GAMES103: Intro to Physics-Based Animation

Rigid Contacts

Huamin Wang

Nov 2021

Last week…

In practice, we update the same state variable $\mathbf{s} = \{\mathbf{v}, \mathbf{x}, \boldsymbol{\omega}, \mathbf{q}\}$ over time.

$\mathbf{v}$

$\mathbf{x}$

$$\boxed{\mathbf{f}_i \leftarrow \mathrm{Force}(\mathbf{x}_i, \mathbf{v}_i)}$$

$$\mathbf{f} \leftarrow \sum \mathbf{f}_i$$

$$\mathbf{v} \leftarrow \mathbf{v} + \Delta t M^{-1} \mathbf{f}$$

$$\mathbf{x} \leftarrow \mathbf{x} + \Delta t \mathbf{v}$$

$\mathbf{v}$

$\mathbf{x}$

$\boldsymbol{\omega}$

$\mathbf{q}$

$$\boxed{\begin{aligned} &\mathbf{R} \leftarrow \mathrm{Matrix.Rotate}(\mathbf{q}) \\ &\boldsymbol{\tau}_i \leftarrow (\mathbf{R}\mathbf{r}_i) \times \mathbf{f}_i \end{aligned}}$$

$$\boldsymbol{\tau} \leftarrow \sum \boldsymbol{\tau}_i$$

$$\mathbf{I} \leftarrow \mathbf{R}\mathbf{I}_{\mathbf{ref}}\mathbf{R}^{\mathrm{T}}$$

$$\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} + \Delta t (\mathbf{I})^{-1} \boldsymbol{\tau}$$

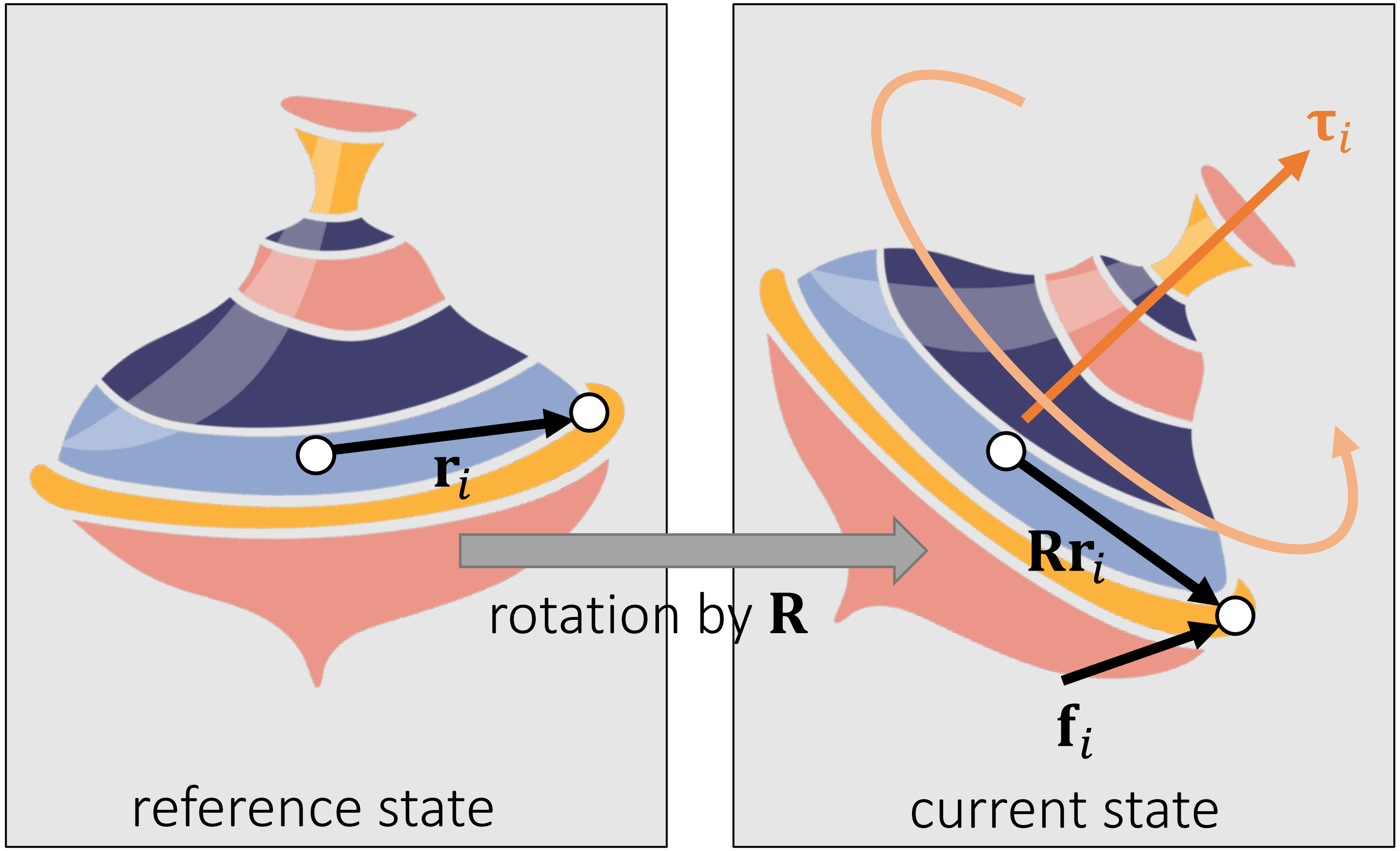
$$\mathbf{q} \leftarrow \mathbf{q} + \left[0 \quad \frac{\Delta t}{2}\boldsymbol{\omega}\right] \times \mathbf{q}$$

$\boldsymbol{\omega}$

$\mathbf{q}$

What is a torque?

A torque is the rotational equivalent of a force. It describes the rotational tendency caused by a force.



$\boldsymbol{\tau}_i$ is perpendicular to both vectors: $\mathbf{Rr}_i$ and $\mathbf{f}_i$.

$\boldsymbol{\tau}_i$ is porportional to $\|\mathbf{Rr}_i\|$ and $\|\mathbf{f}_i\|$.

$\boldsymbol{\tau}_i$ is porportional to $\sin\theta$.
(θ is the angle between two vectors.)

$$\boxed{\boldsymbol{\tau}_i \longleftarrow (\mathbf{Rr}_i) \times \mathbf{f}_i}$$

Last week…

In practice, we update the same state variable $\mathbf{s} = \{\mathbf{v}, \mathbf{x}, \boldsymbol{\omega}, \mathbf{q}\}$ over time.

$\mathbf{v}$ $\mathbf{x}$

$$\mathbf{f}_i \leftarrow \mathrm{Force}(\mathbf{x}_i, \mathbf{v}_i)$$

$$\boxed{\mathbf{f} \leftarrow \sum \mathbf{f}_i}$$

$$\mathbf{v} \leftarrow \mathbf{v} + \Delta t M^{-1}\mathbf{f}$$

$$\mathbf{x} \leftarrow \mathbf{x} + \Delta t \mathbf{v}$$

$\mathbf{v}$ $\mathbf{x}$

$\boldsymbol{\omega}$ $\mathbf{q}$

$$\mathbf{R} \leftarrow \mathrm{Matrix.Rotate}(\mathbf{q})$$

$$\boldsymbol{\tau}_i \leftarrow (\mathbf{R}\mathbf{r}_i) \times \mathbf{f}_i$$

$$\boxed{\boldsymbol{\tau} \leftarrow \sum \boldsymbol{\tau}_i}$$

$$\mathbf{I} \leftarrow \mathbf{R}\mathbf{I}_{\mathbf{ref}}\mathbf{R}^{\mathrm{T}}$$

$$\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} + \Delta t(\mathbf{I})^{-1}\boldsymbol{\tau}$$

$$\mathbf{q} \leftarrow \mathbf{q} + \left[0 \quad \frac{\Delta t}{2}\boldsymbol{\omega}\right] \times \mathbf{q}$$

$\boldsymbol{\omega}$ $\mathbf{q}$

Last week…

In practice, we update the same state variable $\mathbf{s} = \{\mathbf{v}, \mathbf{x}, \boldsymbol{\omega}, \mathbf{q}\}$ over time.

$\mathbf{v}$

$\mathbf{x}$

$$\mathbf{f}_i \leftarrow \mathrm{Force}(\mathbf{x}_i, \mathbf{v}_i)$$

$$\mathbf{f} \leftarrow \sum \mathbf{f}_i$$

$$\boxed{\mathbf{v} \leftarrow \mathbf{v} + \Delta t M^{-1} \mathbf{f}}$$

$$\mathbf{x} \leftarrow \mathbf{x} + \Delta t \mathbf{v}$$

$\mathbf{v}$

$\mathbf{x}$

$\boldsymbol{\omega}$

$\mathbf{q}$

$$\mathbf{R} \leftarrow \mathrm{Matrix.Rotate}(\mathbf{q})$$

$$\boldsymbol{\tau}_i \leftarrow (\mathbf{R}\mathbf{r}_i) \times \mathbf{f}_i$$

$$\boldsymbol{\tau} \leftarrow \sum \boldsymbol{\tau}_i$$

$$\boxed{\begin{aligned} &\mathbf{I} \leftarrow \mathbf{R}\mathbf{I}_{\mathbf{ref}}\mathbf{R}^{\mathrm{T}} \\ &\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} + \Delta t (\mathbf{I})^{-1} \boldsymbol{\tau} \end{aligned}}$$

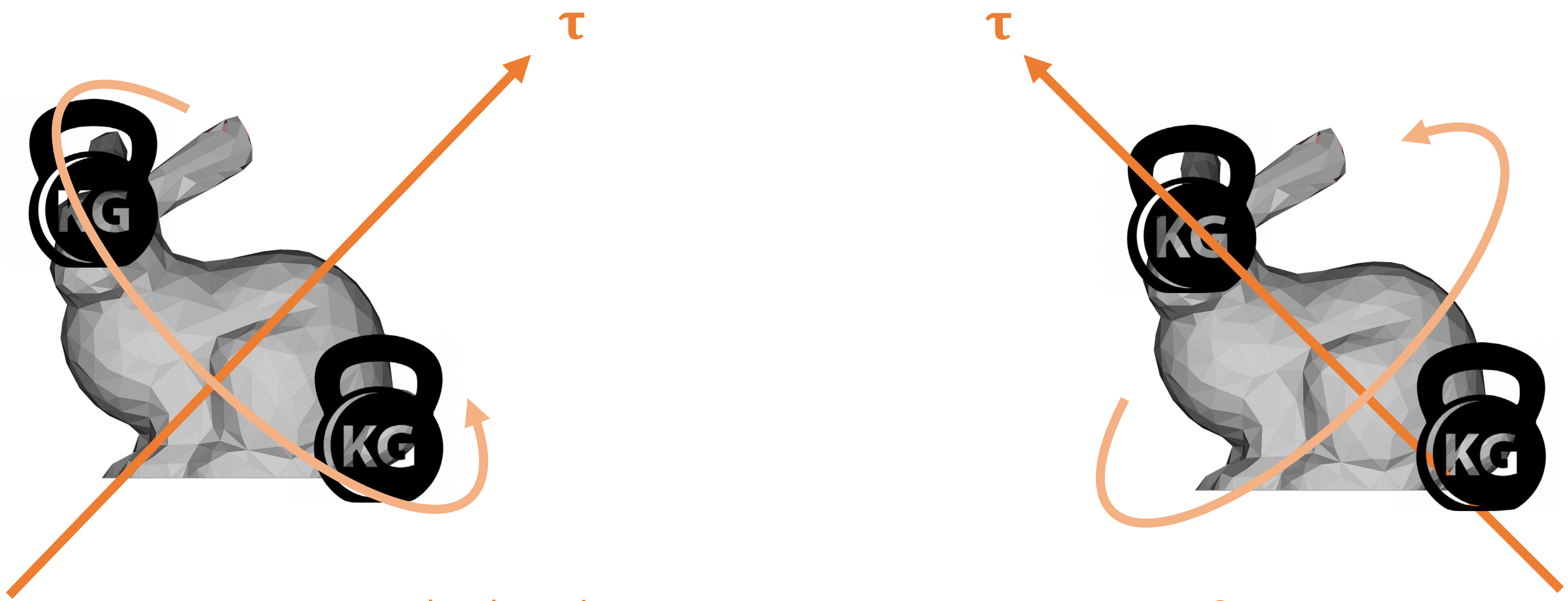
$$\mathbf{q} \leftarrow \mathbf{q} + \left[0 \quad \frac{\Delta t}{2}\boldsymbol{\omega}\right] \times \mathbf{q}$$

$\boldsymbol{\omega}$

$\mathbf{q}$

What is an inertia tensor?

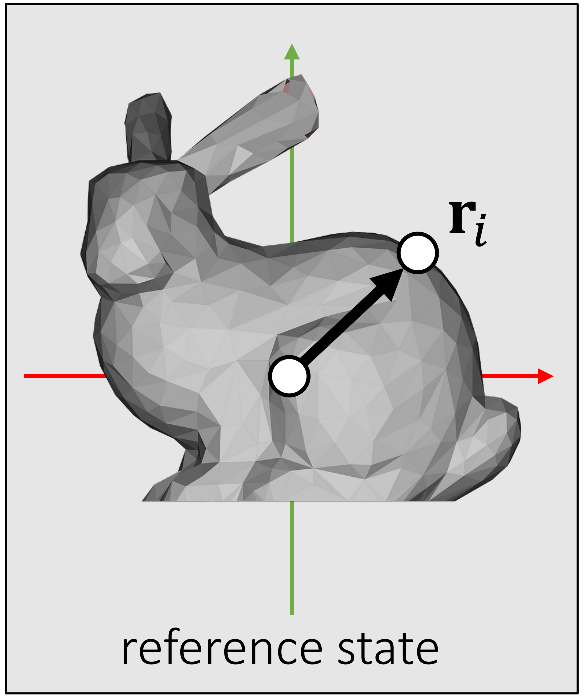
Similar to mass, an inertia tensor describes the resistance to rotational tendency caused by torque. But different from mass, it's not a constant.

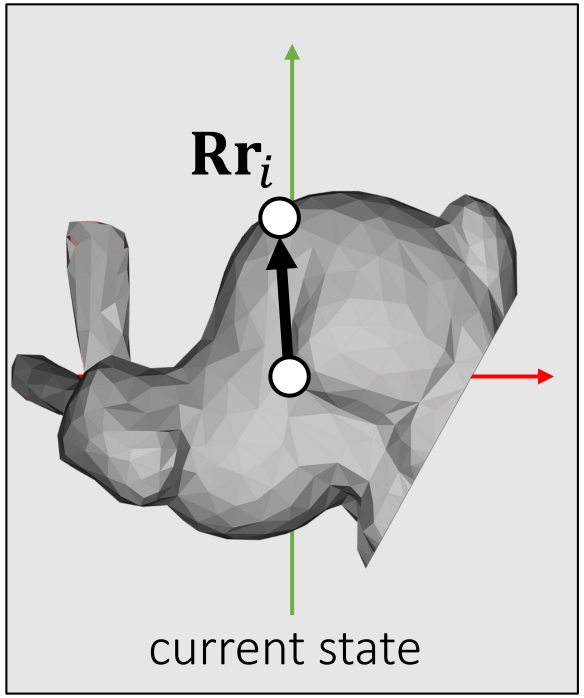


Which side receives greater resistance?

What is an inertia tensor?

It's a matrix! The mass inverse is the resistance (just like mass).





$$\mathbf{I}_{\mathbf{ref}} = \sum m_i(\mathbf{r}_i^{\mathrm{T}}\mathbf{r}_i\mathbf{1} - \mathbf{r}_i\mathbf{r}_i^{\mathrm{T}})$$

$\mathbf{1}$ is the 3-by-3 identity.

What about the current inertia?

$$\mathbf{I} = \sum m_i(\mathbf{r}_i^{\mathrm{T}}\mathbf{R}^{\mathrm{T}}\mathbf{R}\mathbf{r}_i\mathbf{1} - \mathbf{R}\mathbf{r}_i\mathbf{r}_i^{\mathrm{T}}\mathbf{R}^{\mathrm{T}})$$

$$= \sum m_i(\mathbf{R}\mathbf{r}_i^{\mathrm{T}}\mathbf{r}_i\mathbf{1}\mathbf{R}^{\mathrm{T}} - \mathbf{R}\mathbf{r}_i\mathbf{r}_i^{\mathrm{T}}\mathbf{R}^{\mathrm{T}})$$

$$= \sum m_i\mathbf{R}(\mathbf{r}_i^{\mathrm{T}}\mathbf{r}_i\mathbf{1} - \mathbf{r}_i\mathbf{r}_i^{\mathrm{T}})\mathbf{R}^{\mathrm{T}}$$

$$= \mathbf{R}\mathbf{I}_{\mathbf{ref}}\mathbf{R}^{\mathrm{T}}$$

Last week…

In practice, we update the same state variable $\mathbf{s} = \{\mathbf{v}, \mathbf{x}, \boldsymbol{\omega}, \mathbf{q}\}$ over time.

$\mathbf{v}$

$\mathbf{x}$

$$\mathbf{f}_i \leftarrow \text{Force}(\mathbf{x}_i, \mathbf{v}_i)$$

$$\mathbf{f} \leftarrow \sum \mathbf{f}_i$$

$$\mathbf{v} \leftarrow \mathbf{v} + \Delta t M^{-1}\mathbf{f}$$

$$\boxed{\mathbf{x} \leftarrow \mathbf{x} + \Delta t \mathbf{v}}$$

$\mathbf{v}$

$\mathbf{x}$

$\boldsymbol{\omega}$

$\mathbf{q}$

$$\mathbf{R} \leftarrow \text{Matrix.Rotate}(\mathbf{q})$$

$$\boldsymbol{\tau}_i \leftarrow (\mathbf{R}\mathbf{r}_i) \times \mathbf{f}_i$$

$$\boldsymbol{\tau} \leftarrow \sum \boldsymbol{\tau}_i$$

$$\mathbf{I} \leftarrow \mathbf{R}\mathbf{I}_{\mathbf{ref}}\mathbf{R}^{\mathrm{T}}$$

$$\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} + \Delta t(\mathbf{I})^{-1}\boldsymbol{\tau}$$

$$\boxed{\mathbf{q} \leftarrow \mathbf{q} + \begin{bmatrix} 0 & \frac{\Delta t}{2}\boldsymbol{\omega} \end{bmatrix} \times \mathbf{q}}$$

After class reading (Appendix B)

$\boldsymbol{\omega}$

$\mathbf{q}$

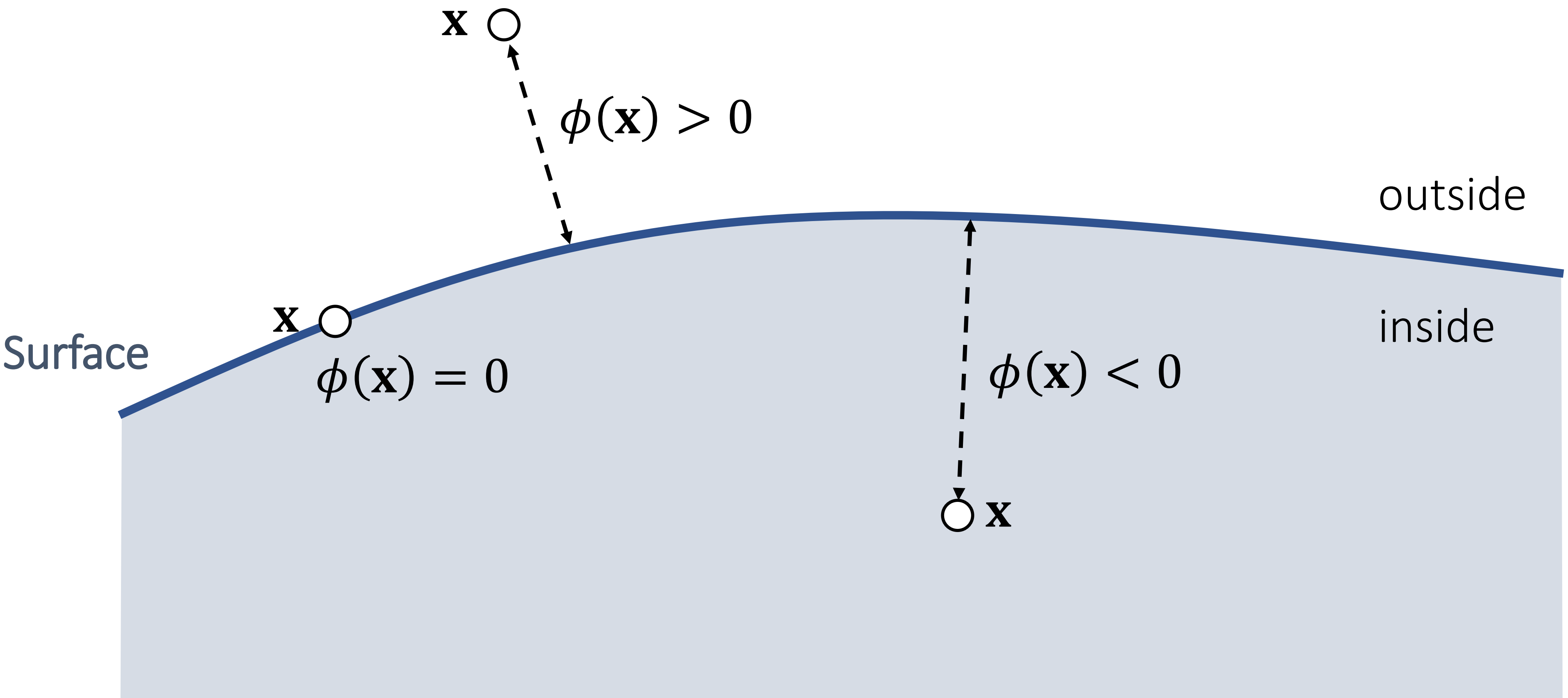
Topics for the Day

- Particle Collision Detection and Response
  - Penalty methods
  - Impulse methods
- Rigid Collision Detection and Response by Impulse
- Shape Matching

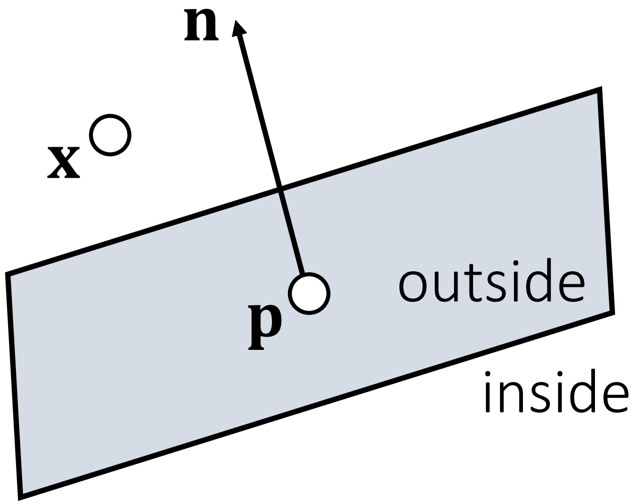
Particle Collision Detection and Response

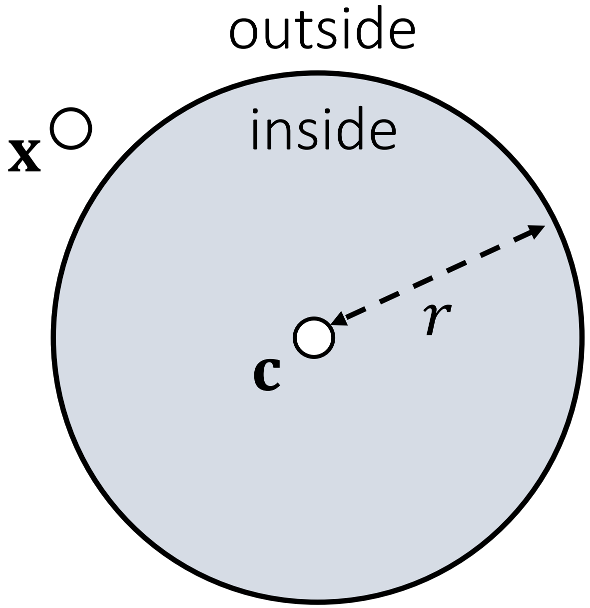
Signed Distance Function

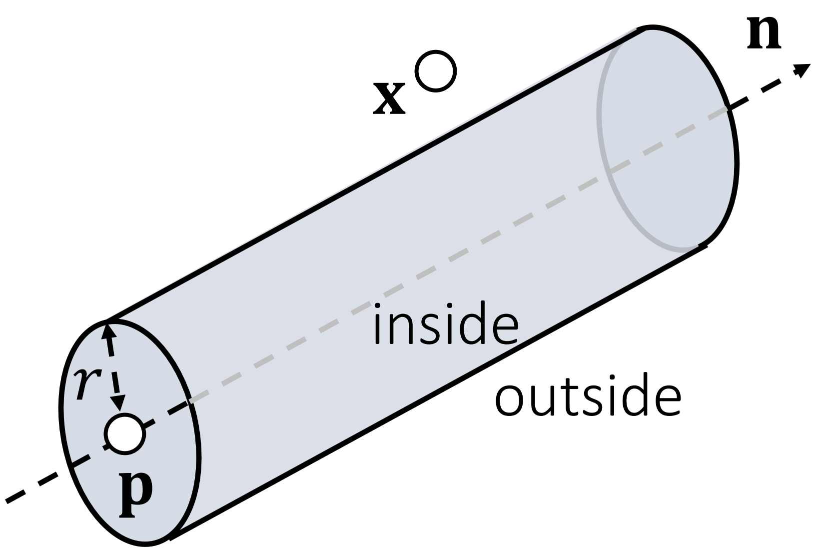
A signed distance function $\phi(\mathbf{x})$ defines the distance from $\mathbf{x}$ to a surface with a sign. The sign indicates on which side $\mathbf{x}$ is located.



Signed Distance Function Examples





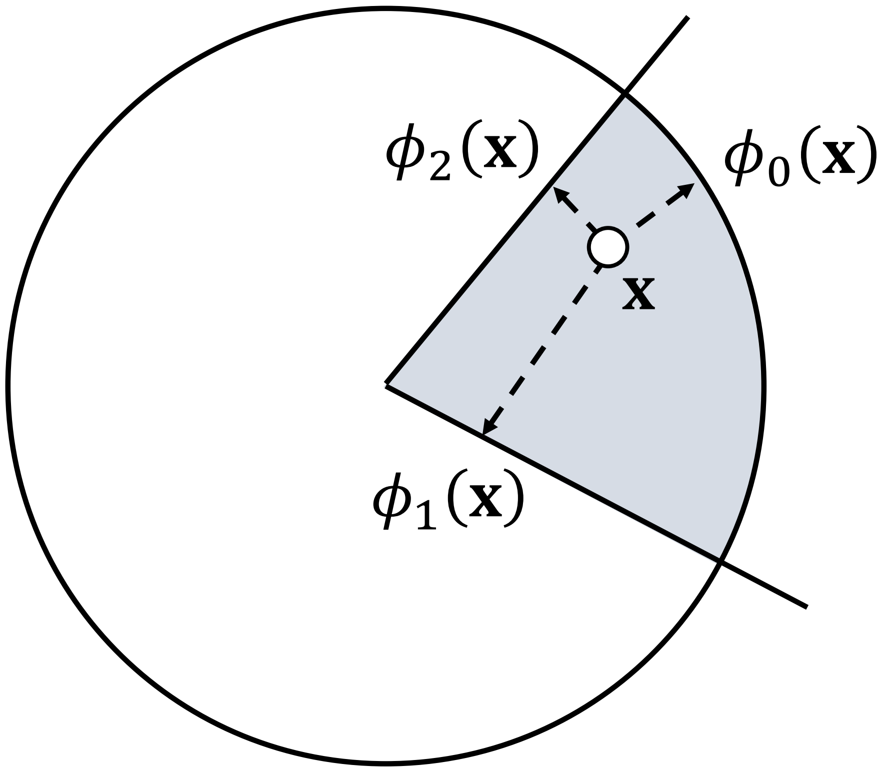


$$\phi(\mathbf{x}) = (\mathbf{x} - \mathbf{p}) \cdot \mathbf{n}$$

$$\phi(\mathbf{x}) = \|\mathbf{x} - \mathbf{c}\| - r$$

$$\phi(\mathbf{x}) = \sqrt{\|\mathbf{x} - \mathbf{p}\|^2 - \left((\mathbf{x} - \mathbf{p}) \cdot \mathbf{n}\right)^2} - r$$

Intersection of Signed Distance Functions



If $\phi_0(\mathbf{x}) < 0$ and $\phi_1(\mathbf{x}) < 0$ and $\phi_2(\mathbf{x}) < 0$

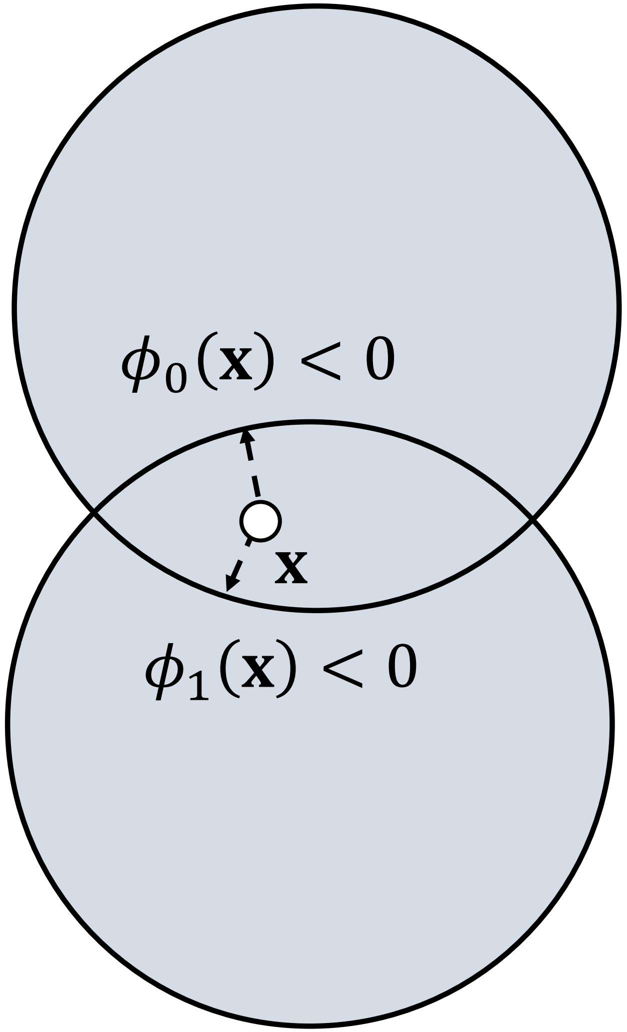
then inside

$$\phi(\mathbf{x}) = \max\big(\phi_0(\mathbf{x}), \phi_1(\mathbf{x}), \phi_2(\mathbf{x})\big)$$

Else outside

$$\phi(\mathbf{x}) = ?$$

Union of Signed Distance Functions



If $\phi_0(\mathbf{x}) < 0$ or $\phi_1(\mathbf{x}) < 0$

then inside

$$\phi(\mathbf{x}) \boxed{\approx} \min\big(\phi_0(\mathbf{x}), \phi_1(\mathbf{x})\big)$$

Else outside

Correct near outer boundary

$$\phi(\mathbf{x}) = \min\big(\phi_0(\mathbf{x}), \phi_1(\mathbf{x})\big)$$

Intuitively, we can consider collision detection with the union of two objects as collision detection with two separate objects.

Quadratic Penalty Method

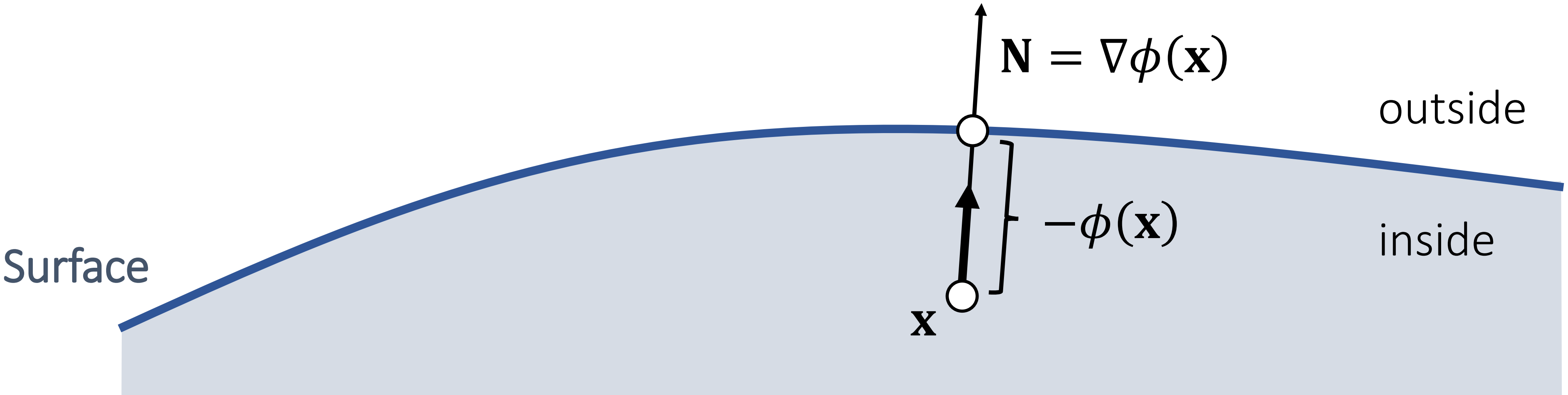
A penalty method applies a penalty force in the next update. When the penalty potential is quadratic, the force is linear.

Is $\phi(\mathbf{x}) < 0$?

yes → Apply a force in the next update:

$$\mathbf{f} \longleftarrow -\underbrace{k}_{\text{penalty strength}} \, \phi(\mathbf{x}) \, \mathbf{N}$$

no → Done



Quadratic Penalty Method with a Buffer

A buffer helps lessen the penetration issue. But it cannot strictly prevent penetration, no matter how large k is.

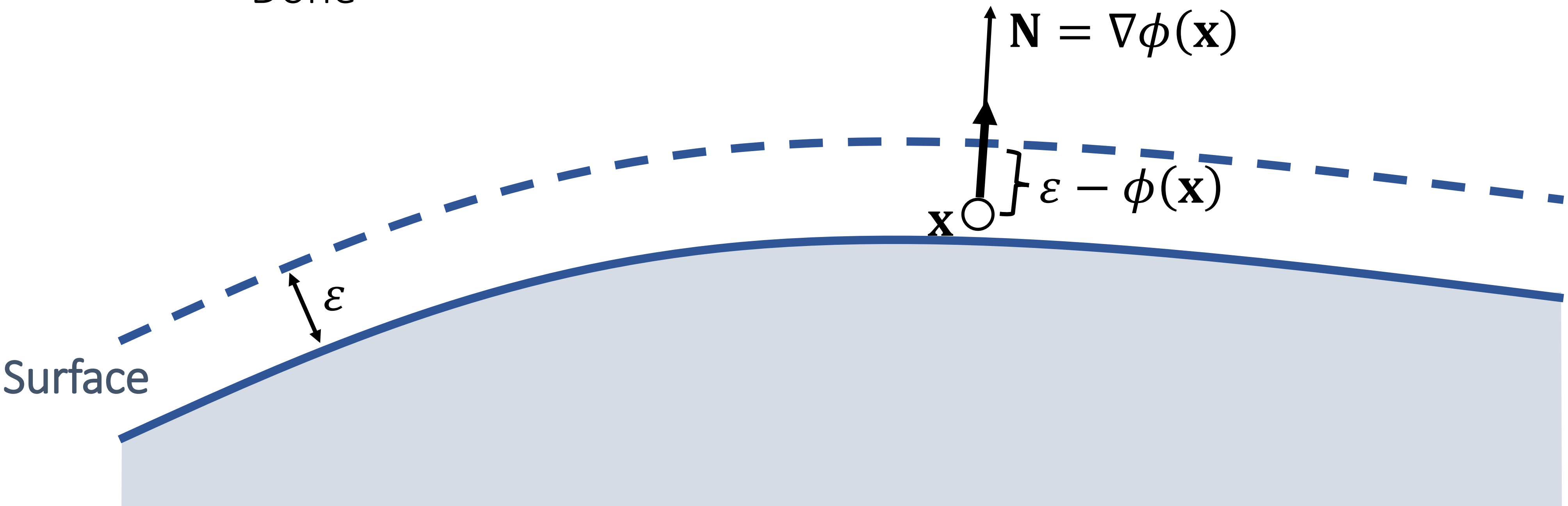
Is $\phi(\mathbf{x}) < \varepsilon$?

yes →

Apply a force in the next update:
$\mathbf{f} \leftarrow k\big(\varepsilon - \phi(\mathbf{x})\big)\mathbf{N}$

no ↓

Done

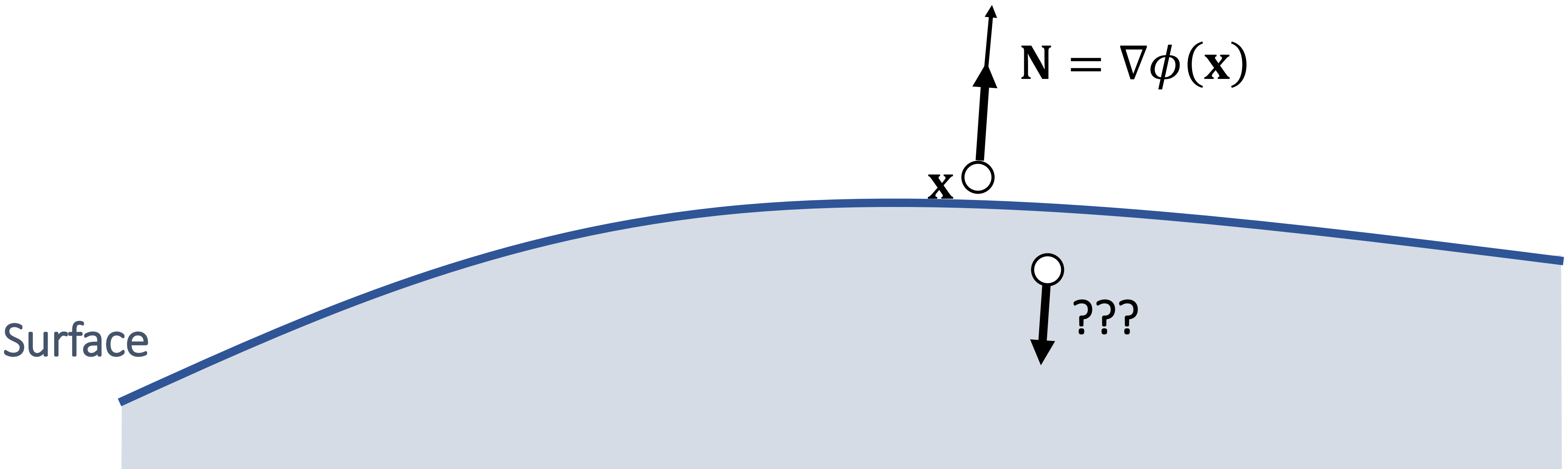


Log-Barrier Penalty Method

A log-barrier penalty potential ensures that the force can be large enough. But it assumes $\phi(\mathbf{x}) < 0$ will never happen!!! To achieve that, it needs to adjust Δt.

Always apply the penalty force as:

$$\mathbf{f} \longleftarrow \underbrace{\rho}_{\text{barrier strength}} \frac{1}{\phi(\mathbf{x})}\mathbf{N}$$



A Short Summary of Penalty Methods

- The use of step size adjustment is a must.
  - To avoid overshooting.
  - To avoid penetration in log-barrier methods.
- Log-barrier method can be limited within a buffer as well.
  - Li et al. 2020. *Incremental Potential Contact: Intersection- and Inversion-free Large Deformation Dynamics*. TOG.
  - Wu et al. 2020. *A Safe and Fast Repulsion Method for GPU-based Cloth Self Collisions*. TOG.
- Frictional contacts are difficult to handle.

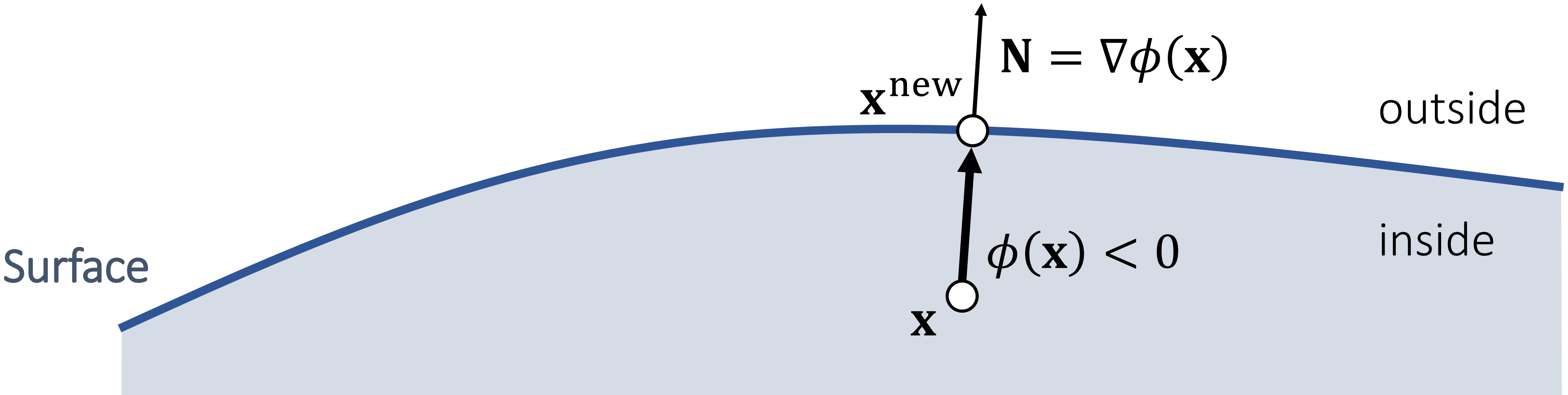
Impulse Method

An impulse method assumes that collision changes the position and the velocity all of sudden.

Is $\phi(\mathbf{x}) < 0$? —yes→ collision: $\mathbf{x}^{\text{new}} \leftarrow \mathbf{x} + |\phi(\mathbf{x})|\mathbf{N} = \mathbf{x} - \phi(\mathbf{x})\nabla\phi(\mathbf{x})$

no ↓ Done

What about velocity and friction?



Impulse Method

Changing the position is not enough, we must change the velocity as well.

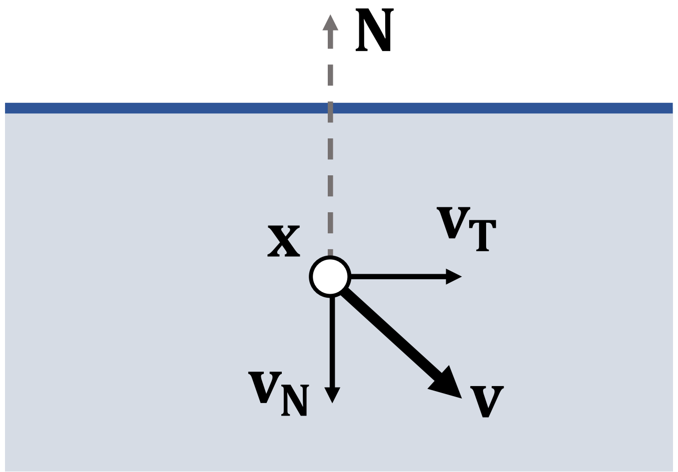
Is $\mathbf{v} \cdot \mathbf{N} < 0?$ yes →

$$\begin{cases} \mathbf{v_N} \leftarrow (\mathbf{v} \cdot \mathbf{N})\mathbf{N} \\ \mathbf{v_T} \leftarrow \mathbf{v} - \mathbf{v_N} \end{cases} \Rightarrow \begin{cases} \mathbf{v_N^{new}} \leftarrow -\mu_\mathbf{N} \mathbf{v_N} \\ \mathbf{v_T^{new}} \leftarrow a\, \mathbf{v_T} \end{cases} \Rightarrow \mathbf{v^{new}} \leftarrow \mathbf{v_N^{new}} + \mathbf{v_T^{new}}$$

Done

no ↓

Done



a should be minimized but not violating Coulomb's law

$$\|\mathbf{v_T^{new}} - \mathbf{v_T}\| \le \mu_\mathbf{T} \|\mathbf{v_N^{new}} - \mathbf{v_N}\|$$

$$(1-a)\|\mathbf{v_T}\| \le \mu_\mathbf{T}(1+\mu_\mathbf{N})\|\mathbf{v_N}\|$$

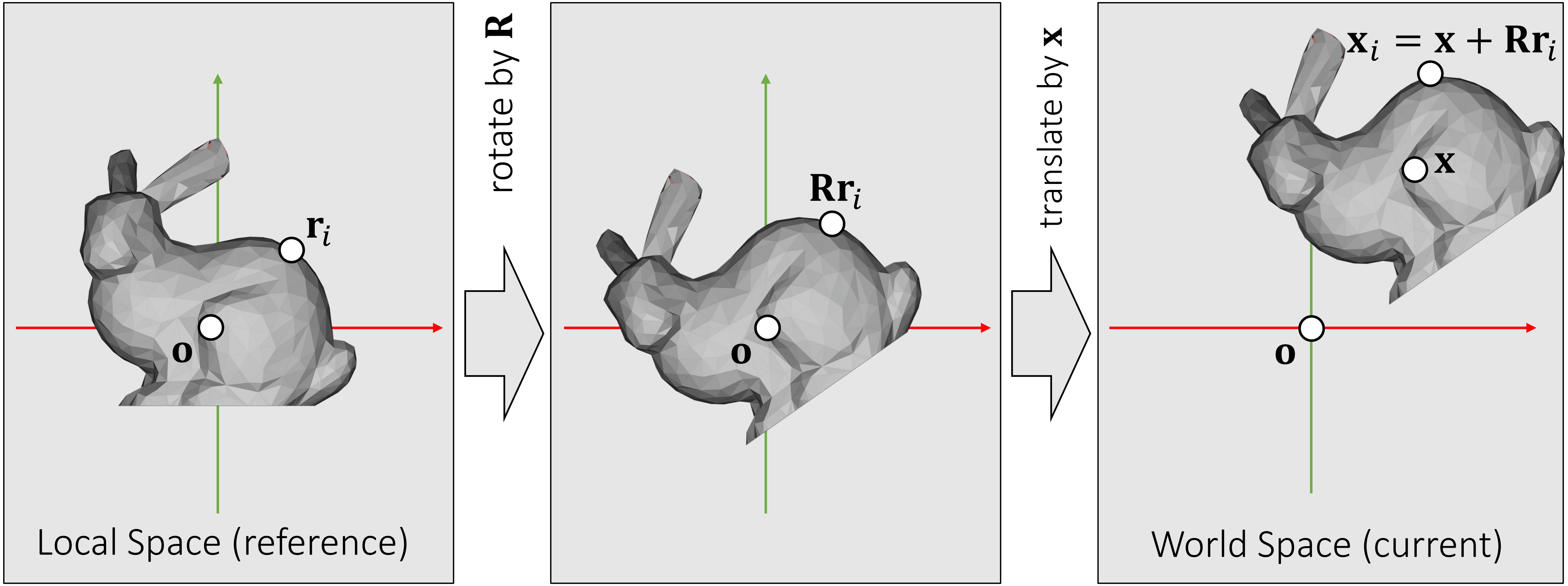
Therefore,

$$a \leftarrow \max(\underbrace{\boxed{1 - \mu_\mathbf{T}(1+\mu_\mathbf{N})\|\mathbf{v_N}\| / \|\mathbf{v_T}\|}}_{\text{dynamic friction}}, \underbrace{\boxed{0}}_{\text{static friction}})$$

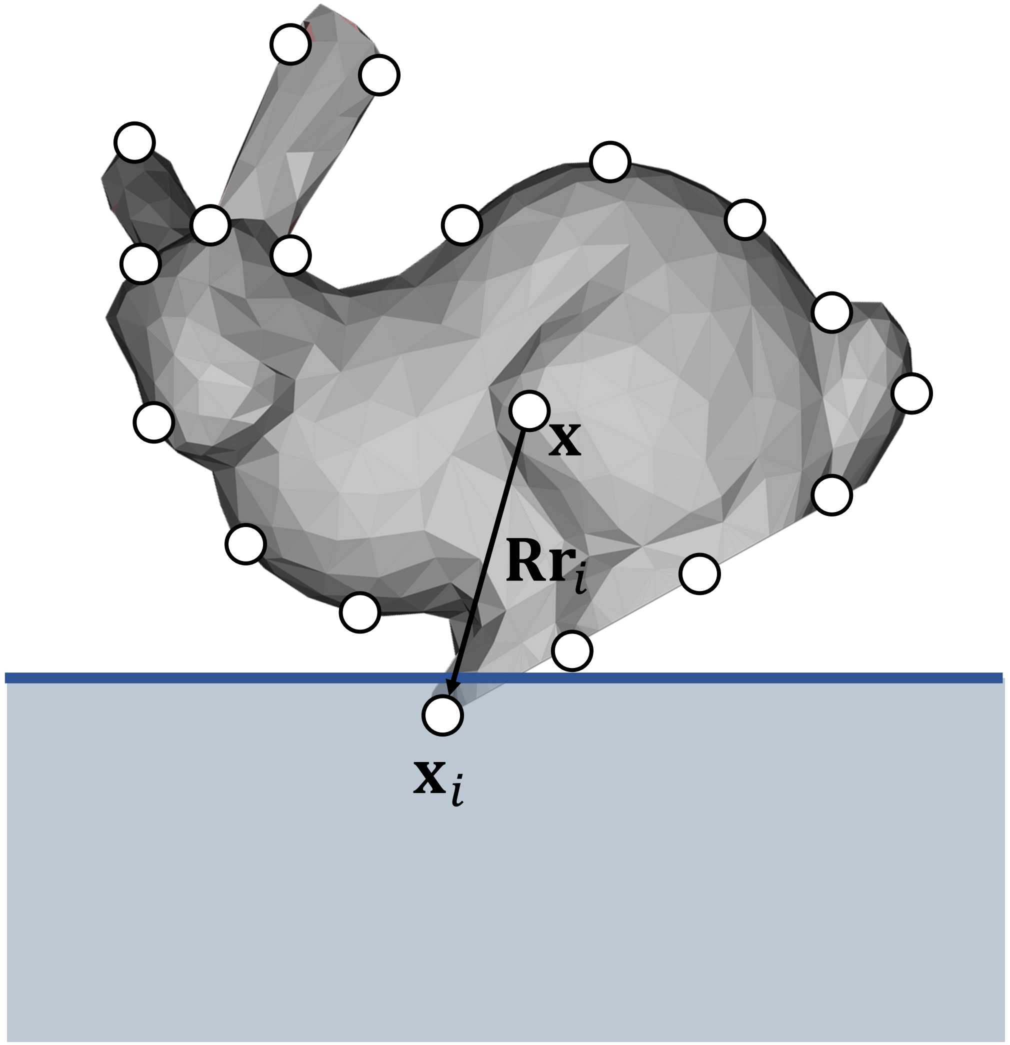
Rigid Body Collision Detection and Response

Remember that…

If a rigid body cannot deform, its motion consists of two parts: translation and rotation.



Rigid Body Collision Detection

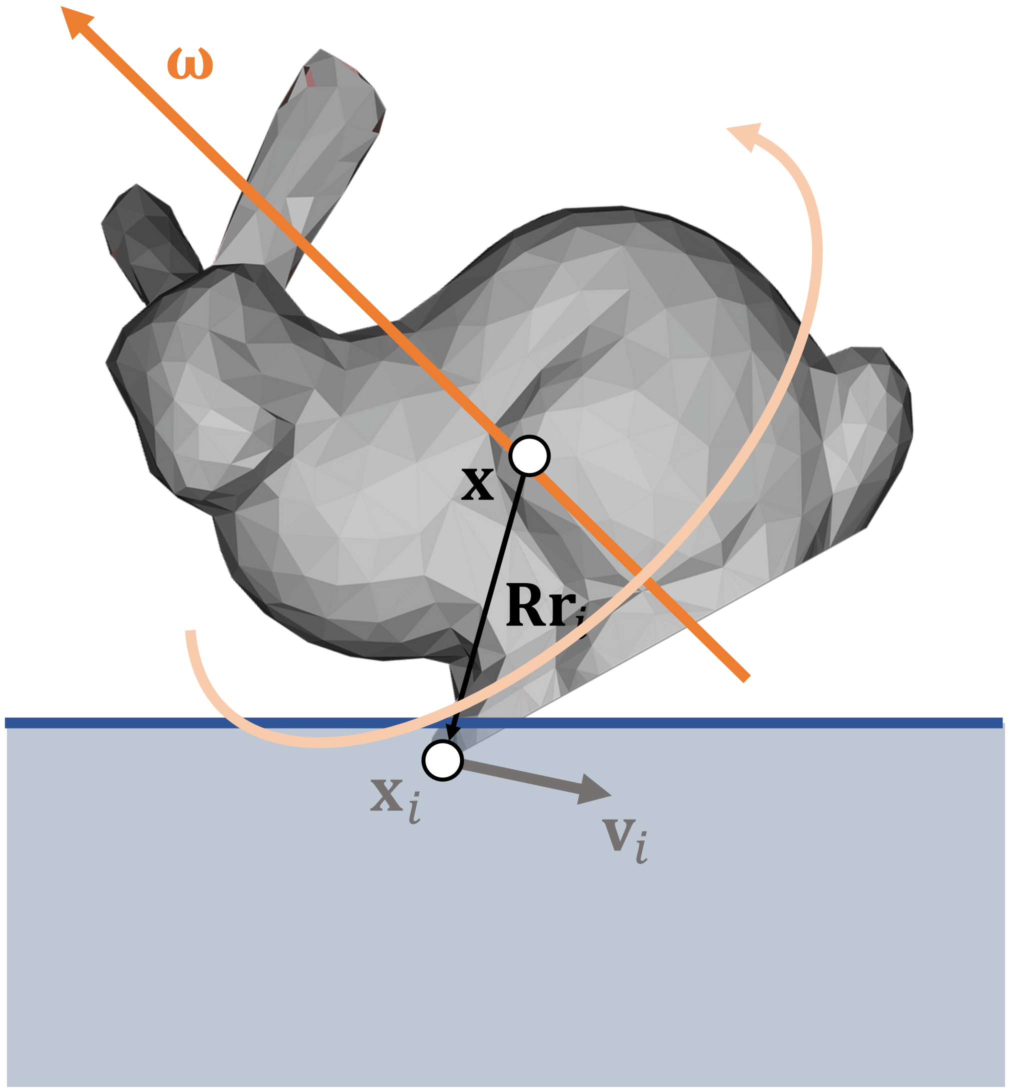


When the body is made of many vertices, we can detect its collision by testing each vertex:

$$\mathbf{x}_i \longleftarrow \mathbf{x} + \mathbf{R}\mathbf{r}_i$$

No a perfect solution, but acceptable (will come back to this weeks later…)

Rigid Body Collision Response by Impulse



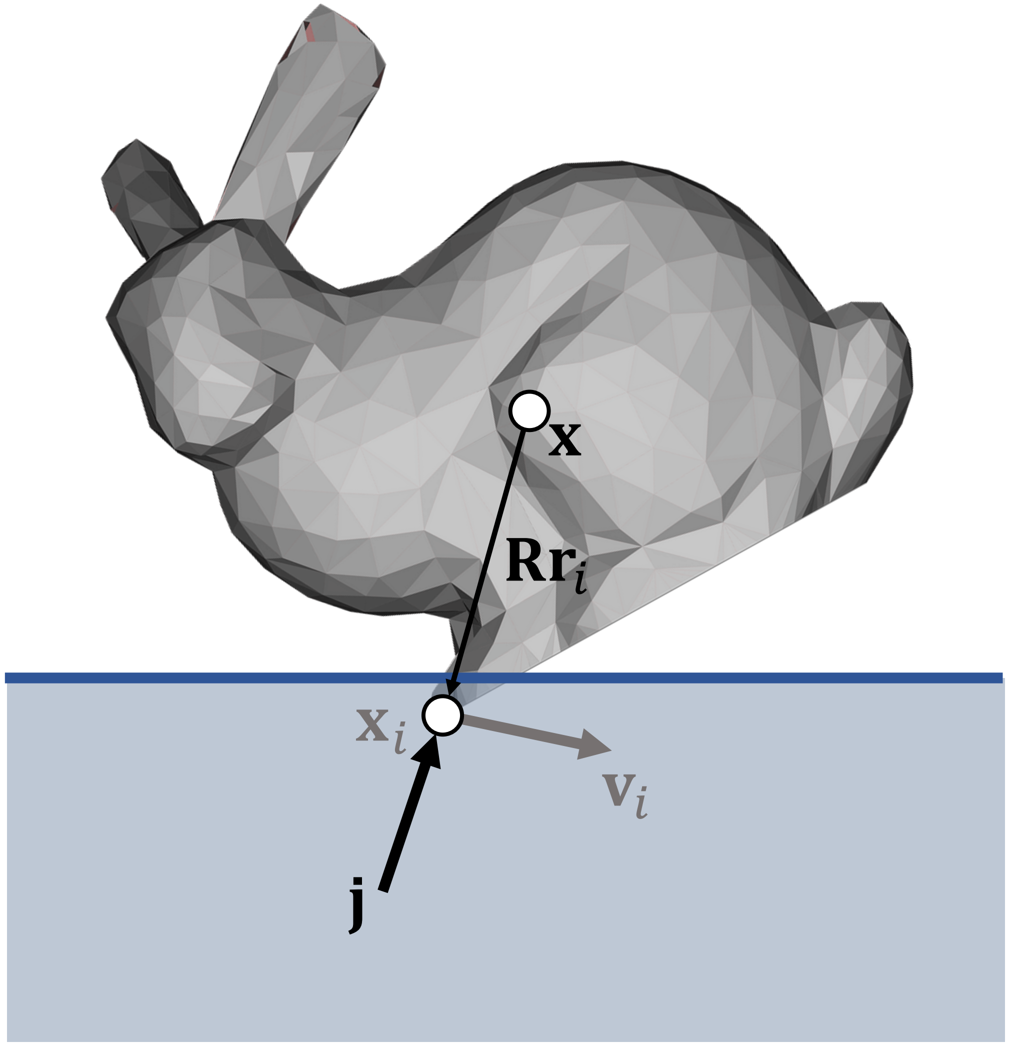
Vertex *i*:

$$\begin{cases} \mathbf{x}_i \leftarrow \mathbf{x} + \mathbf{R}\mathbf{r}_i & \text{(Position)} \\ \mathbf{v}_i \leftarrow \underbrace{\mathbf{v}}_{\text{linear velocity}} + \underbrace{\boldsymbol{\omega}}_{\text{angular velocity}} \times \mathbf{R}\mathbf{r}_i & \text{(Velocity)} \end{cases}$$

Problem: we cannot directly modify $\mathbf{x}_i$ or $\mathbf{v}_i$, since they not state variables. They are indirectly determined.

Solution: we will find a way to modify $\mathbf{v}$ and $\boldsymbol{\omega}$.

Rigid Body Collision Response by Impulse



What happens to $\mathbf{v}_i$ when an impulse $\mathbf{j}$ is applied at vertex i?

$$\begin{cases} \mathbf{v}^{\text{new}} \longleftarrow \mathbf{v} + \frac{1}{M}\mathbf{j} \\ \boldsymbol{\omega}^{\text{new}} \longleftarrow \boldsymbol{\omega} + \mathbf{I}^{-1}(\underbrace{\boxed{\mathbf{Rr}_i \times \mathbf{j}}}_{\text{torque induced by } \mathbf{j}}) \end{cases}$$

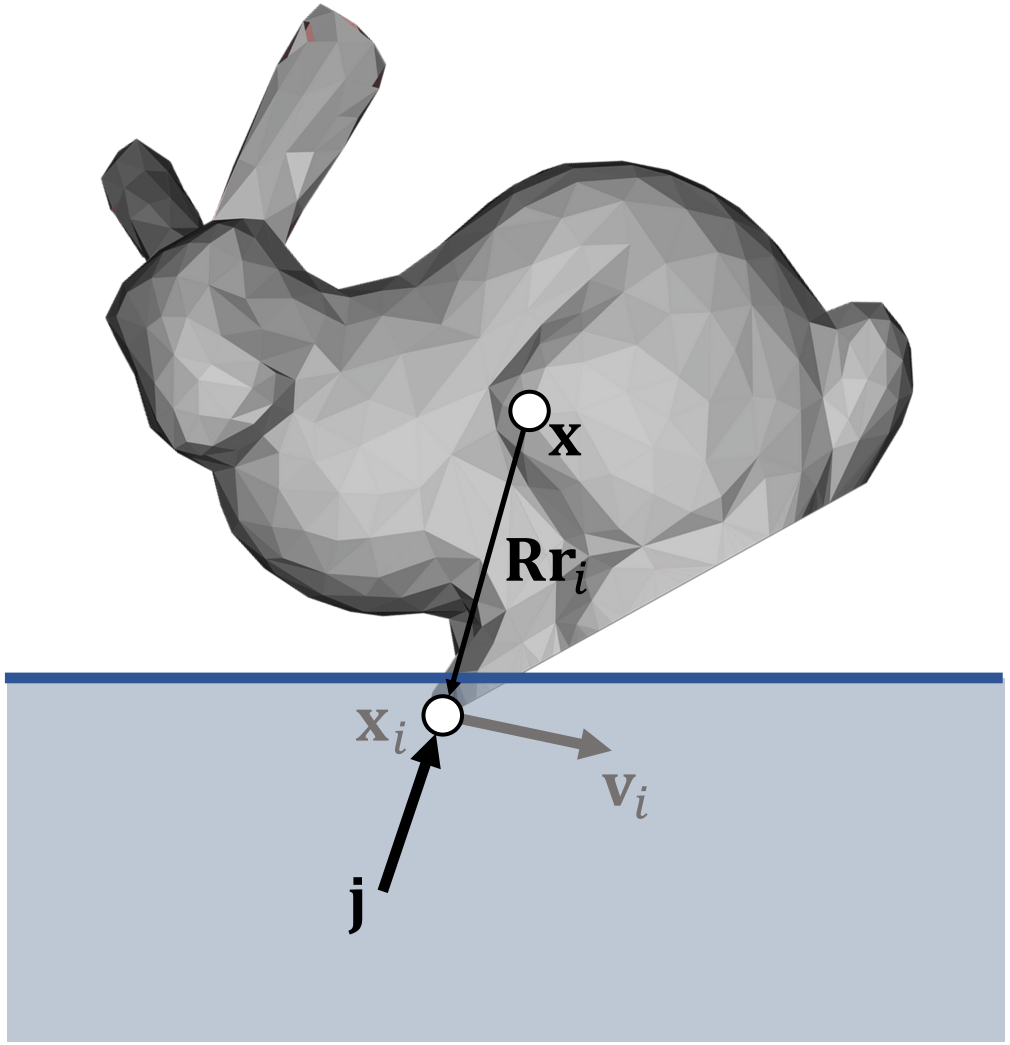
$$\begin{aligned} \mathbf{v}_i^{\text{new}} &= \mathbf{v}^{\text{new}} + \boldsymbol{\omega}^{\text{new}} \times \mathbf{Rr}_i \\ &= \mathbf{v} + \frac{1}{M}\mathbf{j} + (\boldsymbol{\omega} + \mathbf{I}^{-1}(\mathbf{Rr}_i \times \mathbf{j})) \times \mathbf{Rr}_i \\ &= \mathbf{v}_i + \frac{1}{M}\mathbf{j} + (\mathbf{I}^{-1}(\mathbf{Rr}_i \times \mathbf{j})) \times \mathbf{Rr}_i \\ &= \mathbf{v}_i + \frac{1}{M}\mathbf{j} - (\mathbf{Rr}_i) \times (\mathbf{I}^{-1}(\mathbf{Rr}_i \times \mathbf{j})) \end{aligned}$$

Cross Product as a Matrix Product

We can convert the cross product $\mathbf{r}\times$ into a matrix product $\mathbf{r}^*$.

$$\mathbf{r}\times\mathbf{q} = \begin{bmatrix} r_y q_z - r_z q_y \\ r_z q_x - r_x q_z \\ r_x q_y - r_y q_x \end{bmatrix} = \boxed{\begin{bmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix}} \begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix} = \mathbf{r}^*\mathbf{q}$$

Rigid Body Collision Response by Impulse



What happens to $\mathbf{v}_i$ when an impulse $\mathbf{j}$ is applied at vertex *i*? (continuing from page 18)

$$\mathbf{v}_i^{\text{new}} = \mathbf{v}_i + \frac{1}{M}\mathbf{j} - (\mathbf{Rr}_i) \times \left(\mathbf{I}^{-1}(\mathbf{Rr}_i \times \mathbf{j})\right)$$

$$\mathbf{v}_i^{\text{new}} = \mathbf{v}_i + \frac{1}{M}\mathbf{j} - (\mathbf{Rr}_i)^*\mathbf{I}^{-1}(\mathbf{Rr}_i)^*\mathbf{j}$$

$$\mathbf{v}_i^{\text{new}} - \mathbf{v}_i = \mathbf{Kj}$$

$$\mathbf{K} \longleftarrow \frac{1}{M}\mathbf{1} - (\mathbf{Rr}_i)^*\mathbf{I}^{-1}(\mathbf{Rr}_i)^*$$

Rigid Body Collision Response by Impulse

For every vertex $\mathbf{x}_i \leftarrow \mathbf{x} + \mathbf{R}\mathbf{r}_i$
Test if $\phi(\mathbf{x}_i) < 0$

no

Done

$\mathbf{v}_i \leftarrow \mathbf{v} + \boldsymbol{\omega} \times \mathbf{R}\mathbf{r}_i$
Is $\mathbf{v}_i \cdot \mathbf{N} < 0$?

no

Compute the wanted $\mathbf{v}_i^{\text{new}}$

$$\mathbf{v}_{\mathbf{N},i} \leftarrow (\mathbf{v}_i \cdot \mathbf{N})\mathbf{N}$$

$$\mathbf{v}_{\mathbf{T},i} \leftarrow \mathbf{v}_i - \mathbf{v}_{\mathbf{N},i}$$

$$a \leftarrow \max(1 - \mu_{\mathbf{T}}(1 + \mu_{\mathbf{N}})\|\mathbf{v}_{\mathbf{N},i}\| / \|\mathbf{v}_{\mathbf{T},i}\|, 0)$$

$$\mathbf{v}_{\mathbf{N},i}^{\text{new}} \leftarrow -\mu_{\mathbf{N}}\mathbf{v}_{\mathbf{N},i}$$

$$\mathbf{v}_{\mathbf{T},i}^{\text{new}} \leftarrow a\,\mathbf{v}_{\mathbf{T},i}$$

$$\mathbf{v}_i^{\text{new}} \leftarrow \mathbf{v}_{\mathbf{N},i}^{\text{new}} + \mathbf{v}_{\mathbf{T},i}^{\text{new}}$$

Compute the impulse $\mathbf{j}$

$$\mathbf{K} \leftarrow \frac{1}{M}\mathbf{1} - (\mathbf{R}\mathbf{r}_i)^*\mathbf{I}^{-1}(\mathbf{R}\mathbf{r}_i)^*$$

$$\mathbf{j} \leftarrow \mathbf{K}^{-1}(\mathbf{v}_i^{\text{new}} - \mathbf{v}_i)$$

Update $\mathbf{v}$ and $\boldsymbol{\omega}$

$$\mathbf{v} \leftarrow \mathbf{v} + \frac{1}{M}\mathbf{j}$$

$$\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} + \mathbf{I}^{-1}(\mathbf{R}\mathbf{r}_i \times \mathbf{j})$$

Done

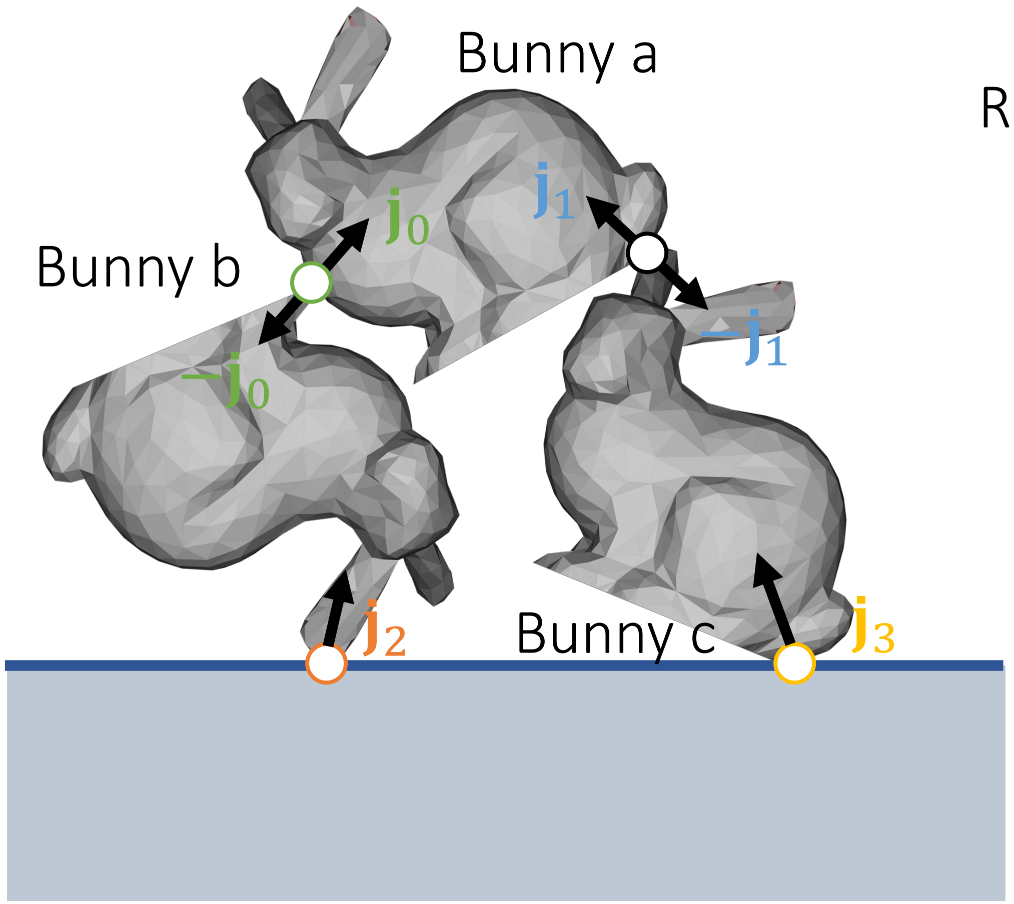
Some Implementation Details

- If there are many vertices in collision, we use their average.

- We can decrease the restitution $\mu_{\mathbf{N}}$ to reduce oscillation.

- We don't update the position here. Why?
  - Because the problem is nonlinear.
  - We will come back to this later when we talk about constraints.

Rigid Body Collision Response by Impulse



Relative velocity at joints

$$\begin{cases} \mathbf{v}_0^{\text{new}} - \mathbf{v}_0 = \mathbf{K}_{a00}\mathbf{j}_0 + \mathbf{K}_{a01}\mathbf{j}_1 - (-\mathbf{K}_{b00}\mathbf{j}_0 + \mathbf{K}_{b02}\mathbf{j}_2) \\ \mathbf{v}_1^{\text{new}} - \mathbf{v}_1 = \mathbf{K}_{a10}\mathbf{j}_0 + \mathbf{K}_{a11}\mathbf{j}_1 - (-\mathbf{K}_{c11}\mathbf{j}_0 + \mathbf{K}_{c13}\mathbf{j}_3) \\ \mathbf{v}_2^{\text{new}} - \mathbf{v}_2 = -\mathbf{K}_{b20}\mathbf{j}_0 + \mathbf{K}_{b22}\mathbf{j}_2 \\ \mathbf{v}_3^{\text{new}} - \mathbf{v}_3 = -\mathbf{K}_{c31}\mathbf{j}_1 + \mathbf{K}_{c33}\mathbf{j}_3 \end{cases}$$

$$\begin{bmatrix} \mathbf{K}_{a00} + \mathbf{K}_{b00} & \mathbf{K}_{a01} & -\mathbf{K}_{b02} & \\ \mathbf{K}_{a10} & \mathbf{K}_{a11} + \mathbf{K}_{c11} & & -\mathbf{K}_{c13} \\ -\mathbf{K}_{b20} & & \mathbf{K}_{b22} & \\ & -\mathbf{K}_{c31} & & \mathbf{K}_{c33} \end{bmatrix} \begin{bmatrix} \mathbf{j}_0 \\ \mathbf{j}_1 \\ \mathbf{j}_2 \\ \mathbf{j}_3 \end{bmatrix} = \begin{bmatrix} \Delta\mathbf{v}_0 \\ \Delta\mathbf{v}_1 \\ \Delta\mathbf{v}_2 \\ \Delta\mathbf{v}_3 \end{bmatrix}$$

$\mathbf{K}_{a01}\mathbf{j}_1$ stands for the velocity change of bunny *a* at joint 0, caused by impulse $\mathbf{j}_1$.

After-Class Reading (Before Collision)

https://graphics.pixar.com/pbm2001

Physically Based Modeling

ONLINE SIGGRAPH 2001 COURSE NOTES

Please note: the lecture notes served from this page are copyright ©2001 by the authors Andrew Witkin and David Baraff. Chapters may be freely duplicated and distributed so long as no consideration is received in return, and this copyright notice remains intact. The slide sets are copyright ©2001 and may be neither distributed nor duplicated without permission of the authors.

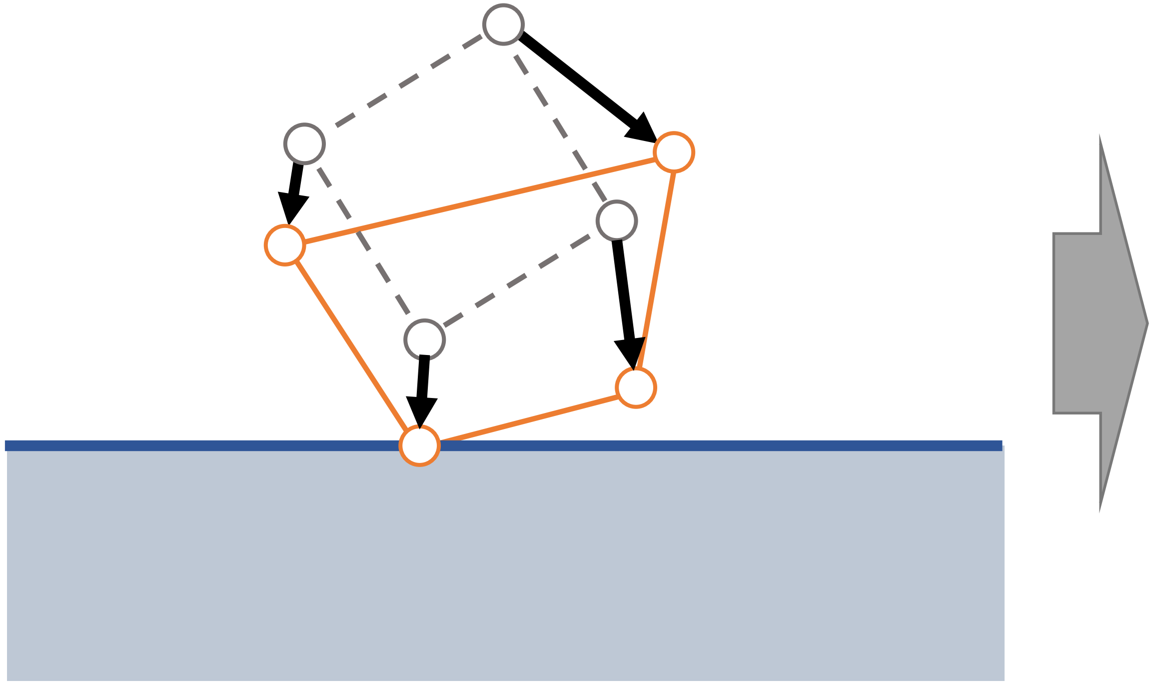
All documents on this page are in Adobe Acrobat format. If you need to obtain an Acrobat reader, please visit the Adobe Acrobat Reader page.

- **Introduction**
- **Differential Equation Basics**
  - Lecture Notes
  - Slides
- **Particle Dynamics**
  - Lecture Notes
  - Slides
- **Implicit Methods**
  - Lecture Notes
  - Slides
- **Cloth and Fur Energy Functions**
  - Slides
- **Rigid Body Dynamics**
  - Lecture Notes
  - Slides
- **Constrained Dynamics**
  - Lecture Notes
  - Slides
- **Collision and Contact**
  - Slides

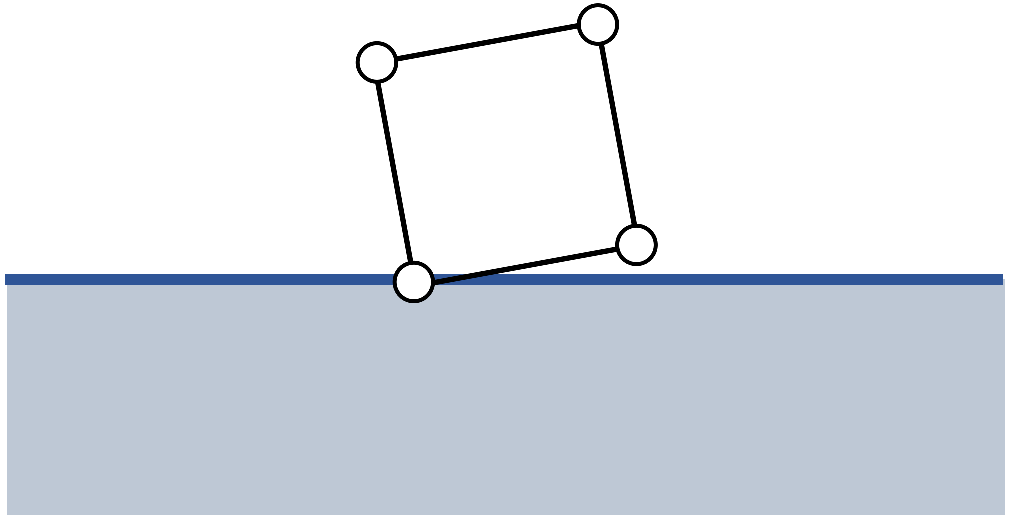
Shape Matching

Basic Idea

We allow each vertex to have its own velocity, so it can move by itself.

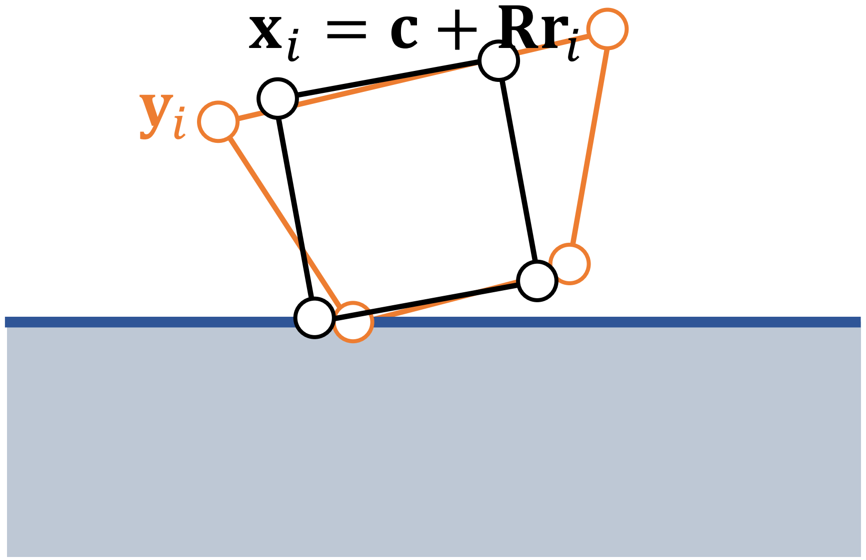


First, move vertices independently by its velocity, with collision and friction being handled.



Second, enforce the rigidity constraint to become a rigid body again.

Mathematical Formulation



Now $\mathbf{c}$ and $\mathbf{R}$ are unknowns we want to find out from:

$$\{\mathbf{c}, \mathbf{R}\} = \operatorname{argmin} \sum_i \frac{1}{2}\|\mathbf{c} + \mathbf{R}\mathbf{r}_i - \mathbf{y}_i\|^2$$

$$\{\mathbf{c}, \boxed{\mathbf{A}}\} = \operatorname{argmin} \sum_i \boxed{\frac{1}{2}\|\mathbf{c} + \mathbf{A}\mathbf{r}_i - \mathbf{y}_i\|^2}$$

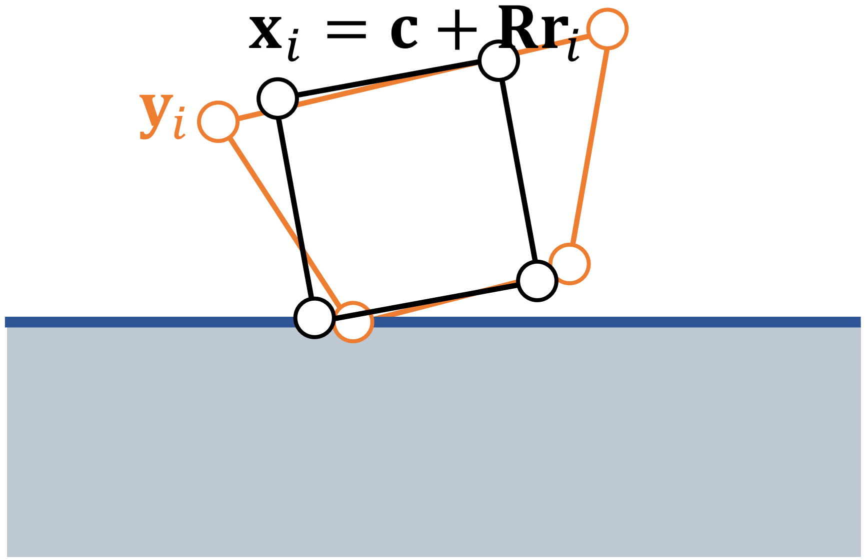
any matrix

The objective E

$$\frac{\partial E}{\partial \mathbf{c}} = \sum_i \mathbf{c} + \mathbf{A}\mathbf{r}_i - \mathbf{y}_i = \sum_i \mathbf{c} - \mathbf{y}_i = \mathbf{0}$$

$$\mathbf{c} = \frac{1}{N}\sum_i \mathbf{y}_i$$

Mathematical Formulation



Next, $\{\mathbf{c}, \mathbf{A}\} = \operatorname{argmin} \sum_i \frac{1}{2} \|\mathbf{c} + \mathbf{A}\mathbf{r}_i - \mathbf{y}_i\|^2$

$$\frac{\partial E}{\partial \mathbf{A}} = \sum_i (\mathbf{c} + \mathbf{A}\mathbf{r}_i - \mathbf{y}_i)\mathbf{r}_i^{\mathrm{T}} = \mathbf{0}$$

$$\mathbf{A} = \left(\sum_i (\mathbf{y}_i - \mathbf{c})\ \mathbf{r}_i^{\mathrm{T}}\right)\left(\sum_i \mathbf{r}_i\ \mathbf{r}_i^{\mathrm{T}}\right)^{-1}$$

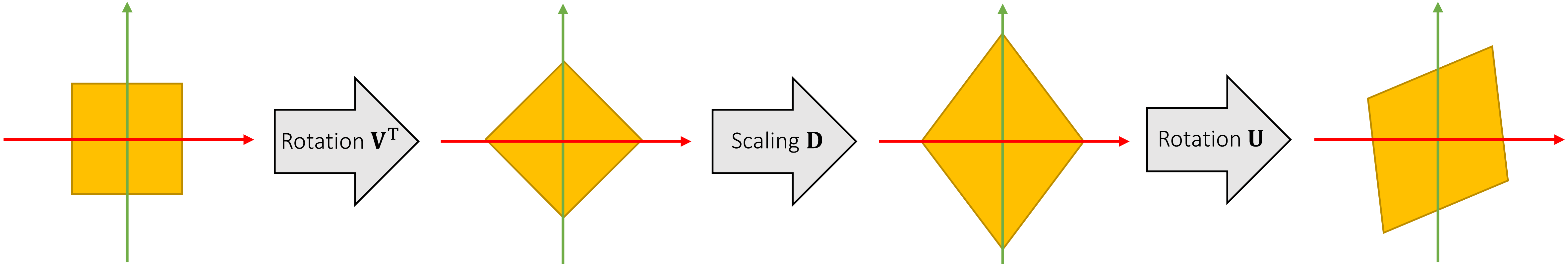
Polar Decomposition

$$\mathbf{A} = \underbrace{\mathbf{R}}_{\text{rotation}}\ \underbrace{\mathbf{S}}_{\text{deformation}}$$

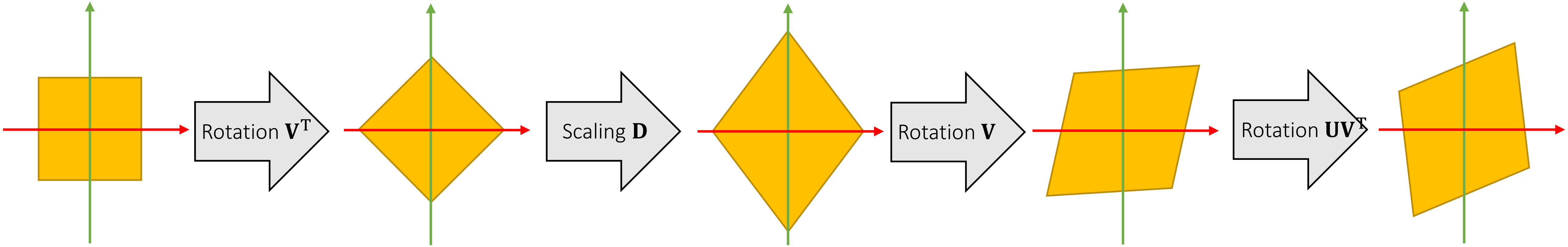
But why?

Remember that...

Singular value decomposition says any matrix can be decomposed into: rotation, scaling and rotation: $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^{\mathrm{T}}$.

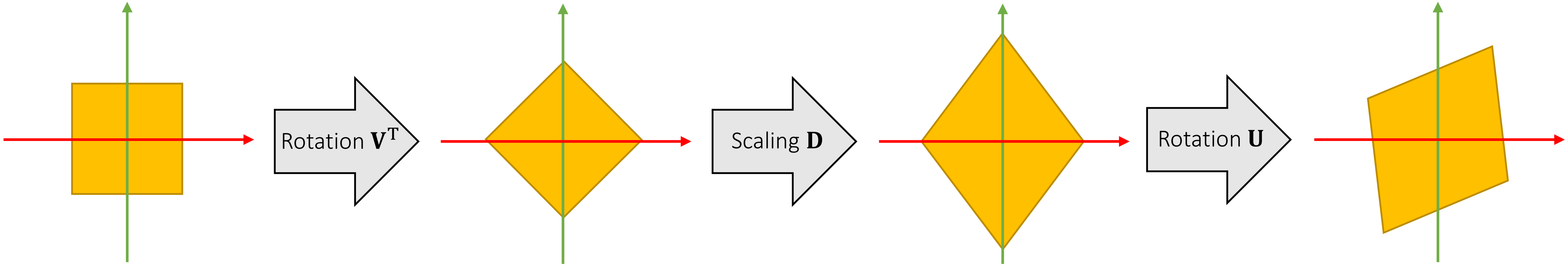


We can rotate the object back before the final rotation: $\mathbf{A} = \left(\mathbf{U}\mathbf{V}^{\mathrm{T}}\right)\left(\mathbf{V}\mathbf{D}\mathbf{V}^{\mathrm{T}}\right)$.

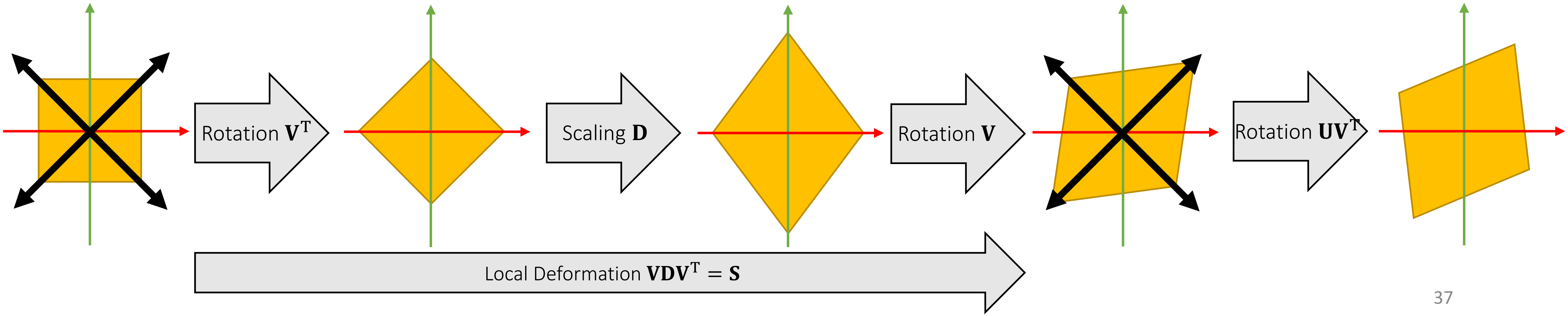


Remember that…

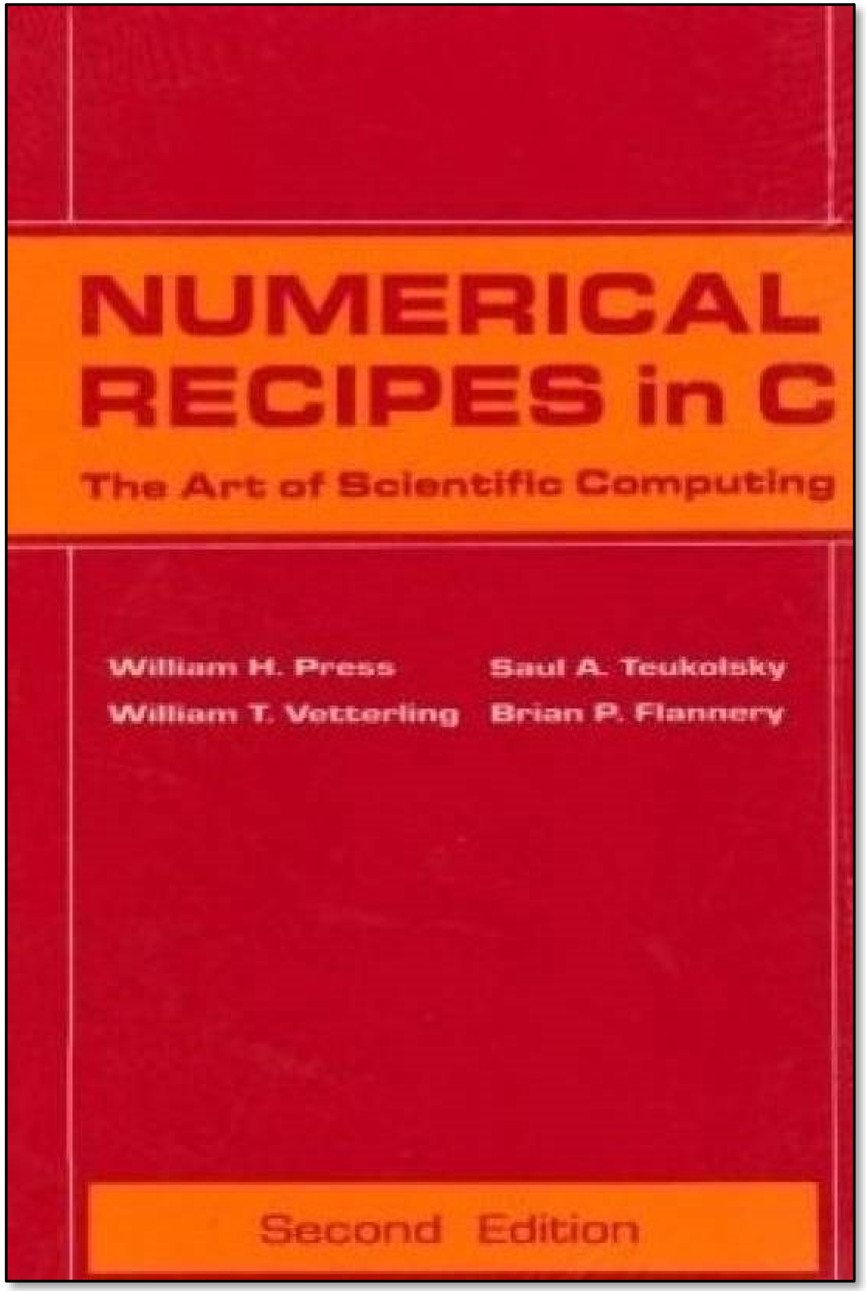
Singular value decomposition says any matrix can be decomposed into: rotation, scaling and rotation: $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^{\mathrm{T}}$.



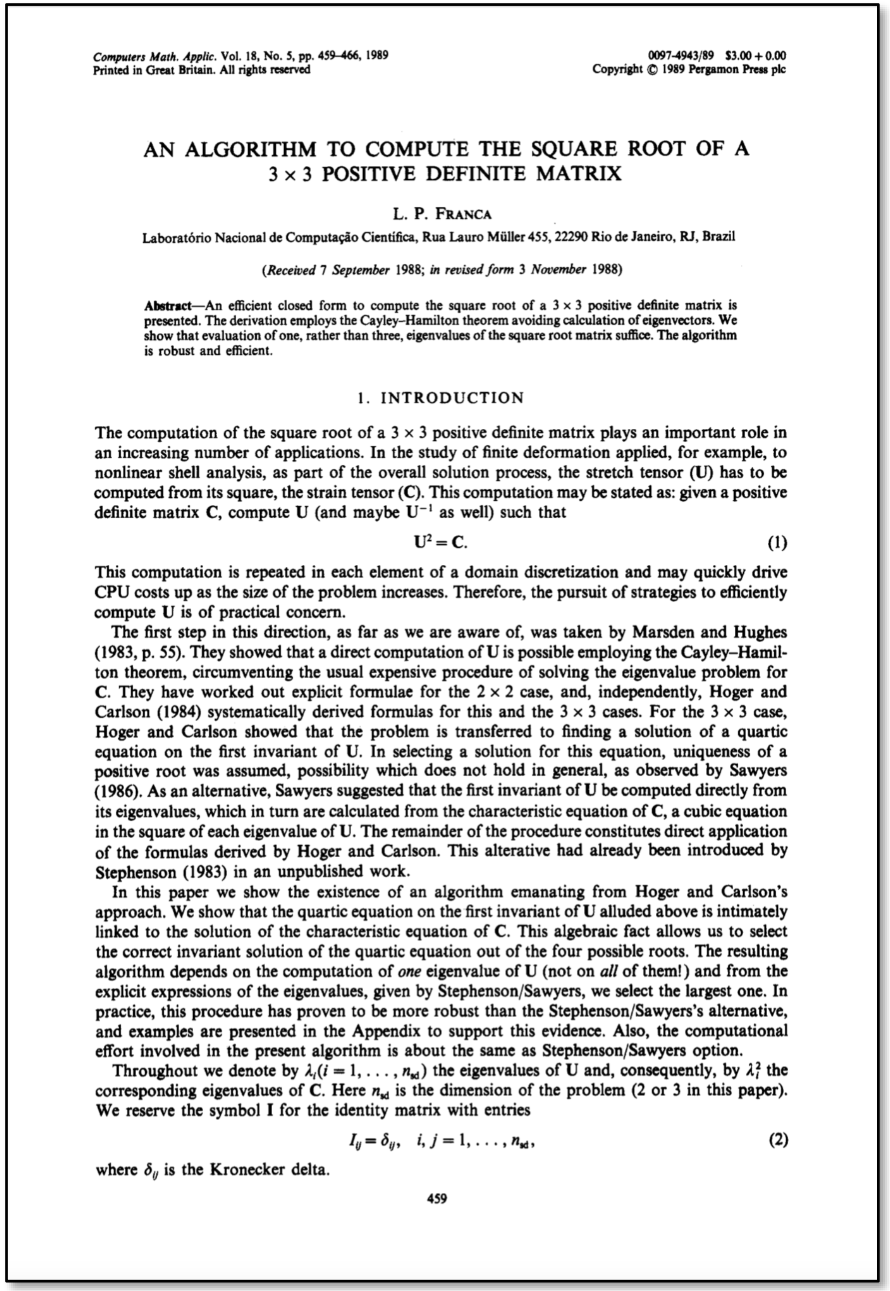
We can rotate the object back before the final rotation: $\mathbf{A} = \left(\mathbf{U}\mathbf{V}^{\mathrm{T}}\right)\left(\mathbf{V}\mathbf{D}\mathbf{V}^{\mathrm{T}}\right)$.



Polar Decomposition



Iterative methods



Computers Math. Applic. Vol. 18, No. 5, pp. 459–466, 1989
Printed in Great Britain. All rights reserved
0097-4943/89 \$3.00 + 0.00

AN ALGORITHM TO COMPUTE THE SQUARE ROOT OF A 3×3 POSITIVE DEFINITE MATRIX

L. P. Franca
Laboratório Nacional de Computação Científica, Rua Lauro Müller 455, 22290 Rio de Janeiro, RJ, Brazil

(*Received* 7 *September* 1988; *in revised form* 3 *November* 1988)

Abstract—An efficient closed form to compute the square root of a 3×3 positive definite matrix is presented. The derivation employs the Cayley–Hamilton theorem avoiding calculation of eigenvectors. We show that evaluation of one, rather than three, eigenvalues of the square root matrix suffice. The algorithm is robust and efficient.

1. INTRODUCTION

The computation of the square root of a 3×3 positive definite matrix plays an important role in an increasing number of applications. In the study of finite deformation applied, for example, to nonlinear shell analysis, as part of the overall solution process, the stretch tensor ($\mathbf{U}$) has to be computed from its square, the strain tensor ($\mathbf{C}$). This computation may be stated as: given a positive definite matrix $\mathbf{C}$, compute $\mathbf{U}$ (and maybe $\mathbf{U}^{-1}$ as well) such that

$$\mathbf{U}^2 = \mathbf{C}. \tag{1}$$

This computation is repeated in each element of a domain discretization and may quickly drive CPU costs up as the size of the problem increases. Therefore, the pursuit of strategies to efficiently compute $\mathbf{U}$ is of practical concern.

The first step in this direction, as far as we are aware of, was taken by Marsden and Hughes (1983, p. 55). They showed that a direct computation of $\mathbf{U}$ is possible employing the Cayley–Hamilton theorem, circumventing the usual expensive procedure of solving the eigenvalue problem for $\mathbf{C}$. They have worked out explicit formulae for the 2×2 case, and, independently, Hoger and Carlson (1984) systematically derived formulas for this and the 3×3 cases. For the 3×3 case, Hoger and Carlson showed that the problem is transferred to finding a solution of a quartic equation on the first invariant of $\mathbf{U}$. In selecting a solution for this equation, uniqueness of a positive root was assumed, possibility which does not hold in general, as observed by Sawyers (1986). As an alternative, Sawyers suggested that the first invariant of $\mathbf{U}$ be computed directly from its eigenvalues, which in turn are calculated from the characteristic equation of $\mathbf{C}$, a cubic equation in the square of each eigenvalue of $\mathbf{U}$. The remainder of the procedure constitutes direct application of the formulas derived by Hoger and Carlson. This alterative had already been introduced by Stephenson (1983) in an unpublished work.

In this paper we show the existence of an algorithm emanating from Hoger and Carlson's approach. We show that the quartic equation on the first invariant of $\mathbf{U}$ alluded above is intimately linked to the solution of the characteristic equation of $\mathbf{C}$. This algebraic fact allows us to select the correct invariant solution of the quartic equation out of the four possible roots. The resulting algorithm depends on the computation of *one* eigenvalue of $\mathbf{U}$ (not on *all* of them!) and from the explicit expressions of the eigenvalues, given by Stephenson/Sawyers, we select the largest one. In practice, this procedure has proven to be more robust than the Stephenson/Sawyers's alternative, and examples are presented in the Appendix to support this evidence. Also, the computational effort involved in the present algorithm is about the same as Stephenson/Sawyers option.

Throughout we denote by $\lambda_i (i = 1, \ldots, n_{sd})$ the eigenvalues of $\mathbf{U}$ and, consequently, by λ_i^2 the corresponding eigenvalues of $\mathbf{C}$. Here n_{sd} is the dimension of the problem (2 or 3 in this paper). We reserve the symbol $\mathbf{I}$ for the identity matrix with entries

$$I_{ij} = \delta_{ij}, \quad i, j = 1, \ldots, n_{sd}, \tag{2}$$

where δ_{ij} is the Kronecker delta.

Direct methods

$$\mathbf{A} = \mathbf{RS}$$

$$\mathbf{A}^{\mathrm{T}}\mathbf{A} = \mathbf{S}^{\mathrm{T}}\mathbf{S} = \mathbf{S}^2$$

unique

Shape Matching

Independent Update

For every vertex

$$\mathbf{f}_i \leftarrow \text{Force}(\mathbf{x}_i, \mathbf{v}_i)$$
$$\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t m_i^{-1} \mathbf{f}_i$$
$$\mathbf{y}_i \leftarrow \mathbf{x}_i + \Delta t\, \mathbf{v}_i$$

Rigidify the vertices

$$\mathbf{c} = \frac{1}{N} \sum_i \mathbf{y}_i$$
$$\mathbf{A} = \left(\sum_i (\mathbf{y}_i - \mathbf{c})\ \mathbf{r}_i^{\mathrm{T}} \right) \left(\sum_i \mathbf{r}_i\ \mathbf{r}_i^{\mathrm{T}} \right)^{-1}$$
$$\mathbf{R} = \text{Polar}(\mathbf{A})$$

Update $\mathbf{v}_i$ and $\mathbf{x}_i$

For every vertex

$$\mathbf{v}_i \leftarrow (\mathbf{c} + \mathbf{R}\mathbf{r}_i - \mathbf{x}_i)/\Delta t$$
$$\mathbf{x}_i \leftarrow \mathbf{c} + \mathbf{R}\mathbf{r}_i$$

Done

Physical quantities are attached to each vertex, not to the entire body.

Shape Matching

- Easy to implement and compatible with other nodal systems, i.e., cloth, soft bodies and even particle fluids.

- Difficult to strictly enforce friction and other goals.
  - The rigidification process will destroy them.

- More suitable when the friction accuracy is unimportant, i.e., buttons on clothes.

After-Class Reading

Muller et al. 2005.
Meshless Deformations Based on Shape Matching.
TOG (SIGGRAPH).

Meshless Deformations Based on Shape Matching

Matthias Müller, NovodeX/AGEIA & ETH Zürich
Bruno Heidelberger, ETH Zürich
Matthias Teschner, University of Freiburg
Markus Gross, ETH Zürich

Figure 1: The presented technique is stable under all circumstances and allows to simulate hundreds of deformable objects in real-time.

Abstract

We present a new approach for simulating deformable objects. The underlying model is geometrically motivated. It handles point-based objects and does not need connectivity information. The approach does not require any pre-processing, is simple to compute, and provides unconditionally stable dynamic simulations.

The main idea of our deformable model is to replace energies by geometric constraints and forces by distances of current positions to goal positions. These goal positions are determined via a generalized shape matching of an undeformed rest state with the current deformed state of the point cloud. Since points are always drawn towards well-defined locations, the overshooting problem of explicit integration schemes is eliminated. The versatility of the approach in terms of object representations that can be handled, the efficiency in terms of memory and computational complexity, and the unconditional stability of the dynamic simulation make the approach particularly interesting for games.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically Based Modeling; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation and Virtual Reality

Keywords: deformable modeling, geometric deformation, shape matching, real-time simulation

1 Introduction

Since Terzopoulos' pioneering work on simulating deformable objects in the context of computer graphics [Terzopoulos et al. 1987], many deformable models have been proposed. In general, these approaches focus on an accurate material representation, on stability aspects of the dynamic simulation and on versatility in terms of advanced object characteristics that can be handled, e. g. plastic deformation or fracturing.

Despite the long history of deformable modeling in computer graphics, research results have rarely been applied in computer games. Nowadays, deformable cloth models with simple geometries can be found in a few games, but in general, games are dominated by rigid bodies. Although rigid bodies can be linked with joints to represent articulated structures, there exist no practical solution which allows to simulate elastically deformable three-dimensional objects in a stable and efficient way. There are several reasons that prevent current deformable models from being used in interactive applications.

Efficiency. Existing deformable models based on complex material laws in conjunction with stable, implicit integration schemes are computationally expensive. Such approaches do not allow for interactive simulations of objects with a reasonable geometrical complexity. Further, these approaches might require a specific object representation and the algorithms can be hard to implement and debug. In contrast, interactive applications such as games constitute hard constraints on the computational efficiency of a deformable modeling approach. The approach is only allowed to use a small fraction of the available computing resources. Further, specific volumetric representations of deformable objects are often not available since the geometries are typically represented by surfaces only.

Stability. In interactive applications, the simulation of deformable objects needs to remain stable under all circumstances. While sophisticated approaches allow for stable numerical integration of velocities and positions, additional error sources such as degenerated geometries, physically incorrect states, or problematic situations with large object interpenetrations are not addressed by many approaches. A first contribution to this research area has been presented in [Irving et al. 2004], where large deformations and the inversion of elements in FE approaches can be handled in a robust way. However, this approach is not intended to be used in interactive applications.