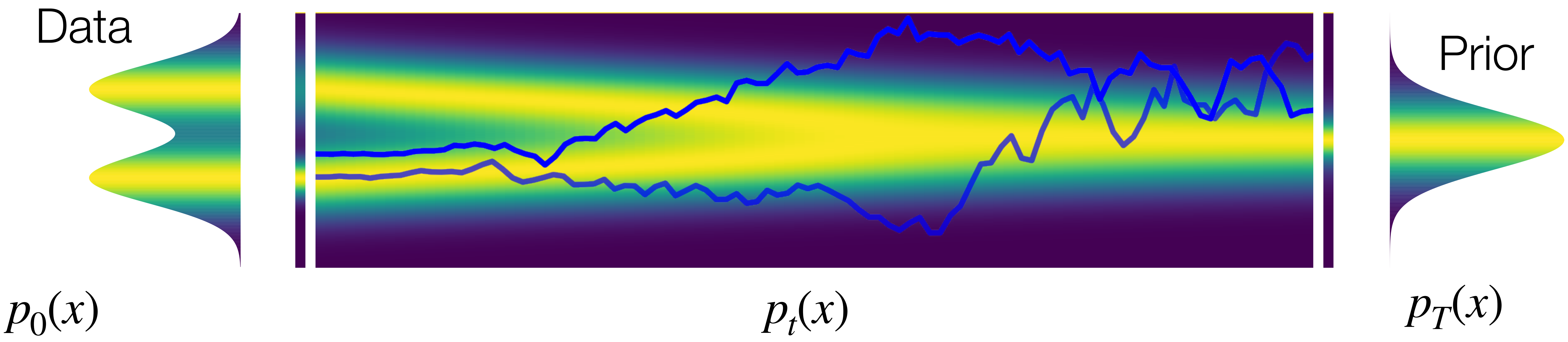
$$d\mathbf{x}_t = -\sigma(t)^2 \underline{\nabla_x \log p_t(\mathbf{x}_t)} dt + \underline{\sigma(t) dW_t}$$

Predictor-Corrector Sampling methods



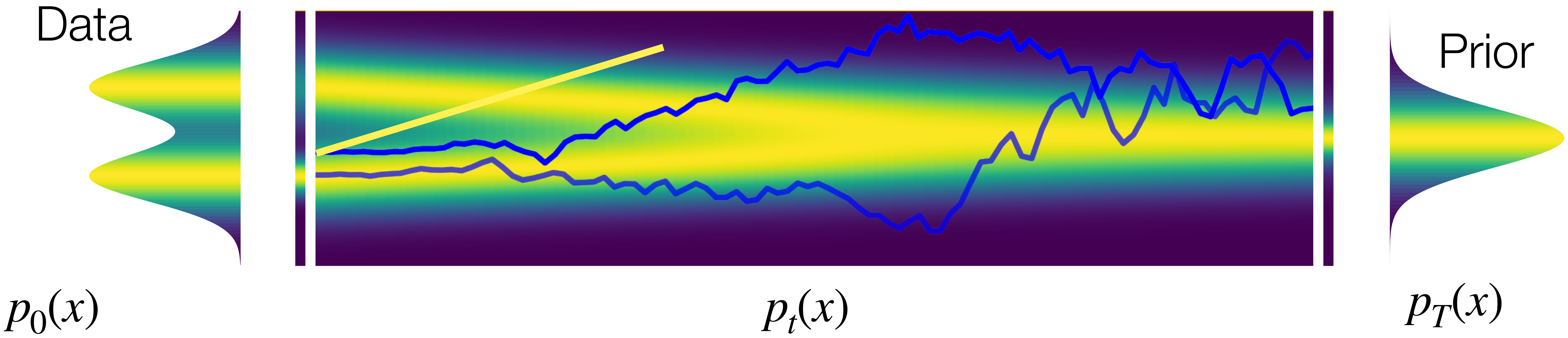
Predictor-Corrector Sampling methods

Predictor: Numerical SDE solver (as shown in the previous slide)



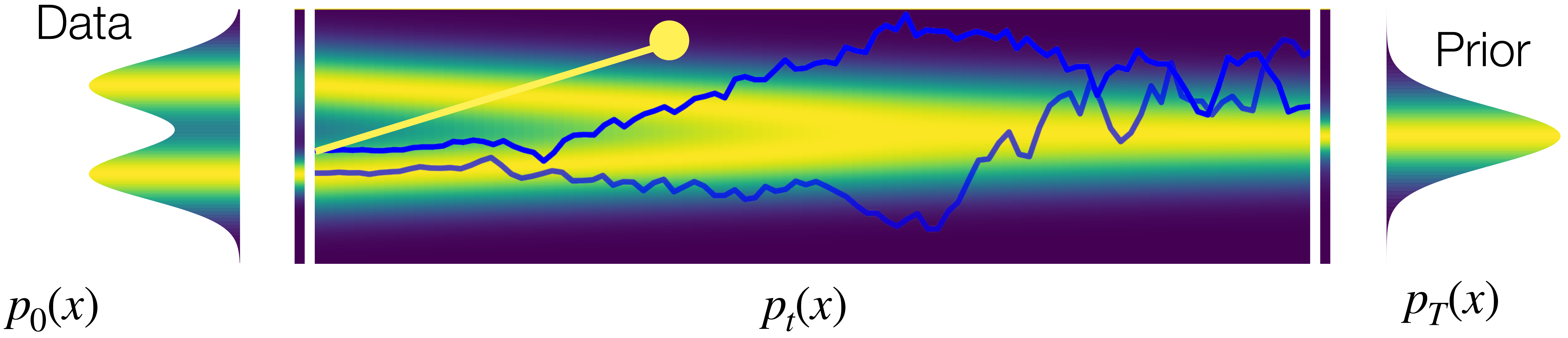
Predictor-Corrector Sampling methods

Predictor: Numerical SDE solver (as shown in the previous slide)



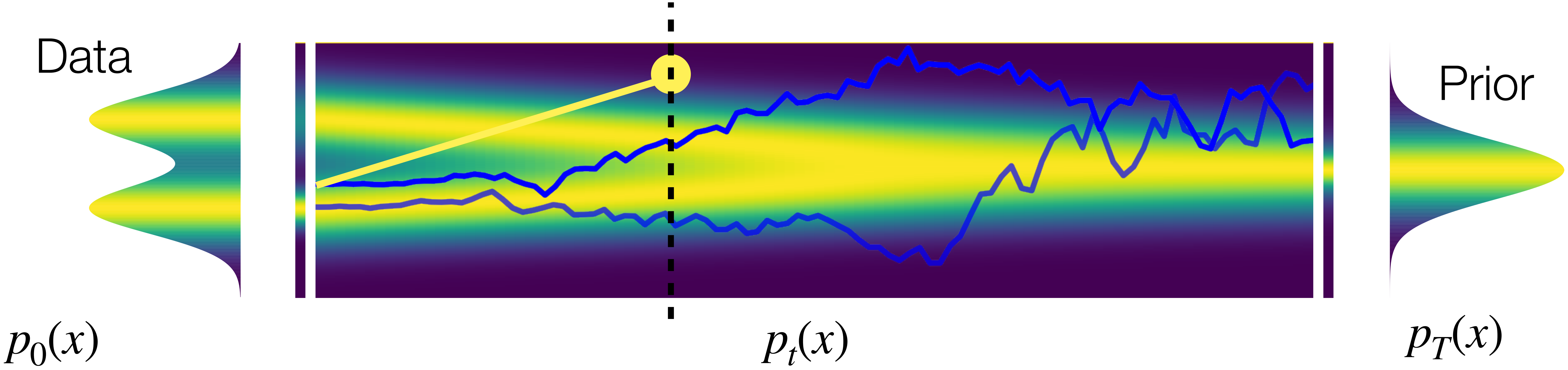
Predictor-Corrector Sampling methods

Predictor: Numerical SDE solver (as shown in the previous slide)



Predictor-Corrector Sampling methods

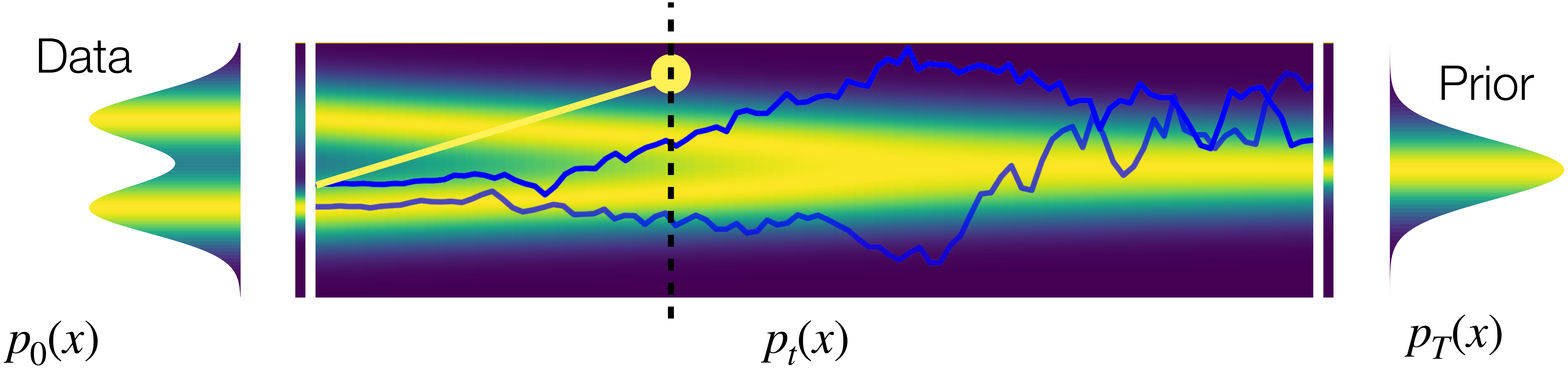
Predictor: Numerical SDE solver (as shown in the previous slide)



Predictor-Corrector Sampling methods

Predictor: Numerical SDE solver (as shown in the previous slide)

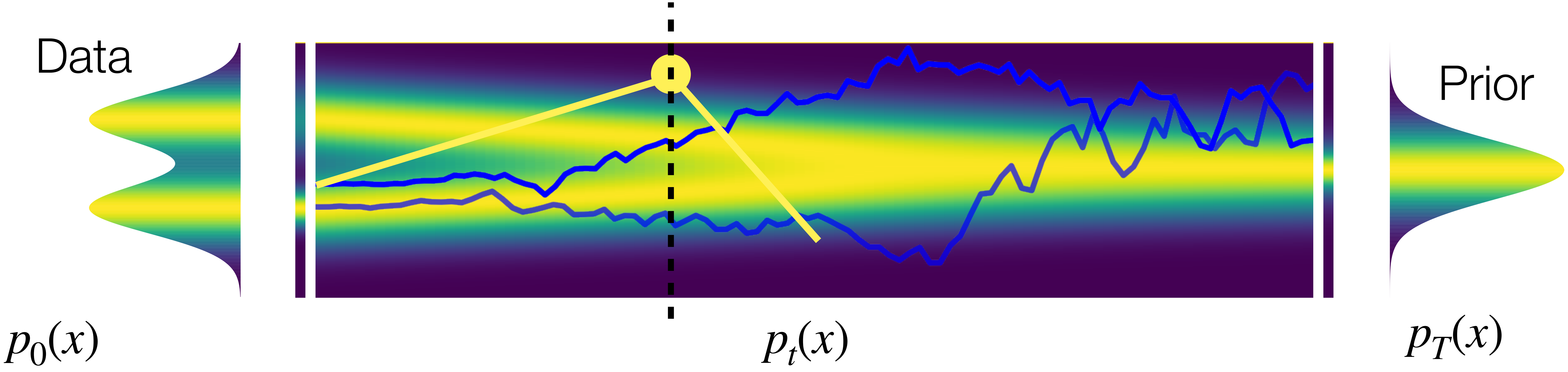
Corrector: Score-based MCMC (as discussed in score matching)



Predictor-Corrector Sampling methods

Predictor: Numerical SDE solver (as shown in the previous slide)

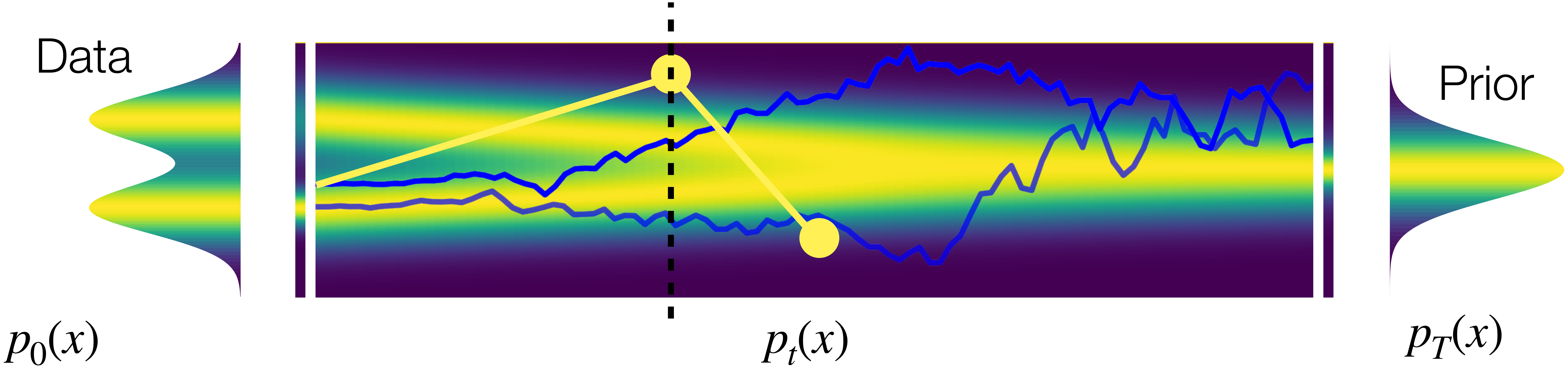
Corrector: Score-based MCMC (as discussed in score matching)



Predictor-Corrector Sampling methods

Predictor: Numerical SDE solver (as shown in the previous slide)

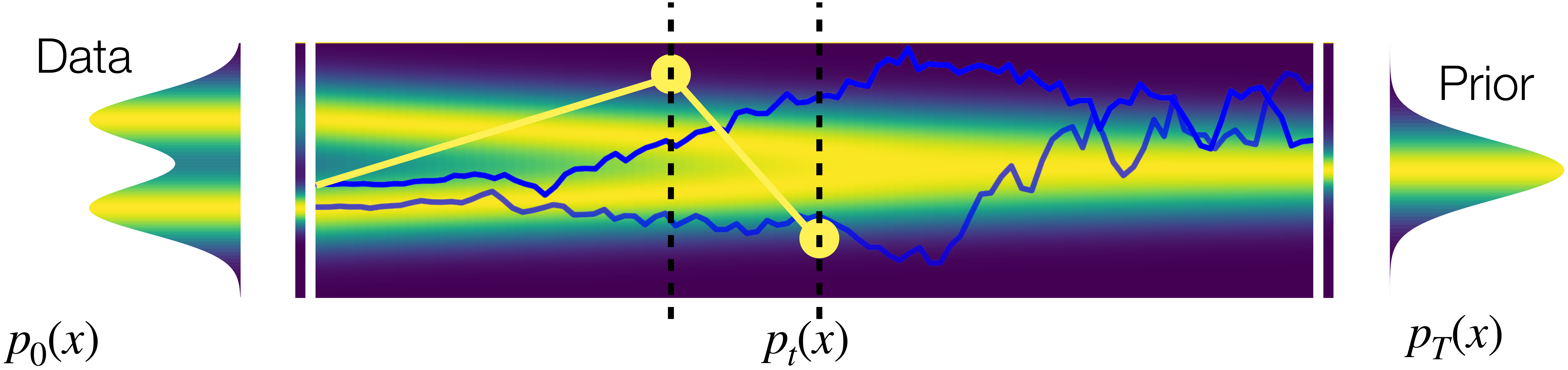
Corrector: Score-based MCMC (as discussed in score matching)



Predictor-Corrector Sampling methods

Predictor: Numerical SDE solver (as shown in the previous slide)

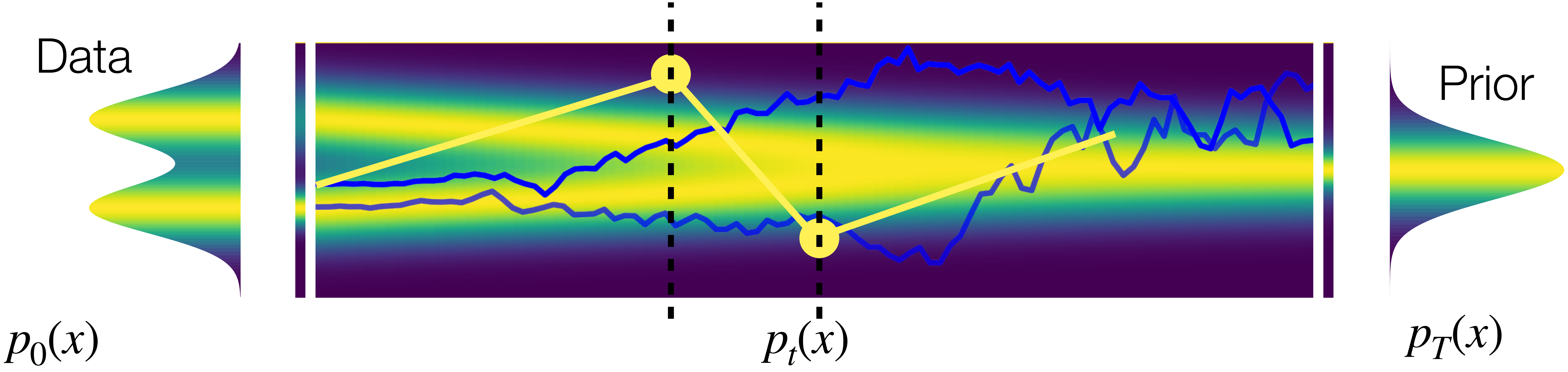
Corrector: Score-based MCMC (as discussed in score matching)



Predictor-Corrector Sampling methods

Predictor: Numerical SDE solver (as shown in the previous slide)

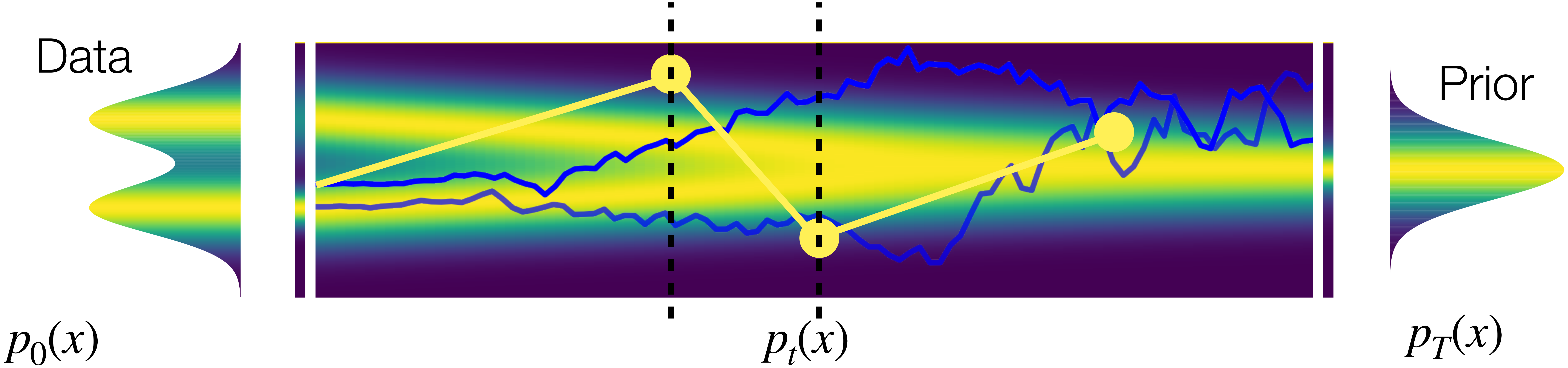
Corrector: Score-based MCMC (as discussed in score matching)



Predictor-Corrector Sampling methods

Predictor: Numerical SDE solver (as shown in the previous slide)

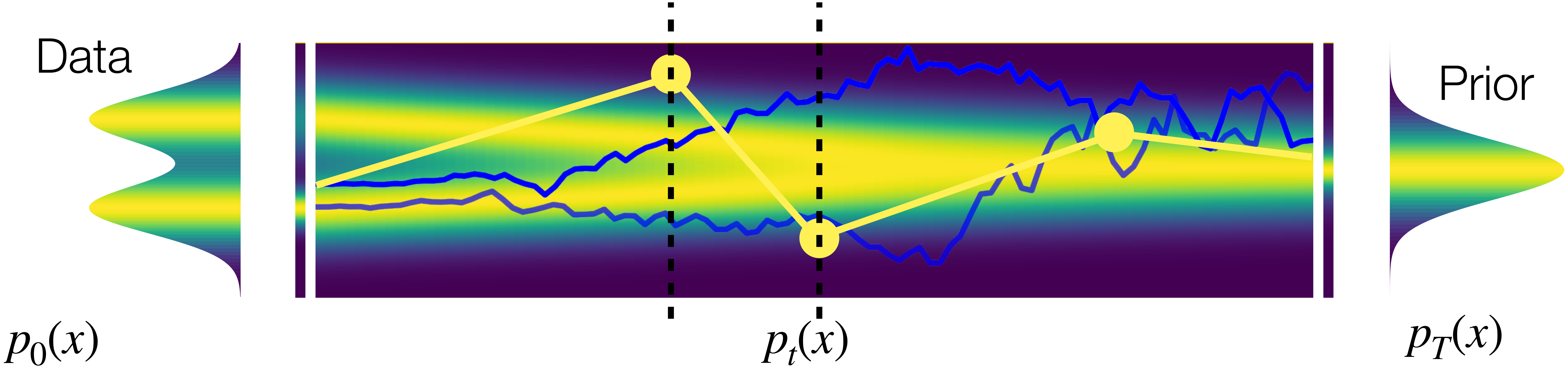
Corrector: Score-based MCMC (as discussed in score matching)



Predictor-Corrector Sampling methods

Predictor: Numerical SDE solver (as shown in the previous slide)

Corrector: Score-based MCMC (as discussed in score matching)



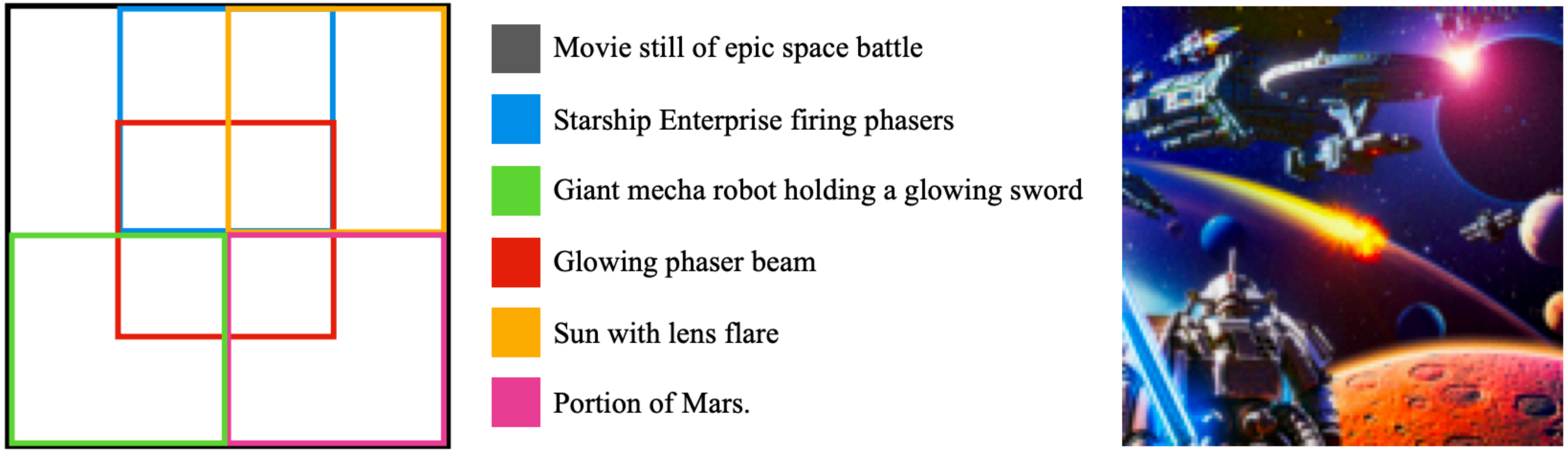

⬛ Movie still of epic space battle
🟦 Starship Enterprise firing phasers
🟩 Giant mecha robot holding a glowing sword
🟥 Glowing phaser beam
🟧 Sun with lens flare
🟪 Portion of Mars.

Figure 8: **Composition enables controllable image tapestries.**

Reduce, Reuse, Recycle: Compositional Generation with Energy-Based Diffusion Models and MCMC (Du et al. 2024)

Generative models

Langevin dynamics

Stochastic differential equations (SDEs)

Generative models

Langevin dynamics

Stochastic differential equations (SDEs)

Physically based rendering

Metropolis Hastings

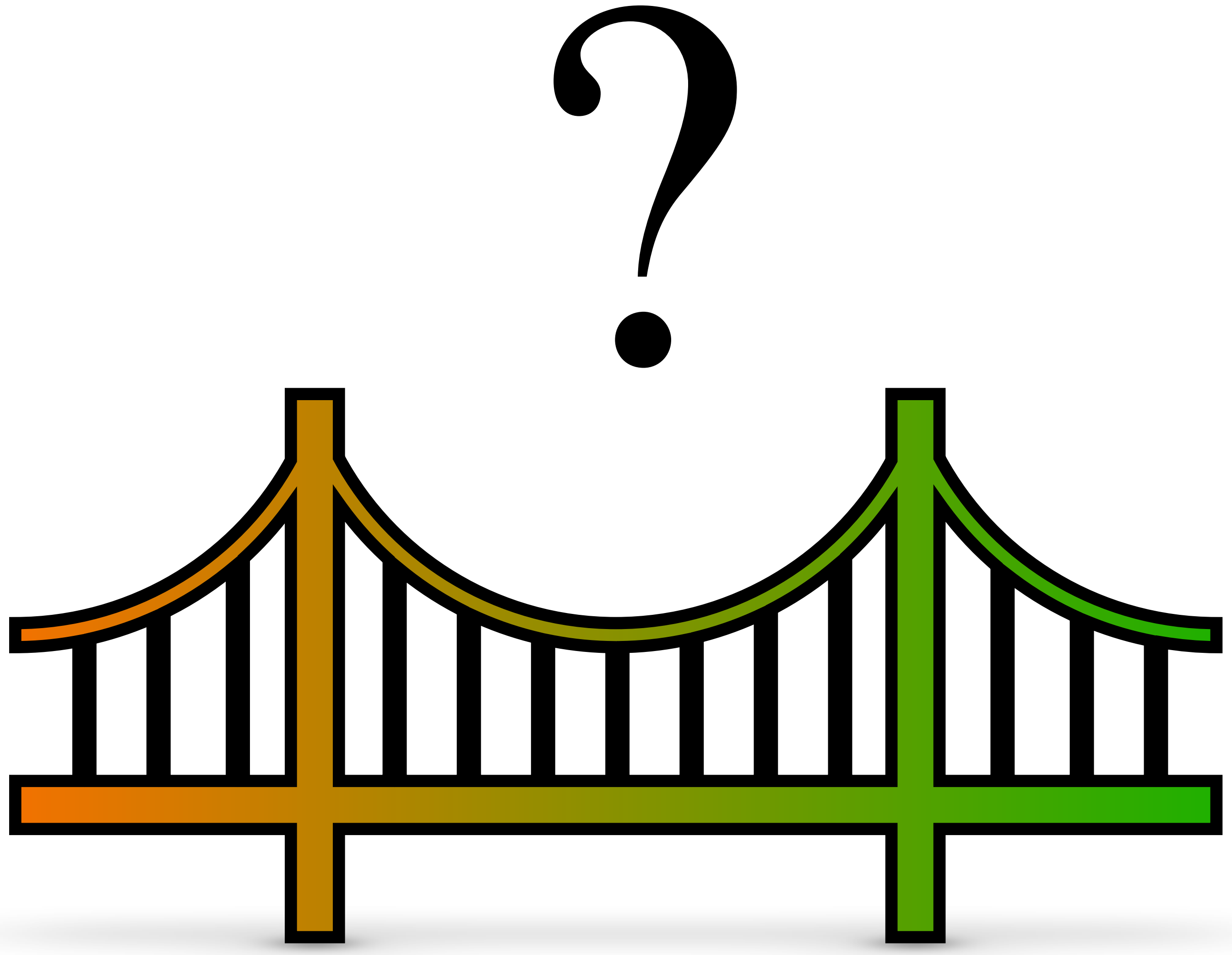
Langevin Monte Carlo

Hamiltonian Monte Carlo

Generative models

Langevin dynamics

Stochastic differential equations (SDEs)



Physically based rendering

Metropolis Hastings

Langevin Monte Carlo

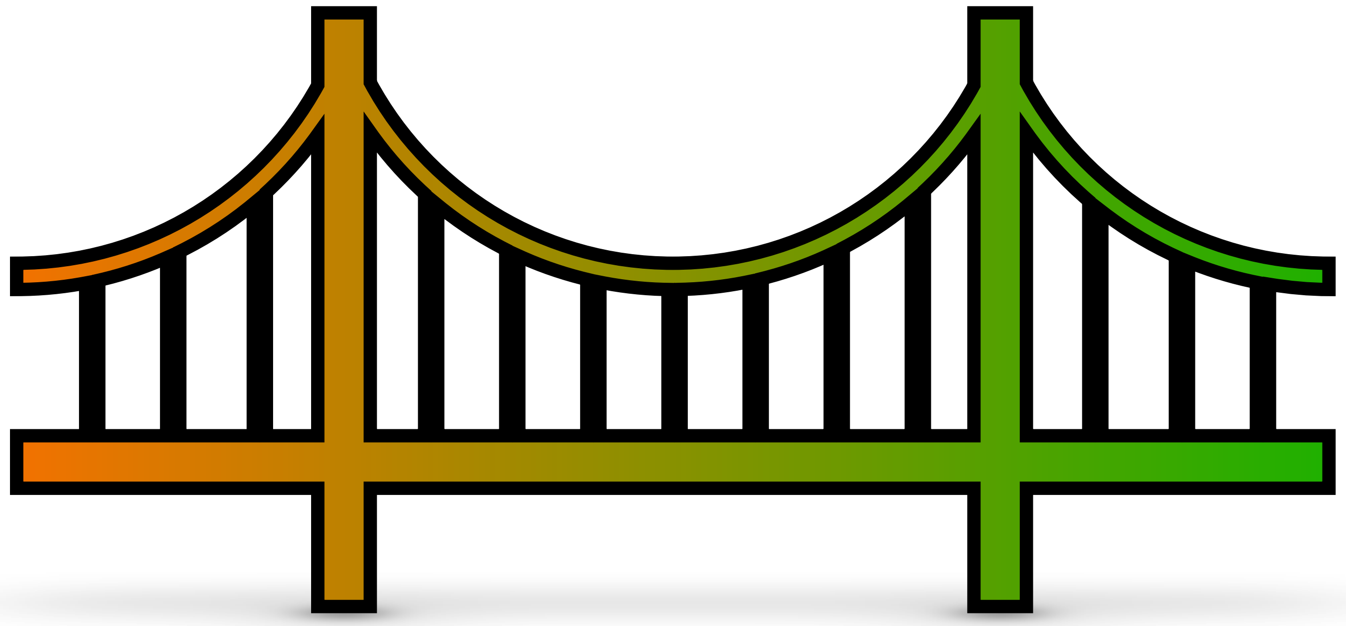
Hamiltonian Monte Carlo

Markov chain Monte Carlo

Generative models

Langevin dynamics

Stochastic differential equations (SDEs)



Physically based rendering

Metropolis Hastings

Langevin Monte Carlo

Hamiltonian Monte Carlo

Future directions

- Improvements in MCMC methods can bring benefits to both the communities

Future directions

- Improvements in MCMC methods can bring benefits to both the communities
    - We are working on this…

Future directions

- Improvements in MCMC methods can bring benefits to both the communities
  - We are working on this…
  - Can we bring these improvements to generative AI?

Future directions

- Improvements in MCMC methods can bring benefits to both the communities
    - We are working on this…
    - Can we bring these improvements to generative AI?

- Can MCMC serves as a link to bring physical accuracy within generative models?

Future directions

- Improvements in MCMC methods can bring benefits to both the communities
  - We are working on this…
  - Can we bring these improvements to generative AI?

- Can MCMC serves as a link to bring physical accuracy within generative models?
  - Many applications in architecture, aircraft design needs physical accuracy before realization in practice

Acknowledgements

Thank you!

&

Enjoy the rest of the SIGGRAPH Asia!