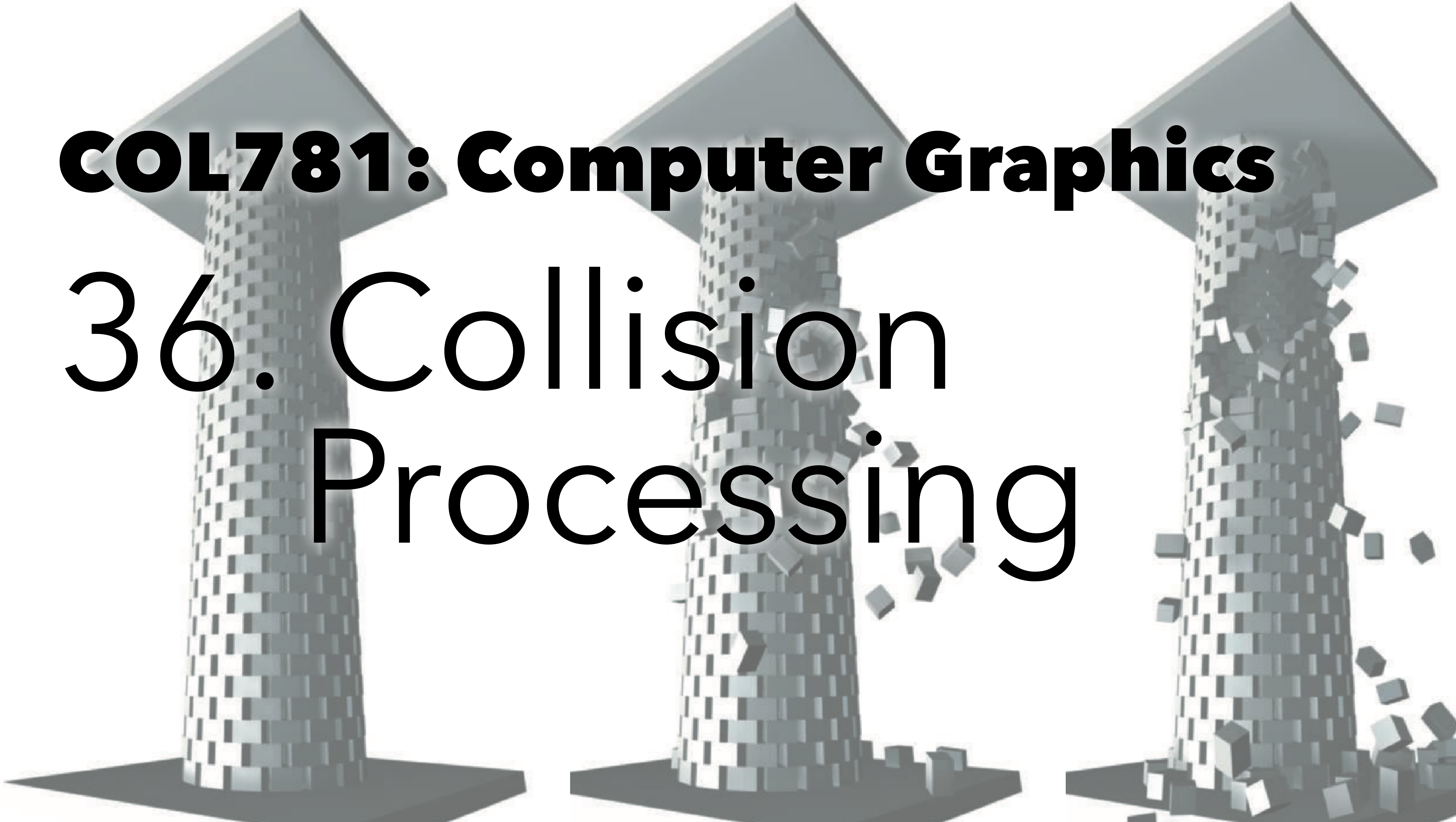


# COL781: Computer Graphics

# 36. Collision Processing

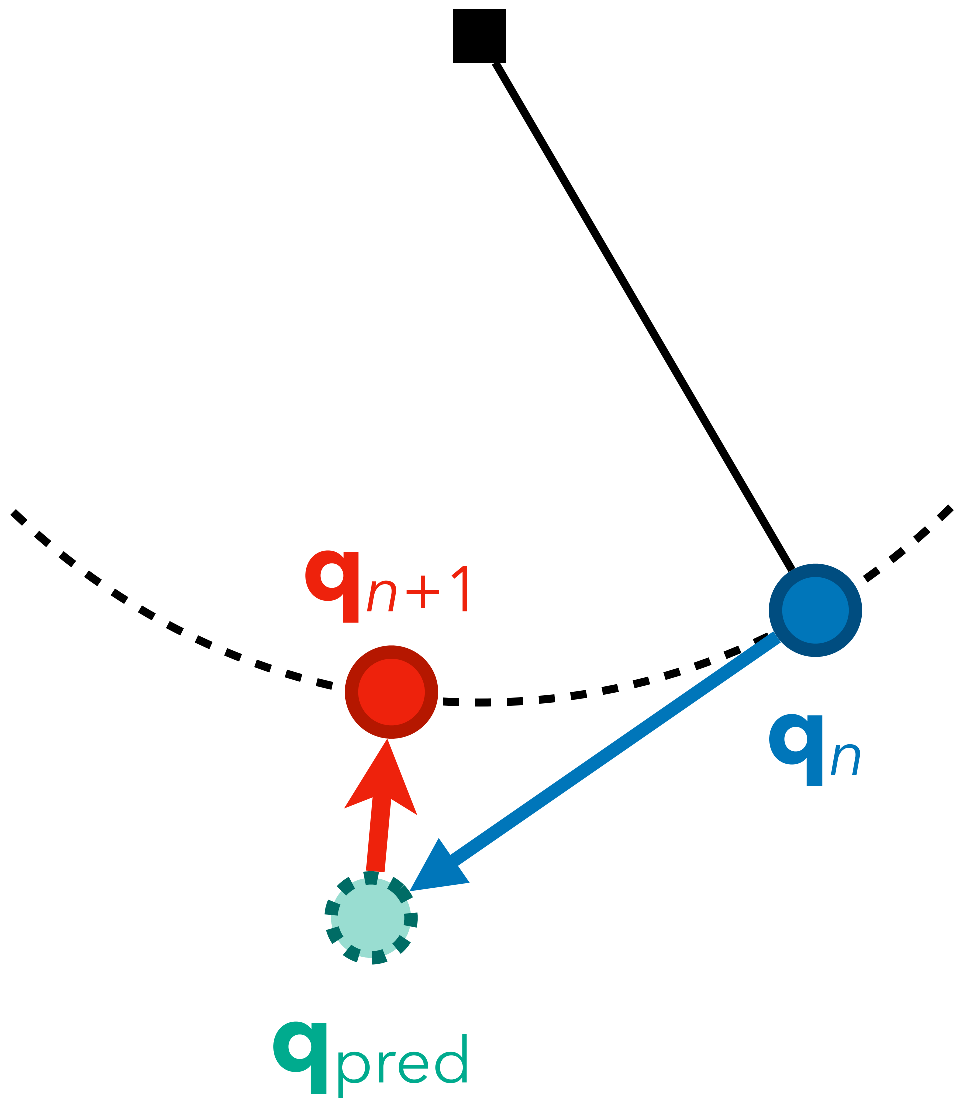
Erleben 2007

# Recap: Constraints

$$c(\mathbf{q}) = 0$$

$$\mathbf{f}_c = \lambda \nabla c(\mathbf{q})$$

**Projection method:**




$$\mathbf{q}_{n+1} = \mathbf{q}_{\text{pred}} + \sum \mathbf{M}^{-1} \lambda_j \nabla c_j(\mathbf{q}_{n+1}) \Delta t^2$$
$$c_j(\mathbf{q}_{n+1}) = 0 \quad \text{for } j = 1, 2, \ldots$$

$$\text{where } \mathbf{q}_{\text{pred}} = \mathbf{q}_n + \mathbf{v}_n \Delta t + \mathbf{M}^{-1} \mathbf{f}(\mathbf{q}_n, \mathbf{v}_n) \Delta t^2.$$

Solve for $\mathbf{q}_{n+1}$ and $\lambda_1, \lambda_2, \ldots$ simultaneously using Newton's method

…Then update $\mathbf{v}_{n+1} = (\mathbf{q}_{n+1} - \mathbf{q}_n)/\Delta t$

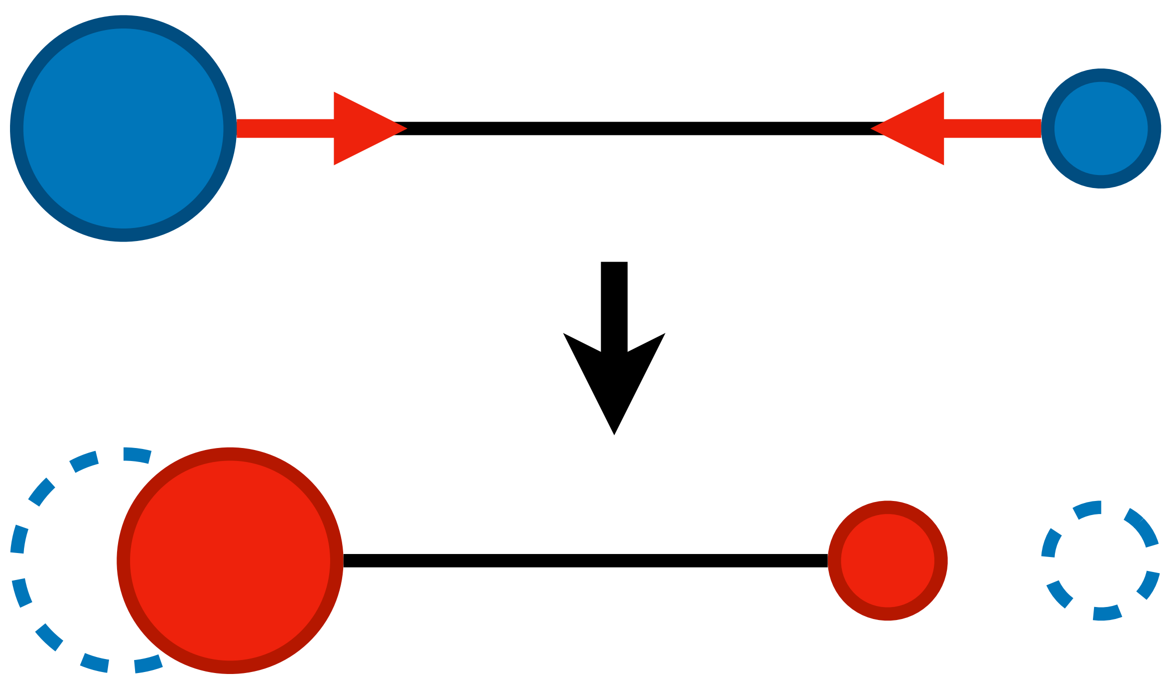
# Position-based dynamics

Solving a big linear system for all $\lambda$'s is too expensive for real-time graphics!
But it's easy to solve one constraint at a time:

**Example:** Inextensible spring between particles $i$ and $j$

$$\|\mathbf{x}_{ij}\| = \ell_0$$

$$\mathbf{f}_{ij} = \lambda\, \hat{\mathbf{x}}_{ij}$$



Recall $\mathbf{q}_{n+1} = \mathbf{q}_{\text{pred}} + \sum \mathbf{M}^{-1} \lambda_j\, \hat{\mathbf{x}}_{ij}\, \Delta t^2$

$$\Delta\mathbf{q}_{n+1} = \mathbf{M}^{-1}\, \Delta\lambda\, \hat{\mathbf{x}}_{ij}\, \Delta t^2$$

Find $\Delta\lambda$ which makes updated positions satisfy $\|\tilde{\mathbf{x}}_{ij} + \Delta\mathbf{x}_{ij}\| = \ell_0$

In general, we have a guess of the next positions: $\tilde{\mathbf{q}}$

1. Applying a constraint force $\Delta\lambda_j$ changes the positions by $\Delta\mathbf{q} = \mathbf{M}^{-1}\,\Delta\lambda_j\,\nabla c_j(\tilde{\mathbf{q}})\,\Delta t^2$
2. Solve for $\Delta\lambda_j$ so that $c_j(\tilde{\mathbf{q}} + \Delta\mathbf{q}) = 0$
3. Update the positions (**constraint projection**): $\tilde{\mathbf{q}} \leftarrow \tilde{\mathbf{q}} + \Delta\mathbf{q}$
4. Repeat for other constraints

Projecting one constraint makes other constraints violated!

- Loop over all constraints = 1 iteration. Have to repeat many iterations
- If not enough iterations, constraints appear soft!

https://www.youtube.com/watch?v=j5igW5-h4ZM

Müller et al. 2006

https://www.youtube.com/watch?v=j5igW5-h4ZM

Müller et al. 2006

# Rigid bodies and collisions

# Rigid bodies

Degrees of freedom: Center of mass position **x**, rotation (matrix **R** or quaternion **q**)
…Basically just the body's coordinate system

Kinematics:

- (Linear) velocity: $\dot{\mathbf{x}} = \mathbf{v}$
- Angular velocity: $\boldsymbol{\omega}$

$$\dot{\mathbf{R}} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \mathbf{R} \quad \text{or} \quad \dot{\mathbf{q}} = \frac{1}{2} \begin{bmatrix} q_x & -q_y & -q_z \\ q_w & q_z & -q_y \\ -q_z & q_w & q_x \\ q_y & -q_x & q_w \end{bmatrix} \boldsymbol{\omega}$$

Dynamics:

$$\dot{\mathbf{v}} = m^{-1}\,\mathbf{f}$$

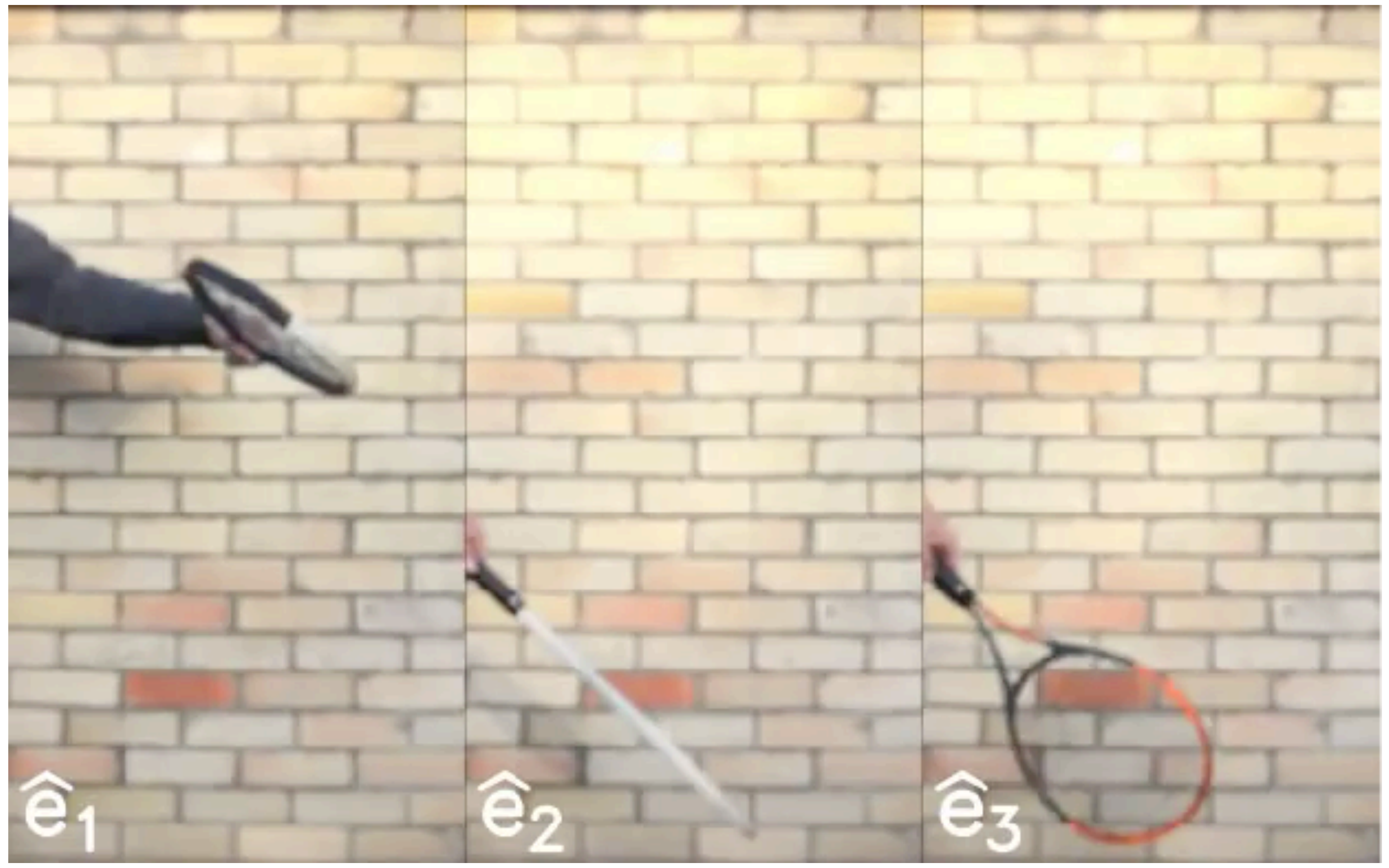
$$\dot{\boldsymbol{\omega}} = \mathbf{I}^{-1}\,(\boldsymbol{\tau} - \boldsymbol{\omega} \times \mathbf{I}\,\boldsymbol{\omega})$$

where $\mathbf{I}$ = moment of inertia, $\boldsymbol{\tau}$ = net torque = $\sum (\mathbf{p}_i - \mathbf{x}) \times \mathbf{f}_i$

$\boldsymbol{\omega} \times \mathbf{I}\,\boldsymbol{\omega}$ = "gyroscopic term" that makes things tumble

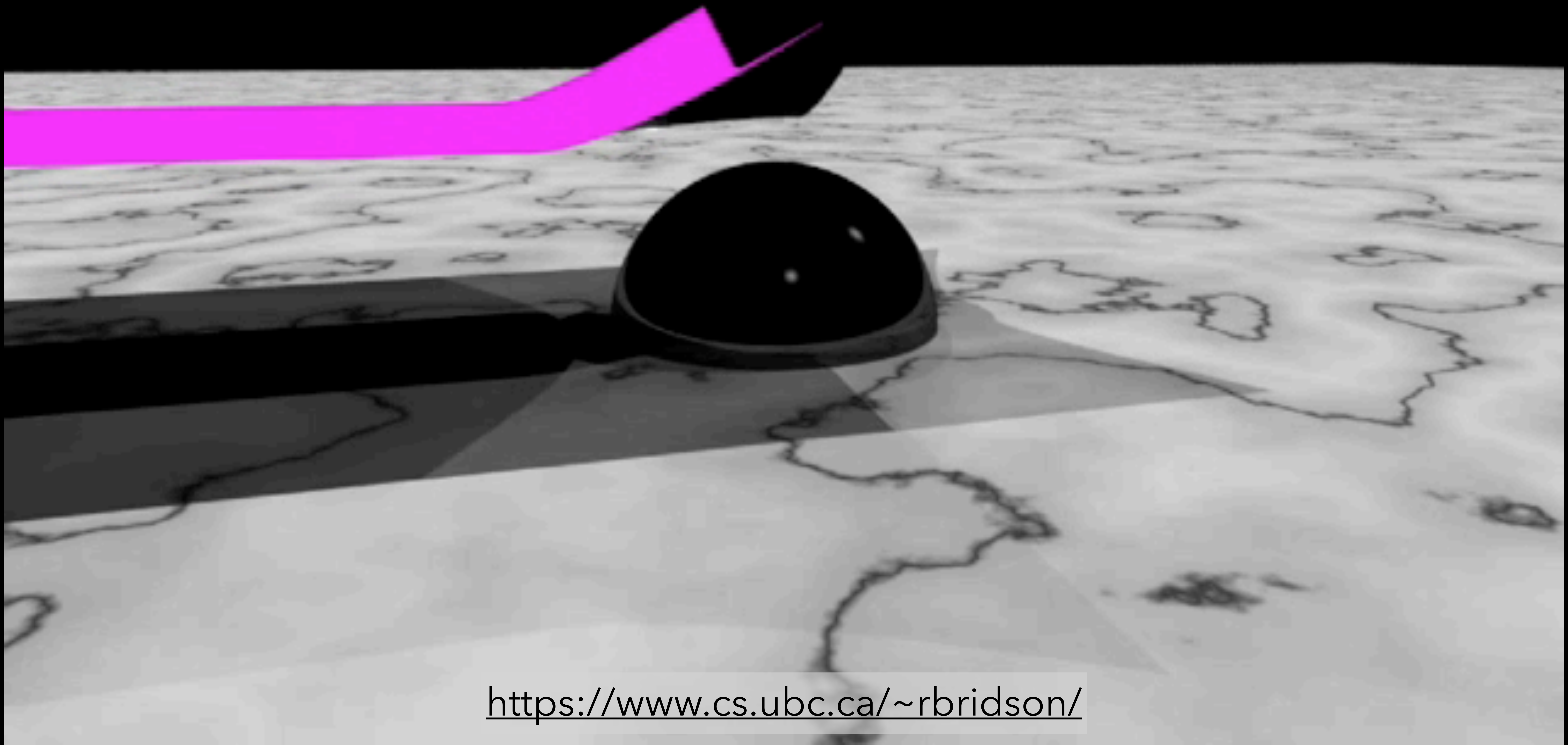
Simulation loop:

- Sum up forces $\mathbf{f}$ and torques $\boldsymbol{\tau}$
- Update velocities $\mathbf{v}$, $\boldsymbol{\omega}$
- Update DOFs $\mathbf{x}$, $\mathbf{q}$. Don't forget to normalize $\mathbf{q}$




https://commons.wikimedia.org/wiki/File:Tennis_racket_theorem.gif

# Collisions

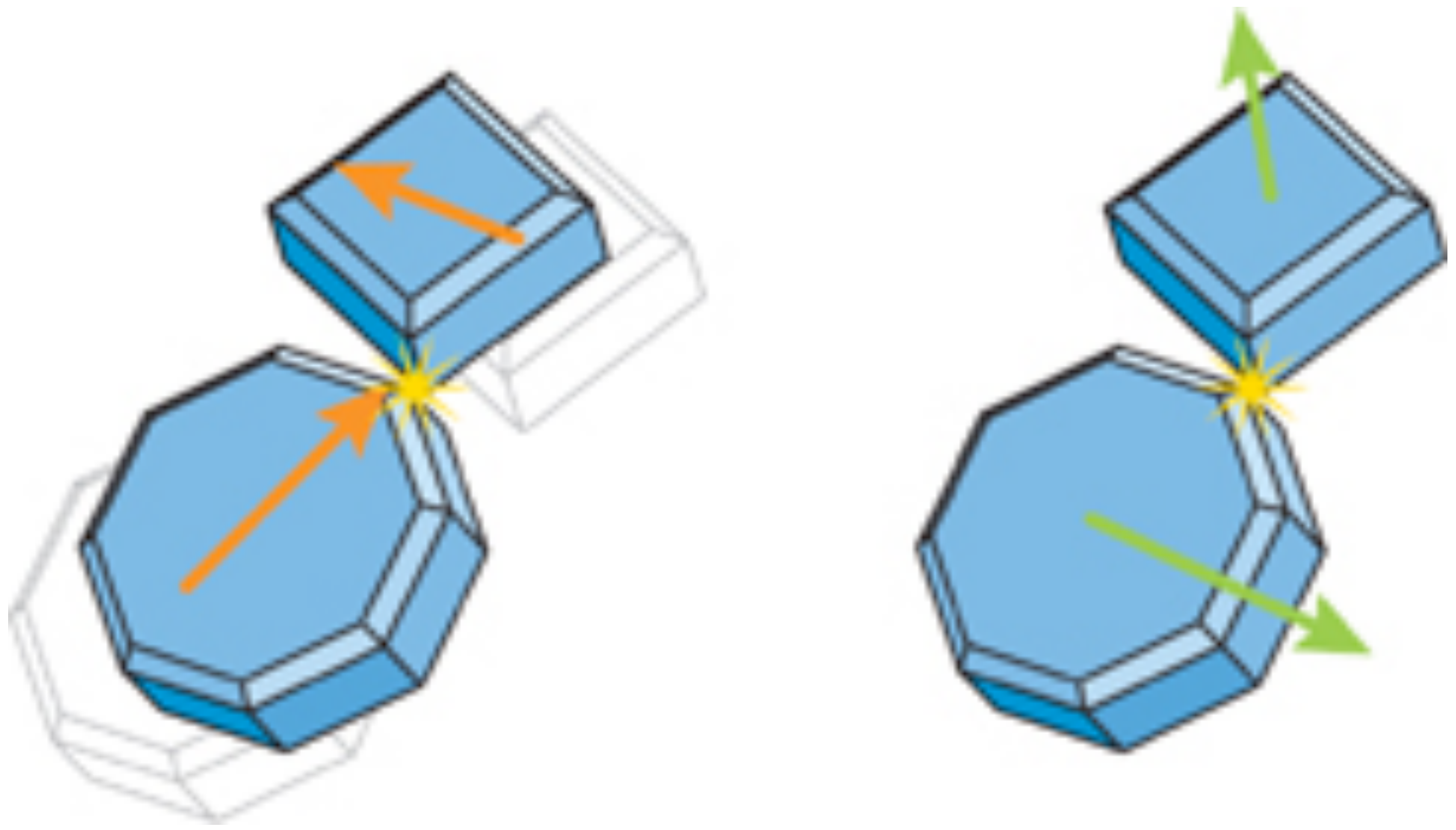


https://www.cs.ubc.ca/~rbridson/

Bridson et al. 2002

**Collision detection:** find out which particles / bodies / etc. are colliding

Purely a geometric problem

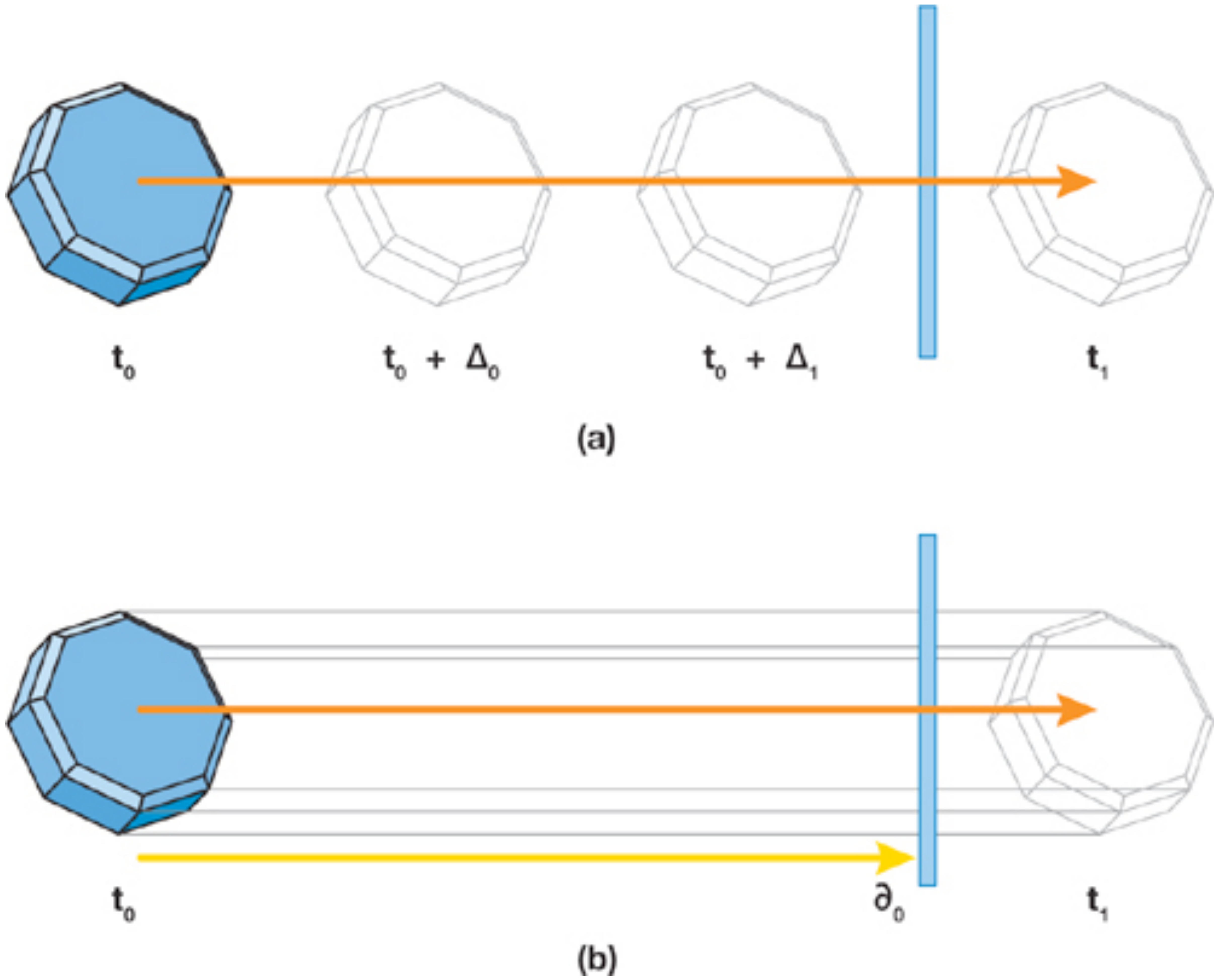


Peter Kipfer

**Collision response:** figure out how to update their velocities / positions

Involves physics of contact forces, friction, etc.

# Collision detection: discrete vs. continuous




Peter Kipfer

**Example:** Suppose I have an infinite cylinder along the $x$-axis with radius $R$.

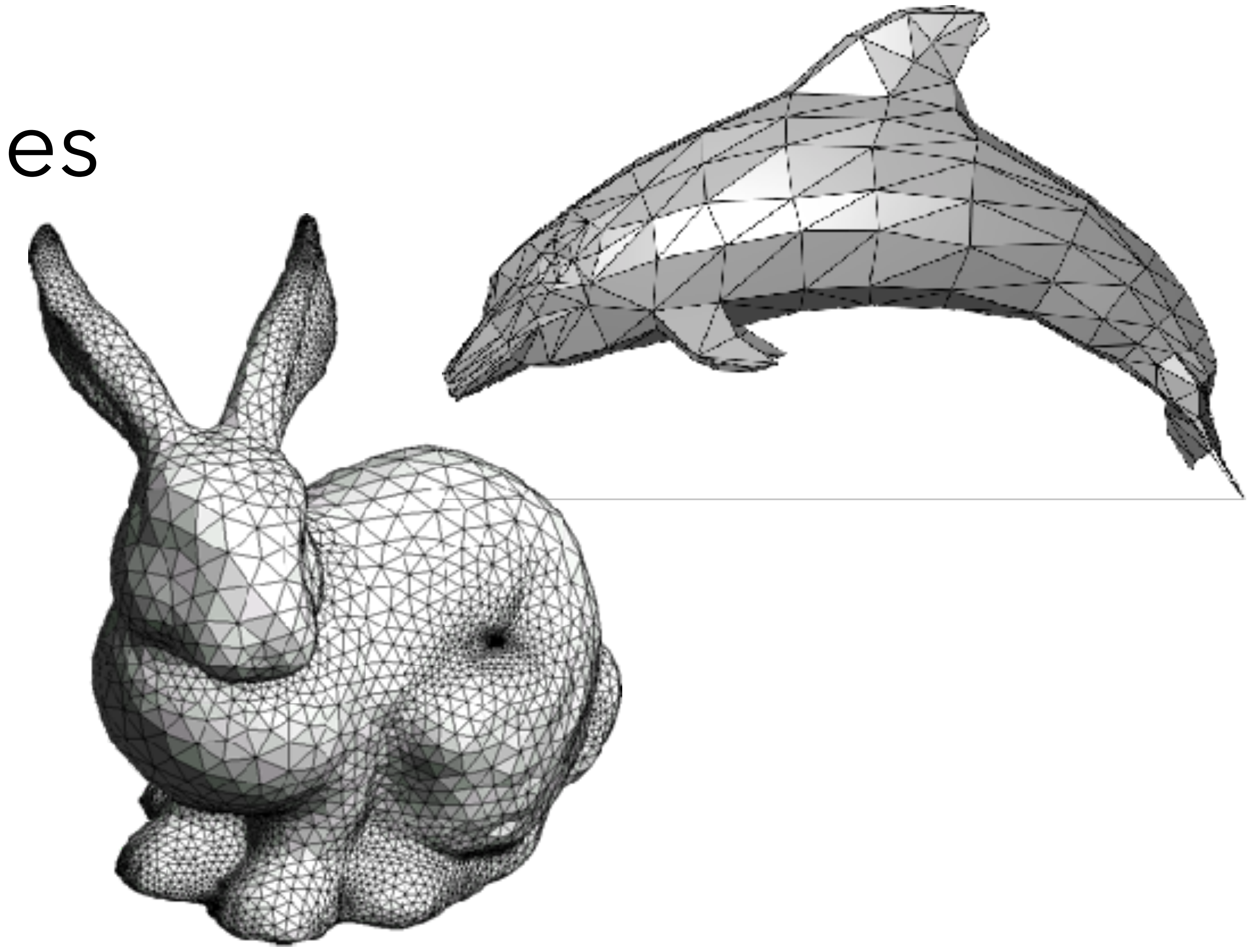
I also have a particle with radius $r$ moving to positions $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots$ at times $t_0, t_1, t_2, \ldots$

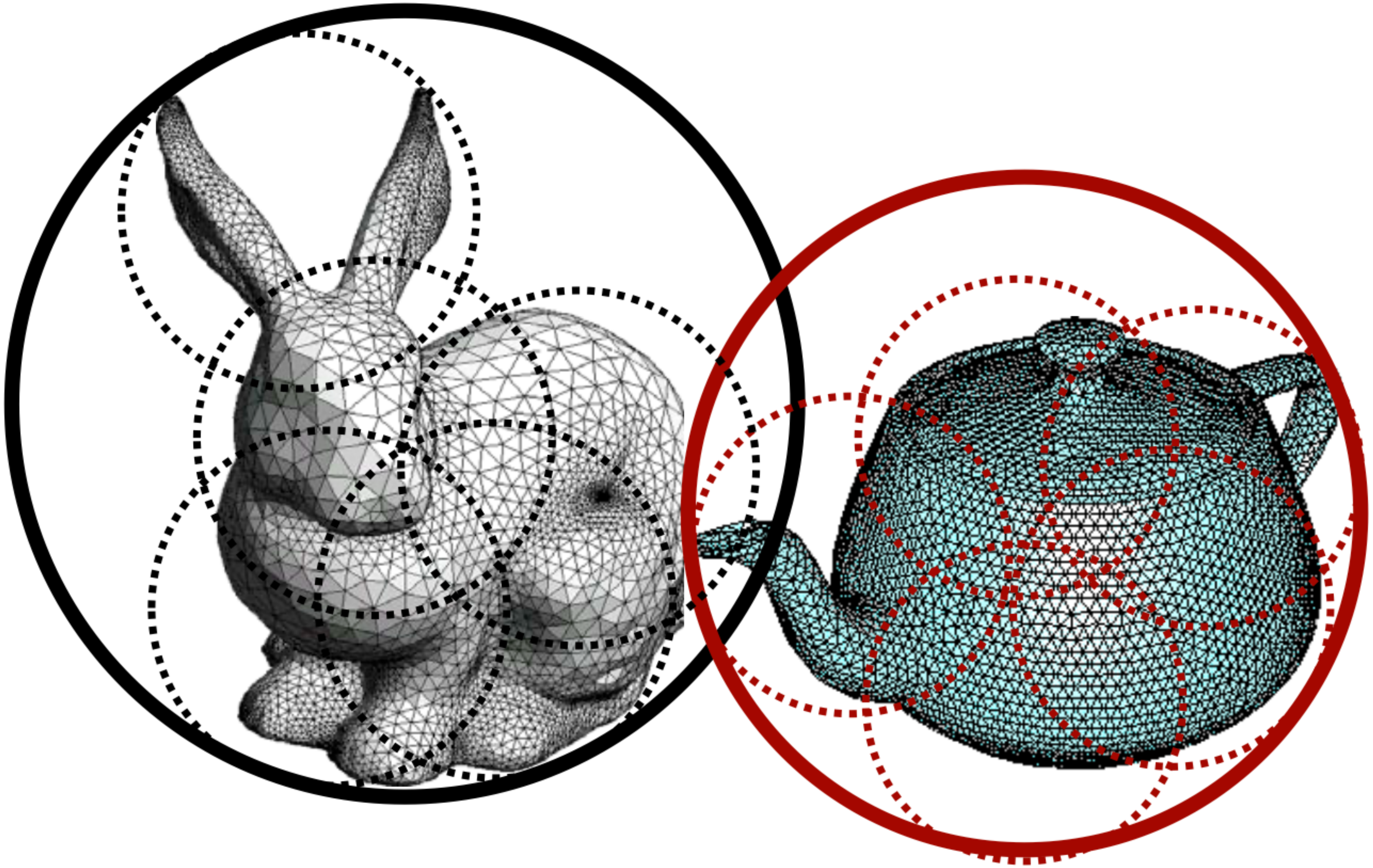
1. How can I do discrete collision detection between the particle and the cylinder?
2. How can I do continuous collision detection for the same?
3. If I model a sheet of cloth as a mass-spring system, is it enough to check that none of the particles are colliding with the cylinder?

How to efficiently detect collisions between complicated shapes without $O(n^2)$ intersection tests?

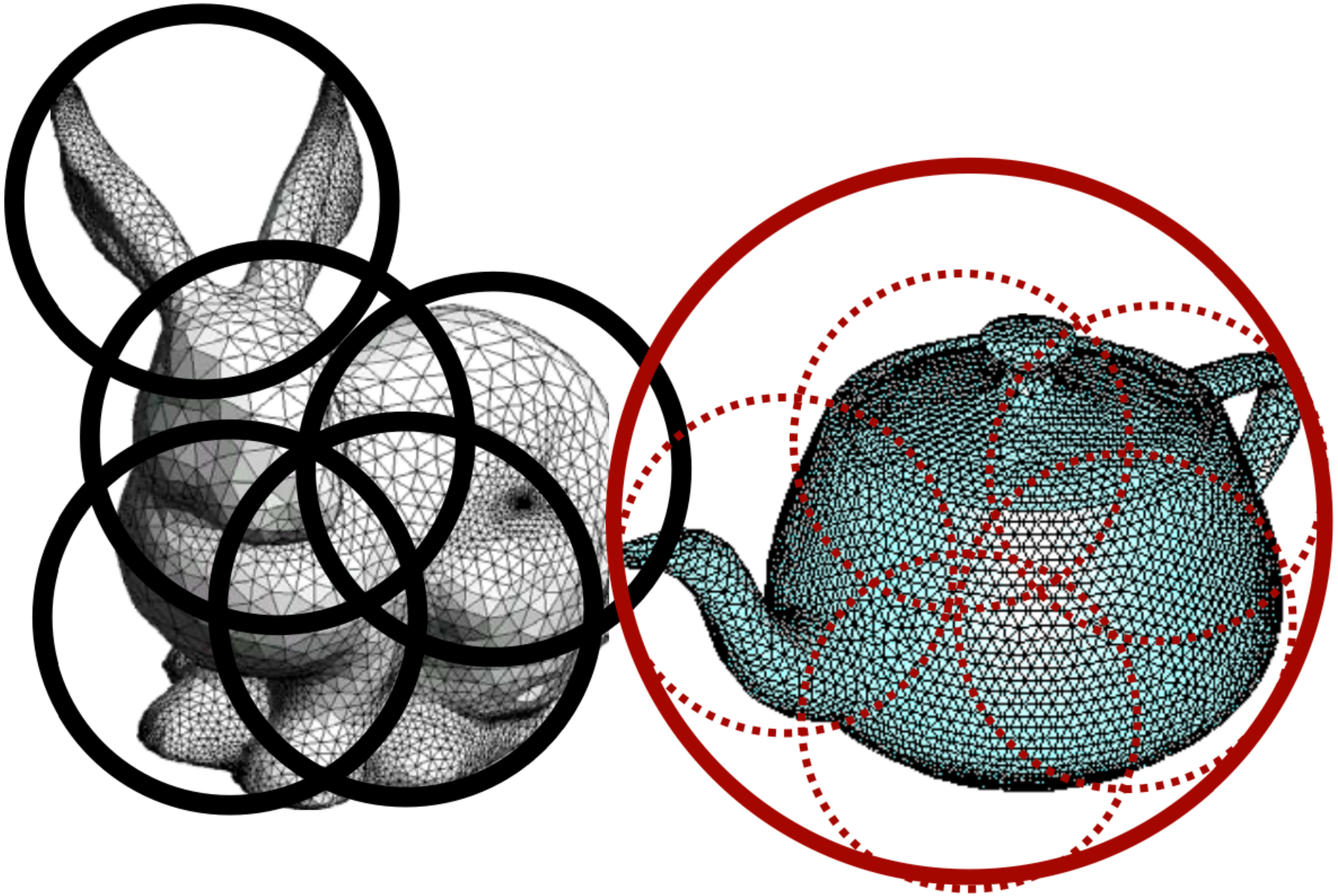
1. **Broad phase**: traverse BVHs of both shapes

2. **Narrow phase**: if BVH leaves intersect, do pairwise intersection tests between primitives

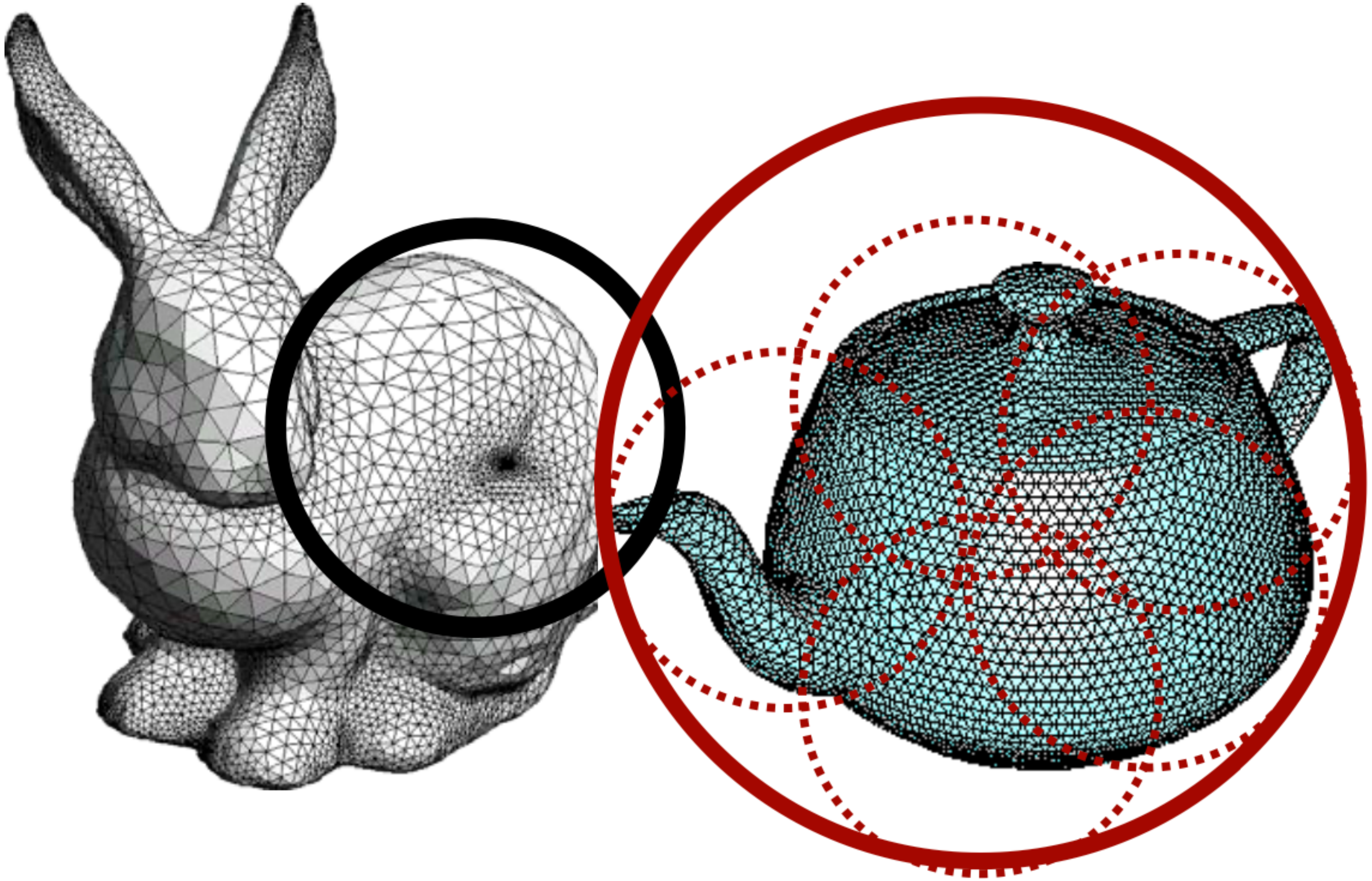




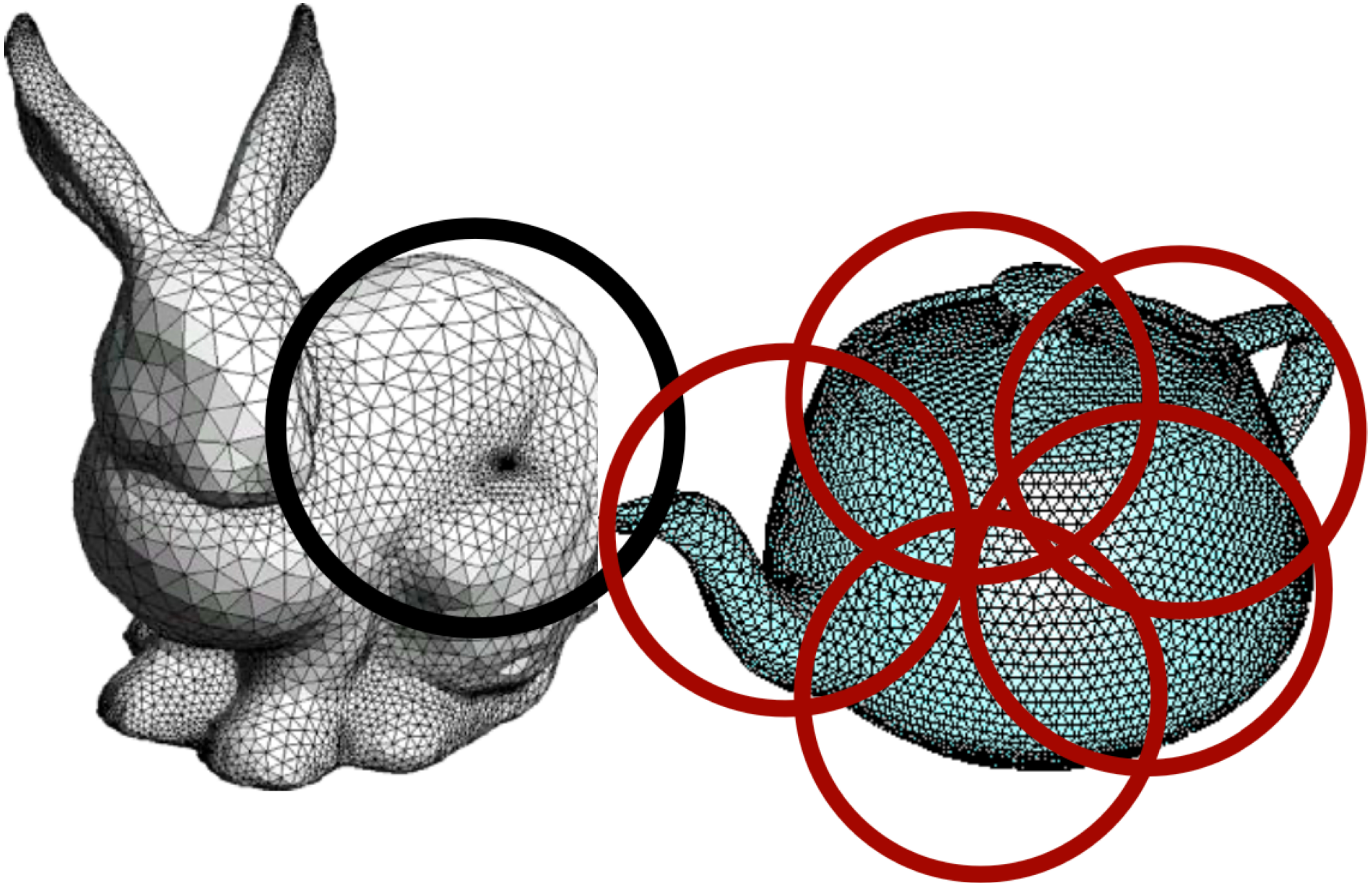
Wojciech Matusik



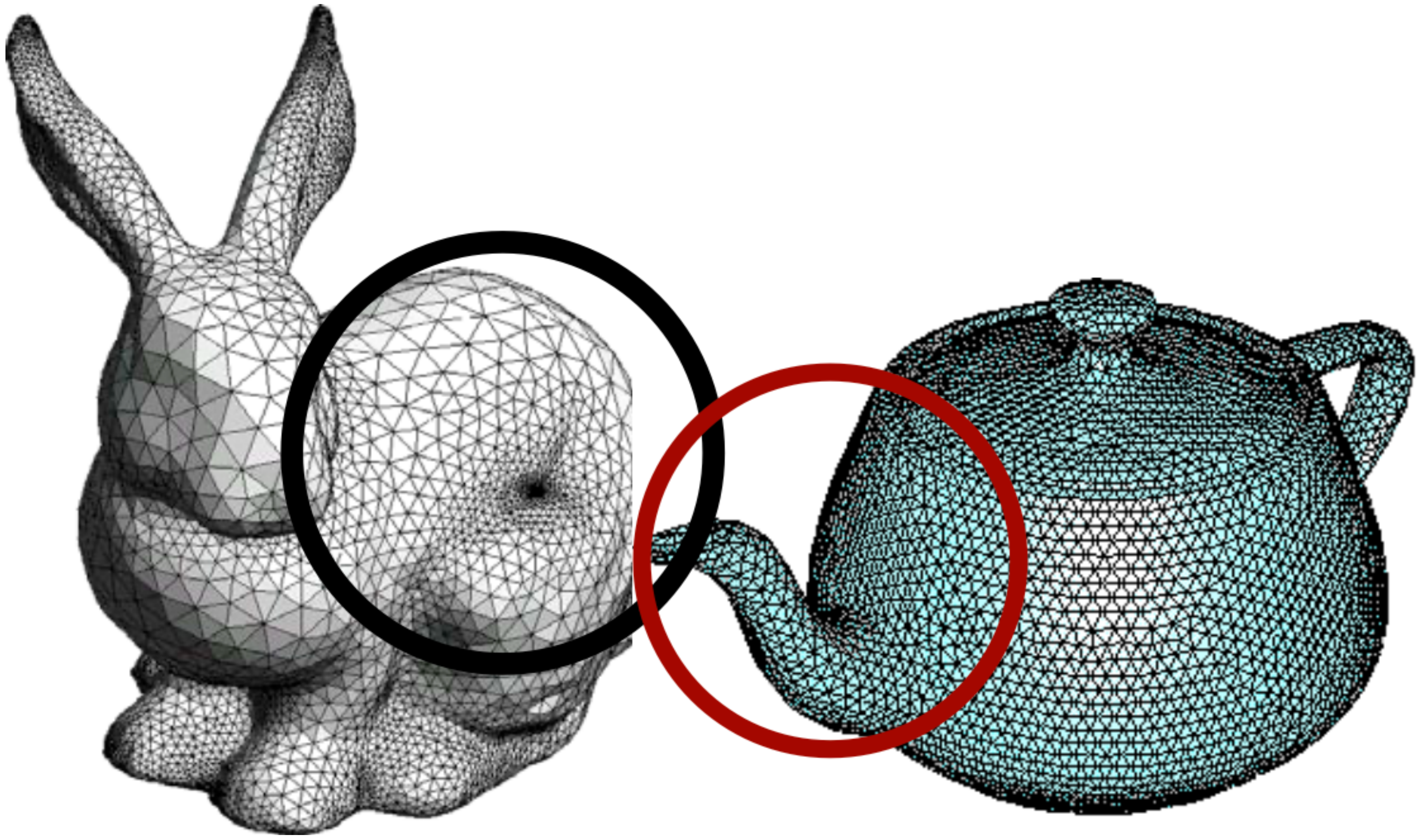
Wojciech Matusik



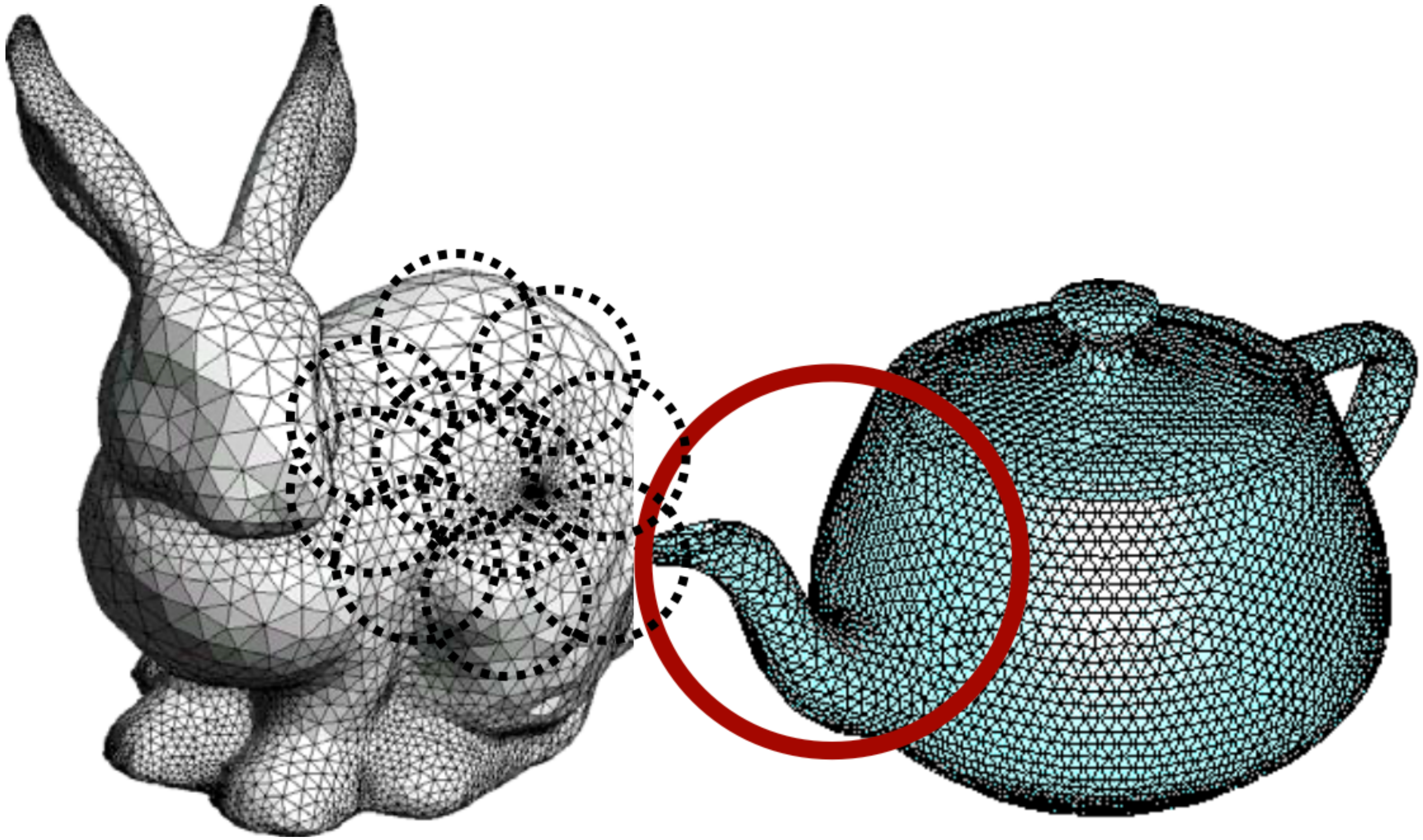
Wojciech Matusik



Wojciech Matusik

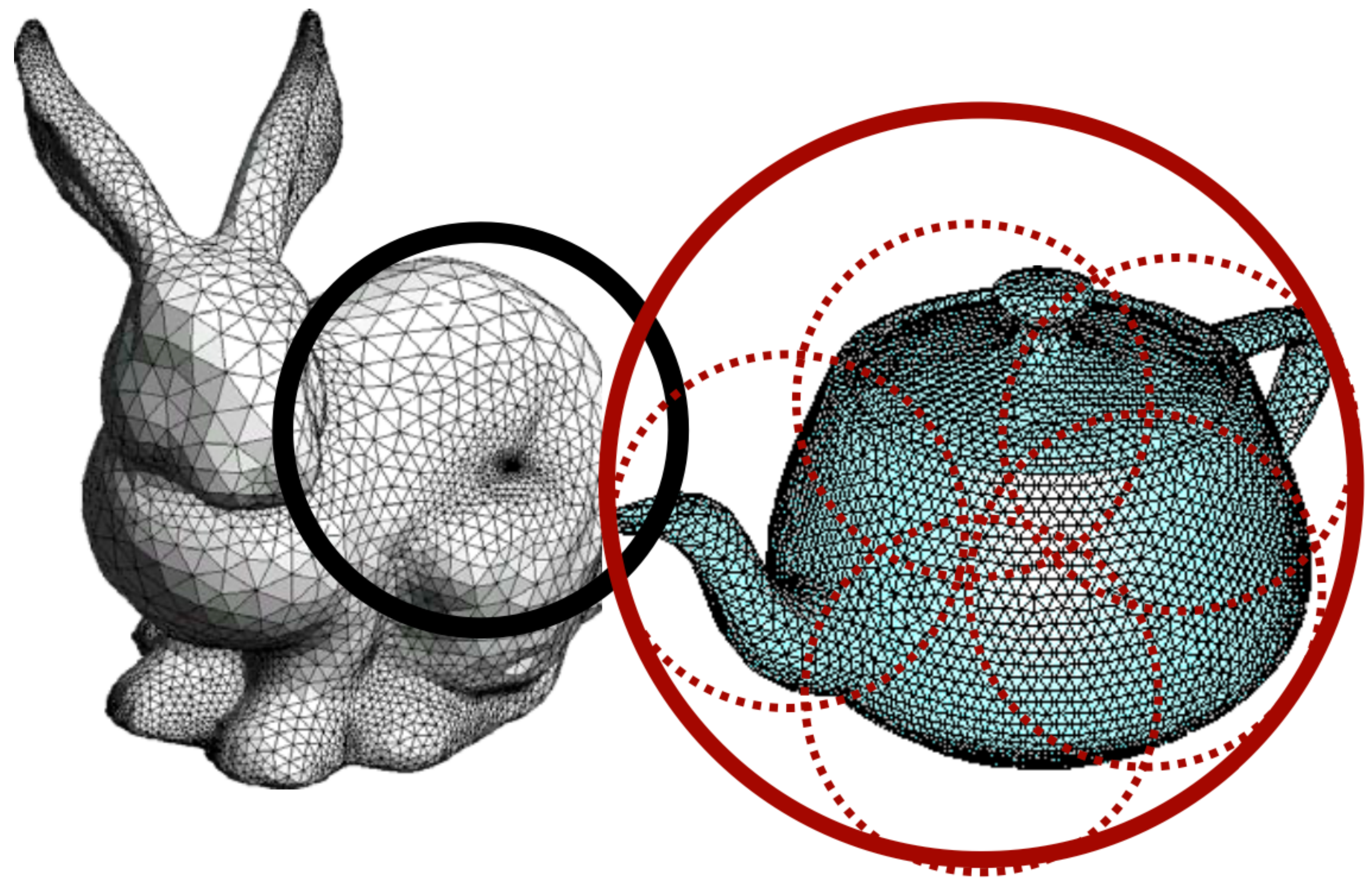


Wojciech Matusik

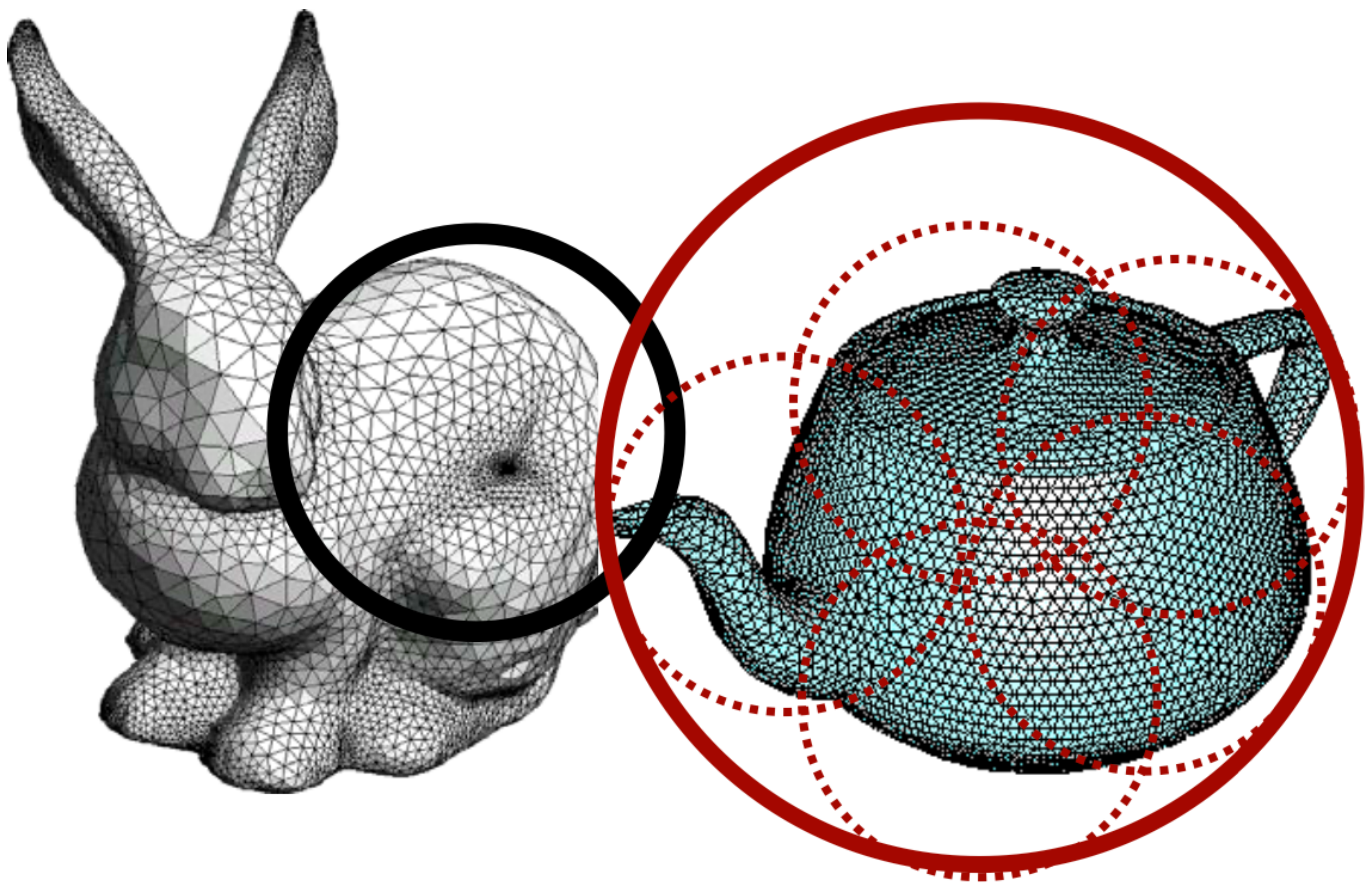


Wojciech Matusik

```
FindIntersections(node₁, node₂):
    if BVs of node₁ and node₂ overlap:
        for each child of bigger node:
            FindIntersections(child, smaller node)
```



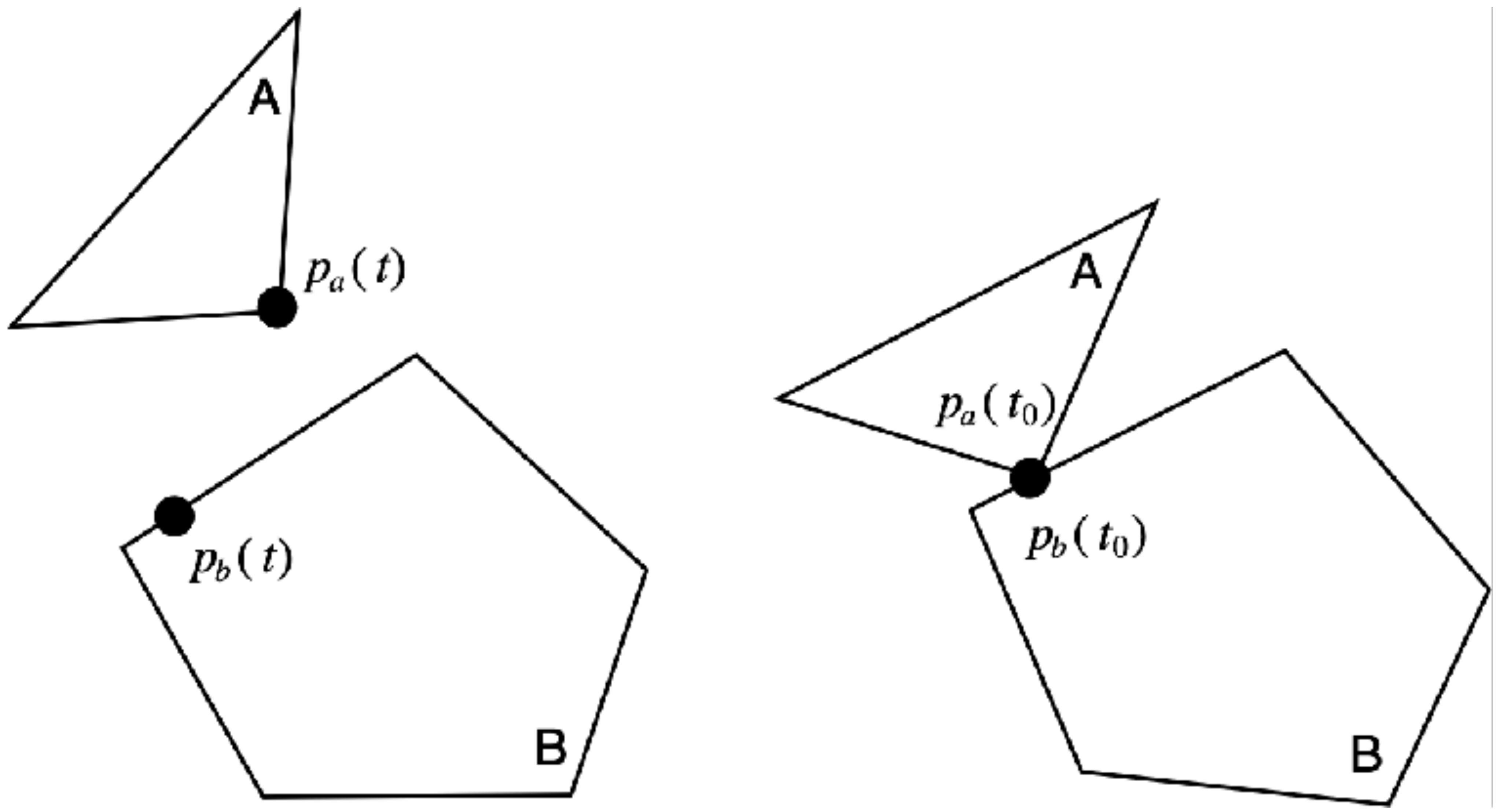
```
FindIntersections(node₁, node₂):
    if BVs of node₁ and node₂ overlap:
        if neither node₁ nor node₂ are leaves:
            for each child of bigger node:
                FindIntersections(child, smaller node)
        else if one is a leaf:
            for each child of non-leaf:
                FindIntersections(child, leaf node)
        else (both are leaves):
            test intersections between all pairs of primitives
```



Output of collision detection: **contact pairs**

- Point $\mathbf{p}_a$ on one body
- Point $\mathbf{p}_b$ on other body
- Contact normal $\mathbf{n}$
- Time of impact $t^*$

Now, what to do with this information?
**Collision resolution**

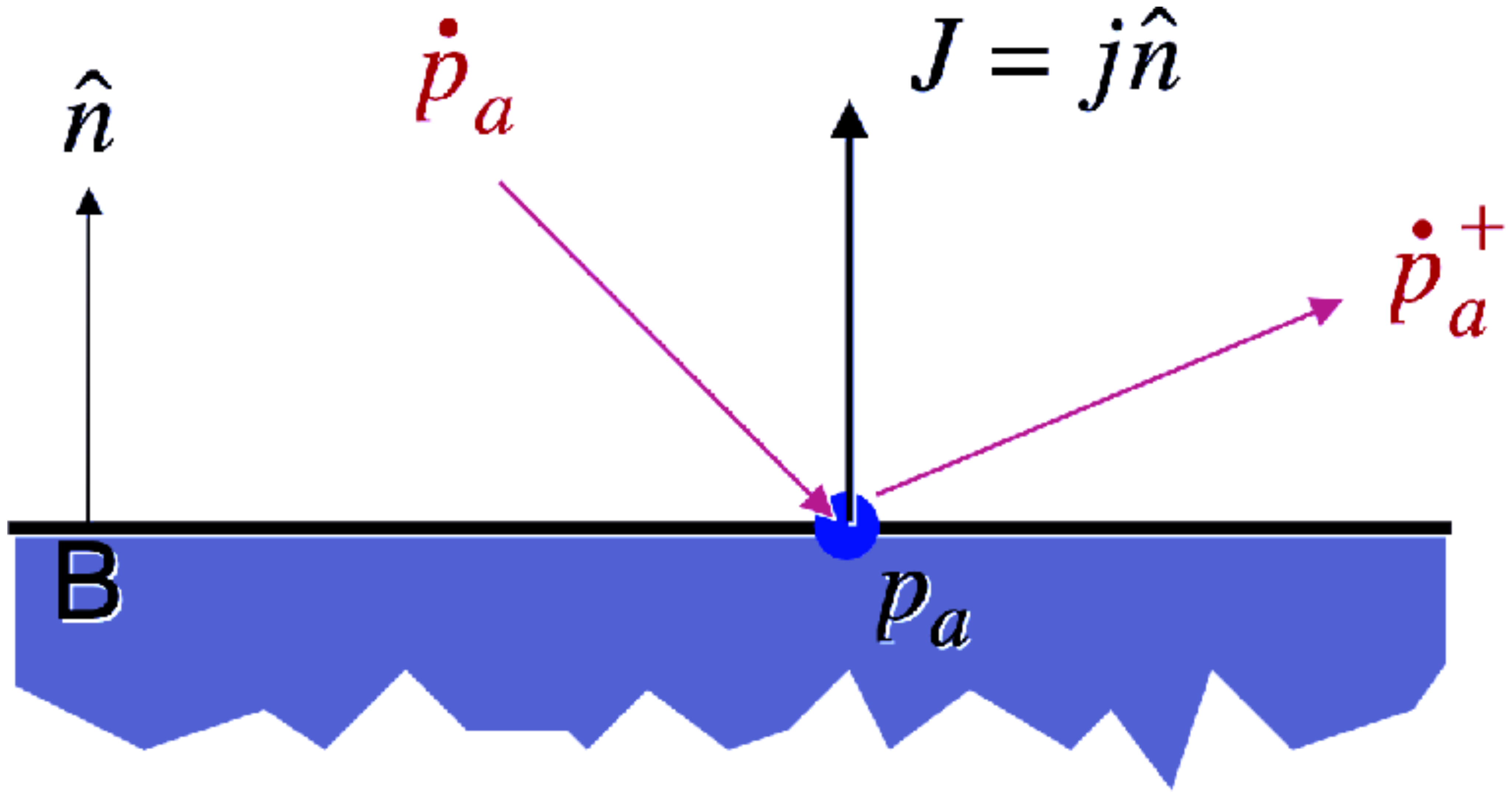



Witkin & Baraff 2001

# Collision resolution

Two components:

- Normal force (prevents interpenetration)
- Frictional force (opposes tangential sliding)



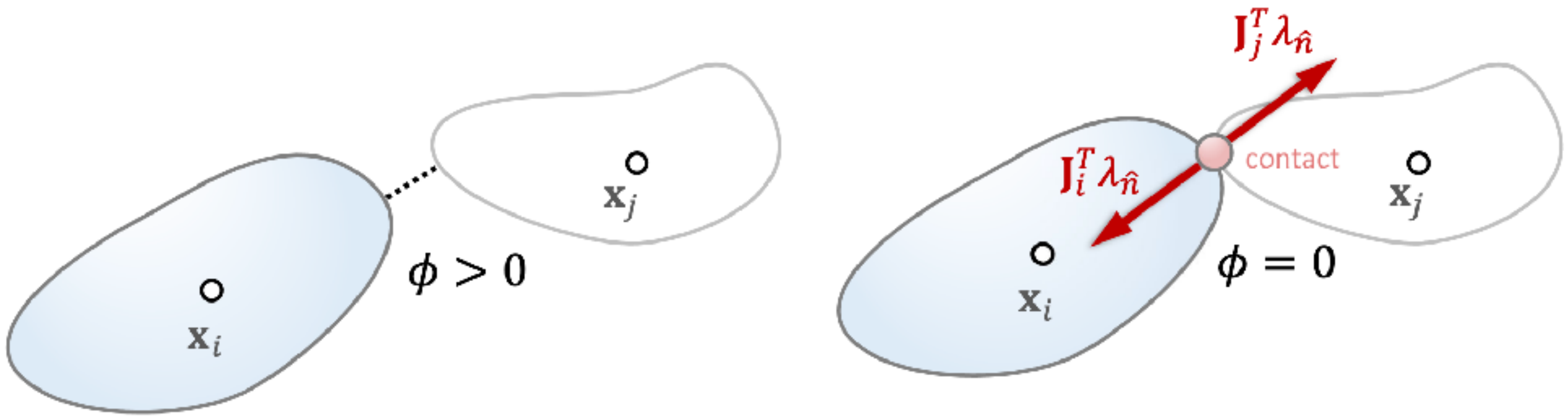

Witkin & Baraff 2001

Actually, collision forces change velocity over
an extremely very short time → treat as an instantaneous impulse

$$\mathbf{v}^+ = \mathbf{v} + m^{-1}\,\mathbf{j}$$

The normal component is like a constraint force.

Define a **gap function** $\varphi(\mathbf{q})$ which measures the distance between the bodies




Andrews et al. 2002

Constraint: $\varphi(\mathbf{q}) \geq 0$

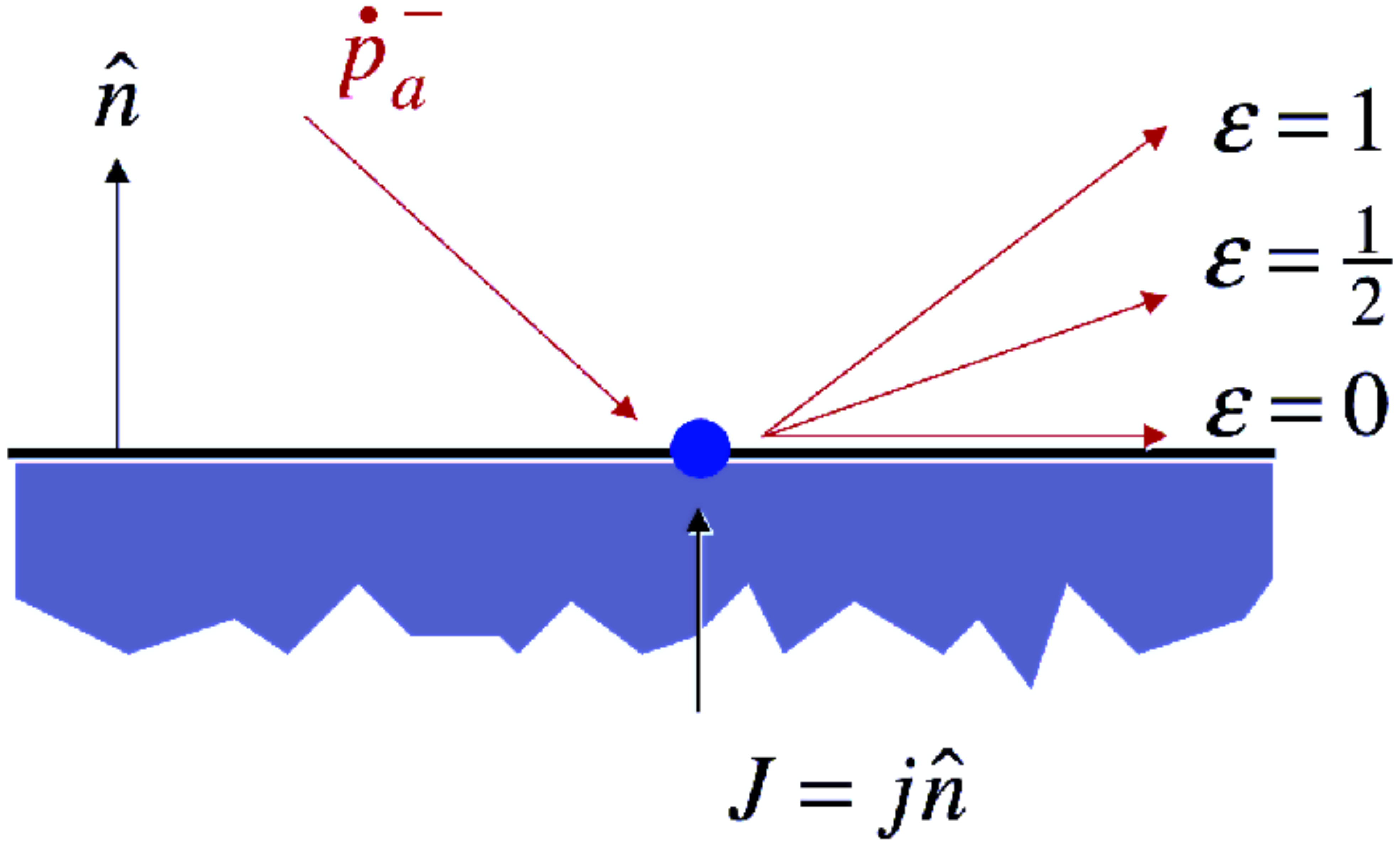
Normal impulse: $\mathbf{j} = \lambda \, \nabla\varphi(\mathbf{q})$, $\lambda \geq 0$ (no sticking)

**Complementarity**: if $\varphi(\mathbf{q}) > 0$ then $\lambda = 0$, if $\lambda > 0$ then $\varphi(\mathbf{q}) = 0$

$$0 \leq \varphi(\mathbf{q}) \quad \perp \quad \lambda \geq 0$$

Coefficient of restitution $\varepsilon$: how elastic the collision is

$$\mathbf{n} \cdot \mathbf{v}^{+} = -\varepsilon\,(\mathbf{n} \cdot \mathbf{v})$$



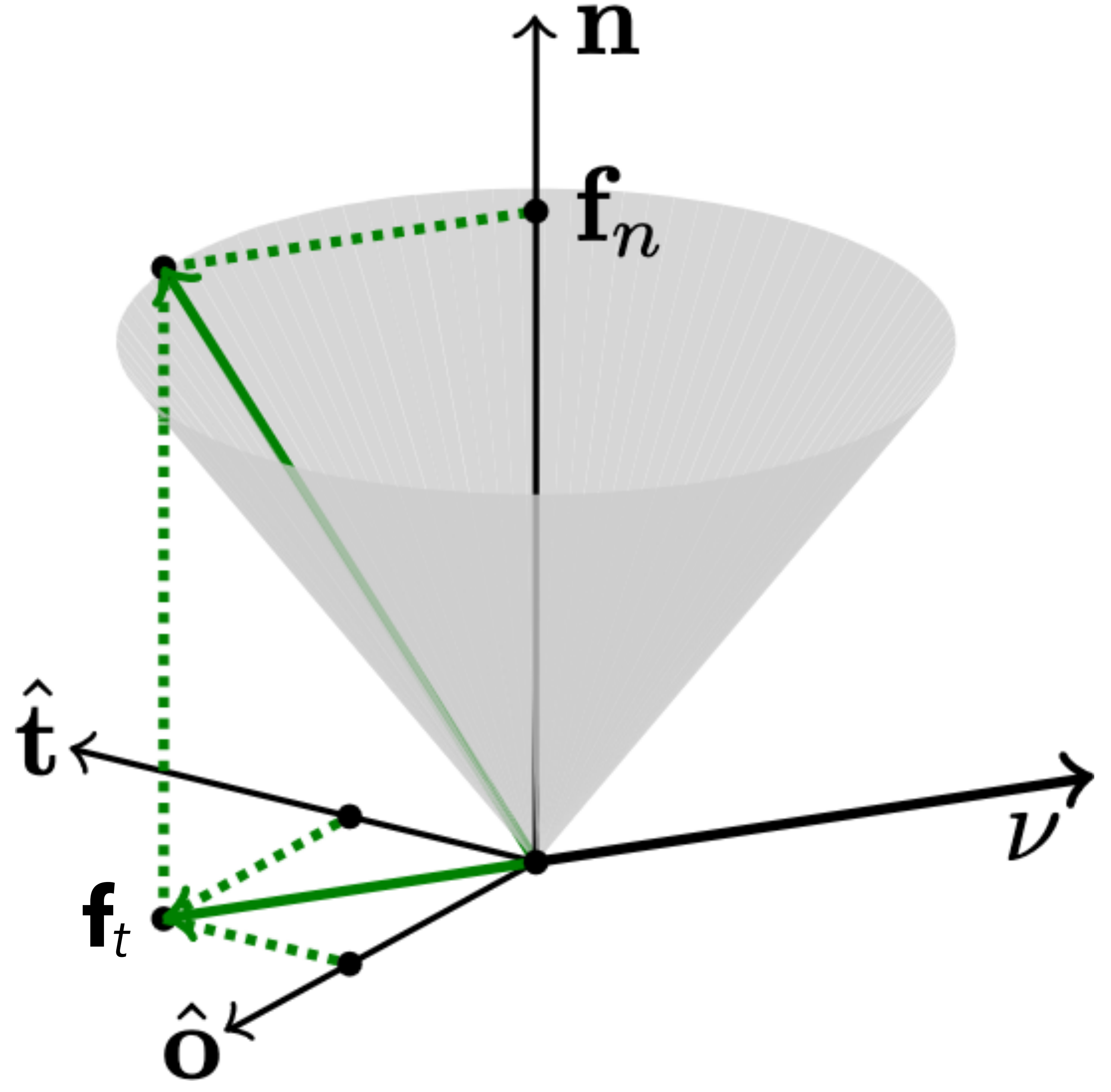

Witkin & Baraff 2001

Friction is described by Coulomb's law

$$\|\mathbf{f}_t\| \leq \mu\, \mathbf{f}_n$$

**Maximum dissipation principle**: Frictional force takes the value which dissipates as much kinetic energy as possible.

1. If $\|\mathbf{v}_t\| > 0$ (slipping) then $\mathbf{f}_t = -(\mu\, \mathbf{f}_n)\, \hat{\mathbf{v}}_t$
2. If $\|\mathbf{v}_t\| = 0$ (sticking) then $\mathbf{f}_t$ is any force in friction cone



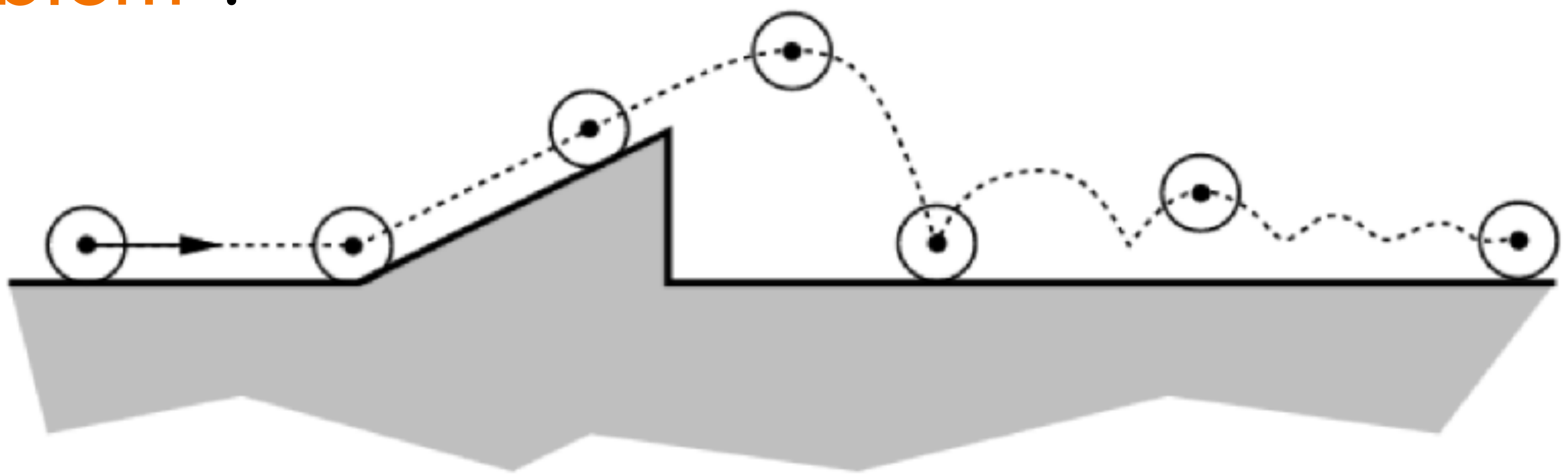

Bender et al. 2012

# Time stepping issues

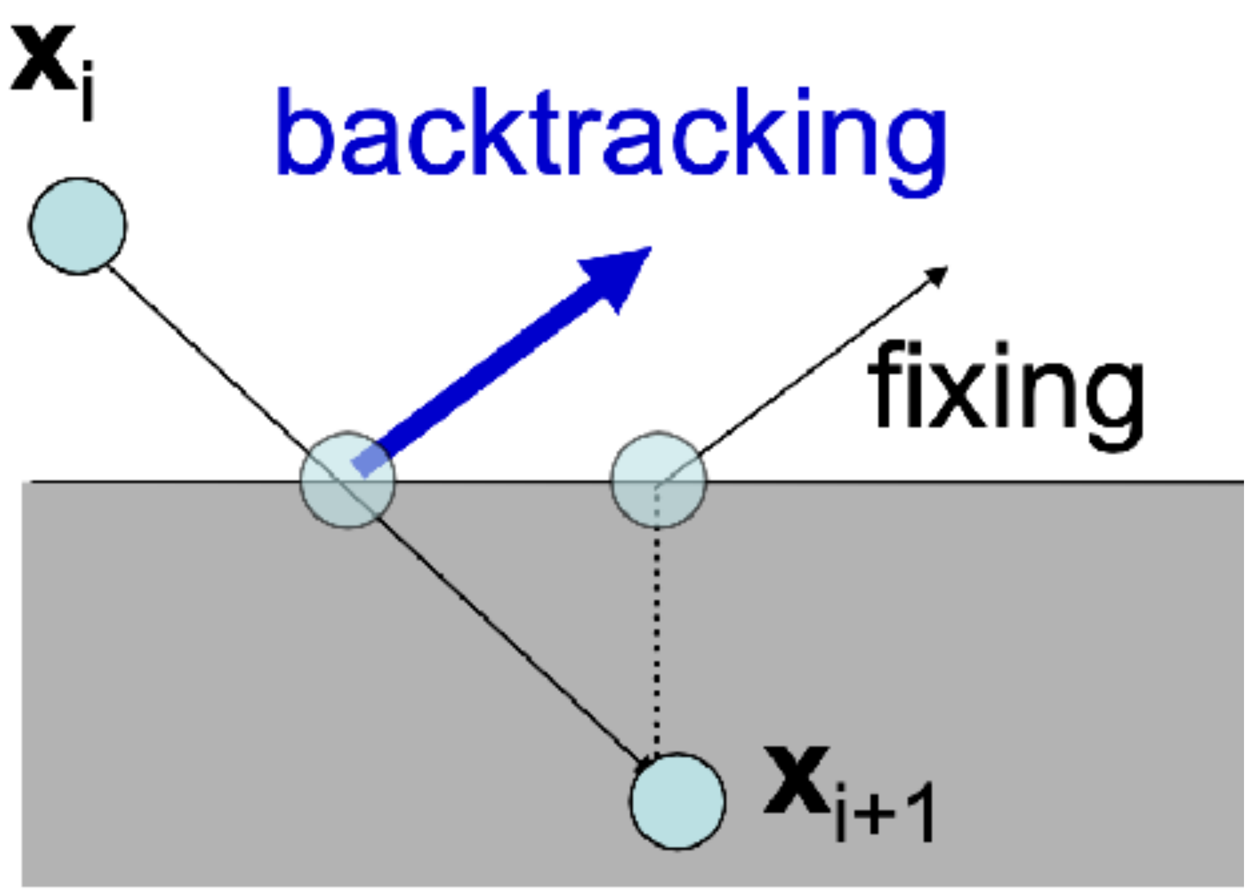
We usually only detect collisions after they've already happened!

- Option 1: Go back to time of impact,
  compute response, step forward for fraction of $\Delta t$

  "Zeno problem":



Mirtich & Canny 1994




Wojciech Matusik

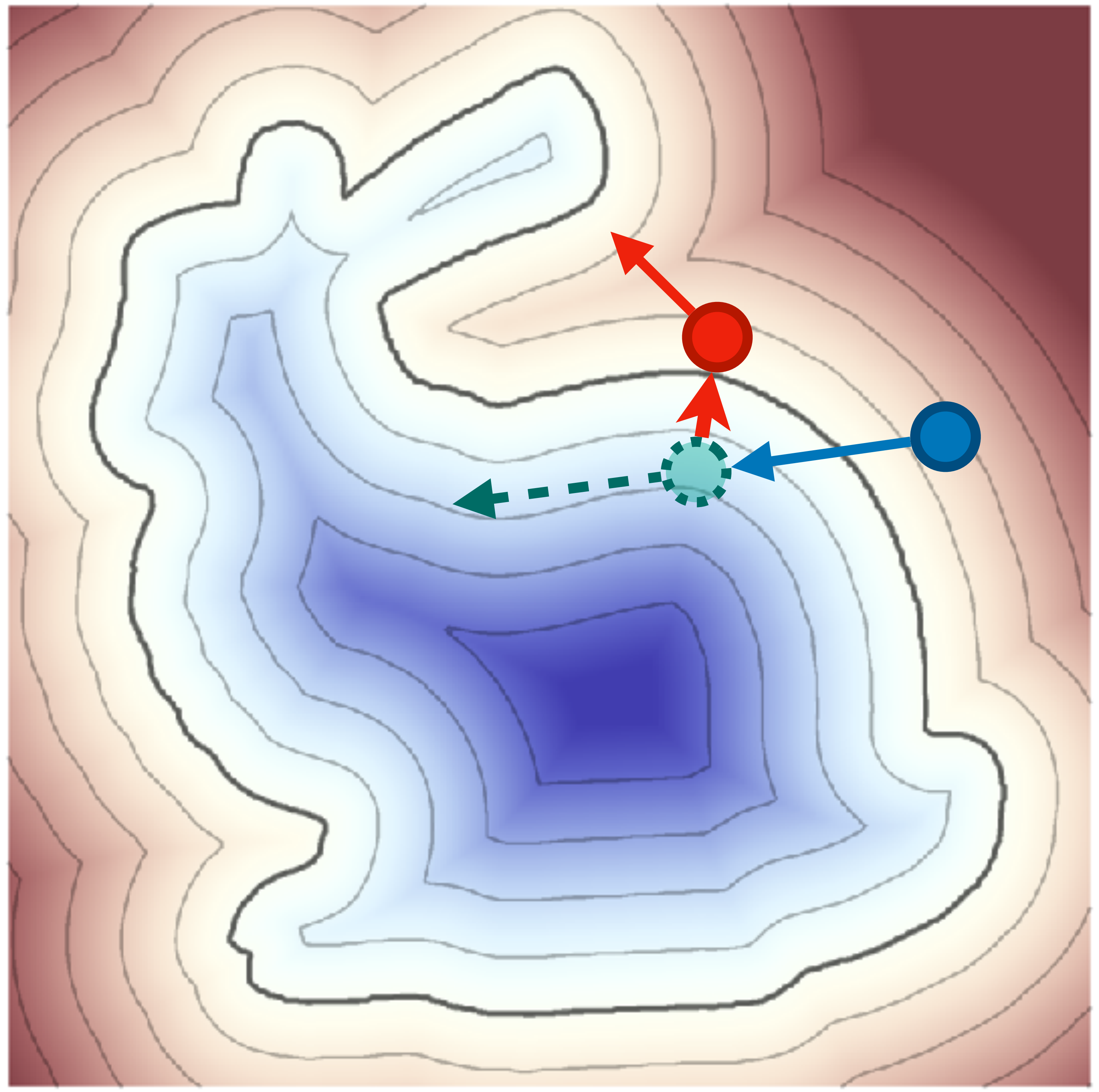
- Option 2: Just lie about it! Project end-of-step positions to remove interpenetration

A simple strategy for particle/implicit collisions:

Perform $\mathbf{v}$, $\mathbf{x}$ update as usual

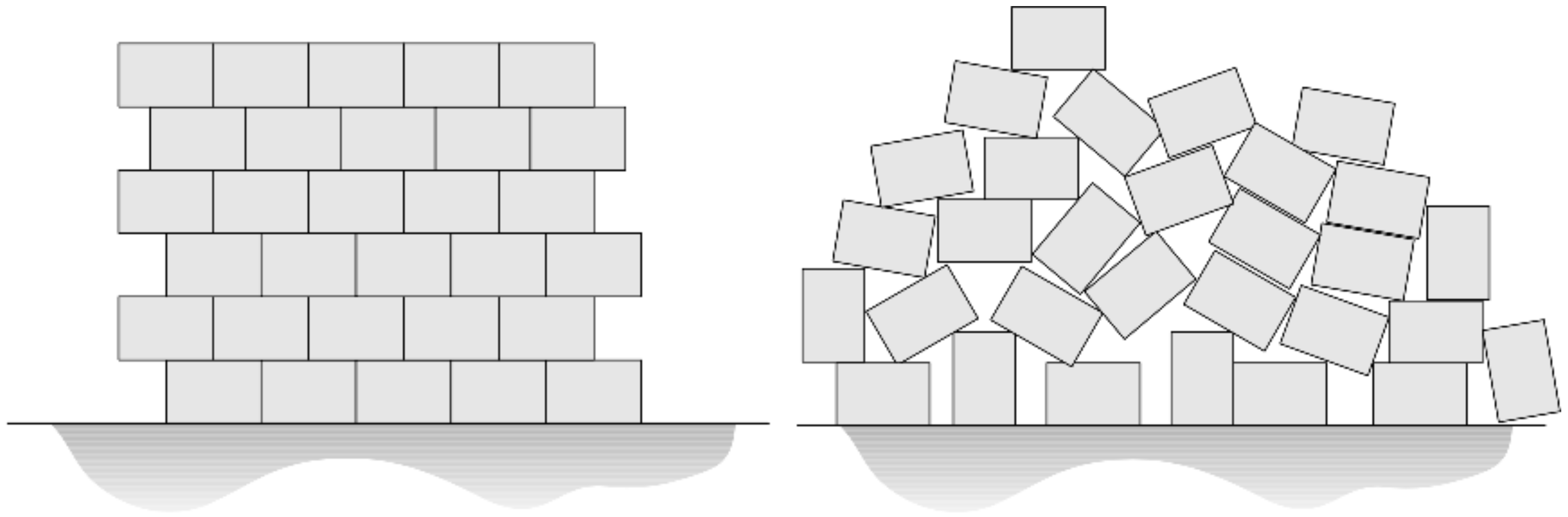
If inside obstacle ($\varphi(\mathbf{x}) < 0$):

- If velocity is also inwards ($\mathbf{n} \cdot \mathbf{v} < 0$):
  - Compute normal impuse: $\mathbf{j}_n = -(1 + \varepsilon)\, m\, \mathbf{v}_n$
  - Compute tangential impulse: $\mathbf{j}_t = -\min(\mu \, \|\mathbf{j}_n\|, m \, \|\mathbf{v}_t\|) \, \hat{\mathbf{v}}_t$
  - Update velocity: $\mathbf{v} \mathrel{+}= m^{-1} (\mathbf{j}_n + \mathbf{j}_t)$
- Compute position correction: $\Delta\mathbf{x}_n = -\varphi(\mathbf{x})\, \mathbf{n}$
- Project particle out: $\mathbf{x} \mathrel{+}= \Delta\mathbf{x}_n$

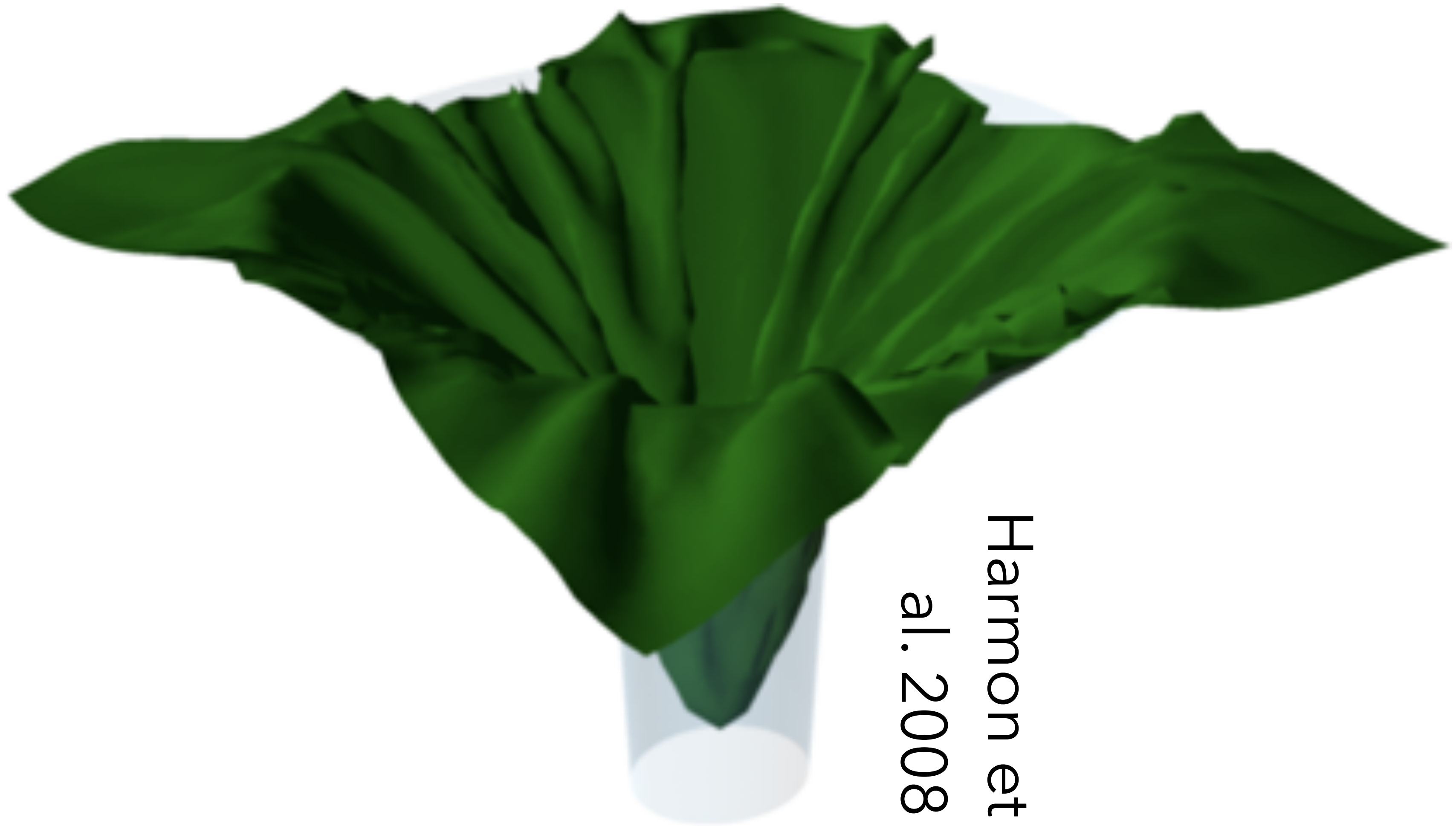


Can also add a tangential position correction to counteract artificial sliding…

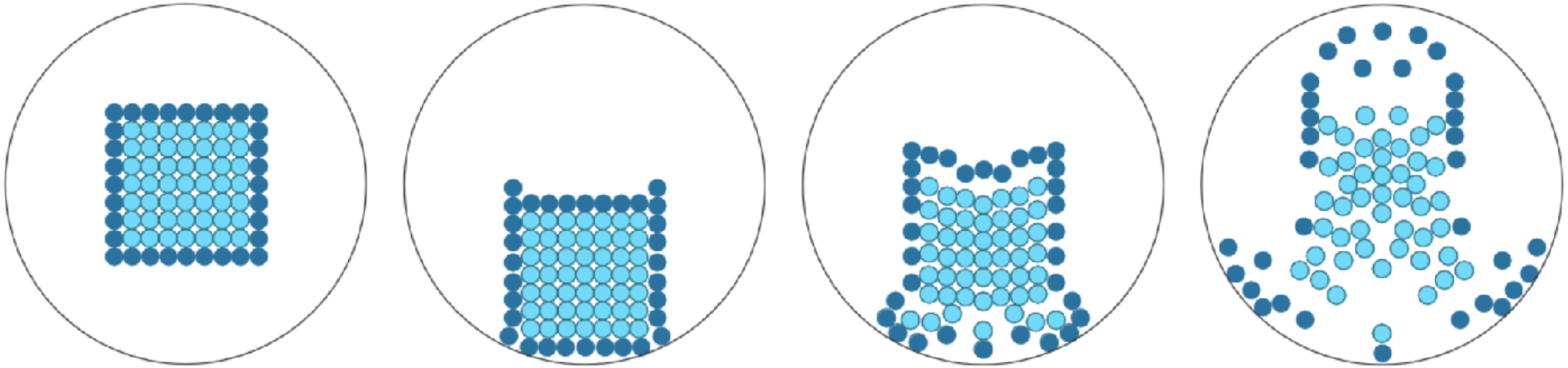
# Multi-contact problems (harder!)



Erleben 2007



Harmon et al. 2008



Smith et al. 2012