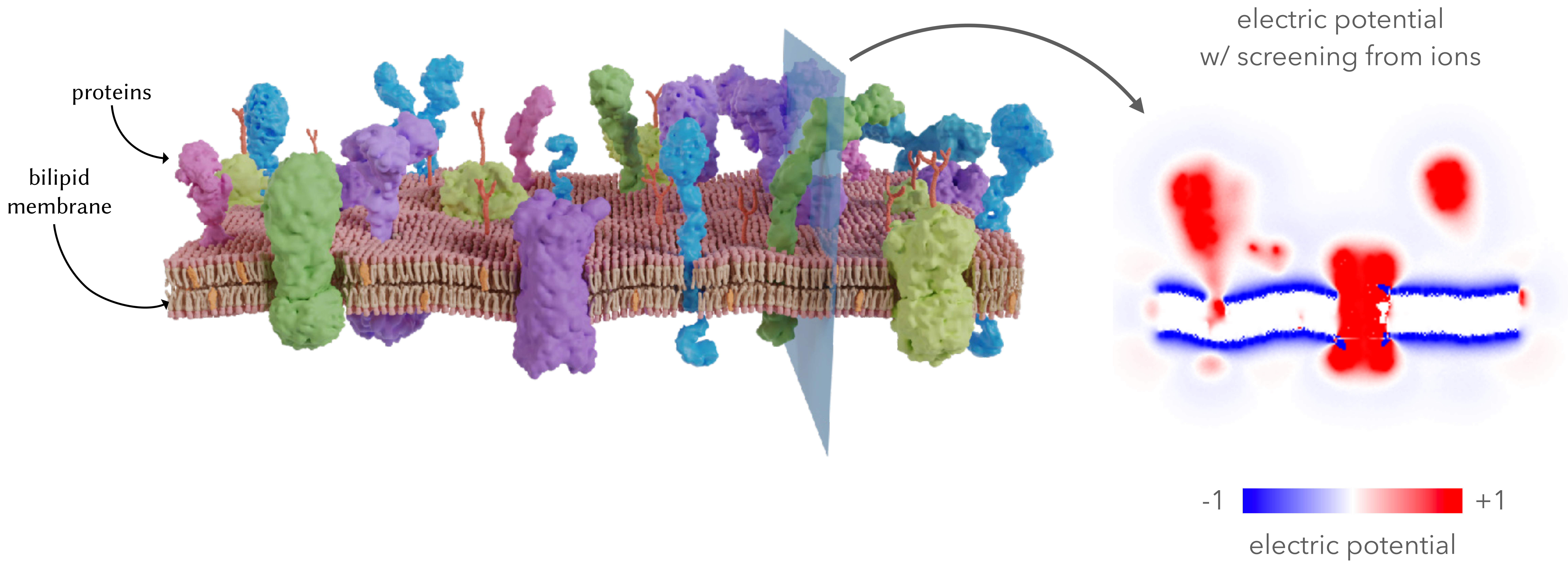
solving partial differential equations in participating media

Bailey Miller
CMU

Rohan Sawhney
NVIDIA

Keenan Crane
CMU

Ioannis Gkioulekas
CMU



Team

Bailey Miller
CMU

Rohan Sawhney
NVIDIA

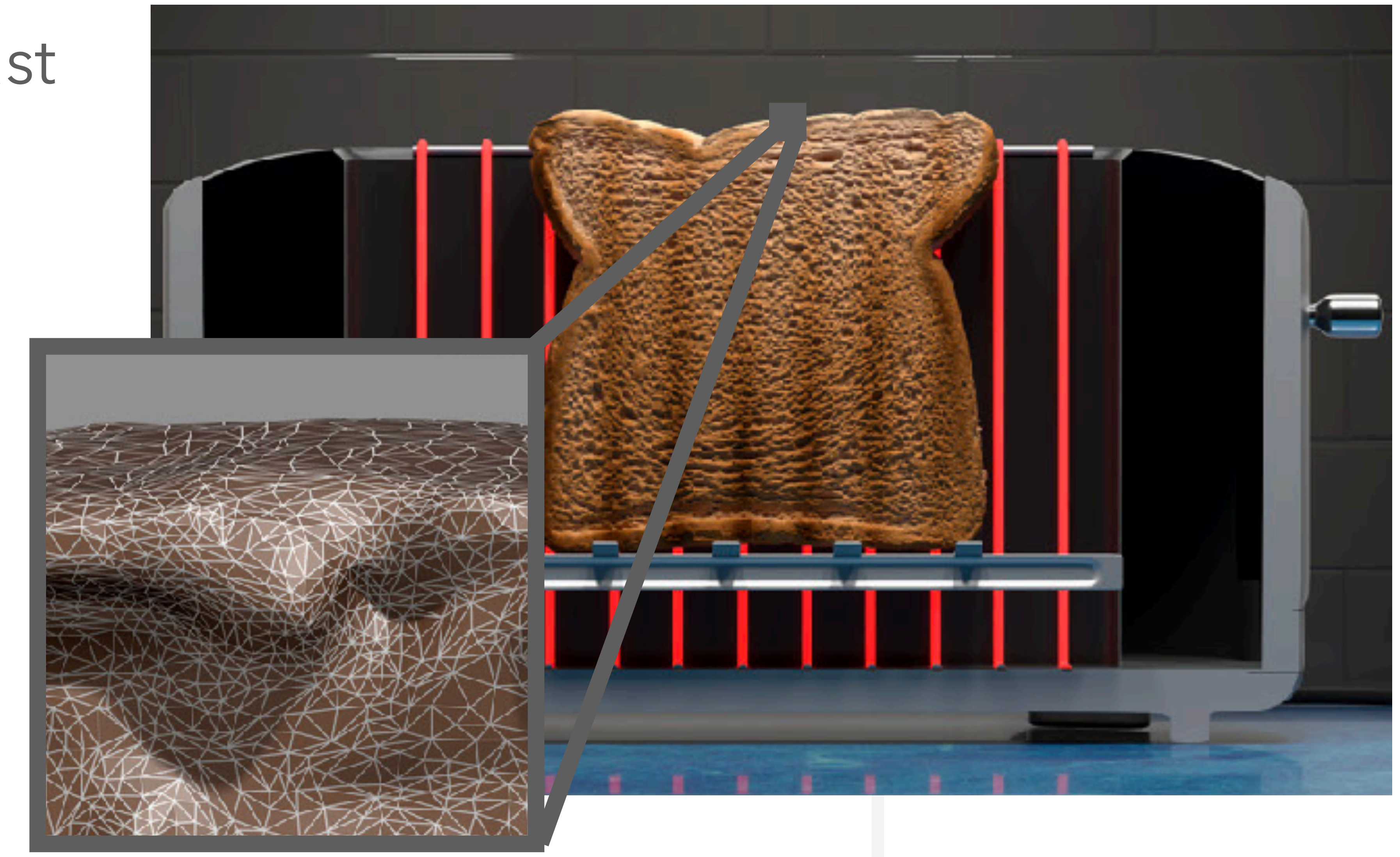
Keenan Crane
CMU

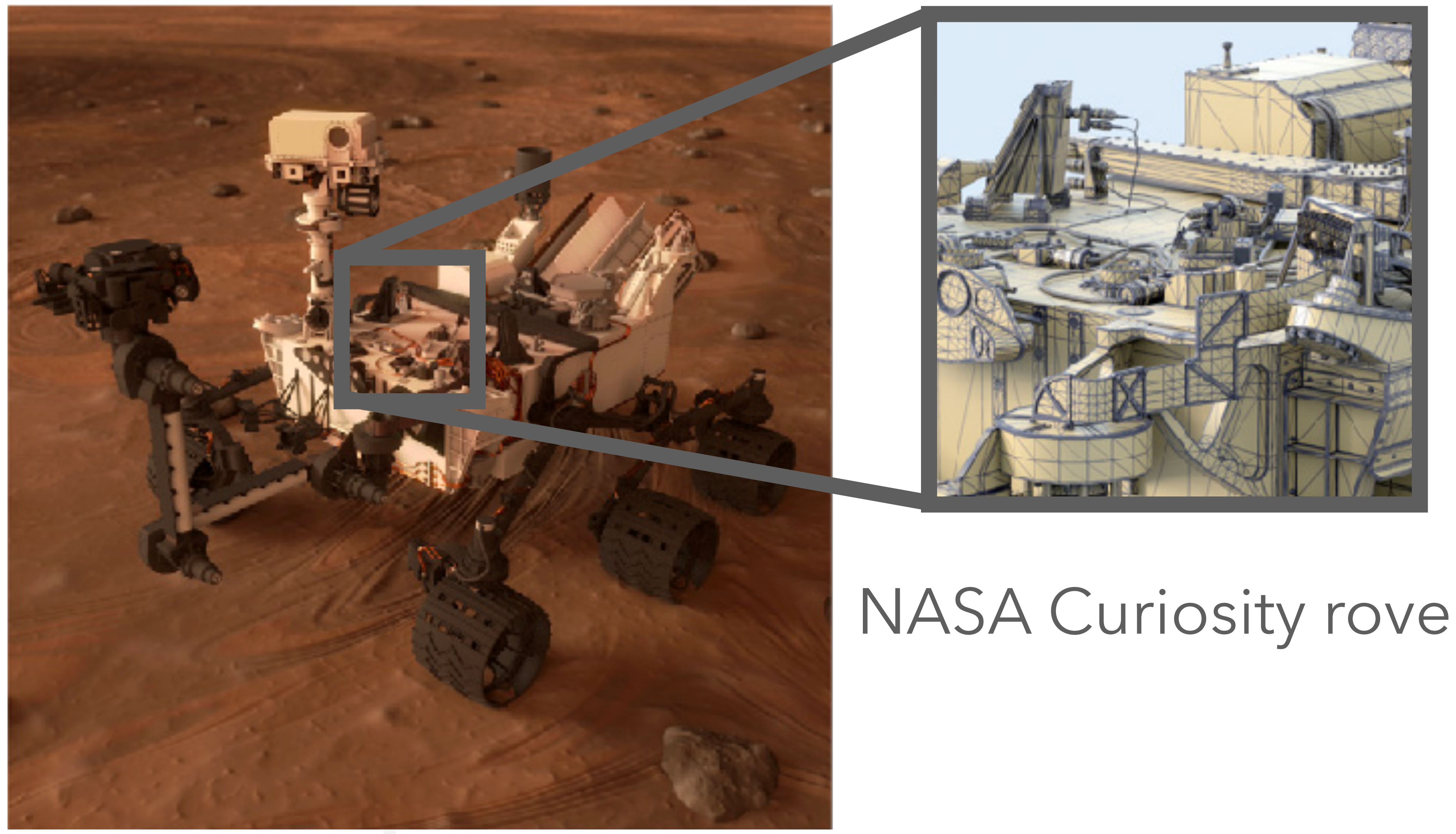
Ioannis Gkioulekas
CMU

Complex geometry is ubiquitous

Complex geometry is ubiquitous

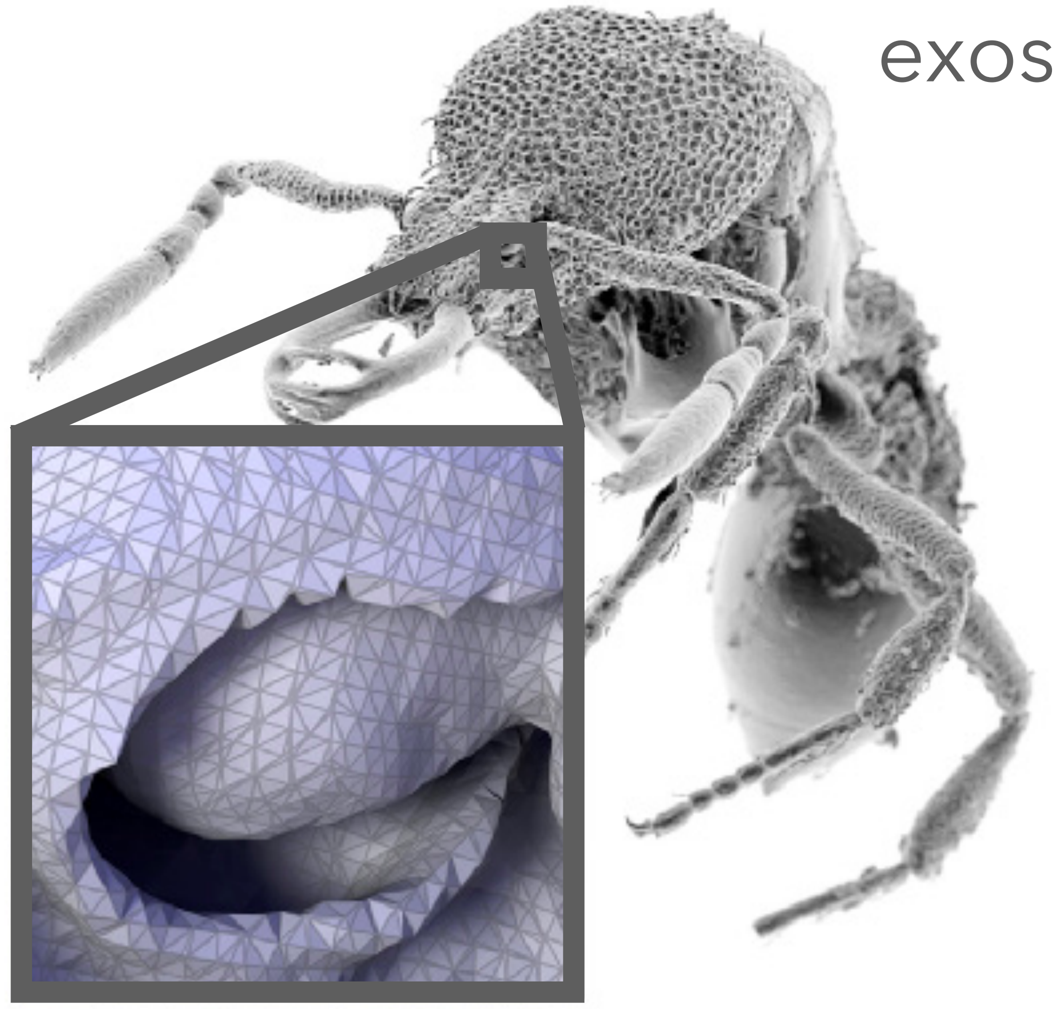
toast

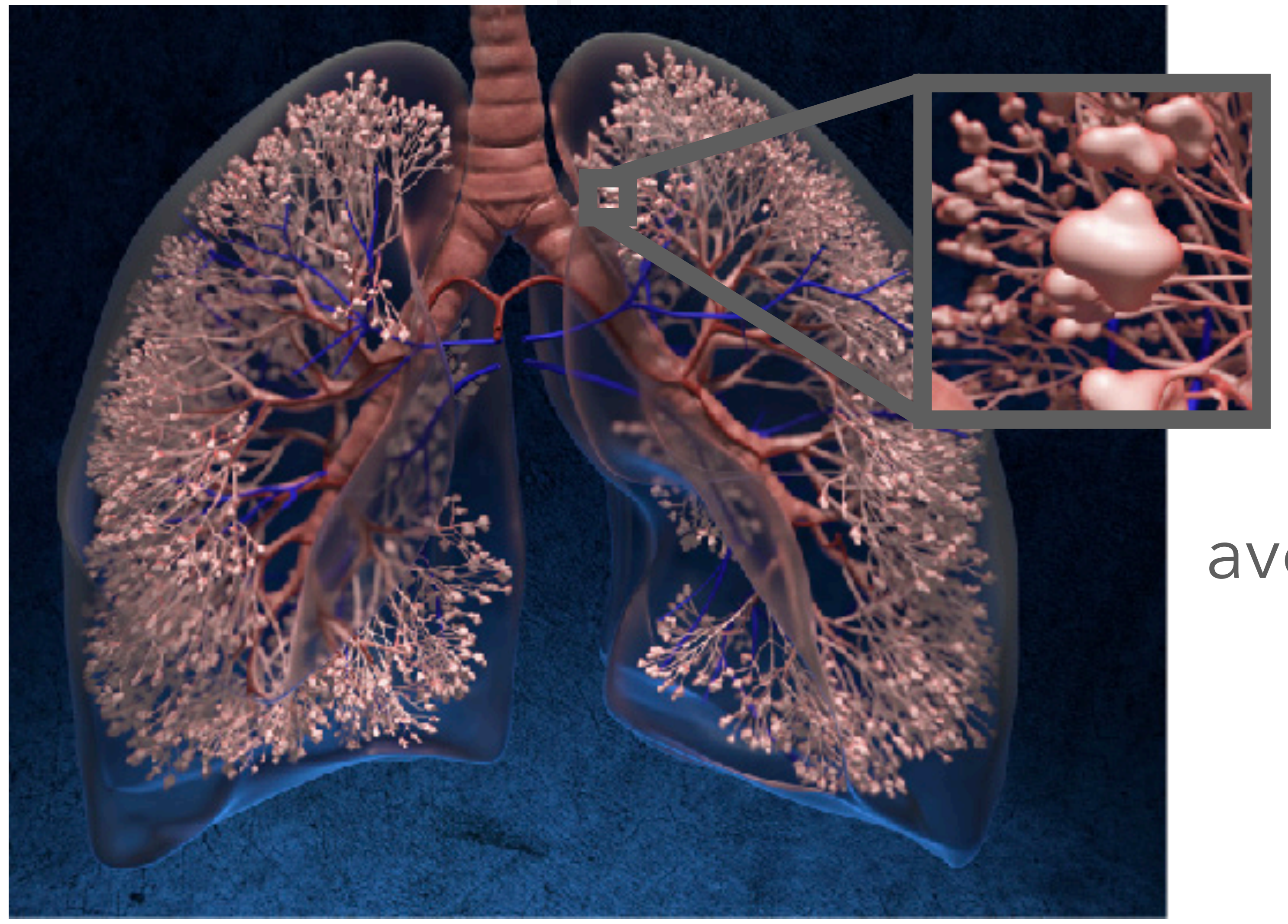




NASA Curiosity rover

exoskeleton

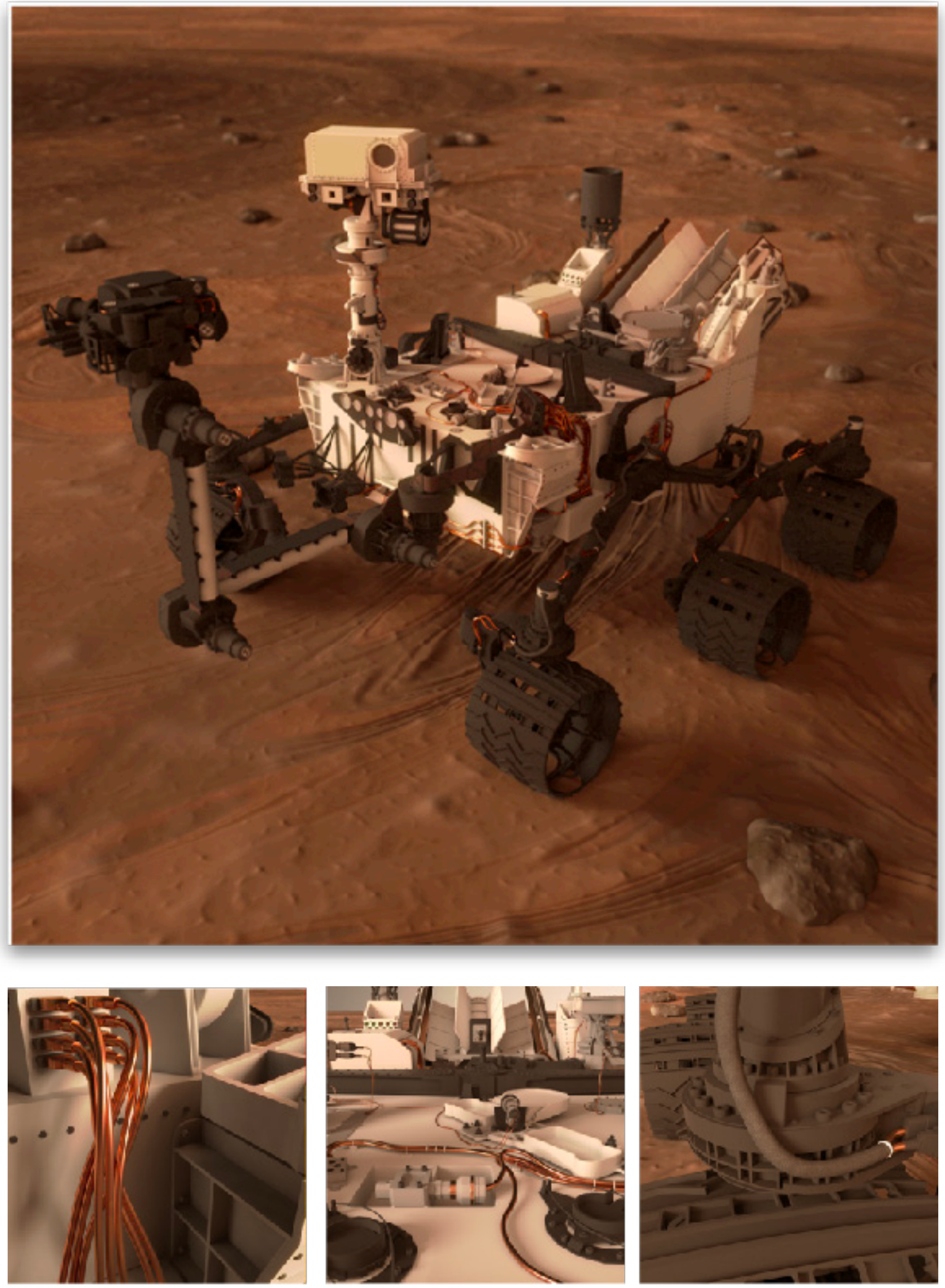




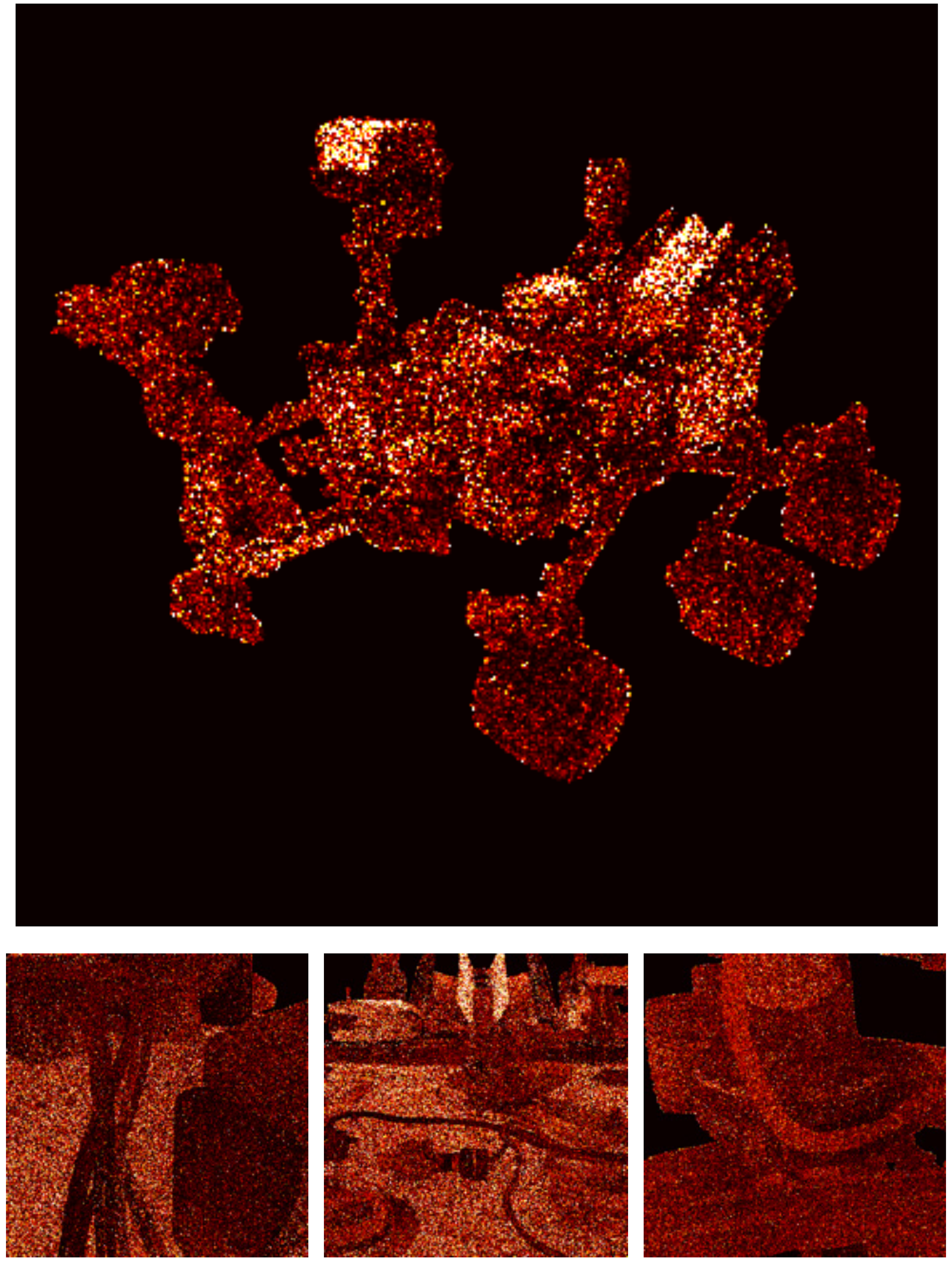
aveoli structure

Monte Carlo makes complex geometry tractable

visualization

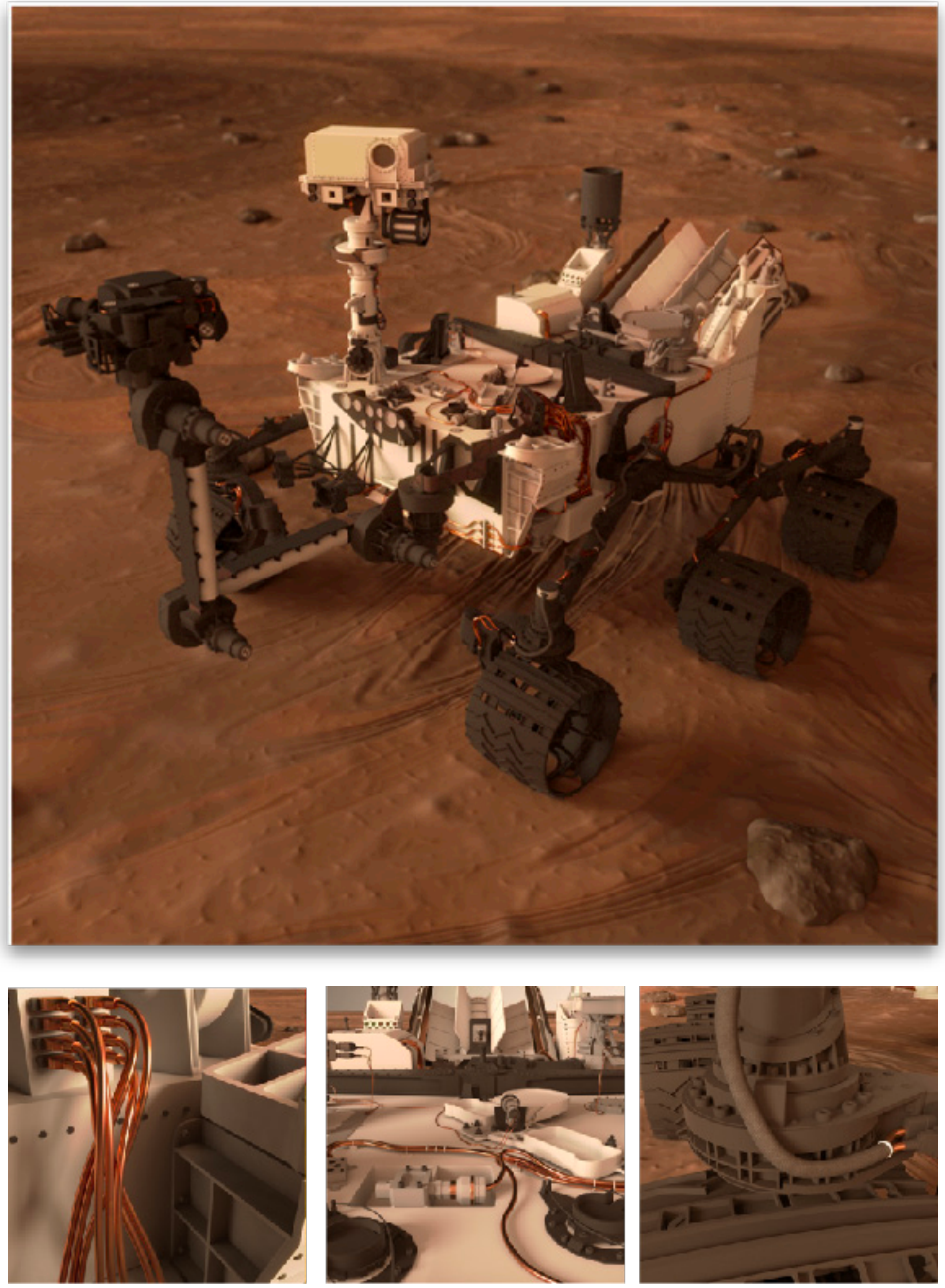


simulation

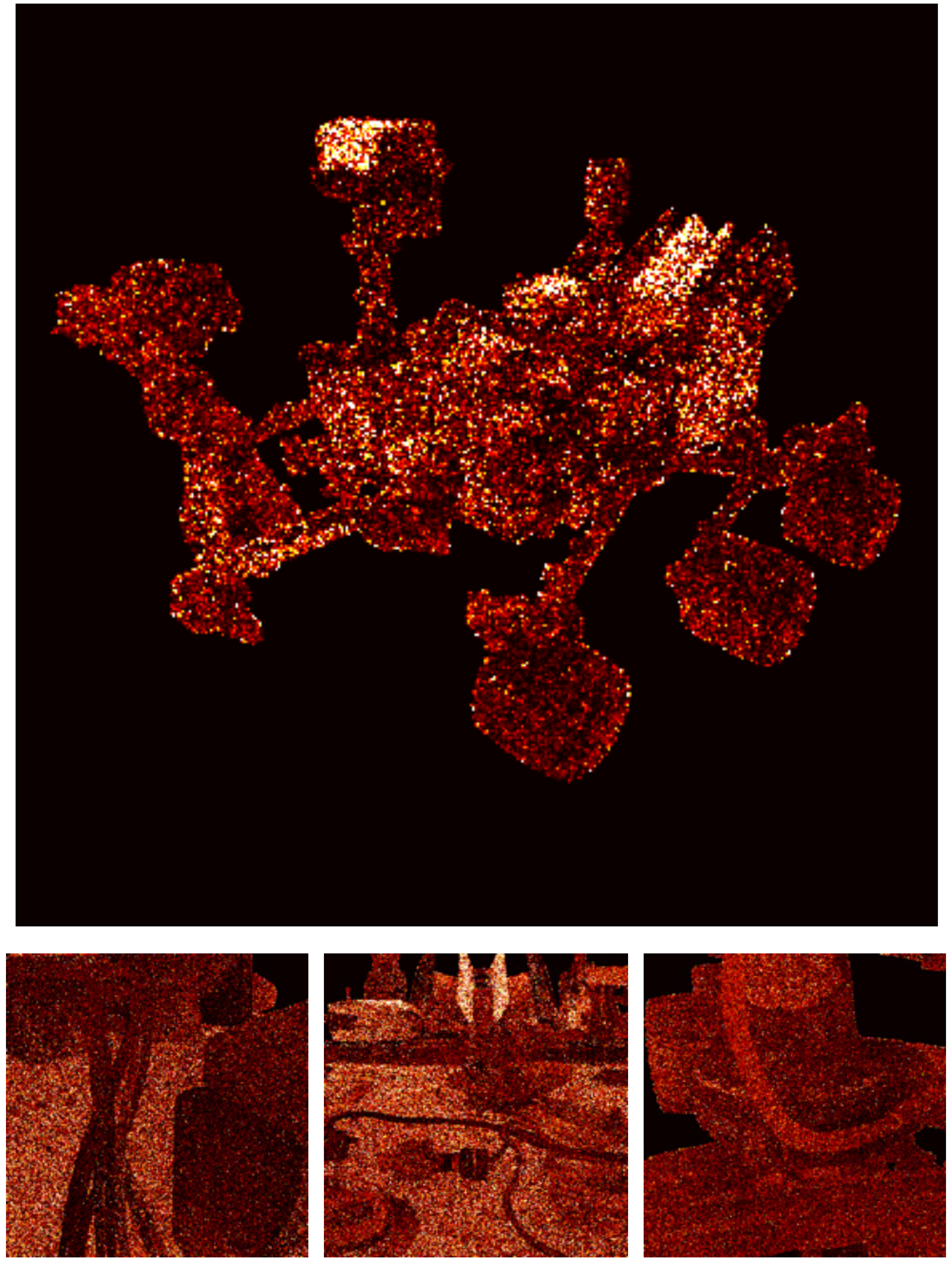


Monte Carlo makes complex geometry tractable

visualization



simulation



Participating media: aggregations of particles

clouds

espresso

tissue

extracellular fluid

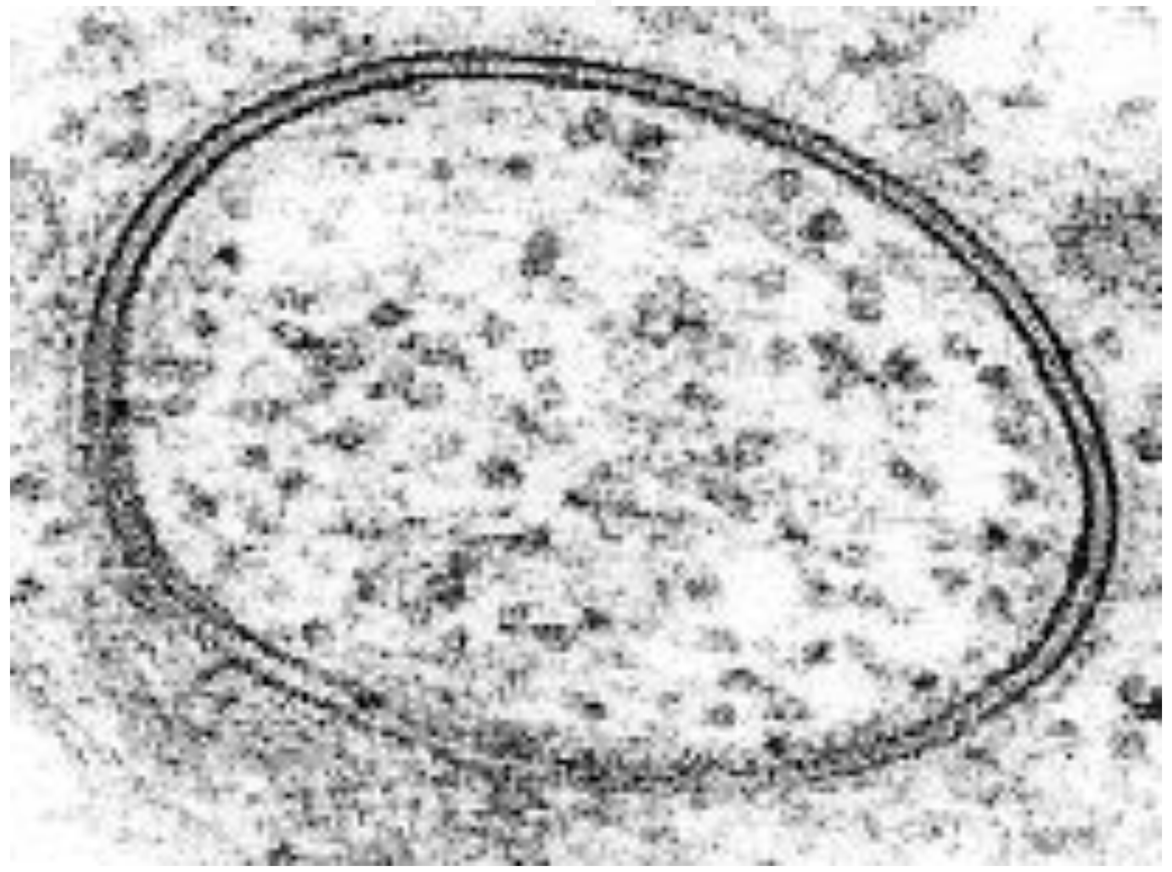
Participating media: aggregations of particles

clouds

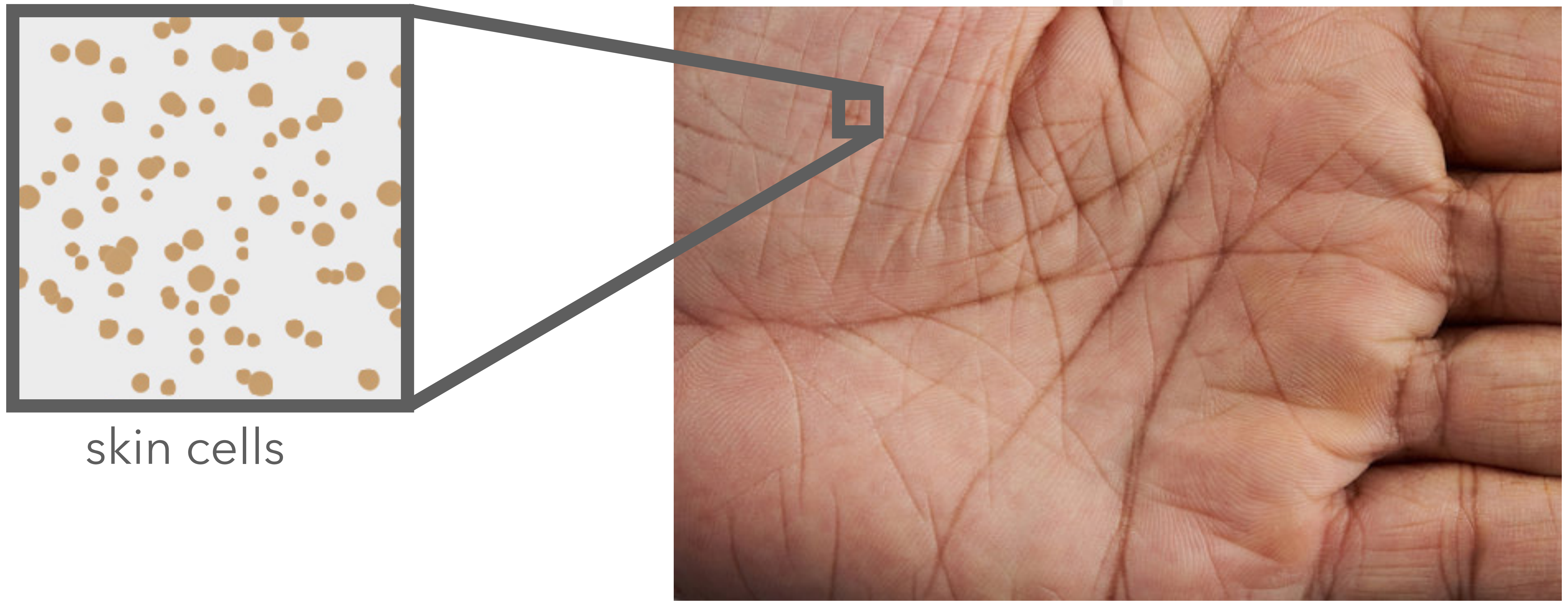
espresso

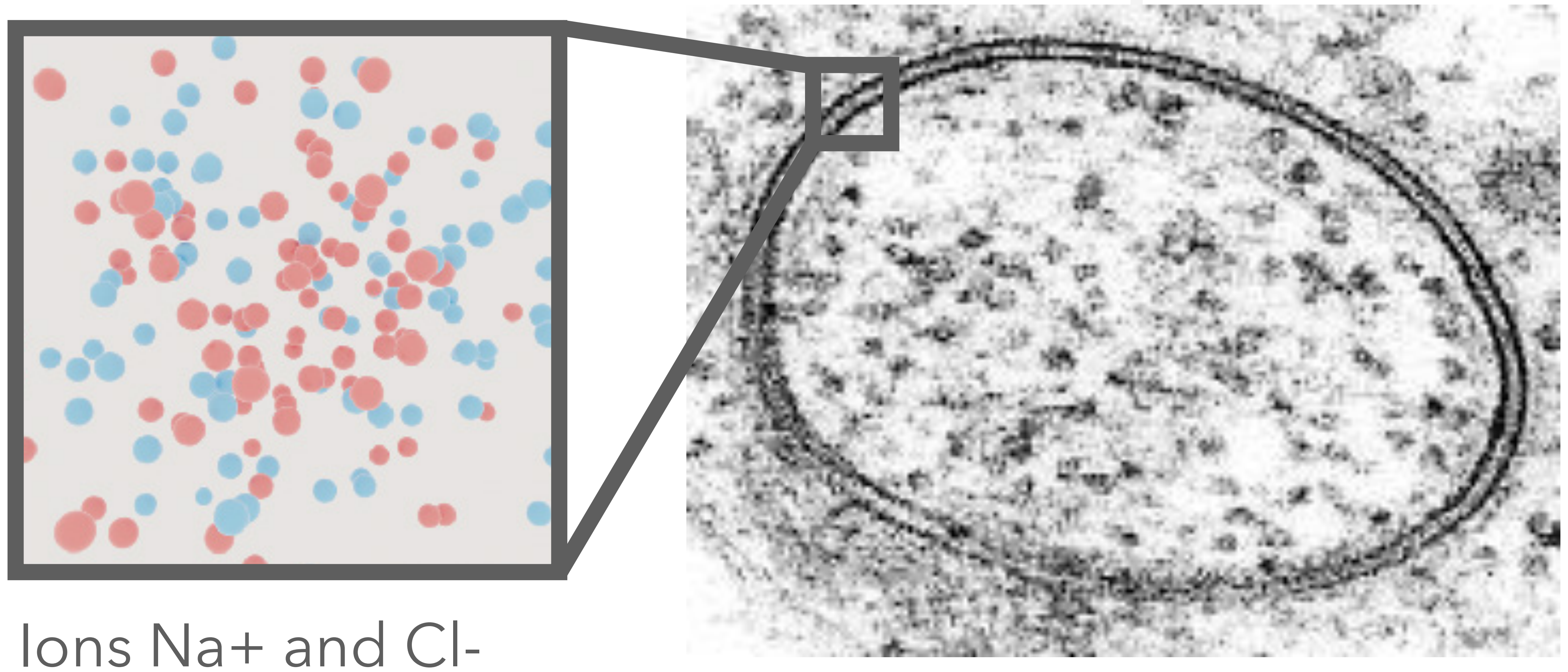
tissue

extracellular fluid

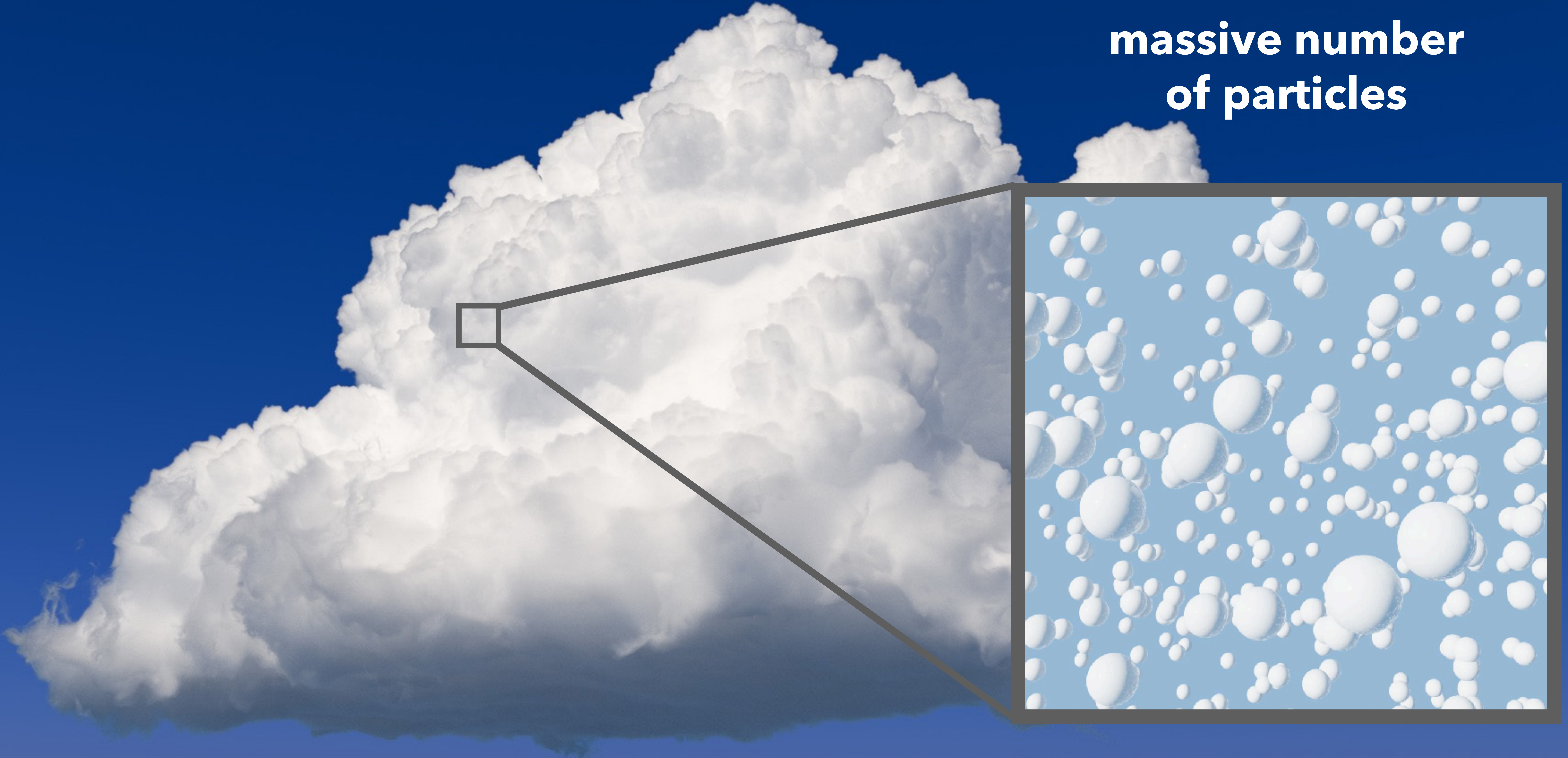


Participating media: aggregations of particles





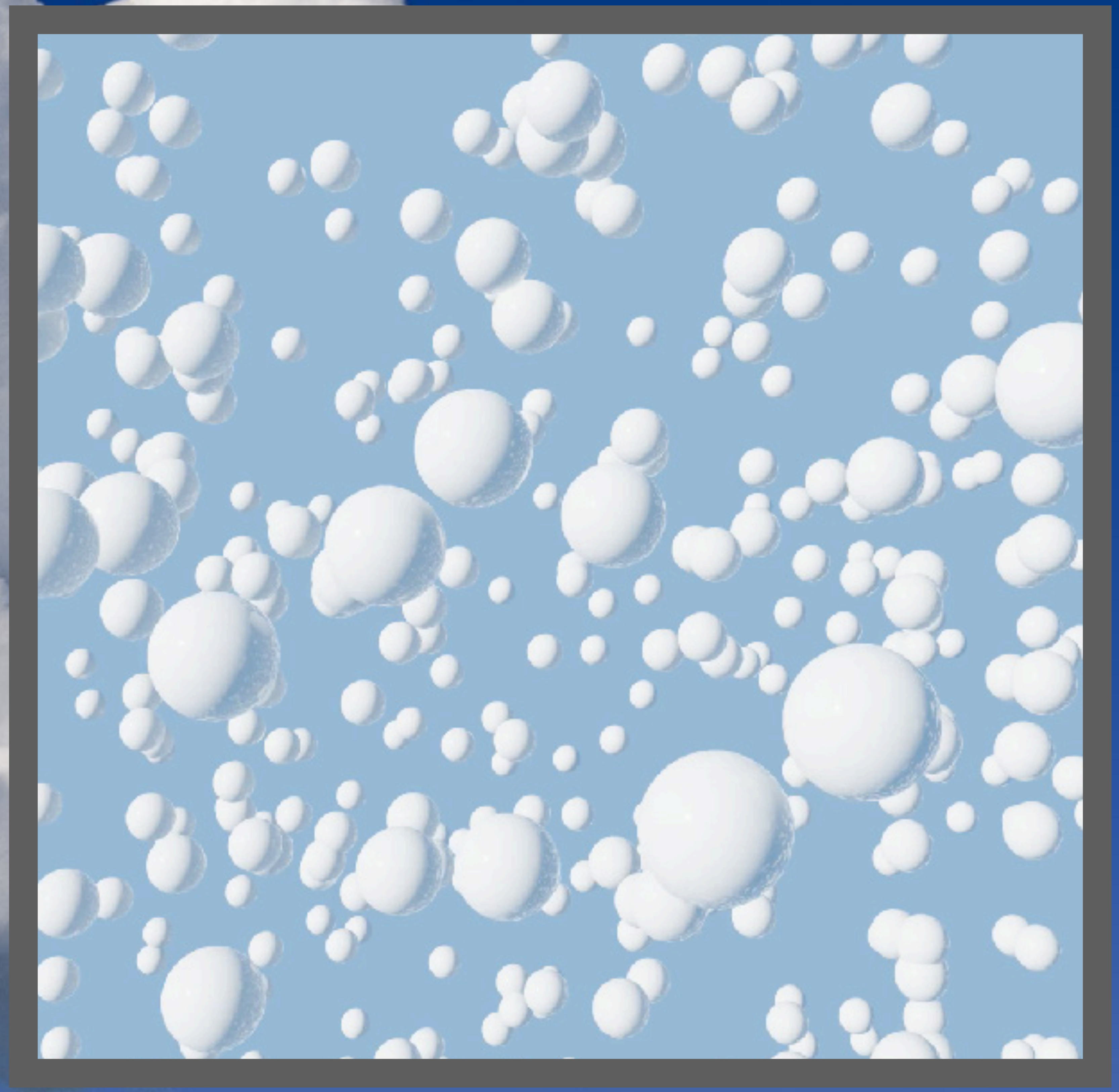
participating media



massive number of particles

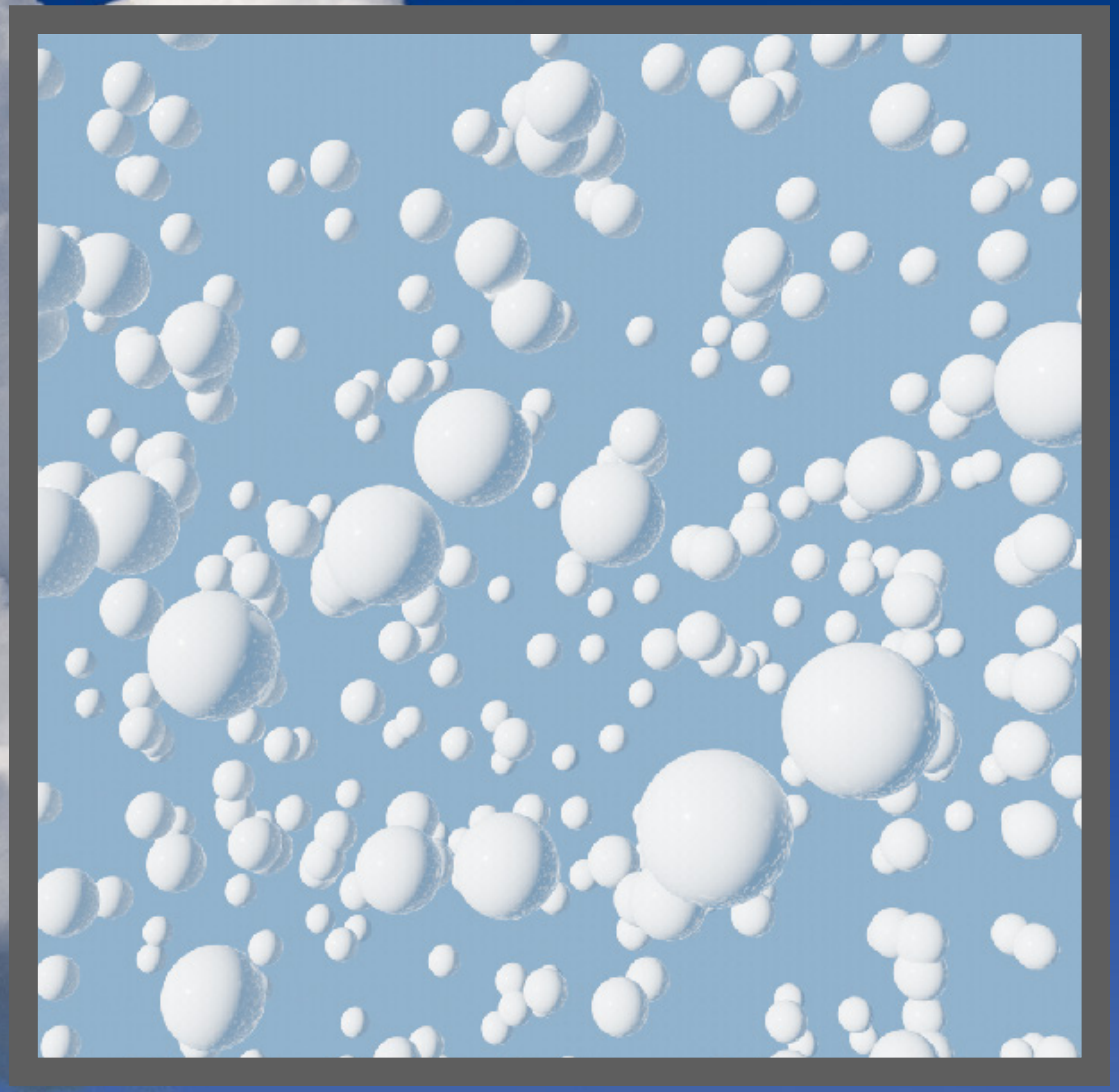
participating media

massive number of particles



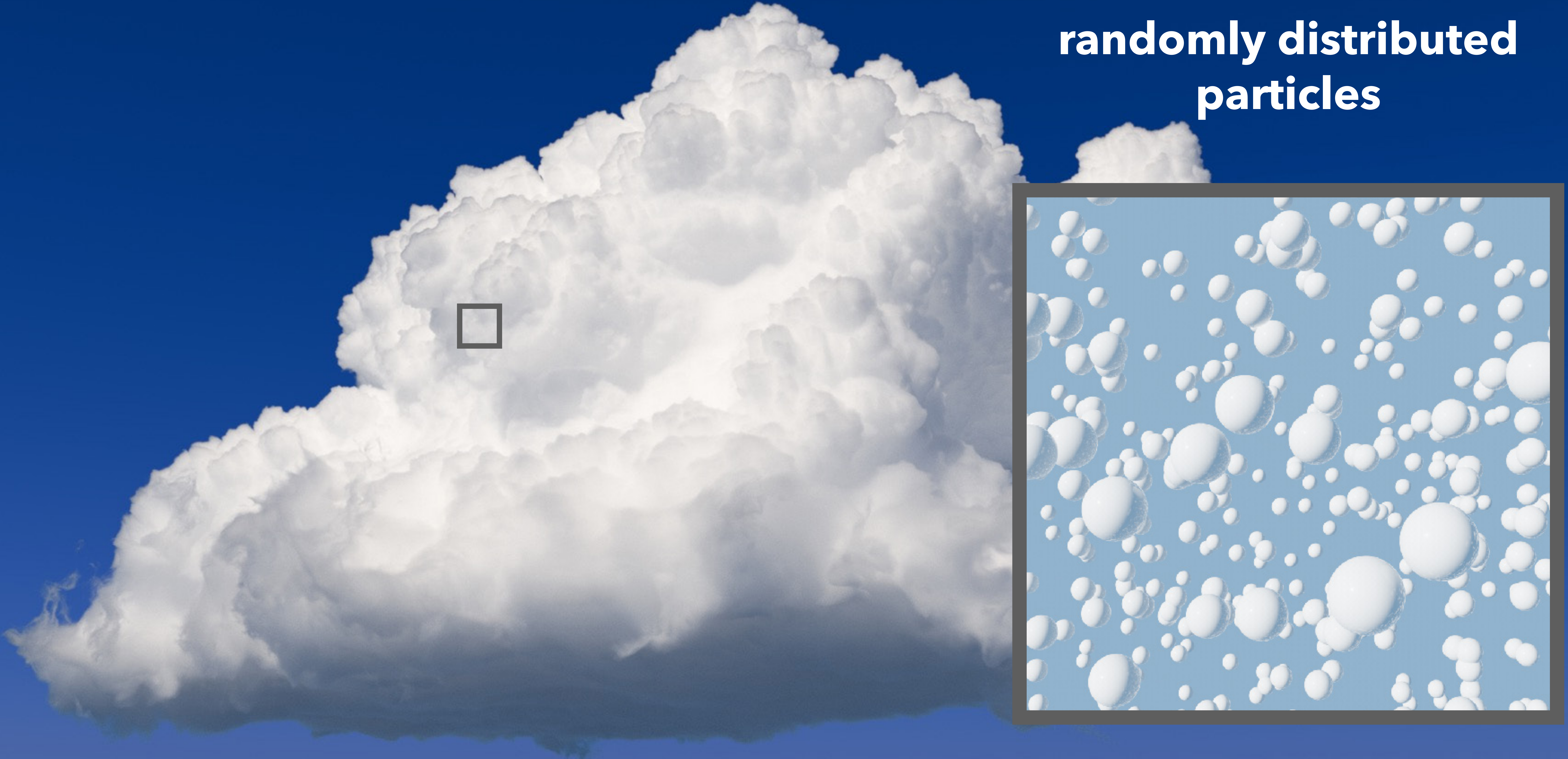
participating media

randomly distributed particles

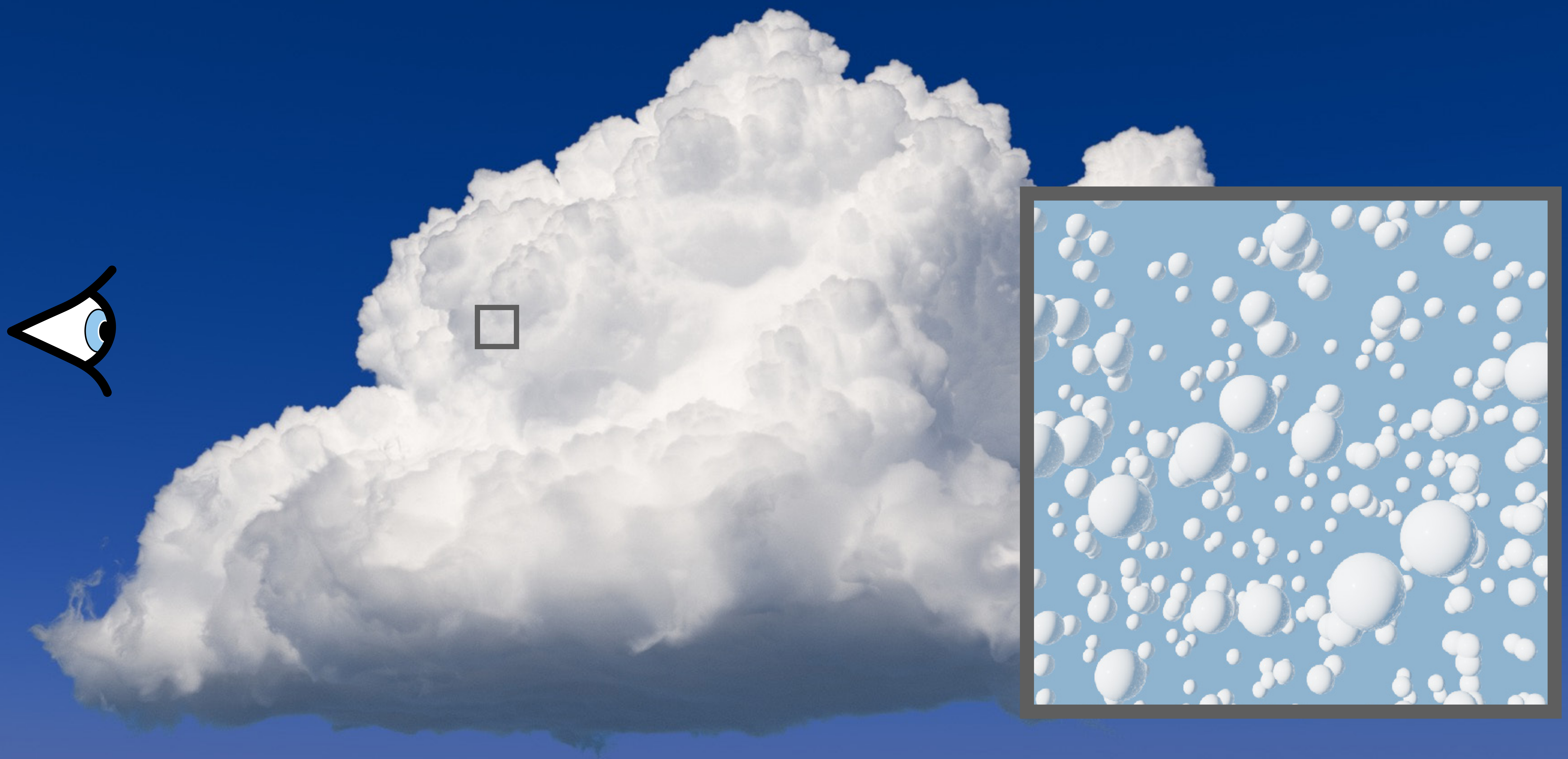


participating media

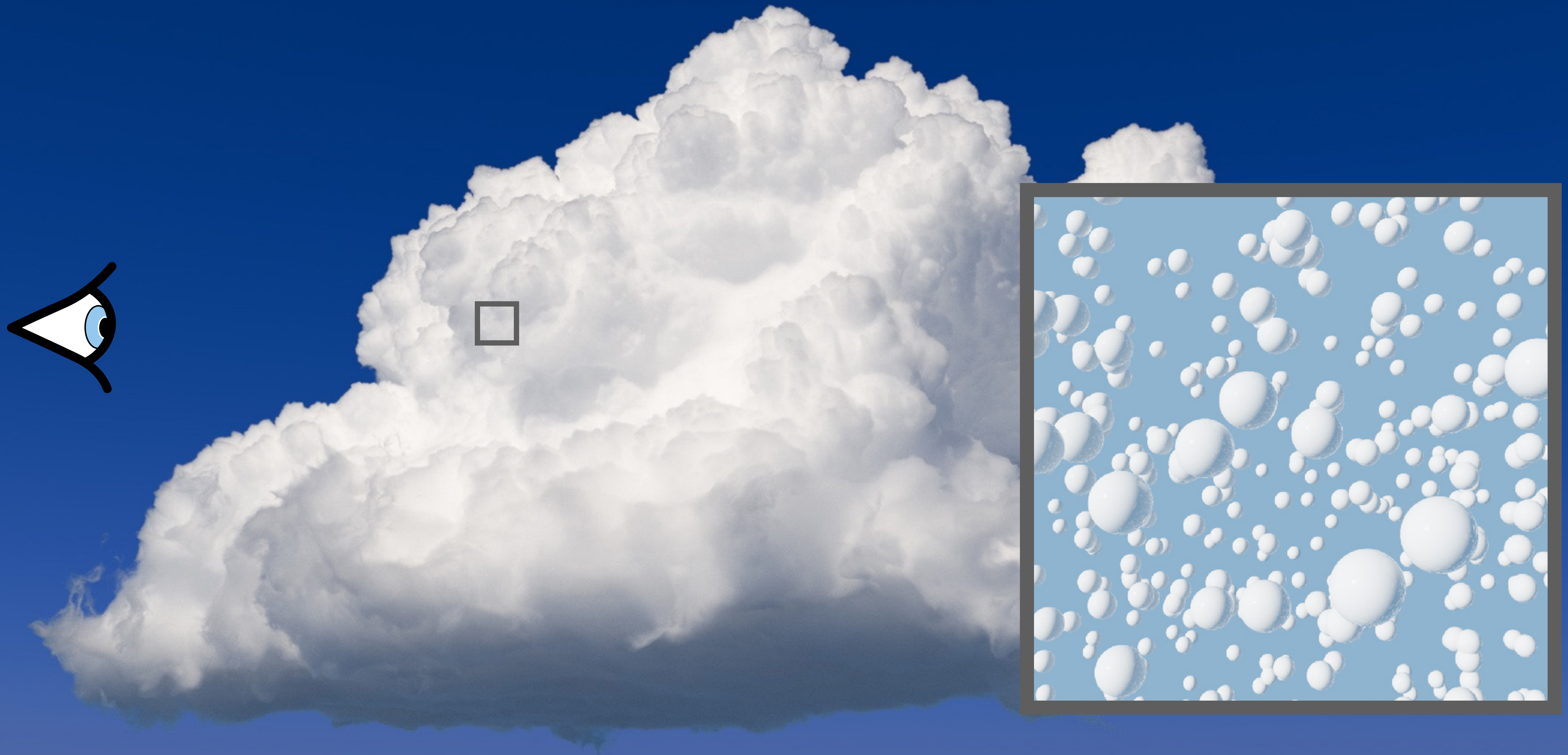
randomly distributed particles



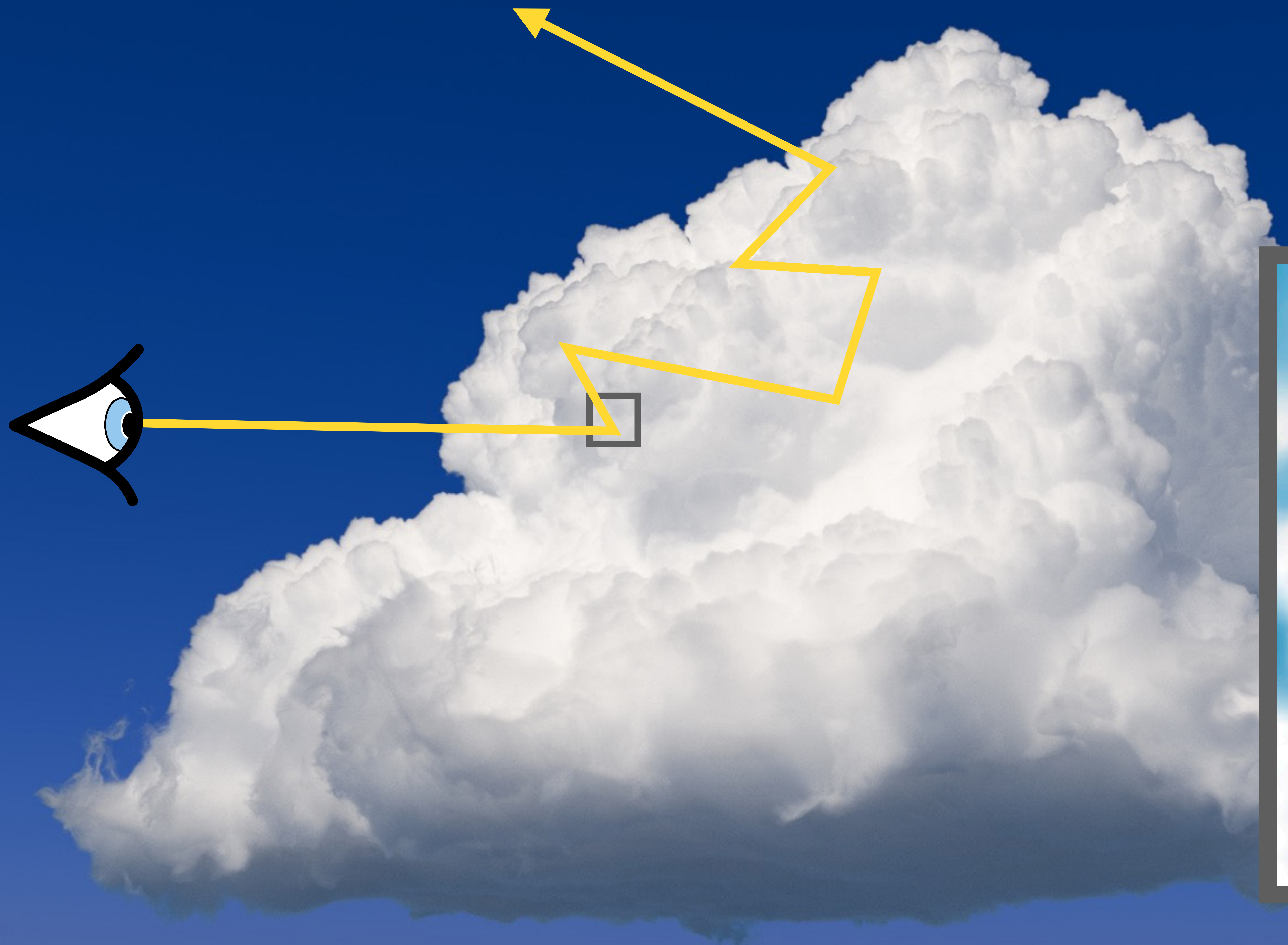
participating media



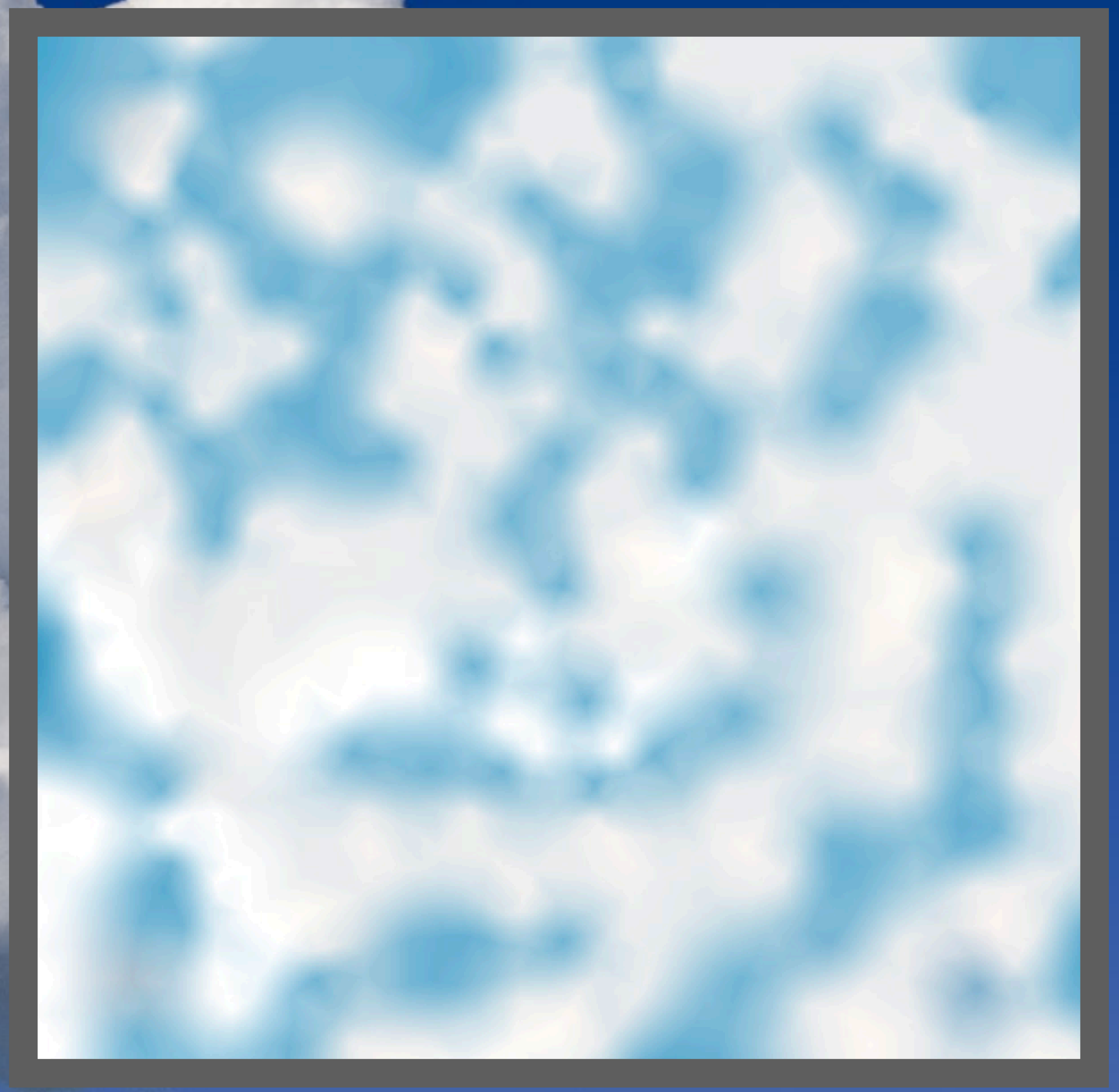
rendering of participating media



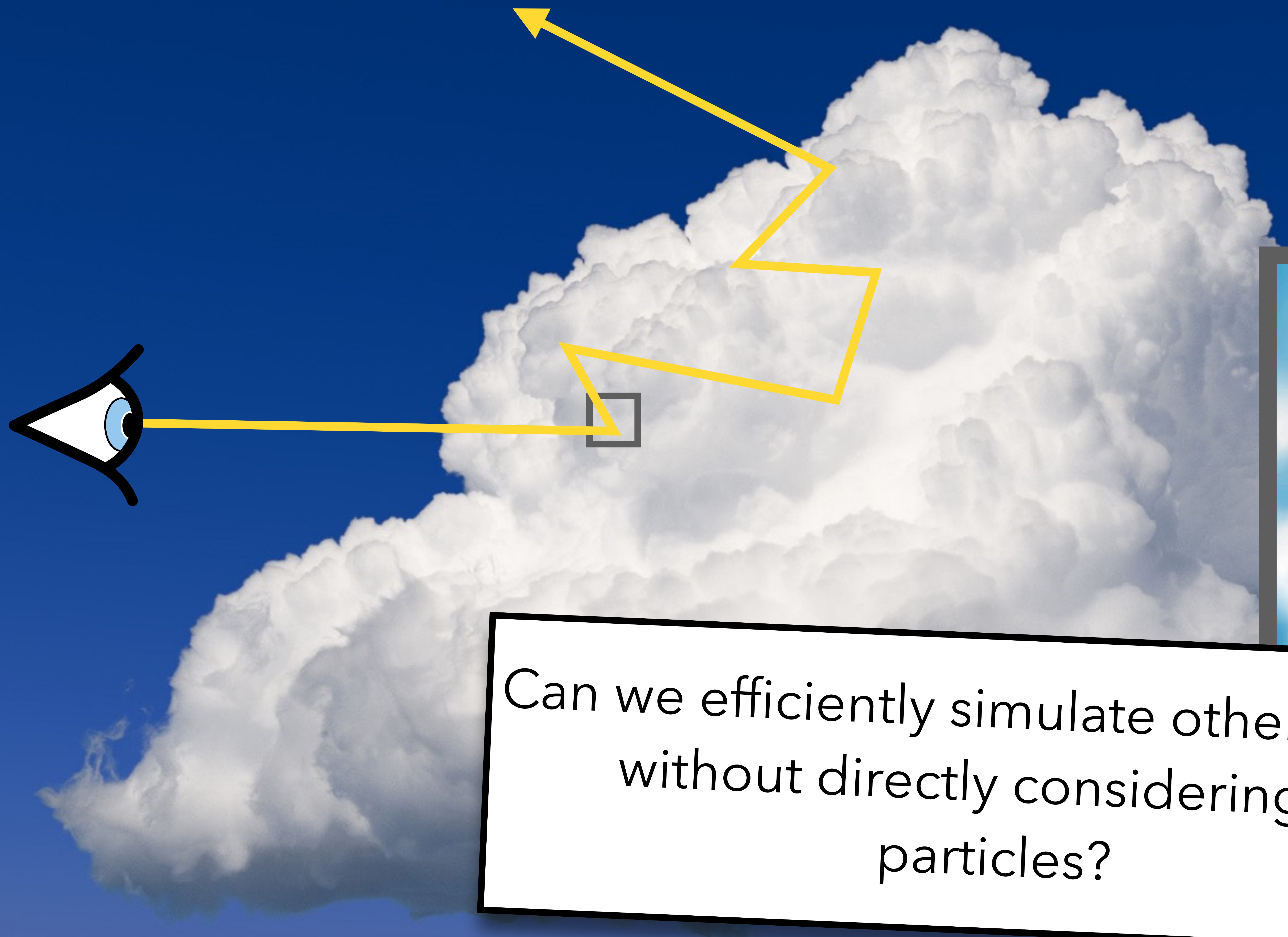
rendering of participating media



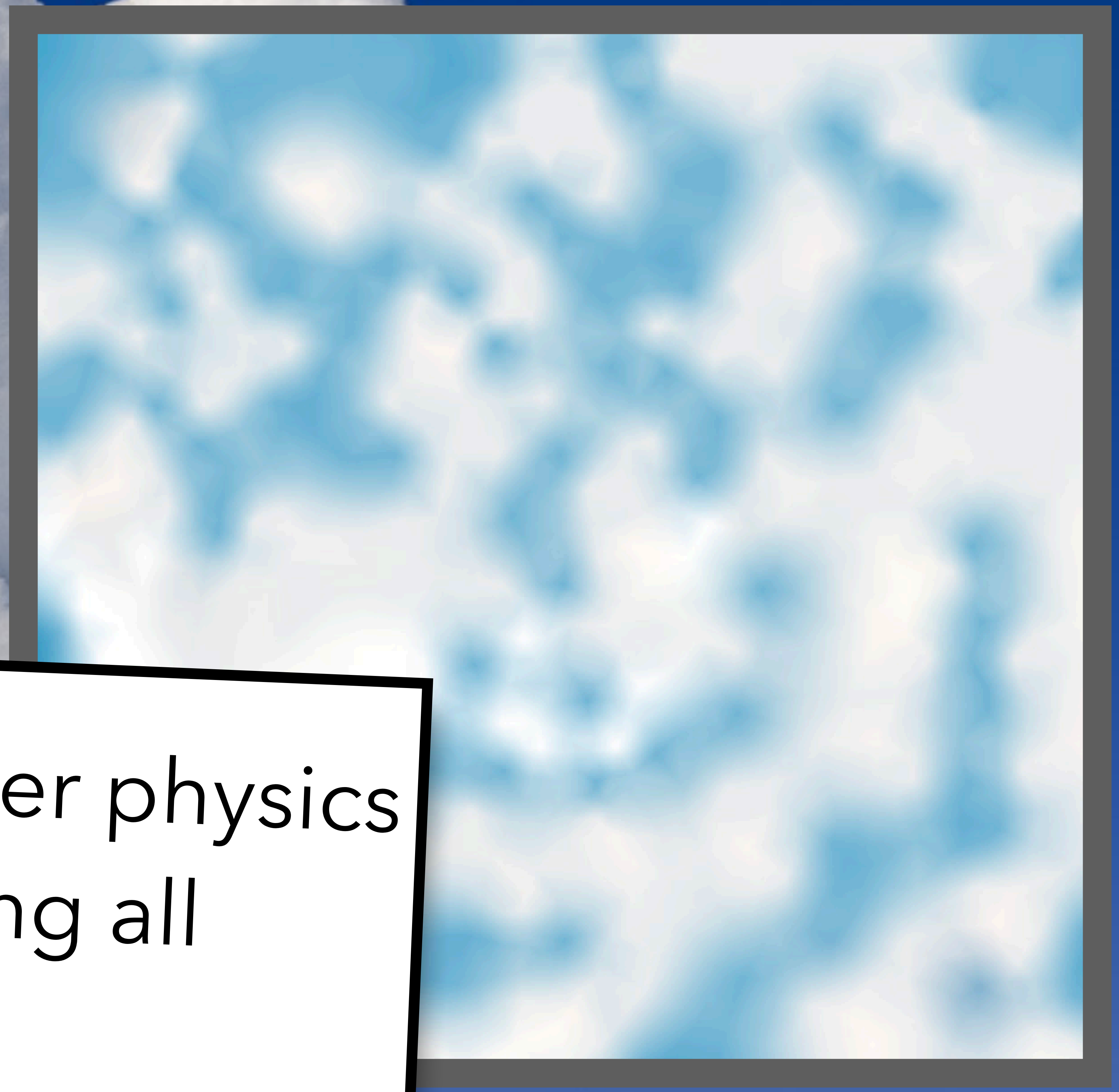
only aggregate effect matters!



rendering of participating media

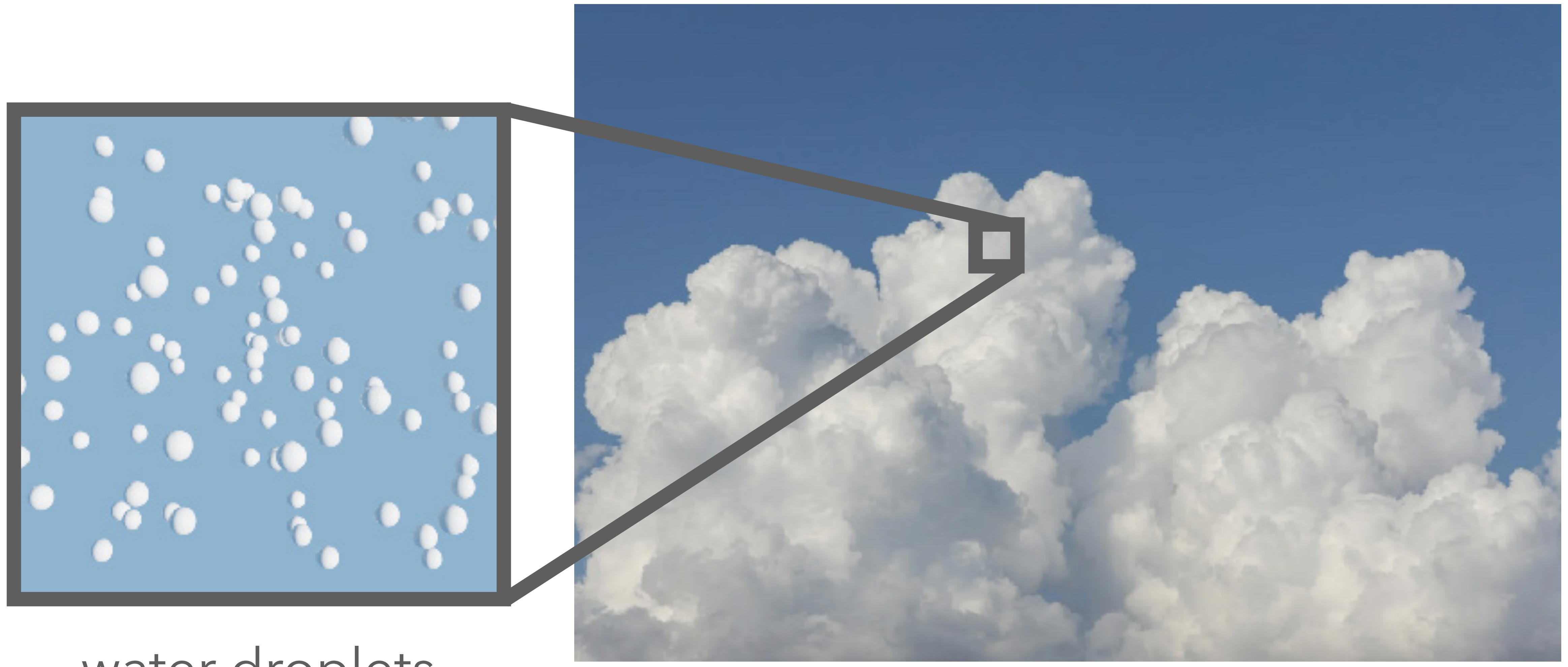


only aggregate effect matters!

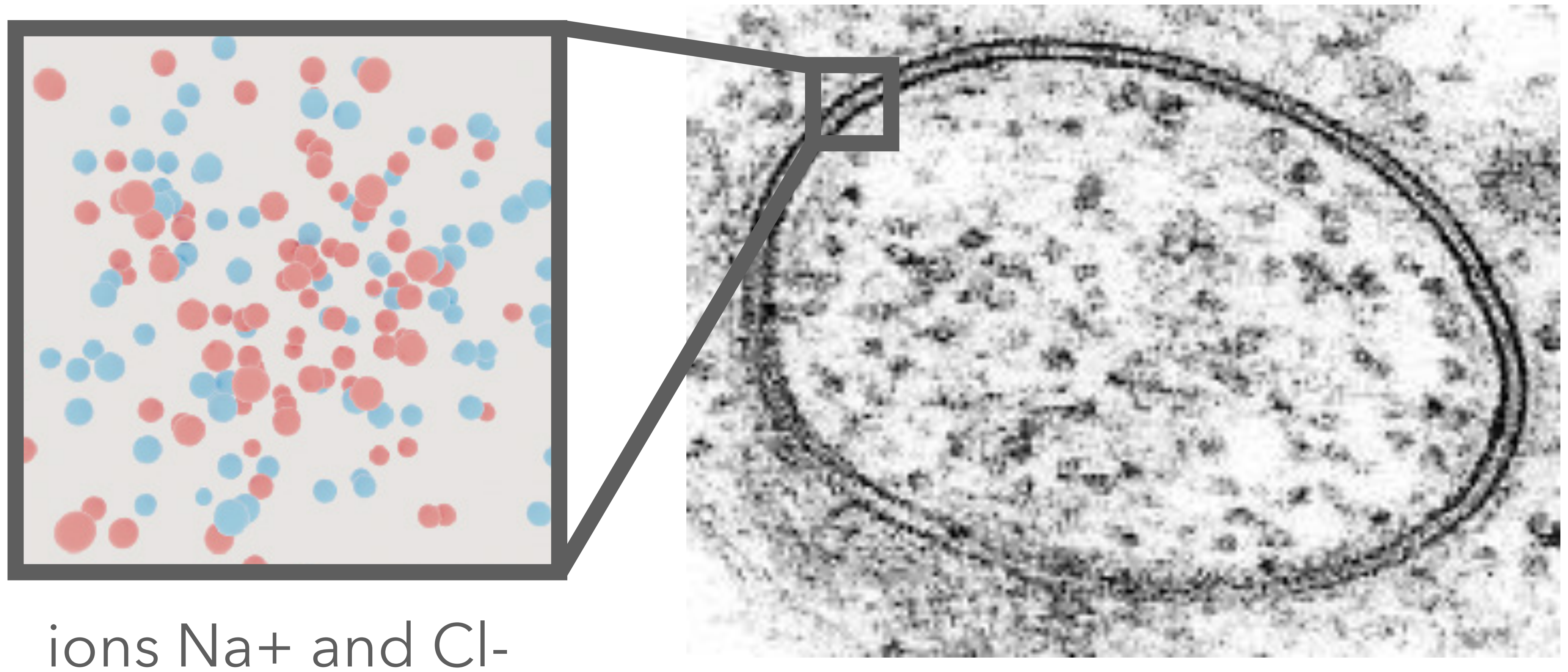


Can we efficiently simulate other physics without directly considering all particles?

Physical systems involving participating media

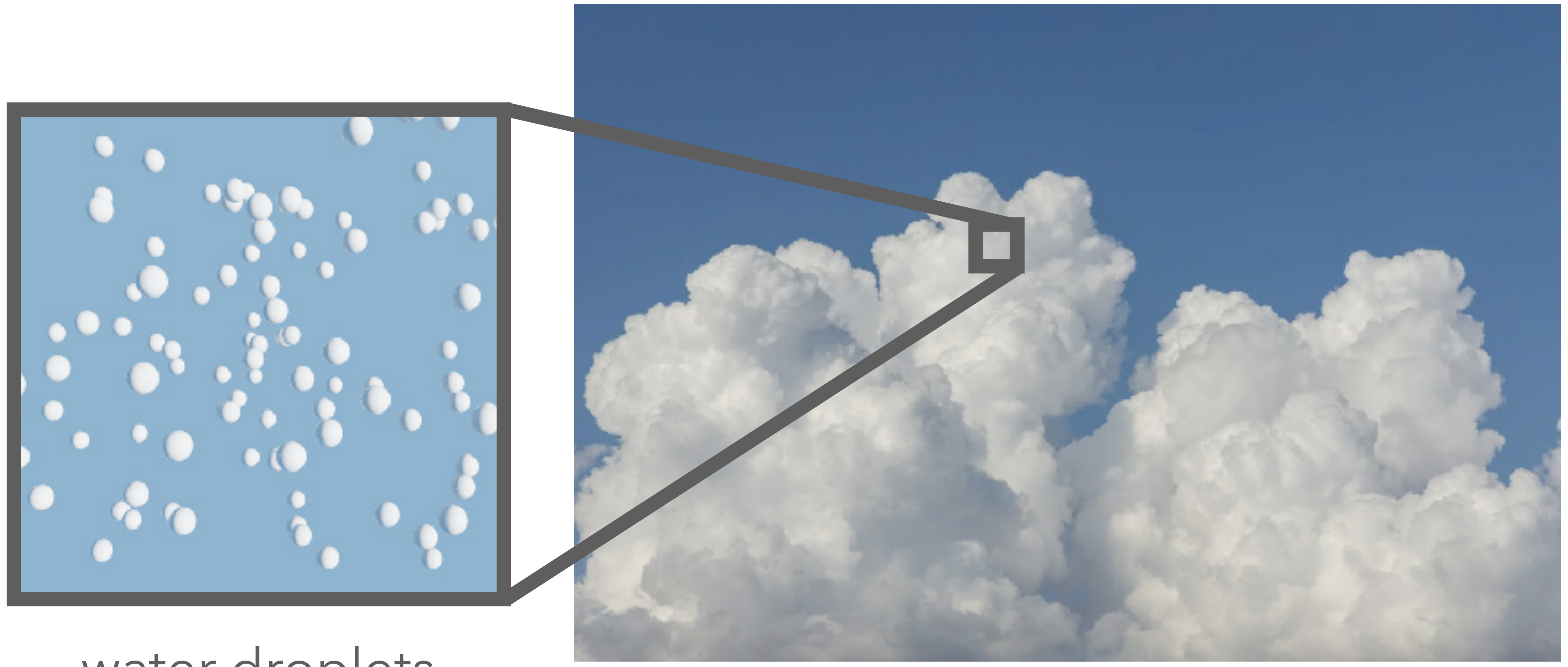


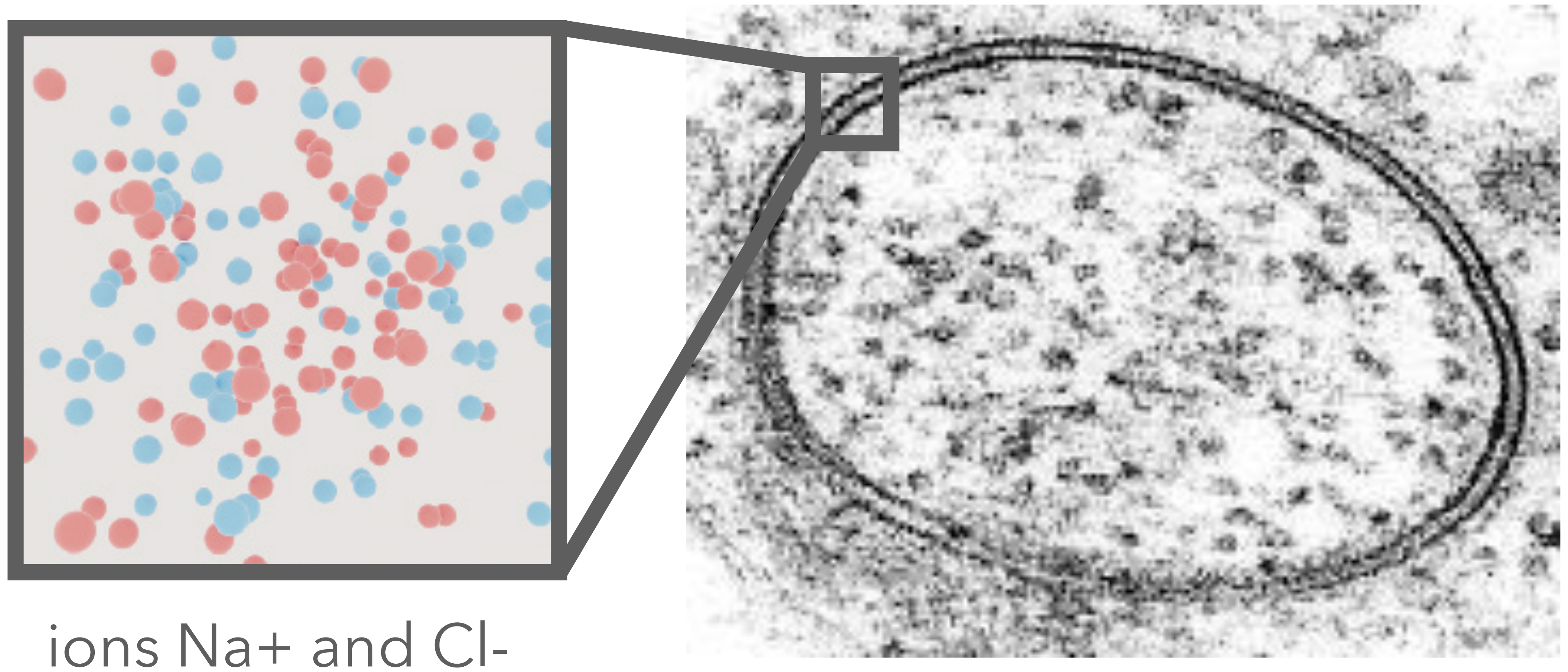
water droplets



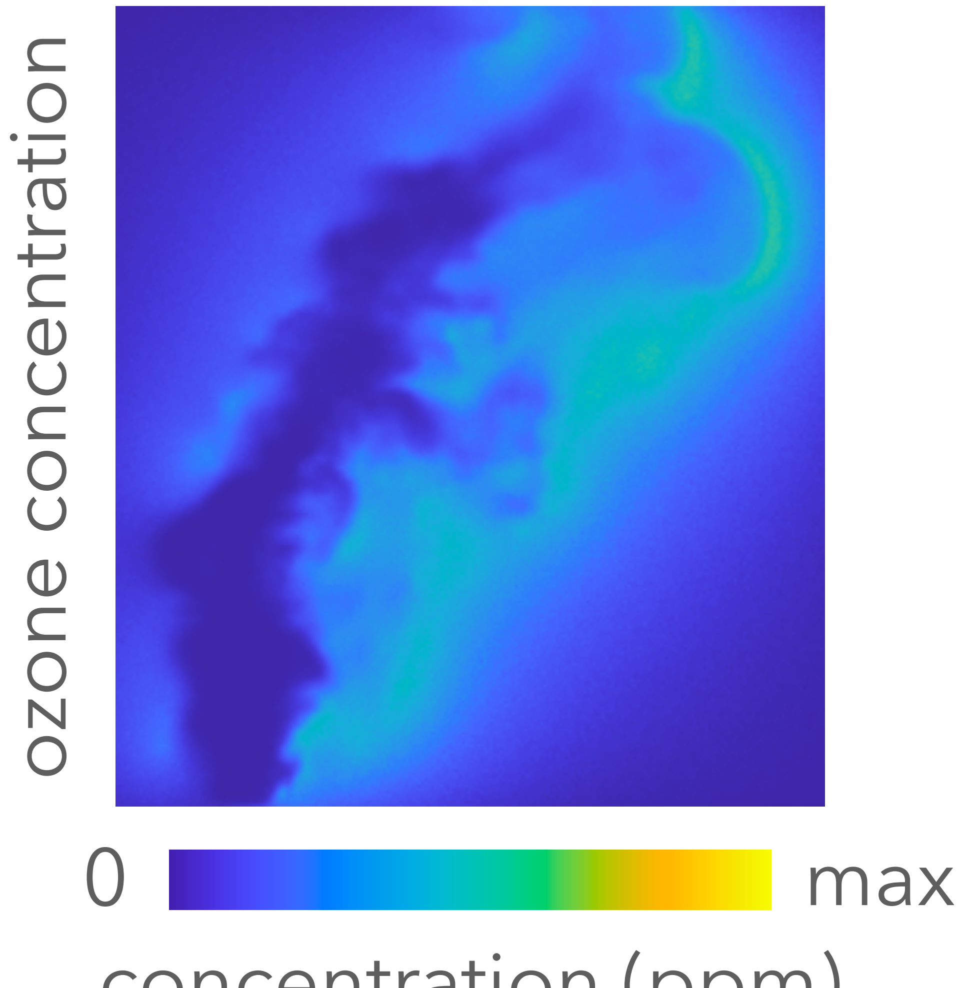
ions Na+ and Cl-

Physical systems involving participating media

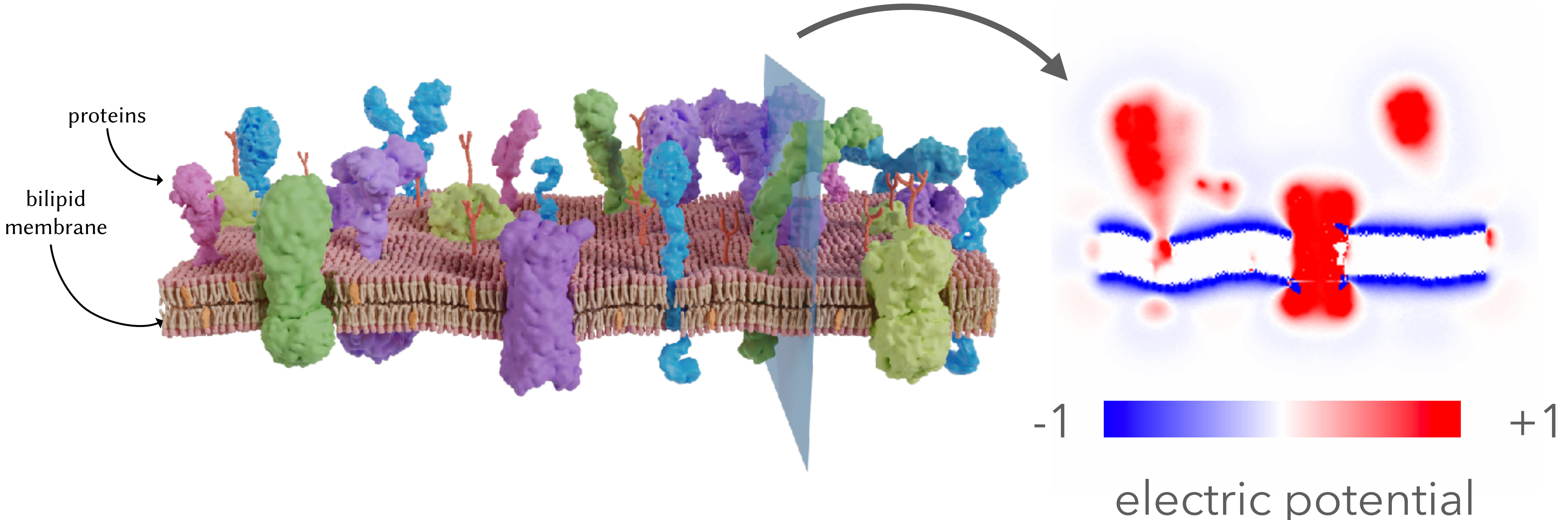




atmospheric photochemical systems



electrostatic potential near membrane



Physical systems involving participating media

Laplace equation

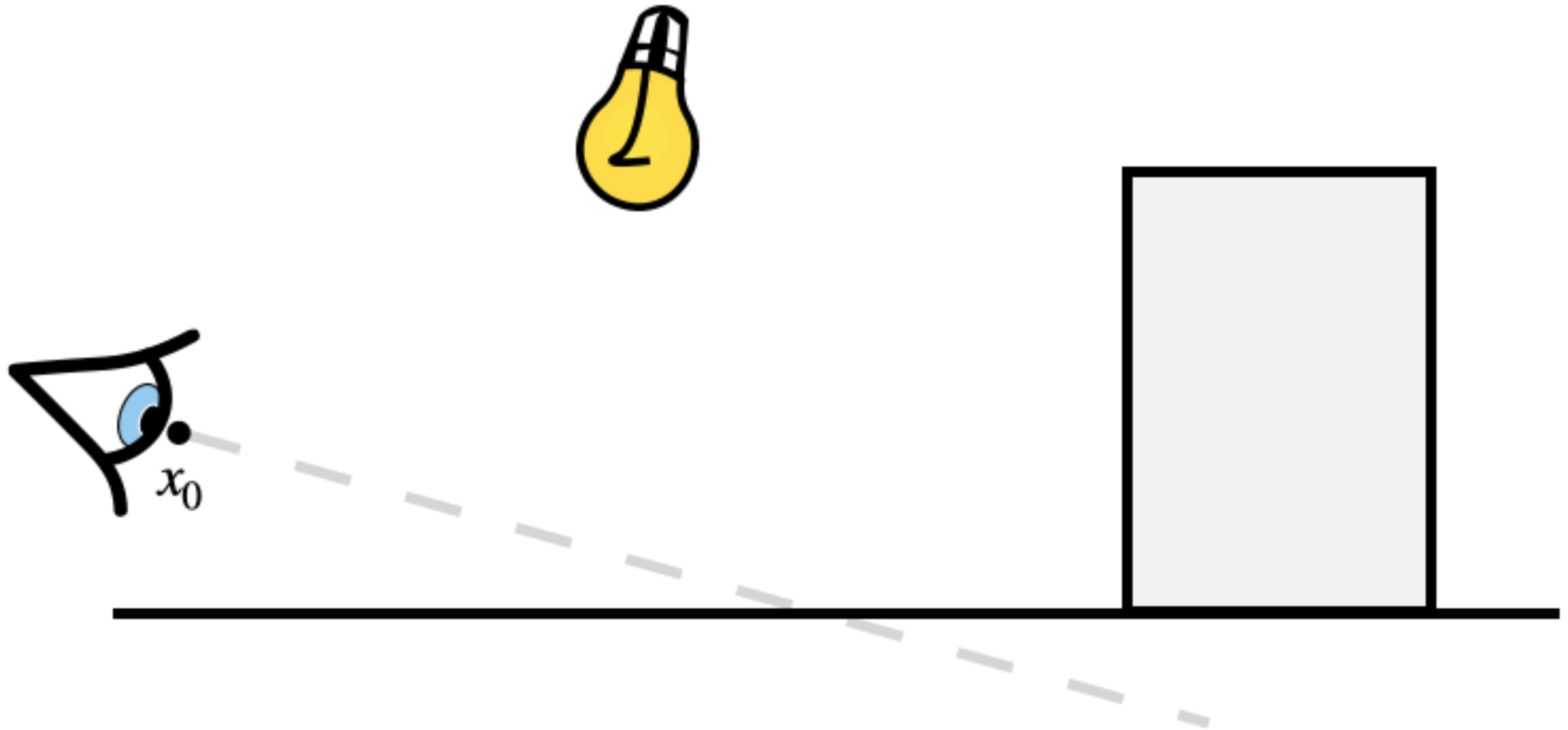
$$\Delta u = 0$$

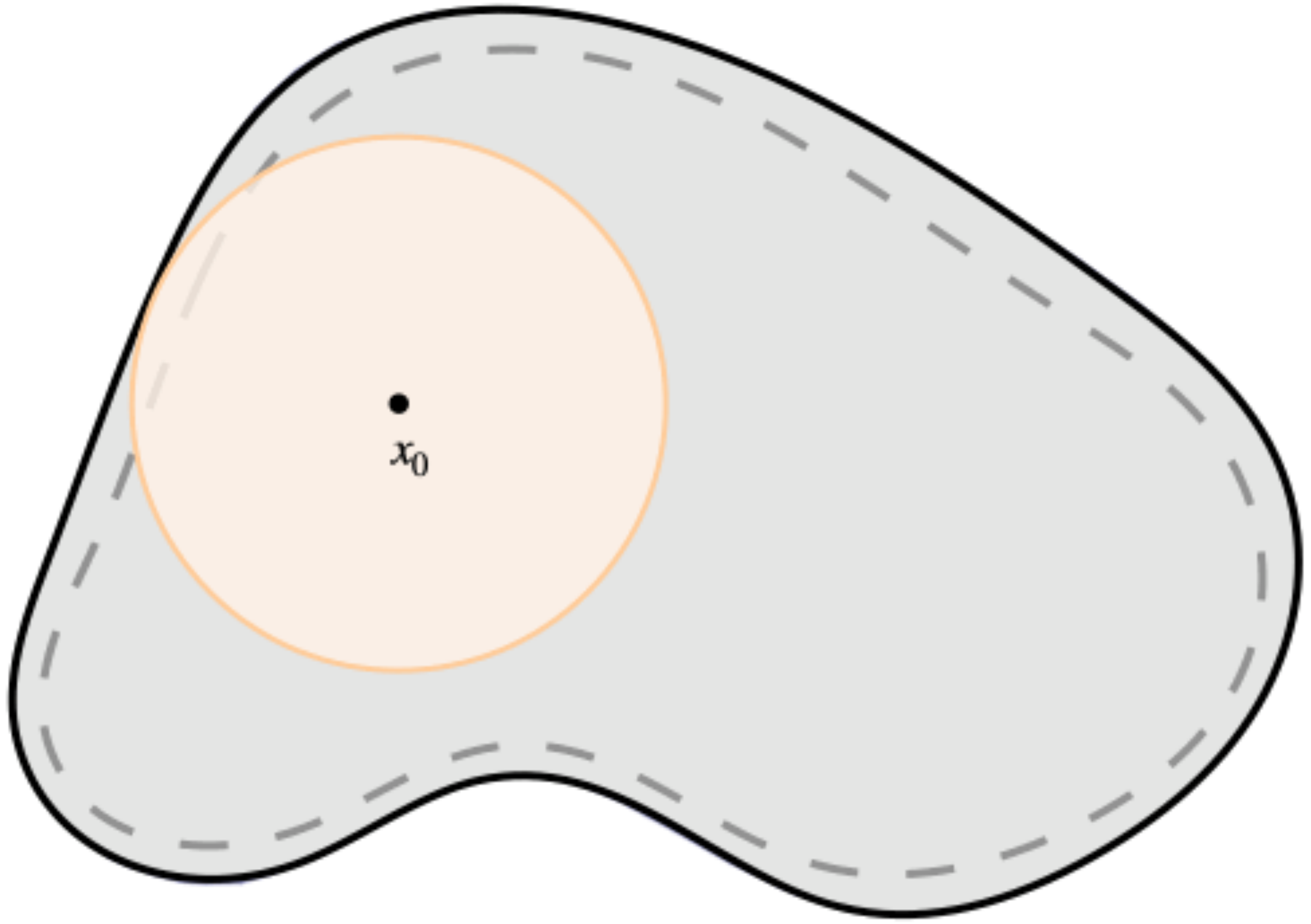
$$\Delta := \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$$

Connection to rendering

rendering
[Kajiya 1986]

walk on spheres
[Mueller 1956, Sawhney and Crane 2020]



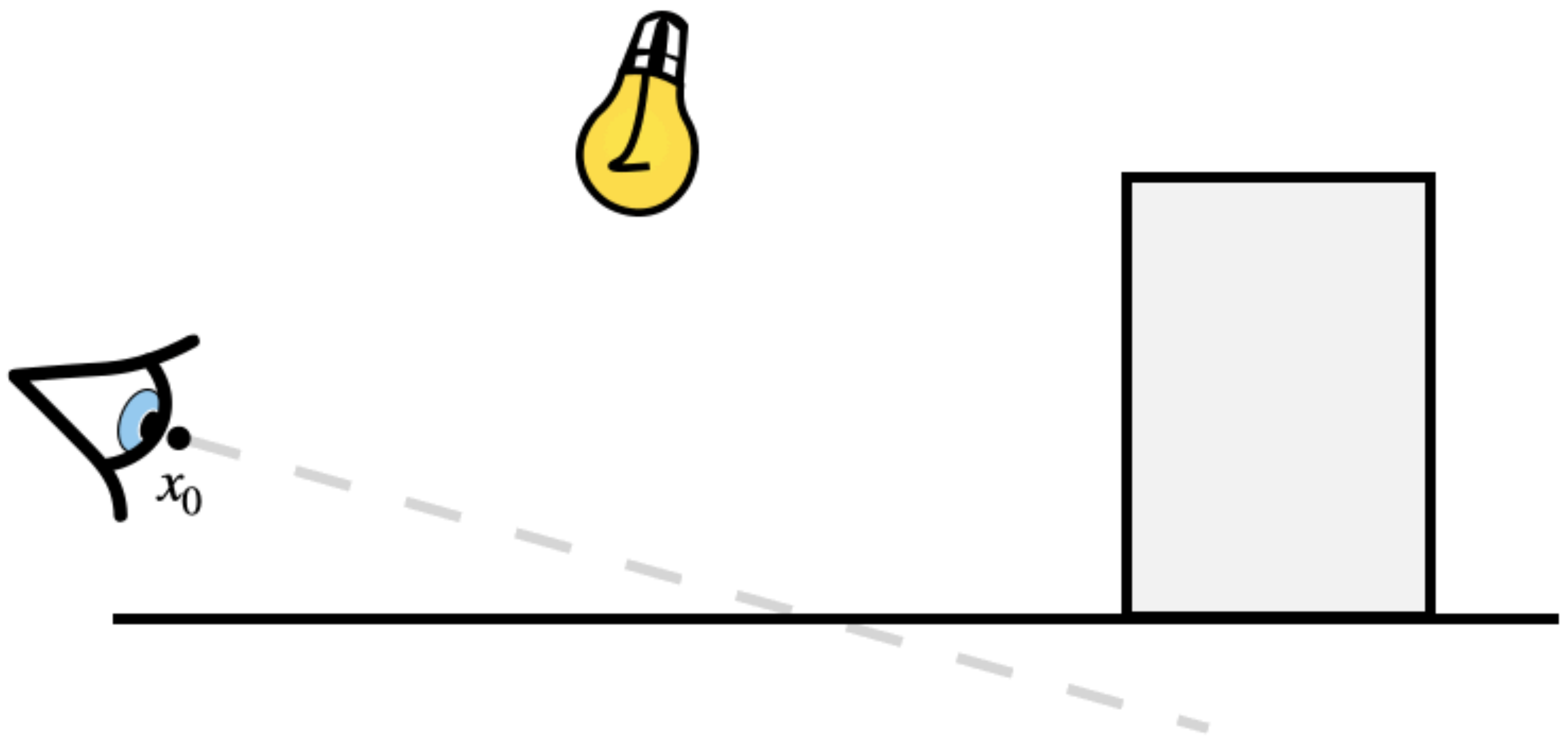


visualization
light transport

simulation
conduction, electrostatics, diffusion, etc.

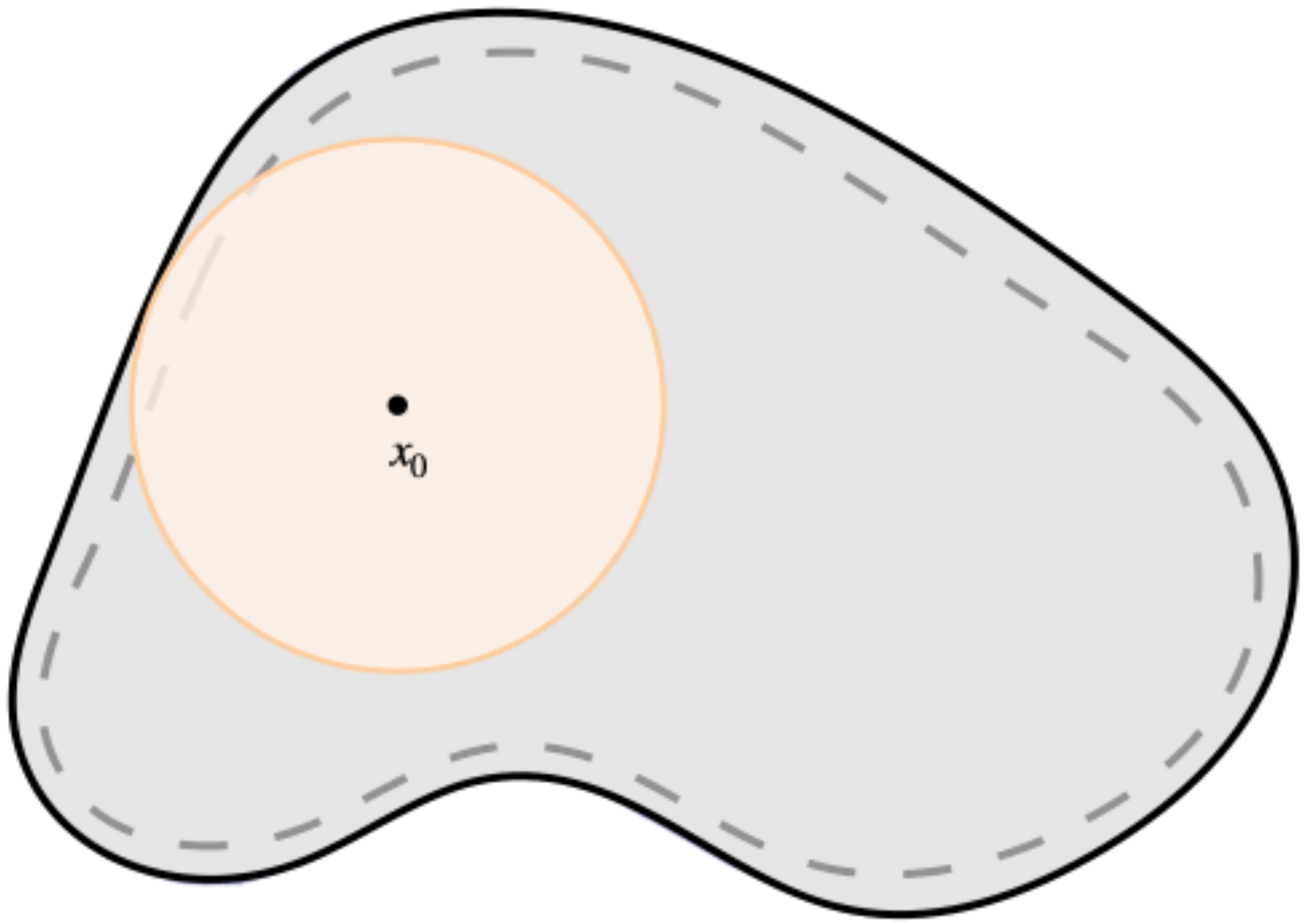
Connection to rendering

rendering
[Kajiya 1986]



visualization
light transport

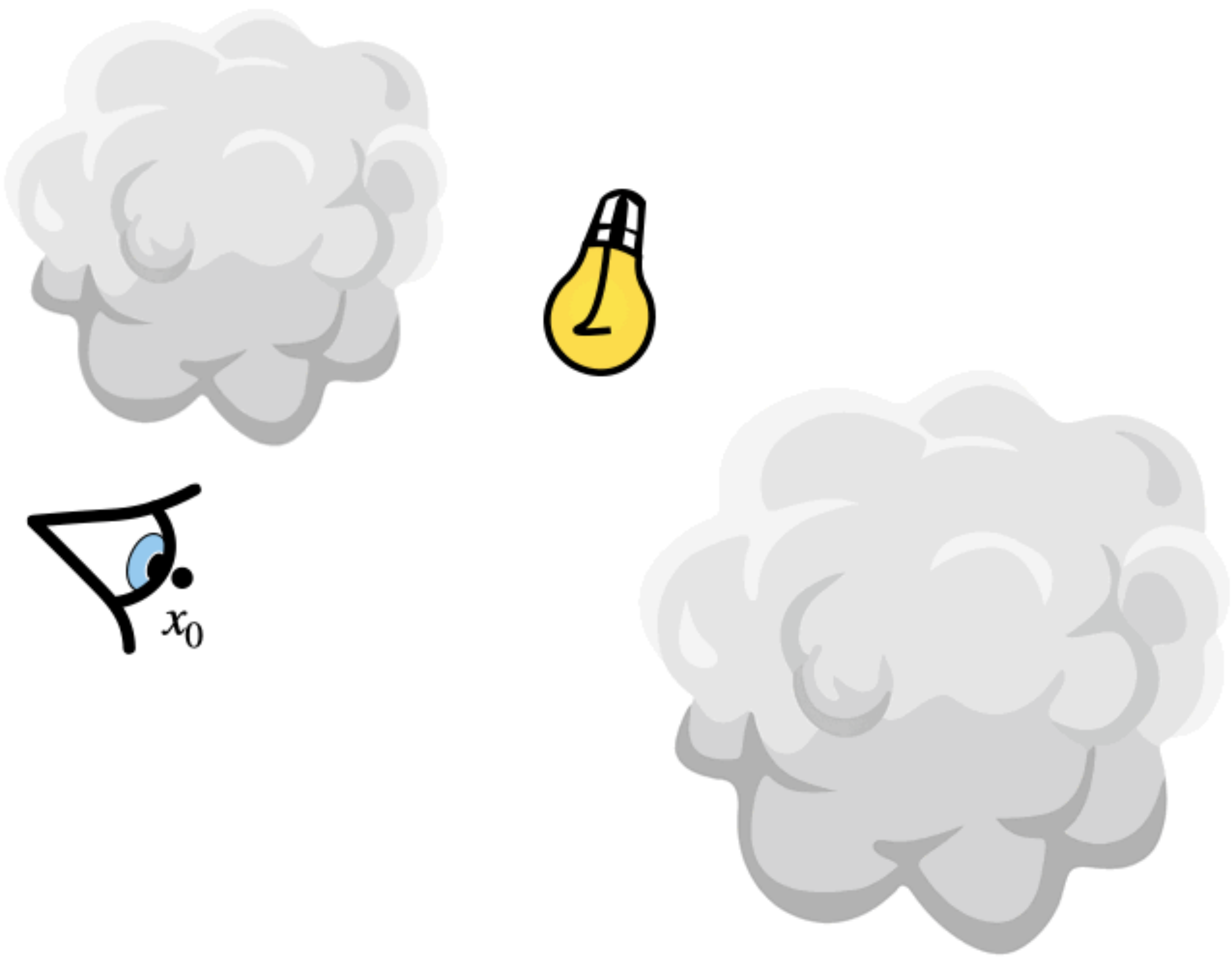
walk on spheres
[Mueller 1956, Sawhney and Crane 2020]



simulation
conduction, electrostatics, diffusion, etc.

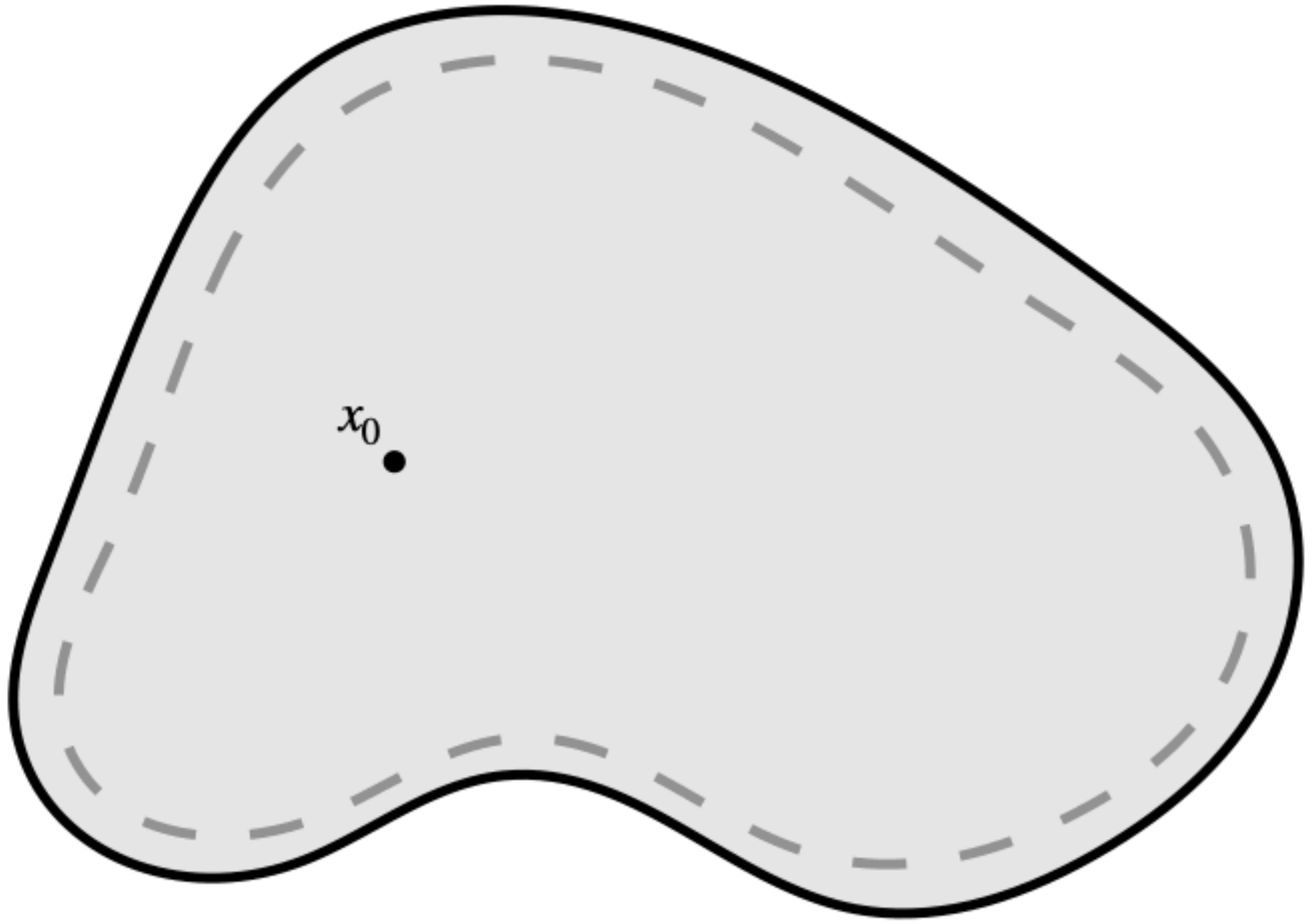
Generalizing to participating media

volume rendering
[Kajiya and Von Herzen 1984]



visualization
light transport

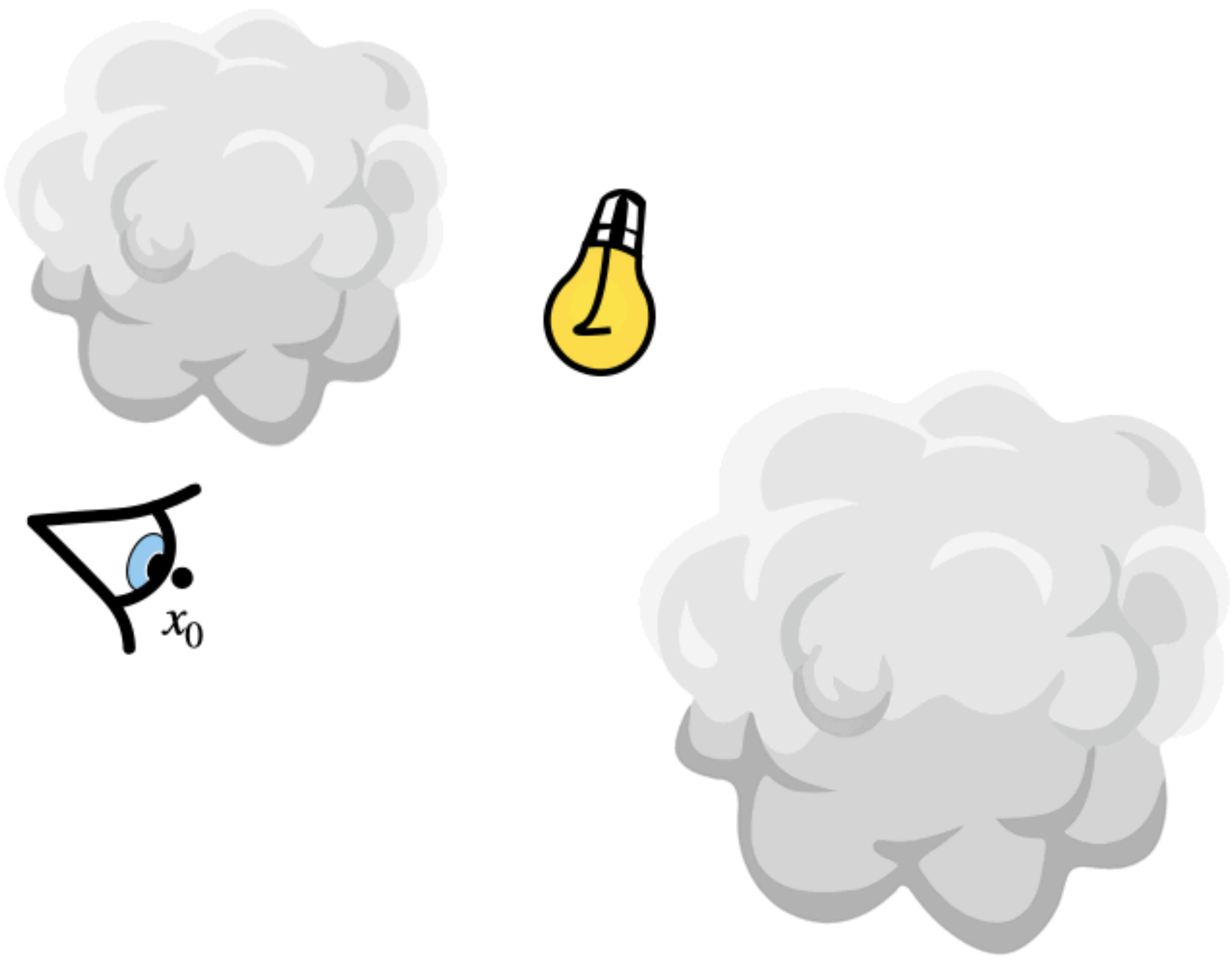
volumetric walk on spheres
ours



simulation
conduction, electrostatics, diffusion, etc.

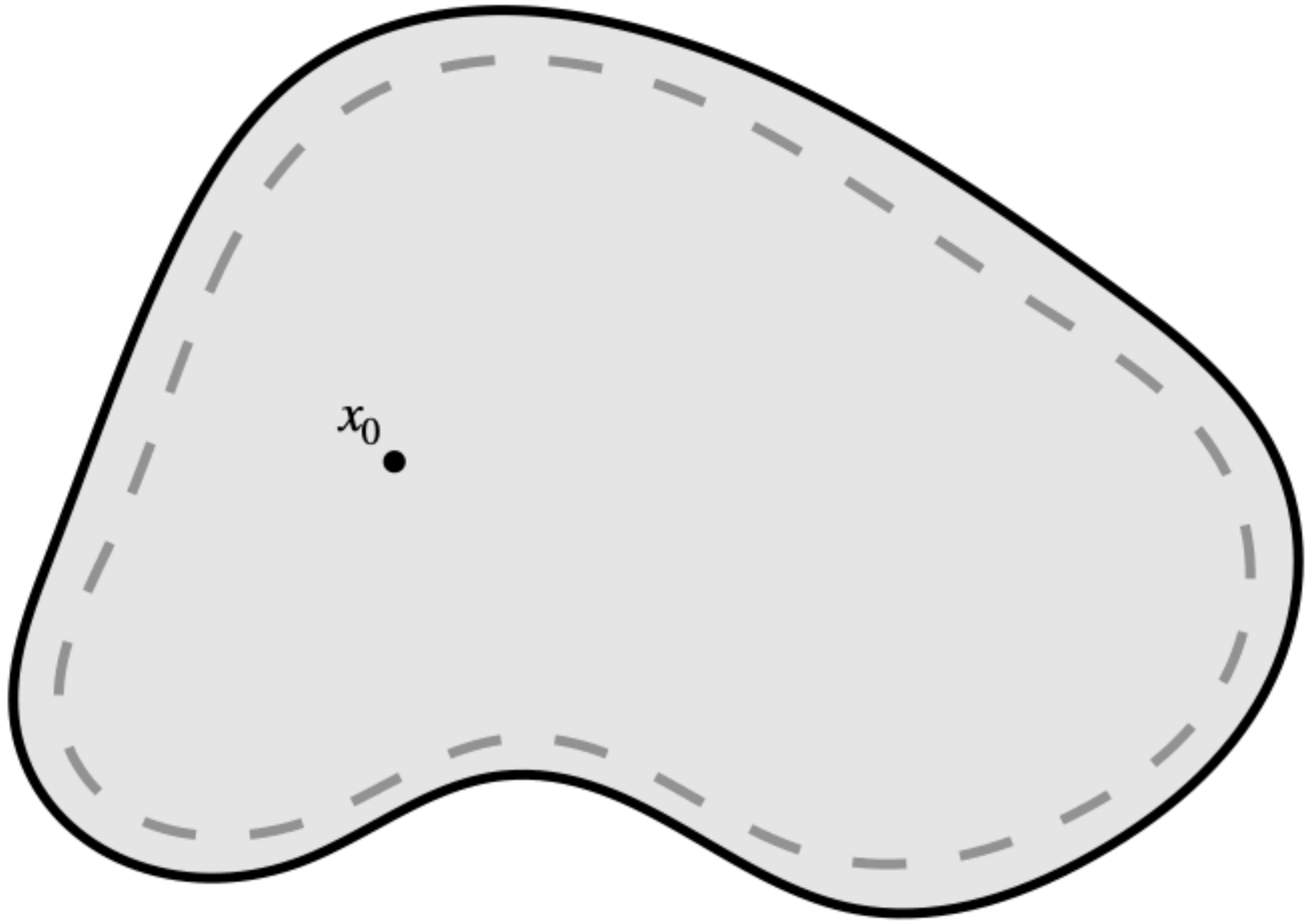
Generalizing to participating media

volume rendering
[Kajiya and Von Herzen 1984]



visualization
light transport

volumetric walk on spheres
ours



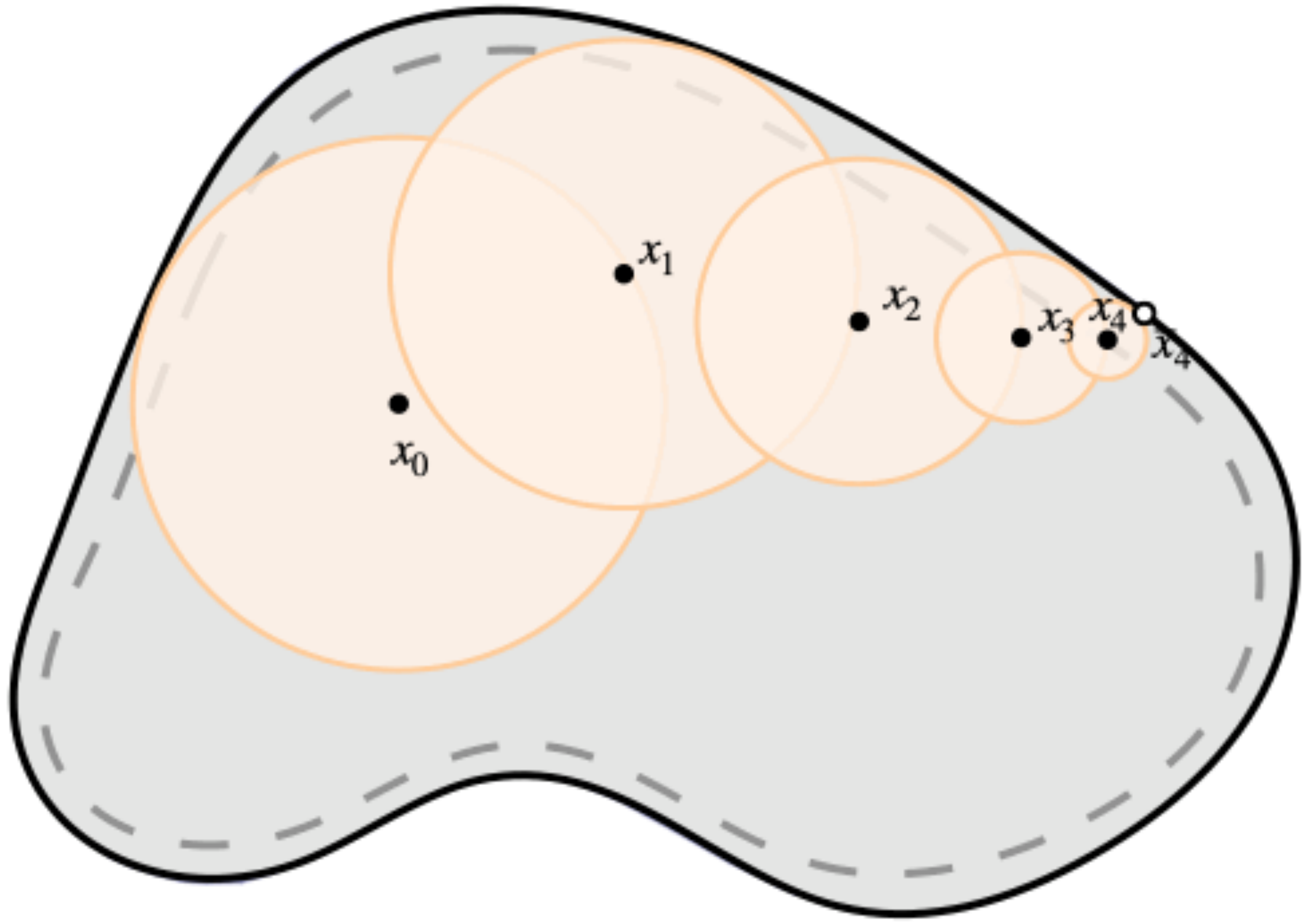
simulation
conduction, electrostatics, diffusion, etc.

Volumetric walk on spheres

walk on spheres

[Mueller 1956, Sawhney and Crane 2020]

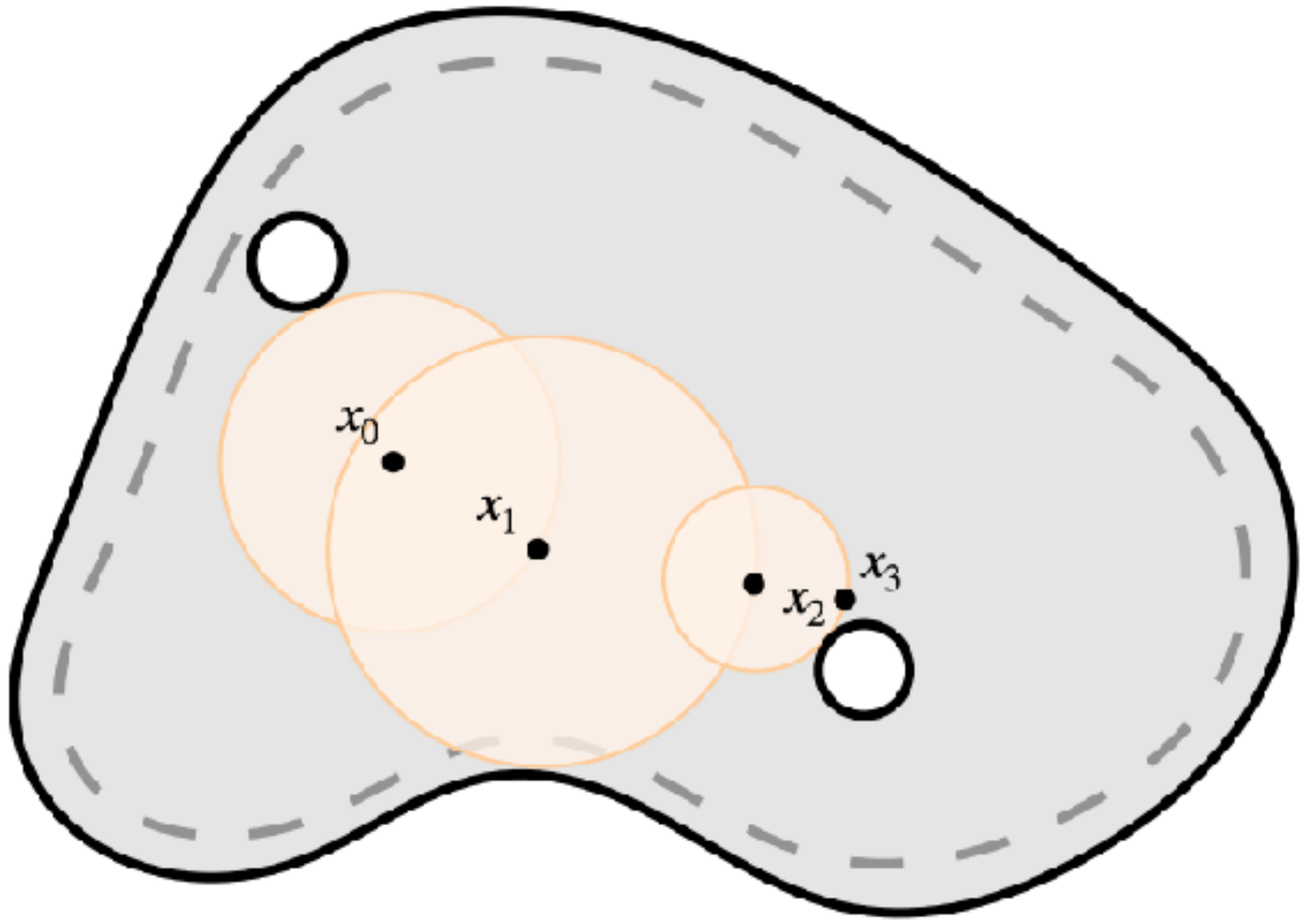
```
x = x0, radius = inf
until radius < ε:
        cp = compute_closest_point(x)
        radius = | cp-x |
        x = sample_point_on_sphere(x, radius)
return g(x)
```



volumetric walk on spheres

ours

```
x = x0, radius = inf, memory=[]
until radius < ε:
        cp = sample_closest_point(x, memory)
        memory.append((x, cp))
        radius = | cp - x |
        x = sample_point_on_sphere(x, radius)
return g(x)
```

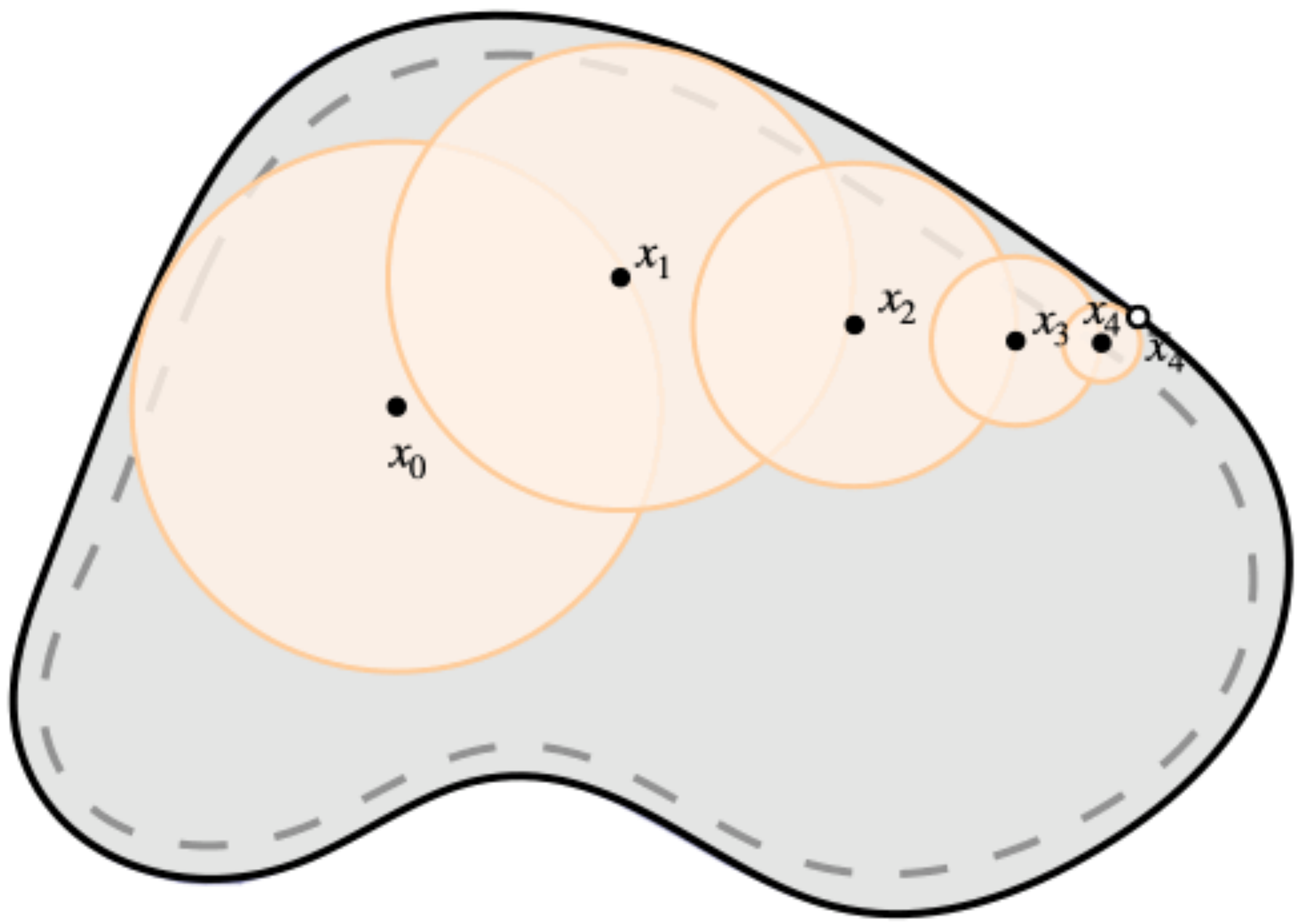


Volumetric walk on spheres

walk on spheres

[Mueller 1956, Sawhney and Crane 2020]

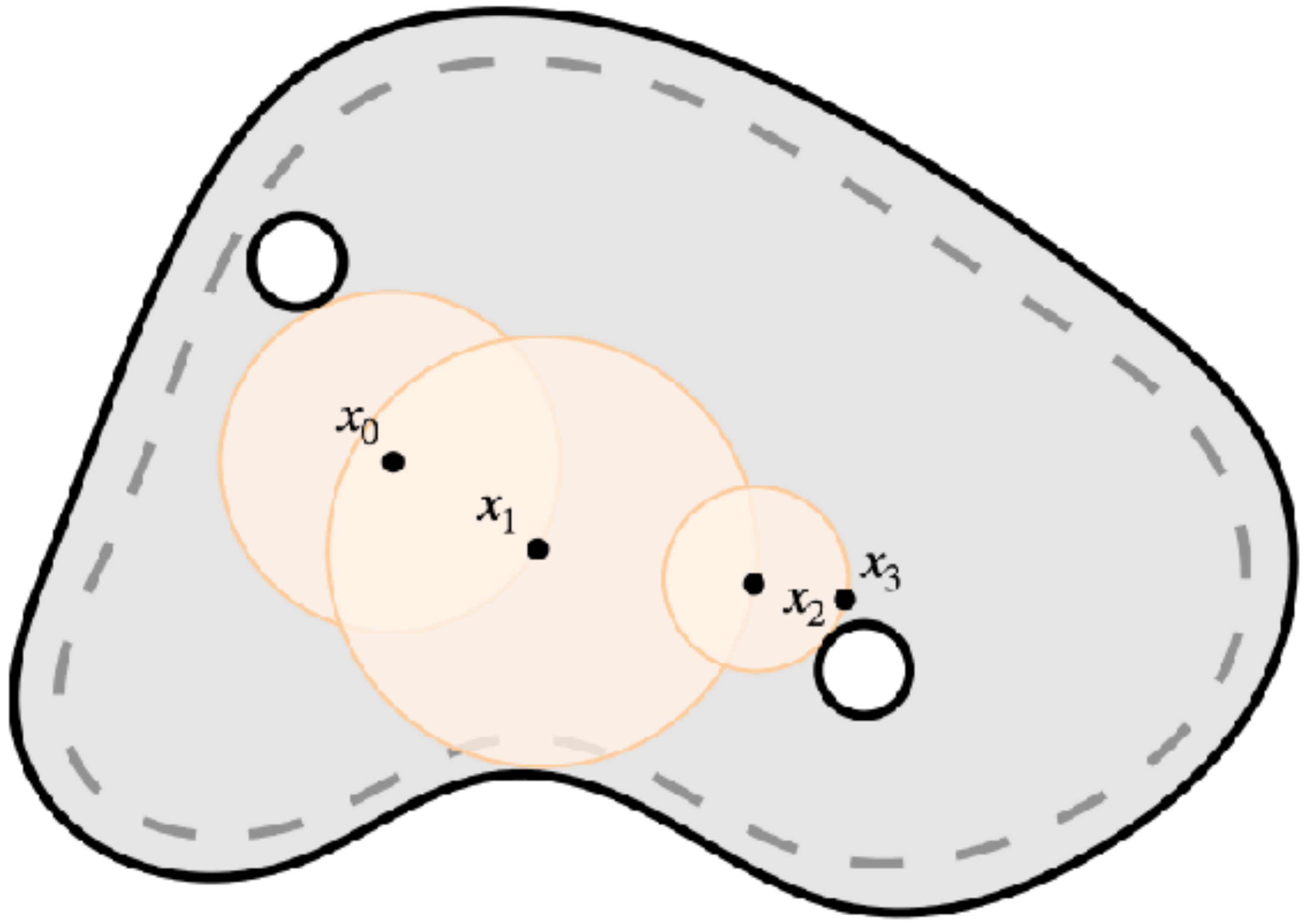
```
x = x0, radius = inf
until radius < ε:
        cp = compute_closest_point(x)
        radius = |cp-x|
        x = sample_point_on_sphere(x, radius)
return g(x)
```



volumetric walk on spheres

ours

```
x = x0, radius = inf, memory=[]
until radius < ε:
        cp = sample_closest_point(x, memory)
        memory.append((x, cp))
        radius = |cp - x|
        x = sample_point_on_sphere(x, radius)
return g(x)
```

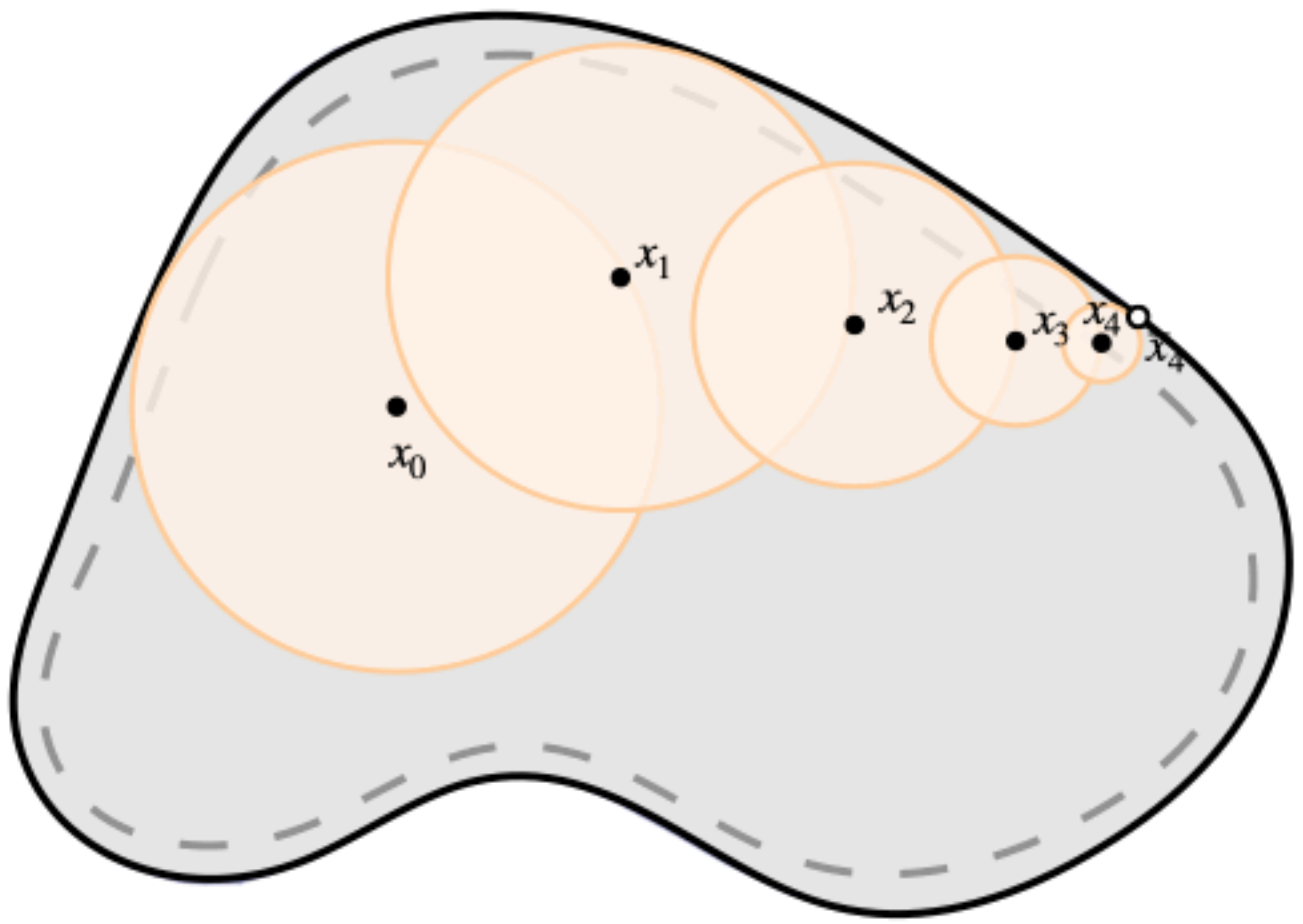


Volumetric walk on spheres

walk on spheres

[Mueller 1956, Sawhney and Crane 2020]

```
x = x0, radius = inf
until radius < ε:
        cp = compute_closest_point(x)
        radius = | cp-x |
        x = sample_point_on_sphere(x, radius)
return g(x)
```

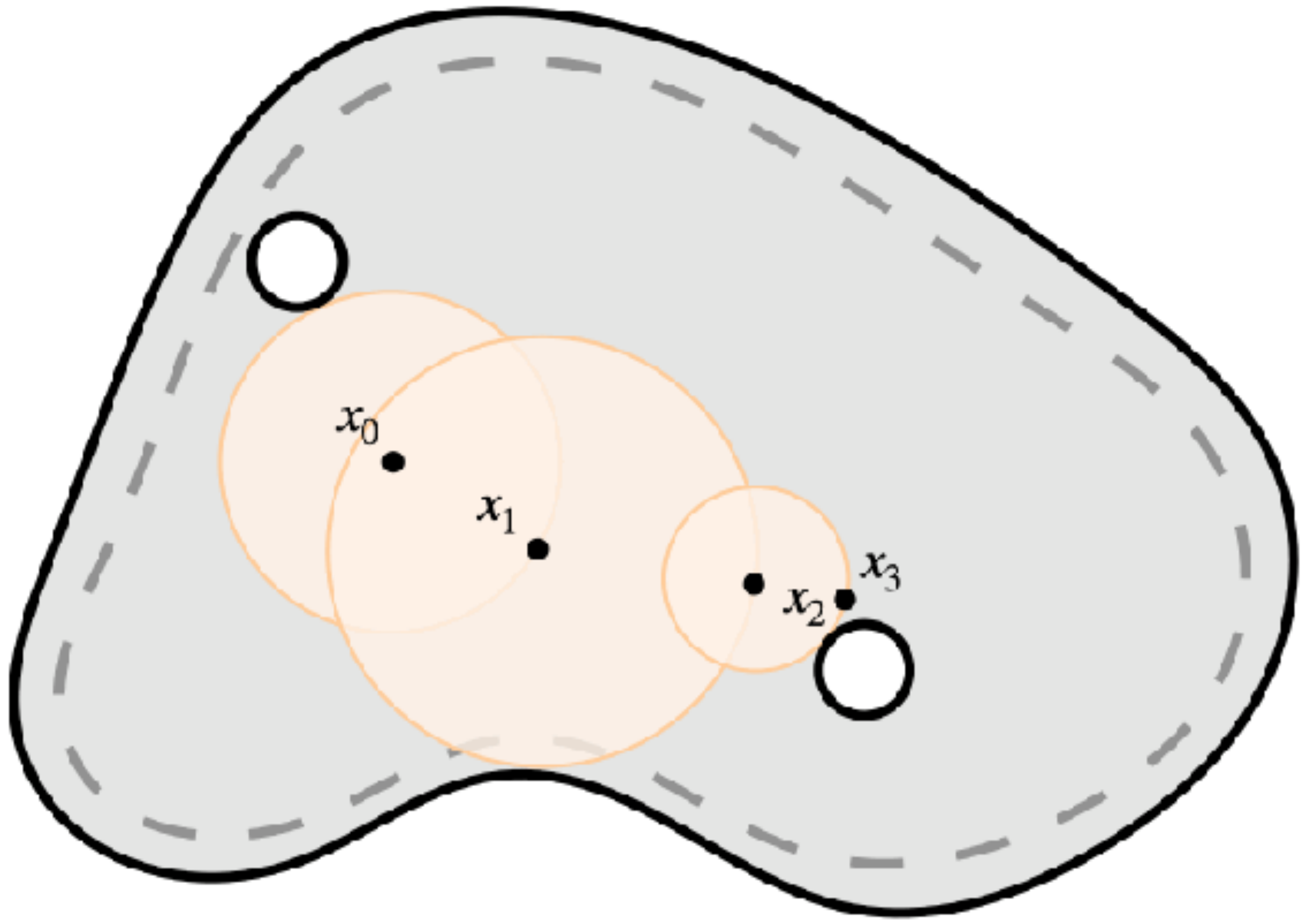


+ stochastic closest point queries →

volumetric walk on spheres

ours

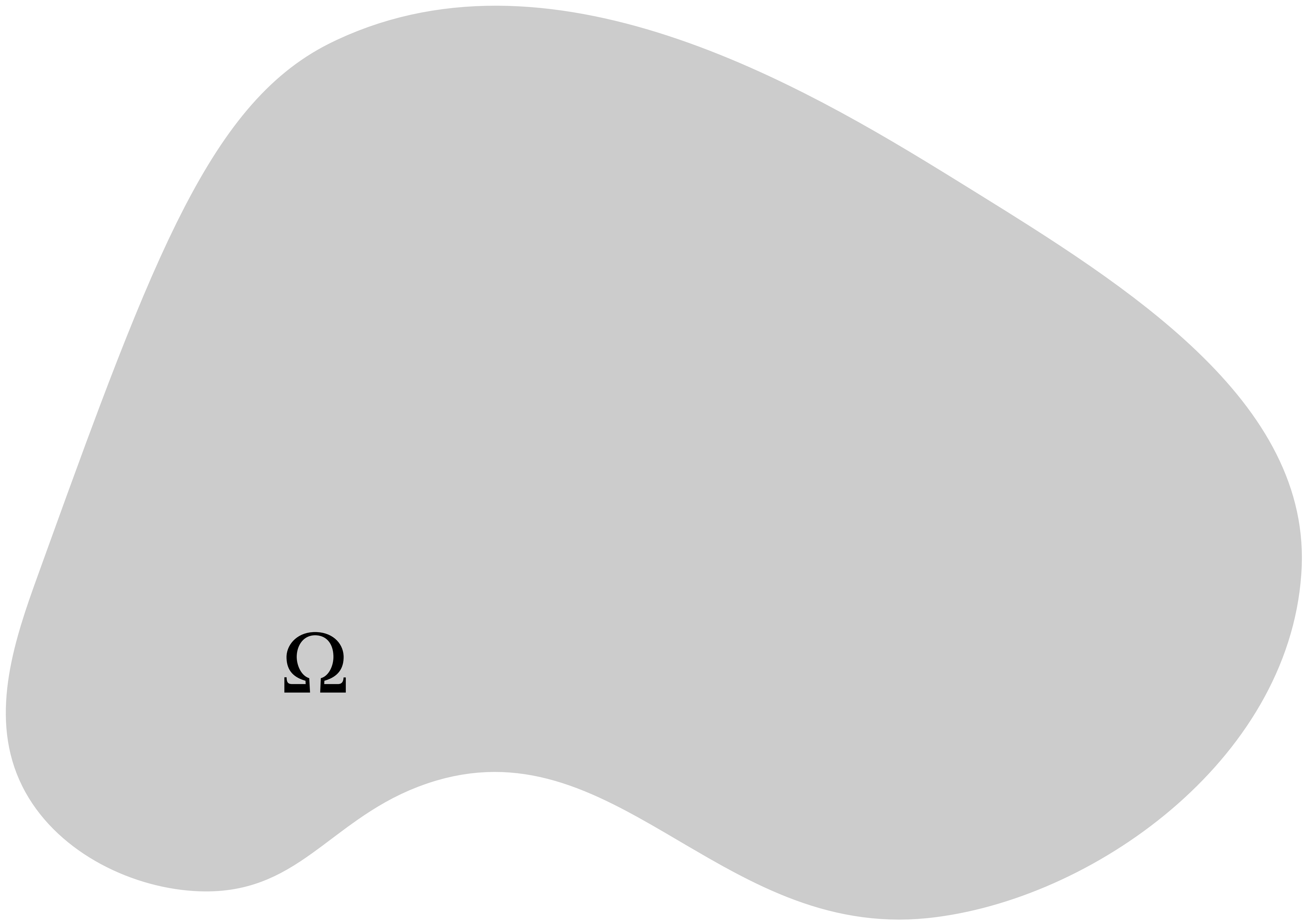
```
x = x0, radius = inf, memory=[]
until radius < ε:
        cp = sample_closest_point(x, memory)
        memory.append((x, cp))
        radius = | cp - x |
        x = sample_point_on_sphere(x, radius)
return g(x)
```



Review of walk on spheres

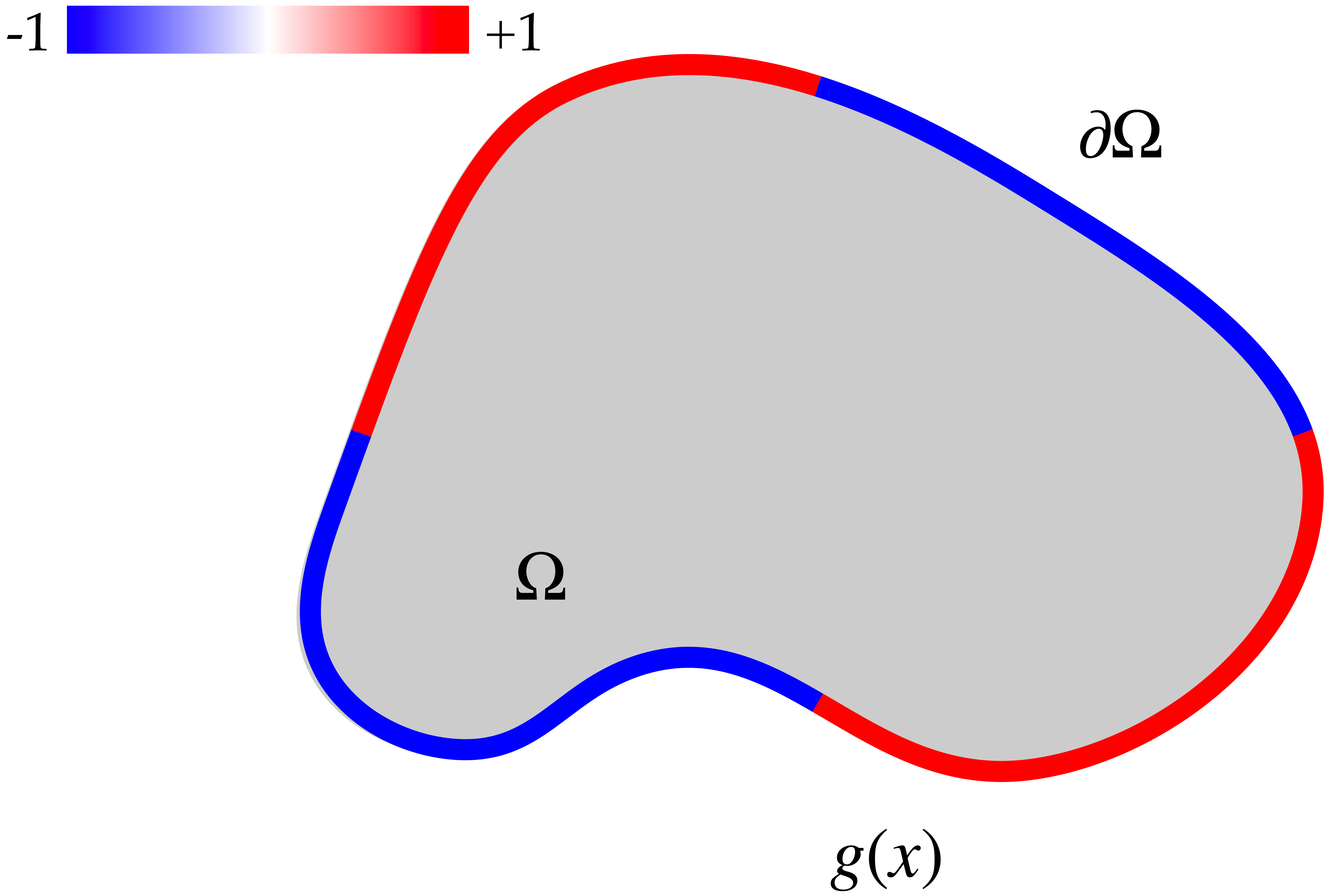
Laplace equation

$$\Delta u = 0 \quad \text{on } \Omega$$
$$u = g \quad \text{on } \partial\Omega$$



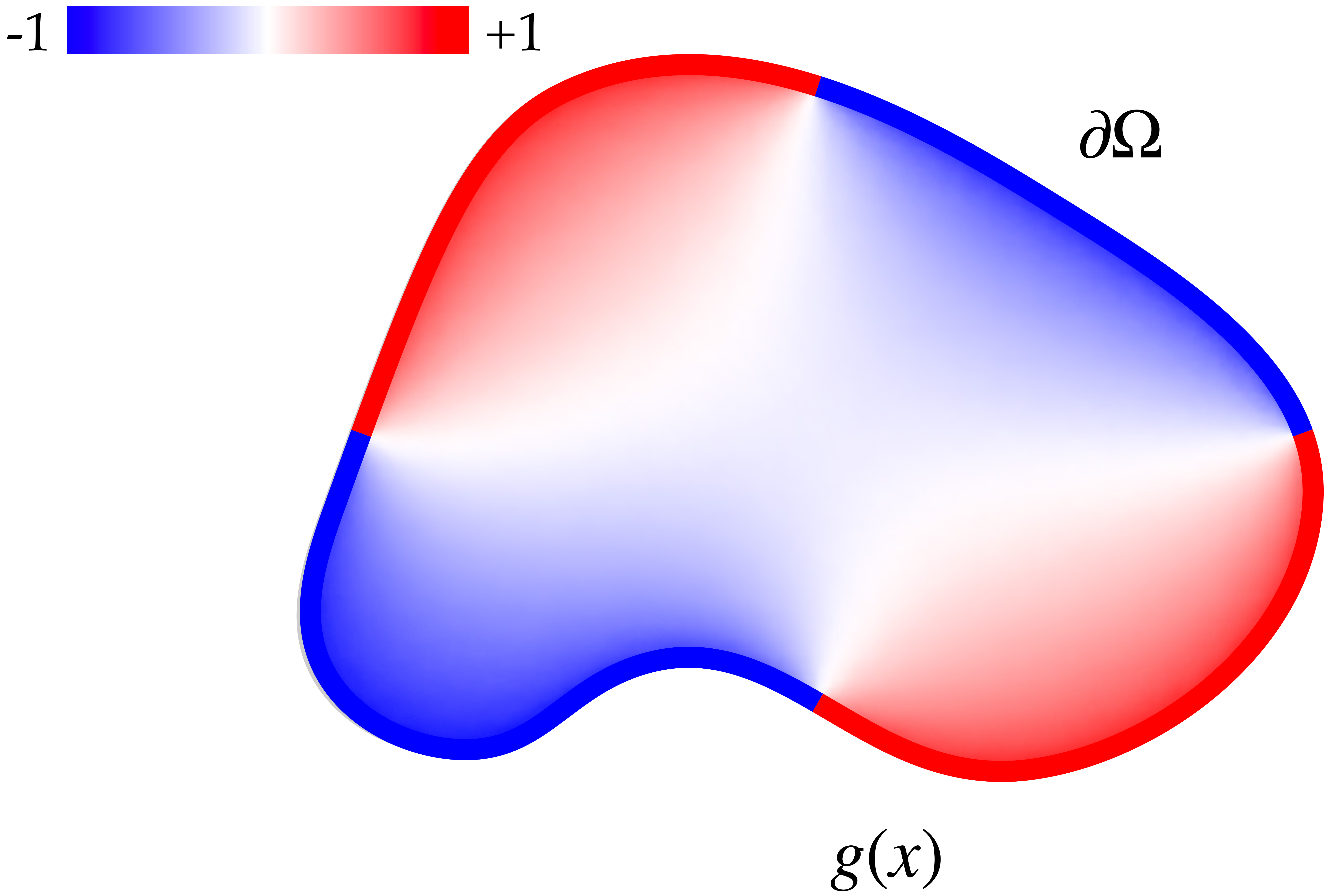
Laplace equation

$$\Delta u = 0 \quad \text{on } \Omega$$
$$u = g \quad \text{on } \partial\Omega$$



Laplace equation

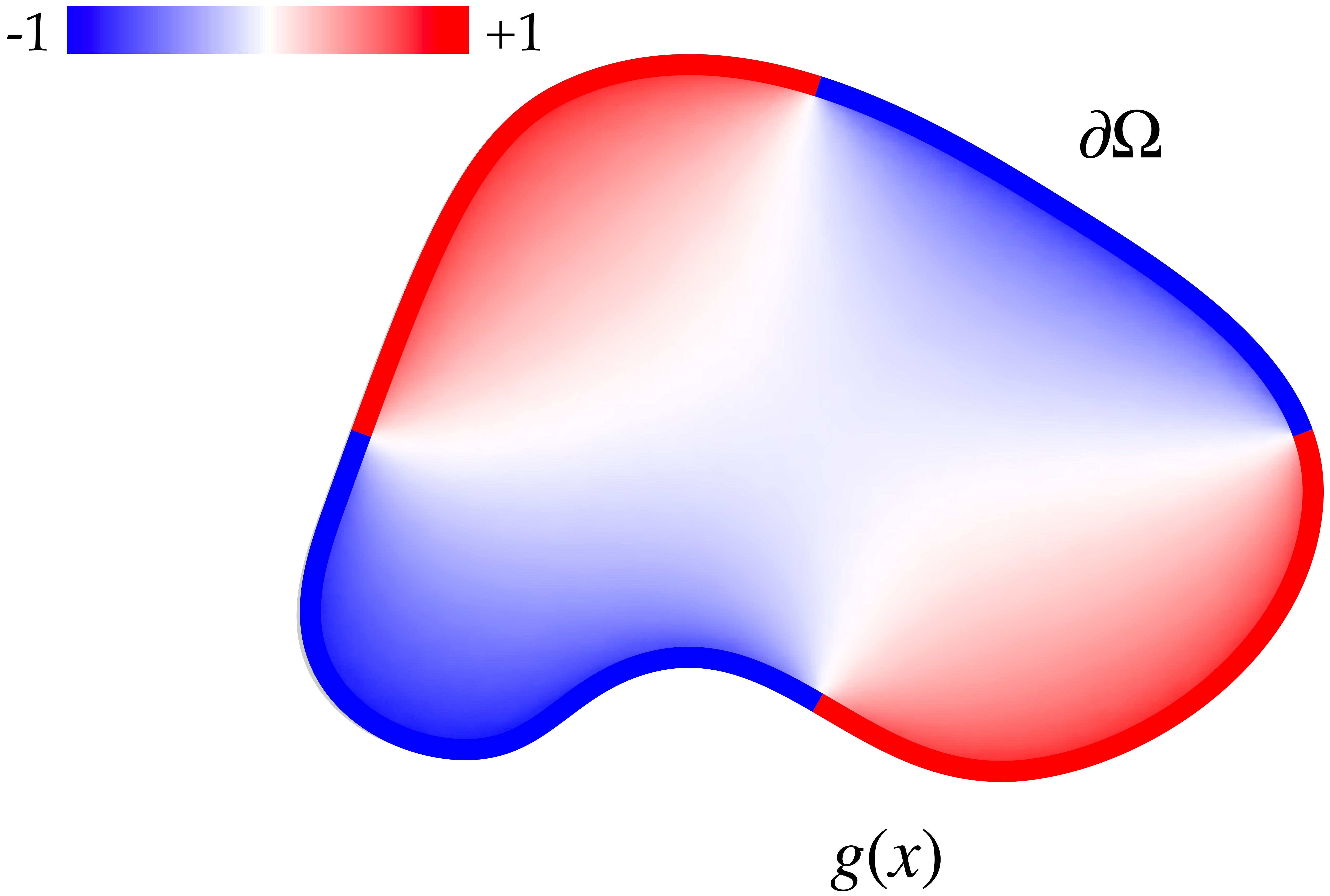
$$\Delta u = 0 \quad \text{on } \Omega$$
$$u = g \quad \text{on } \partial\Omega$$



Laplace equation

$$\Delta u = 0 \quad \text{on } \Omega$$
$$u = g \quad \text{on } \partial\Omega$$

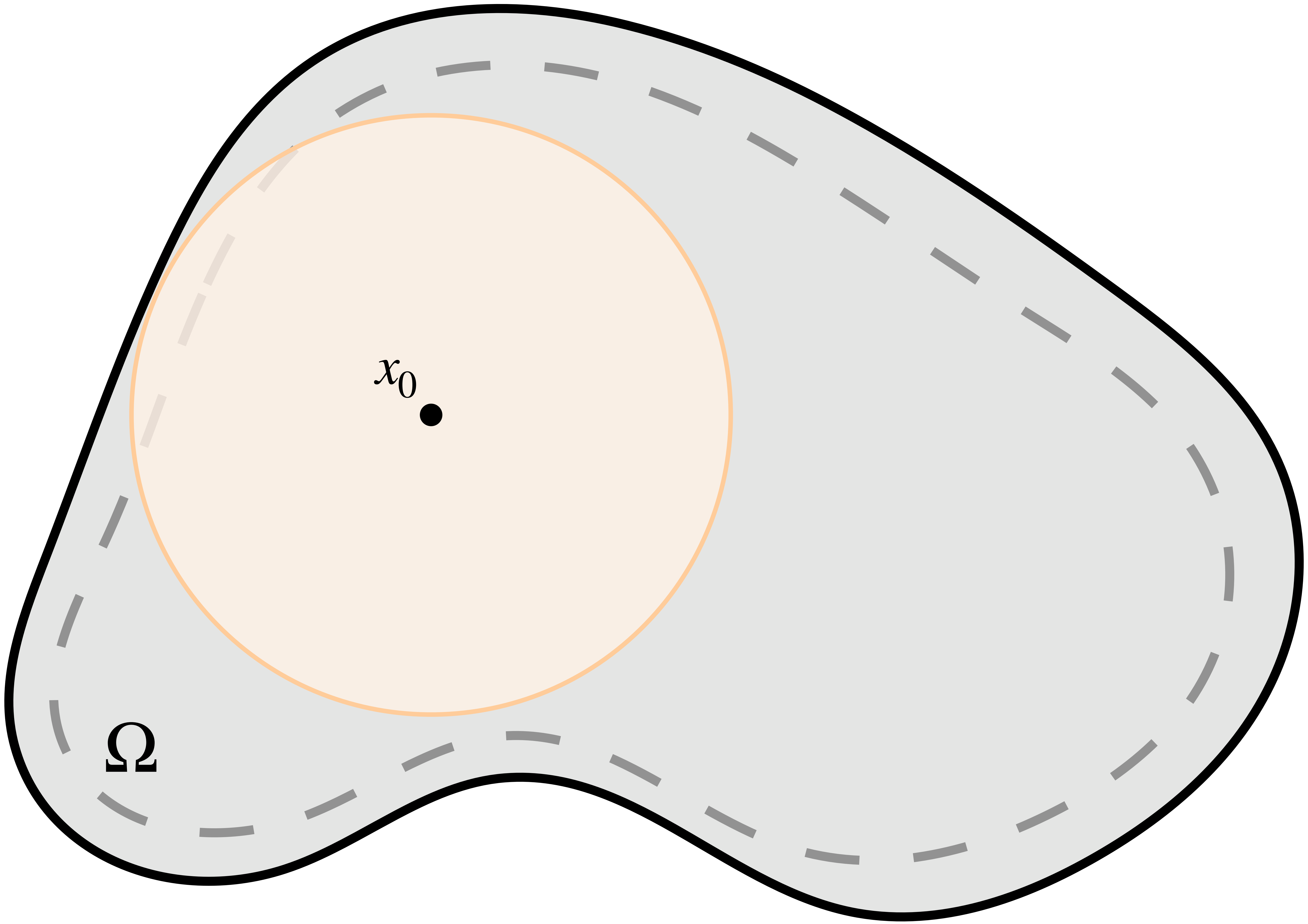
Dirichlet boundary condition



Walk on spheres [Muller 1956, Sawhney and Crane 2020]

$$\langle u(x_i)\rangle = \begin{cases} g(x_{i+1}) & \text{if } x_{i+1} \in \partial\Omega_\varepsilon \\ \langle u(x_i)\rangle & \text{otherwise} \end{cases}$$

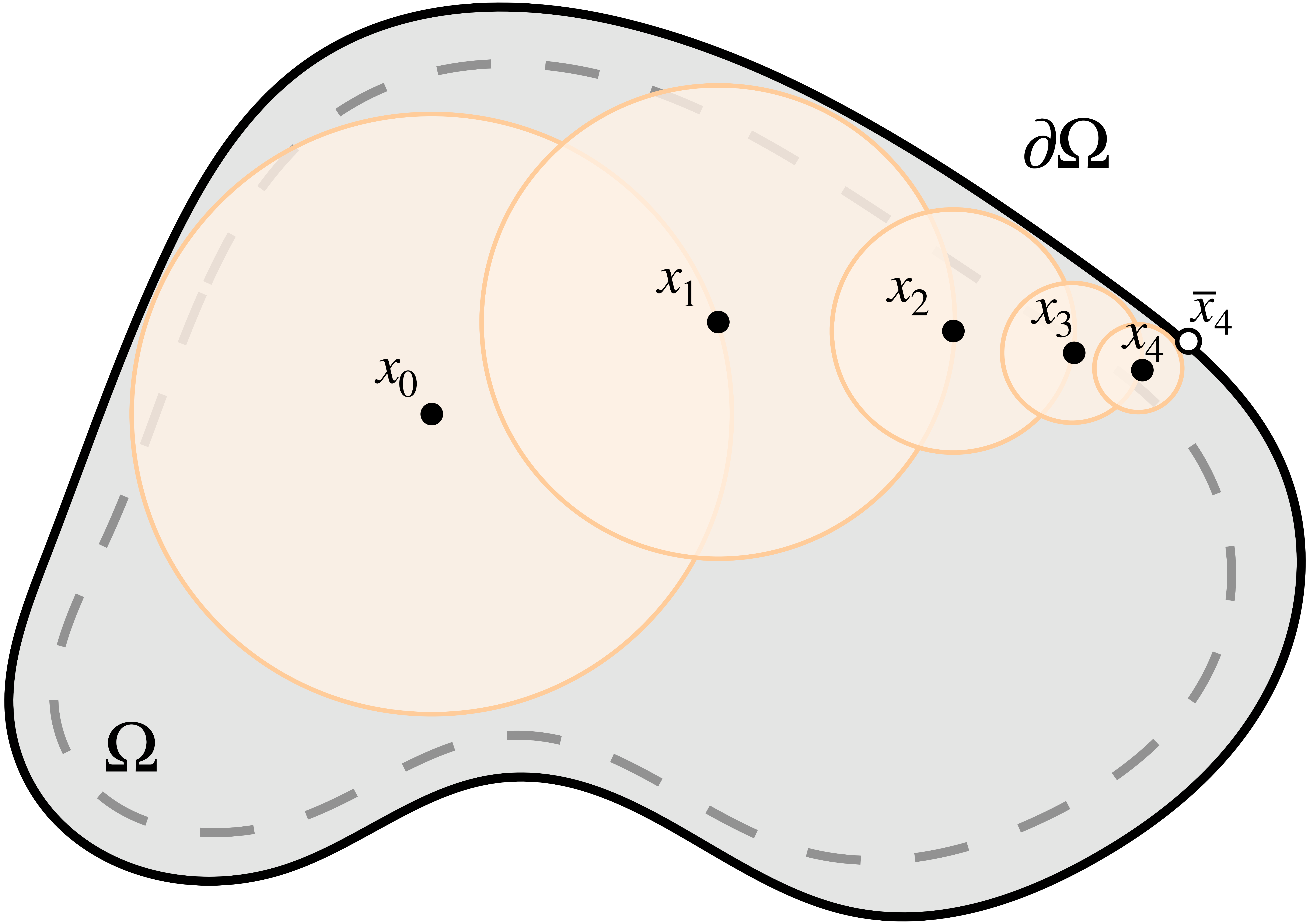
$$x_{i+1} \sim \mathcal{U}\left[\partial B(x_i)\right]$$



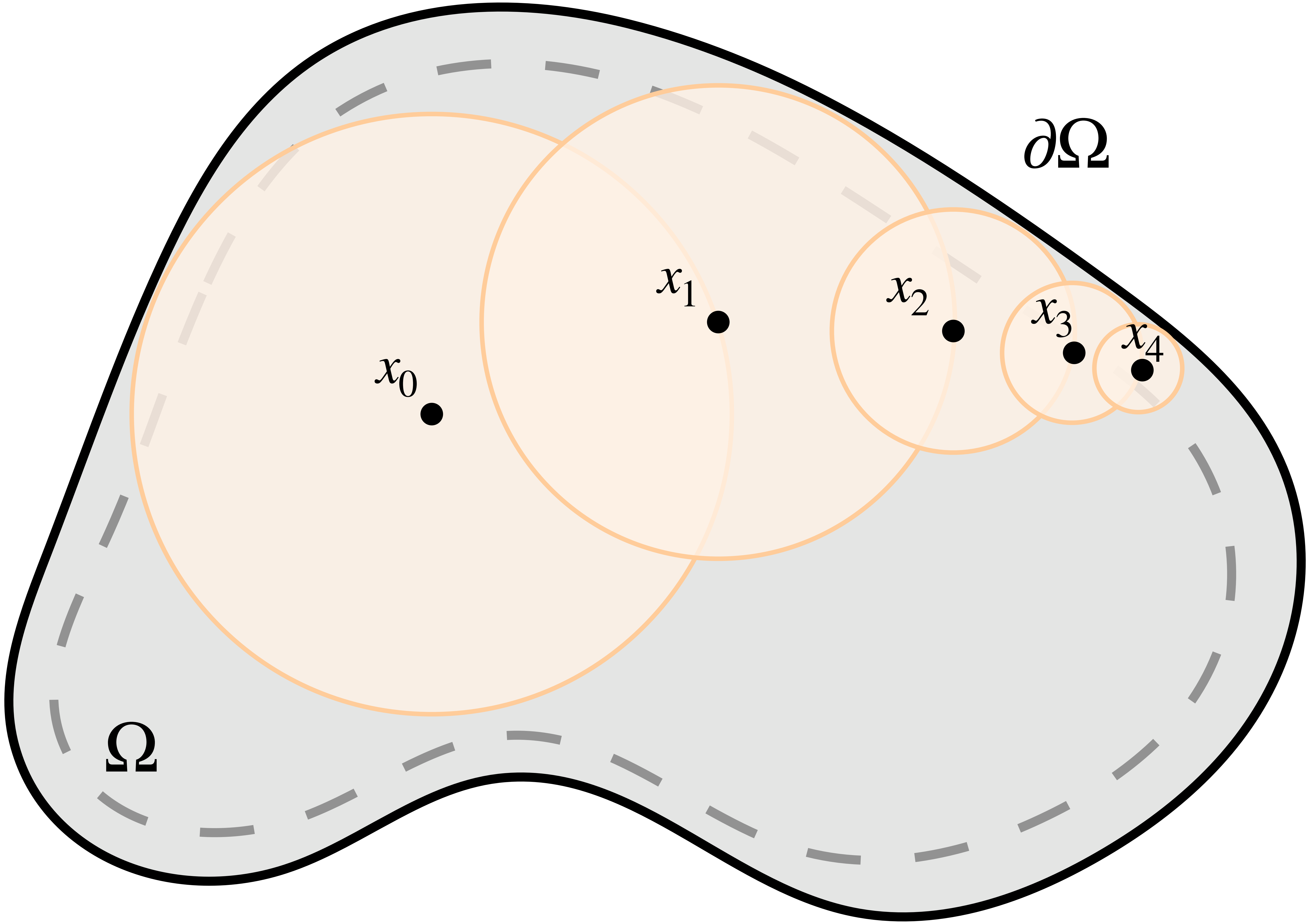
Walk on spheres [Muller 1956, Sawhney and Crane 2020]

$$\langle u(x_i)\rangle = \begin{cases} g(x_{i+1}) & \text{if } x_{i+1} \in \partial\Omega_\varepsilon \\ \langle u(x_i)\rangle & \text{otherwise} \end{cases}$$

$$x_{i+1} \sim \mathcal{U}\left[\partial B(x_i)\right]$$

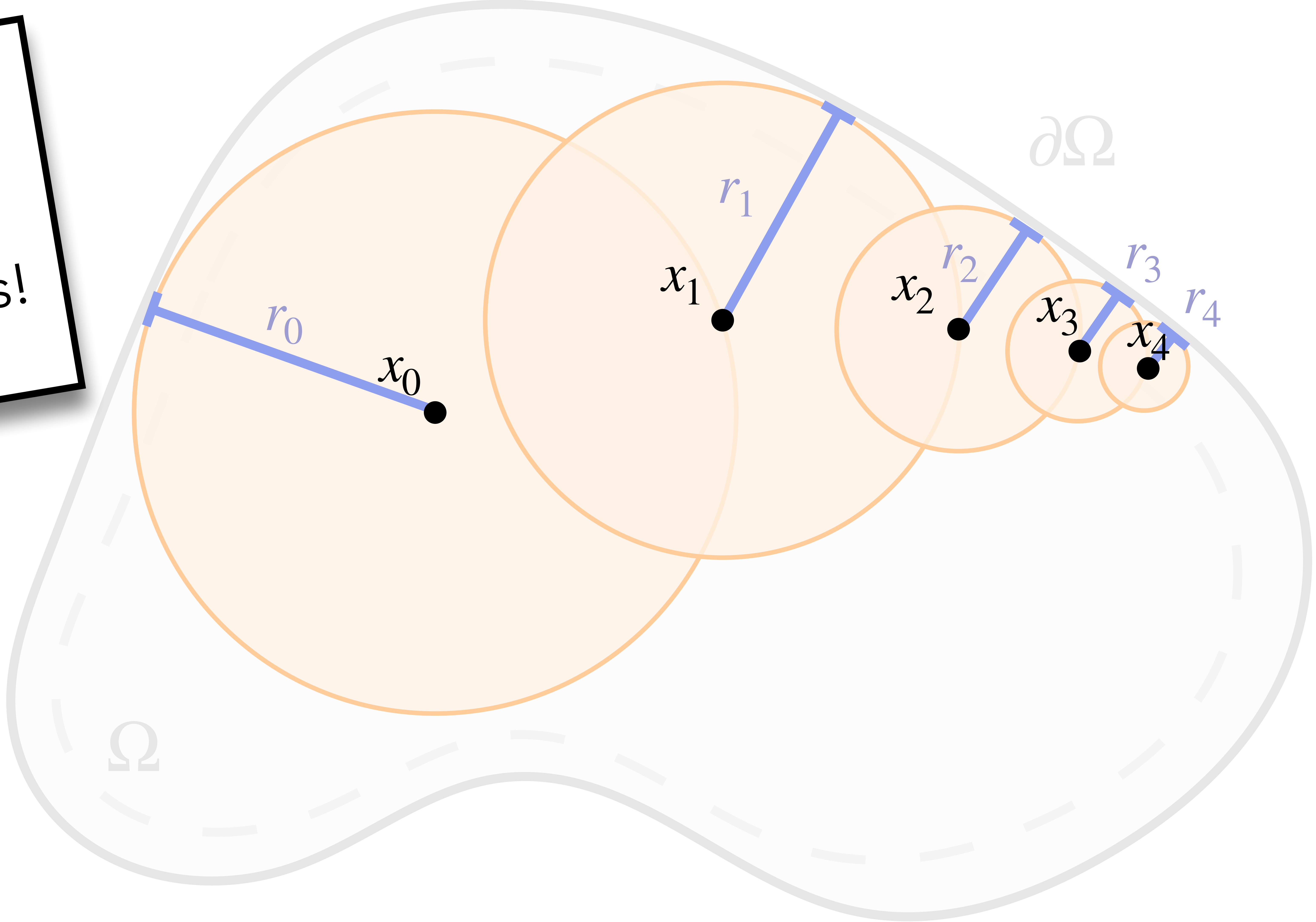


Closest point queries



Closest point queries

Walk on spheres only interacts with geometry through closest point queries!



Closest point queries are fast and general

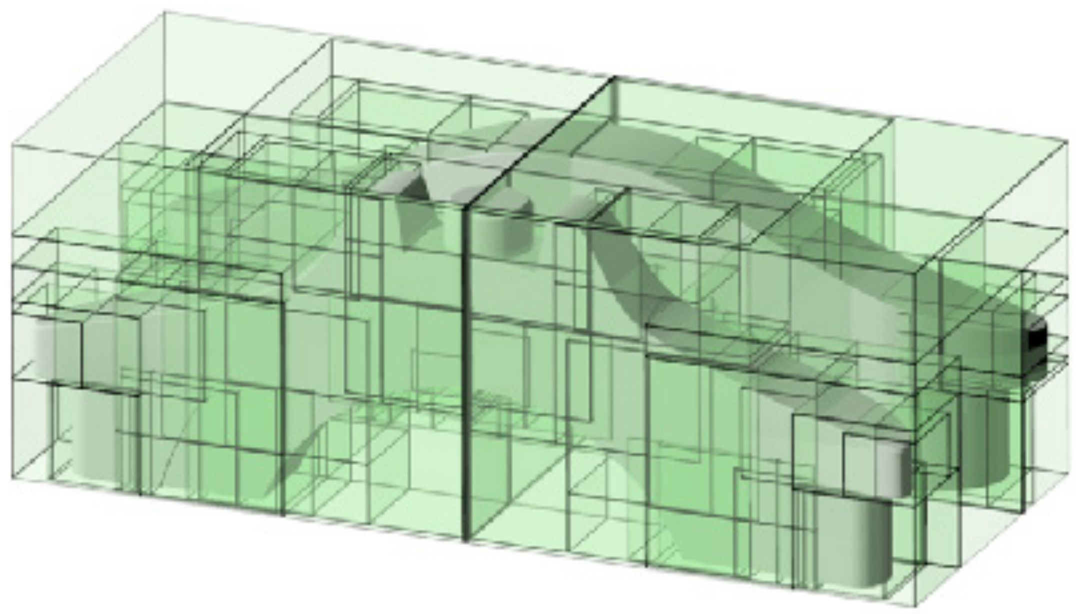
Closest point queries are fast and general

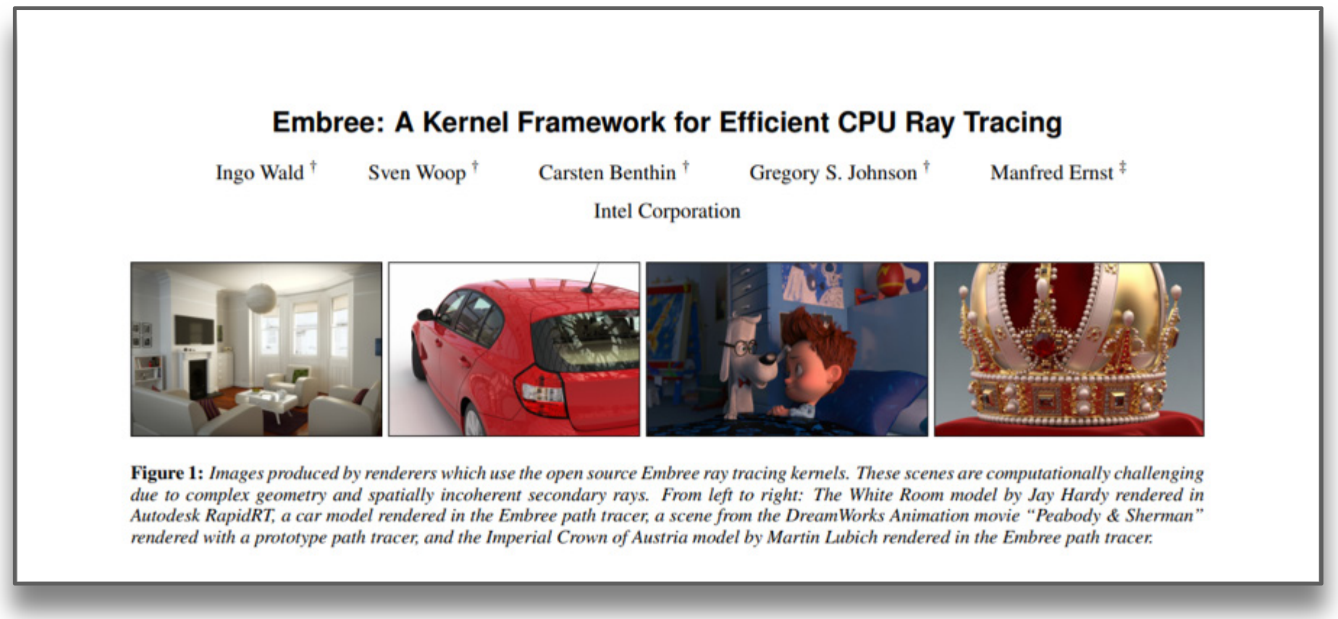
queries are fast

logarithmic-complexity queries with BVH

input
boundary mesh

build BVH for WoS
few milliseconds





Embree: A Kernel Framework for Efficient CPU Ray Tracing

Ingo Wald † Sven Woop † Carsten Benthin † Gregory S. Johnson † Manfred Ernst ‡

Intel Corporation

Figure 1: *Images produced by renderers which use the open source Embree ray tracing kernels. These scenes are computationally challenging due to complex geometry and spatially incoherent secondary rays. From left to right: The White Room model by Jay Hardy rendered in Autodesk RapidRT, a car model rendered in the Embree path tracer, a scene from the DreamWorks Animation movie "Peabody & Sherman" rendered with a prototype path tracer, and the Imperial Crown of Austria model by Martin Lubich rendered in the Embree path tracer.*

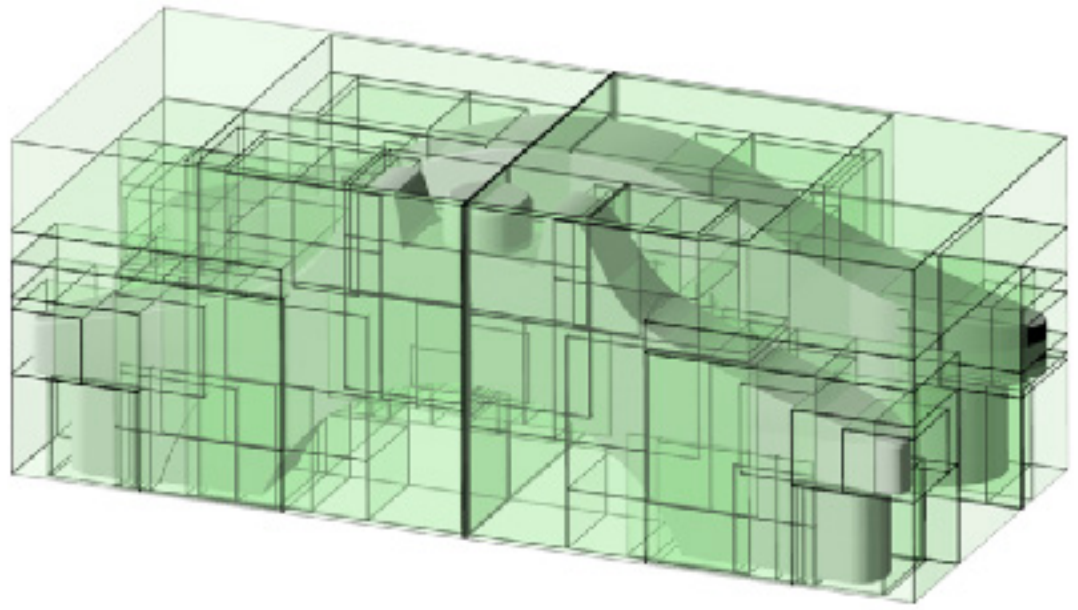
Closest point queries are fast and general

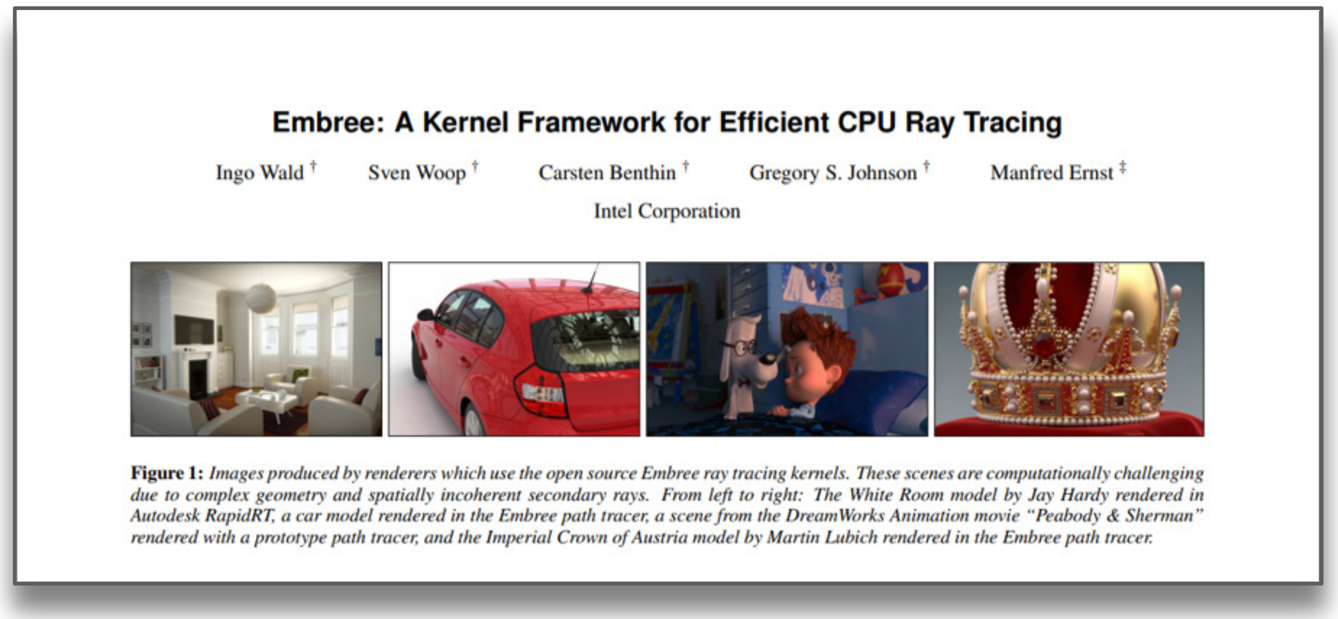
queries are fast

logarithmic-complexity queries with BVH

input
boundary mesh

build BVH for WoS
few milliseconds





Embree: A Kernel Framework for Efficient CPU Ray Tracing

Ingo Wald † Sven Woop † Carsten Benthin † Gregory S. Johnson † Manfred Ernst ‡

Intel Corporation

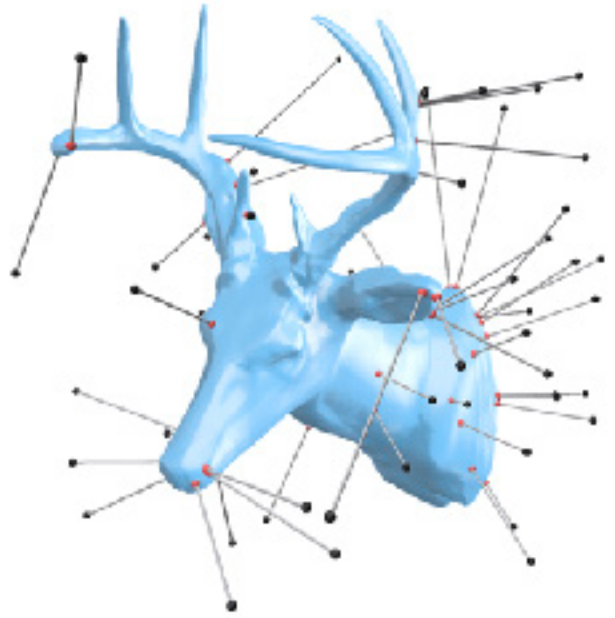
Figure 1: *Images produced by renderers which use the open source Embree ray tracing kernels. These scenes are computationally challenging due to complex geometry and spatially incoherent secondary rays. From left to right: The White Room model by Jay Hardy rendered in Autodesk RapidRT, a car model rendered in the Embree path tracer, a scene from the DreamWorks Animation movie "Peabody & Sherman" rendered with a prototype path tracer, and the Imperial Crown of Austria model by Martin Lubich rendered in the Embree path tracer.*

queries are general

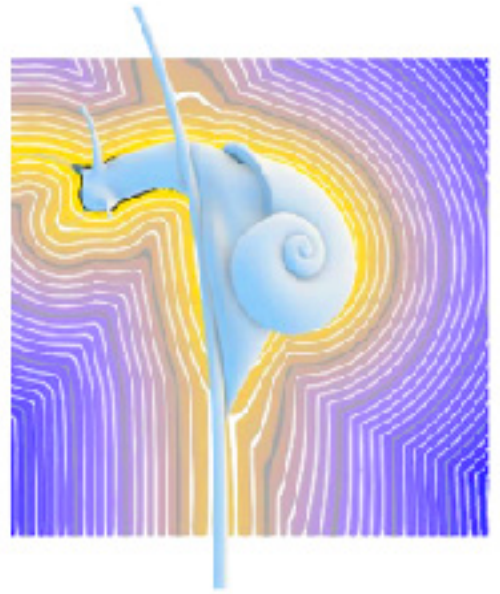
available for many representations

mesh

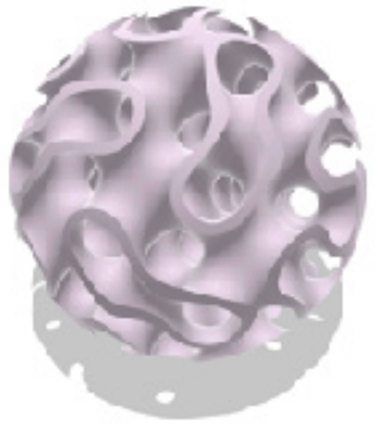
neural
implicit



signed
distance



harmonic
implicit



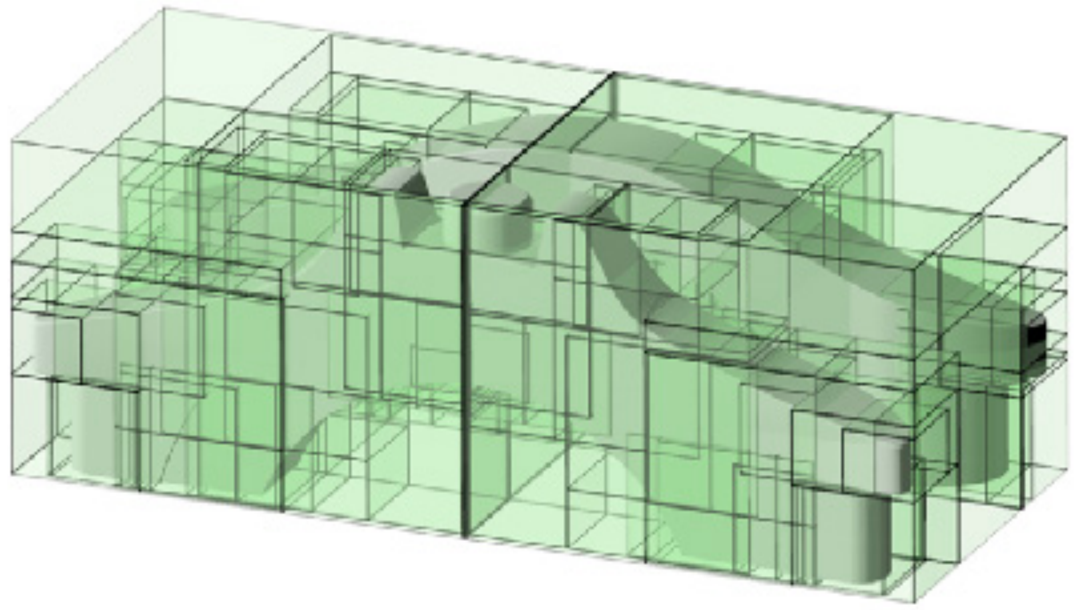
Closest point queries are fast and general

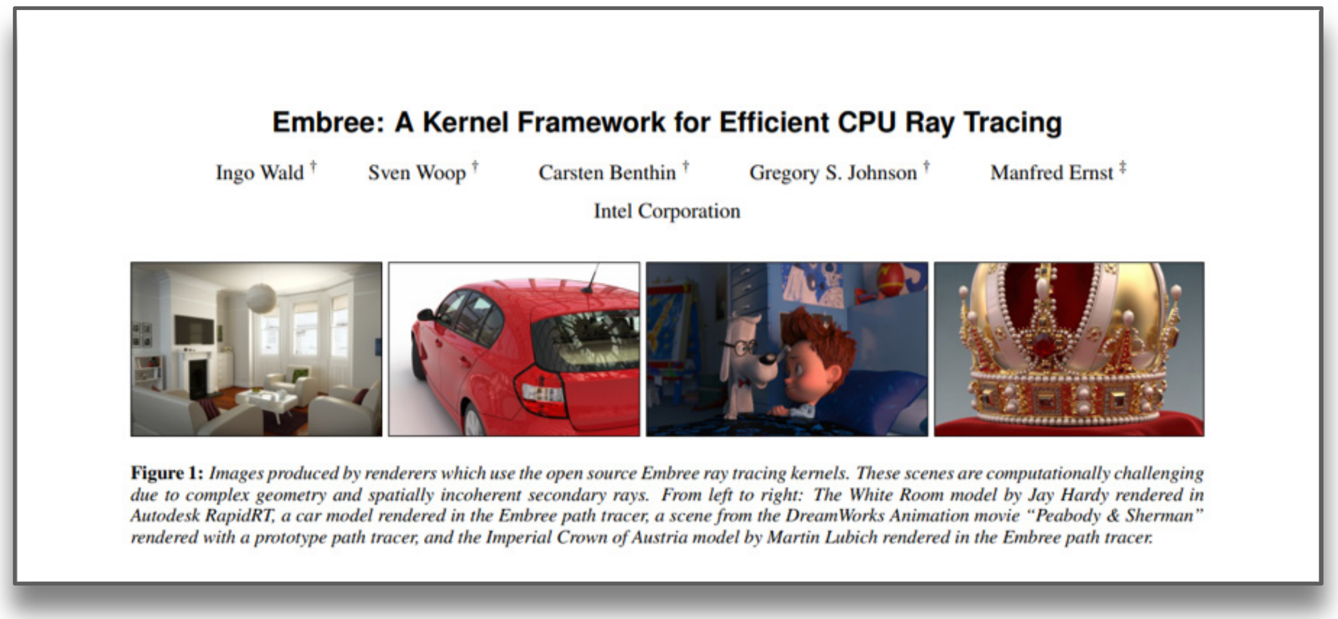
queries are fast

logarithmic-complexity queries with BVH

input
boundary mesh

build BVH for WoS
few milliseconds





Embree: A Kernel Framework for Efficient CPU Ray Tracing

Ingo Wald † Sven Woop † Carsten Benthin † Gregory S. Johnson † Manfred Ernst ‡

Intel Corporation

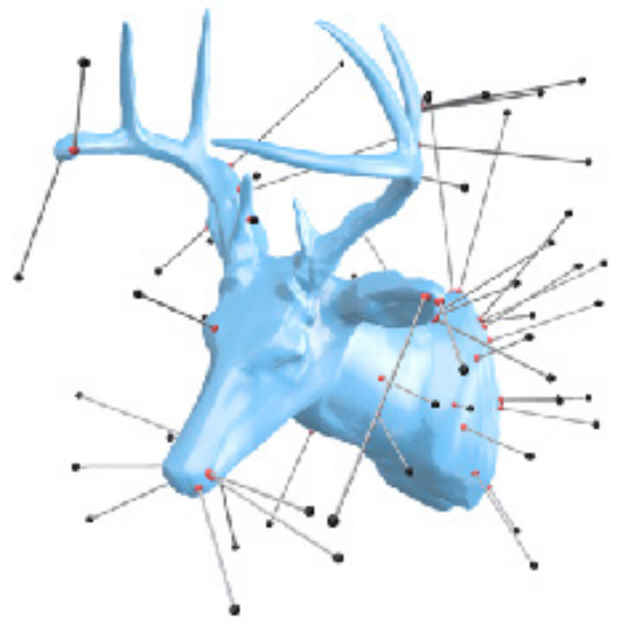
Figure 1: *Images produced by renderers which use the open source Embree ray tracing kernels. These scenes are computationally challenging due to complex geometry and spatially incoherent secondary rays. From left to right: The White Room model by Jay Hardy rendered in Autodesk RapidRT, a car model rendered in the Embree path tracer, a scene from the DreamWorks Animation movie "Peabody & Sherman" rendered with a prototype path tracer, and the Imperial Crown of Austria model by Martin Lubich rendered in the Embree path tracer.*

queries are general

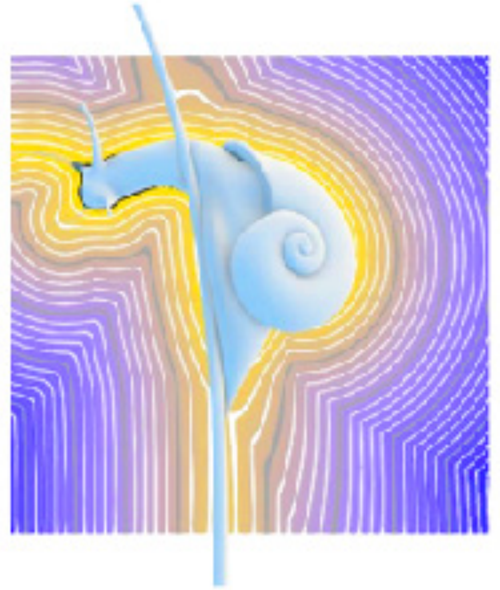
available for many representations

mesh

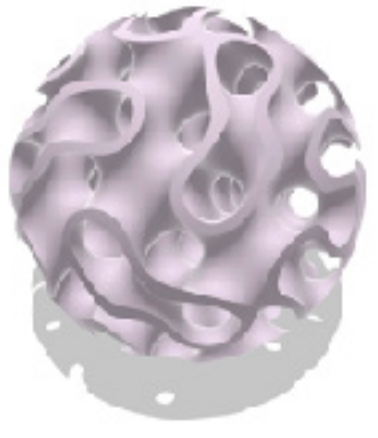
neural implicit

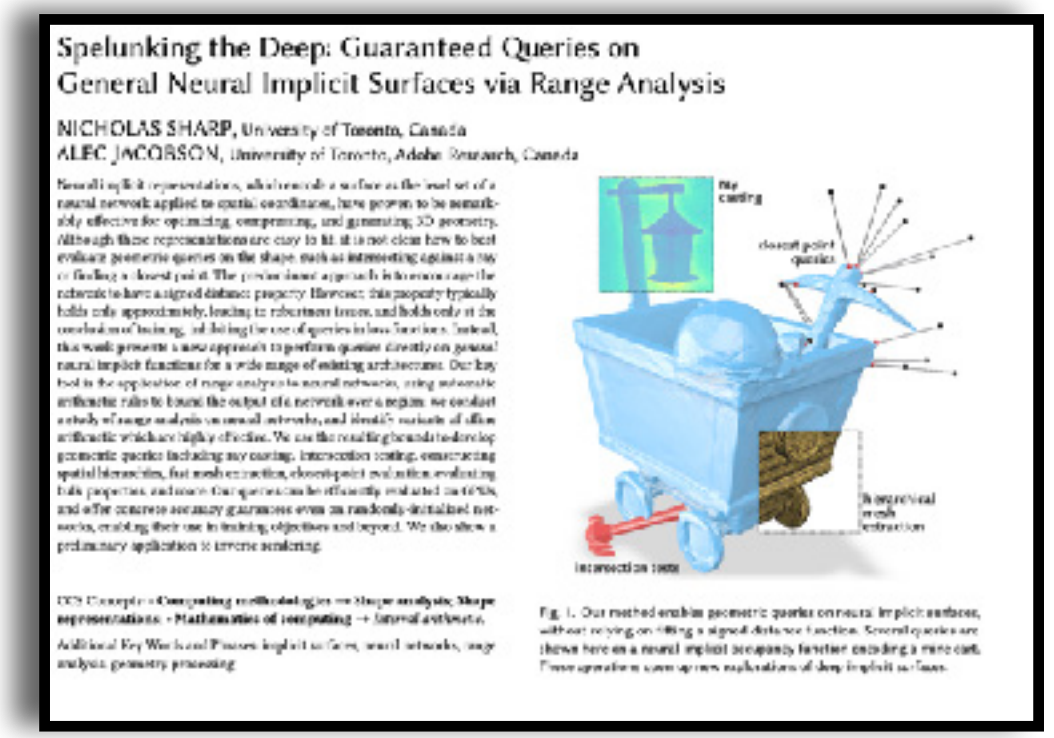


signed distance



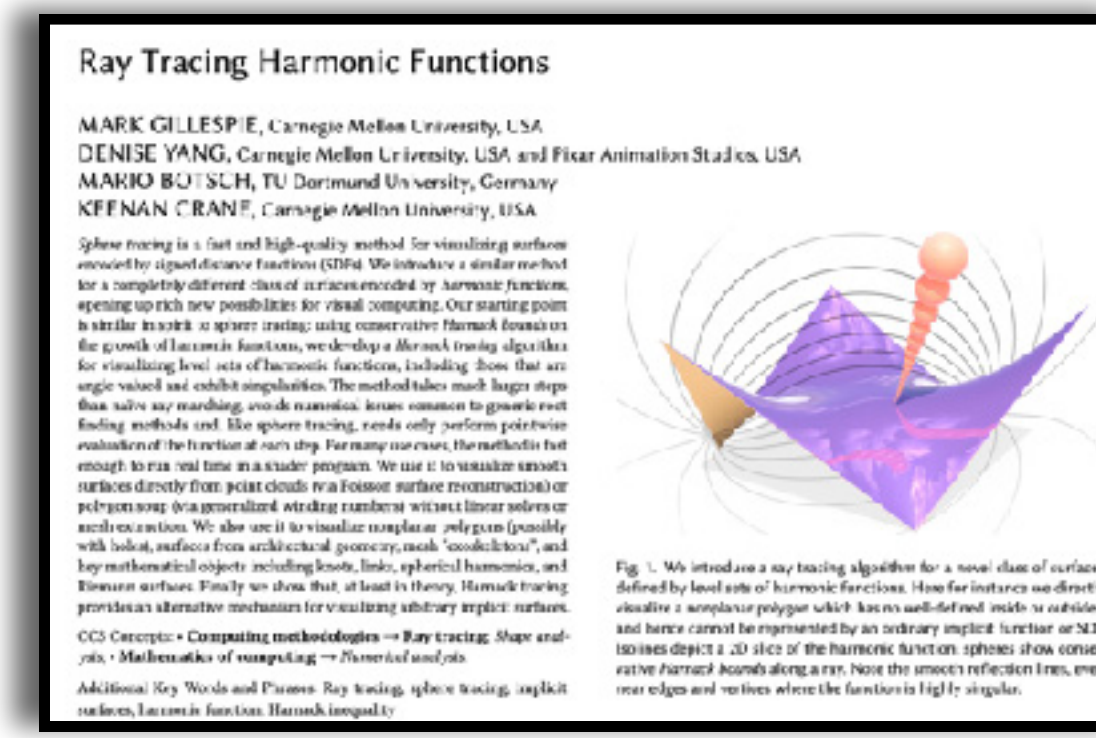
harmonic implicit





Spelunking the Deep: Guaranteed Queries on General Neural Implicit Surfaces via Range Analysis

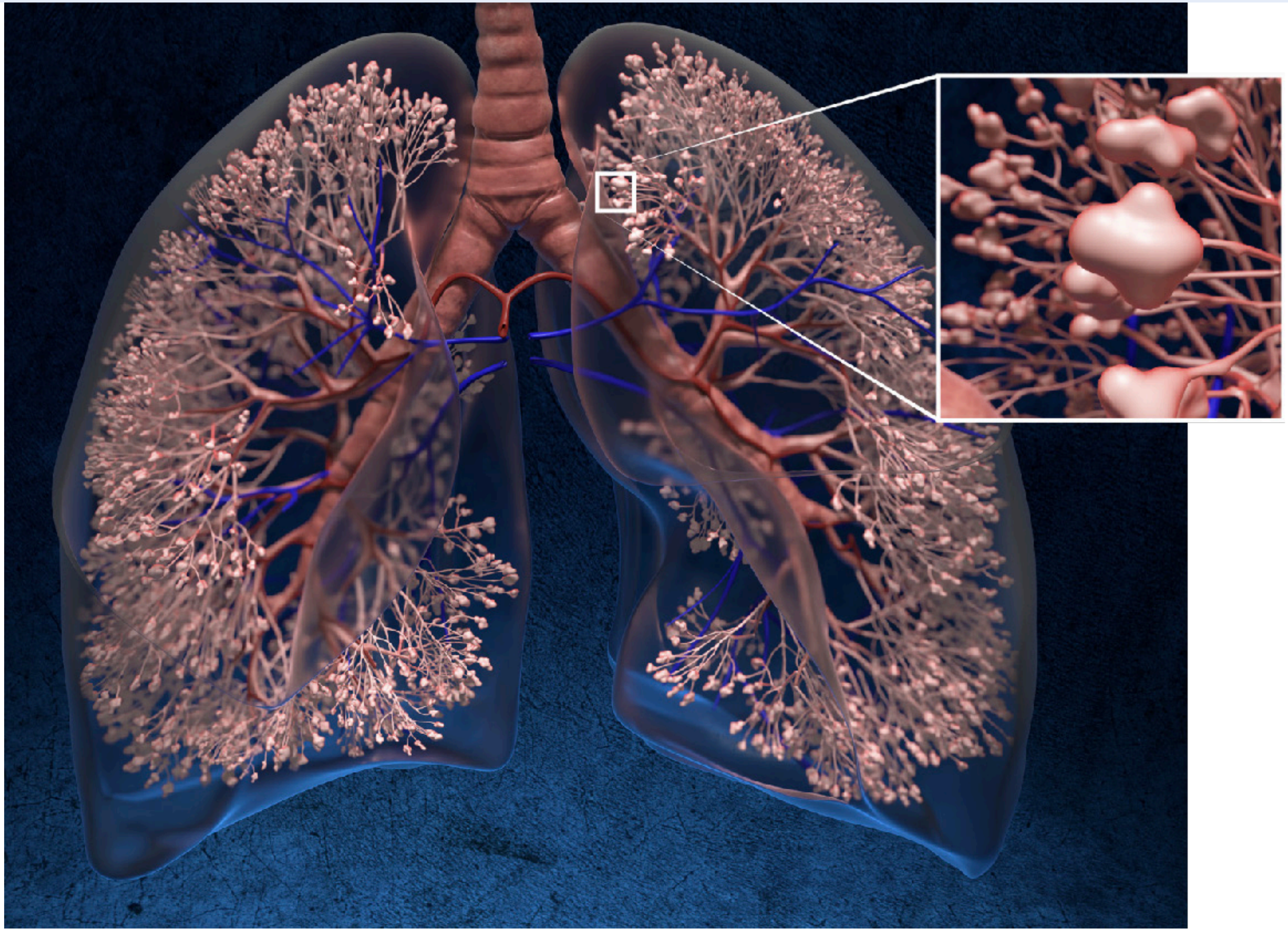
NICHOLAS SHARP, University of Toronto, Canada
ALEC JACOBSON, University of Toronto, Adobe Research, Canada

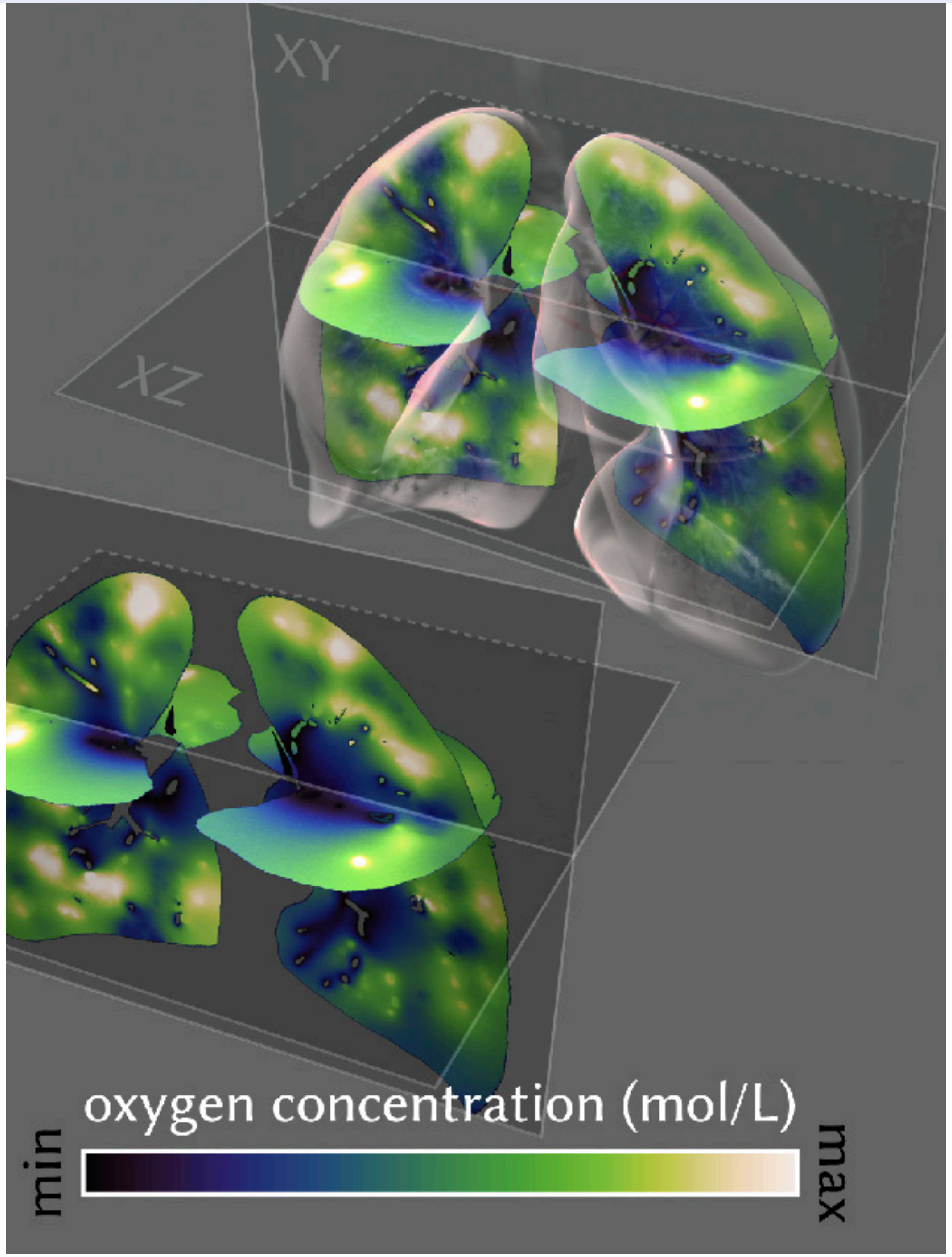


Ray Tracing Harmonic Functions

MARK GILLESPIE, Carnegie Mellon University, USA
DENISE YANG, Carnegie Mellon University, USA and Pixar Animation Studios, USA
MARIO BOTSCH, TU Dortmund University, Germany
KEENAN CRANE, Carnegie Mellon University, USA

Oxygen diffusion on complex model





[Sawhney and Miller et al. 2023]

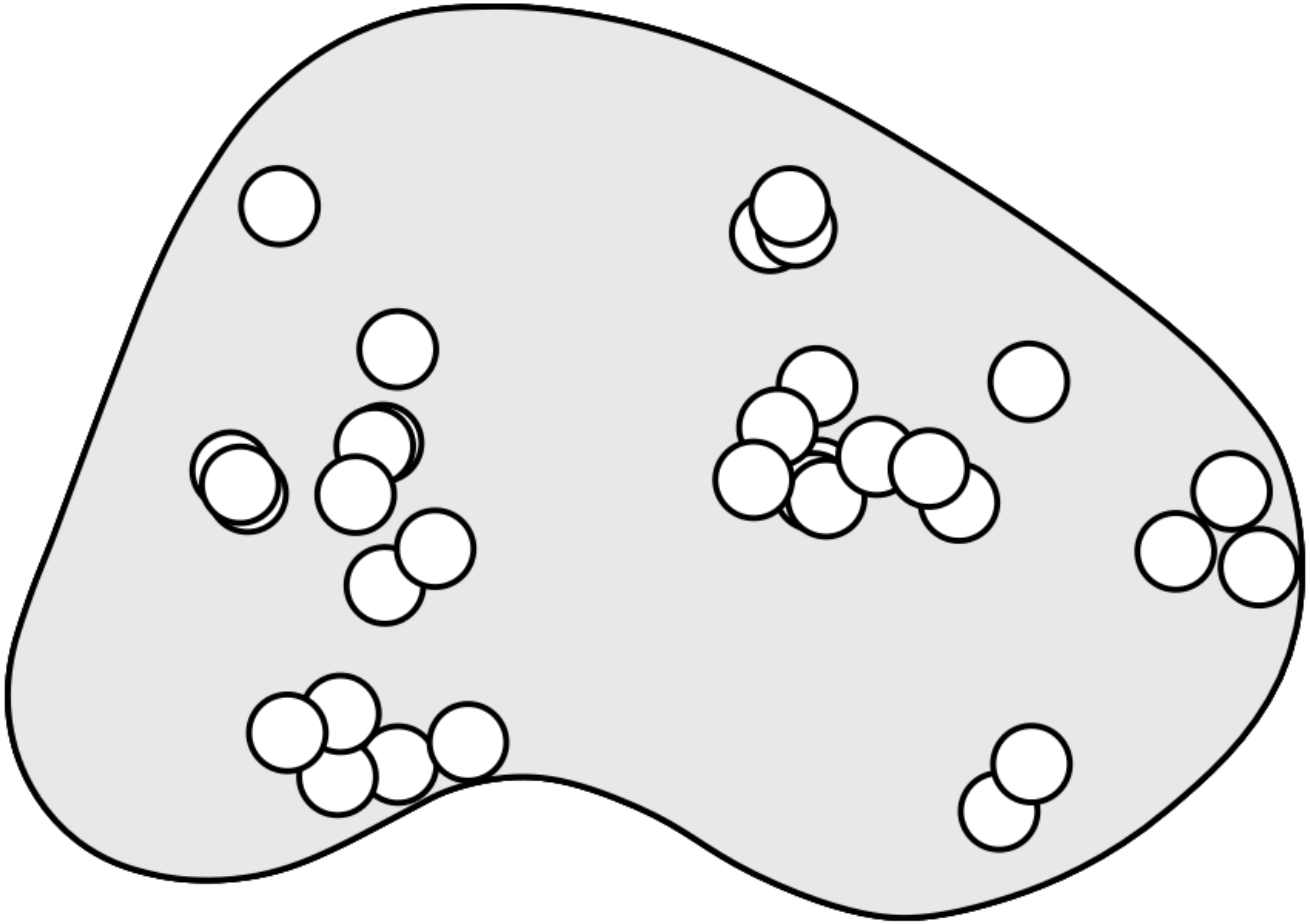
Volumetric walk on spheres

Laplace PDE in participating media

$$\Delta u_O = 0 \quad \text{on } \Omega$$
$$u_O = g \quad \text{on } \partial\Omega$$

where

$\Omega = V \backslash O$
$O = \{B(c_1), B(c_2), \ldots, B(c_N)\}$

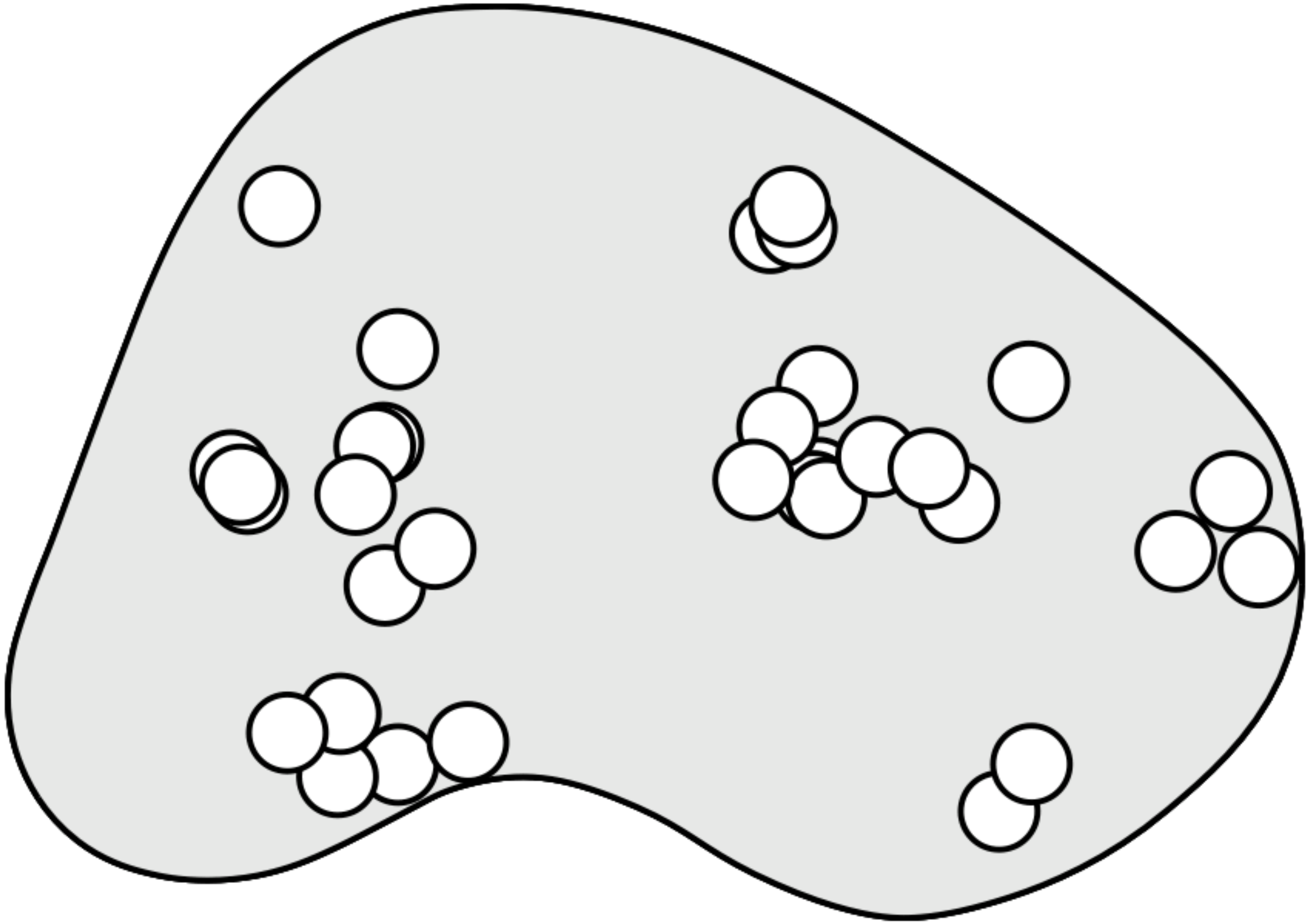


Laplace PDE in participating media

$$\Delta u_O = 0 \quad \text{on } \Omega$$
$$u_O = g \quad \text{on } \partial\Omega$$

where

$$\Omega = V \backslash O$$
$$O = \{B(c_1), B(c_2), \ldots, B(c_N)\}$$

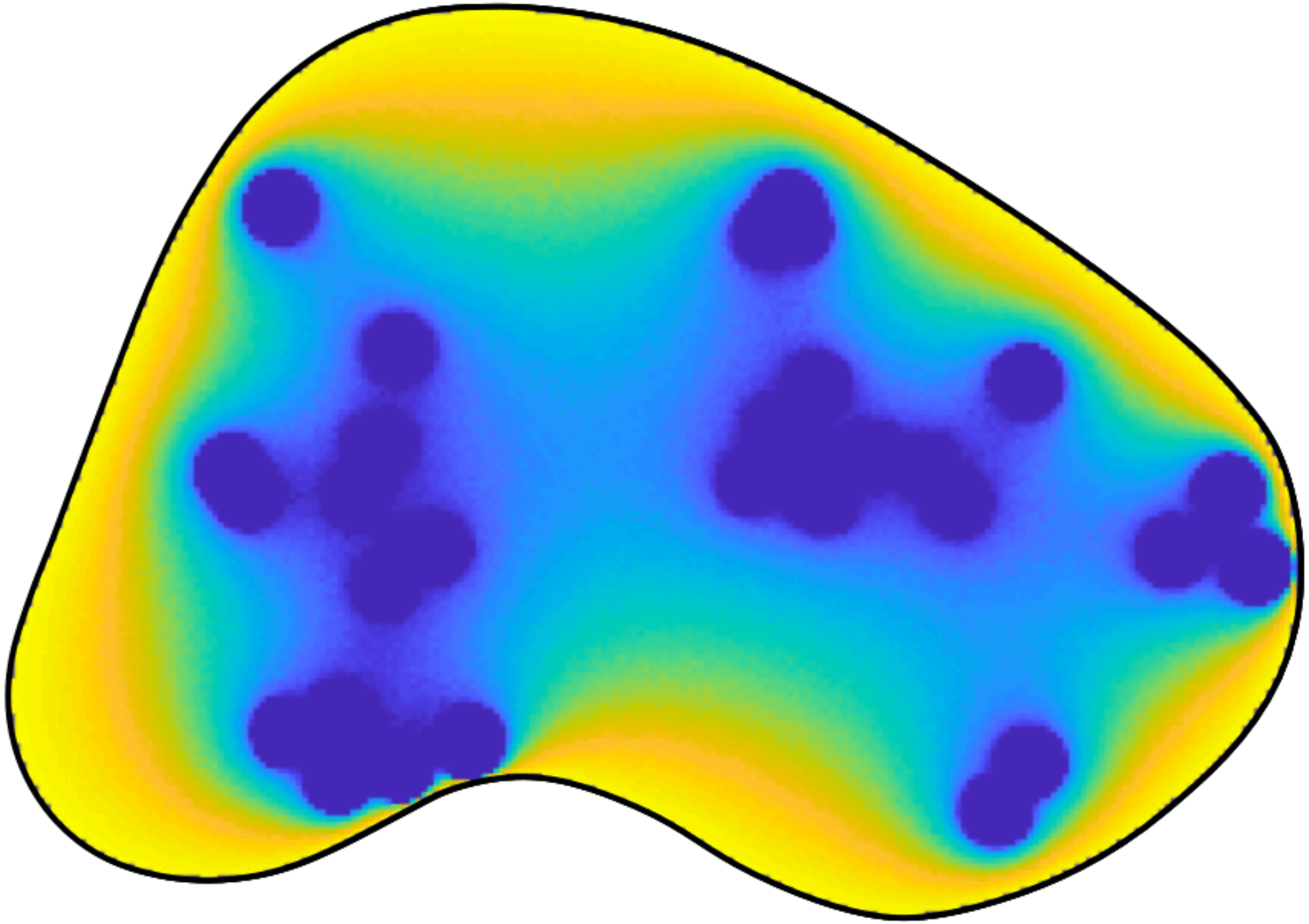


Laplace PDE in participating media

$$\Delta u_O = 0 \quad \text{on } \Omega$$
$$u_O = g \quad \text{on } \partial\Omega$$

where

$$\Omega = V \backslash O$$
$$O = \{B(c_1), B(c_2), \ldots, B(c_N)\}$$



Laplace PDE in participating media

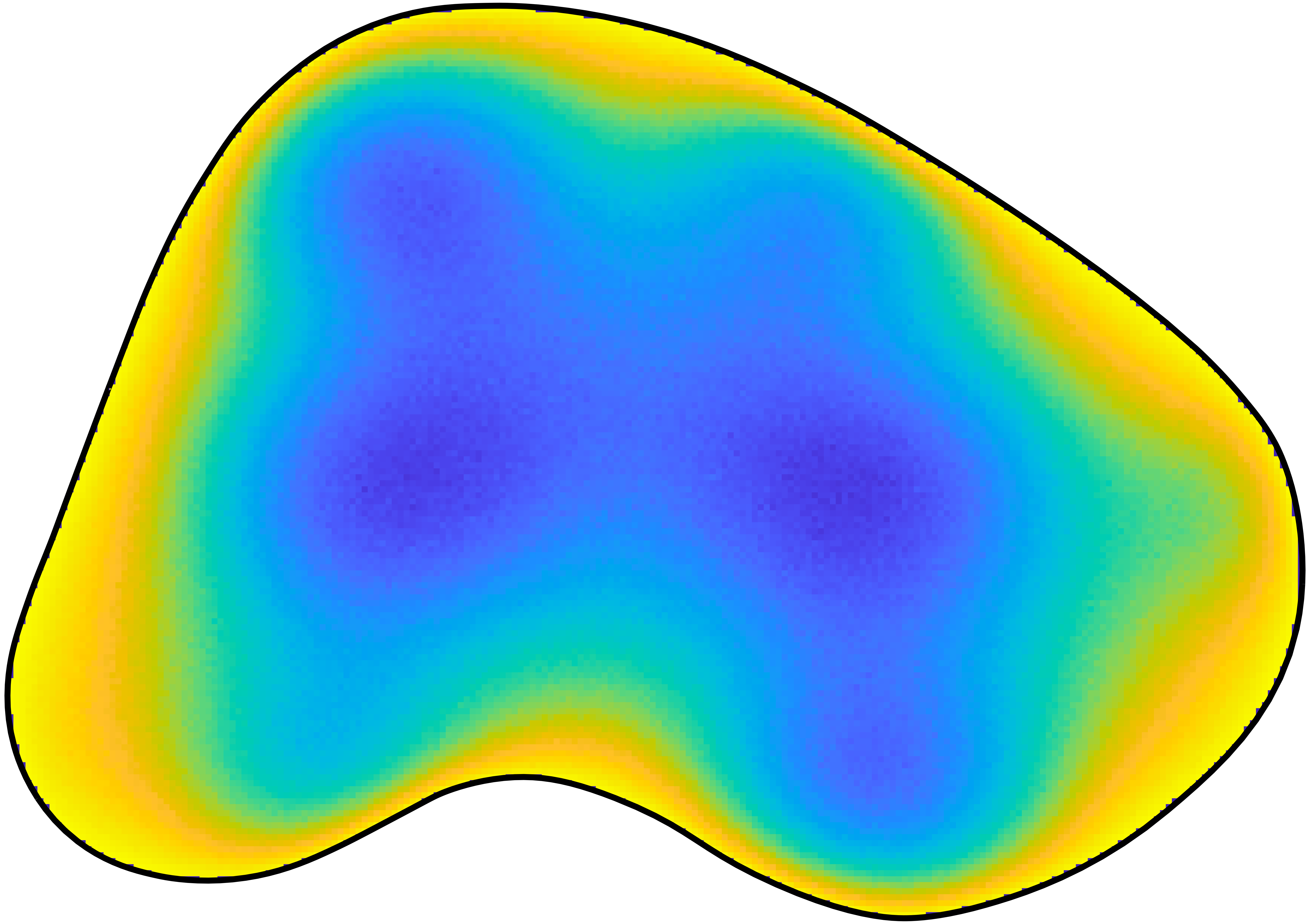
$$\Delta u_O = 0 \quad \text{on } \Omega$$
$$u_O = g \quad \text{on } \partial\Omega$$

where

$$\Omega = V \backslash O$$
$$O = \{B(c_1), B(c_2), \ldots, B(c_N)\}$$

$$\bar{u} = E_O[u_o]$$

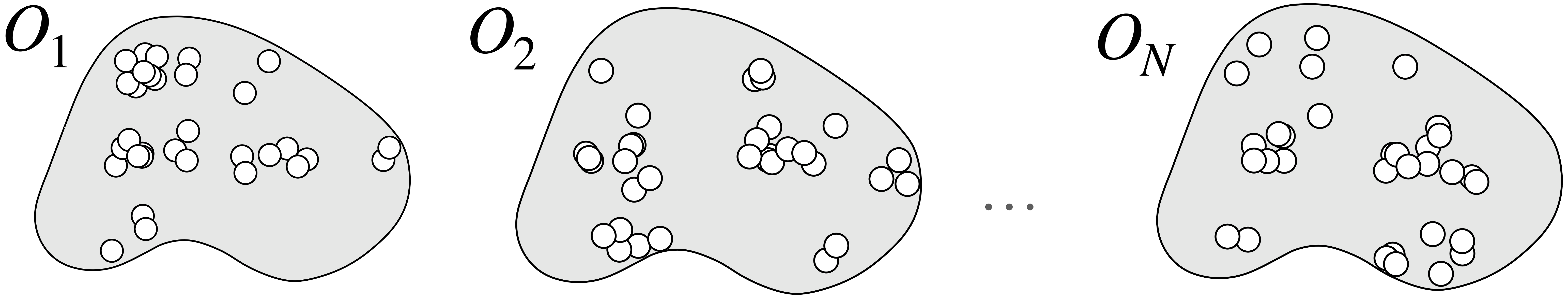
mean solution



Brute force: ensemble average

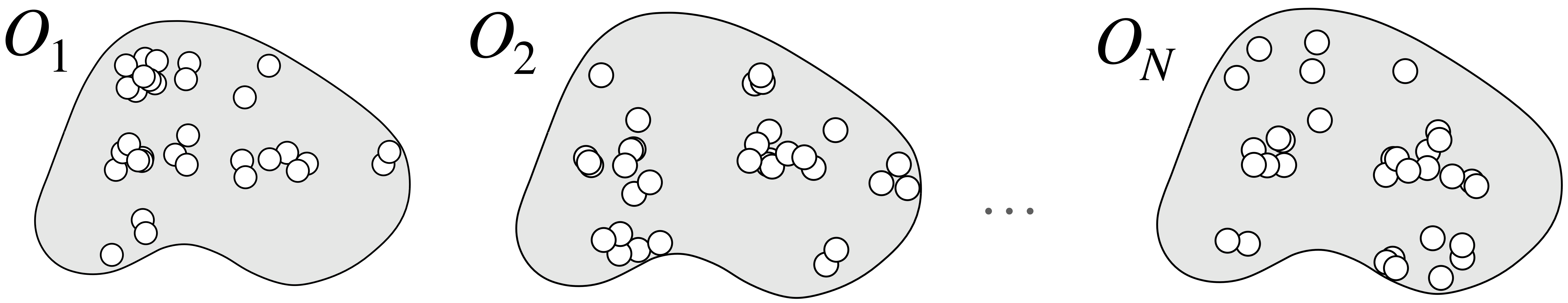
Brute force: ensemble average

sample particle configurations

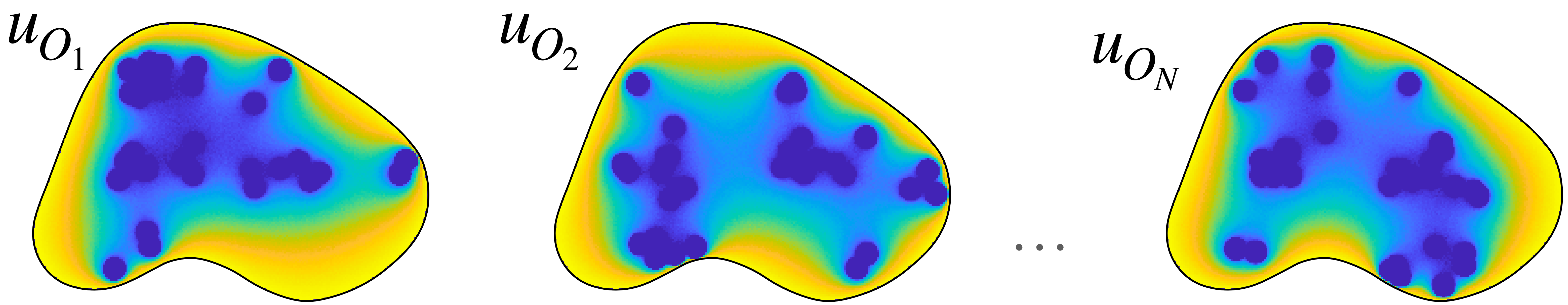


Brute force: ensemble average

sample particle configurations

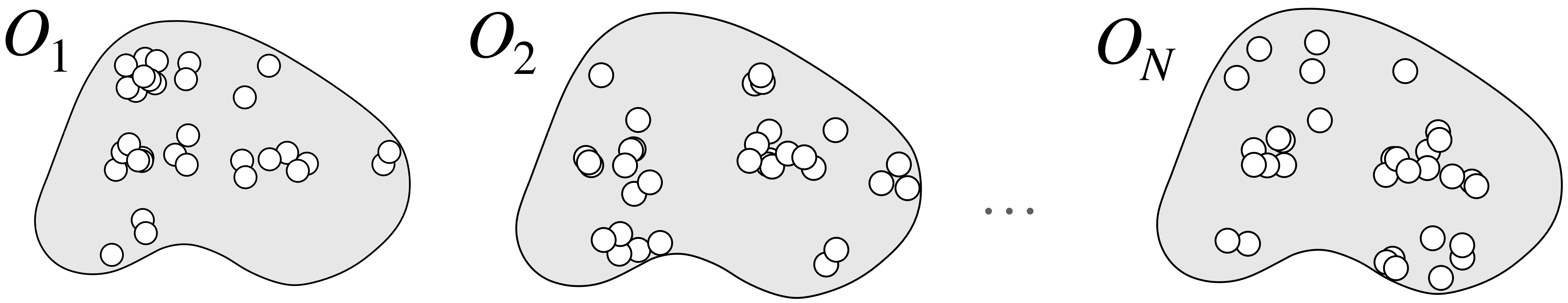


solve each configuration

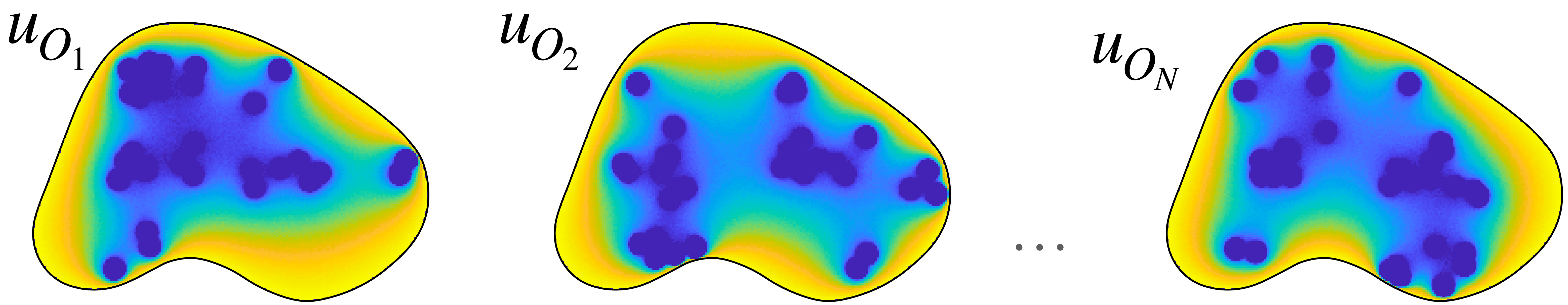


Brute force: ensemble average

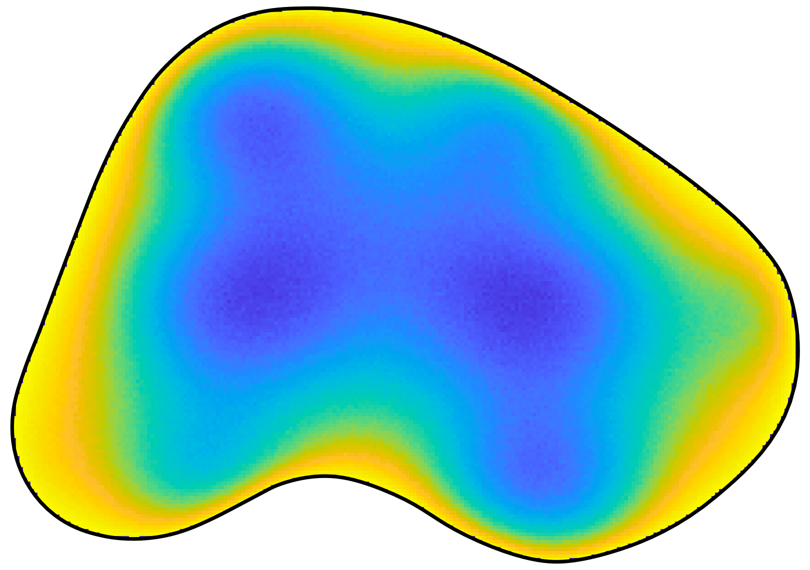
sample particle configurations



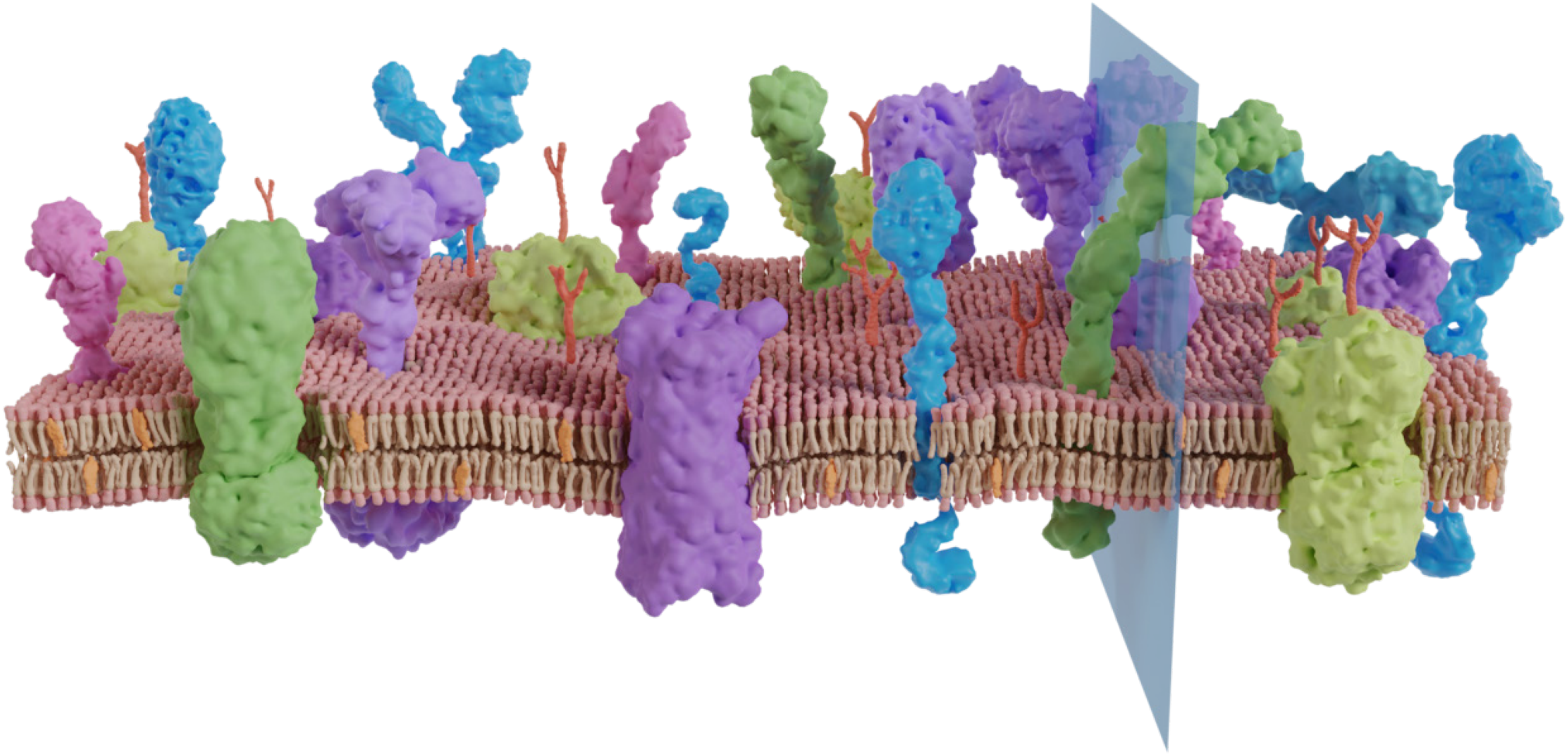
solve each configuration



compute mean solution $\bar{u}$



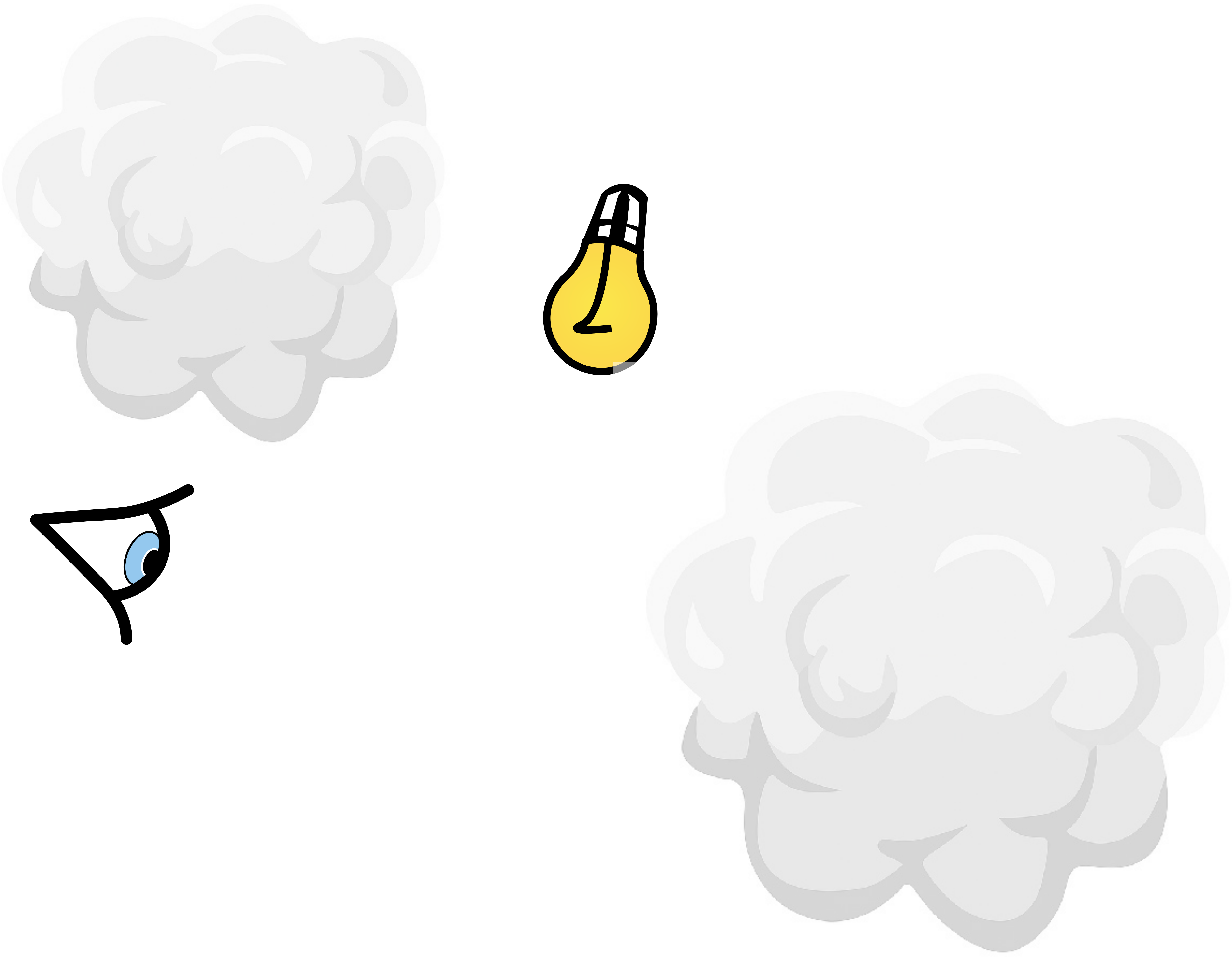
Sampling and querying all particles is slow



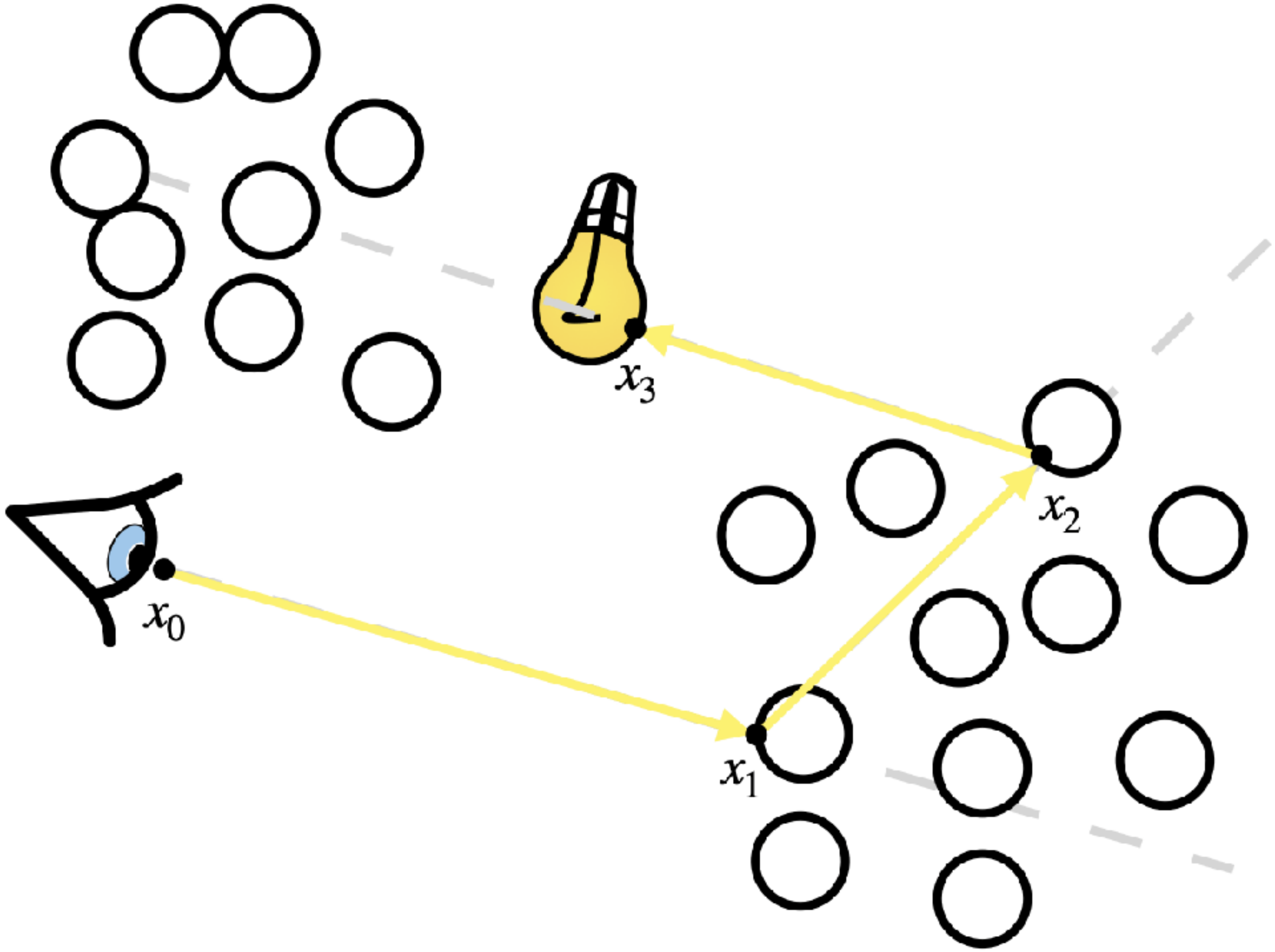
Sampling and querying all particles is slow

E[num particles] = 1.9m

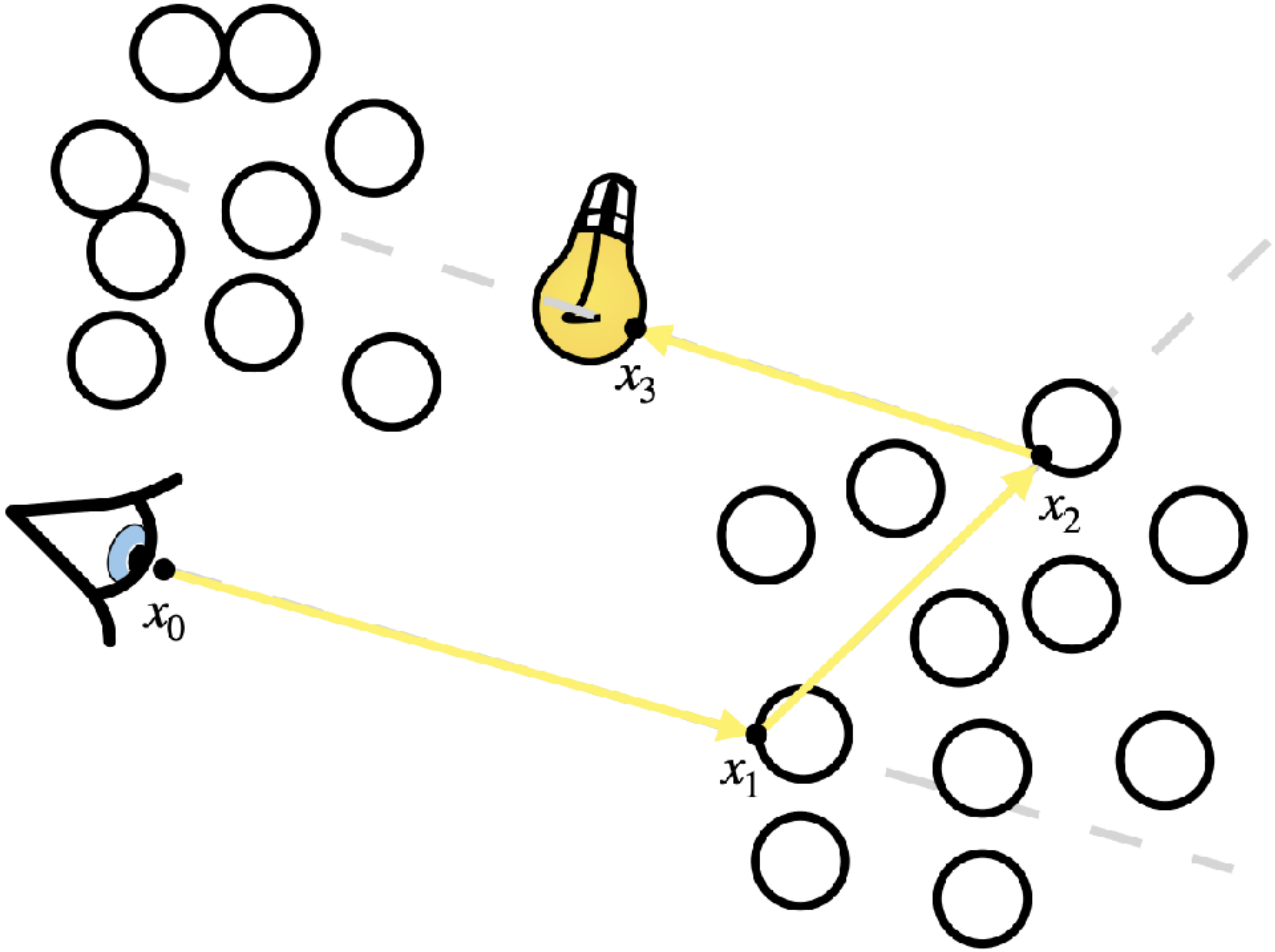
Taking inspiration from volume rendering



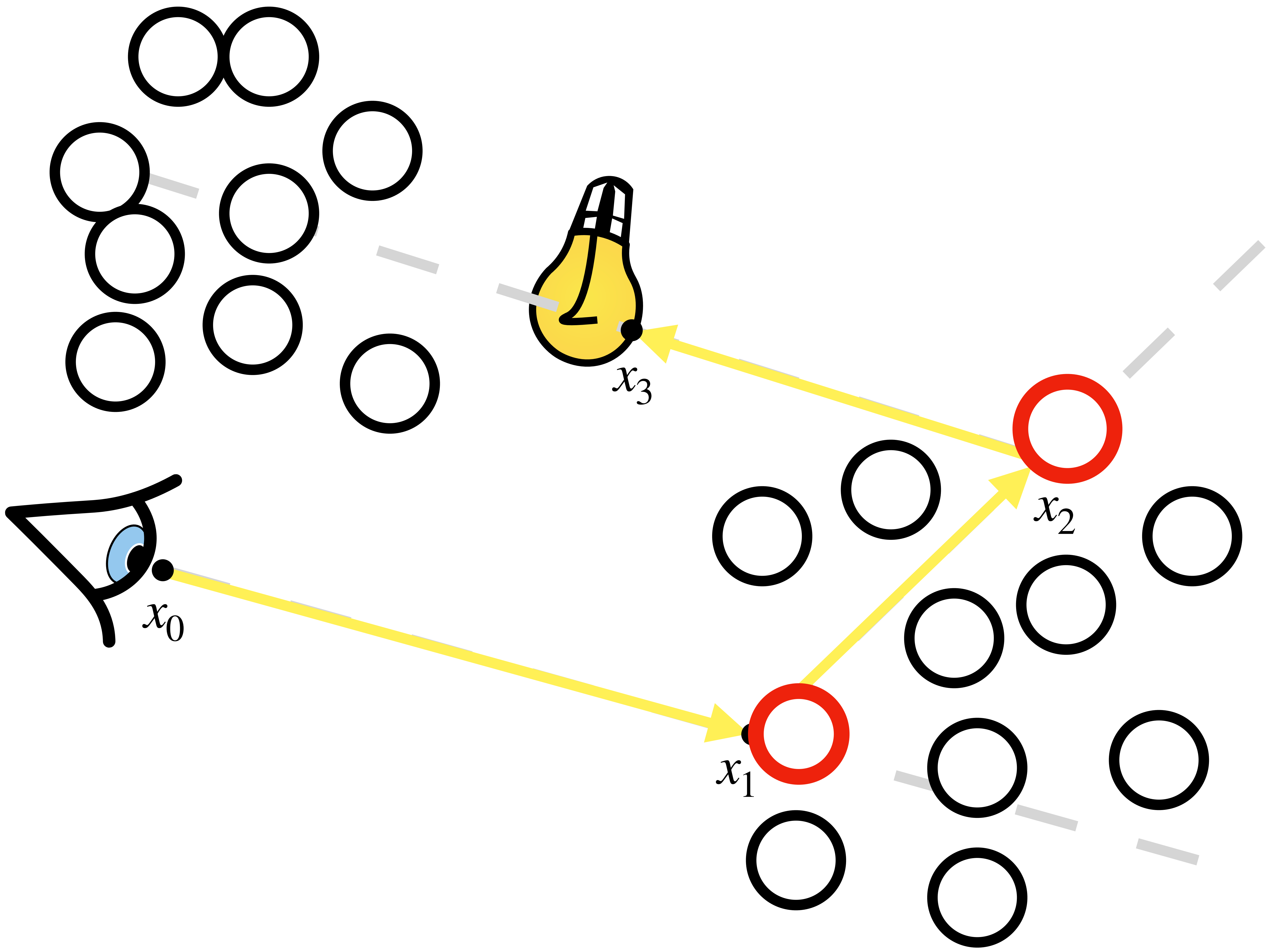
Naive approach to rendering volumes



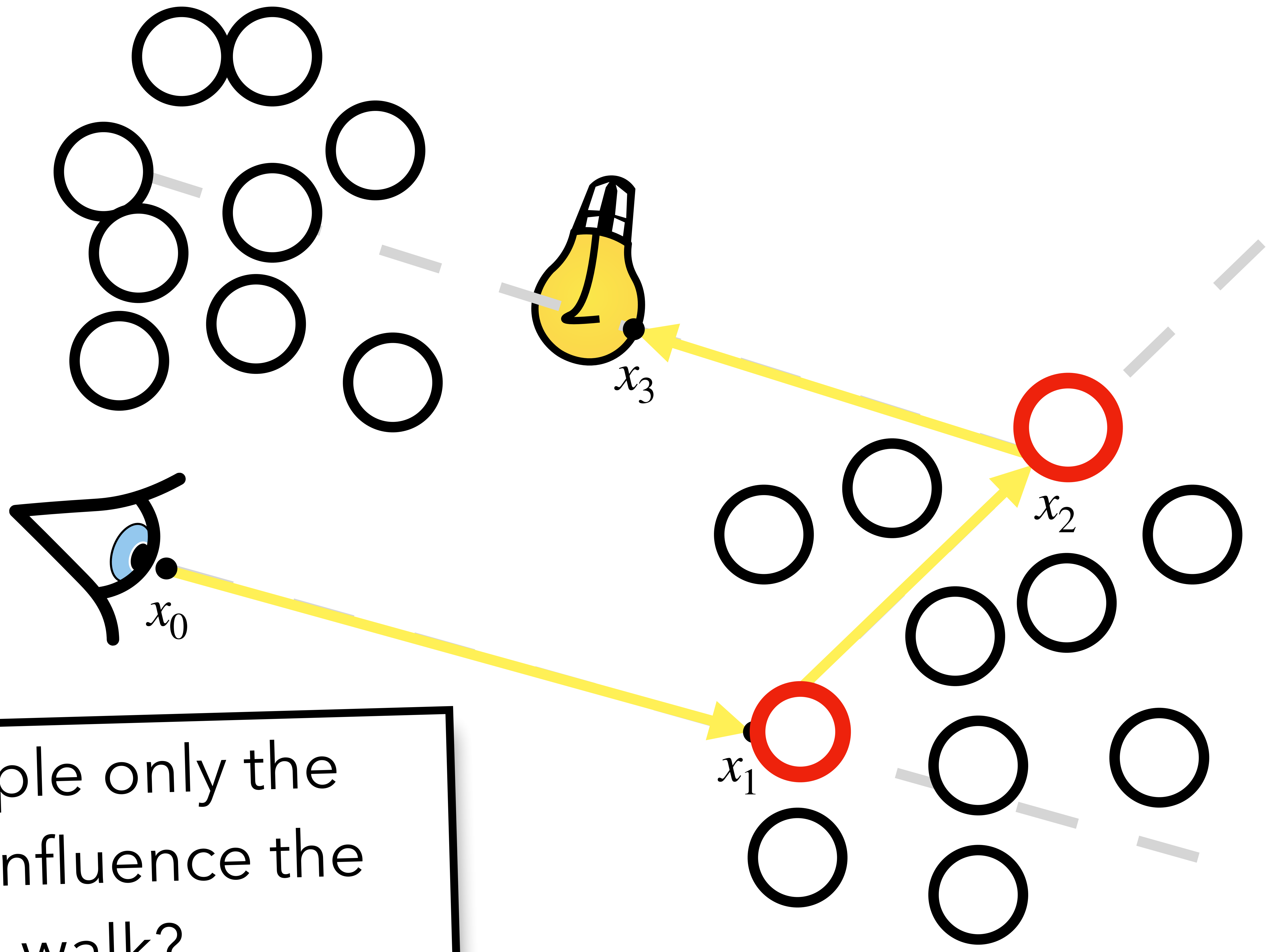
Naive approach to rendering volumes



Naive approach to rendering volumes

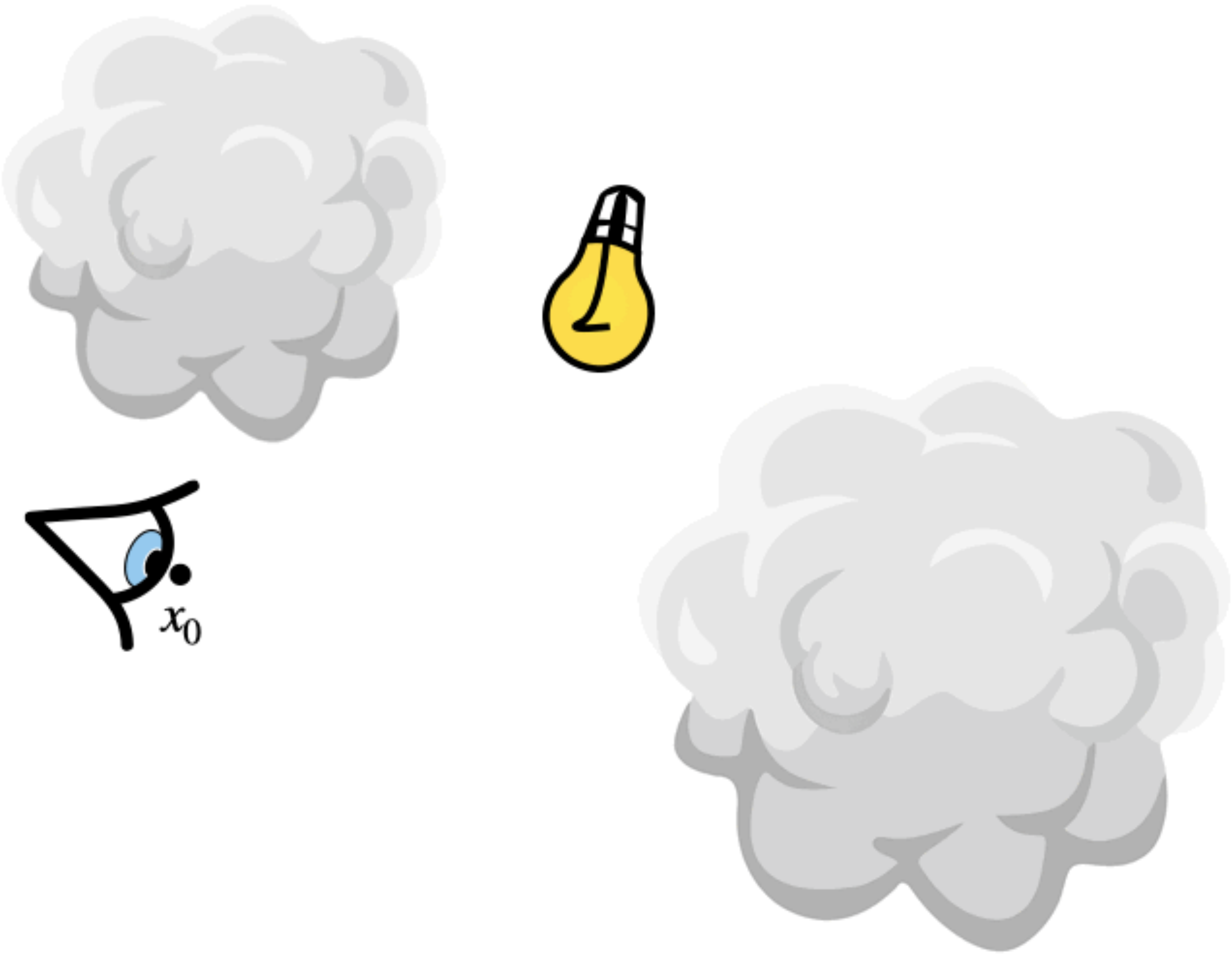


Naive approach to rendering volumes

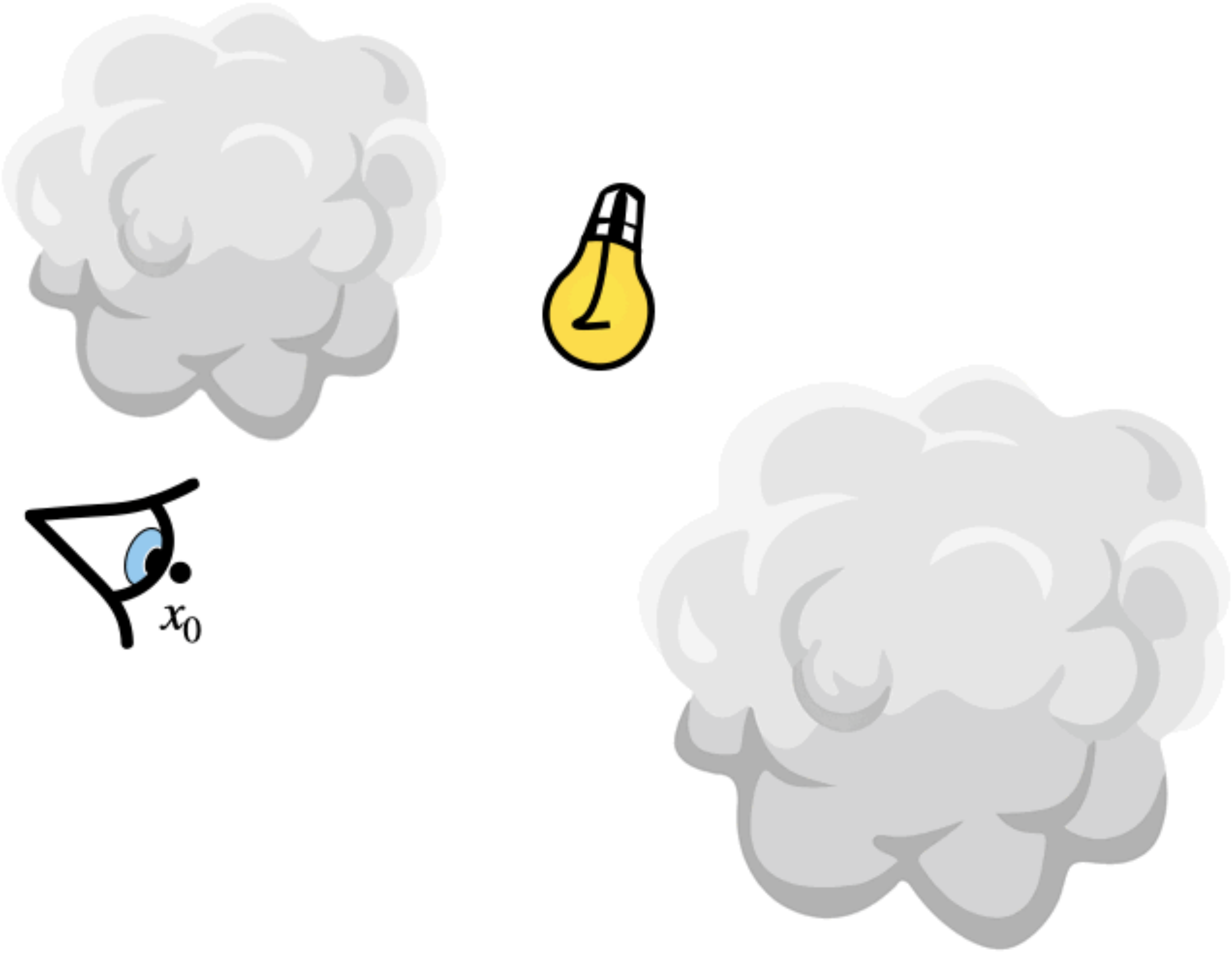


Why not sample only the particles that influence the random walk?

Volume rendering: sample only what you need

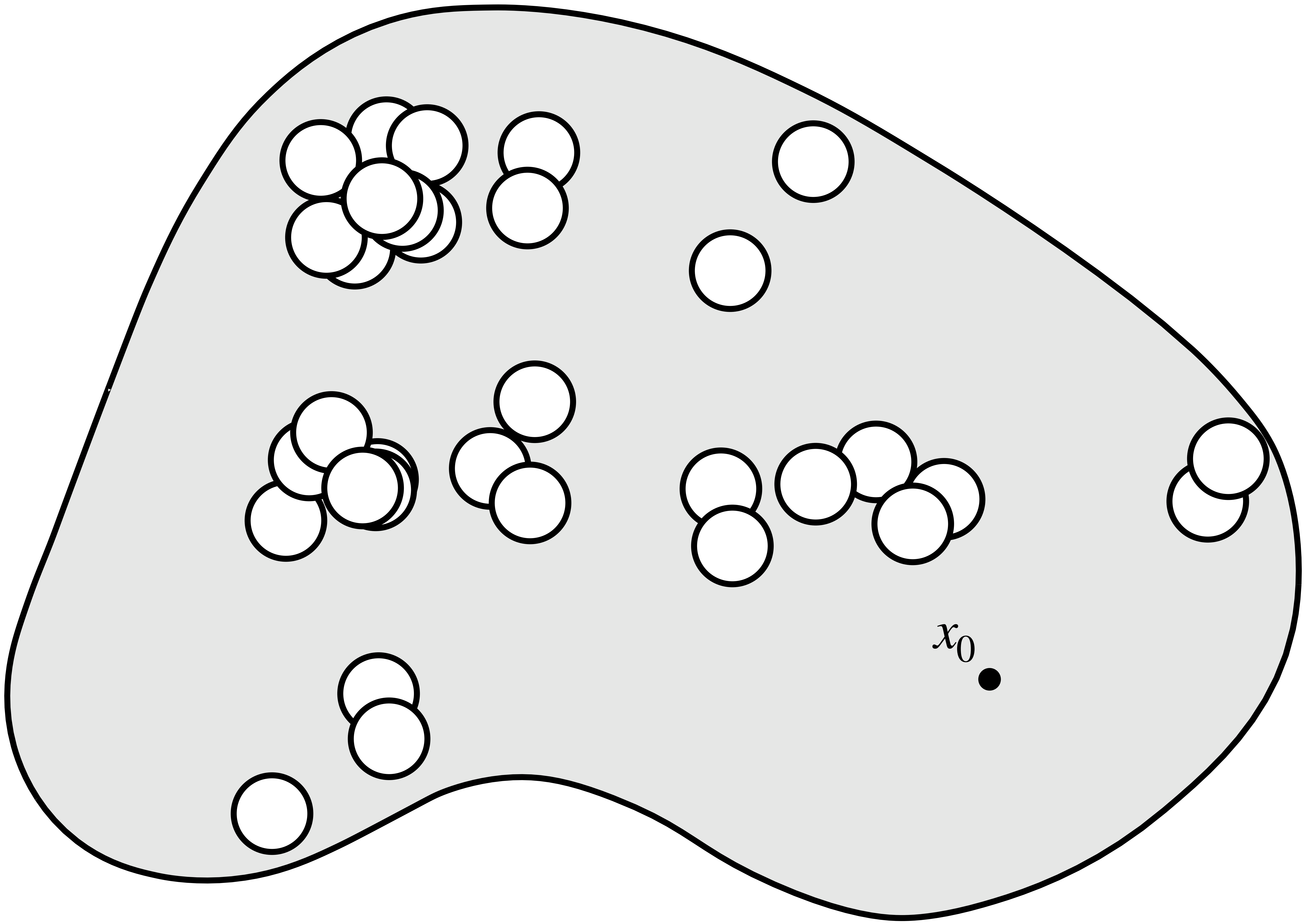


Volume rendering: sample only what you need



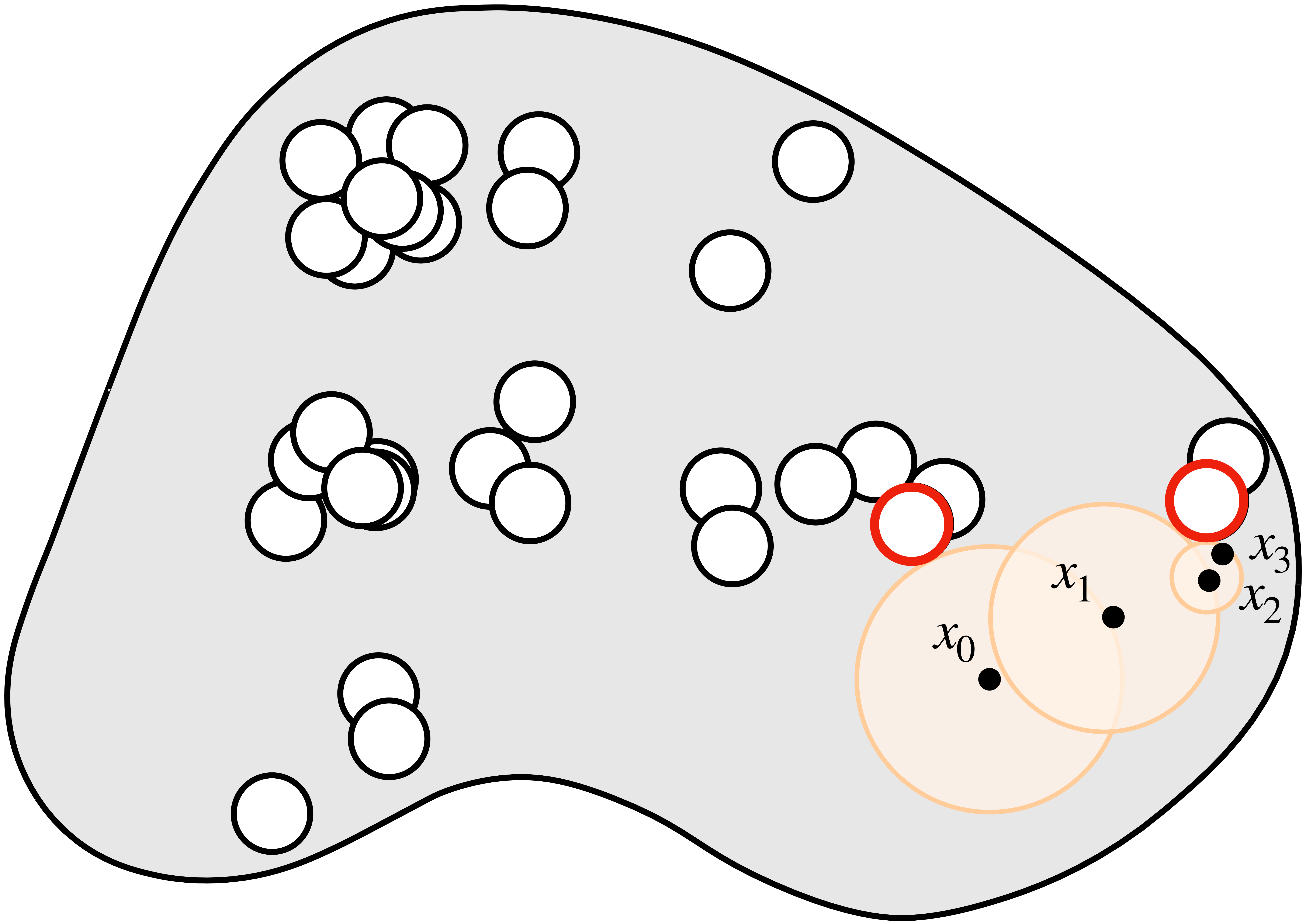
Generalizing to walk on spheres

$$\Delta u_O = 0 \quad \text{on } \Omega$$
$$u_O = g \quad \text{on } \partial\Omega$$



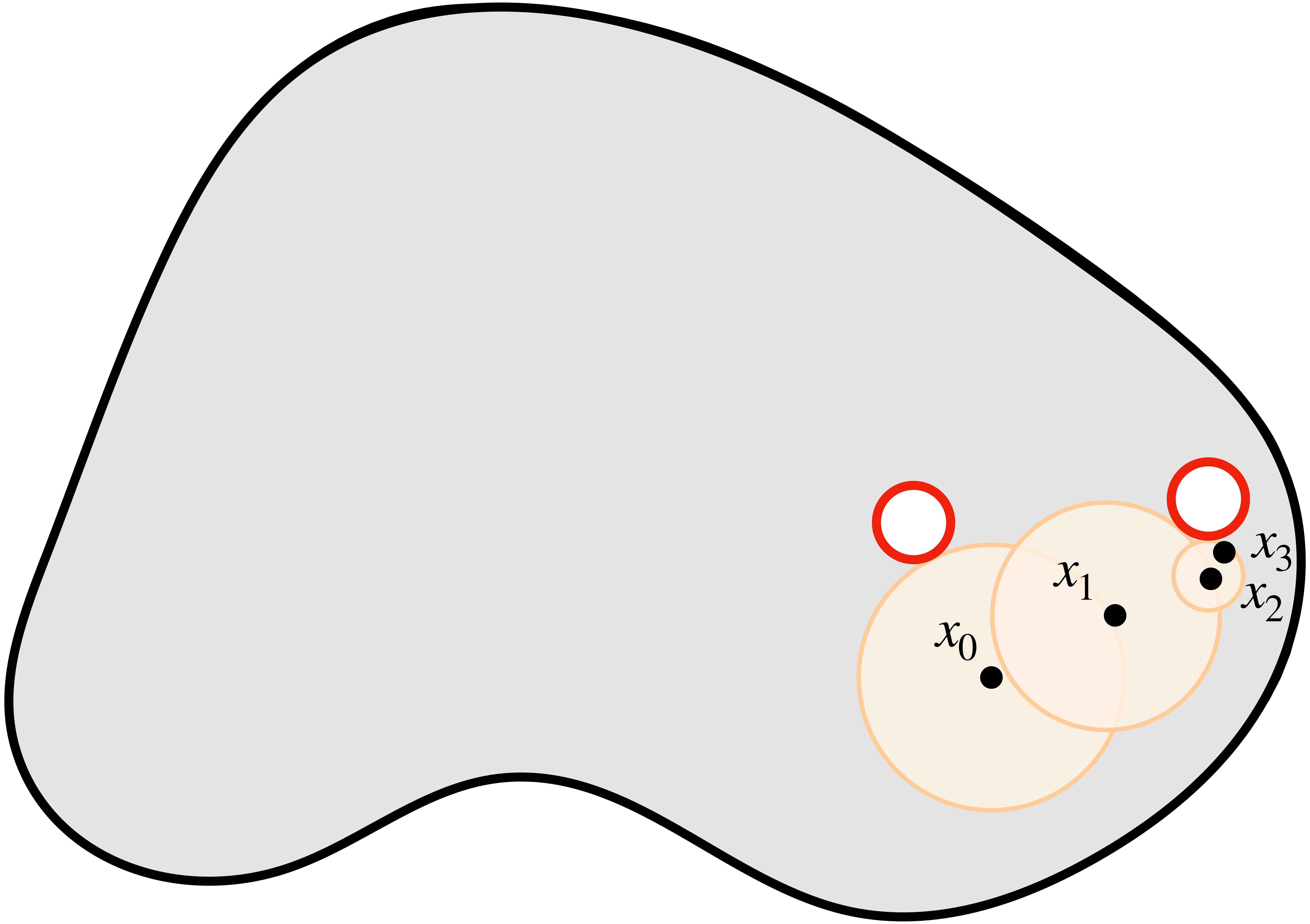
Generalizing to walk on spheres

$$\Delta u_O = 0 \quad \text{on } \Omega$$
$$u_O = g \quad \text{on } \partial\Omega$$



Generalizing to walk on spheres

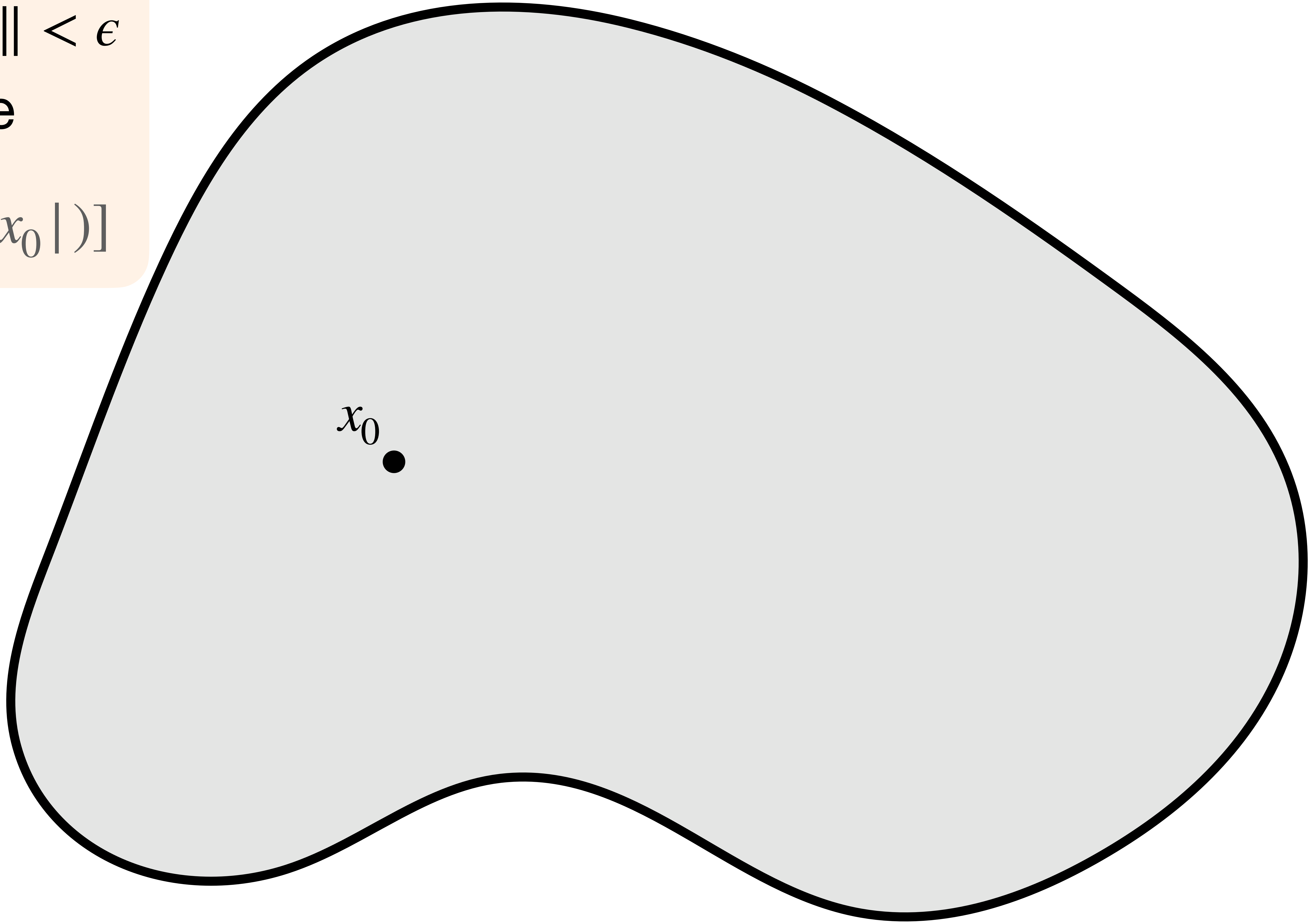
$$\Delta u_O = 0 \quad \text{on } \Omega$$
$$u_O = g \quad \text{on } \partial\Omega$$



Volumetric walk on spheres

$$\langle \bar{u}(x_0) \rangle = \begin{cases} g(y_0) & \text{if } \|x_0 - y_0\| < \epsilon \\ \langle \bar{u}(x_1 \mid y_0) \rangle & \text{otherwise} \end{cases}$$

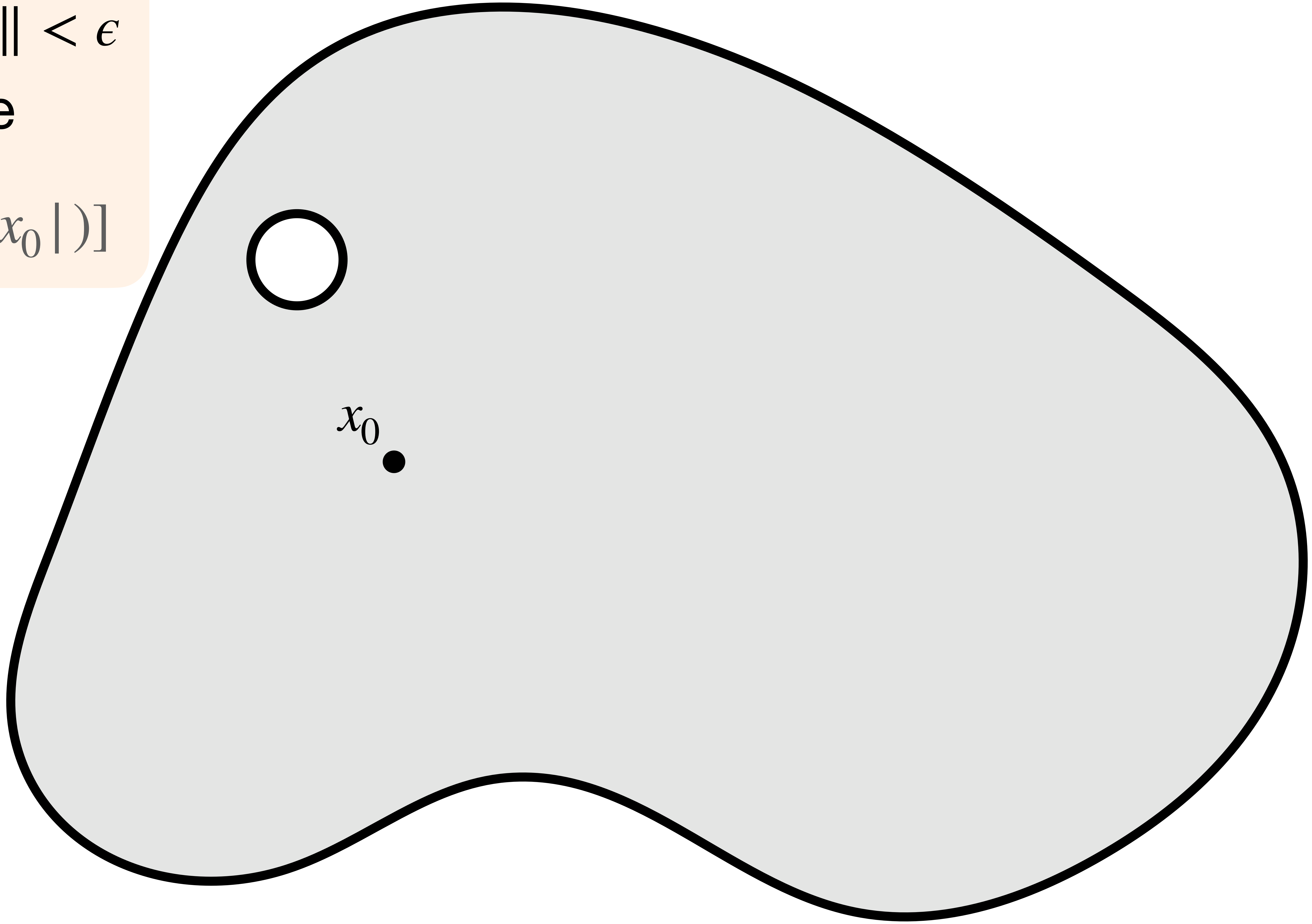
$y_i \sim p_i^{cp} \quad x_1 \sim \mathcal{U}[\partial B(x_0, |y_0 - x_0|)]$



Volumetric walk on spheres

$$\langle \bar{u}(x_0) \rangle = \begin{cases} g(y_0) & \text{if } \|x_0 - y_0\| < \epsilon \\ \langle \bar{u}(x_1 \mid y_0) \rangle & \text{otherwise} \end{cases}$$

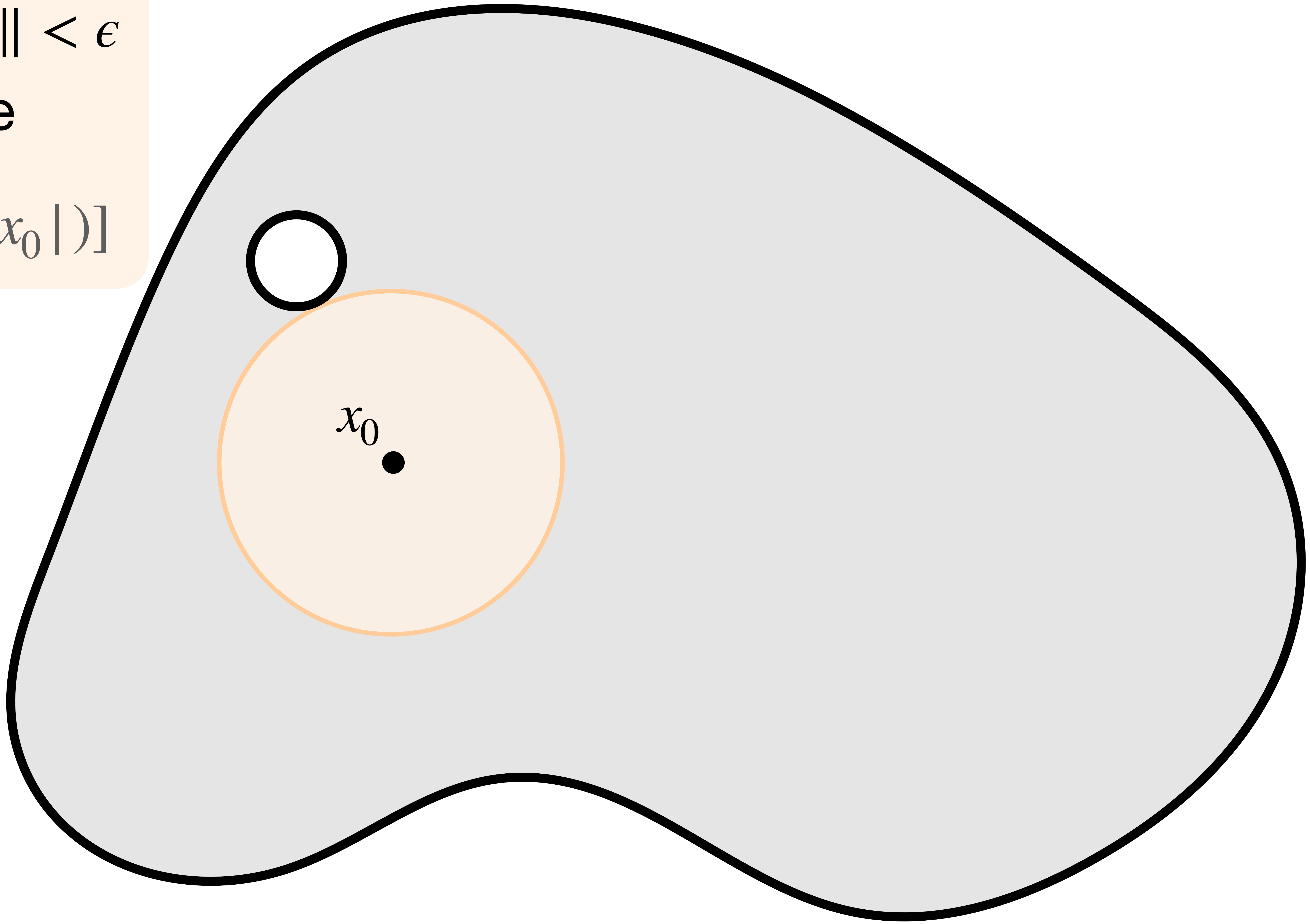
$y_i \sim p_i^{cp} \quad x_1 \sim \mathcal{U}[\partial B(x_0, |y_0 - x_0|)]$



Volumetric walk on spheres

$$\langle \bar{u}(x_0) \rangle = \begin{cases} g(y_0) & \text{if } \|x_0 - y_0\| < \epsilon \\ \langle \bar{u}(x_1 \mid y_0) \rangle & \text{otherwise} \end{cases}$$

$y_i \sim p_i^{cp} \quad x_1 \sim \mathcal{U}[\partial B(x_0, |y_0 - x_0|)]$

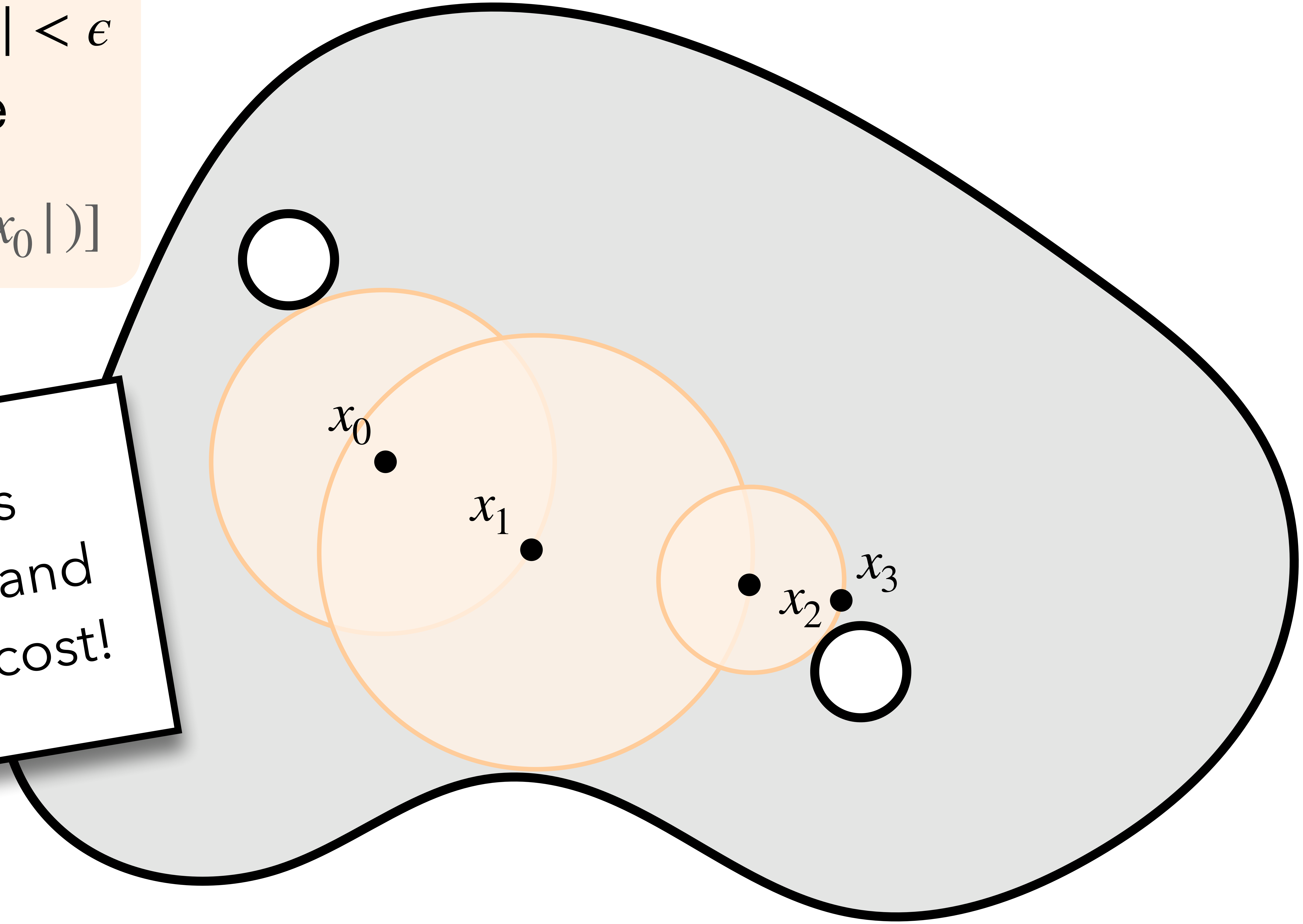


Volumetric walk on spheres

$$\langle \bar{u}(x_0) \rangle = \begin{cases} g(y_0) & \text{if } \|x_0 - y_0\| < \epsilon \\ \langle \bar{u}(x_1 \mid y_0) \rangle & \text{otherwise} \end{cases}$$

$$y_i \sim p_i^{cp} \qquad x_1 \sim \mathcal{U}[\partial B(x_0, |y_0 - x_0|)]$$

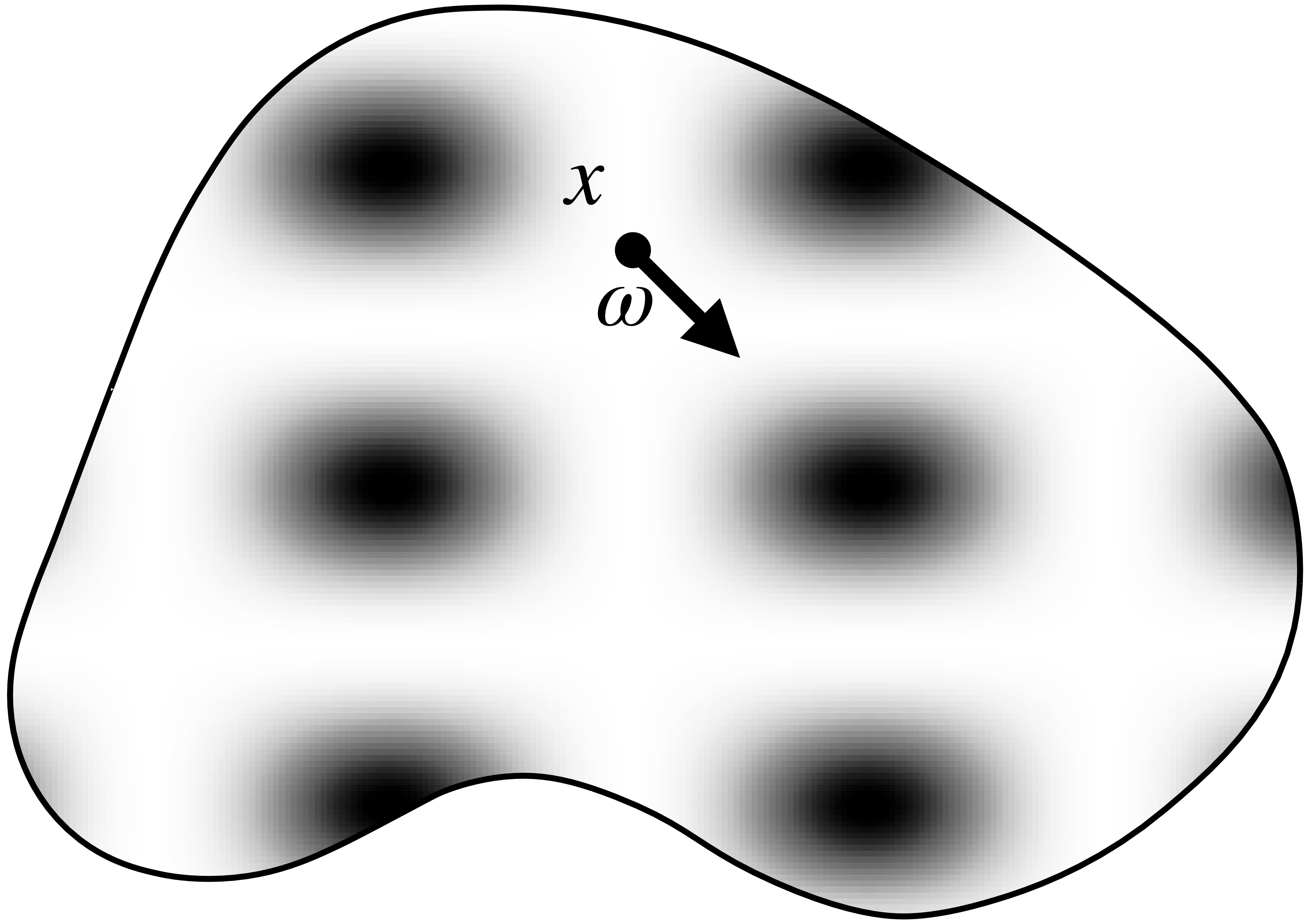
Volumetric walk on spheres avoids sampling all particles and reduces closest-point query cost!



Stochastic queries with Poisson Boolean model

free-flight distribution (volume rendering)

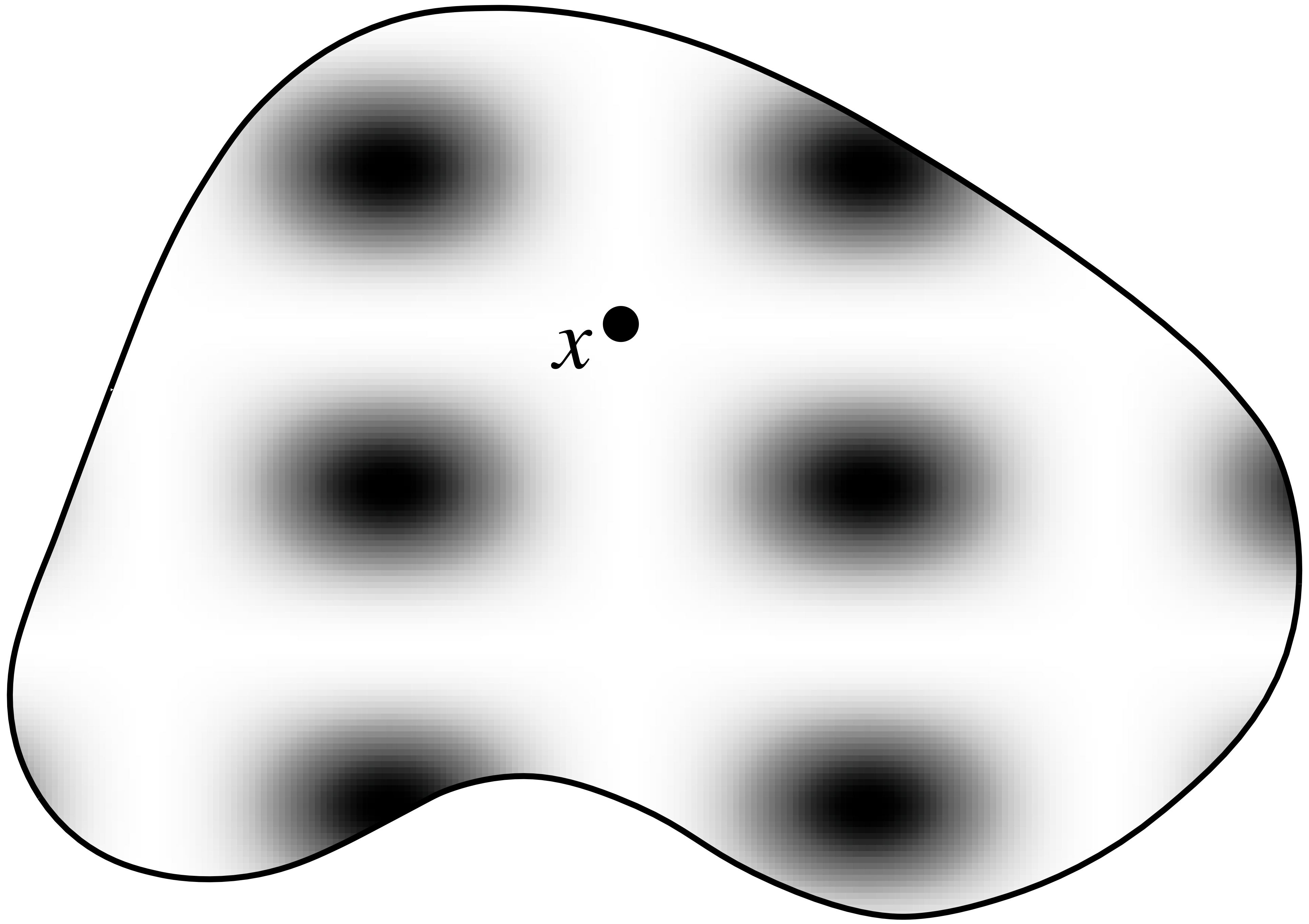
$$p^{ff}_{x,\omega}(t) = \exp\left(-\int_0^t \lambda(x+\omega s)\ \mathrm{d}s\right)\ \lambda(x+\omega t)$$



$\lambda(x)$

spherical-contact distribution (volumetric WoS)

$$p^{cp}_{x}(y) = \exp\left(-\int_{B(x,R)} \lambda(z)\ \mathrm{d}z\right)\ \lambda(y)$$



$\lambda(x)$

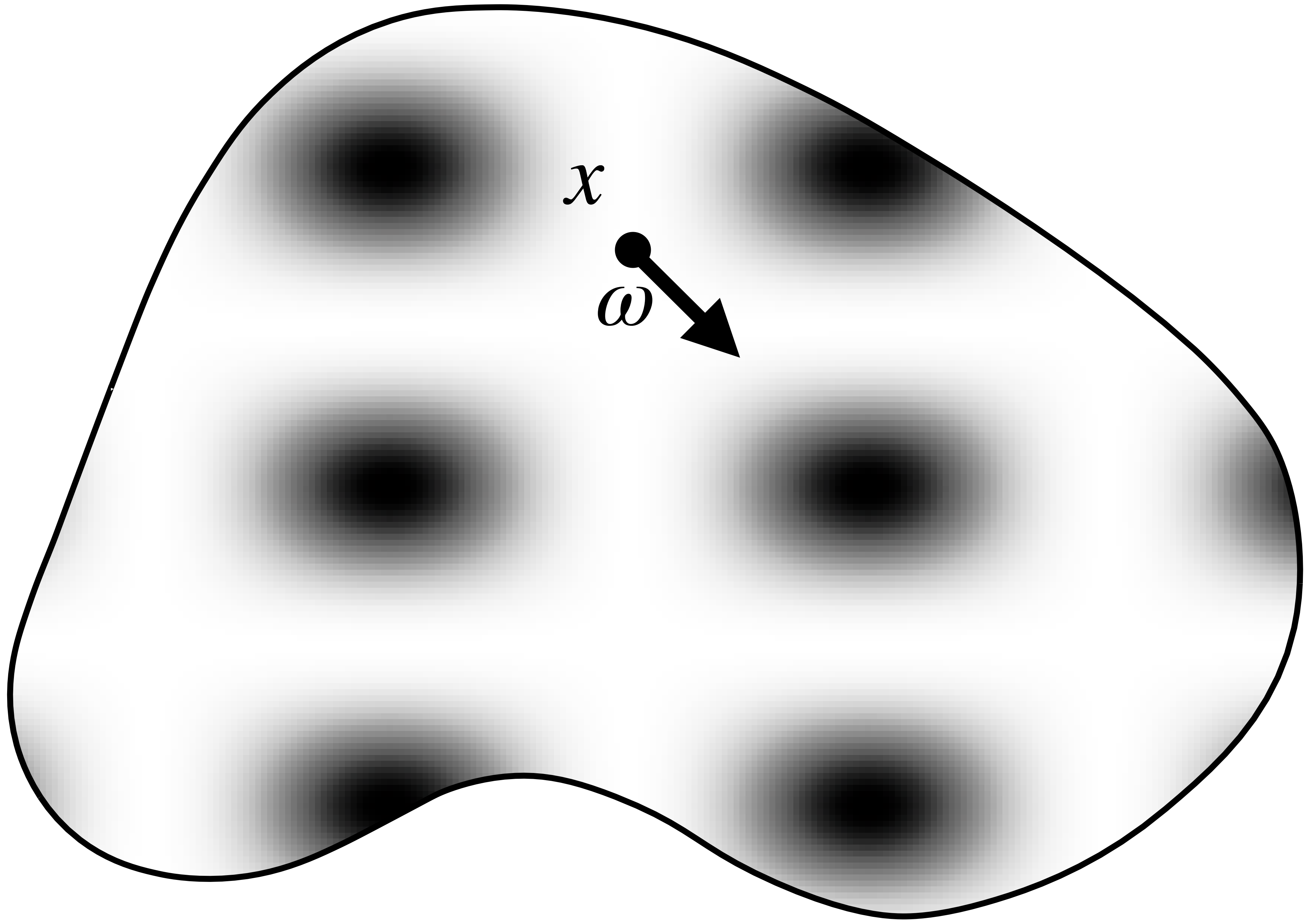
Stochastic queries with Poisson Boolean model

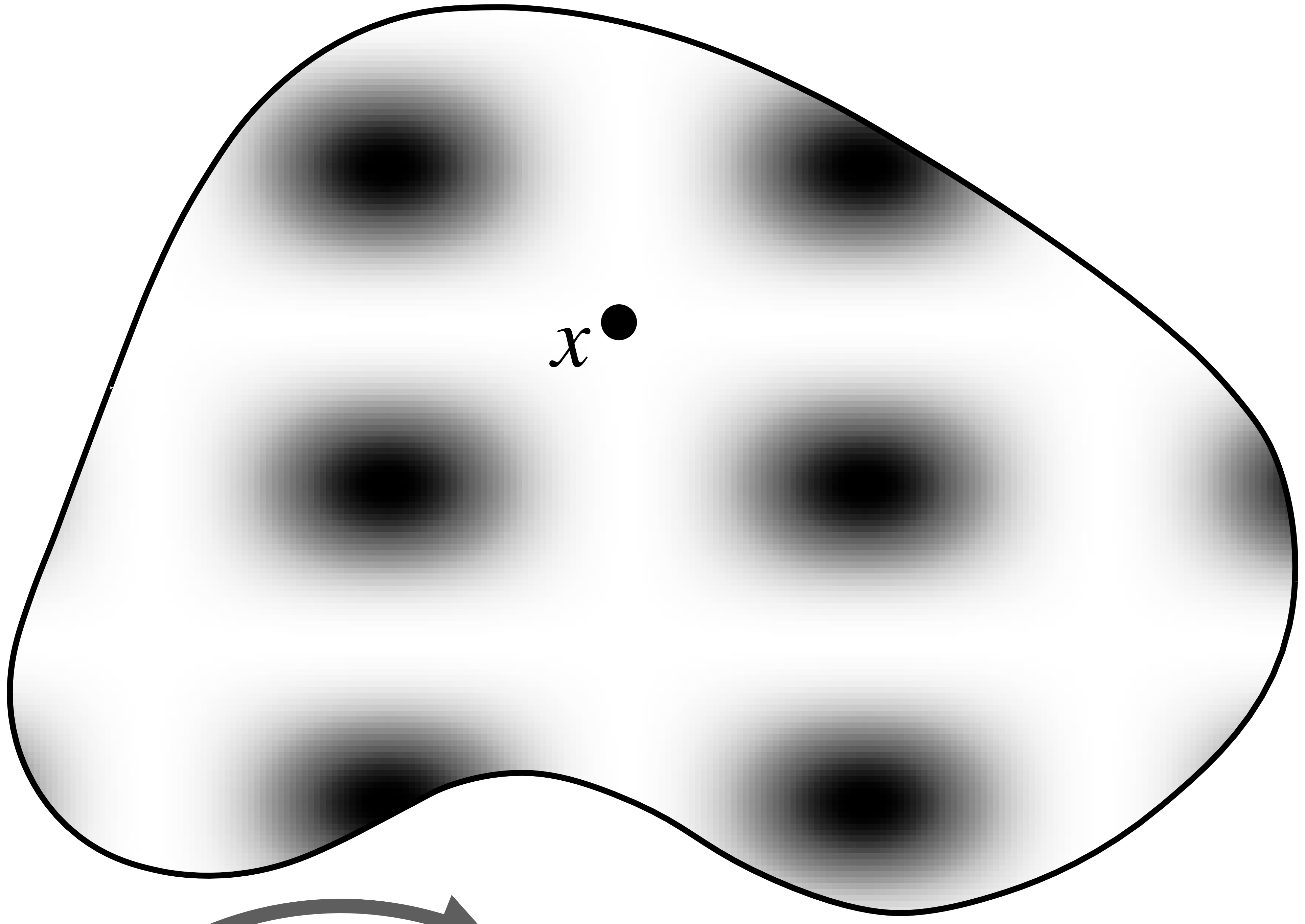
free-flight distribution (volume rendering)

$$p^{ff}_{x,\omega}(t) = \exp\left(-\int_0^t \lambda(x+\omega s)\ \mathrm{d}s\right)\ \lambda(x+\omega t)$$

spherical-contact distribution (volumetric WoS)

$$p^{cp}_{x}(y) = \exp\left(-\int_{B(x,R)} \lambda(z)\ \mathrm{d}z\right)\ \lambda(y)$$





$\lambda(x)$ **expected particles per unit volume** $\lambda(x)$

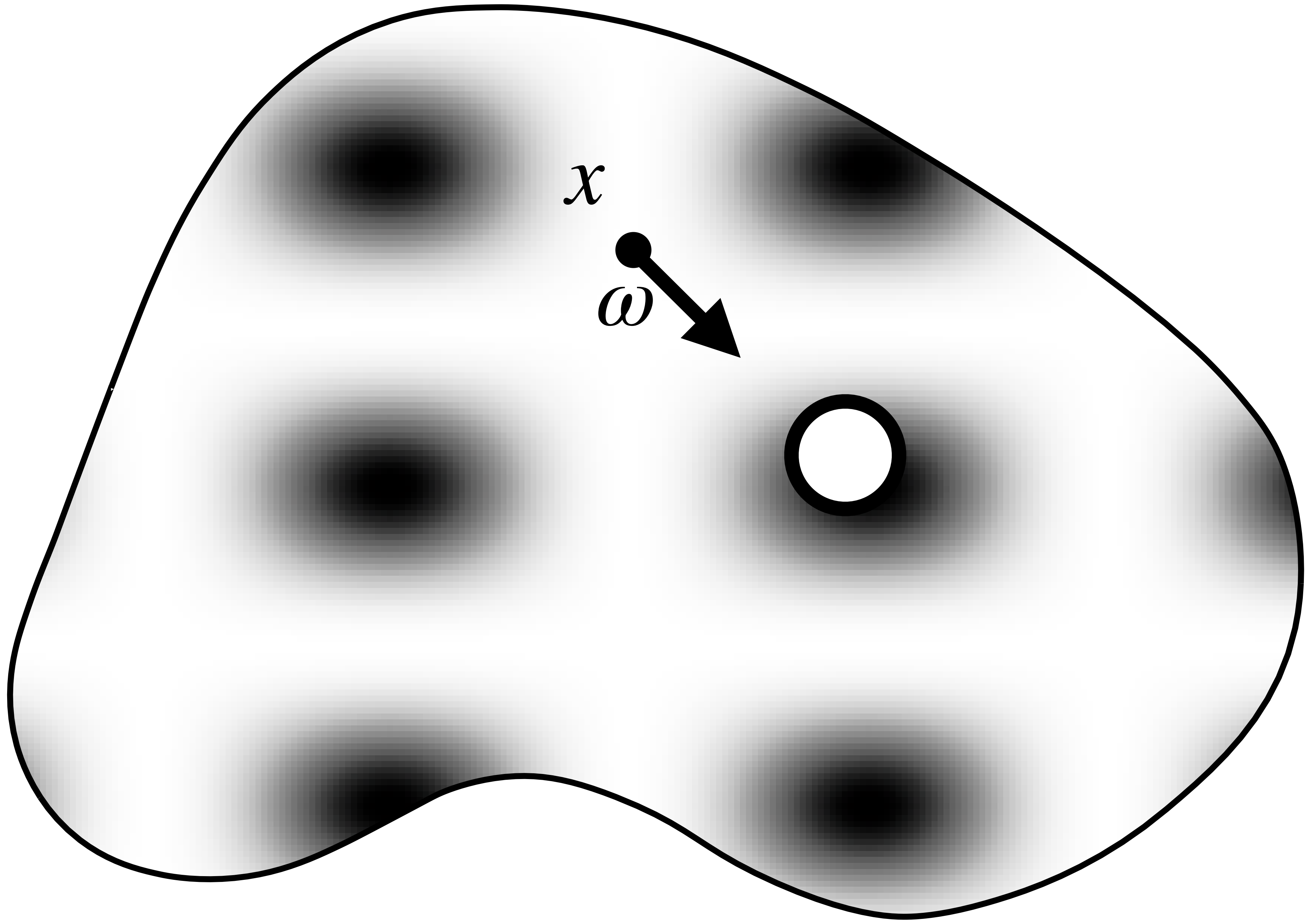
Stochastic queries with Poisson Boolean model

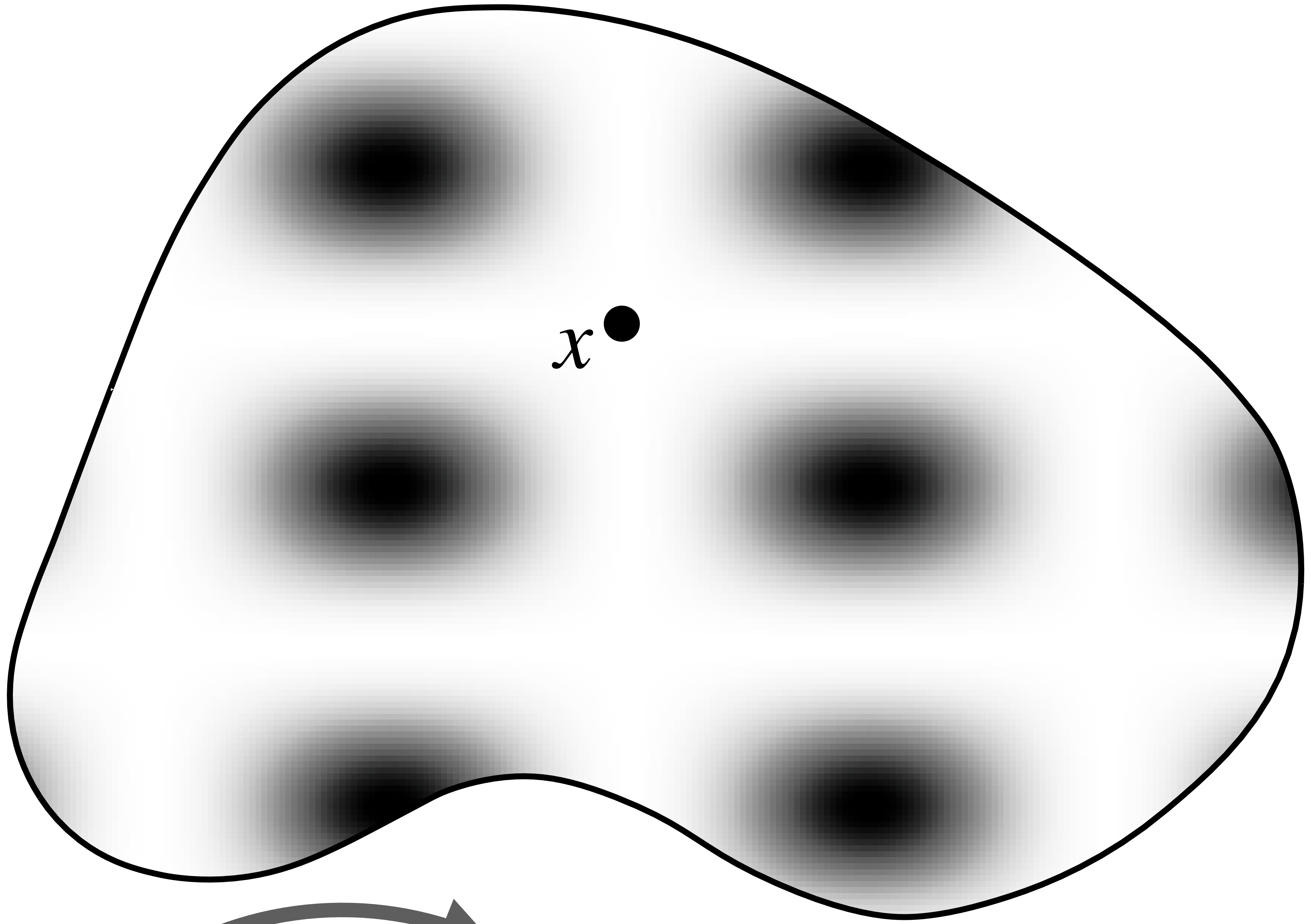
free-flight distribution (volume rendering)

$$p^{ff}_{x,\omega}(t) = \exp\left(-\int_0^t \lambda(x+\omega s)\ \mathrm{d}s\right)\ \lambda(x+\omega t)$$

spherical-contact distribution (volumetric WoS)

$$p^{cp}_{x}(y) = \exp\left(-\int_{B(x,R)} \lambda(z)\ \mathrm{d}z\right)\ \lambda(y)$$





$\lambda(x)$ **expected particles per unit volume** $\lambda(x)$

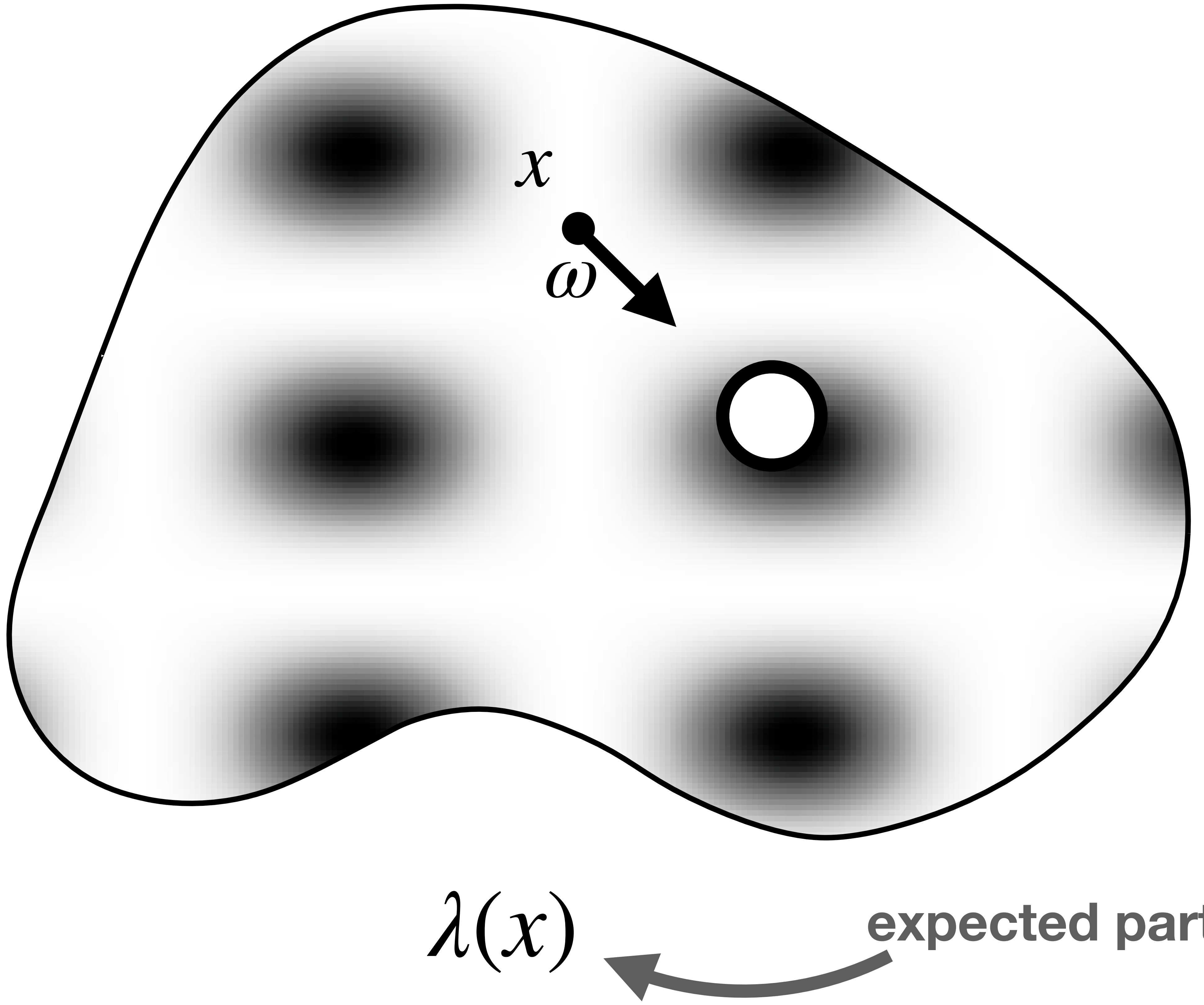
Stochastic queries with Poisson Boolean model

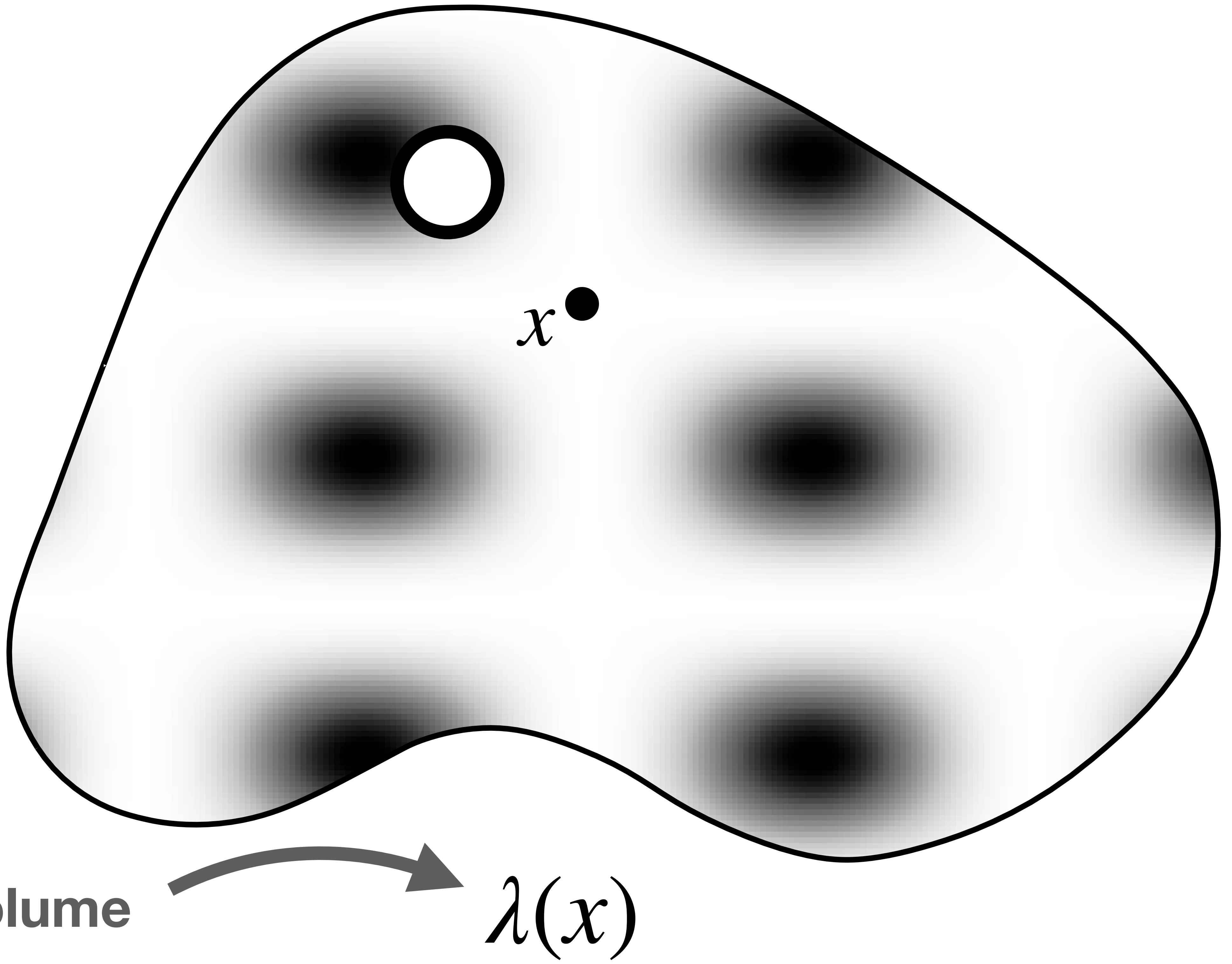
free-flight distribution (volume rendering)

$$p^{ff}_{x,\omega}(t) = \exp\left(-\int_0^t \lambda(x+\omega s)\ \mathrm{d}s\right)\ \lambda(x+\omega t)$$

spherical-contact distribution (volumetric WoS)

$$p^{cp}_{x}(y) = \exp\left(-\int_{B(x,R)} \lambda(z)\ \mathrm{d}z\right)\ \lambda(y)$$





expected particles per unit volume

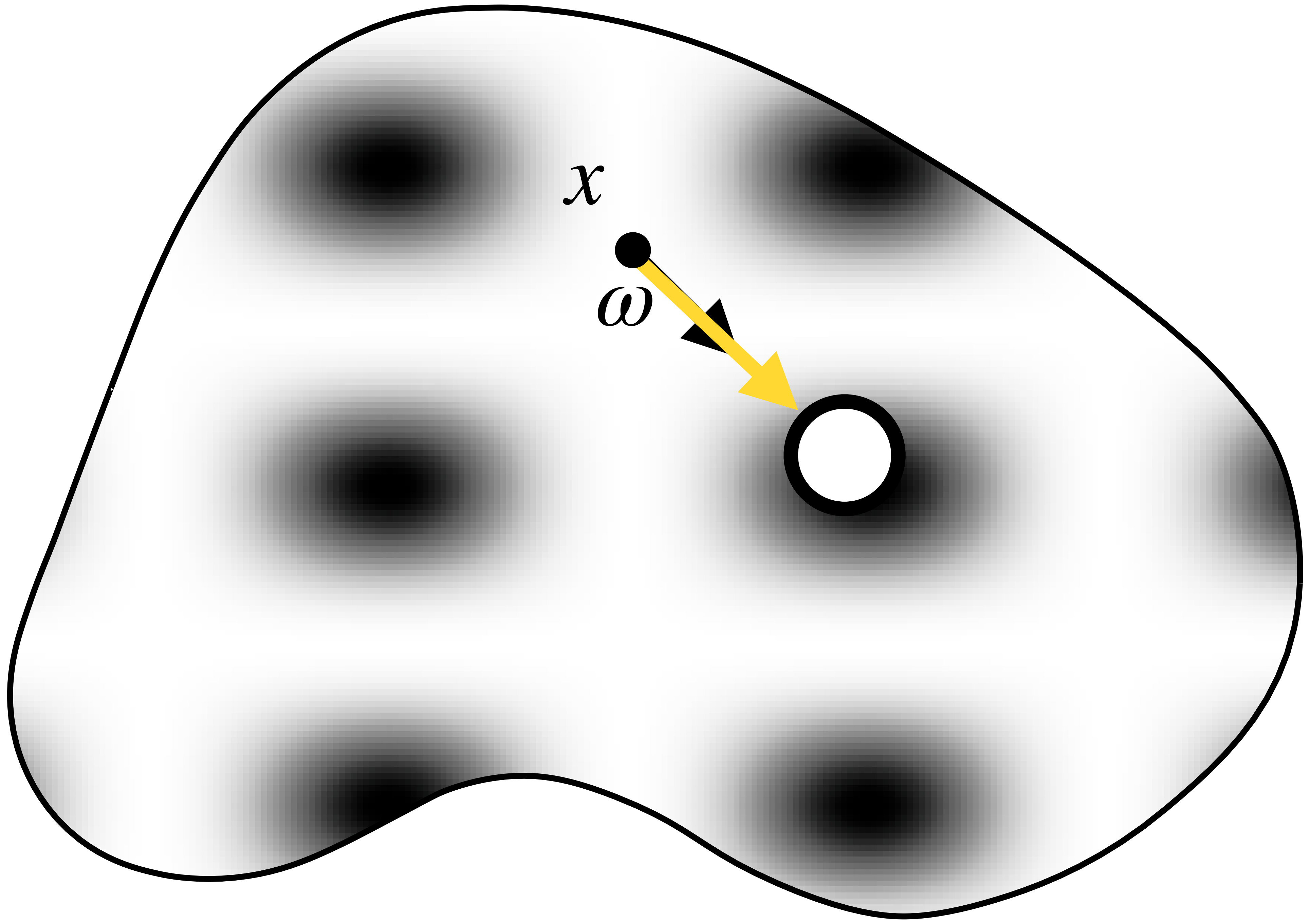
Stochastic queries with Poisson Boolean model

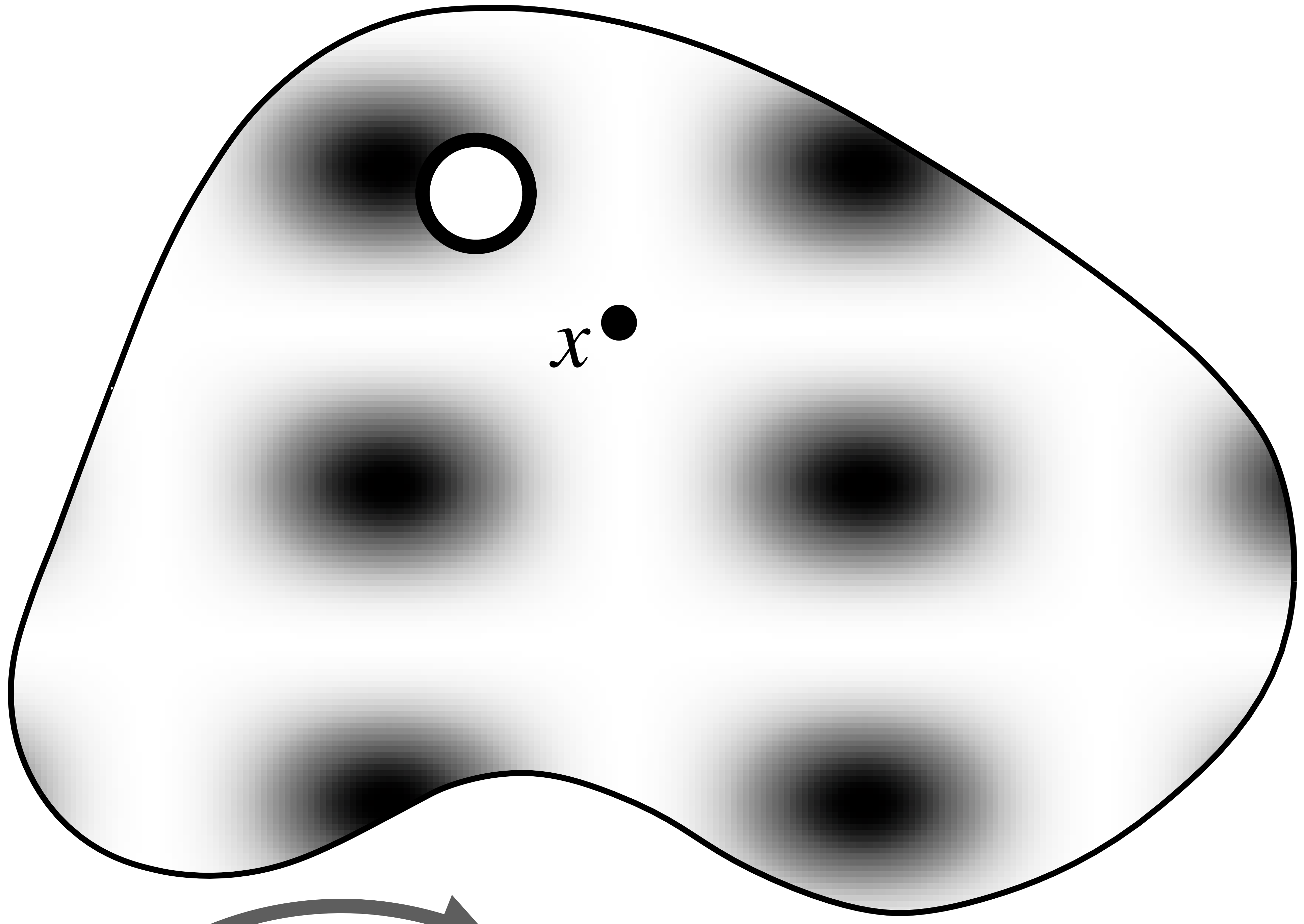
free-flight distribution (volume rendering)

$$p^{ff}_{x,\omega}(t) = \exp\left(-\int_0^t \lambda(x+\omega s)\ \mathrm{d}s\right)\ \lambda(x+\omega t)$$

spherical-contact distribution (volumetric WoS)

$$p^{cp}_{x}(y) = \exp\left(-\int_{B(x,R)} \lambda(z)\ \mathrm{d}z\right)\ \lambda(y)$$





$\lambda(x)$ **expected particles per unit volume** $\lambda(x)$

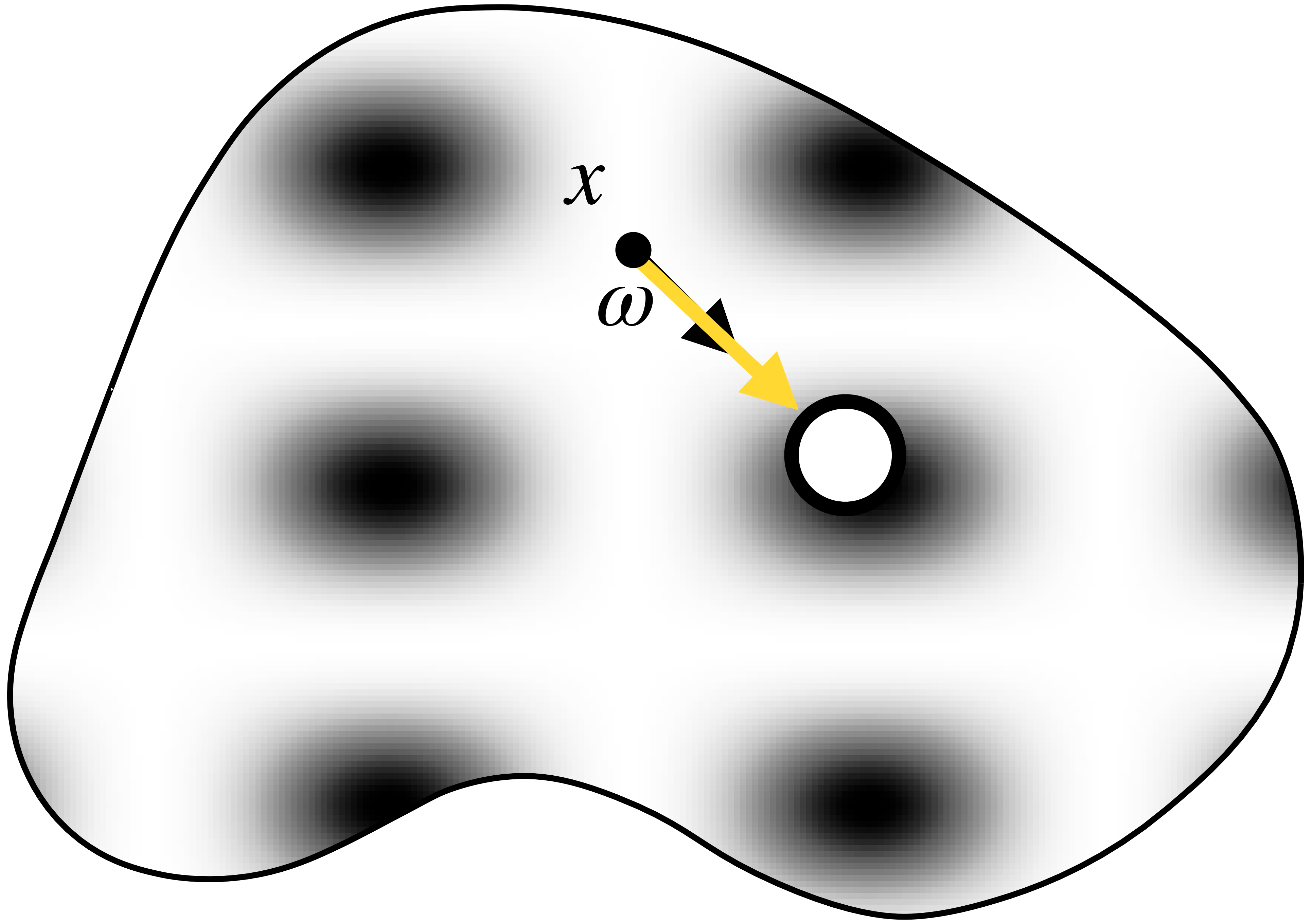
Stochastic queries with Poisson Boolean model

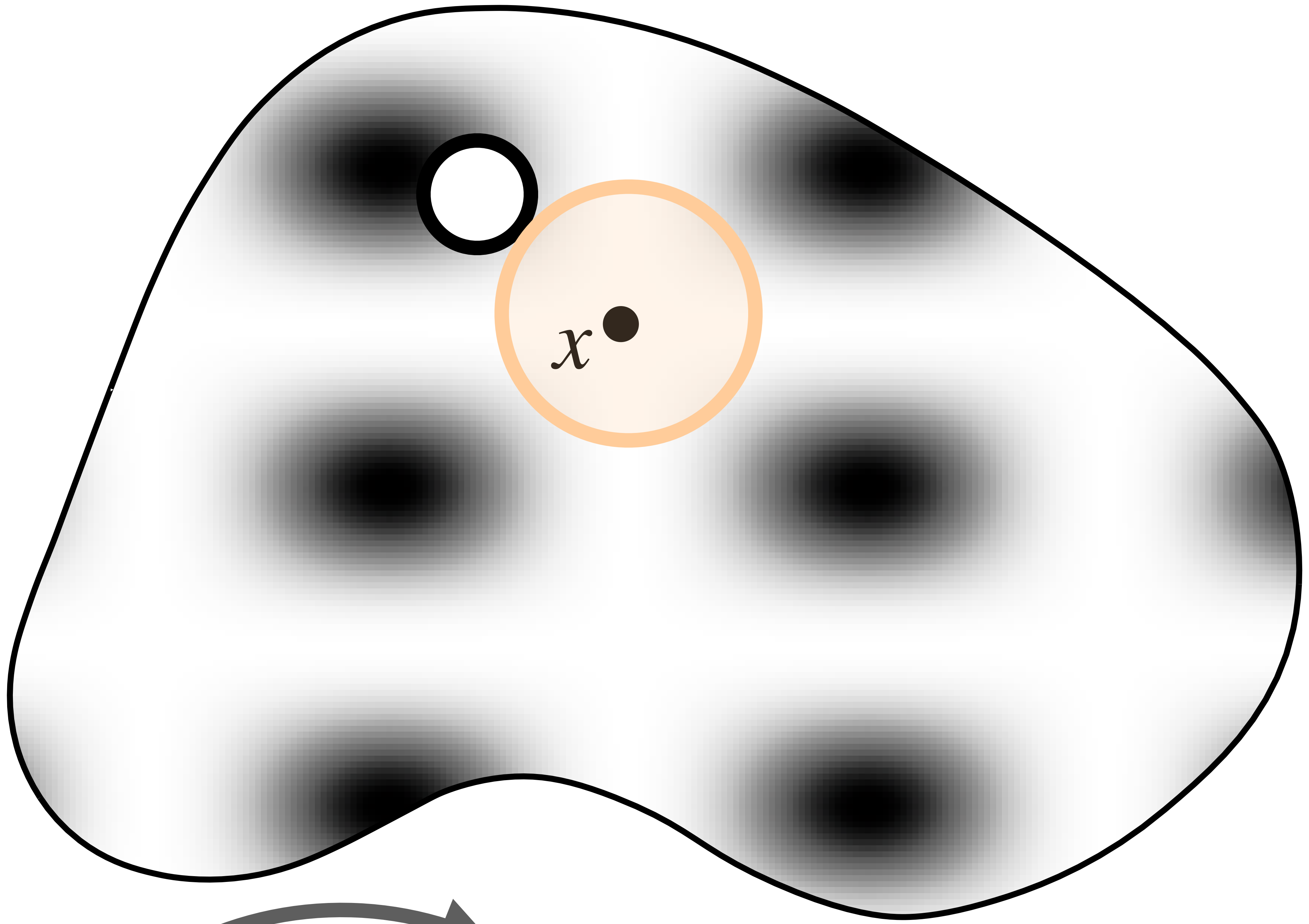
free-flight distribution (volume rendering)

$$p^{ff}_{x,\omega}(t) = \exp\left(-\int_0^t \lambda(x+\omega s)\ \mathrm{d}s\right)\ \lambda(x+\omega t)$$

spherical-contact distribution (volumetric WoS)

$$p^{cp}_{x}(y) = \exp\left(-\int_{B(x,R)} \lambda(z)\ \mathrm{d}z\right)\ \lambda(y)$$

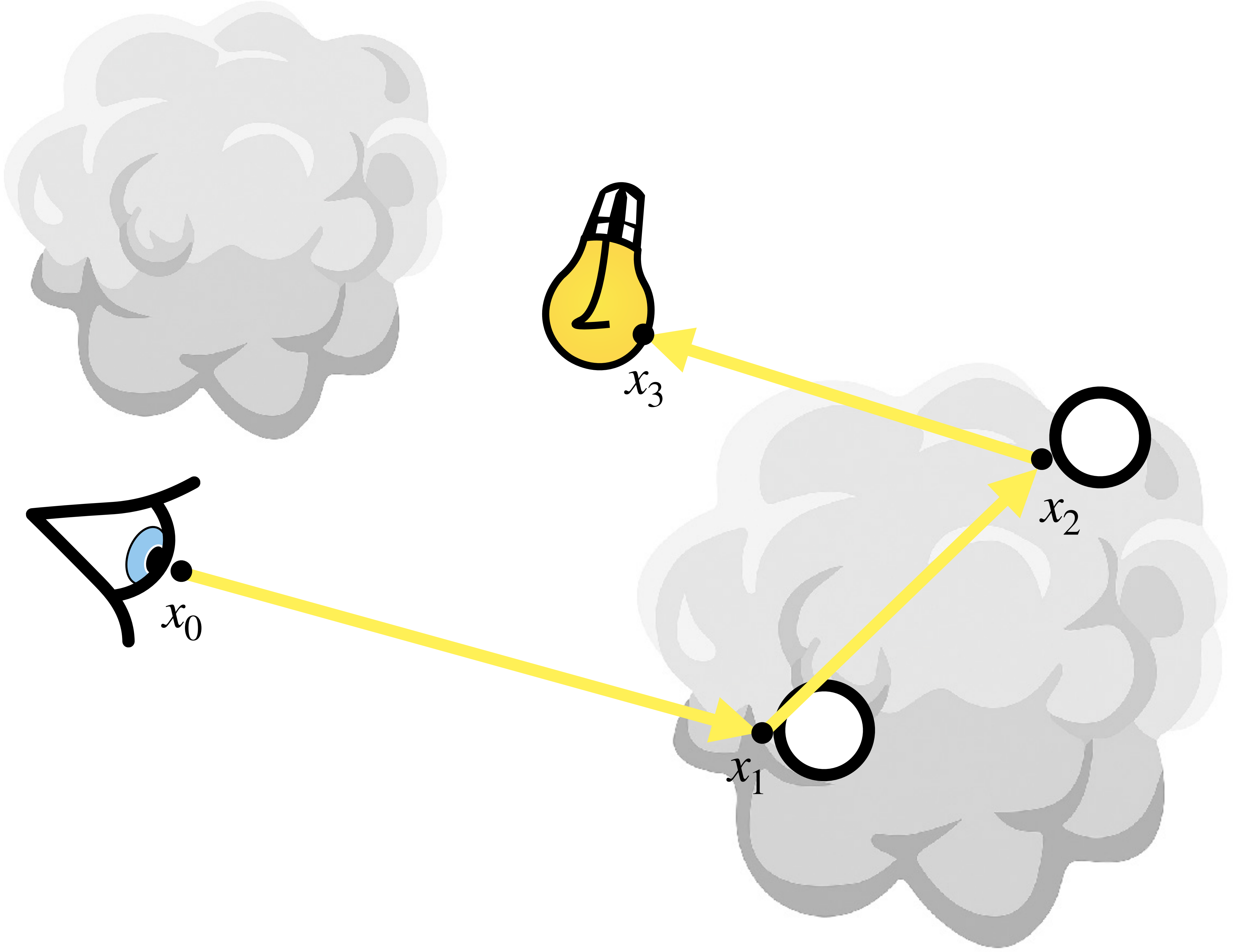


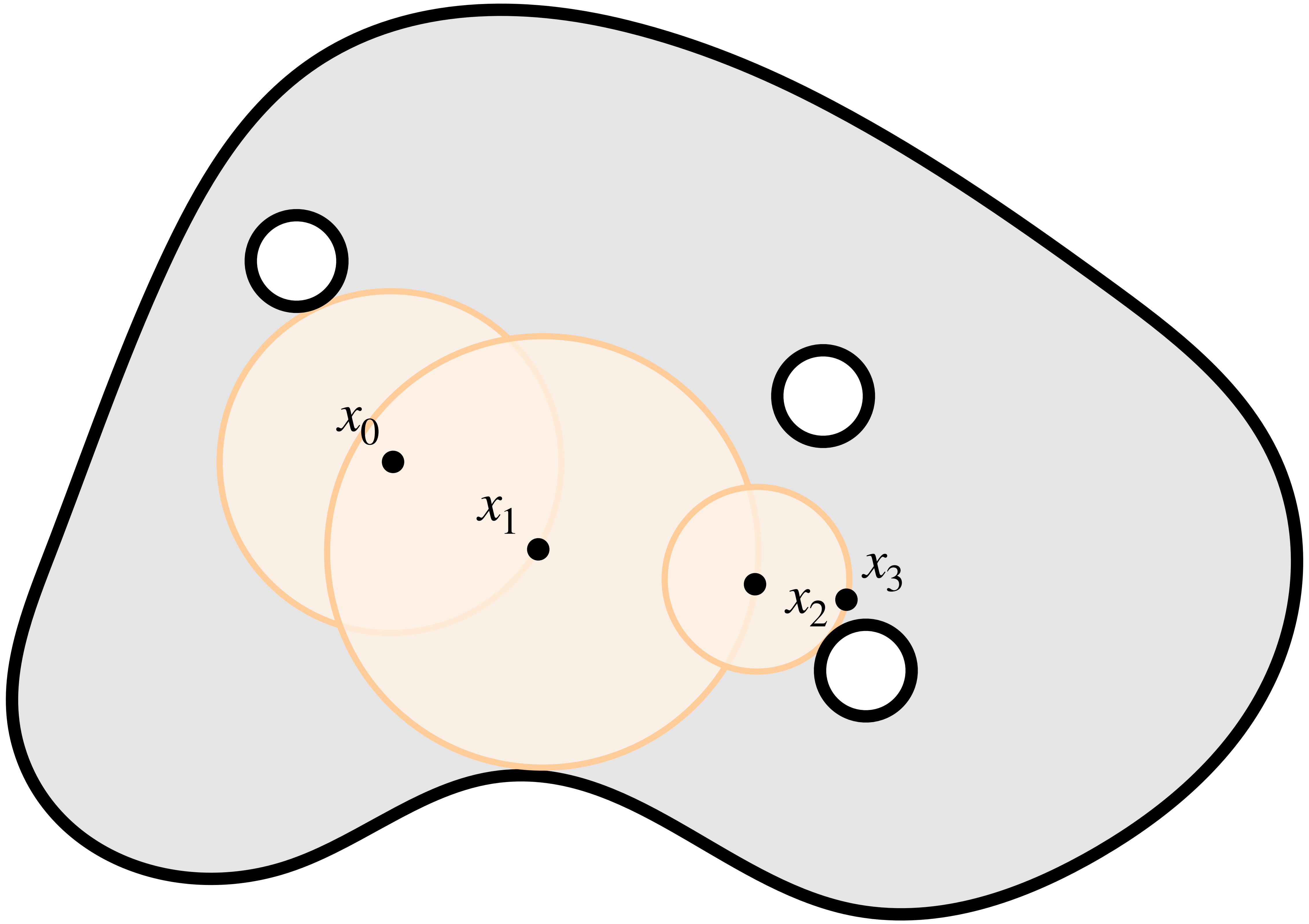


$\lambda(x)$ **expected particles per unit volume** $\lambda(x)$

Memory: rendering vs walk on spheres

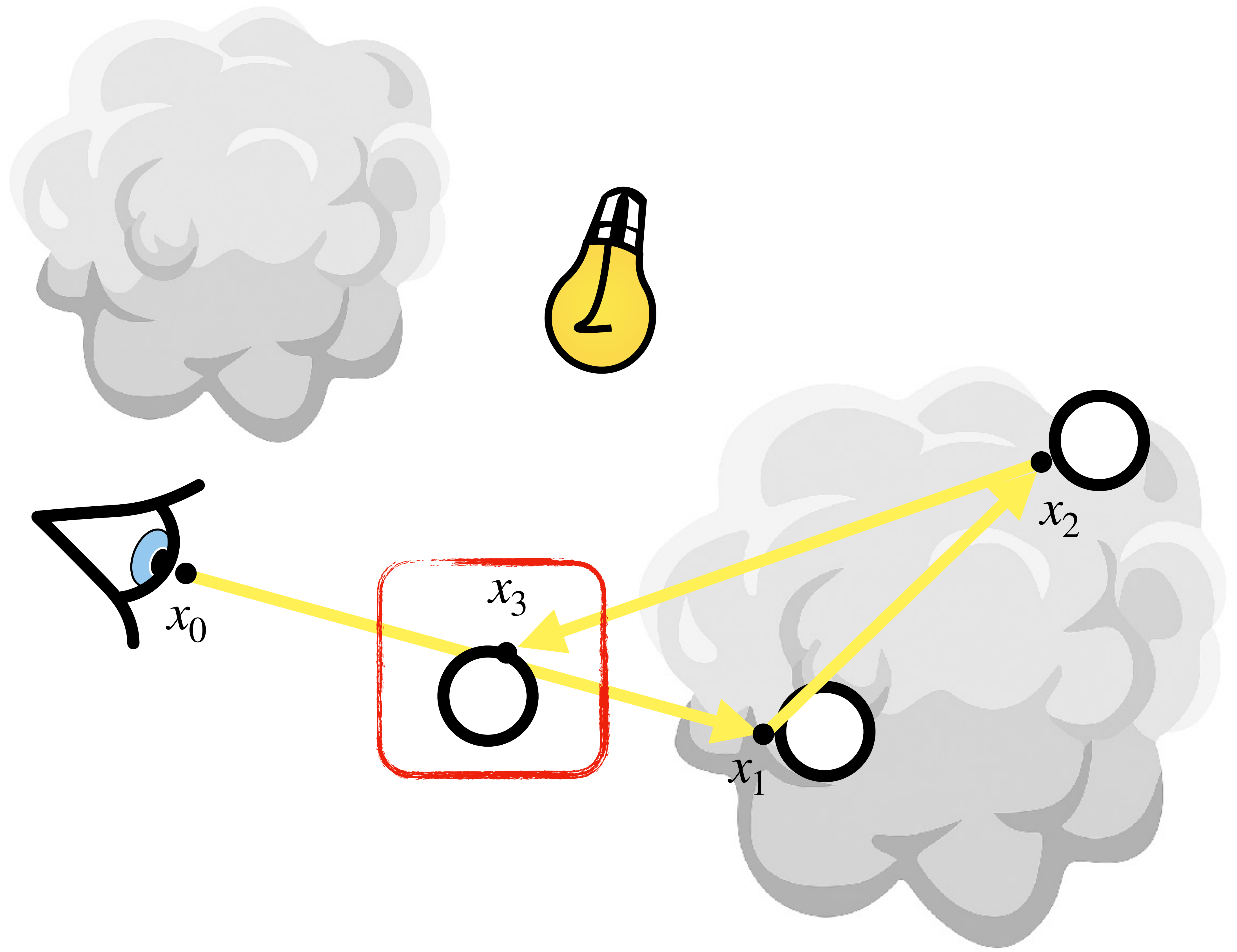
Sampled particles must be consistent across entire walk.

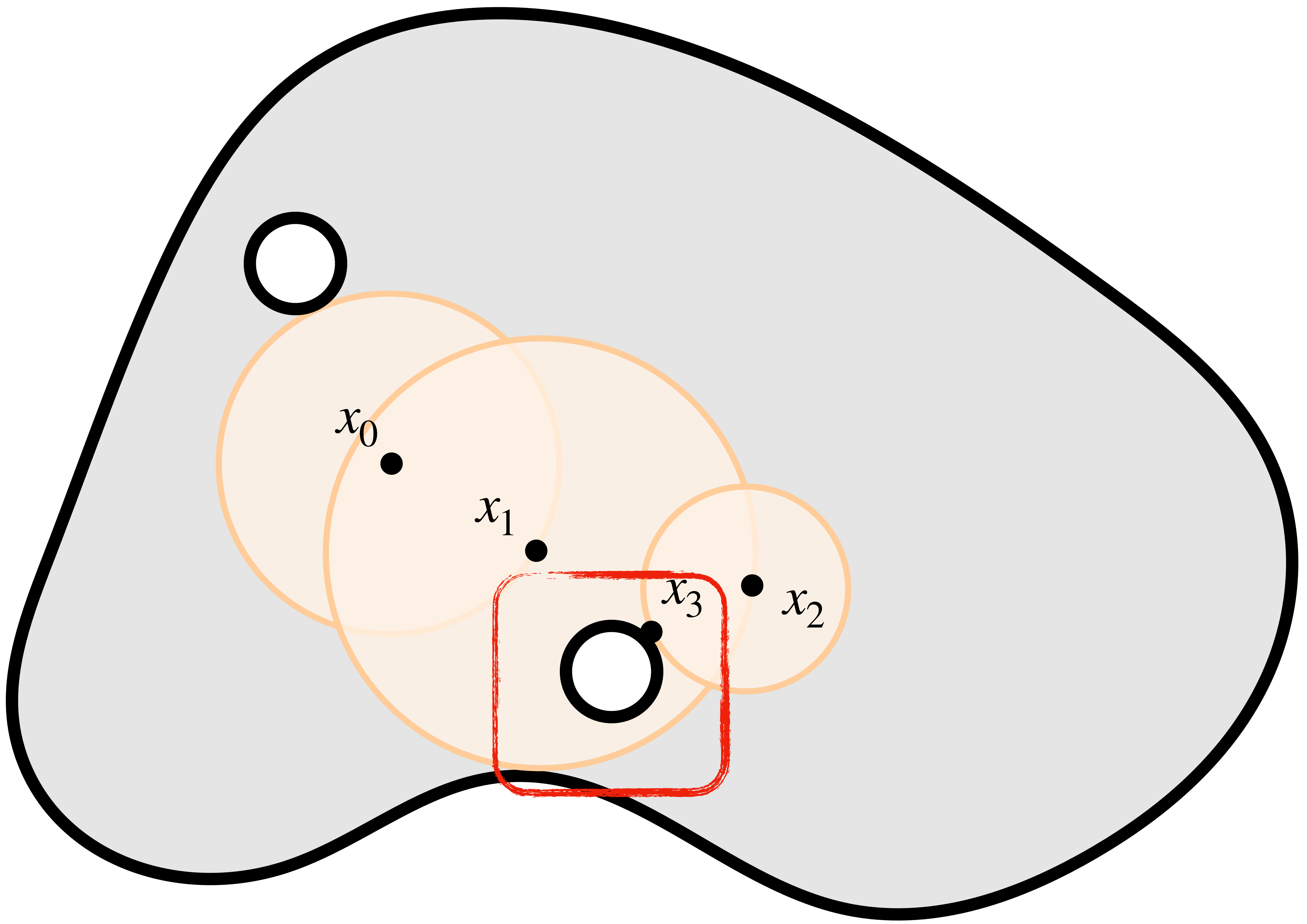




Memory: rendering vs walk on spheres

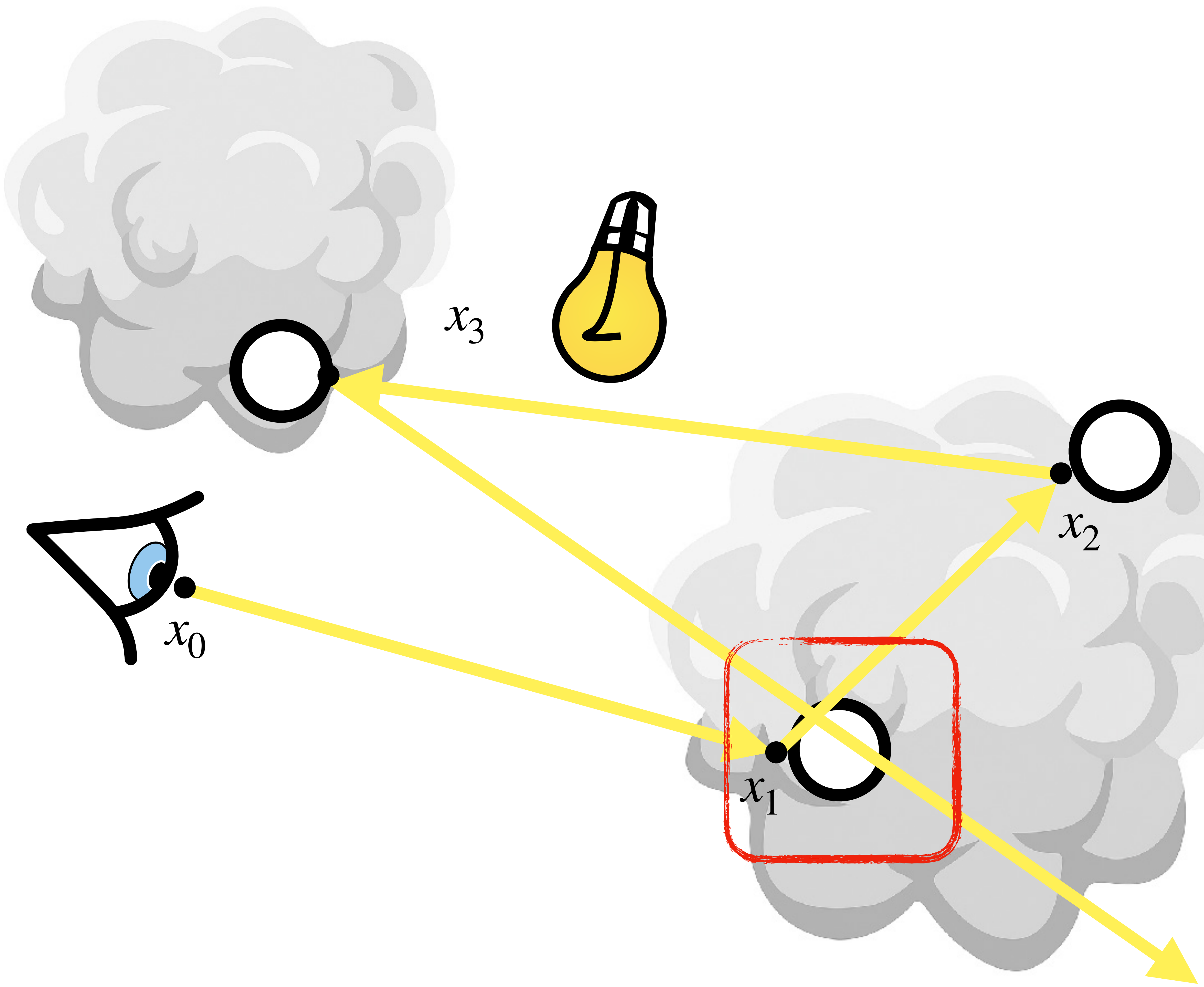
Particles cannot obstruct already sampled path / walk.

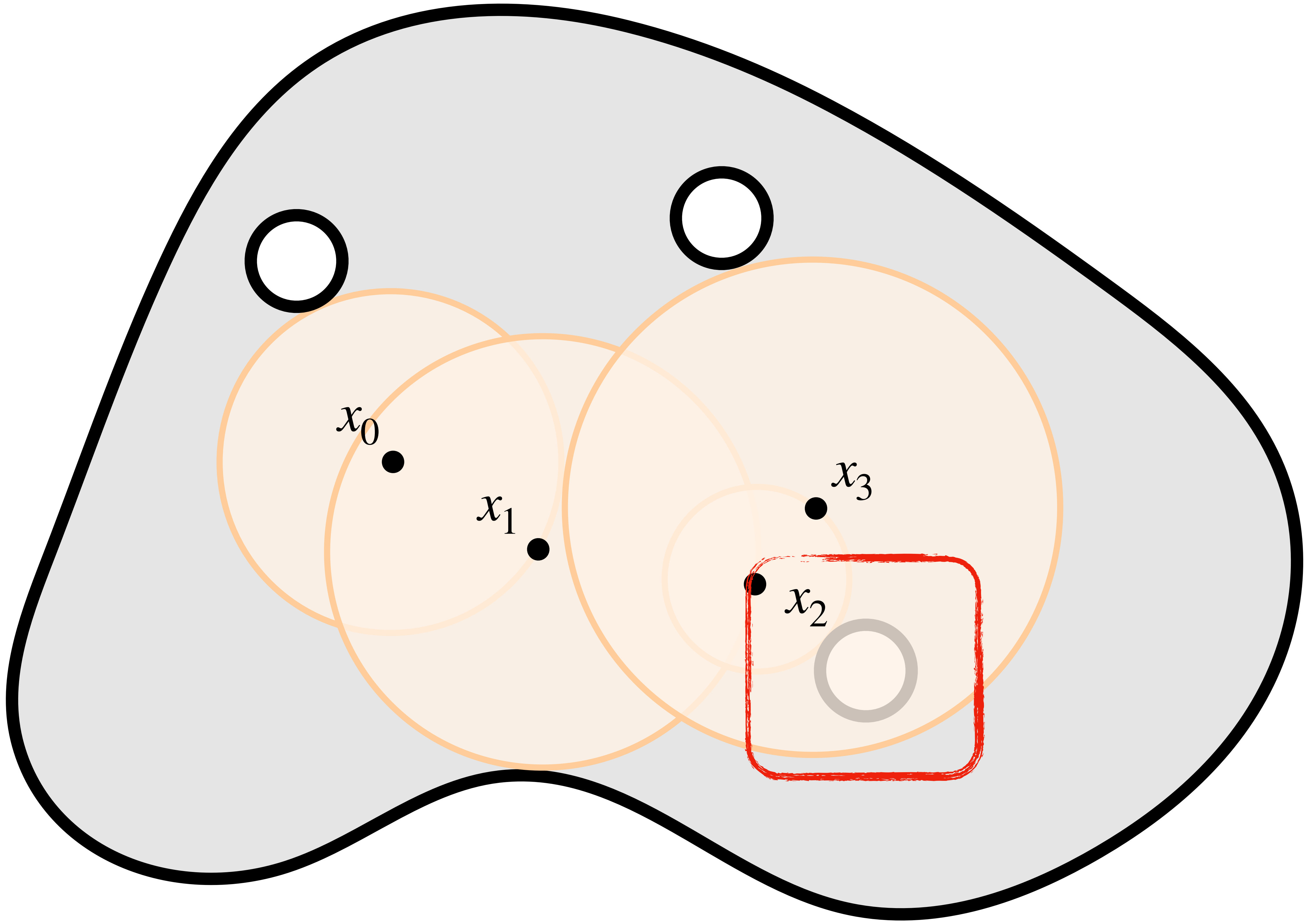




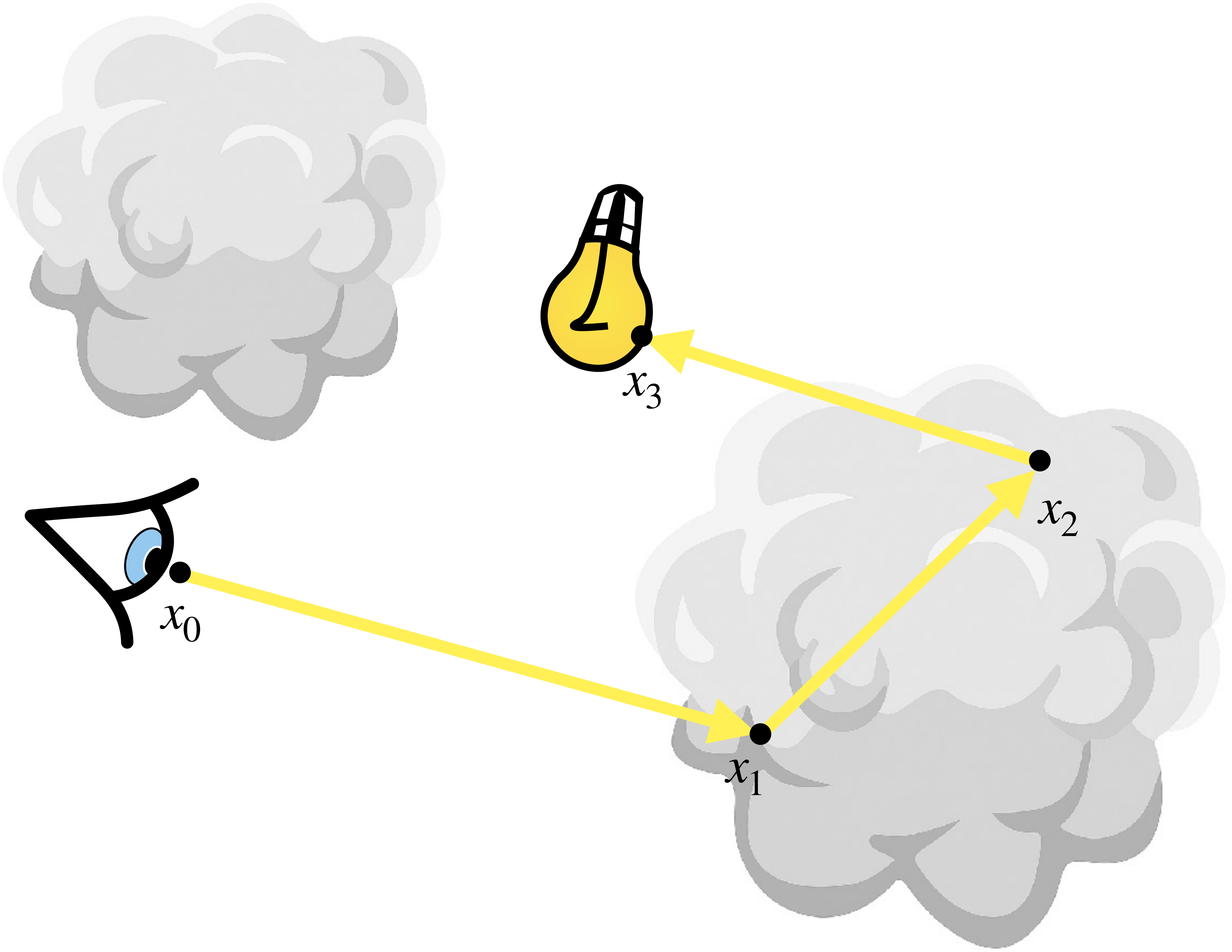
Memory: rendering vs walk on spheres

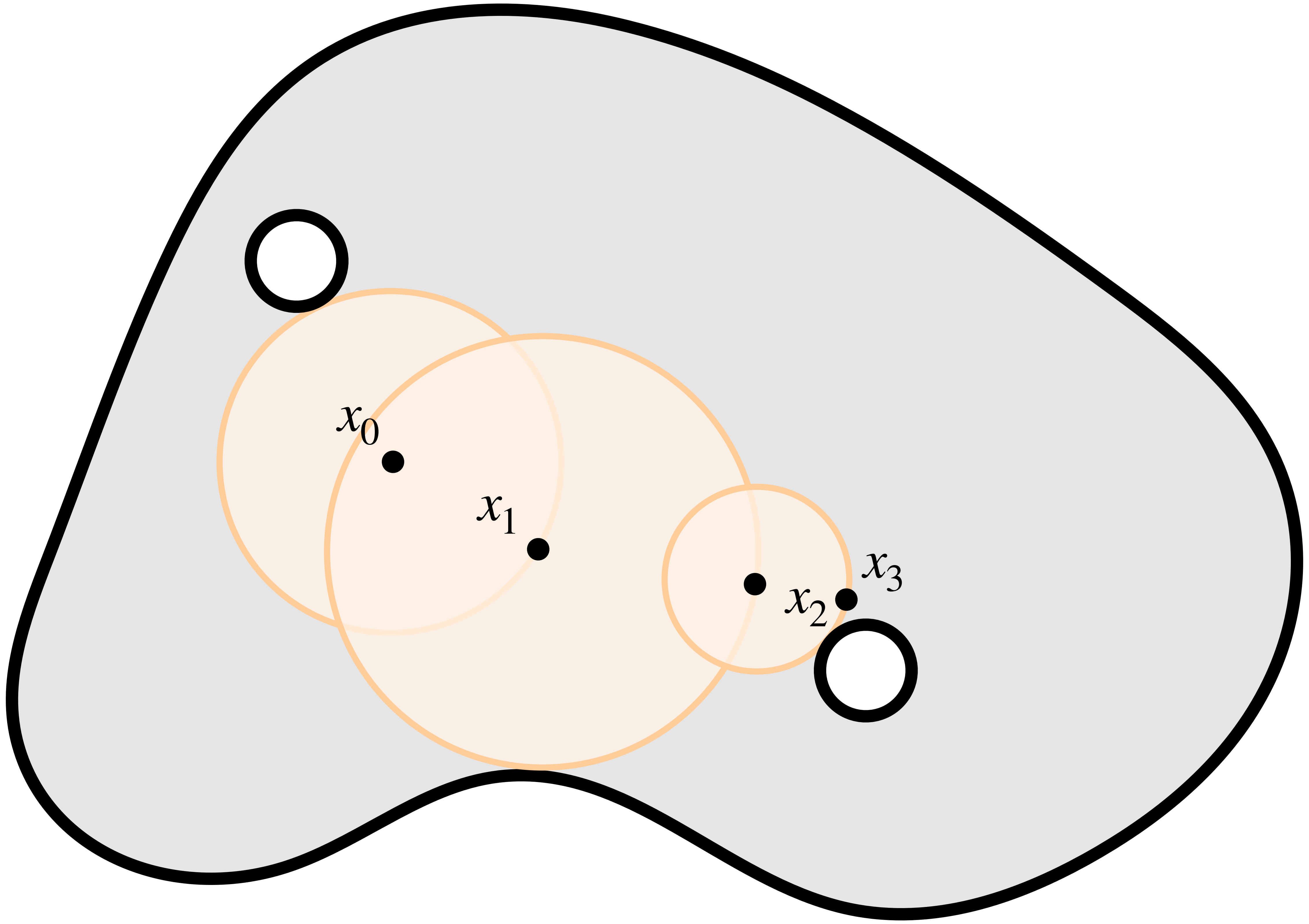
Walk must account for previously sampled particles.





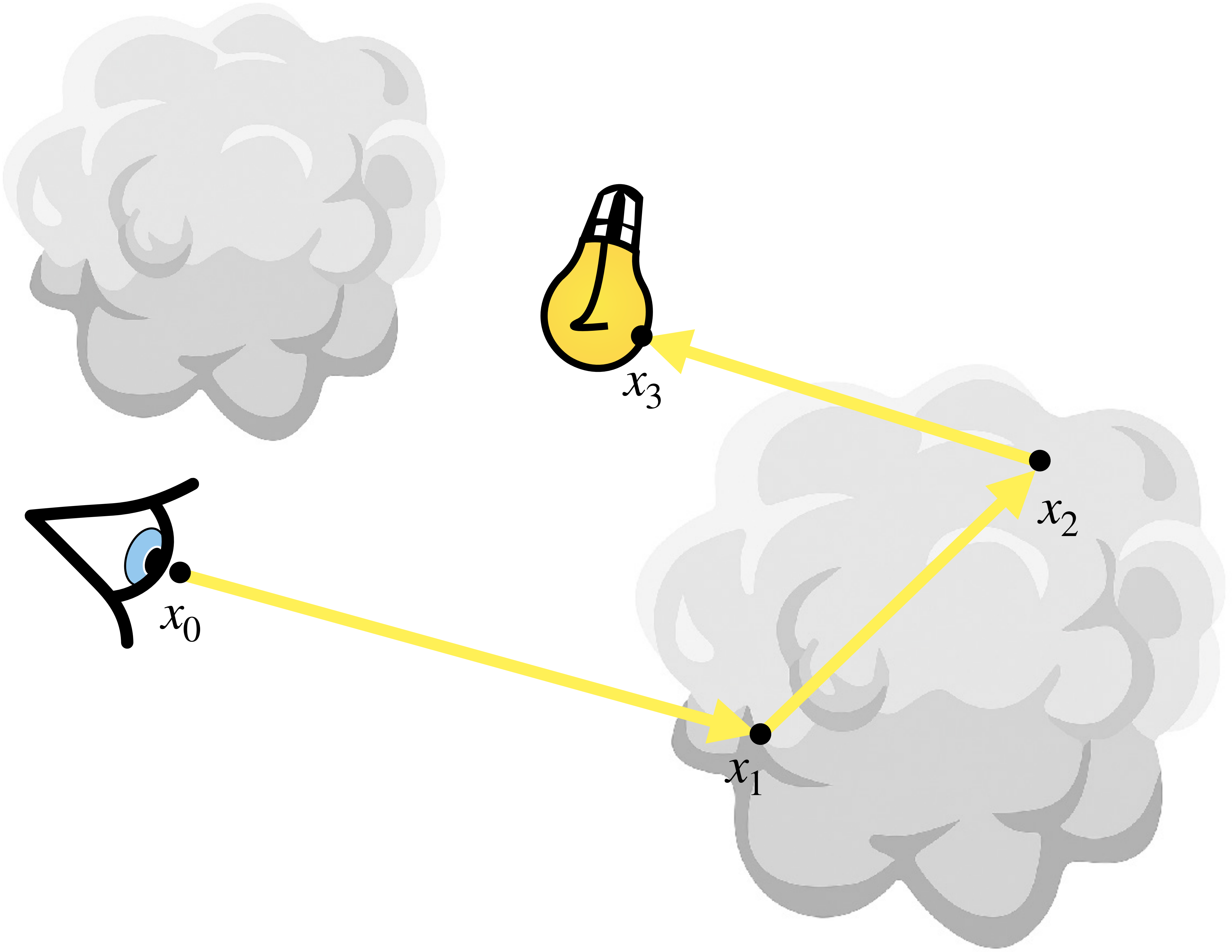
Memory rendering versus walk on spheres

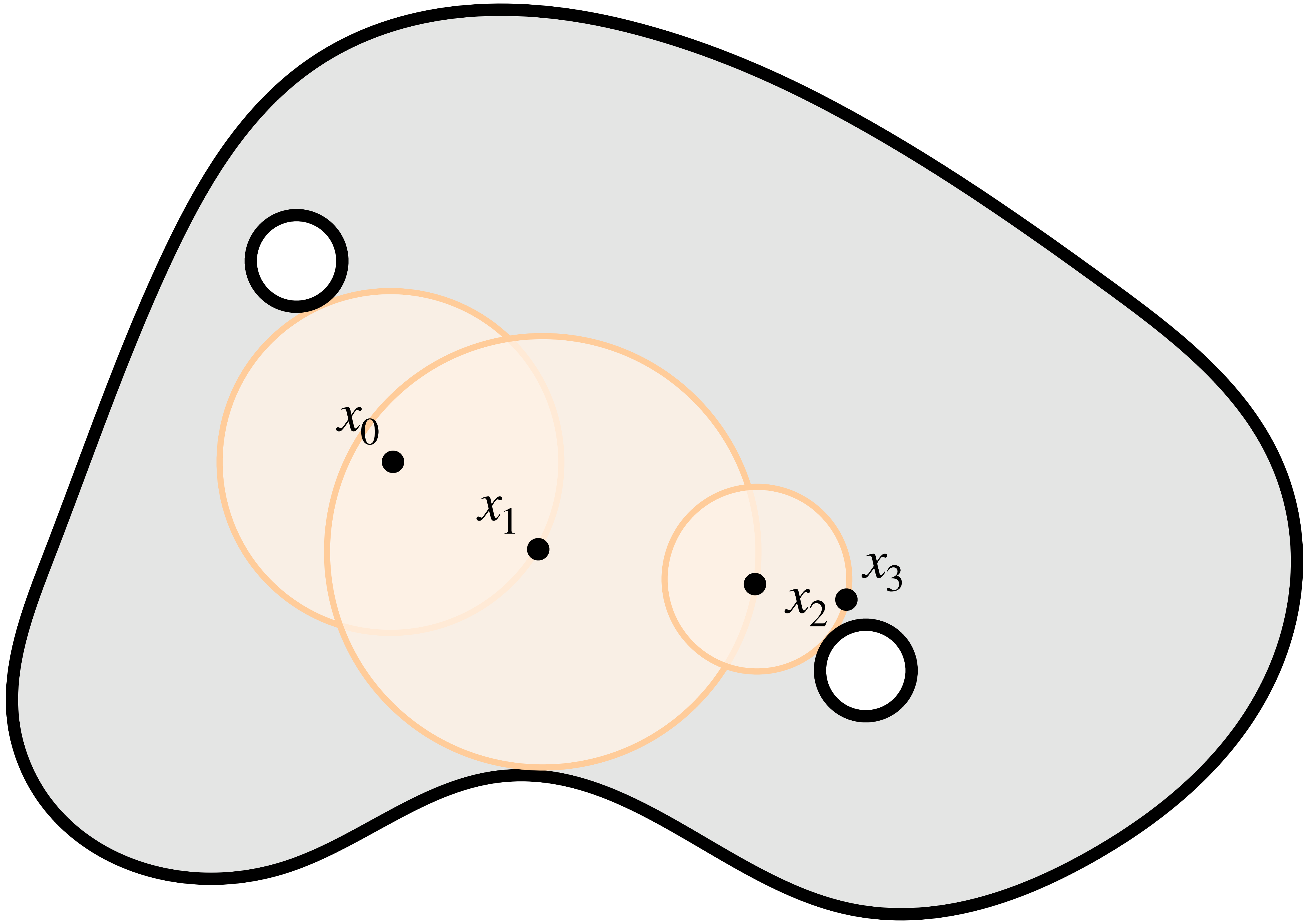




Memory rendering versus walk on spheres

Volume rendering assumes "**memoryless-ness**" since inconsistent events are rare

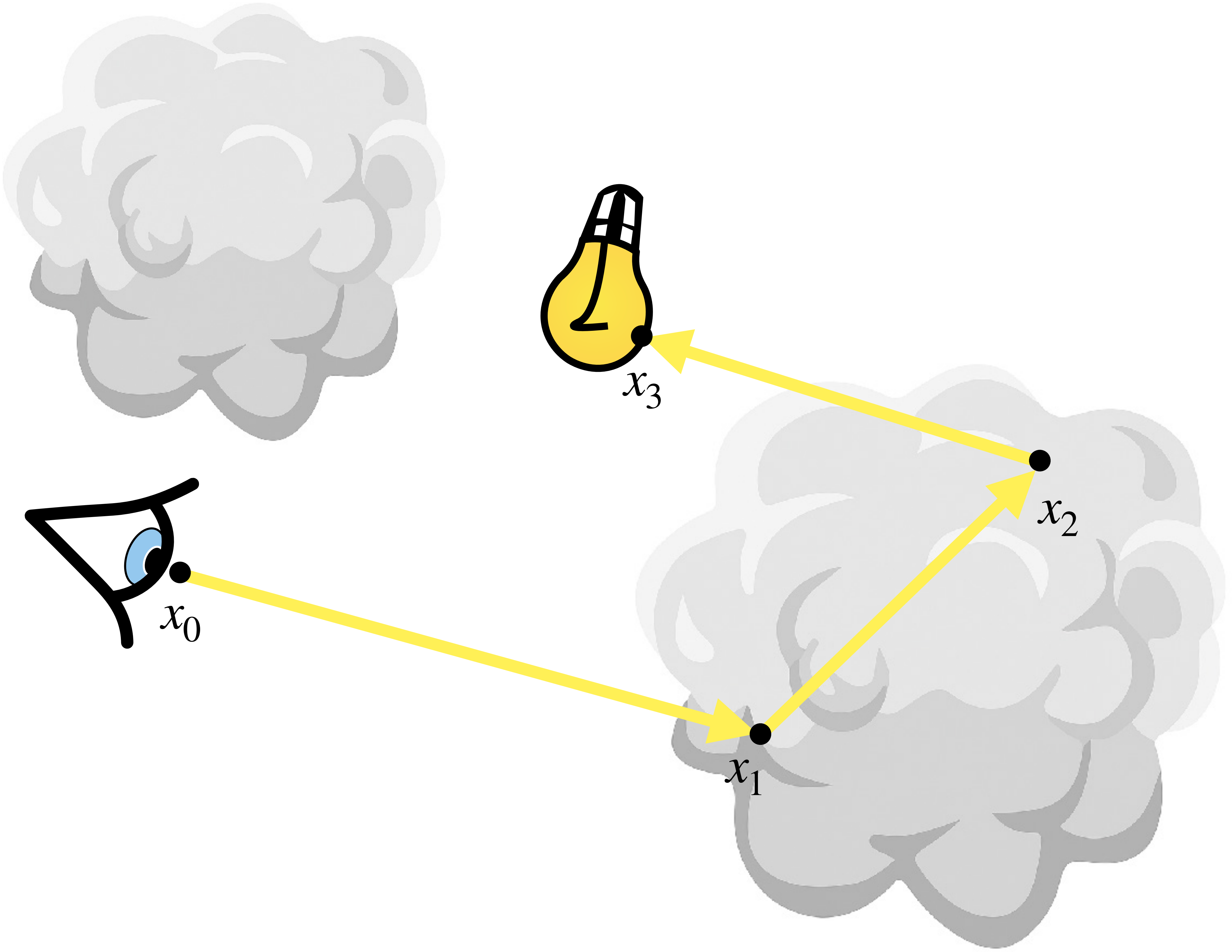


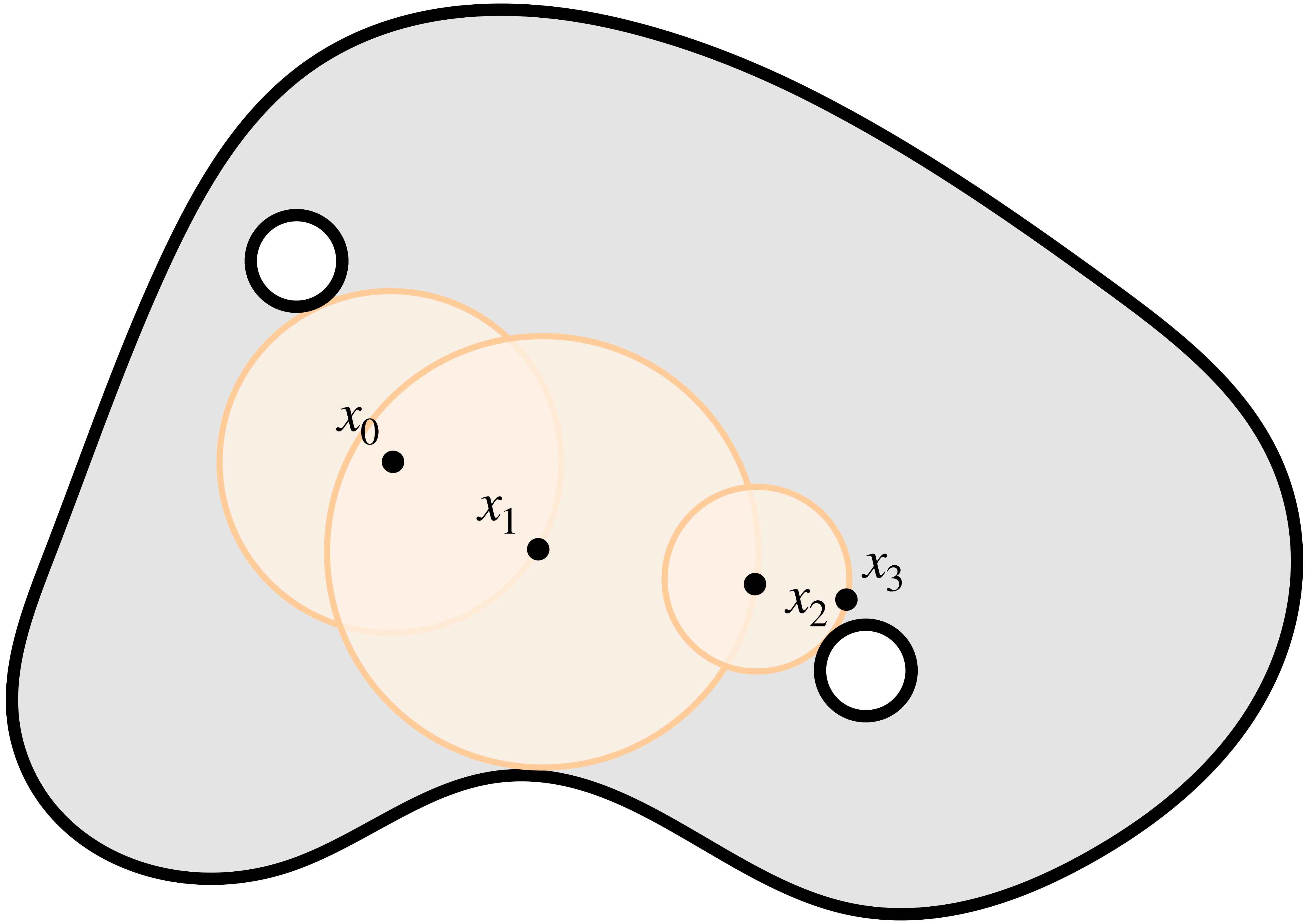


Memory rendering versus walk on spheres

Volume rendering assumes "**memoryless-ness**" since inconsistent events are rare

Volume walk on spheres uses **memory** since inconsistent events are likely

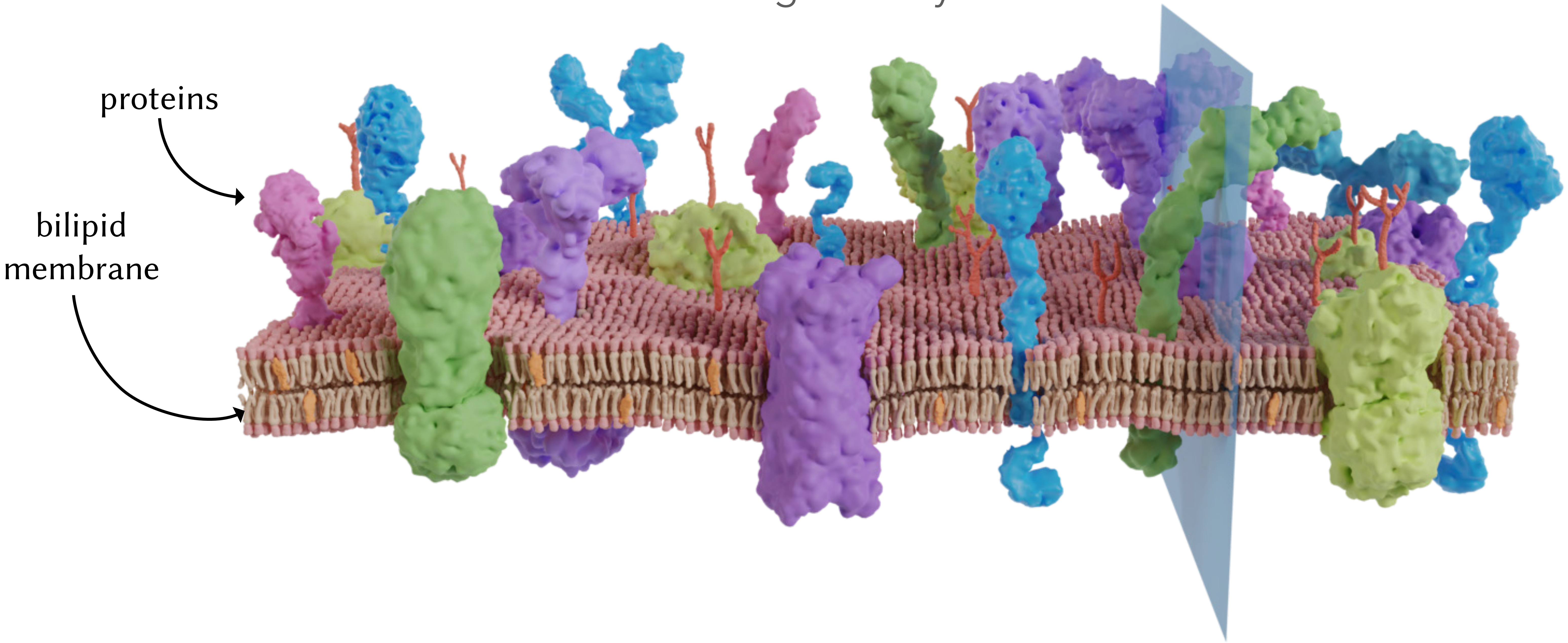




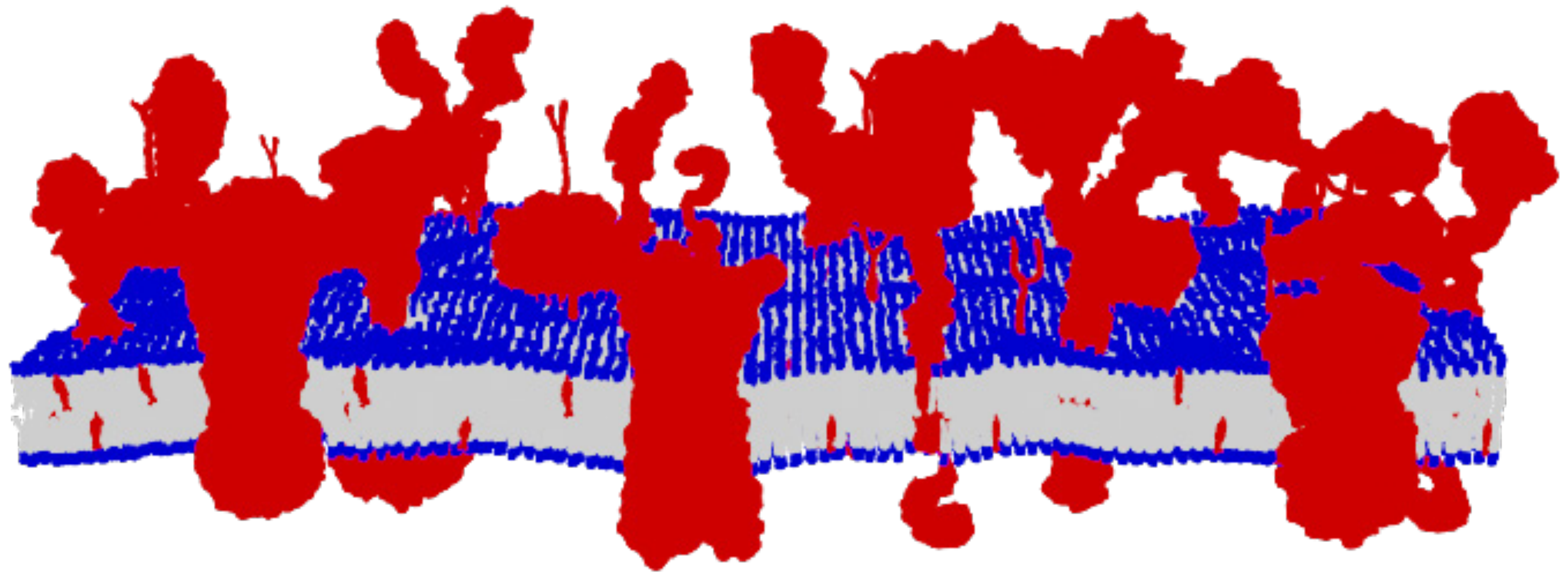
Evaluation

Electrostatic potential near biological membrane

surface geometry



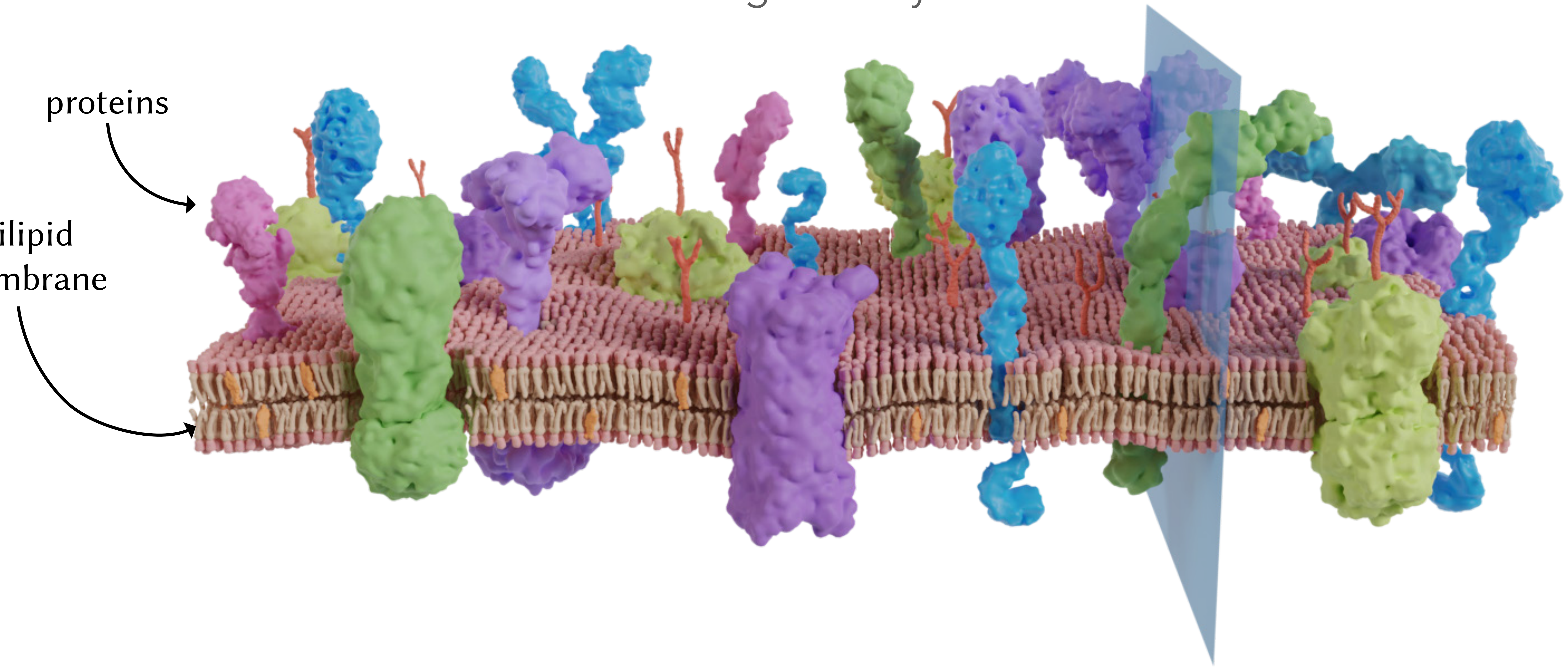
boundary condition



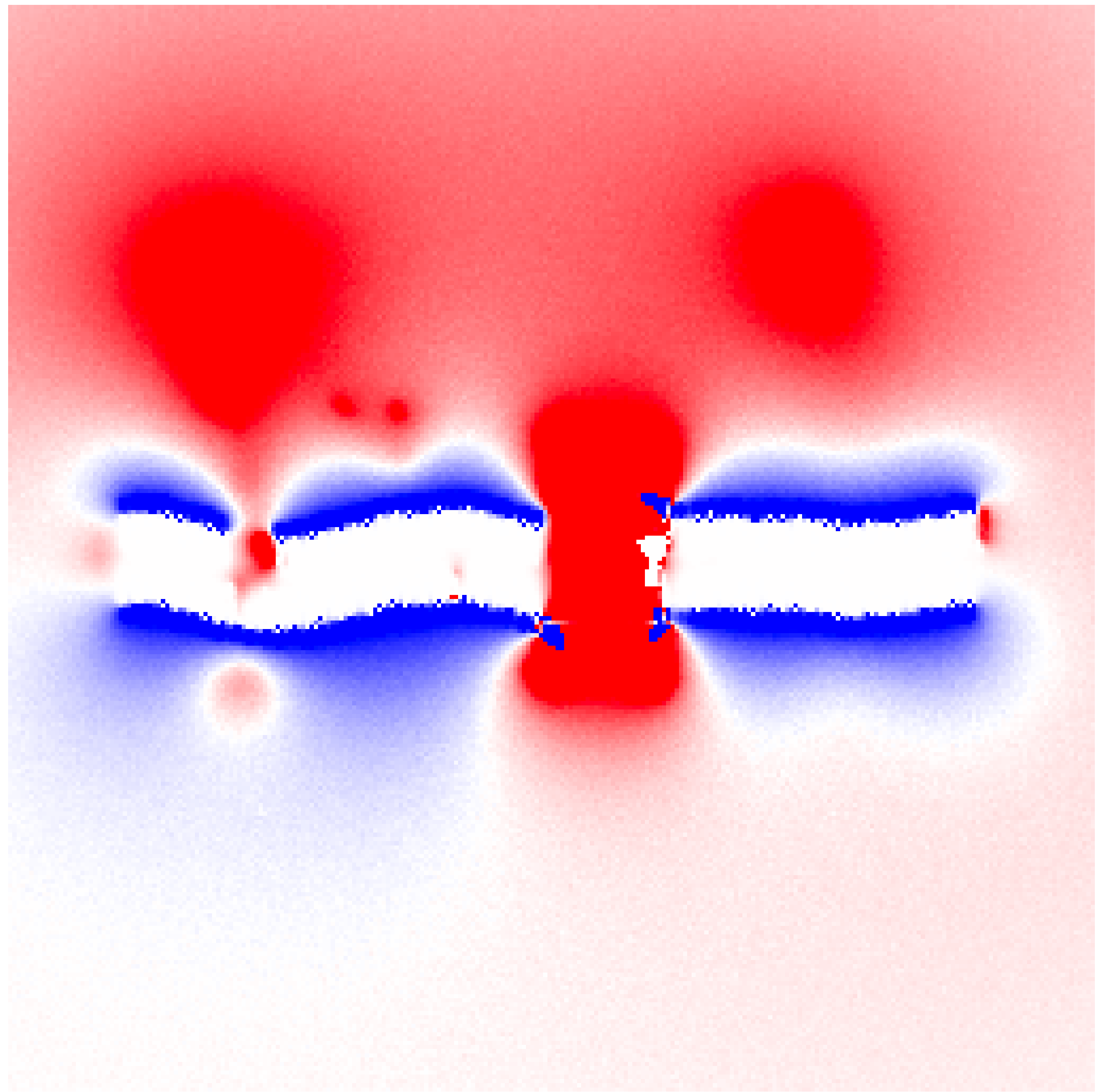
electric potential (V) -1 1

Electrostatic potential near biological membrane

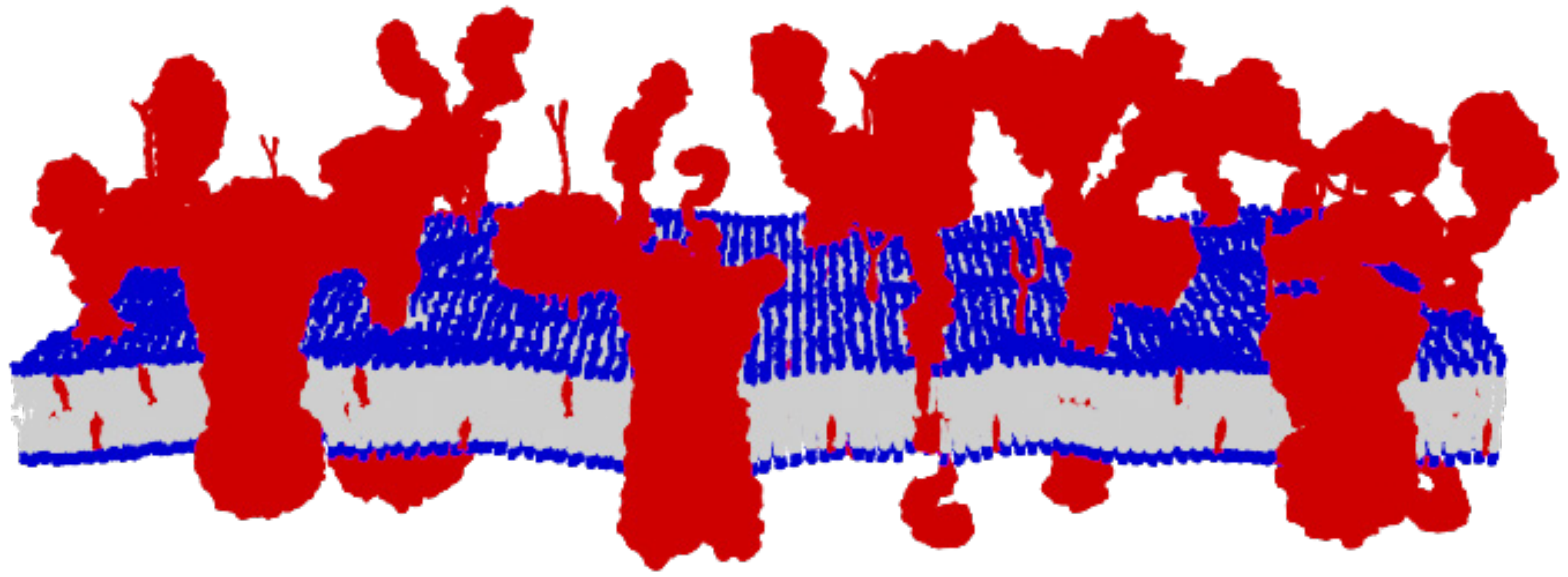
surface geometry



WoS (w/out particles)



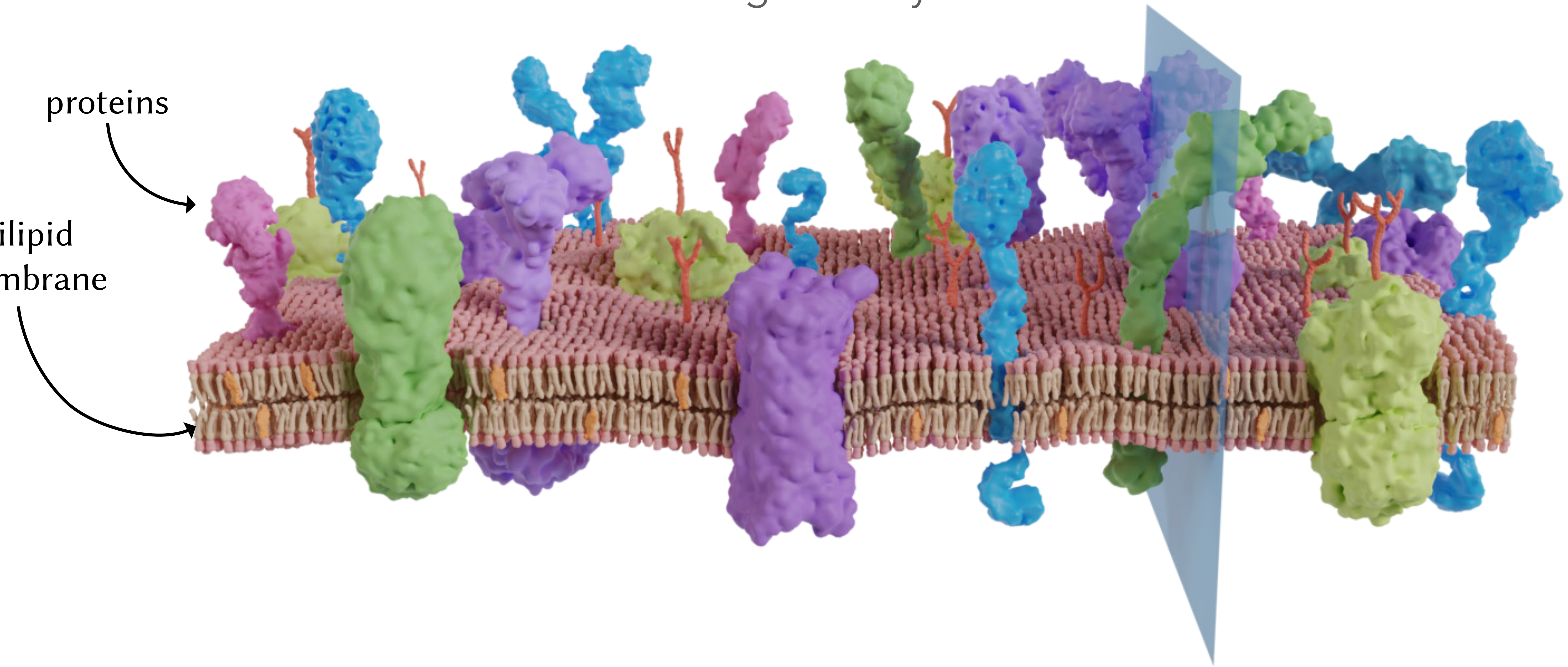
boundary condition

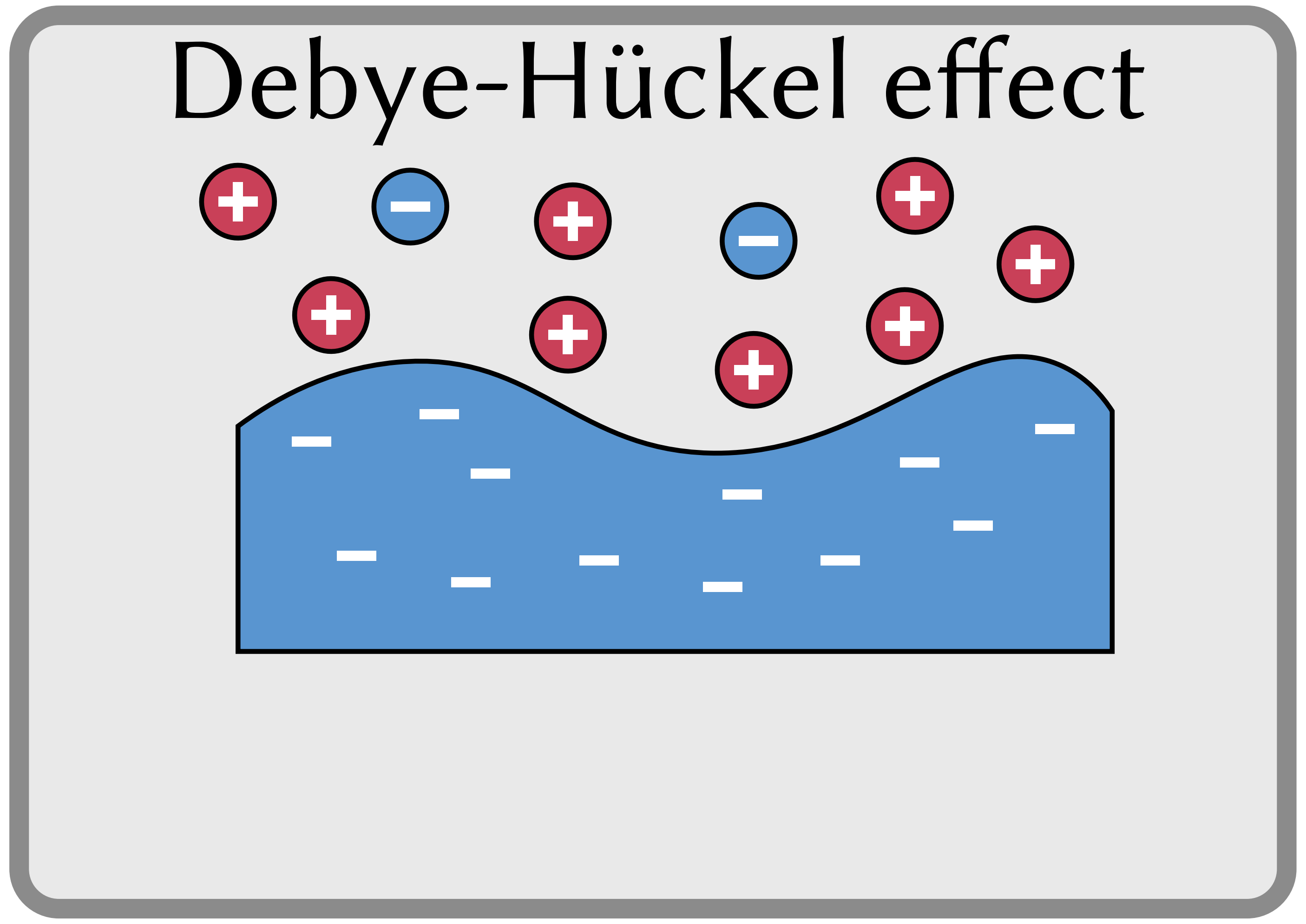


electric potential (V) -1 1

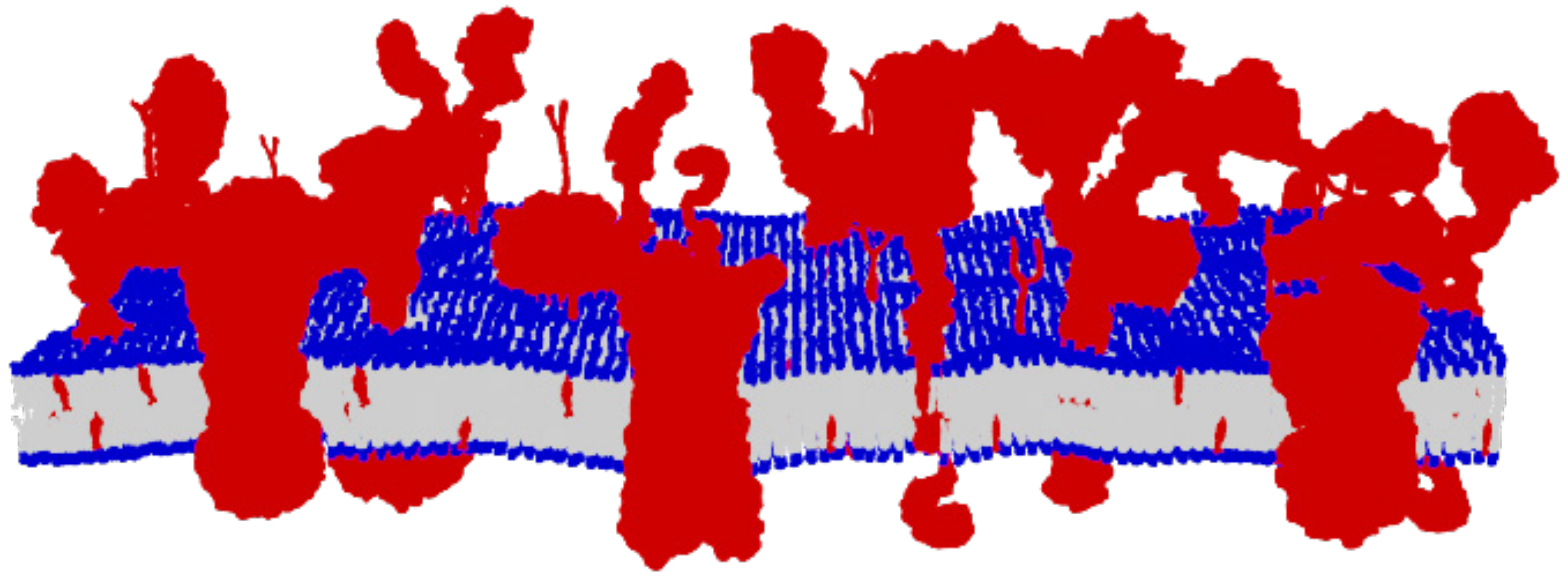
Electrostatic potential near biological membrane

surface geometry



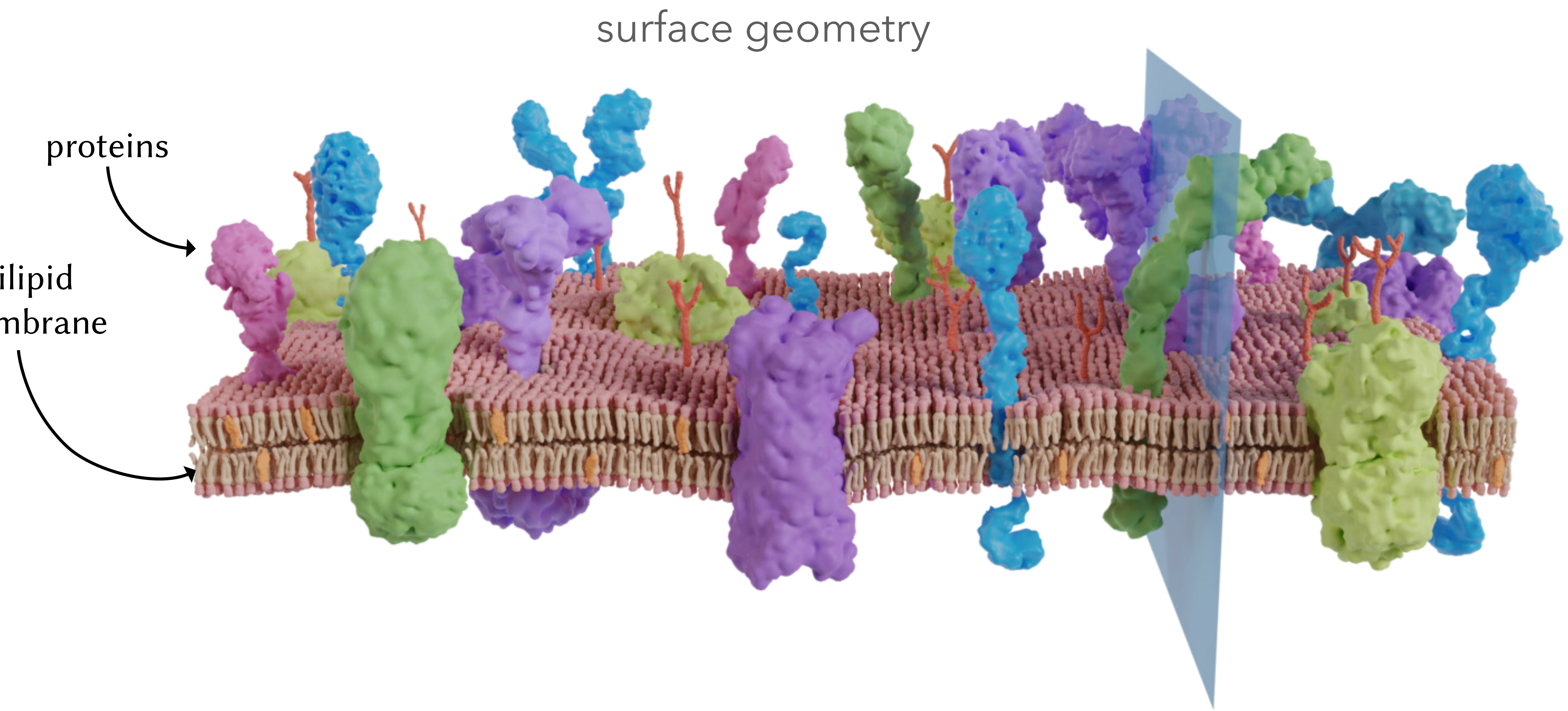


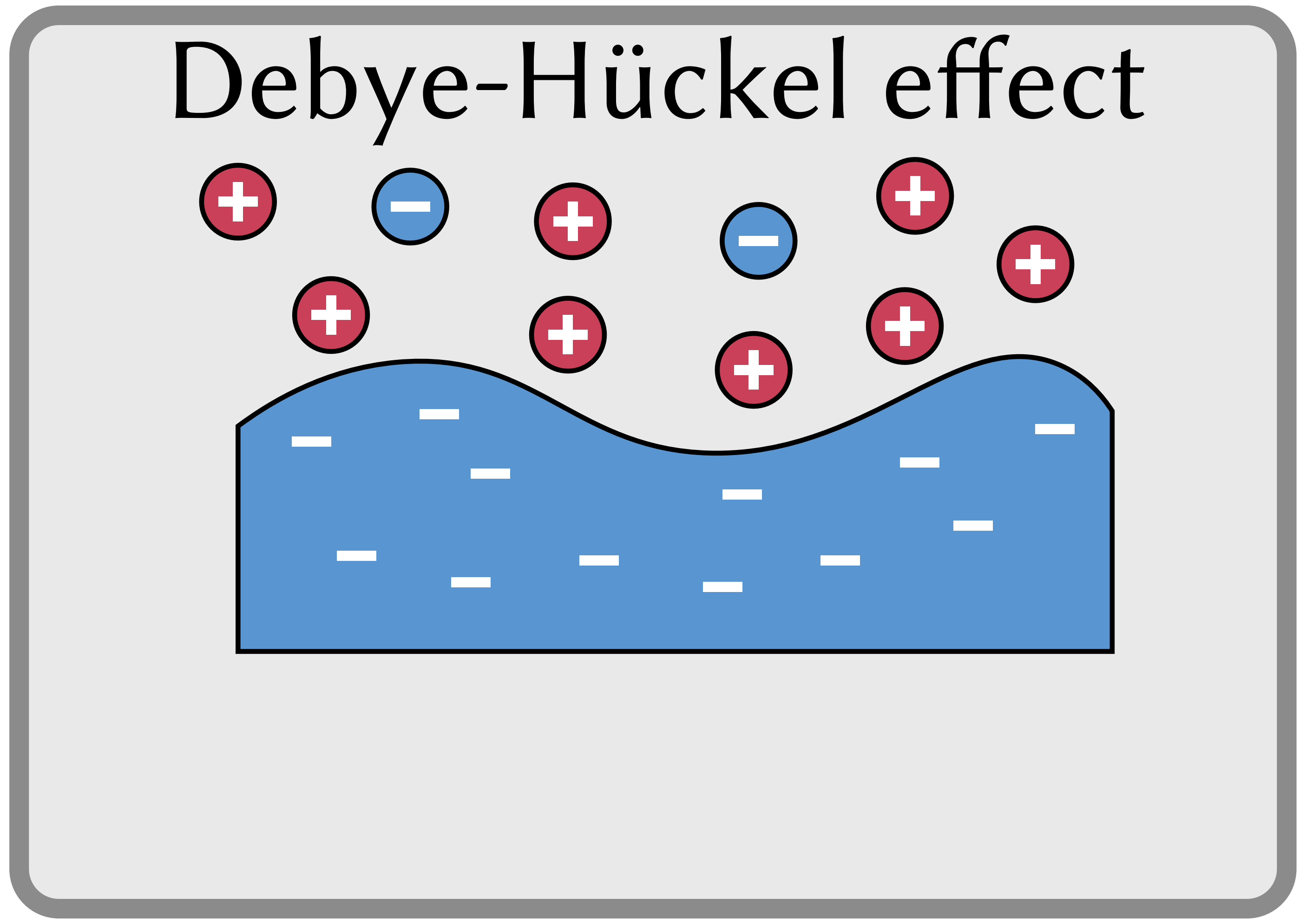
boundary condition



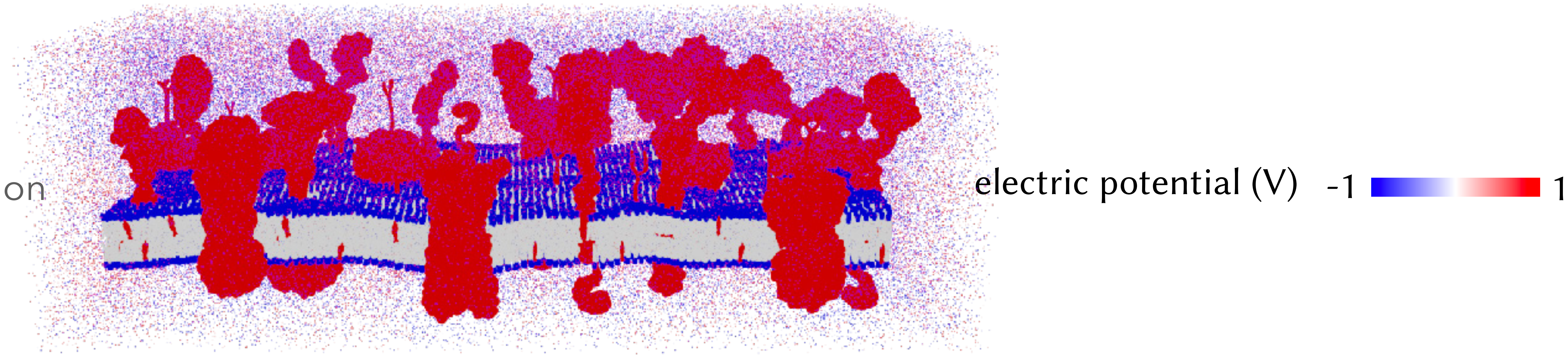
electric potential (V) -1 1

Electrostatic potential near biological membrane



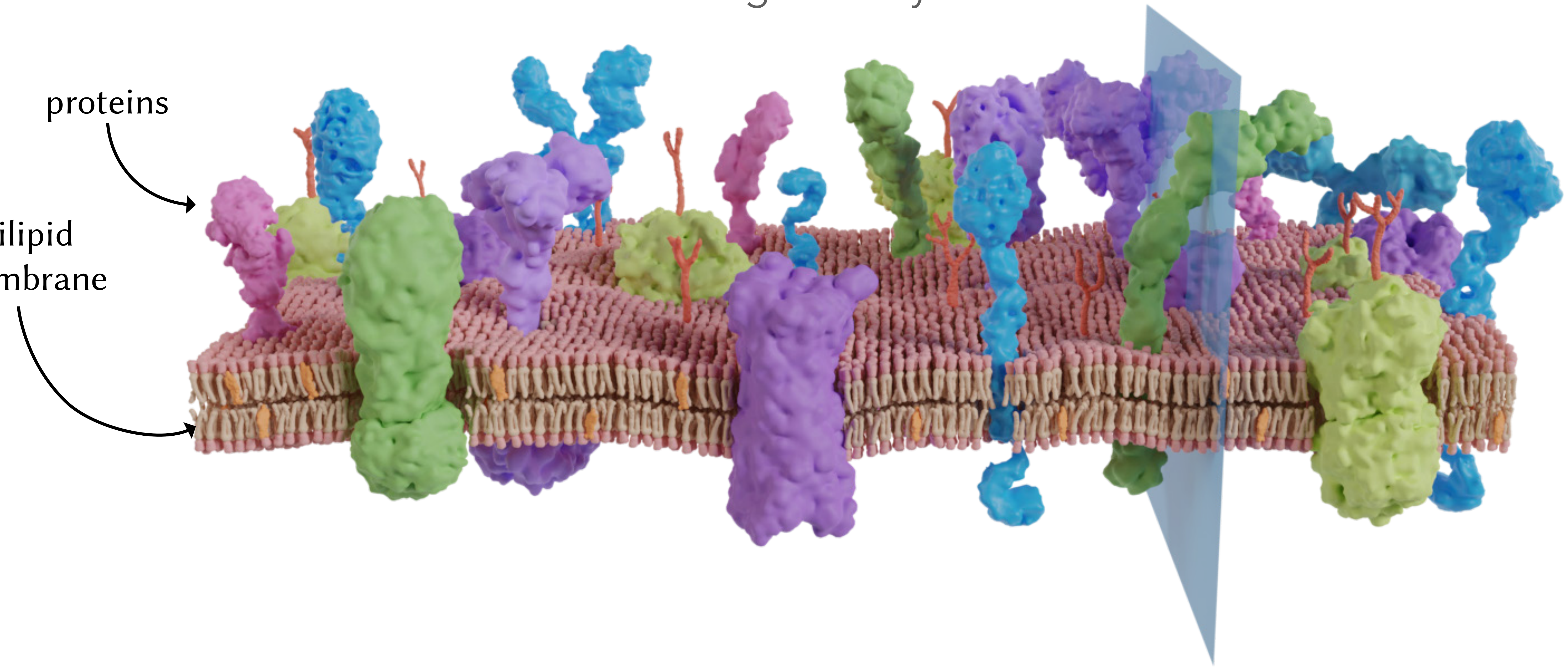


boundary condition
+ ions

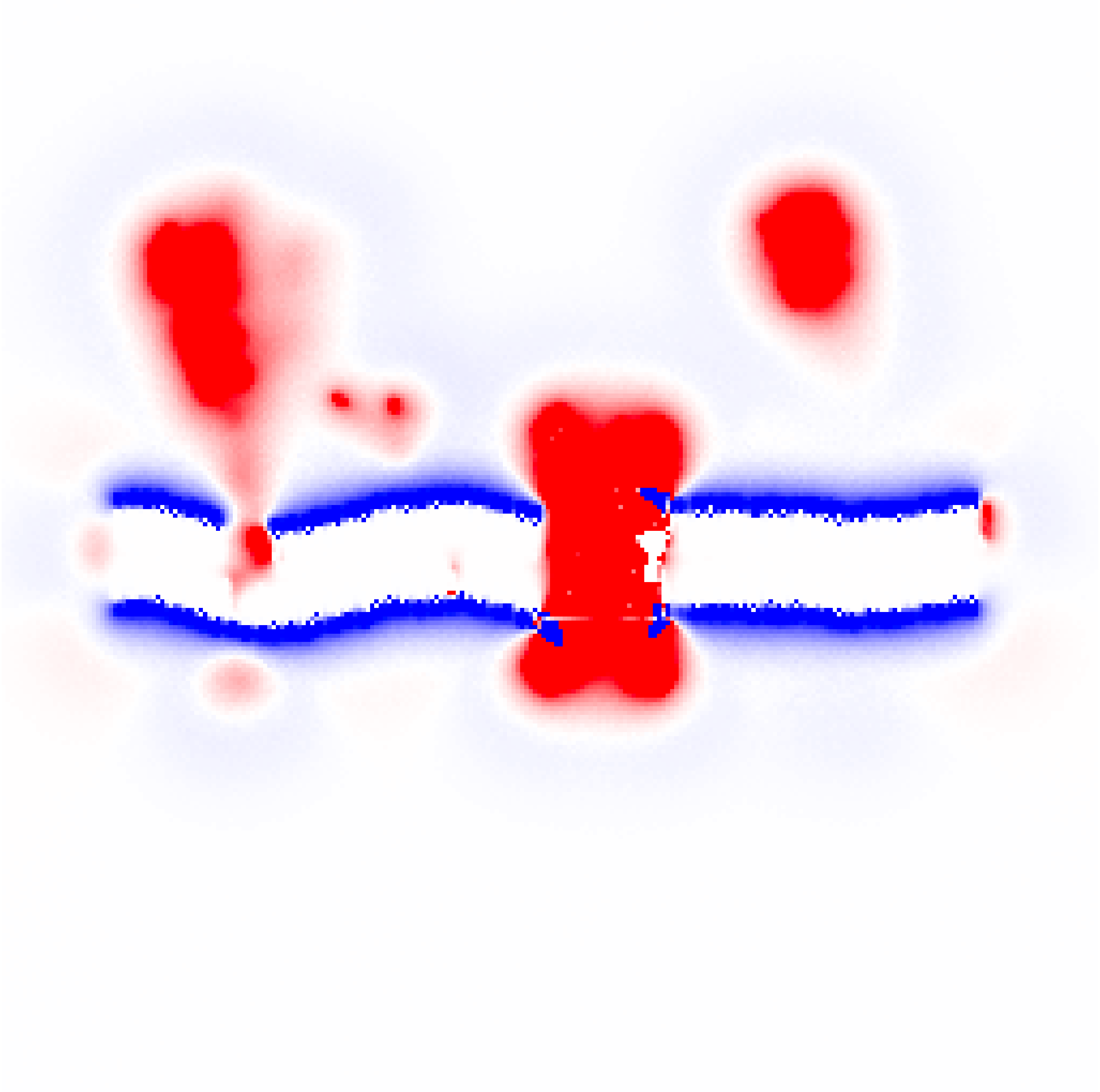


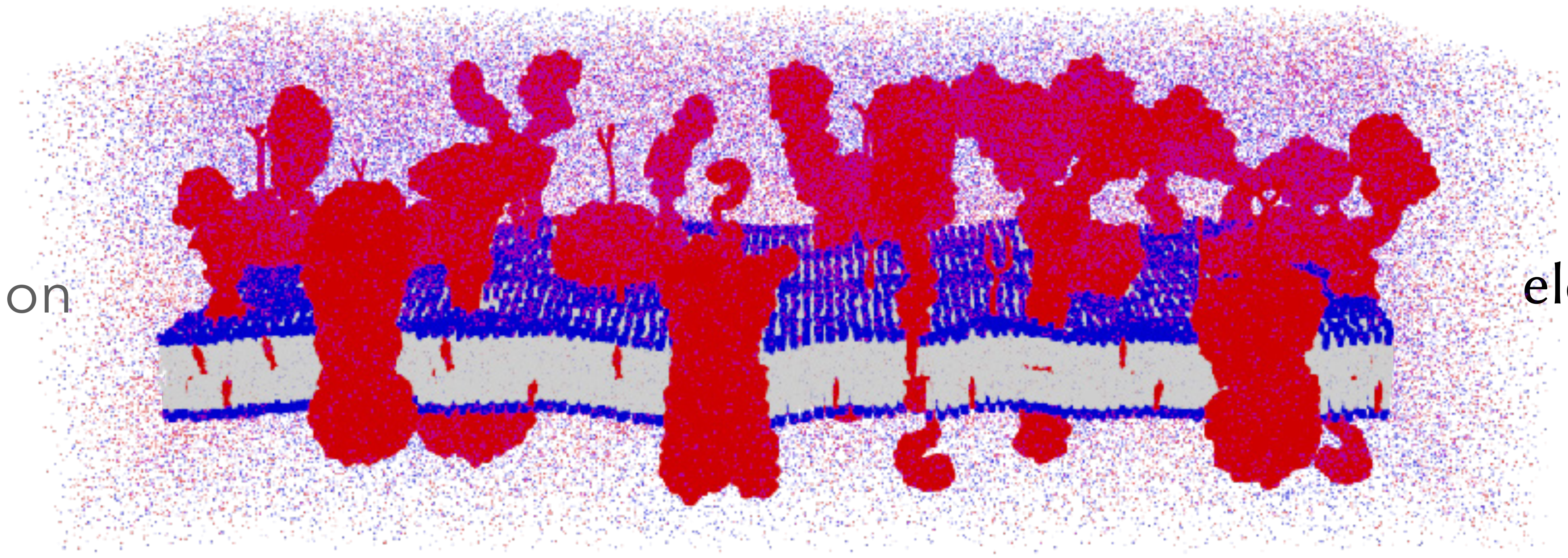
Electrostatic potential near biological membrane

surface geometry



Volumetric WoS



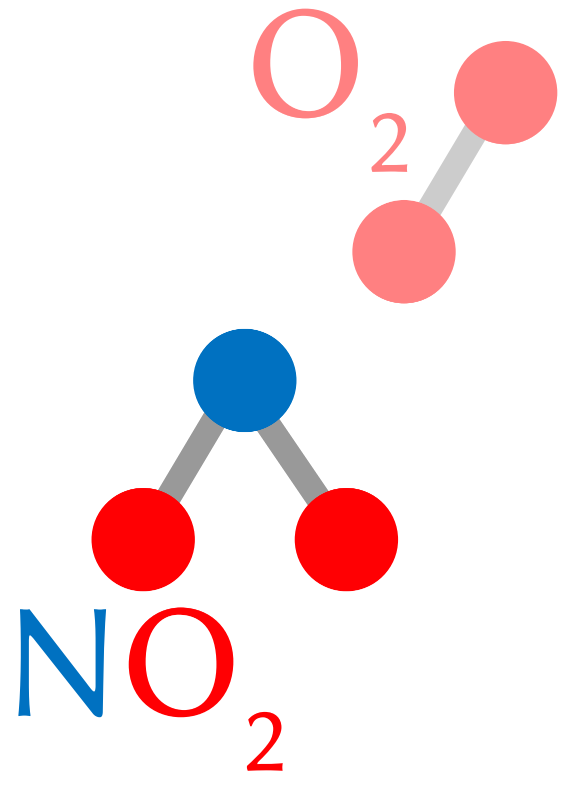


boundary condition
+ ions

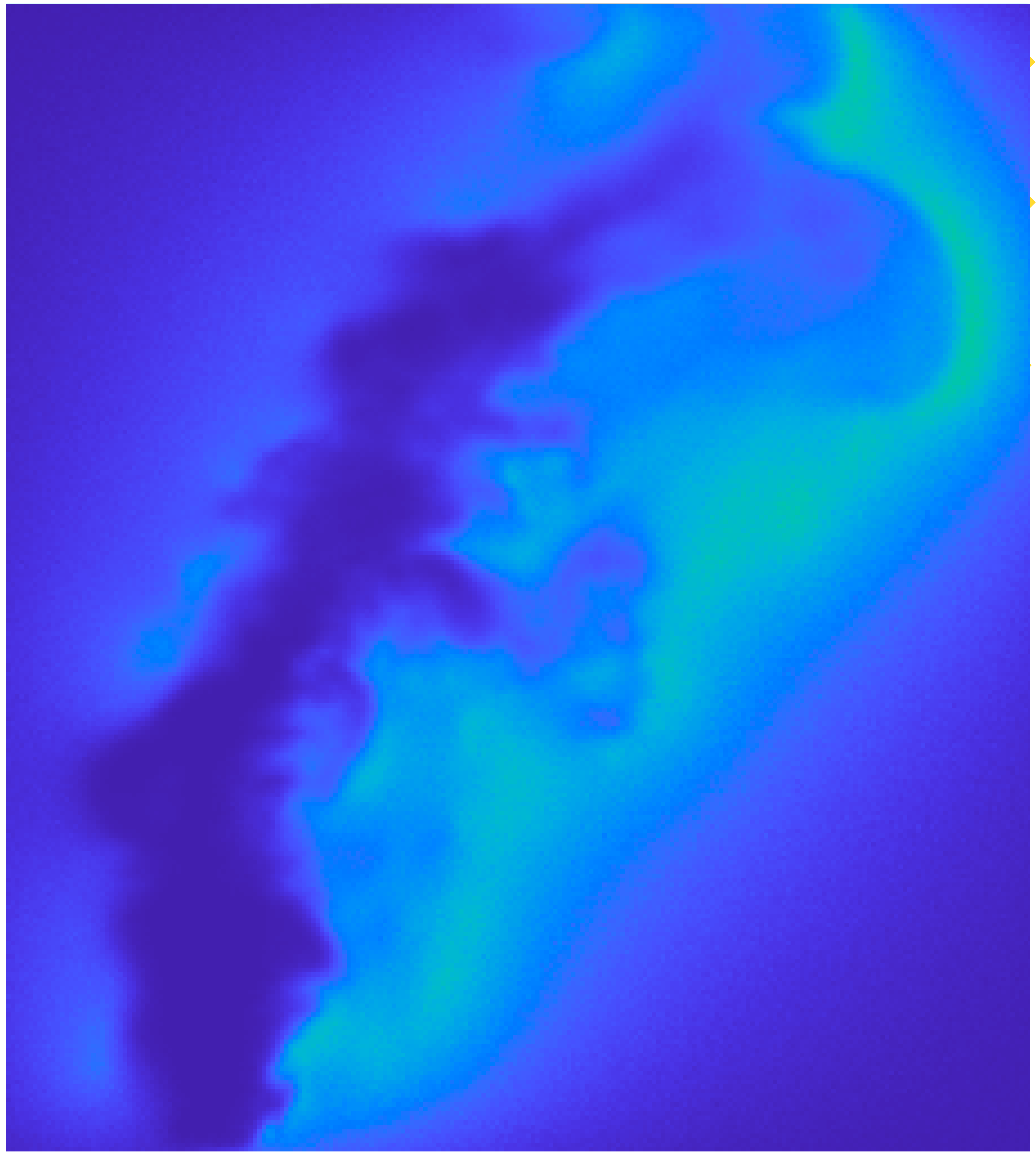
electric potential (V) -1 1

Simulation of photochemical effect

photochemical ozone reaction
[Madronich and Flocke 1999; pg 4-5]

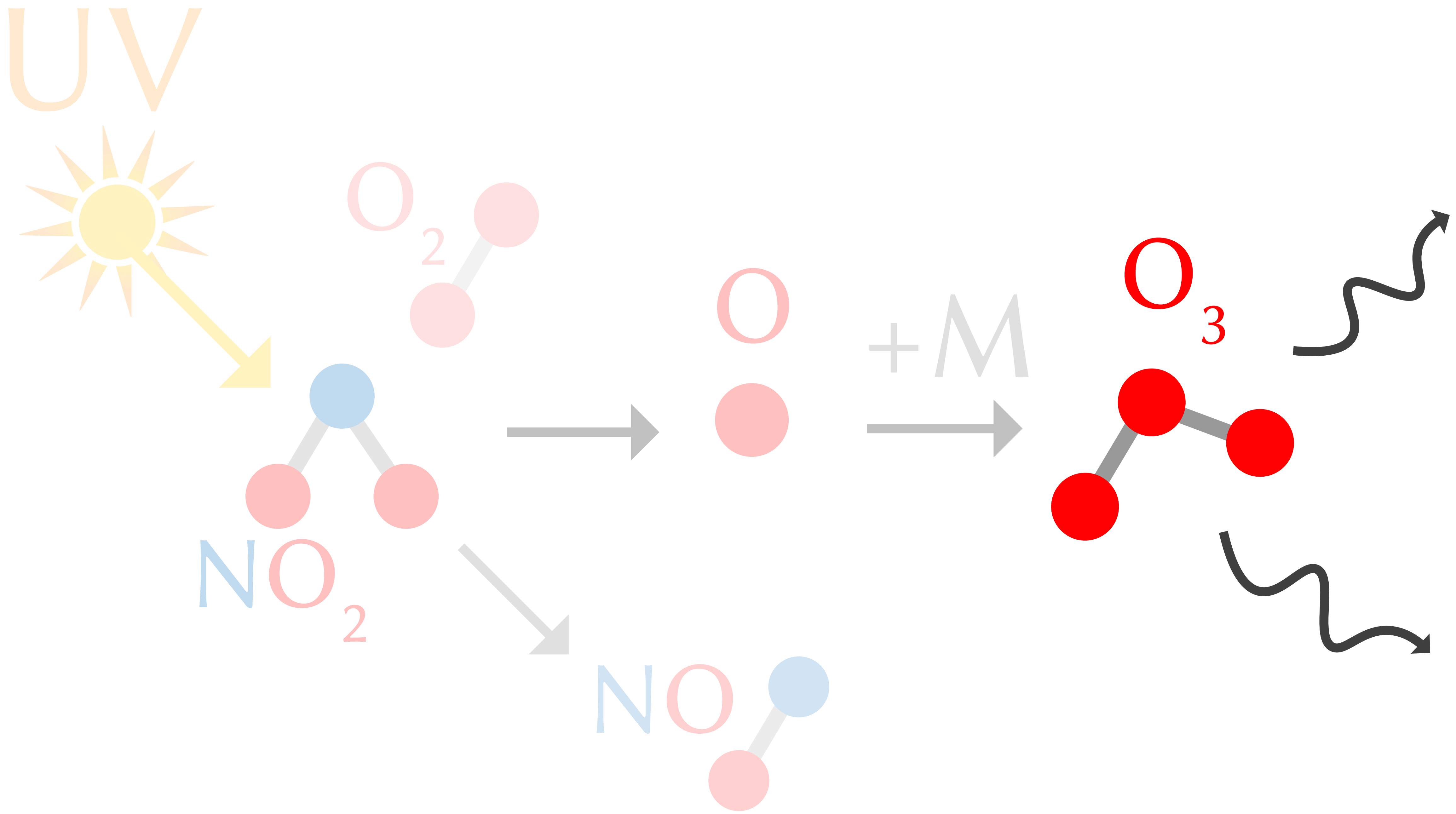


Simulation of photochemical effect

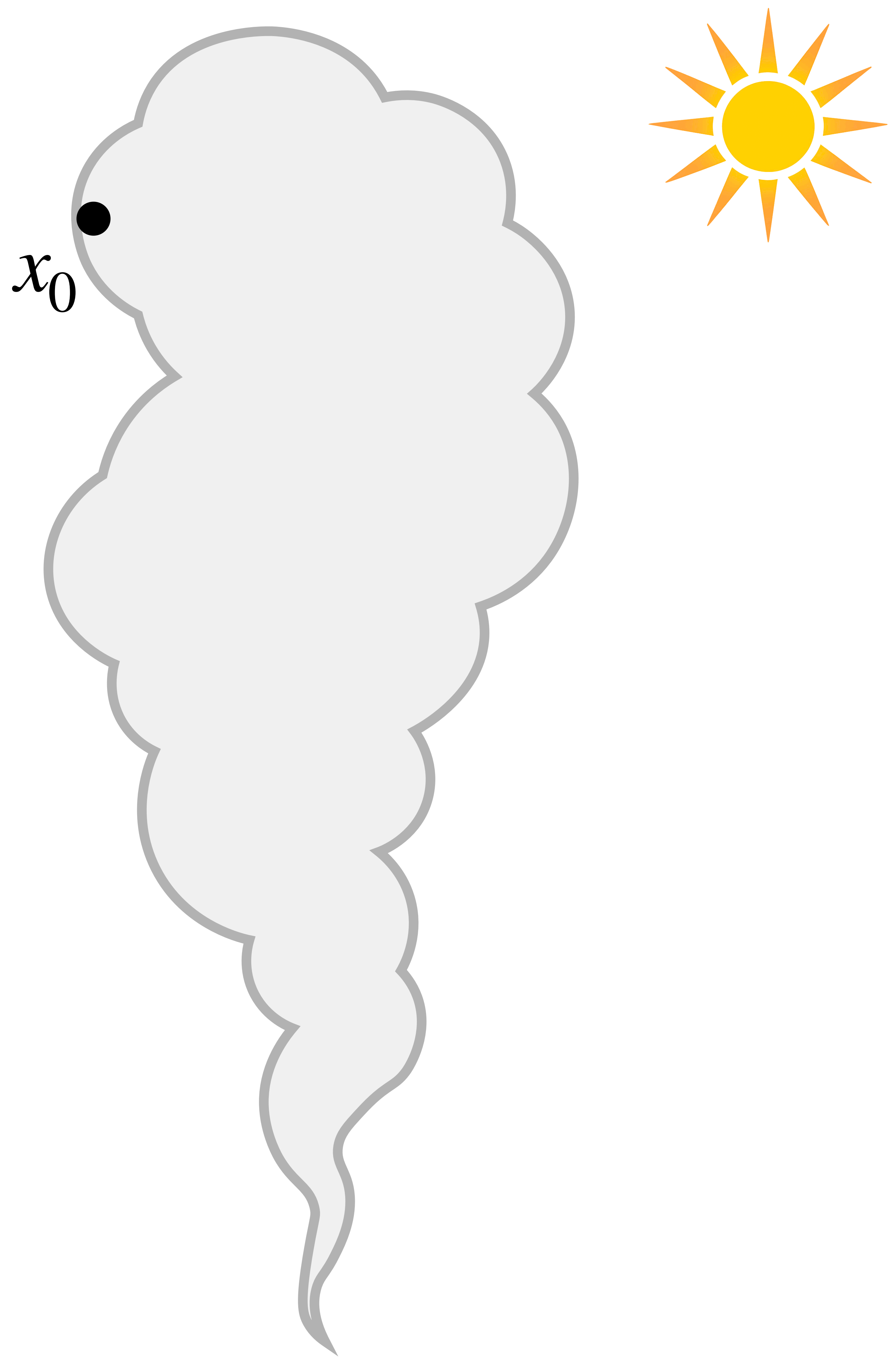


concentration (ppm) 0 max

photochemical ozone reaction
[Madronich and Flocke 1999; pg 4-5]

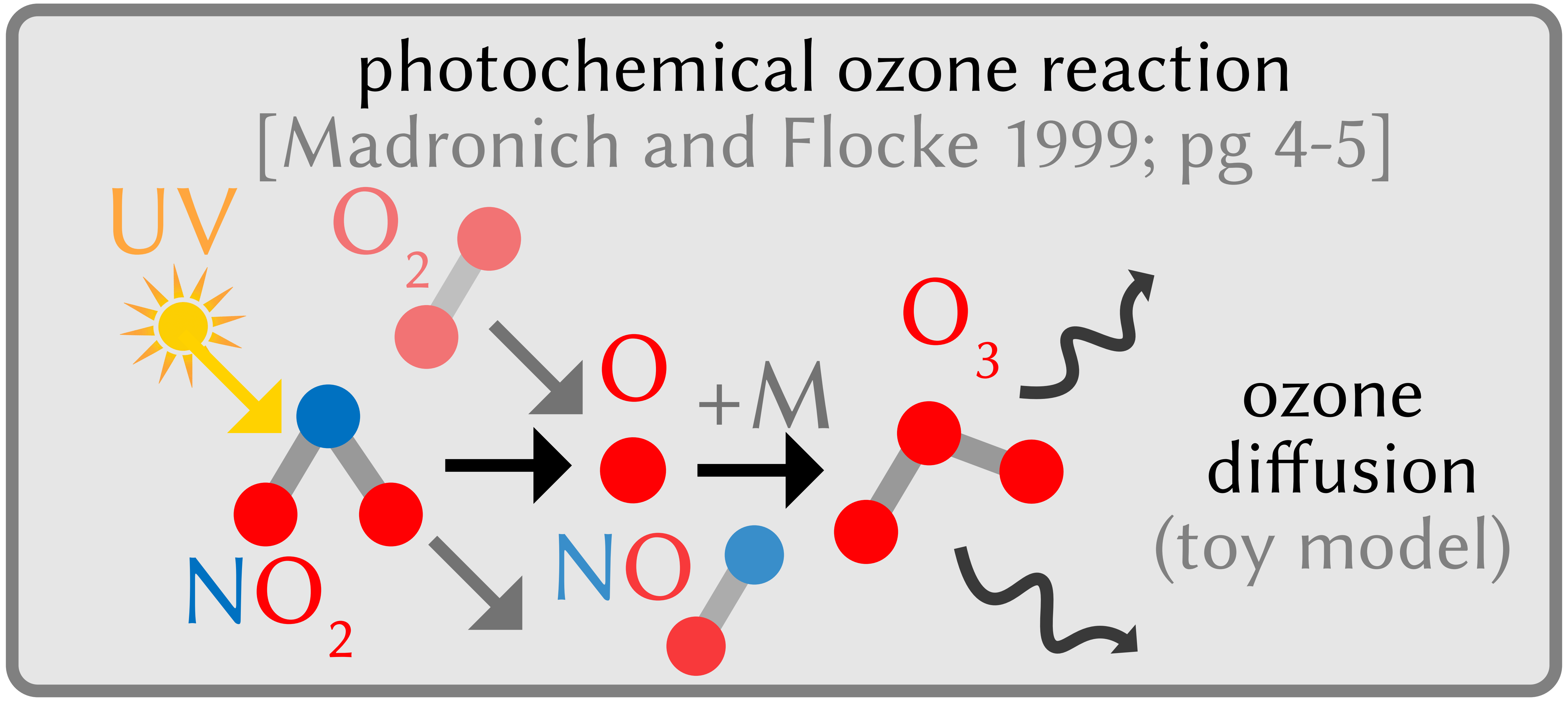


Simulation of photochemical effect



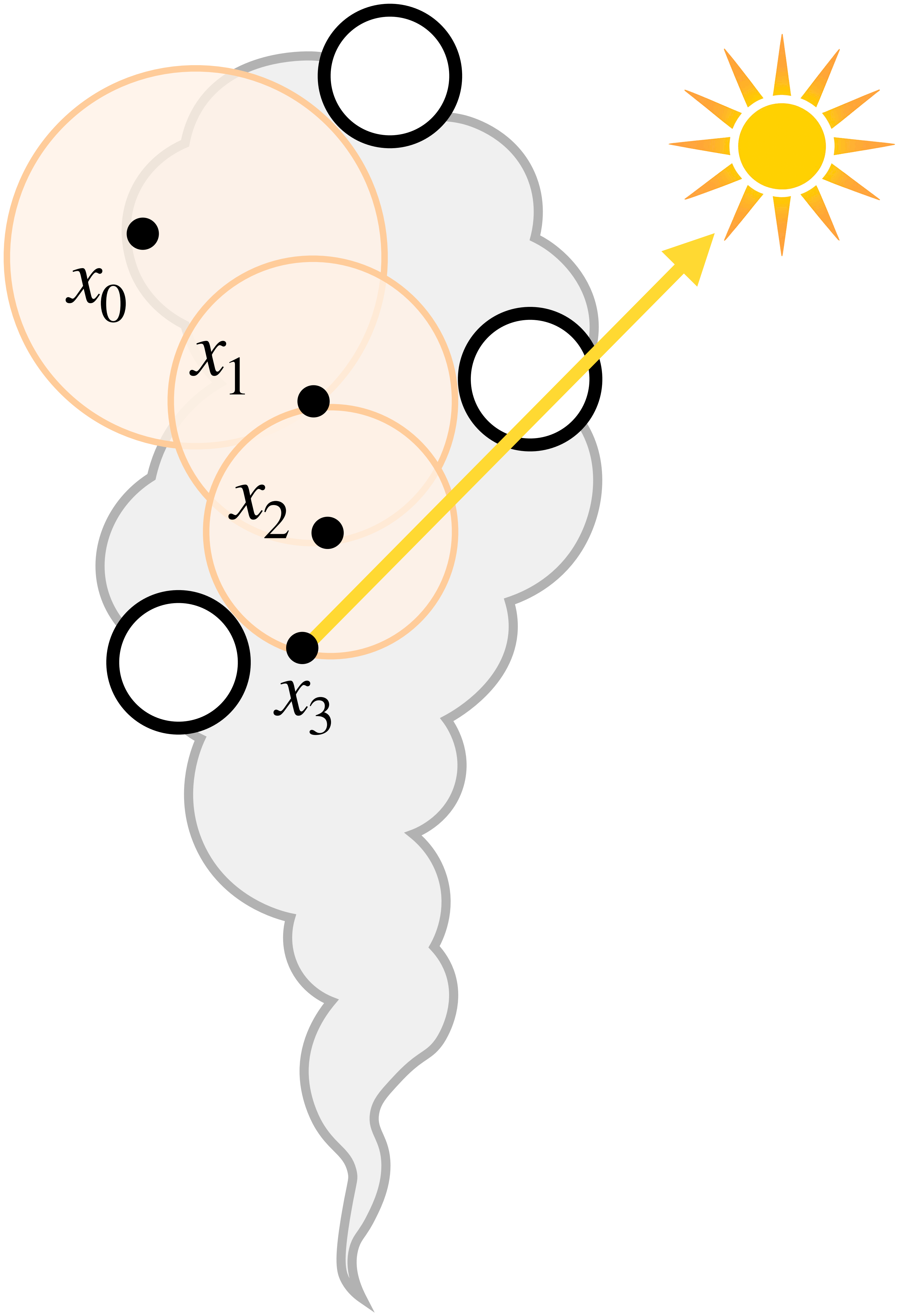
VWoS → VPT
coupled w/out memory

VWoS → VPT
coupled w/ memory



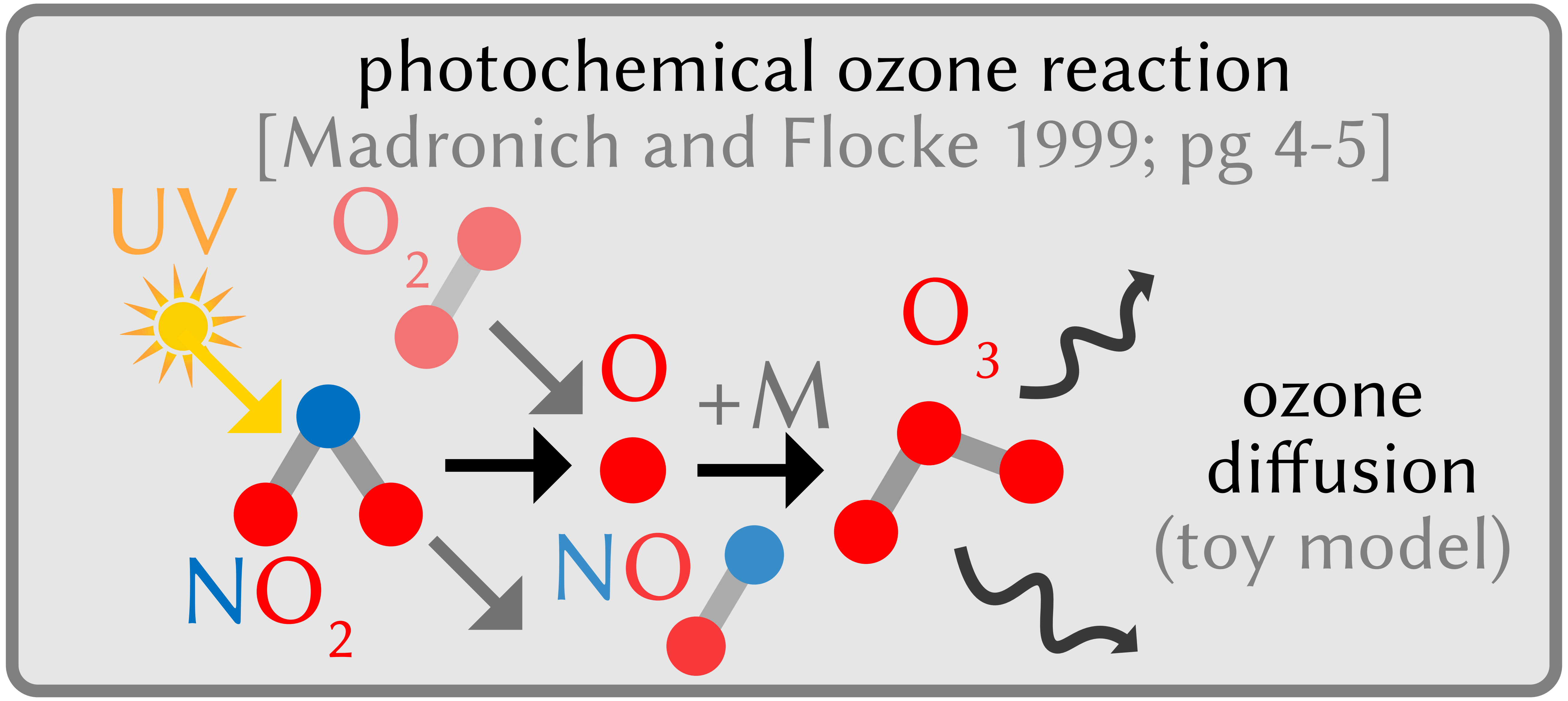
concentration (ppm) 0 max

Simulation of photochemical effect



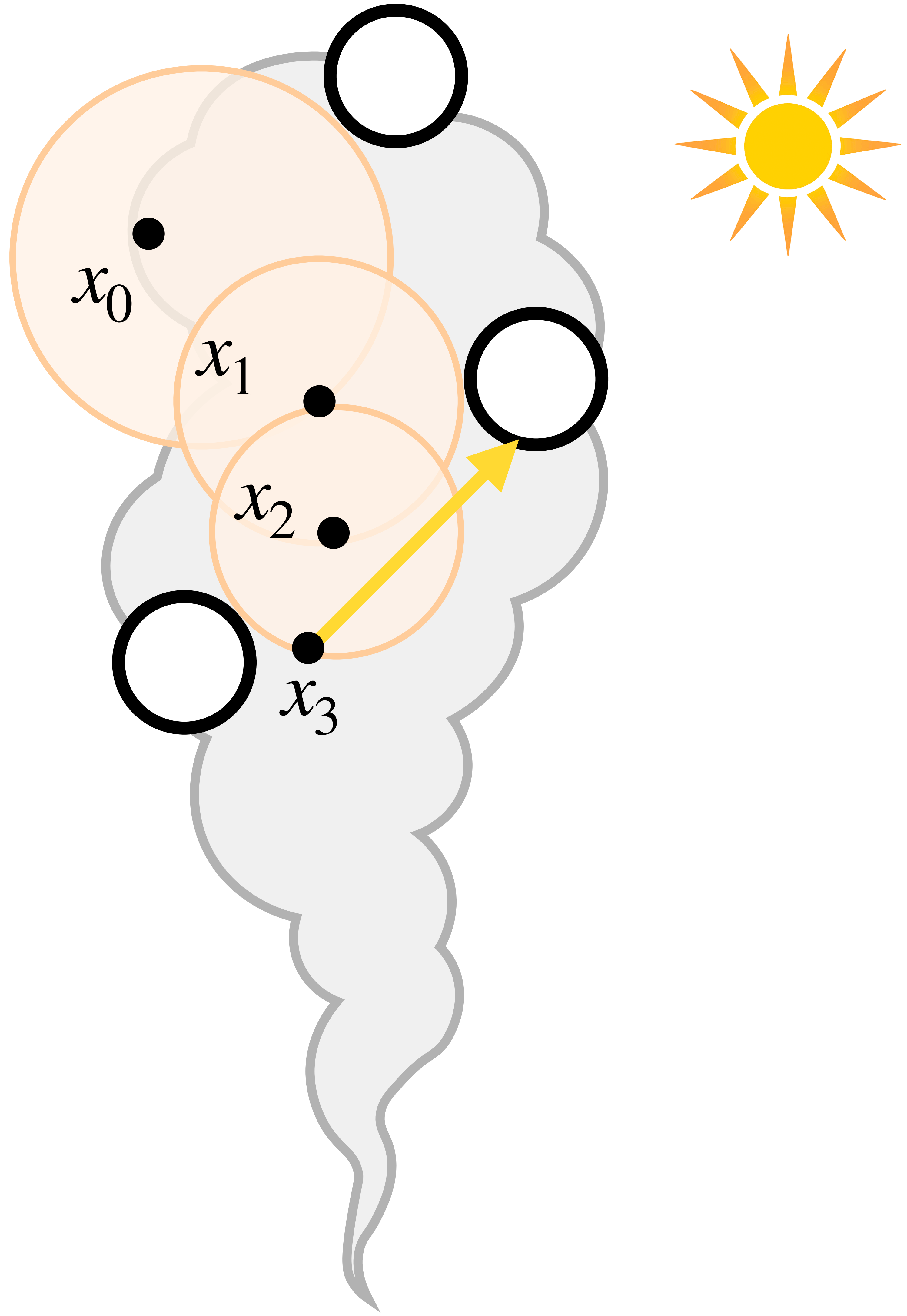
VWoS → VPT
coupled w/out memory

VWoS → VPT
coupled w/ memory



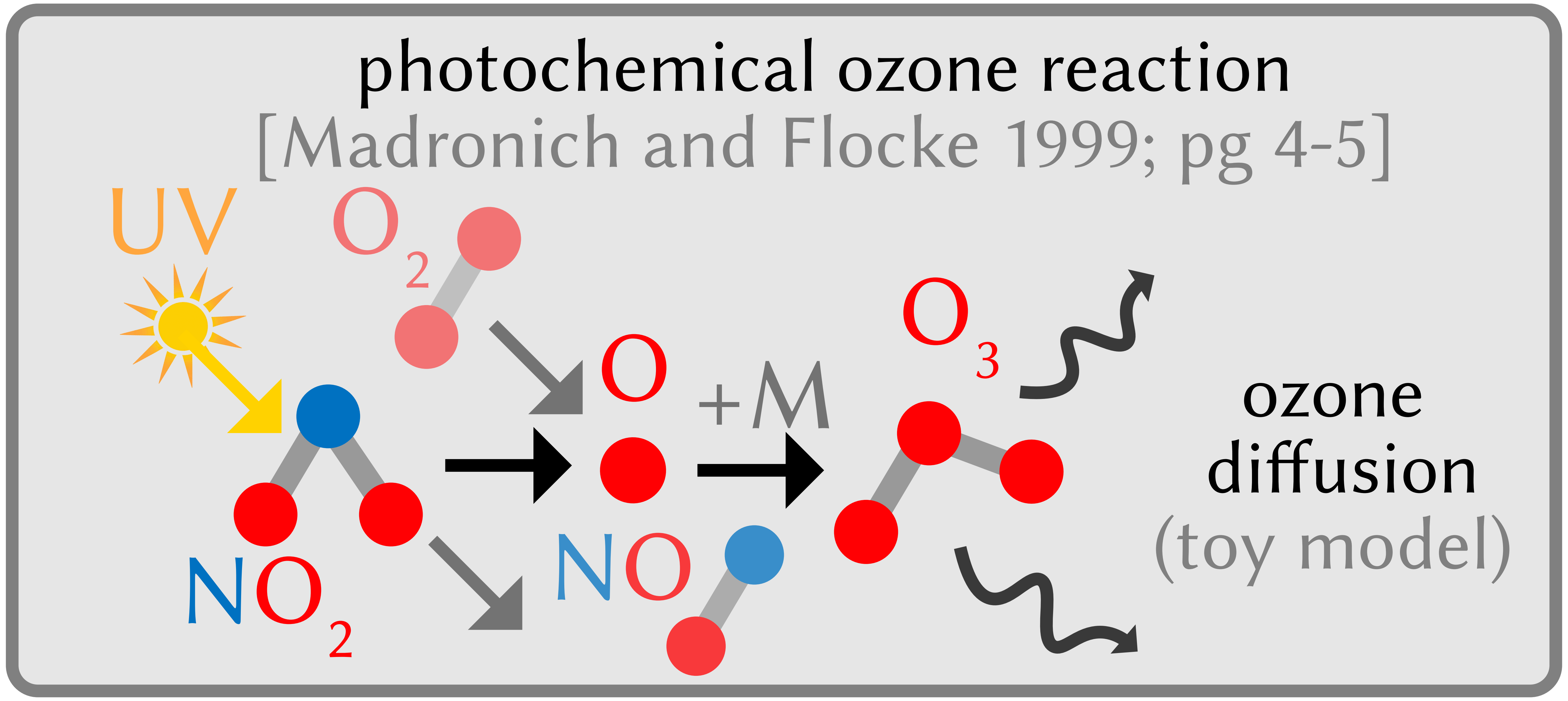
concentration (ppm) 0 max

Simulation of photochemical effect



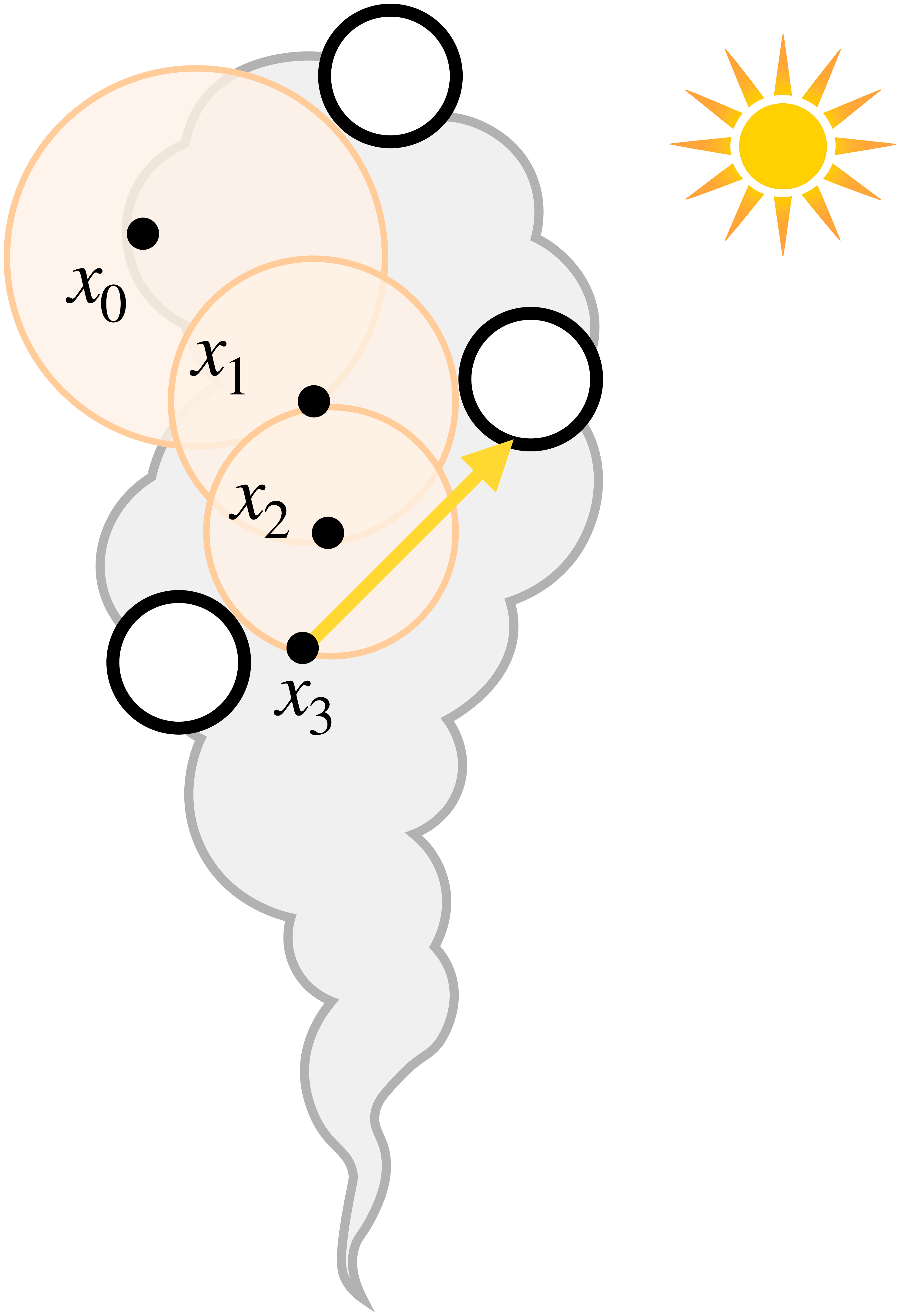
VWoS → VPT coupled w/out memory

VWoS → VPT coupled w/ memory



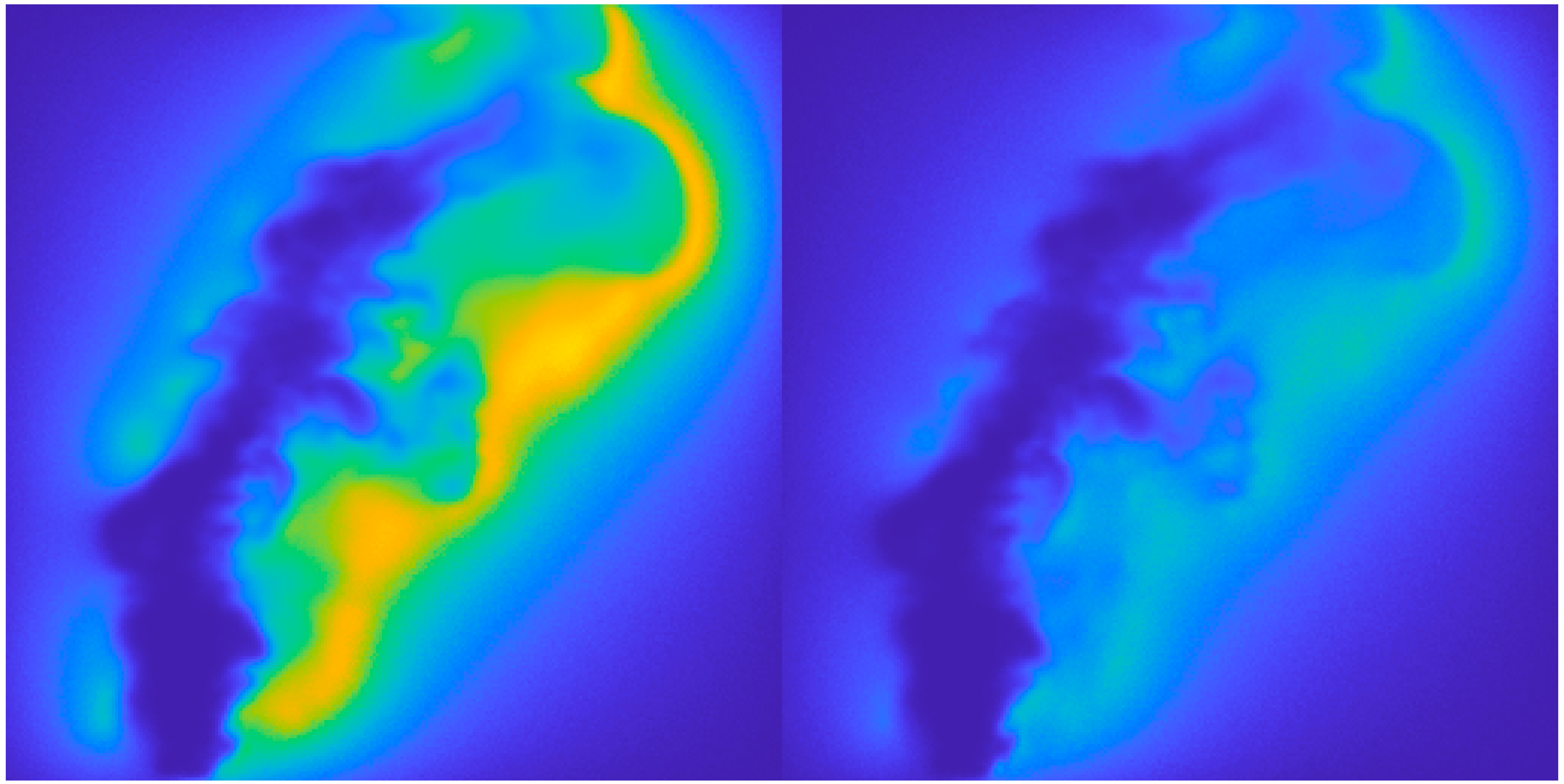
concentration (ppm) 0 max

Simulation of photochemical effect



VWoS → VPT coupled w/out memory

VWoS → VPT coupled w/ memory



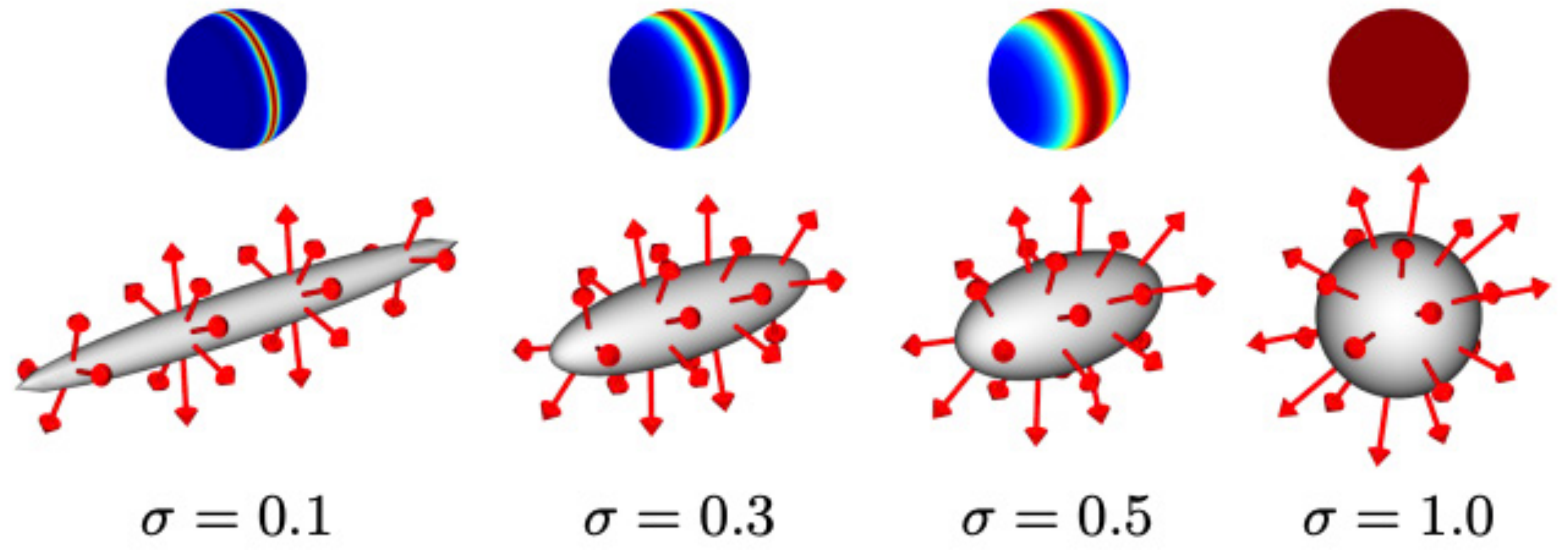
concentration (ppm) 0 max

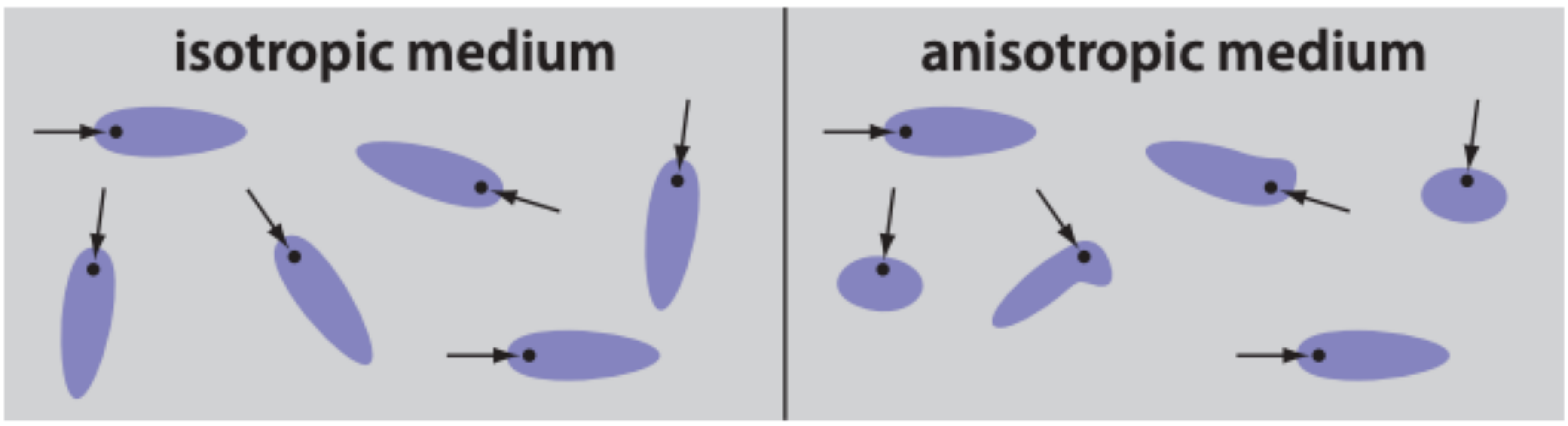
What’s next?

More interesting particles and correlations

anisotropic particles

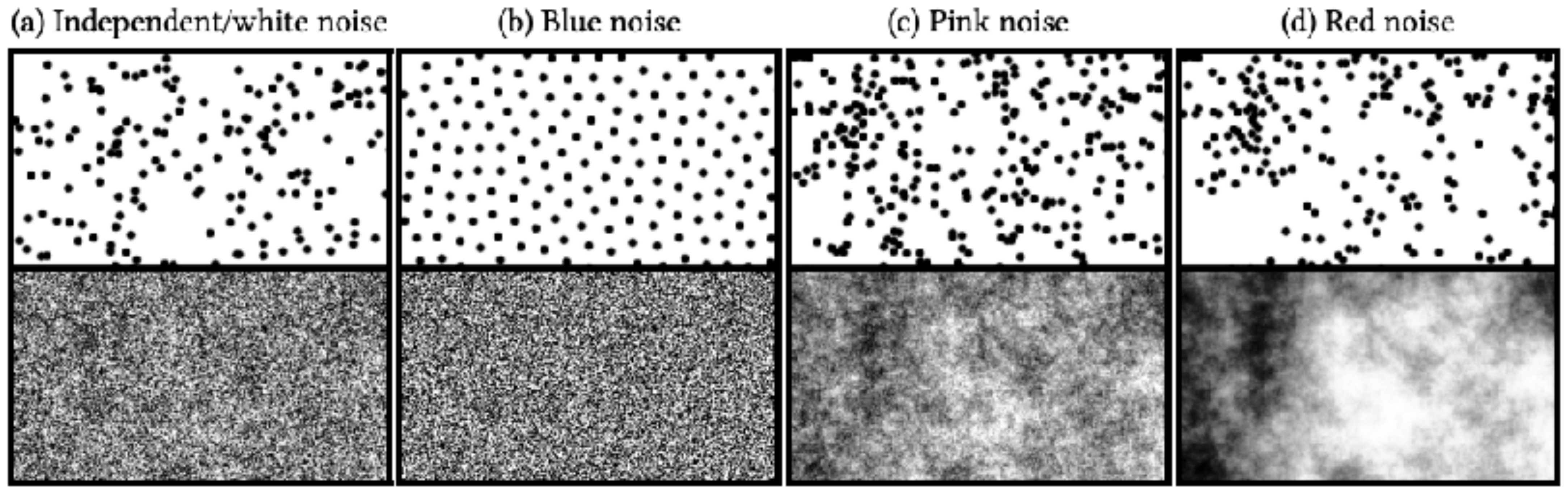
[Heitz et al. 2015, Jakob et al. 2010]





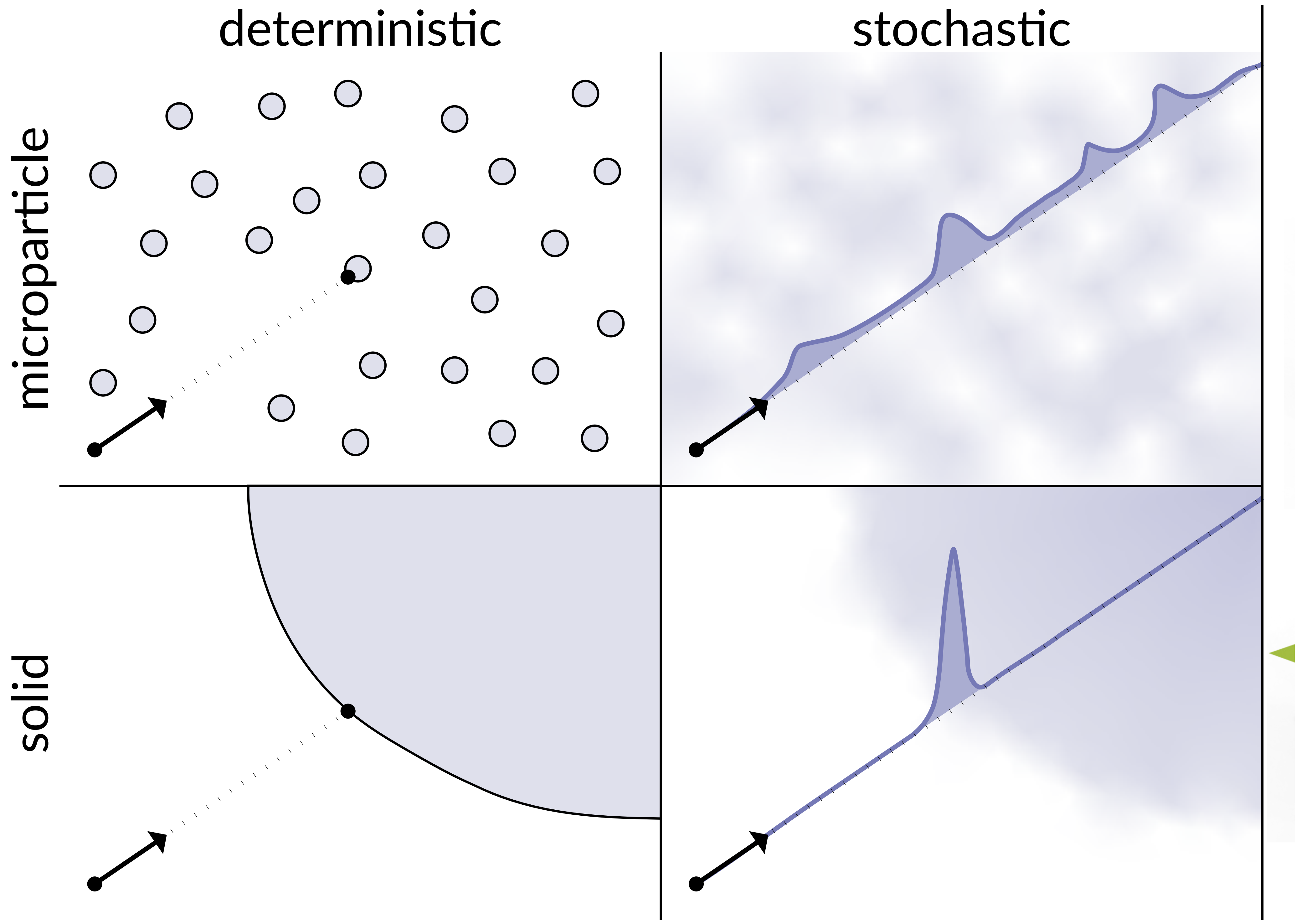
correlated media

[Bitterli et al. 2018, Jarobo et al. 2018, d'Eon et al 2018]

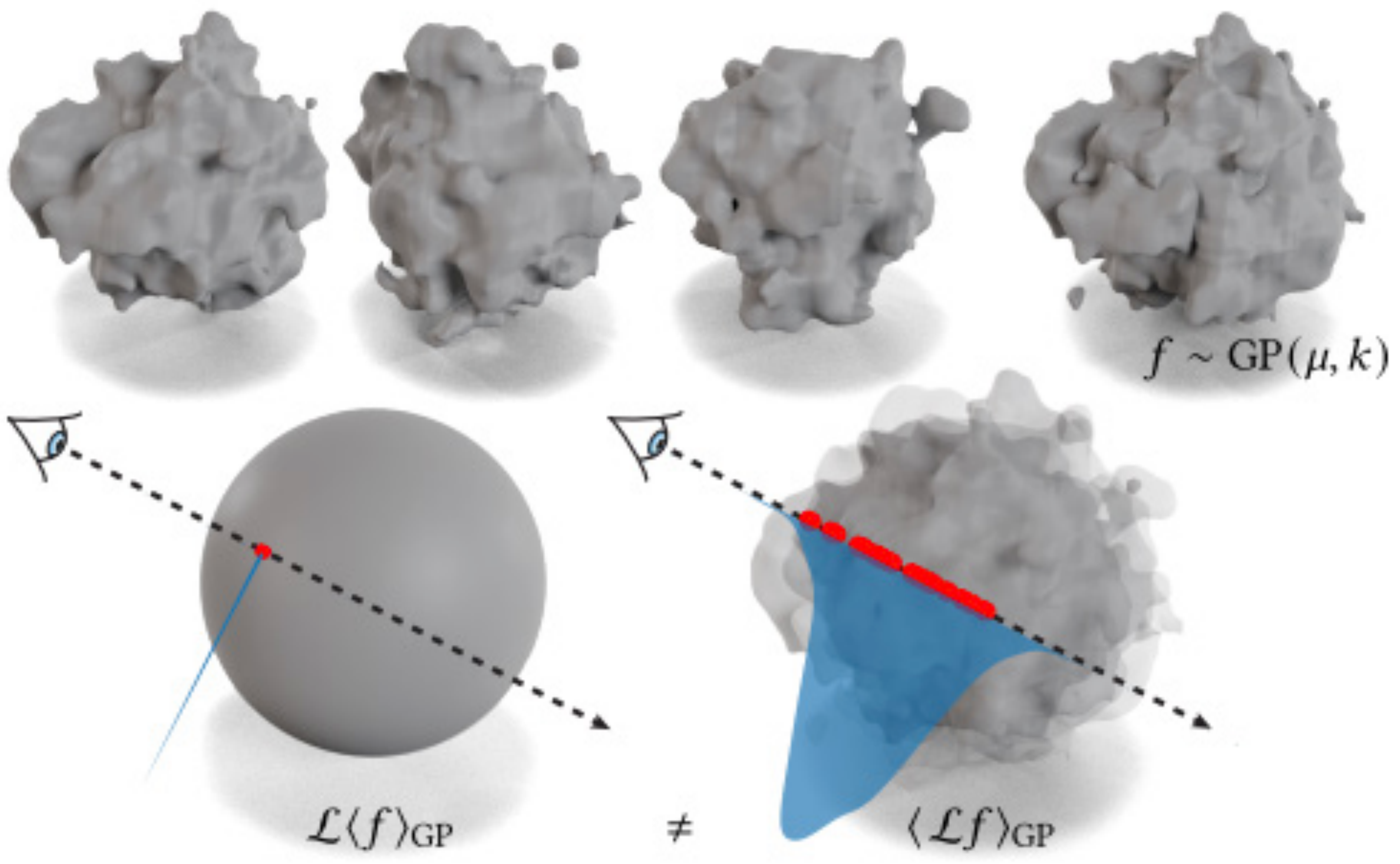


Stochastic geometry beyond microparticles

volume rendering
stochastic opaque solids
[Miller et al. 2024]



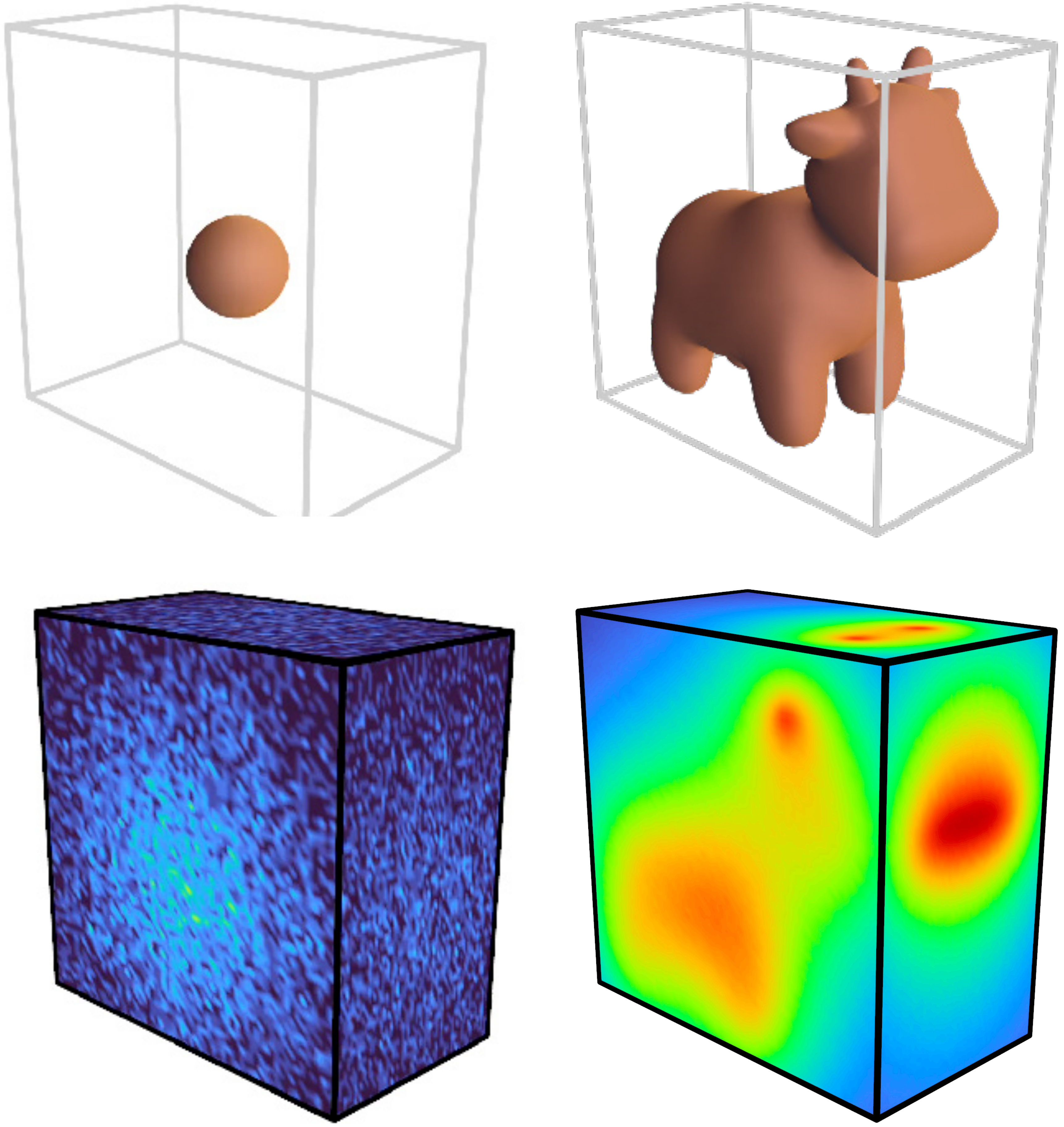
volume rendering
Gaussian process implicit surfaces
[Seyb et al. 2024]



NeRF for Physics

differential walk on spheres

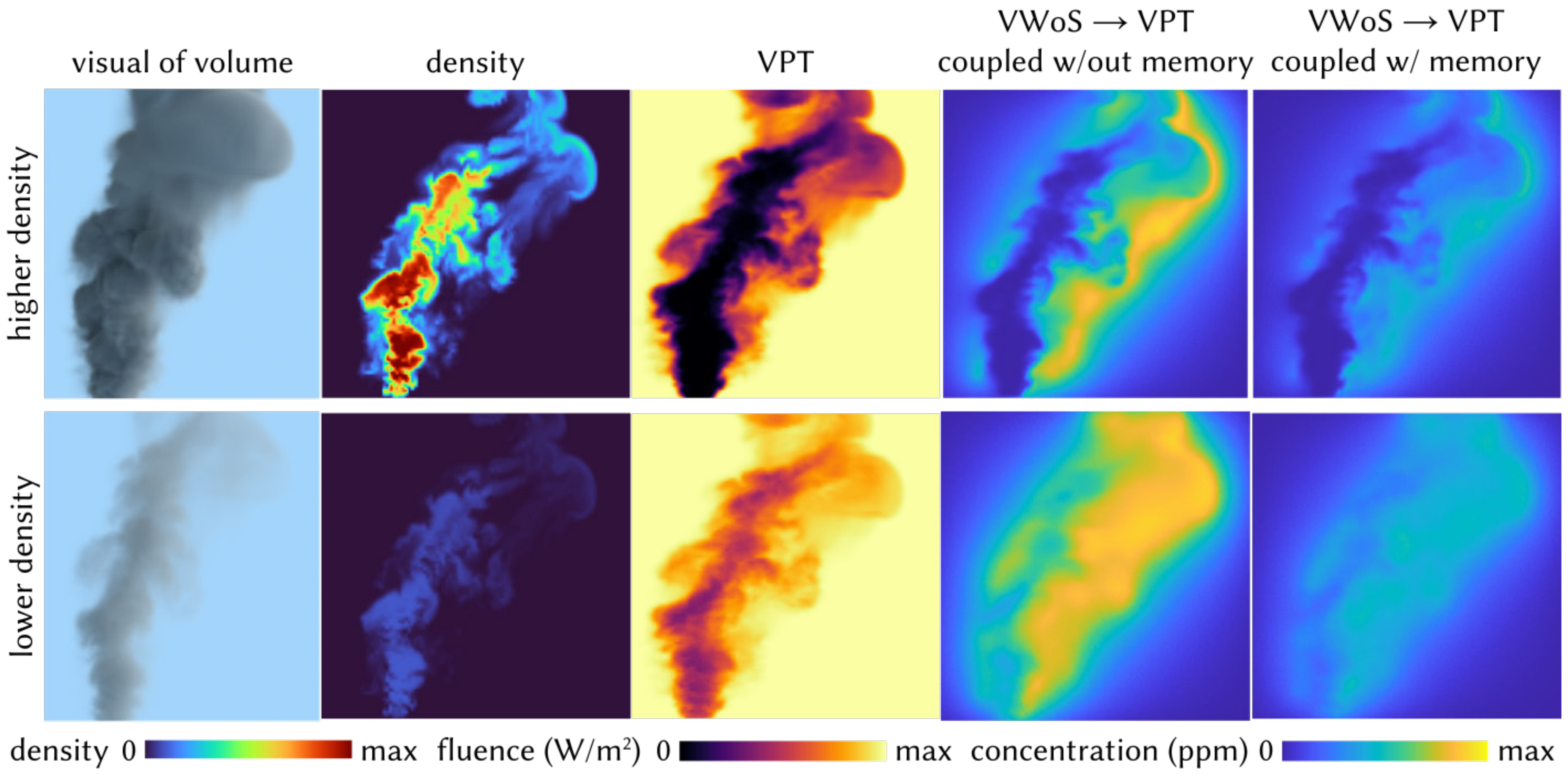
[Miller et al. 2024]



+

volumetric walk on spheres

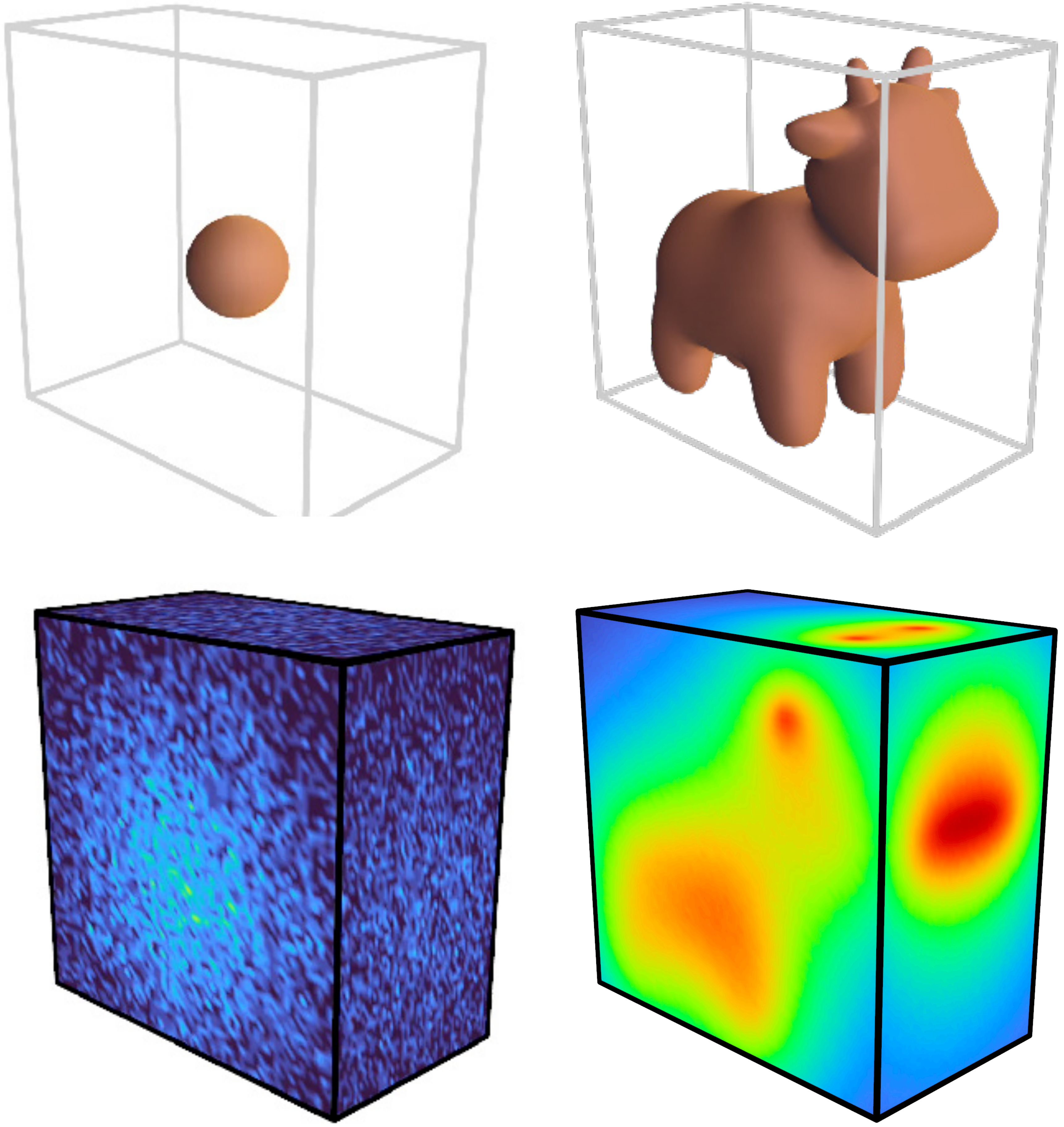
ours



NeRF for Physics

differential walk on spheres

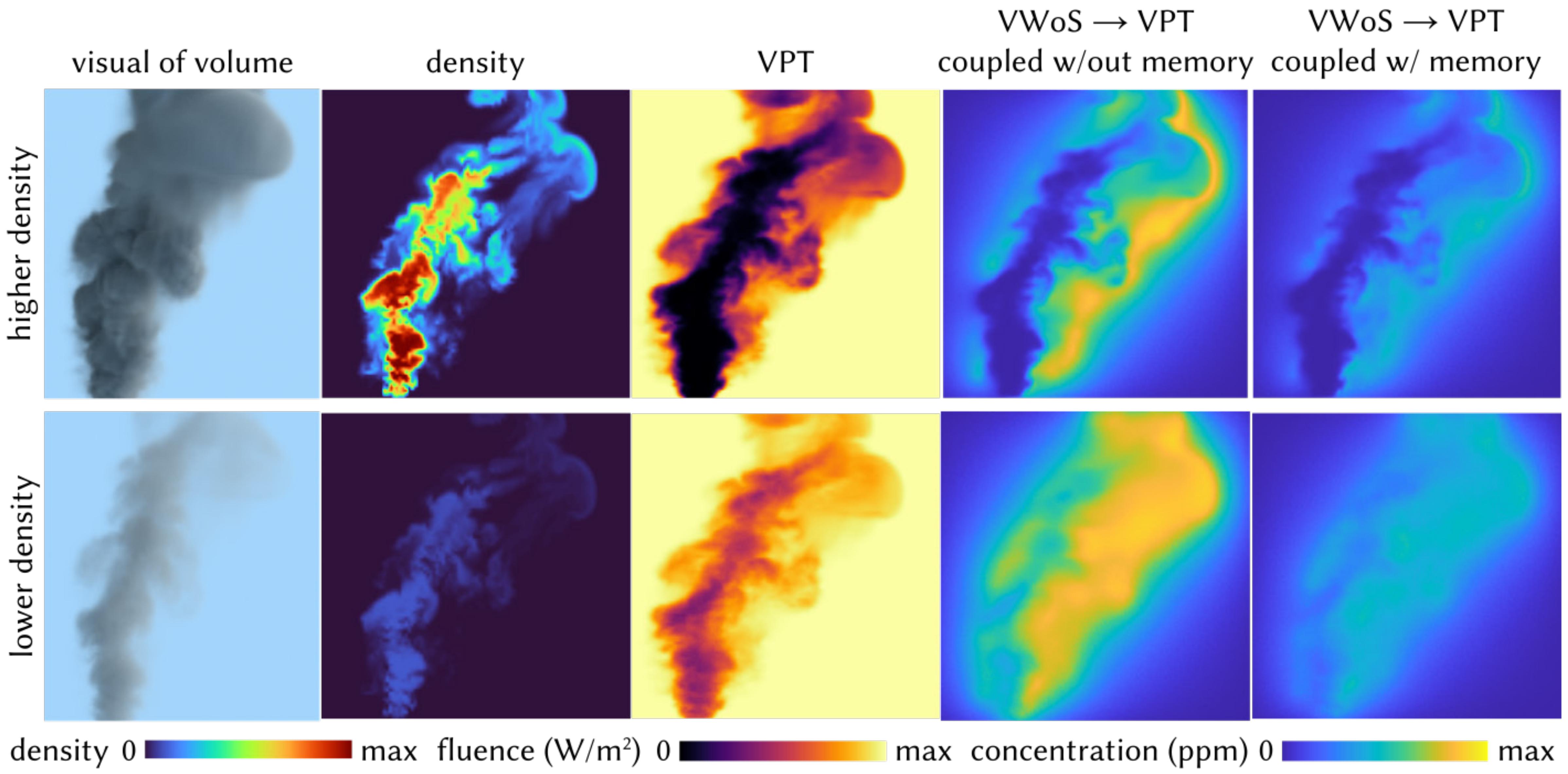
[Miller et al. 2024]



+

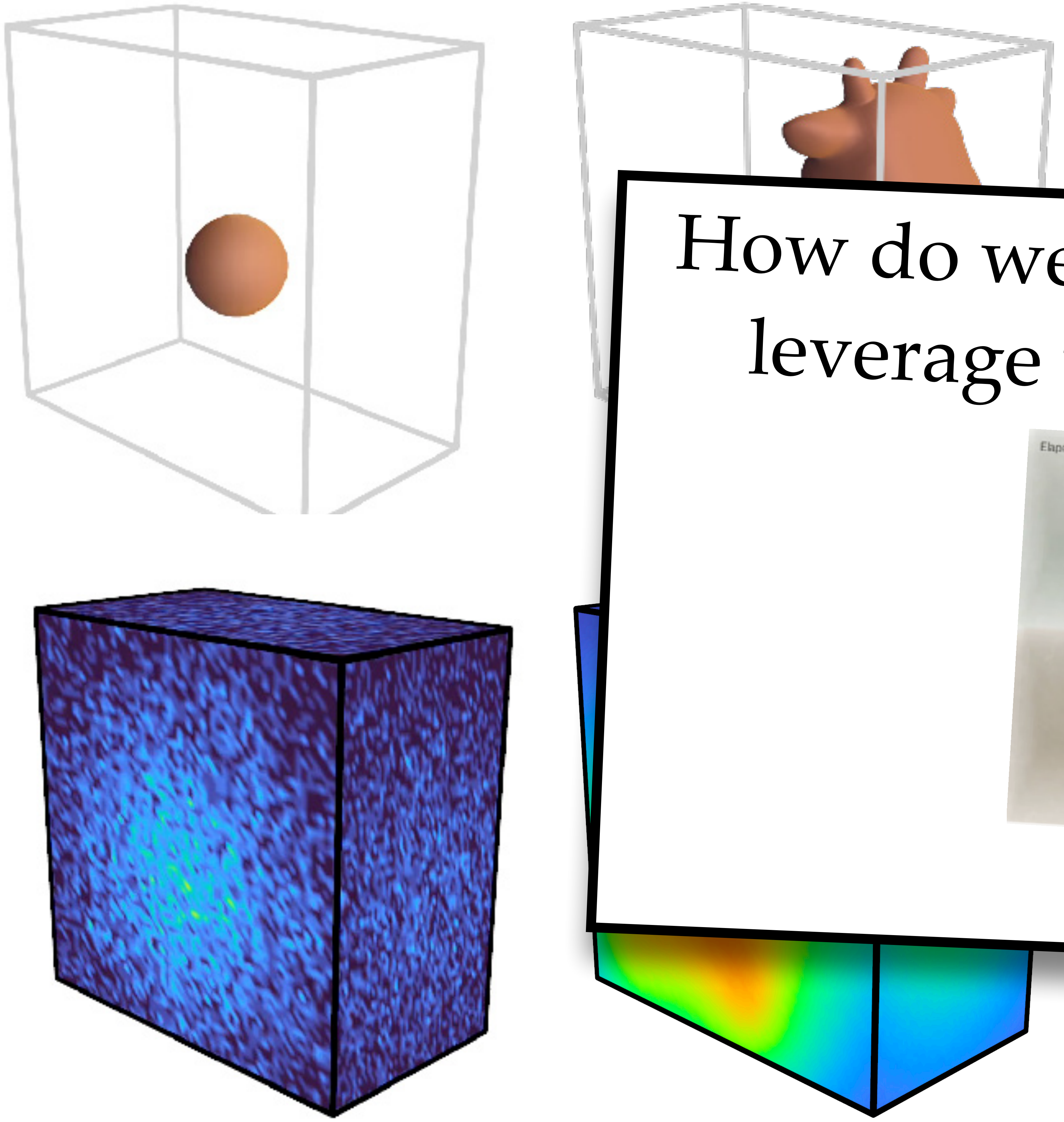
volumetric walk on spheres

ours

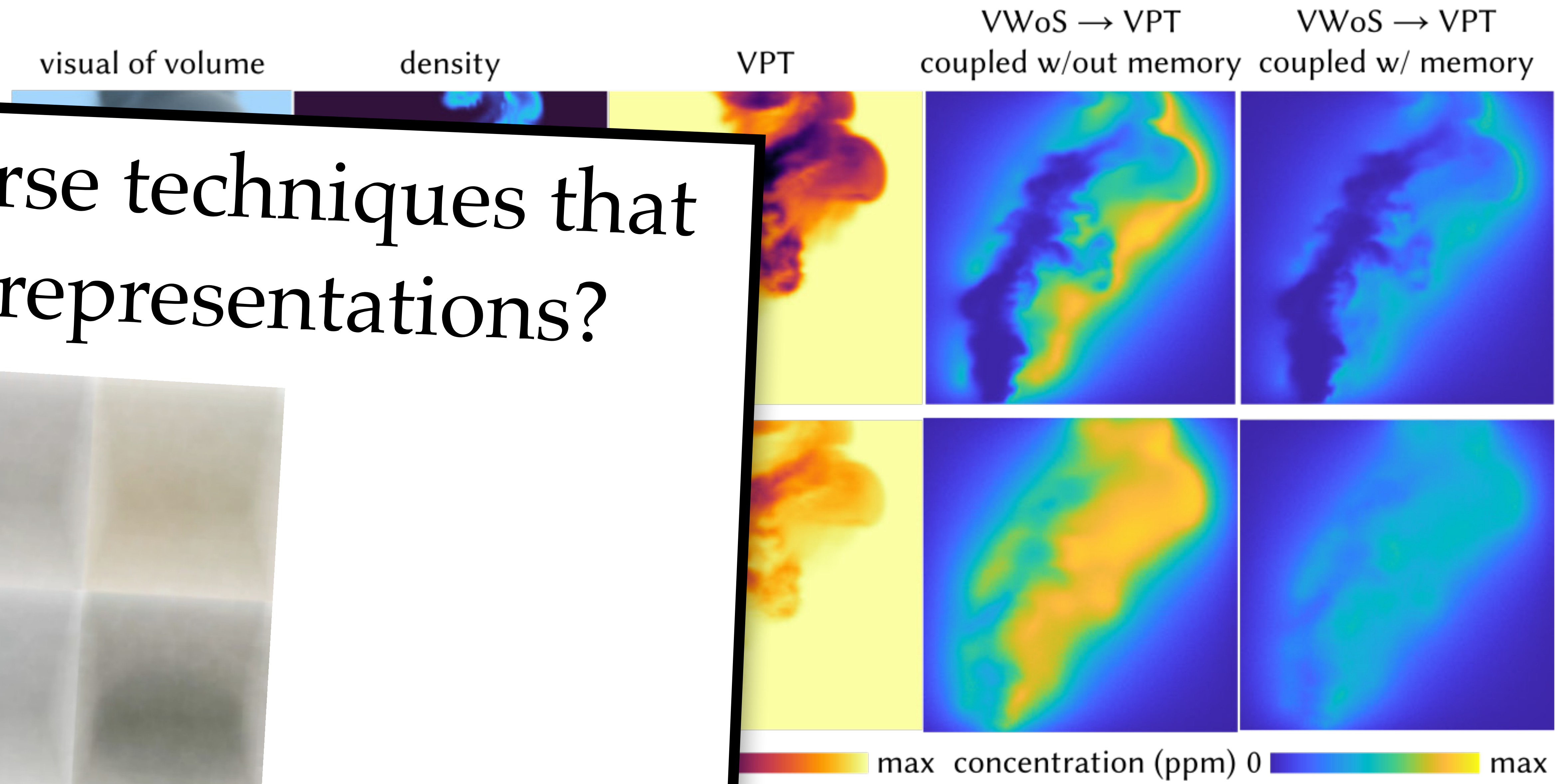


NeRF for Physics

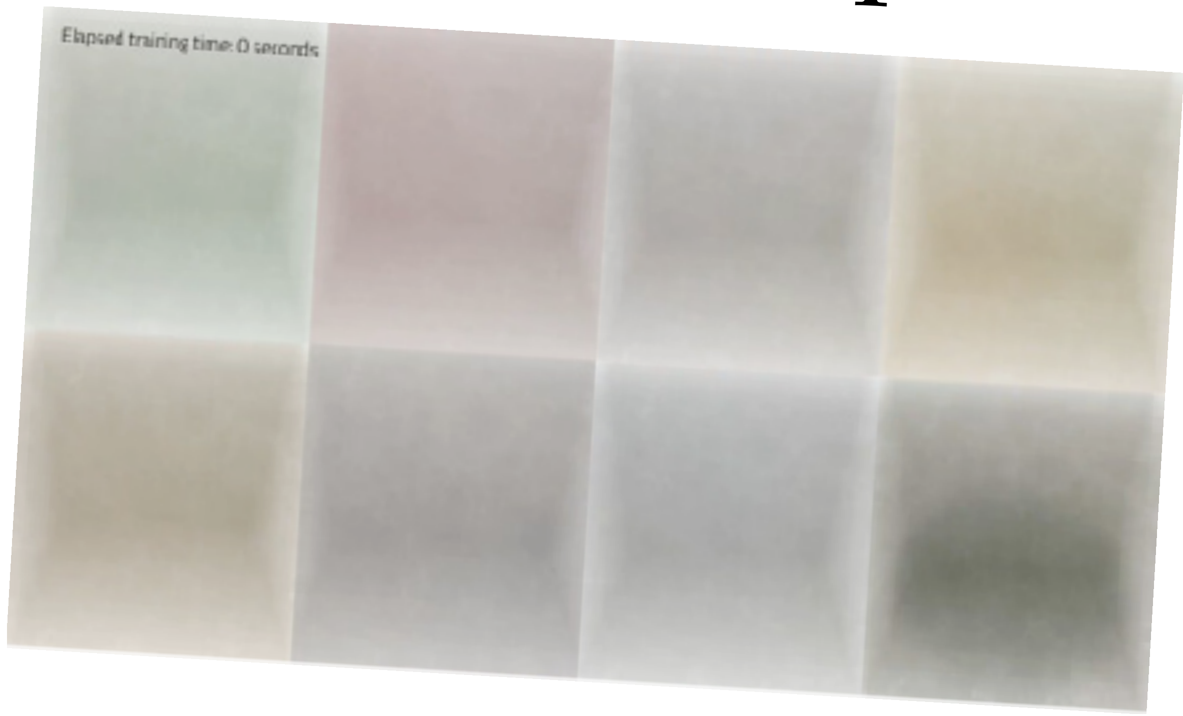
differential walk on spheres
[Miller et al. 2024]



volumetric walk on spheres
ours



How do we build inverse techniques that leverage volumetric representations?



[Instant NGP, Muller et al. 2022]

Thank you!

On faculty job market this fall

bailey-miller.com