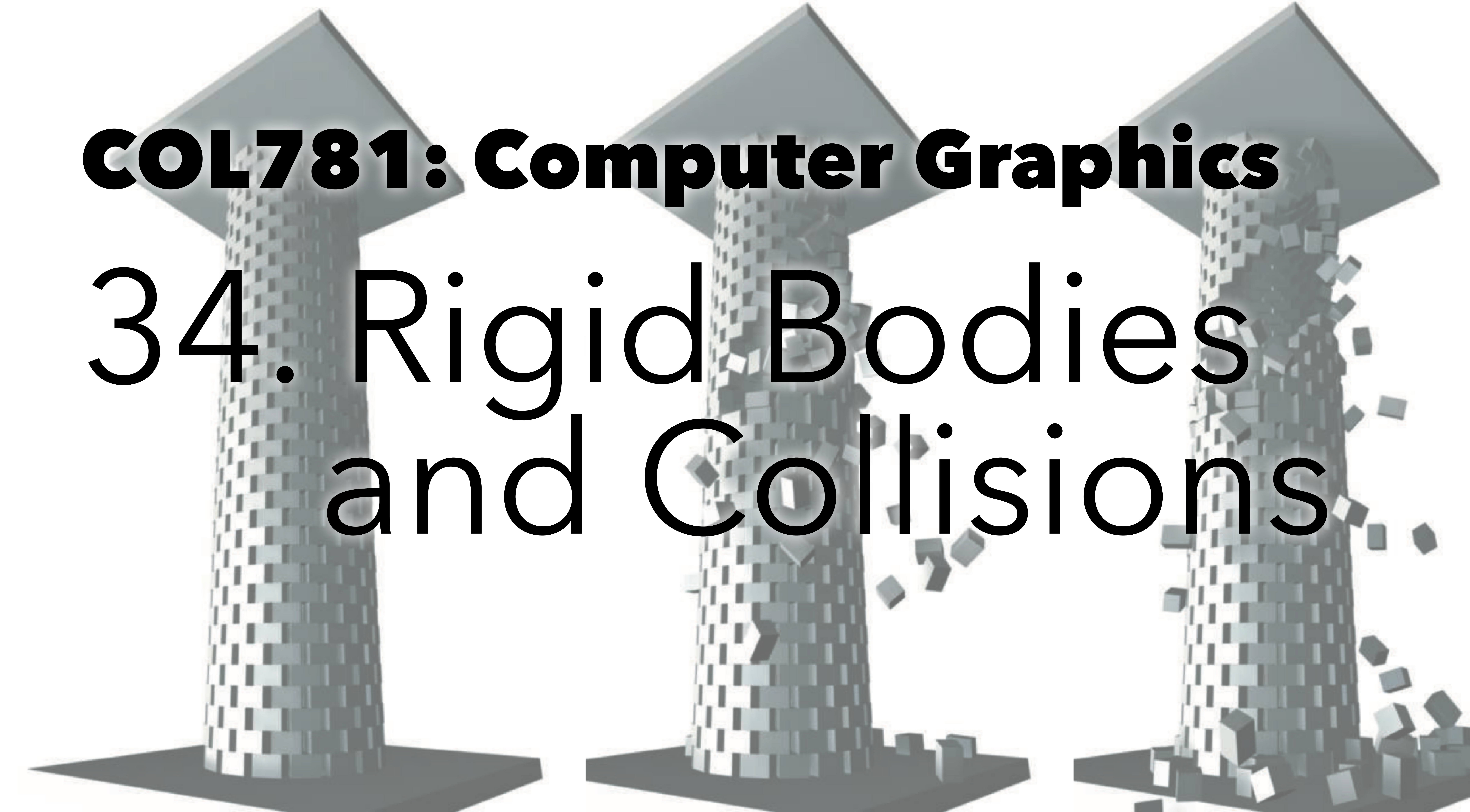
# COL781: Computer Graphics

# 34. Rigid Bodies and Collisions



Erleben 2007

# Announcements

Assignment 3 demos some time next week (TBA)

No class tomorrow (Saturday, 13 April)

# Rigid bodies

Degrees of freedom: Center of mass position $\mathbf{x}$, rotation (matrix $\mathbf{R}$ or quaternion $\boldsymbol{q}$)
…Basically just the body's coordinate system

Kinematics:

- (Linear) velocity: $\mathrm{d}\mathbf{x}/\mathrm{d}t = \mathbf{v}$
- Angular velocity: $\boldsymbol{\omega}$

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

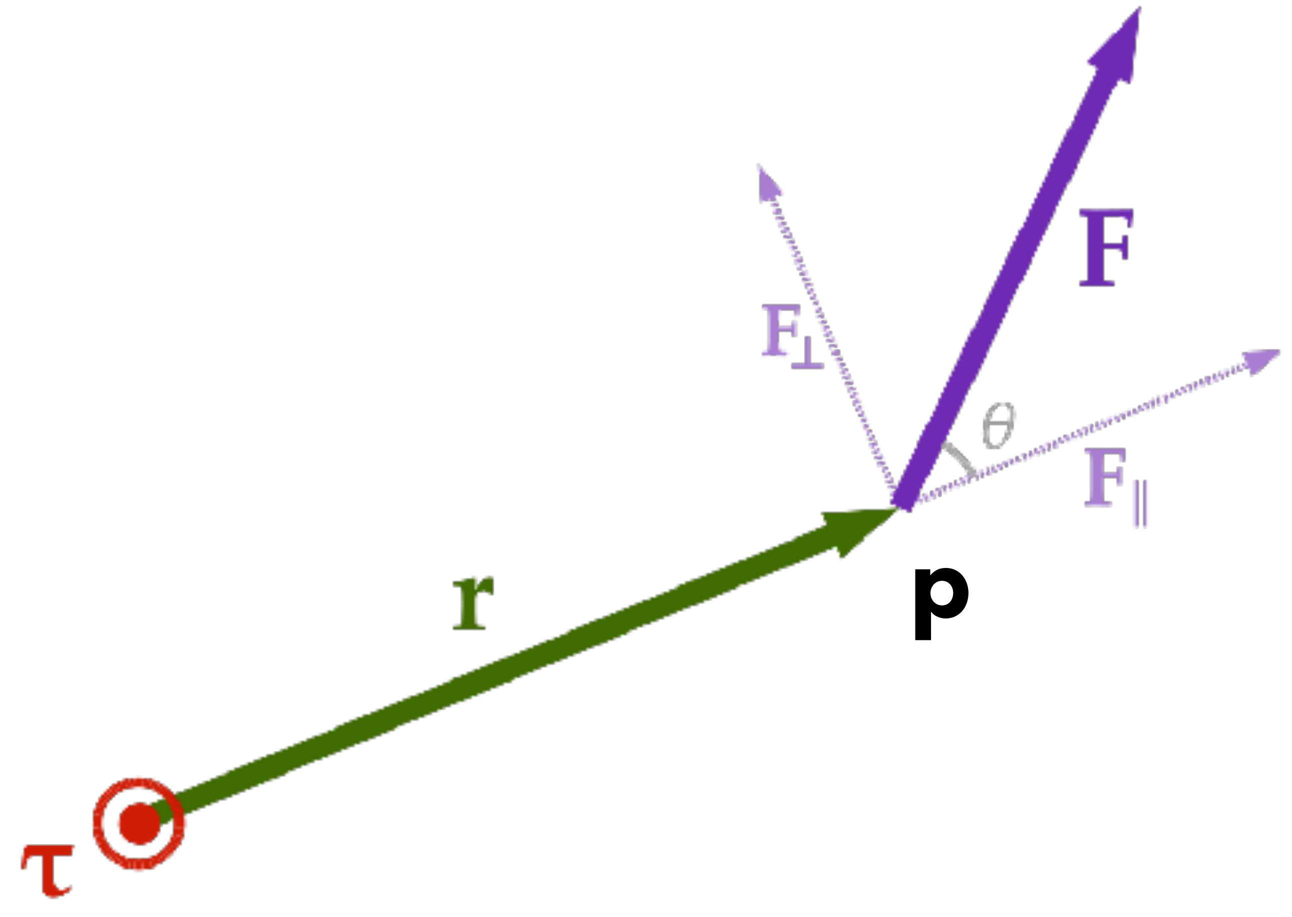
Dynamics:

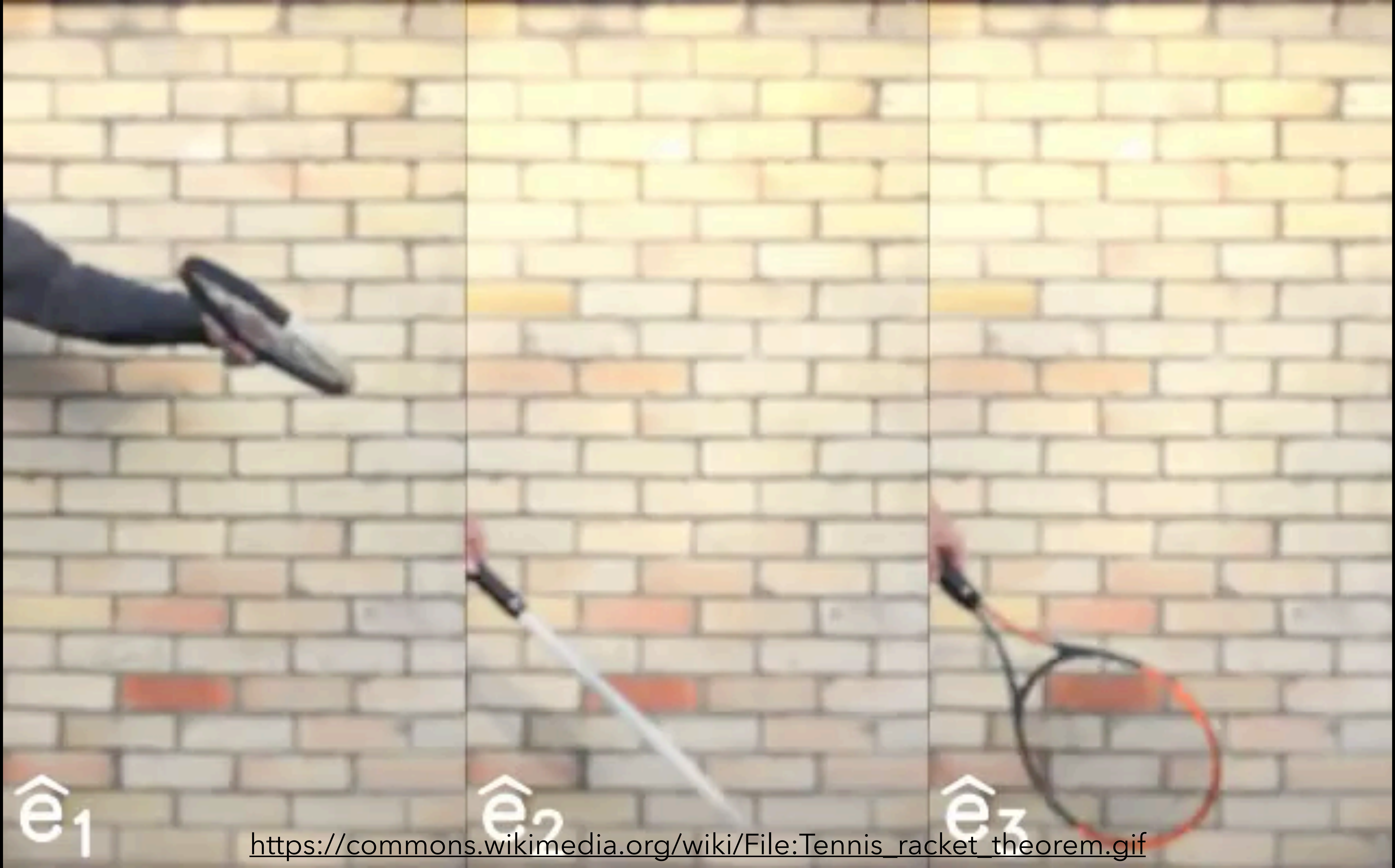
$$d\mathbf{v}/dt = m^{-1}\,\mathbf{f}$$

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

where

- $\mathbf{I}$ = moment of inertia in world space
  $= \mathbf{R}\,\mathbf{I}_0\,\mathbf{R}^T$ where $\mathbf{I}_0$ is moment of inertia in body frame
- $\boldsymbol{\tau}$ = 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






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

A typical simulation loop:

- Sum up forces $\mathbf{f}$ and torques $\boldsymbol{\tau}$
- Update velocities using d$\mathbf{v}$/dt = $m^{-1}\,\mathbf{f}$, d$\boldsymbol{\omega}$/dt = $\cdots$
- Update DOFs using d$\mathbf{x}$/dt = $\mathbf{v}$, d$\boldsymbol{q}$/dt = $\cdots$
- Normalize $\boldsymbol{q}$ to a unit quaternion

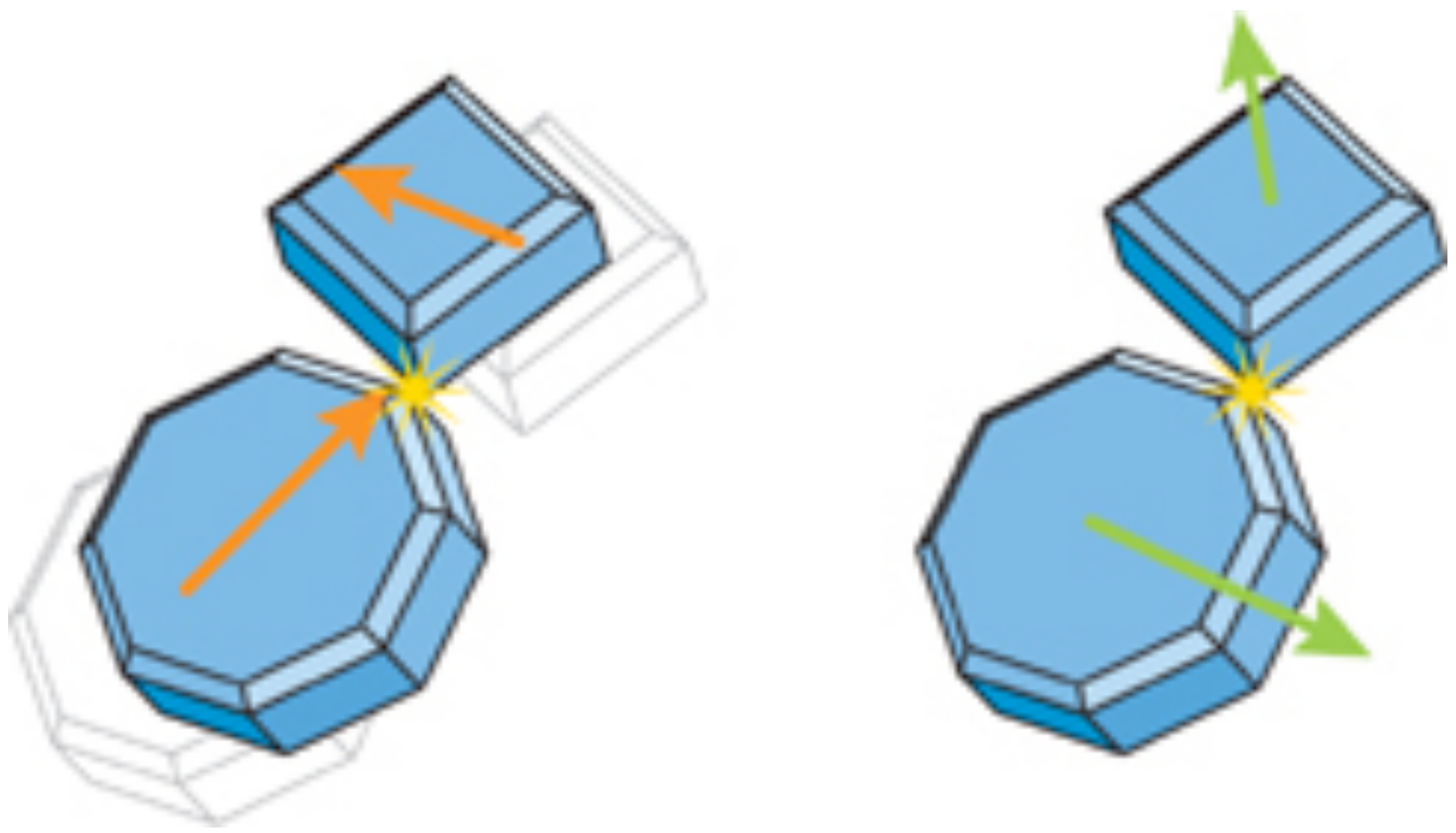
Compare with constrained semi-implicit Euler:

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

# Collisions

**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.

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

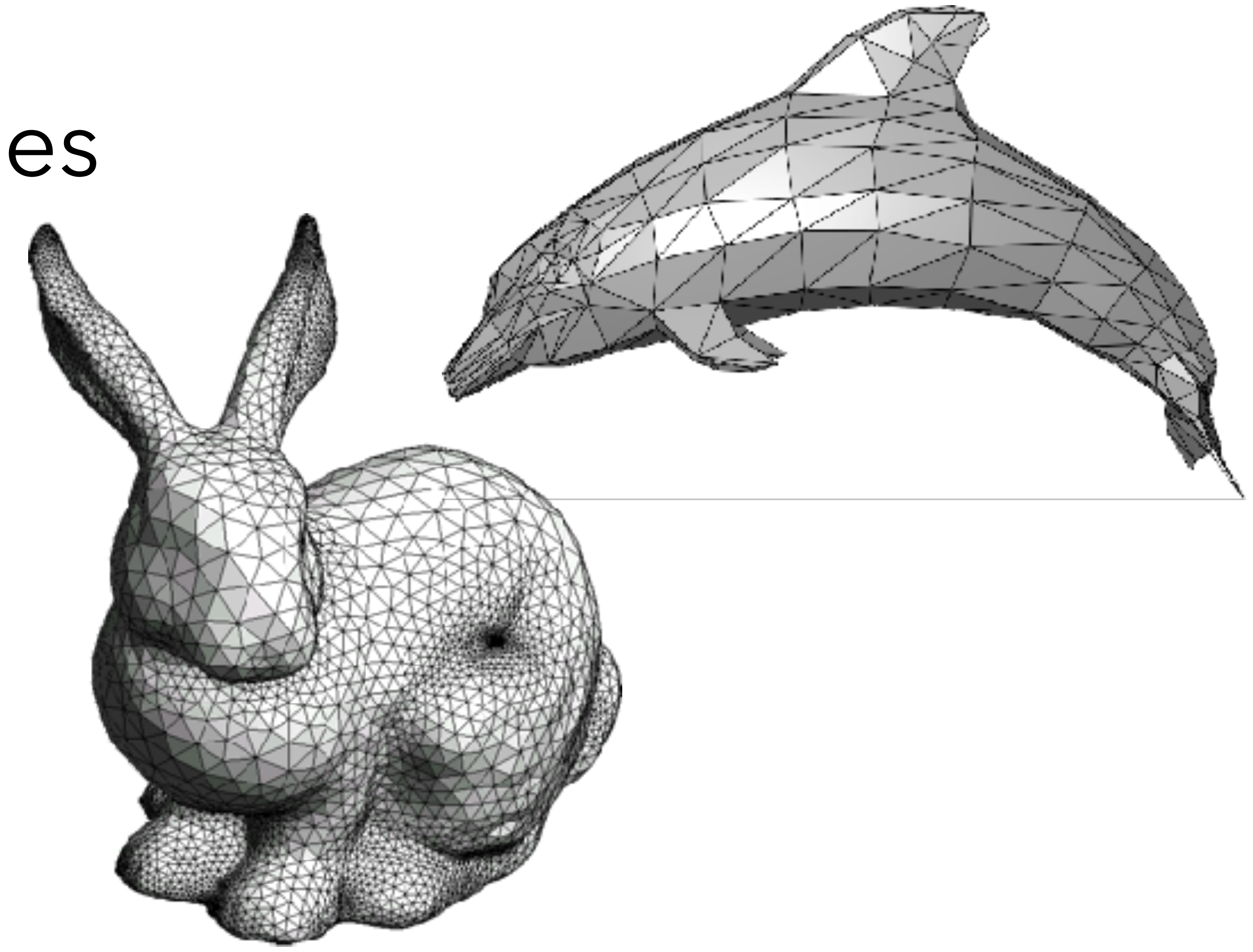
I also have a particle with radius $r$ moving to positions $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots$ at times $t_0, t_1, t_2, \ldots$

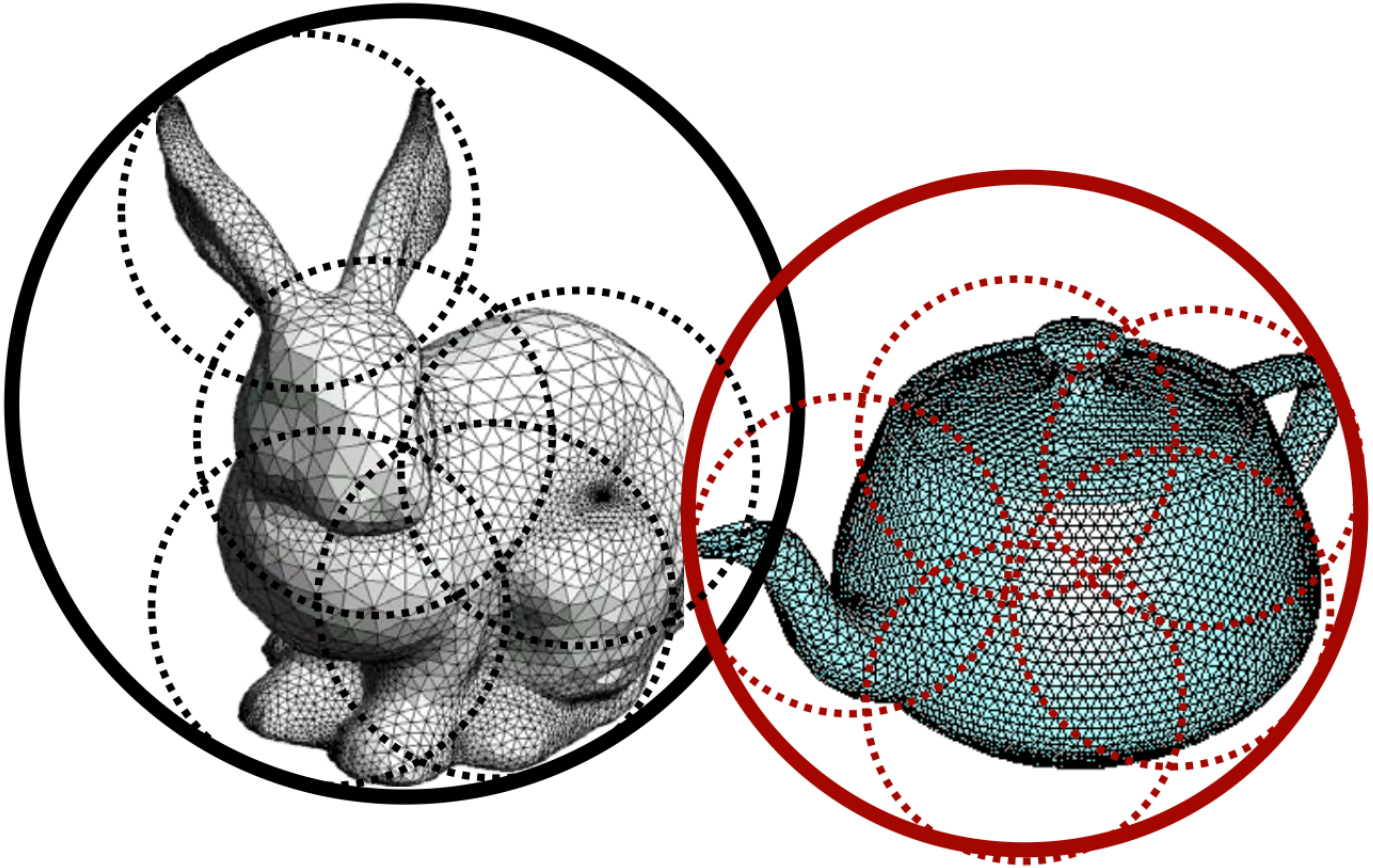
1. How can I do discrete collision detection between the particle and the cylinder?
2. How can I do continuous collision detection for the same?
3. If I model a sheet of cloth as a mass-spring system, is it enough to check that none of the particles are colliding with the cylinder?

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

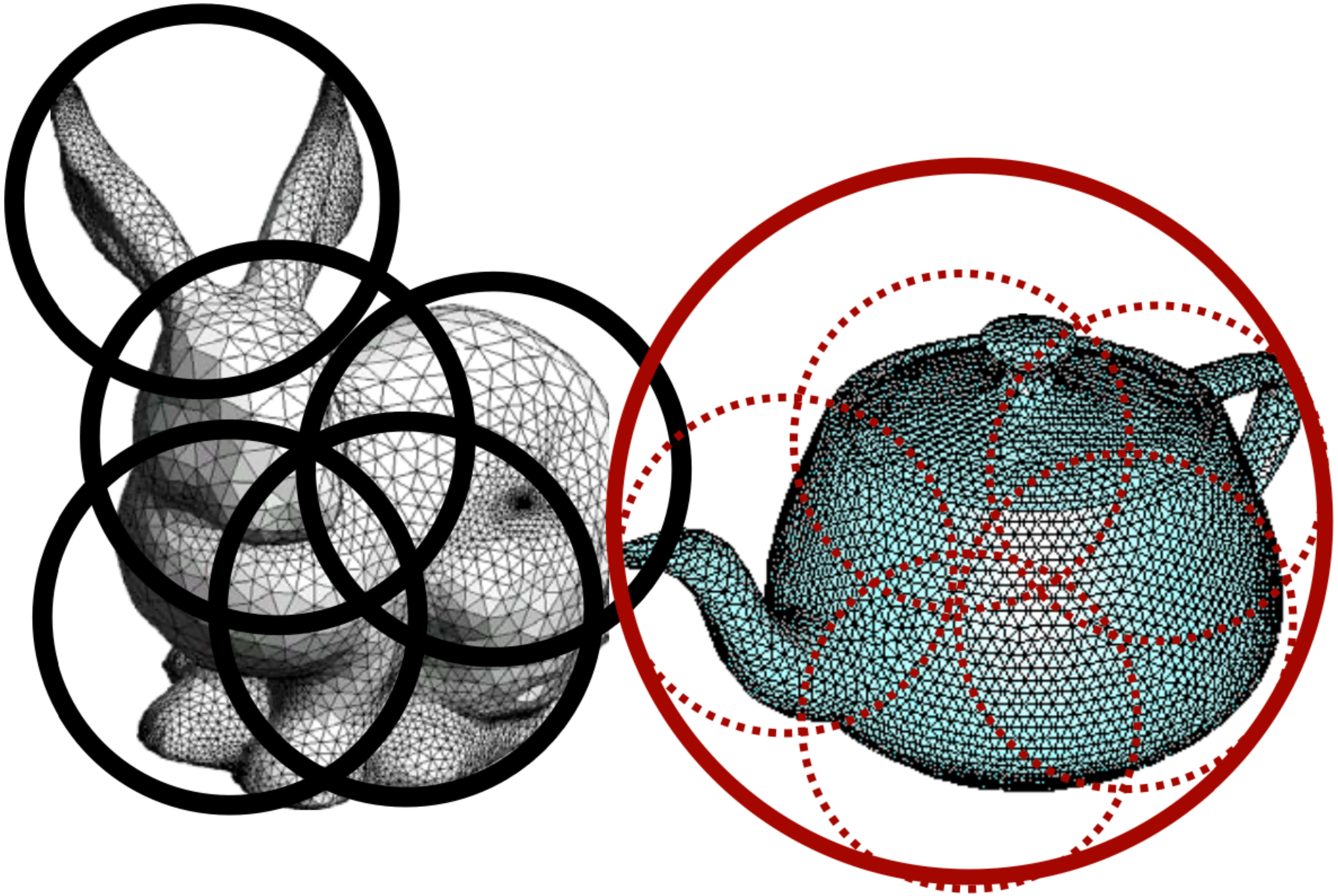
1. **Broad phase**: traverse BVHs of both shapes

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

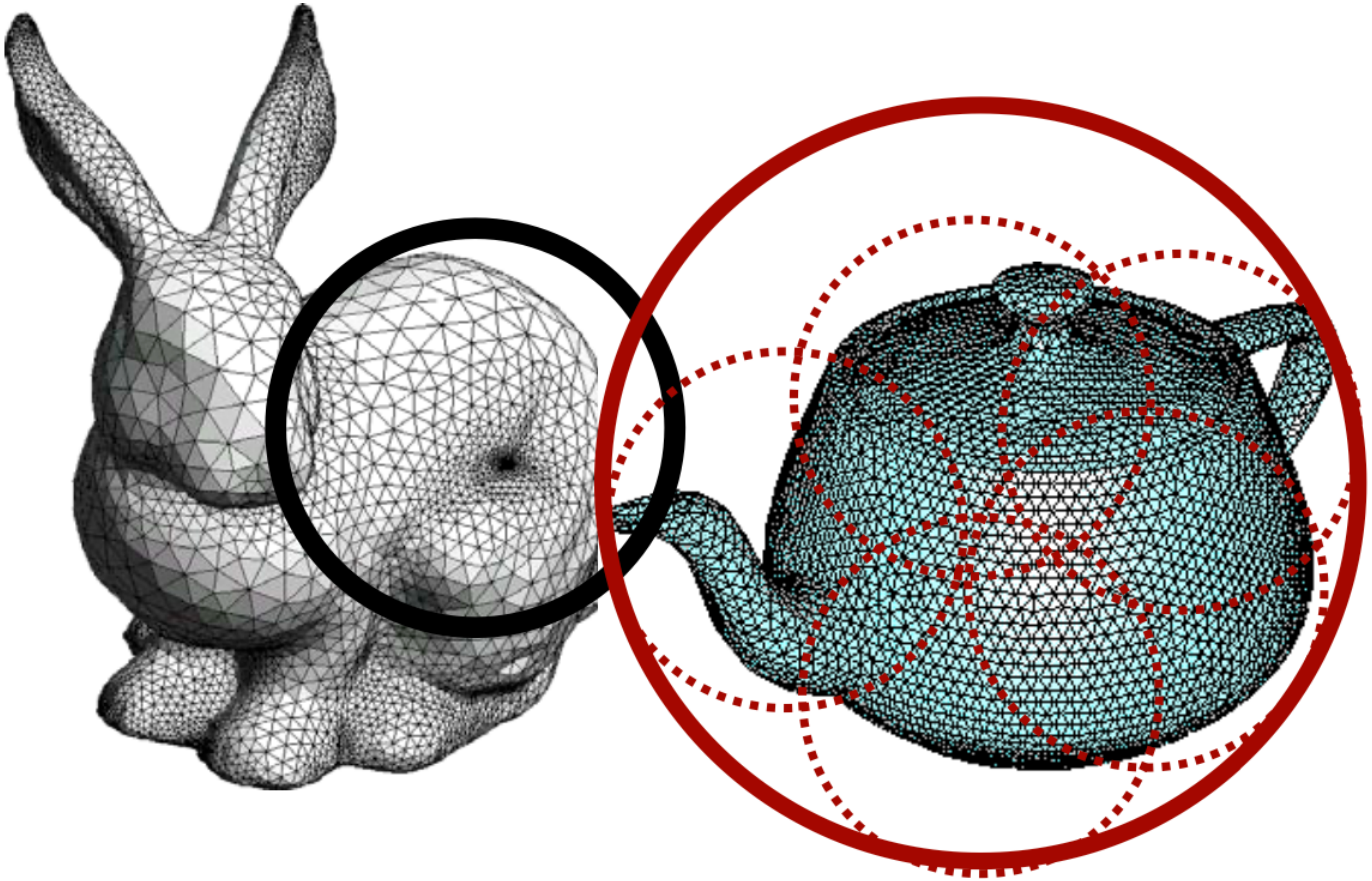




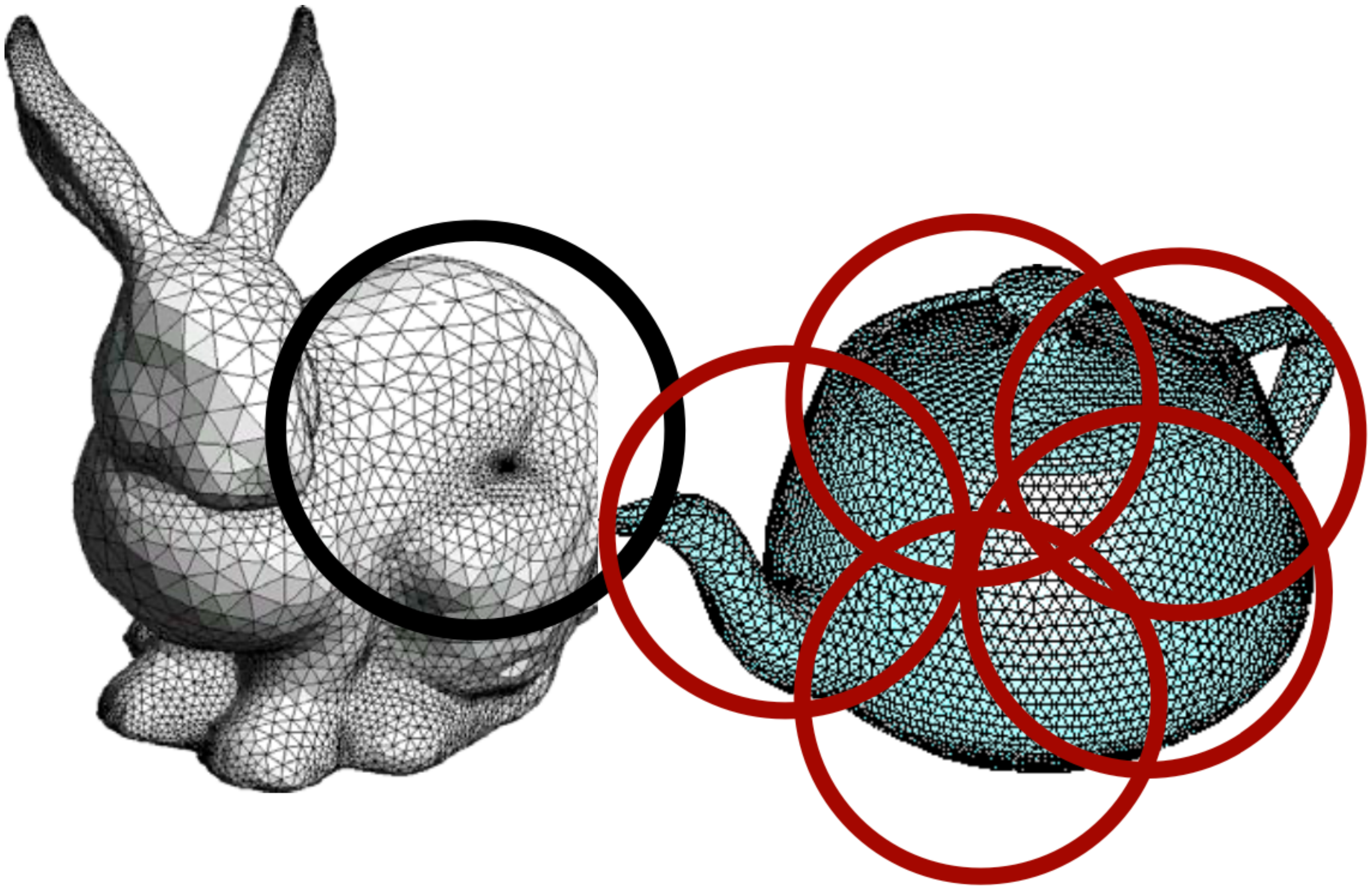
Wojciech Matusik



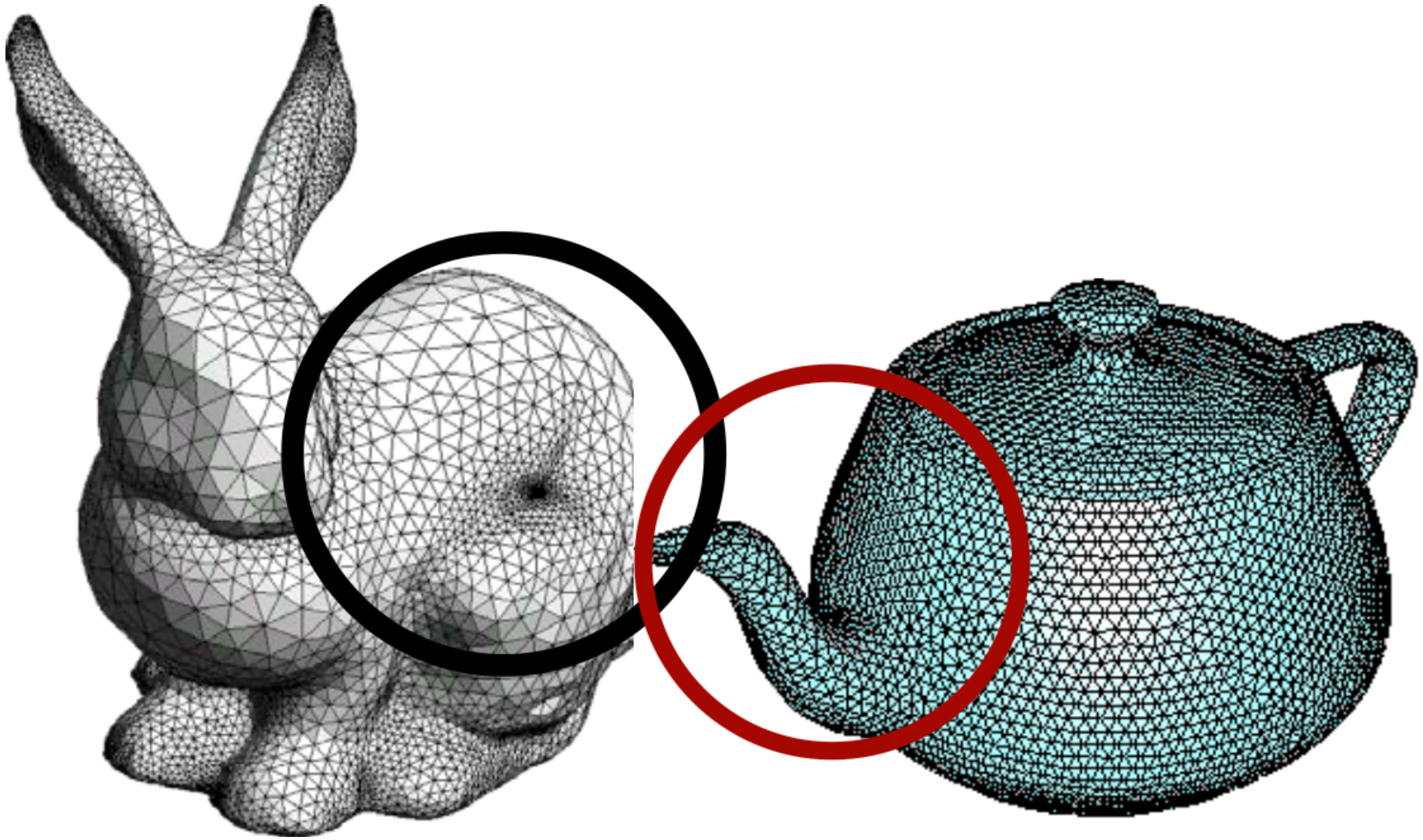
Wojciech Matusik



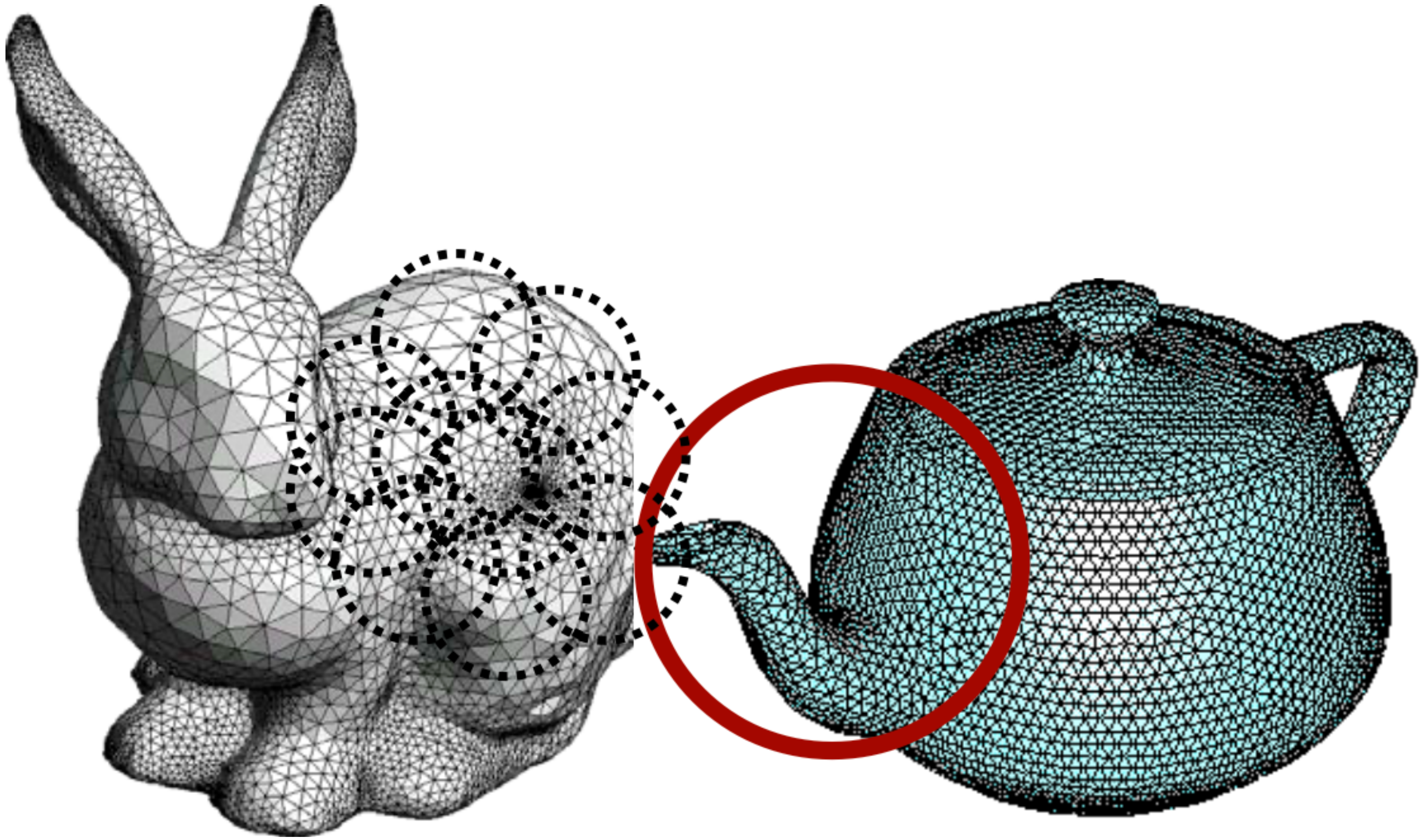
Wojciech Matusik



Wojciech Matusik



Wojciech Matusik



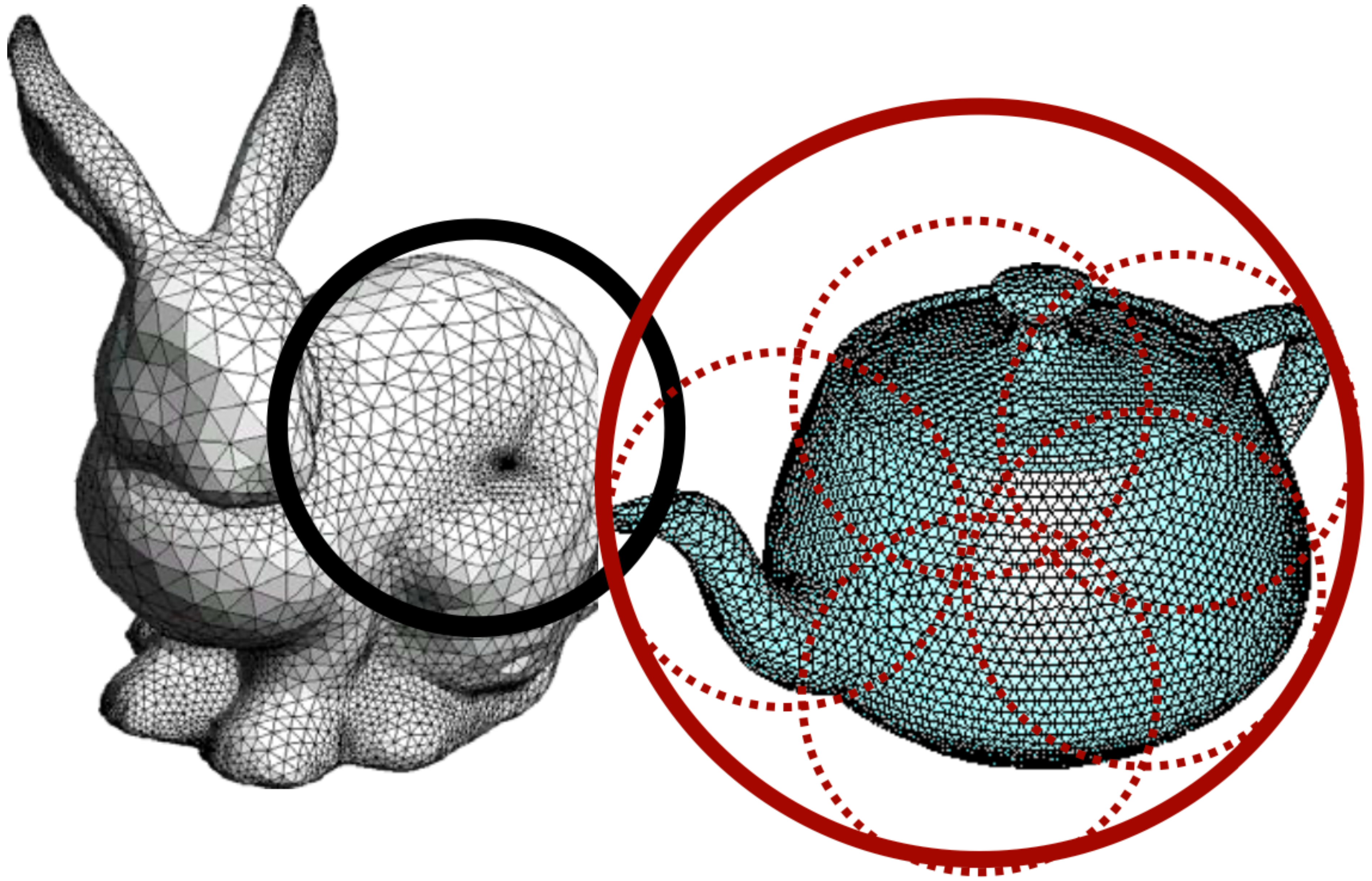
Wojciech Matusik

FindIntersections($node_1$, $node_2$):

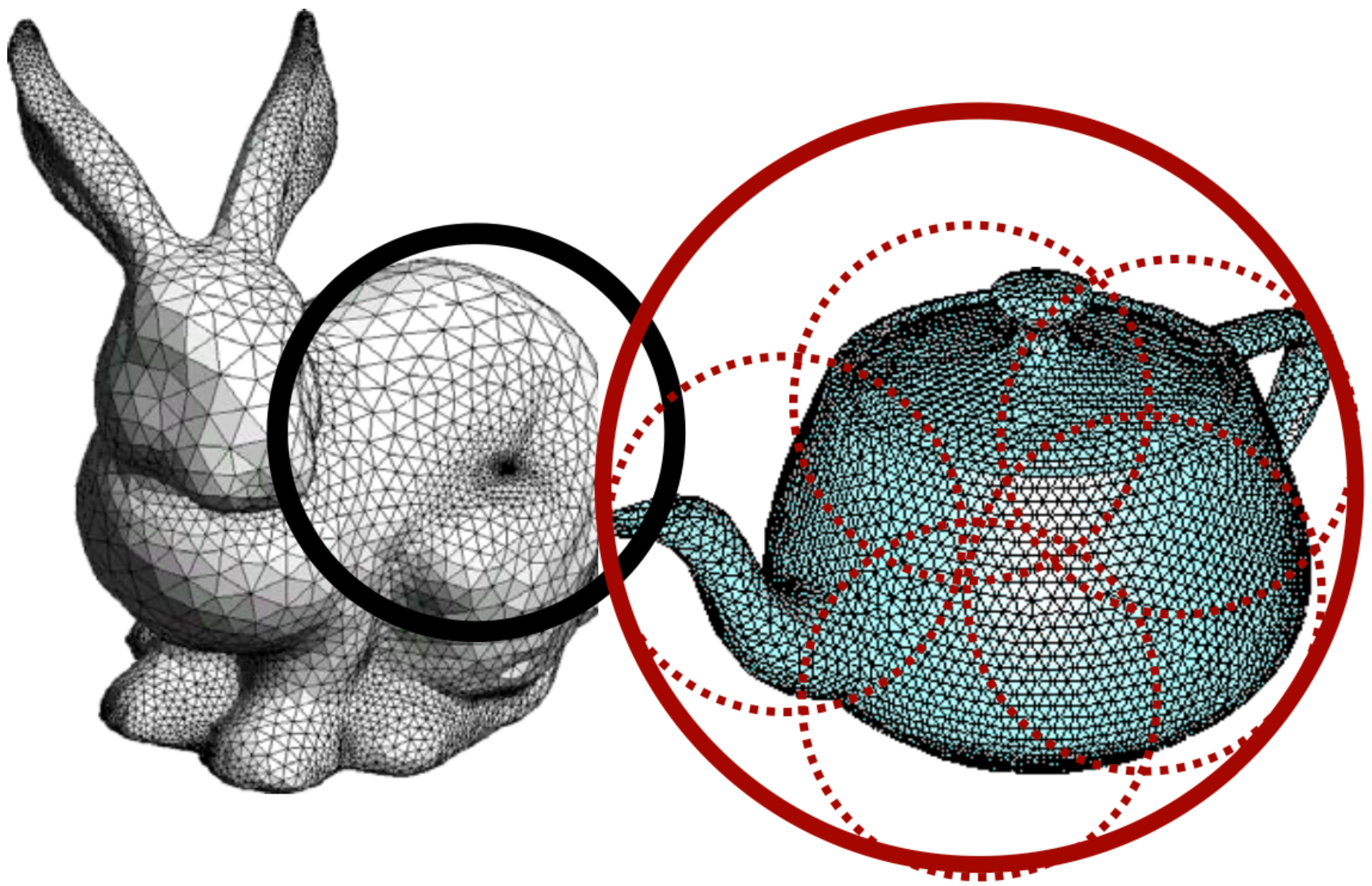
if BVs of $node_1$ and $node_2$ overlap:

for each child of bigger node:

FindIntersections(child, smaller node)

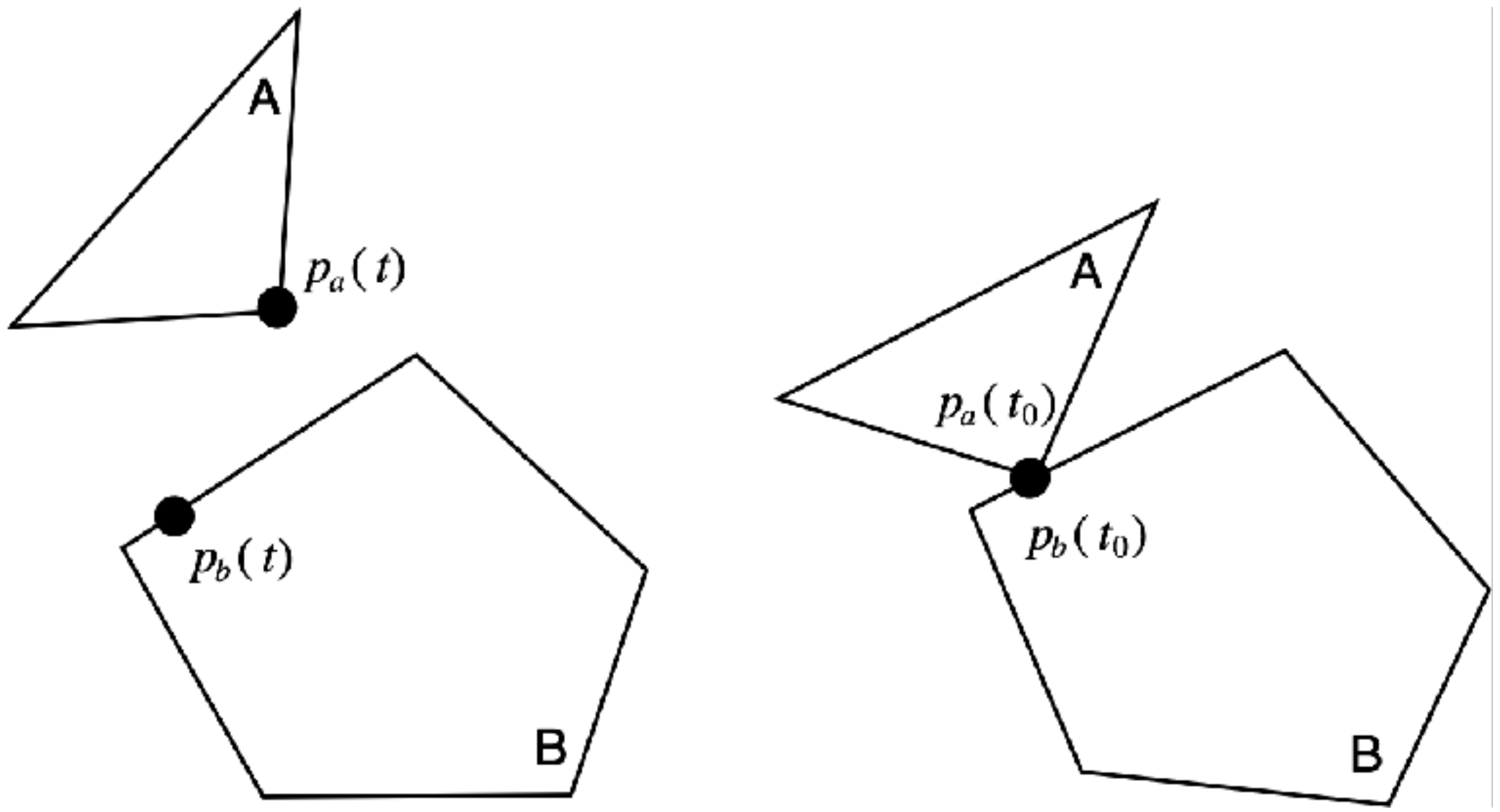


```
FindIntersections(node₁, node₂):
  if BVs of node₁ and node₂ overlap:
    if neither node₁ nor node₂ are leaves:
      for each child of bigger node:
        FindIntersections(child, smaller node)
    else if only one node is a leaf:
      for each child of other node:
        FindIntersections(child, 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 $t^*$



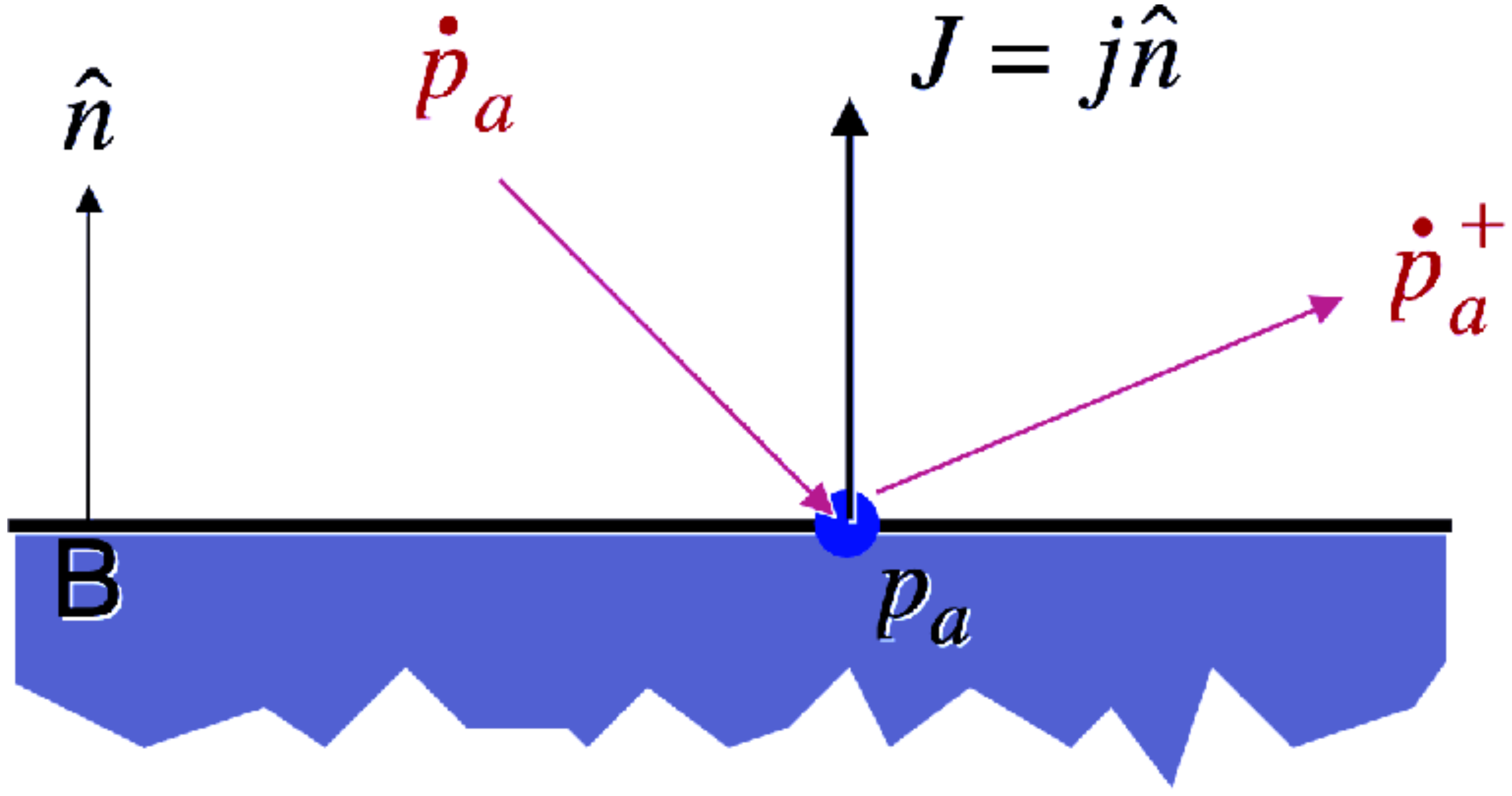

Witkin & Baraff 2001

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

# Collision resolution

Two components:

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



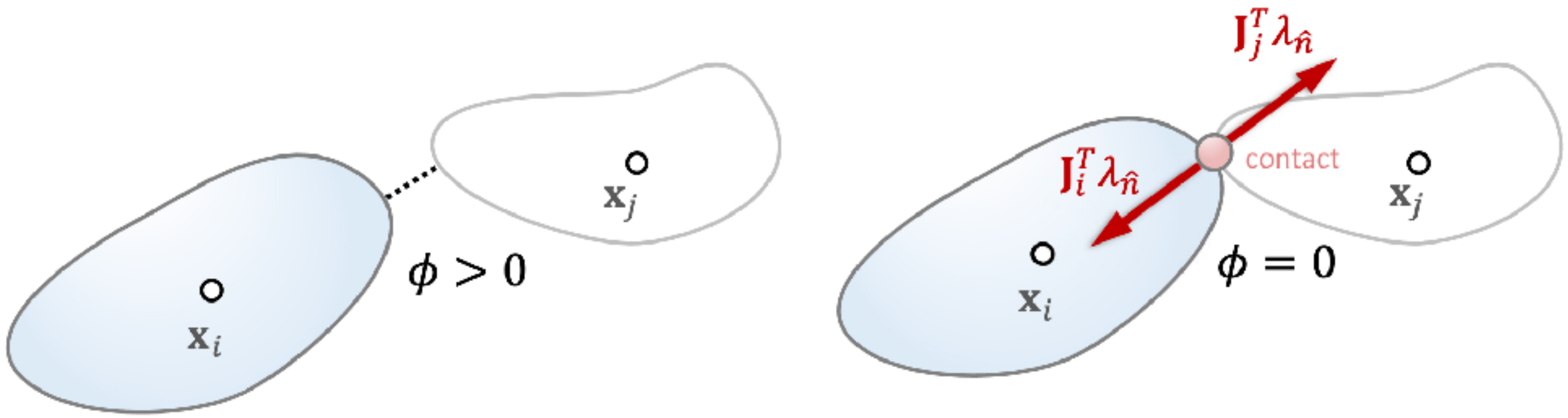

Witkin & Baraff 2001

Actually, collision forces change velocity over an extremely very short time
→ treat as an instantaneous impulse **j** (change in momentum)

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

The normal component is like a constraint force, except it's "one-sided"…

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




Andrews et al. 2002

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

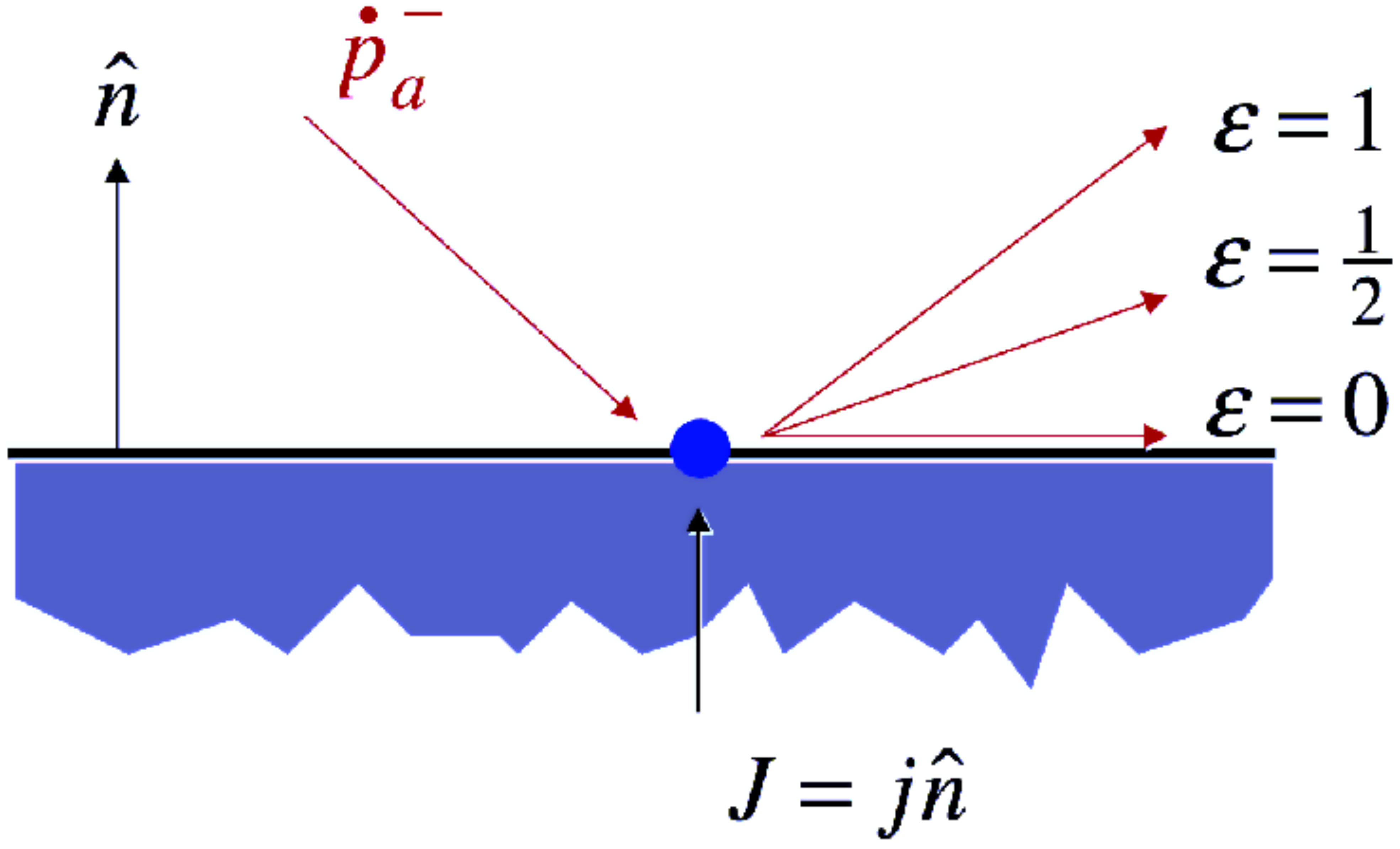
Normal impulse: $\mathbf{j} = \lambda\, \nabla\varphi(\mathbf{q})$, $\lambda \geq 0$ (no sticking)

**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

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



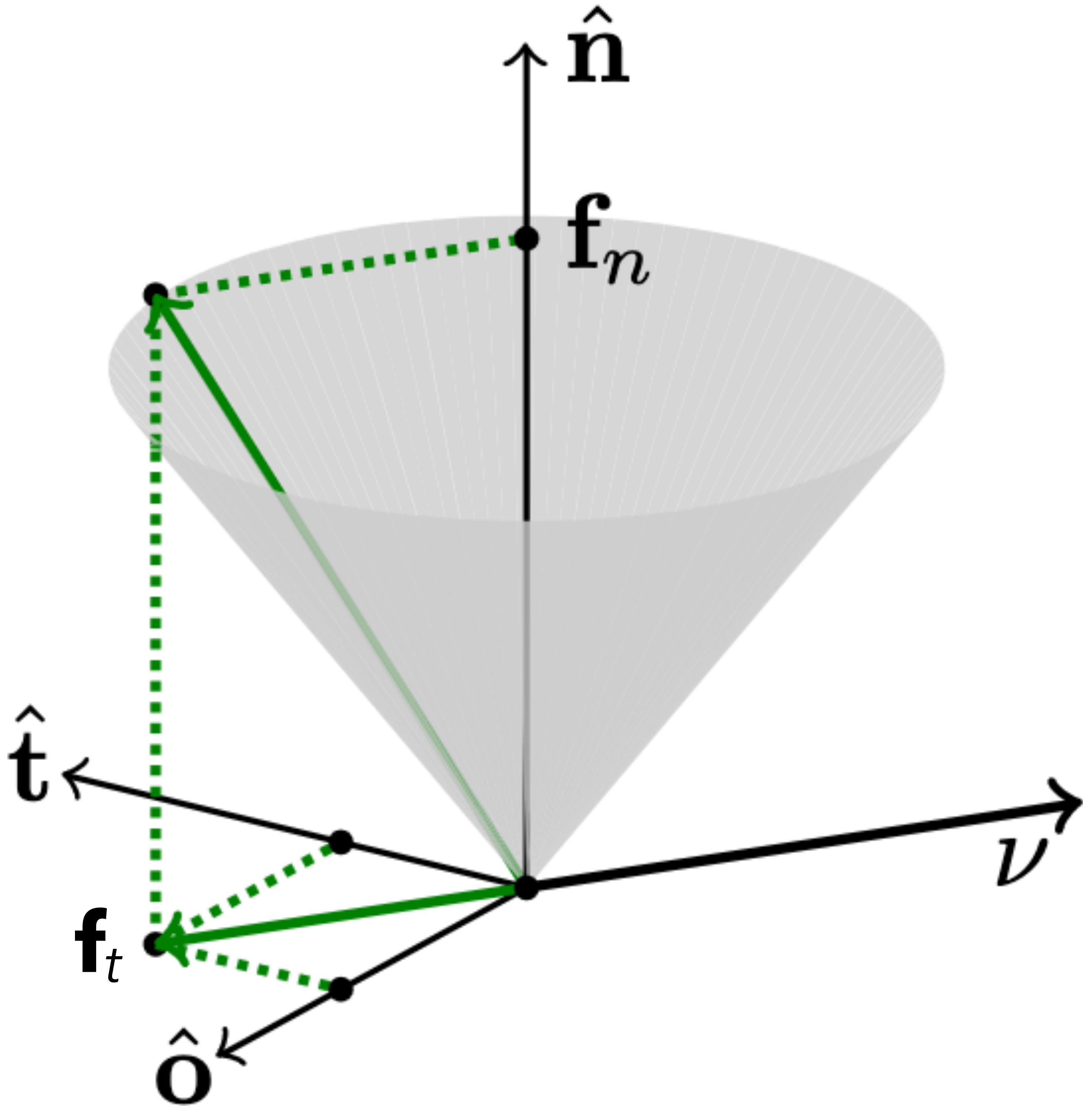

Witkin & Baraff 2001

Friction is described by Coulomb's law

$$\|\mathbf{f}_t\| \leq \mu f_n$$

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

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



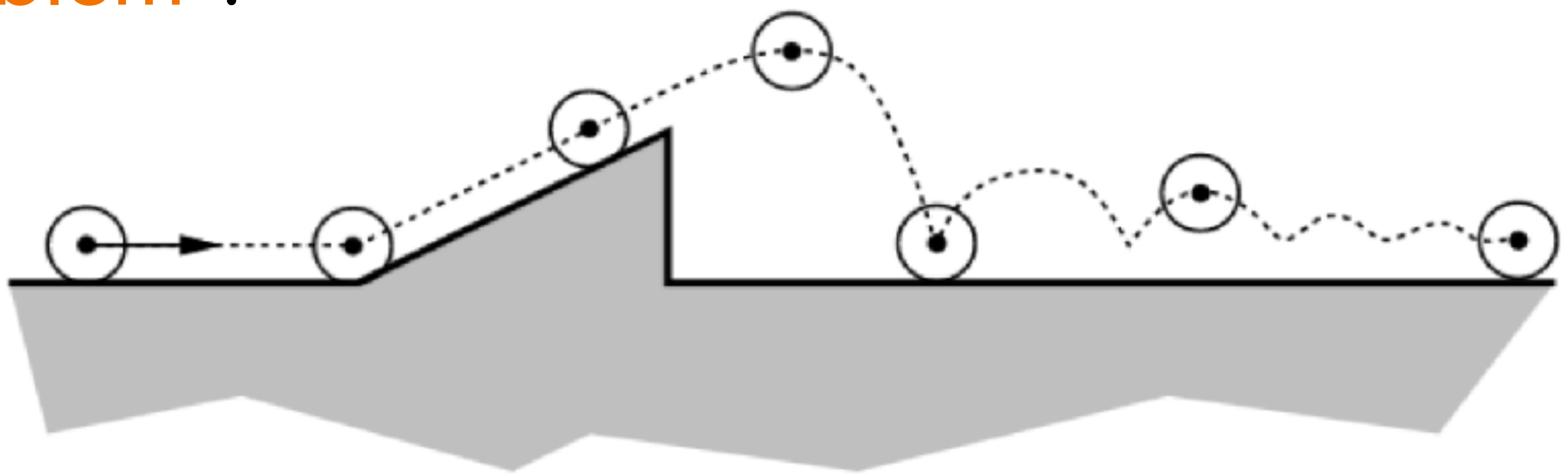

Bender et al. 2012

# Time stepping issues

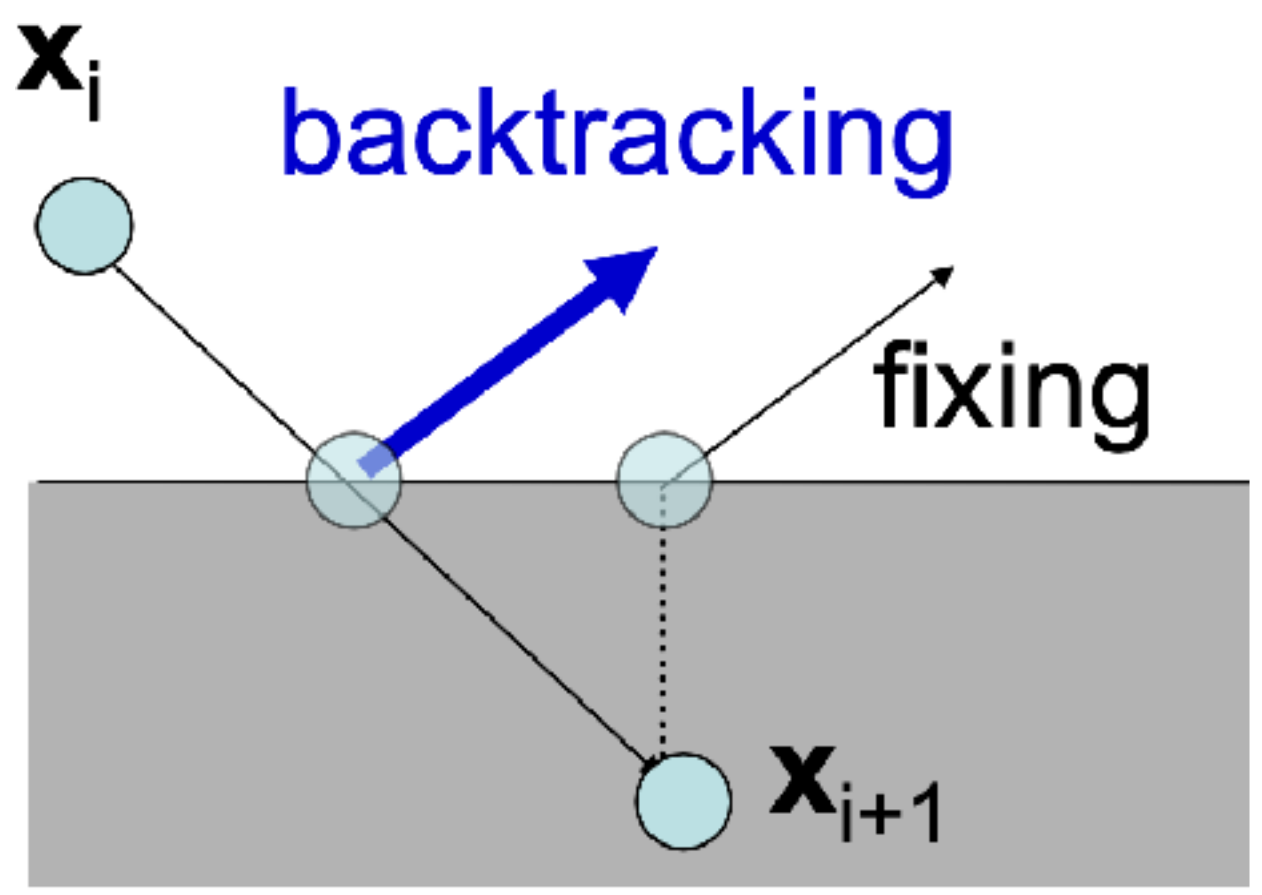
We usually only detect collisions after they've already happened!

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

  "Zeno problem":



Mirtich & Canny 1994




Wojciech Matusik

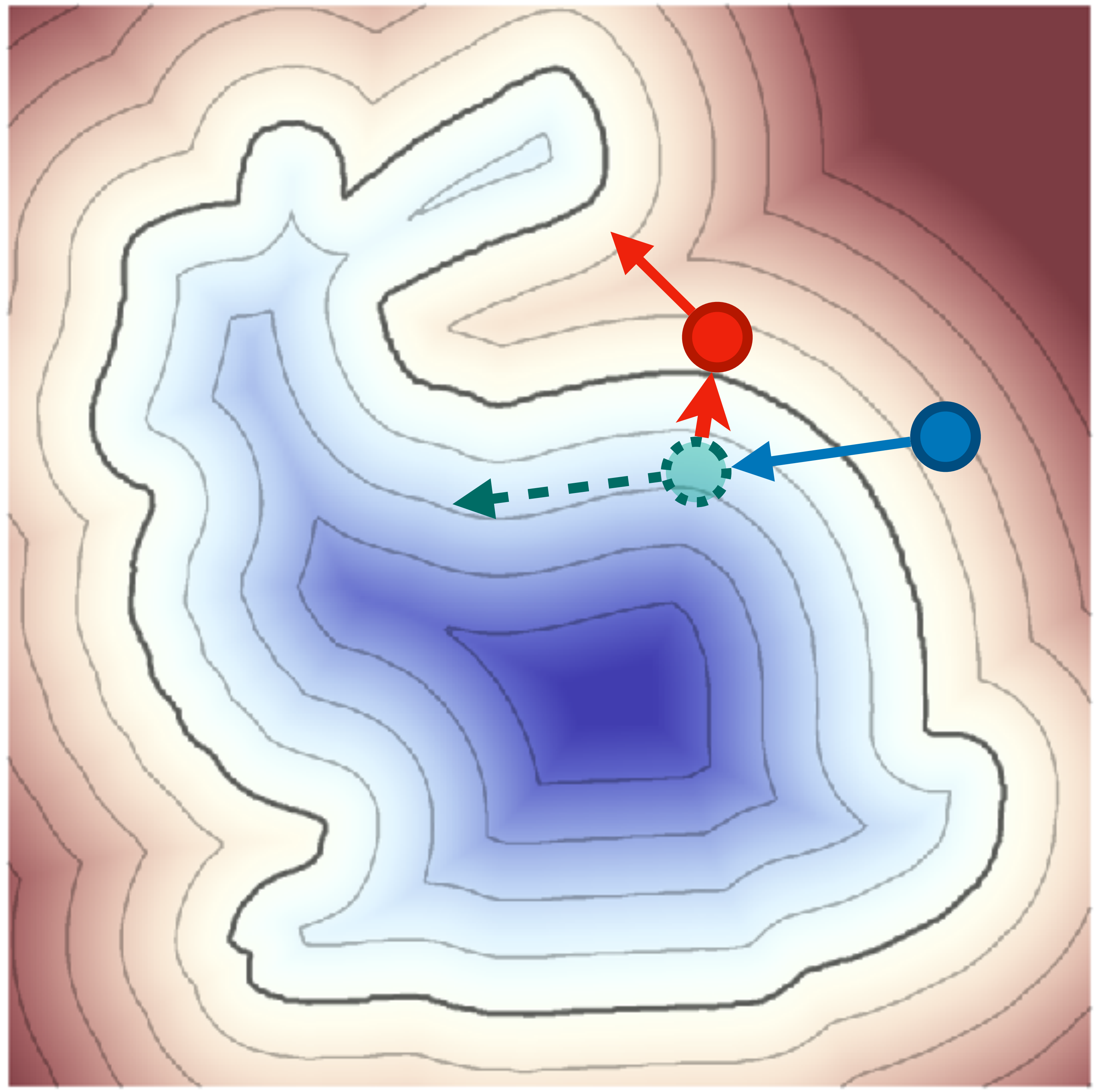
- Option 2: Just lie about it! Project end-of-step positions to remove interpenetration

A simple strategy for particle/implicit collisions:

Perform **v**, **x** update as usual

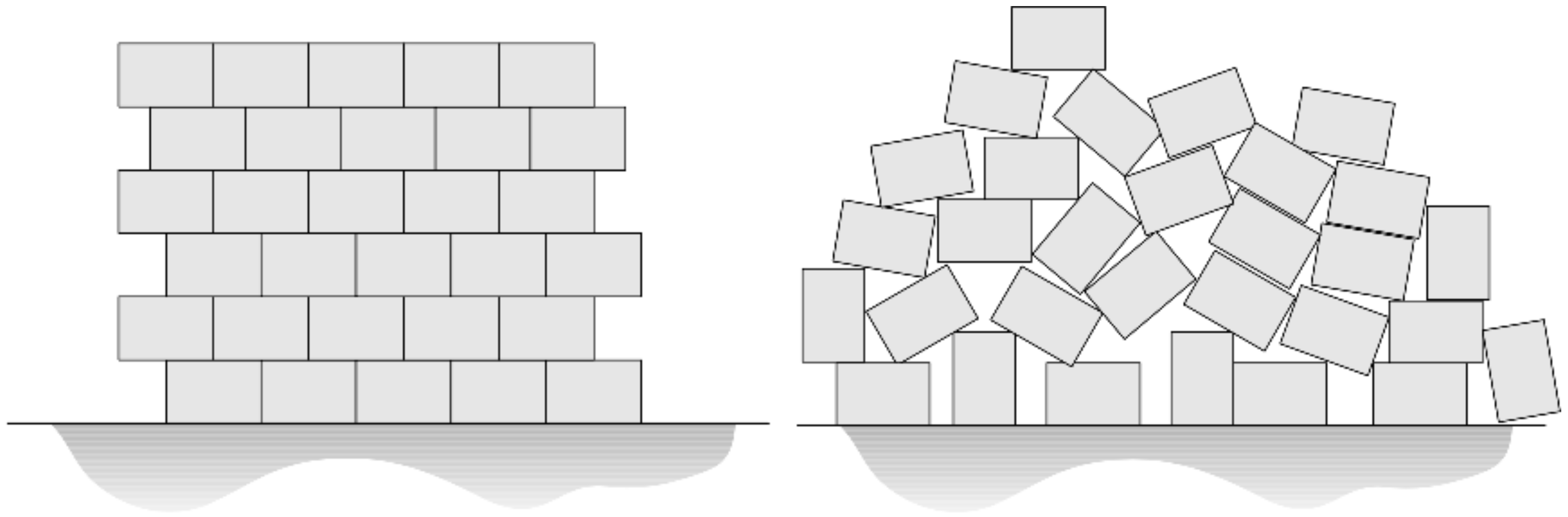
If inside obstacle ($\varphi(\mathbf{x}) < 0$):

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

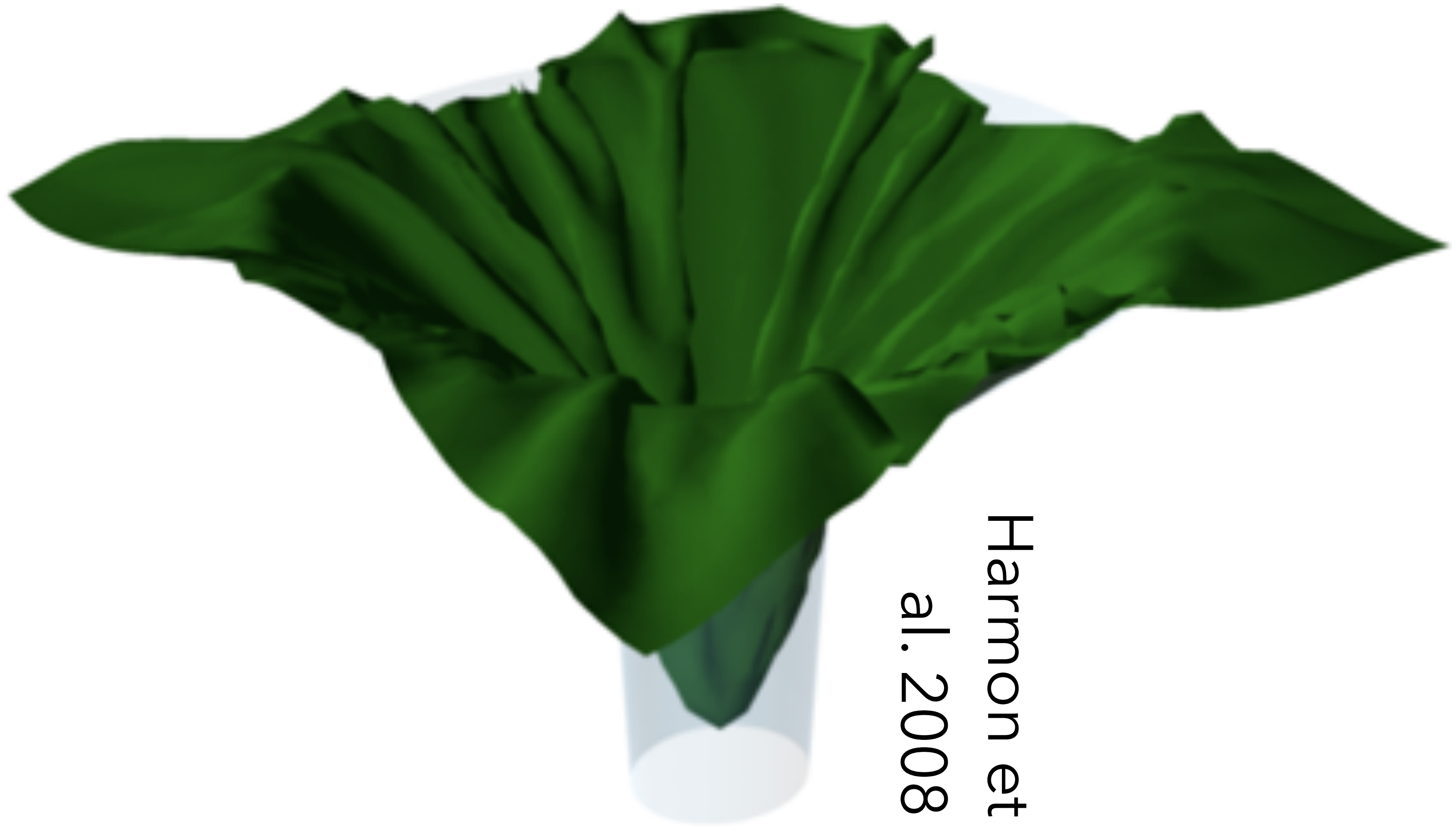


Can also add a tangential position correction
(using $(\mathbf{x}^{n+1} - \mathbf{x}^{n})_t$ instead of $\mathbf{v}_t$) to counteract artificial sliding…

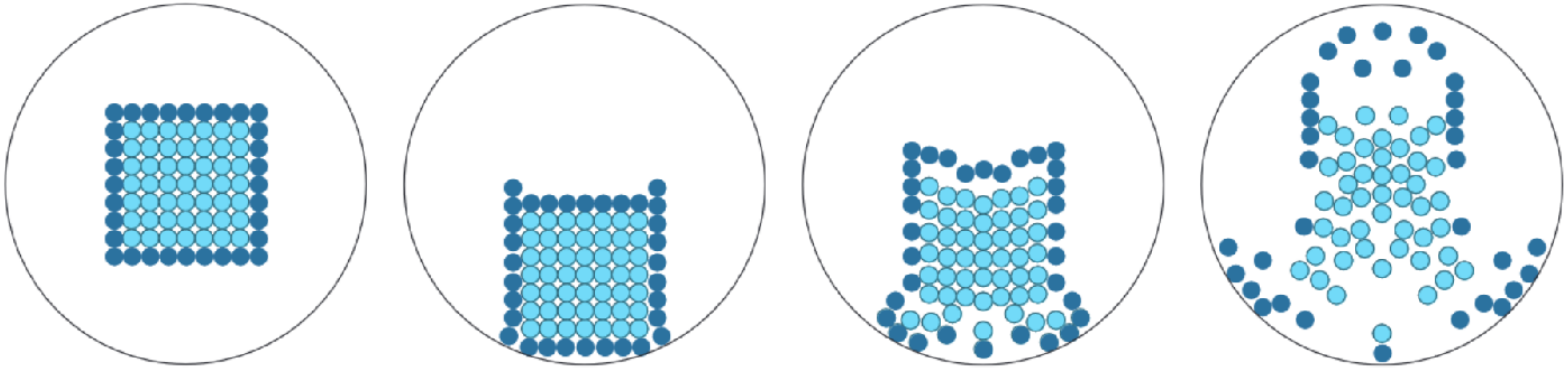
# Multi-contact problems (harder!)



Erleben 2007



Harmon et al. 2008



Smith et al. 2012