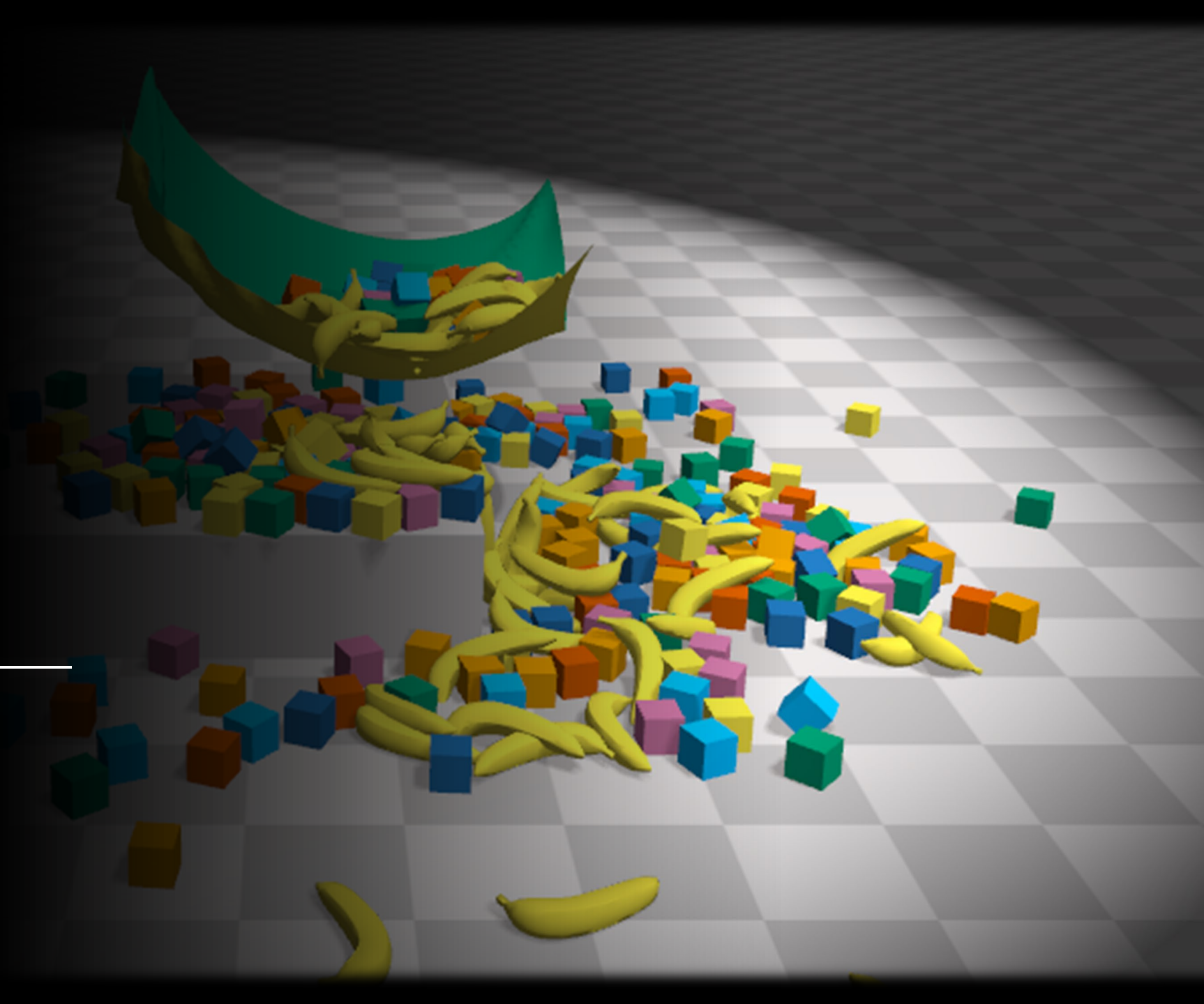
Collision Handling

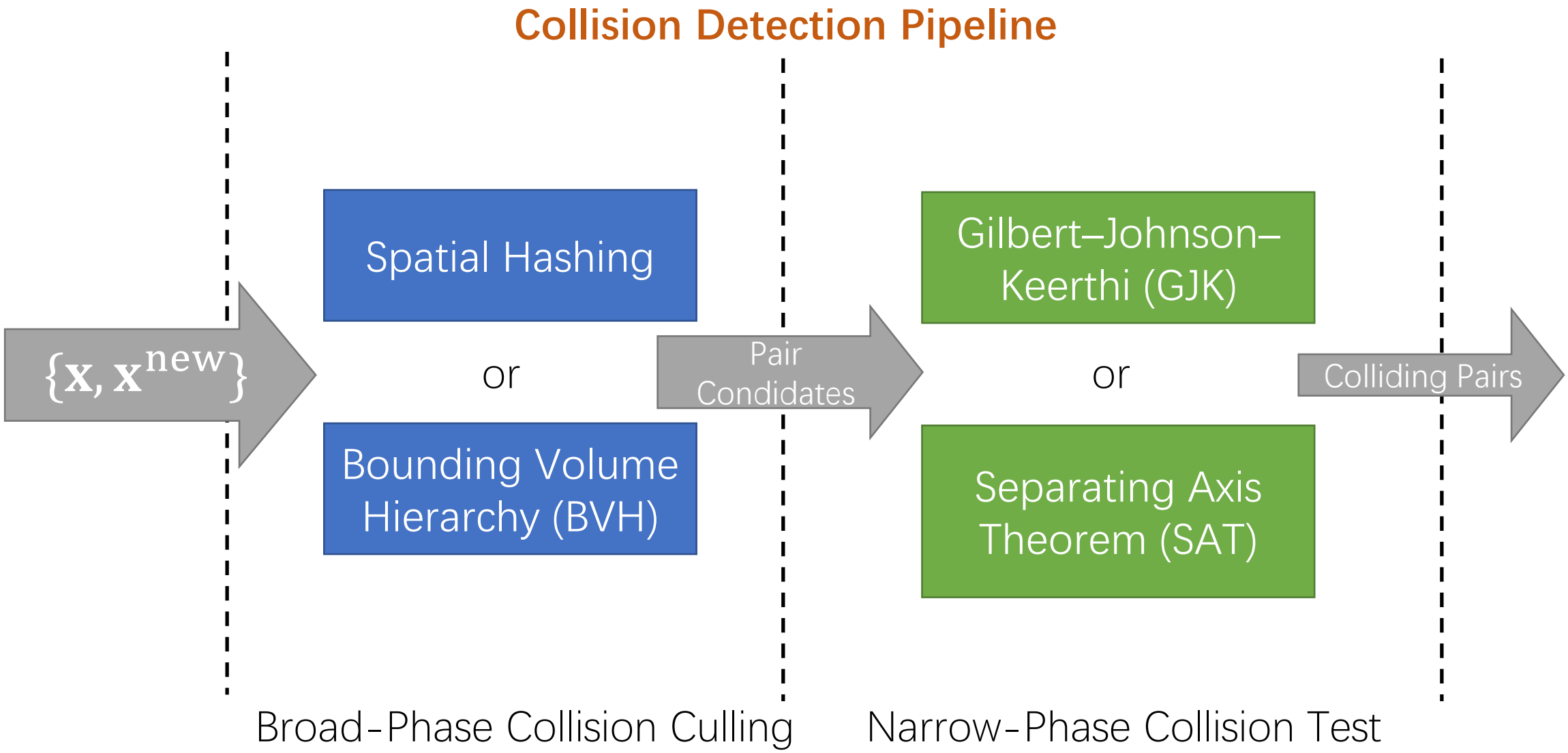
Interior Point Methods

Impact Zone Optimization

Intersection Elimination

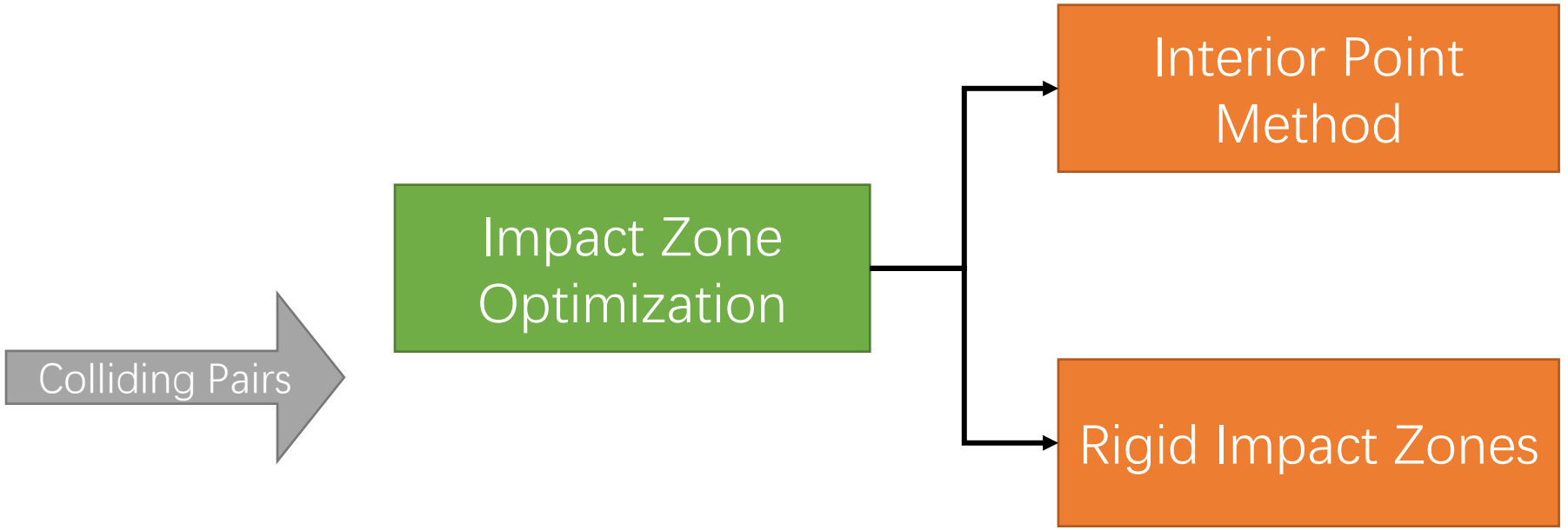


Outline Today



Outline Today

A Practical System



For Cloth Simulation

Intersection Elimination

Background – Implicit Simulation

Explicit methods calculate the state of a system at a later time from the state of the system at the current time, while implicit methods find a solution by solving an equation involving both the current state of the system and the later one.

- Implicit Euler: $x^{n+1} = x^n + hv^{n+1}, \quad v^{n+1} = v^n + hM^{-1}f^{n+1}$
- Increment: $\Delta x = h(v^n + \Delta v), \quad \Delta v = hM^{-1}(x^n + \Delta x, v^n + \Delta v)$
- 1-st order approximation:

$$\Delta v = hM^{-1}(f^n + \frac{\partial f}{\partial x}\Delta x + \frac{\partial f}{\partial v}\Delta v)$$

- Construct the linear system

$$\left(I - hM^{-1}\frac{\partial f}{\partial v} - h^2M^{-1}\frac{\partial f}{\partial x}\right)\Delta v = hM^{-1}(f^n + h\frac{\partial f}{\partial x}v^n)$$

$$\frac{\partial f}{\partial v} \rightarrow -\frac{\partial^2 E}{\partial v^2}$$

$$\frac{\partial f}{\partial x} \rightarrow -\frac{\partial^2 E}{\partial x^2}$$

Interior Point Method

Given the calculated next state $\mathbf{x}^n$, we want to update it into $\bar{\mathbf{x}}^{n+1}$, such that the path from $\mathbf{x}^n$ to $\bar{\mathbf{x}}^{n+1}$ is intersection-free.

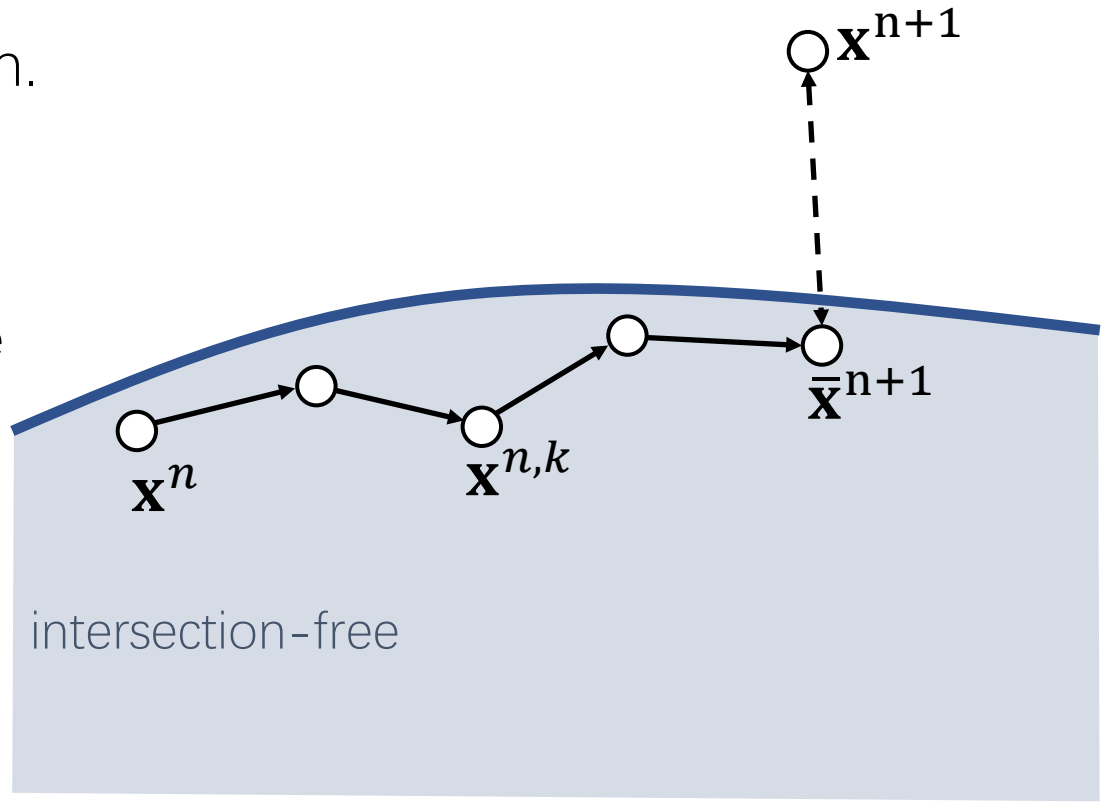
Interior point method reaches a best solution by traversing the interior of the feasible region.

- Do collision detection before assembling linear systems (option)
- Construct collision energies E_{VF}, E_{EE} for all possible collision pairs
- Calculate the Hessian matrix of the energy and assemble the linear system

$$\frac{\partial f}{\partial v} \mathrel{+}= -\frac{\partial^2 E_{VF}}{\partial v^2} - \frac{\partial^2 E_{EE}}{\partial v^2}$$

$$\frac{\partial f}{\partial x} \mathrel{+}= -\frac{\partial^2 E_{VF}}{\partial x^2} - \frac{\partial^2 E_{EE}}{\partial x^2}$$

$$f^n \mathrel{+}= -\frac{\partial E_{VF}}{\partial x} - \frac{\partial E_{EE}}{\partial x}$$



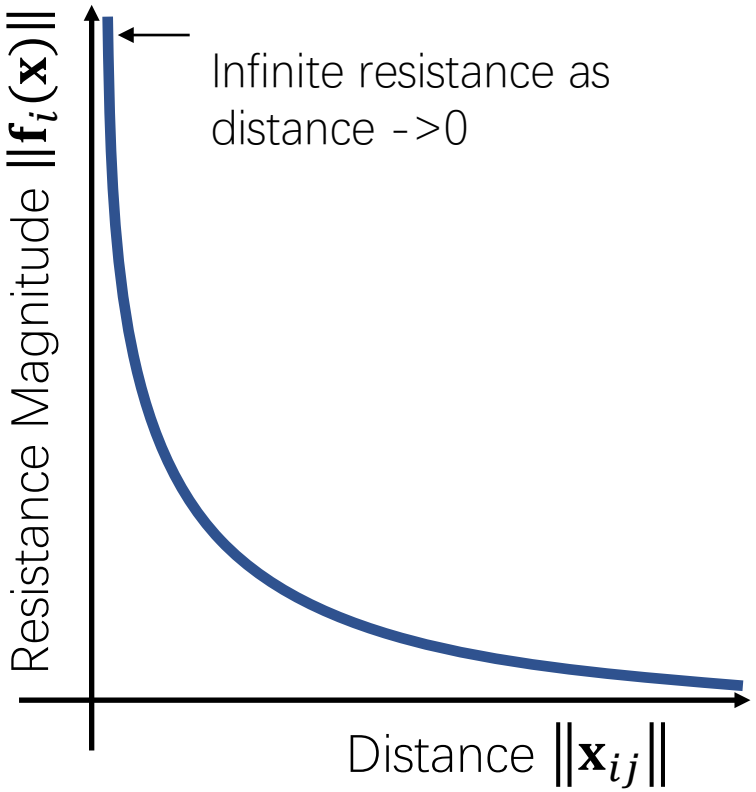
Log-Barrier Interior Point Method

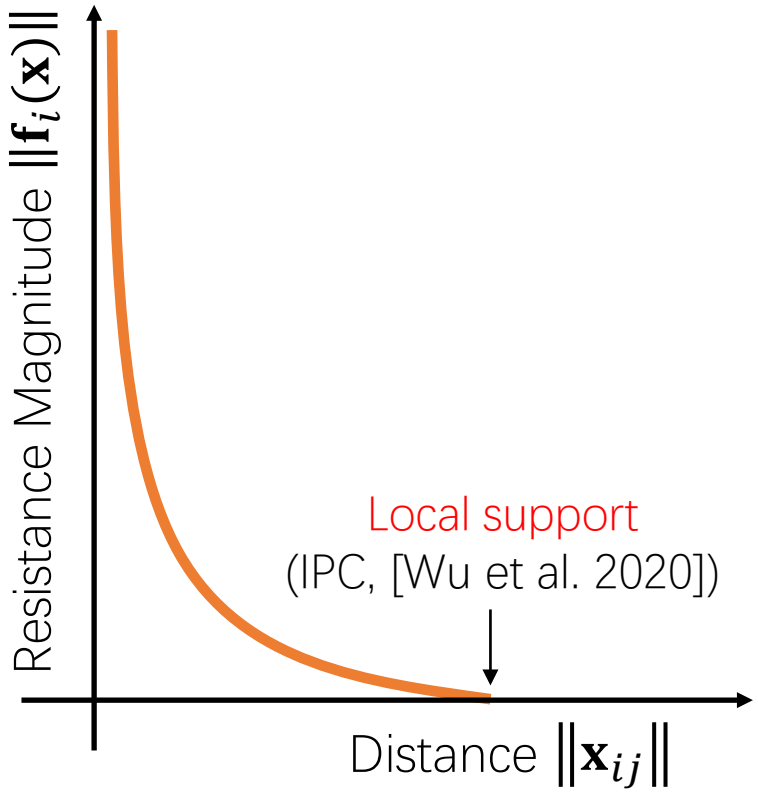
For simplicity, let's consider the Log-barrier repulsion between two vertices.

$$E(\mathbf{x}) = -\rho \log \|\mathbf{x}_{ij}\|$$

$$\mathbf{f}_i(\mathbf{x}) = -\nabla_i E = \rho \frac{\mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\|^2}$$
$$\mathbf{f}_j(\mathbf{x}) = -\nabla_j E = -\rho \frac{\mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\|^2}$$

$$E_{IPC}(\mathbf{x}) = \begin{cases} -(\|\mathbf{x}_{ij}\| - \hat{d}) \log \frac{\|\mathbf{x}_{ij}\|}{\hat{d}} \\ 0 \end{cases}$$





Log-Barrier Interior Point Method

We can then formulate the problem as:

$$\bar{\mathbf{x}}^{n+1} \leftarrow \operatorname{argmin}_{\mathbf{x}} \left(\frac{1}{2}\|\mathbf{x} - \boldsymbol{x}^{n+1}\|^2 - \rho \sum \log\|\mathbf{x}_{ij}\| \right)$$

$\mathbf{x}^{n+1}$

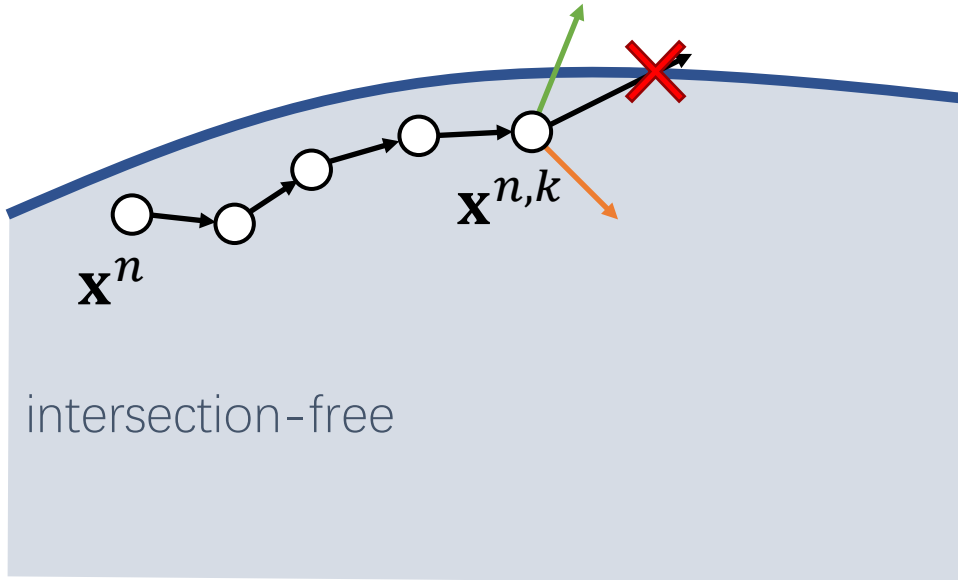
Gradient Descent:

$$\mathbf{x}^{n,0} \leftarrow \mathbf{x}^{n}$$

For $k = 0 \ldots K$

$$\mathbf{x}^{n,k+1} \leftarrow \mathbf{x}^{n,k} + \alpha \left(\boldsymbol{x}^{n+1} - \boldsymbol{x}^{n,k} + \rho \sum \frac{\mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\|^2} \right)$$

$$\bar{\mathbf{x}}^{n+1} \leftarrow \mathbf{x}^{n,k+1}$$



The step size α must be adjusted to ensure that no collision happens on the way. To find α, we need collision tests.

Background – Position Based Dynamics

Position based dynamics (PBD) is based on the projection function.

- The stiffness behavior, i.e., how tightly constraints are enforced, is subject to non-physical factors.
  - The number of iterations
  - The mesh resolution
- The velocity update following projection is important to dynamic effects.
- This method is applicable to other constraints as well, including triangle constraints, volume constraints, and collision constraints.
  - To implement these constraints, simply define their projection functions.

A PBD Simulator

//Do Simulation, update $\boldsymbol{x}$ and $\boldsymbol{v}$

$\boldsymbol{v}^{\mathbf{n+1}} \leftarrow \boldsymbol{\alpha}(\boldsymbol{v}^{\mathbf{n}} + hM^{-1}f_{ext})$

$\boldsymbol{x}^{\mathbf{n+1}} \leftarrow \boldsymbol{x}^{n} + \boldsymbol{v}^{n+1}$

//Now PBD starts.

$\boldsymbol{x}^{\mathbf{new}} \leftarrow \mathbf{Projection}(\boldsymbol{x}^{\mathbf{n+1}})$

$\boldsymbol{v}^{\mathbf{n+1}} \leftarrow \boldsymbol{v}^{\mathbf{n+1}} + (\boldsymbol{x}^{\mathbf{new}} - \boldsymbol{x}^{\mathbf{n+1}})/h$

$\boldsymbol{x}^{\mathbf{n+1}} \leftarrow \boldsymbol{x}^{\mathbf{new}}$

Impact Zone Optimization

Given the calculated next state $\mathbf{x}^n$, we want to update it into $\bar{\mathbf{x}}^{\mathbf{n+1}}$, such that the path from $\mathbf{x}^n$ to $\bar{\mathbf{x}}^{\mathbf{n+1}}$ is intersection-free.

The goal of impact zone optimization is to optimize $\mathbf{x}^{\mathbf{n+1}}$ until it becomes intersection-free.

A PBD Simulator

//Do Simulation, update $\boldsymbol{x}$ and $\boldsymbol{v}$

$\boldsymbol{v}^{\mathbf{n+1}} \leftarrow \boldsymbol{\alpha}(\boldsymbol{v}^{\mathbf{n}} + hM^{-1}f_{ext})$

$\boldsymbol{x}^{\mathbf{n+1}} \leftarrow \boldsymbol{x}^{n} + \boldsymbol{v}^{n+1}$

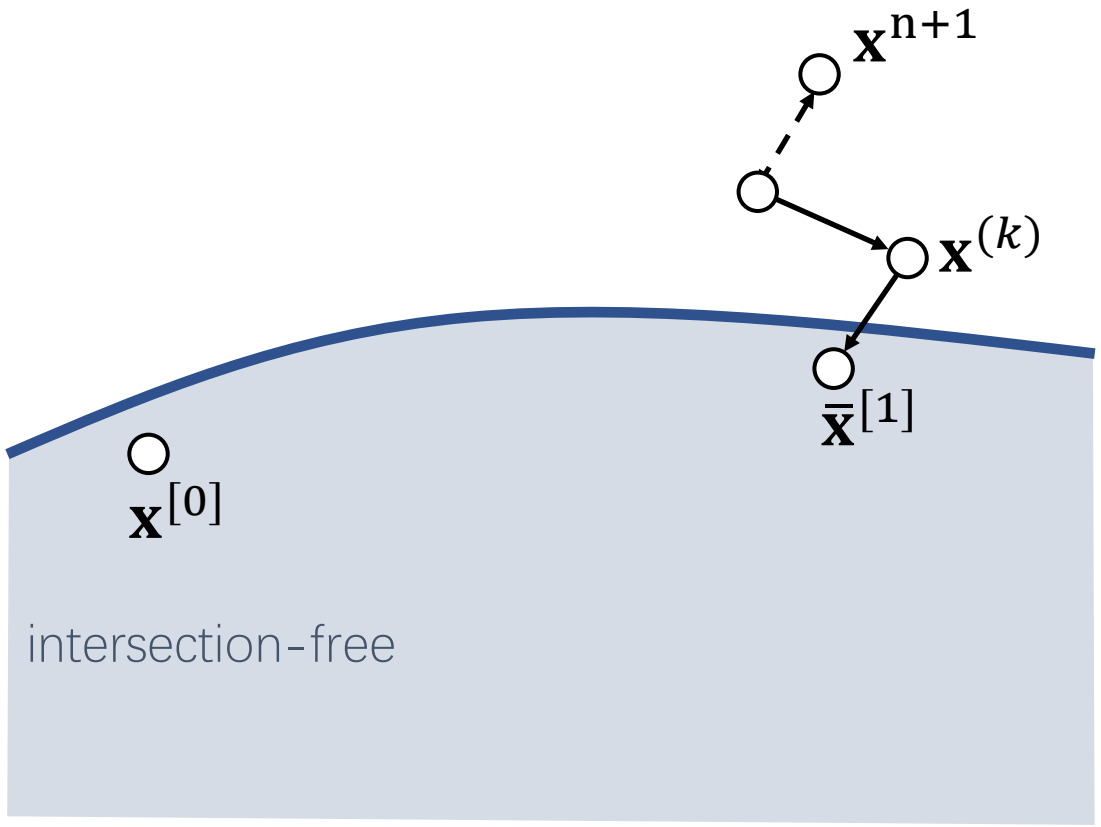
//Now PBD starts.

$\boldsymbol{x}^{\mathbf{new}} \leftarrow \mathbf{Projection}(\boldsymbol{x}^{\mathbf{n+1}})$

$\boldsymbol{v}^{\mathbf{n+1}} \leftarrow \boldsymbol{v}^{\mathbf{n+1}} + (\boldsymbol{x}^{\mathbf{new}} - \boldsymbol{x}^{\mathbf{n+1}})/h$

$\boldsymbol{x}^{\mathbf{n+1}} \leftarrow \boldsymbol{x}^{\mathbf{new}}$

Collision Detection

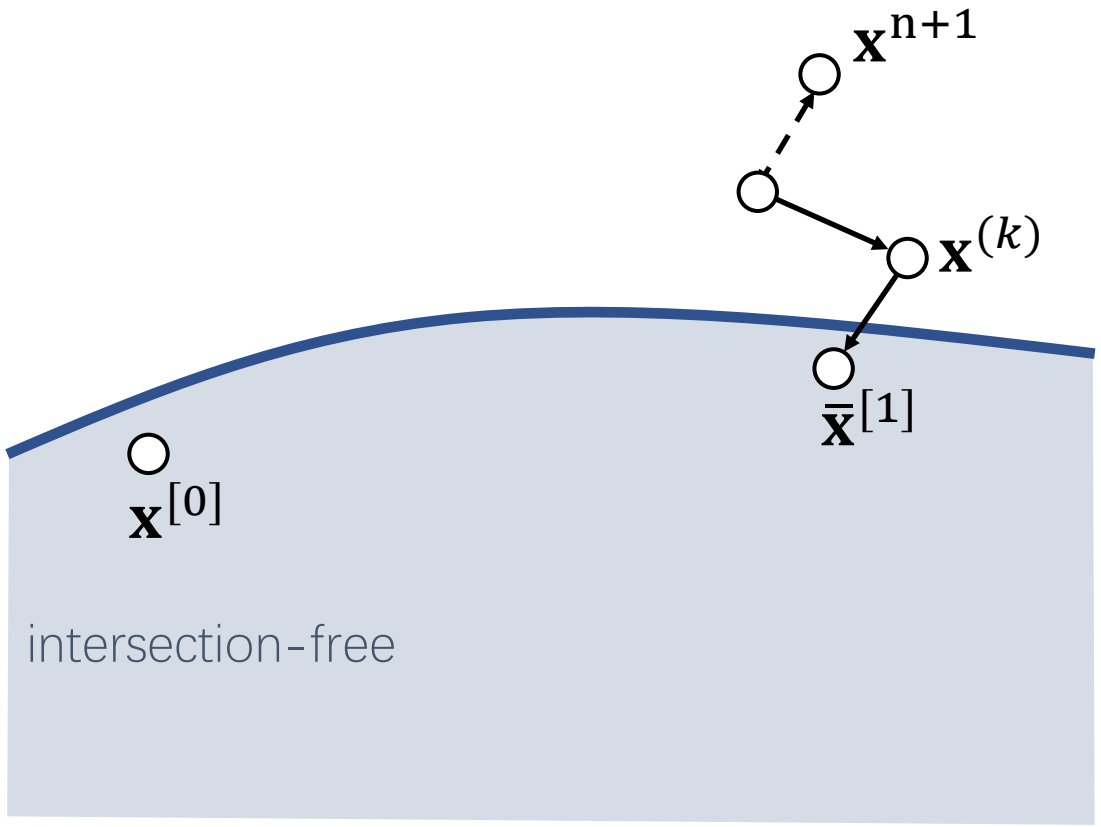


Impact Zone Optimization

Given the calculated next state $\mathbf{x}^n$, we want to update it into $\bar{\mathbf{x}}^{\mathbf{n+1}}$, such that the path from $\mathbf{x}^n$ to $\bar{\mathbf{x}}^{\mathbf{n+1}}$ is intersection-free.

The goal of impact zone optimization is to optimize $\mathbf{x}^{\mathbf{n+1}}$ until it becomes intersection-free.

- Vertex-face pair: vertex-face distance constraints
- Edge-edge pair: edge-edge distance constraints
- Special case: Capsule constraints, etc.



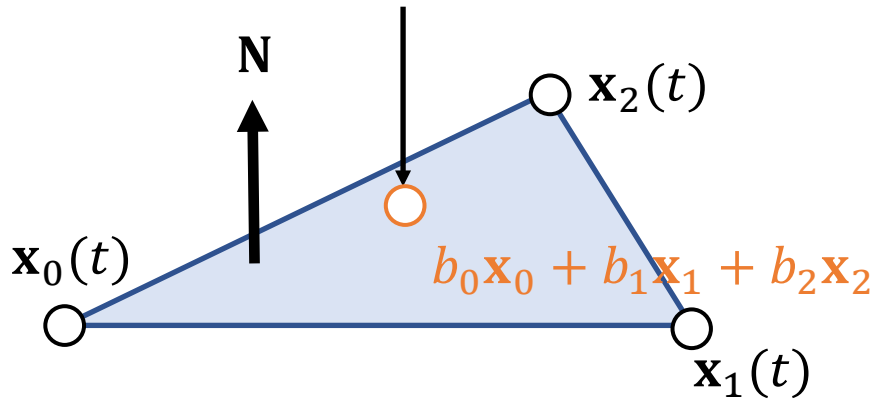
Impact Zone Optimization

The goal of impact zone optimization is to optimize $\mathbf{x}^{\mathbf{n+1}}$ until it becomes intersection-free.

$$C(\mathbf{x}) = (\mathbf{x}_3 - b_0\mathbf{x}_0 - b_1\mathbf{x}_1 - b_2\mathbf{x}_1) \cdot \mathbf{N} \geq 0$$

$$\frac{\partial C}{\partial \mathbf{x}_\mathbf{i}} = -b_i \boldsymbol{N}, \qquad i = 0,1,2$$

$$\frac{\partial C}{\partial \mathbf{x}_3} = \boldsymbol{N}$$

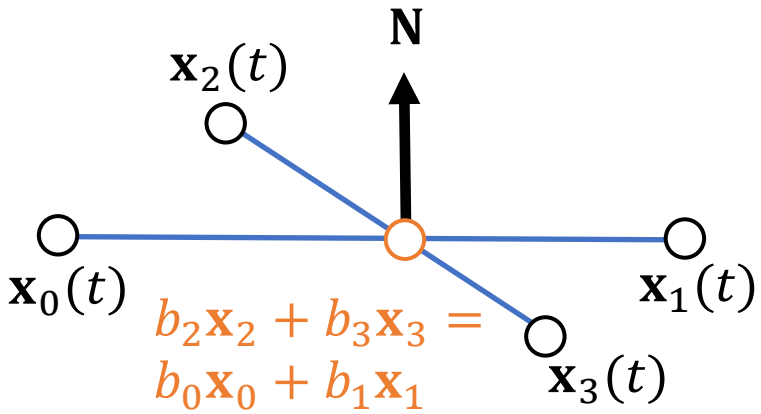


Vertex-Face Pair

$$C(\mathbf{x}) = (b_2\mathbf{x}_2 + b_3\mathbf{x}_3 - b_0\mathbf{x}_0 - b_1\mathbf{x}_1) \cdot \mathbf{N} \geq 0$$

$$\frac{\partial C}{\partial \mathbf{x}_\mathbf{i}} = -b_i \boldsymbol{N}, \qquad i = 0,1$$

$$\frac{\partial C}{\partial \mathbf{x}_\mathbf{j}} = b_j \boldsymbol{N}, \qquad j = 2,3$$



Edge-Edge Pair

Impact Zone Optimization

The goal of impact zone optimization is to optimize $\mathbf{x}^{\mathbf{n+1}}$ until it becomes intersection-free.

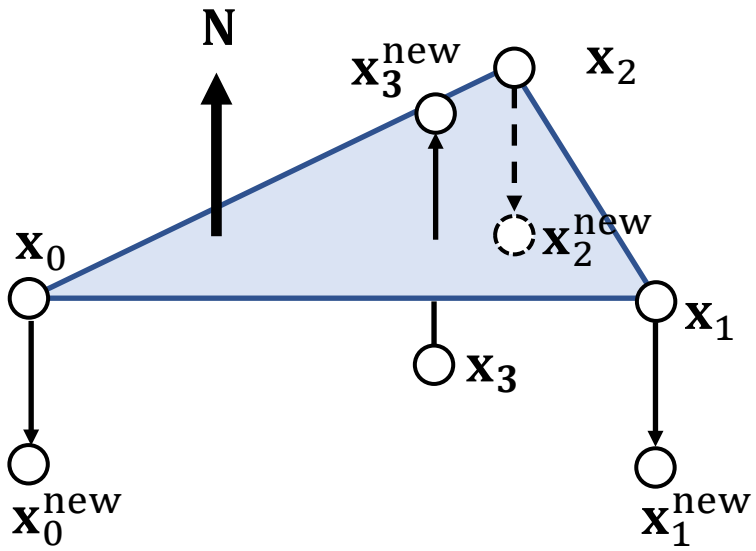
$$C(\boldsymbol{x} + \Delta\boldsymbol{x}) = 0$$

$$C(\boldsymbol{x}) + \nabla C(\boldsymbol{x})\Delta\boldsymbol{x} = 0$$

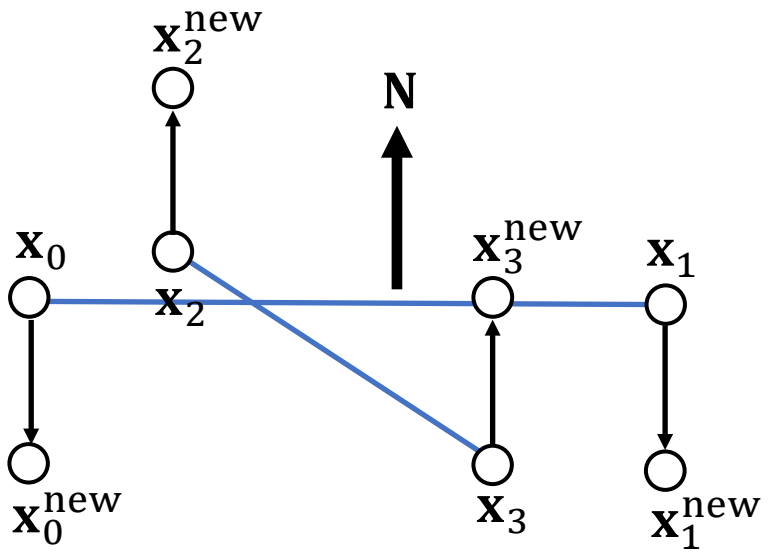
$$\Delta\boldsymbol{x} = \lambda\nabla C(\boldsymbol{x})$$

$$\lambda = -\frac{C(\boldsymbol{x})}{|\nabla C(\boldsymbol{x})|^2}$$

$$\Delta\boldsymbol{x}_i = \frac{\lambda}{m_i}\frac{\partial C}{\partial \boldsymbol{x}_i}$$



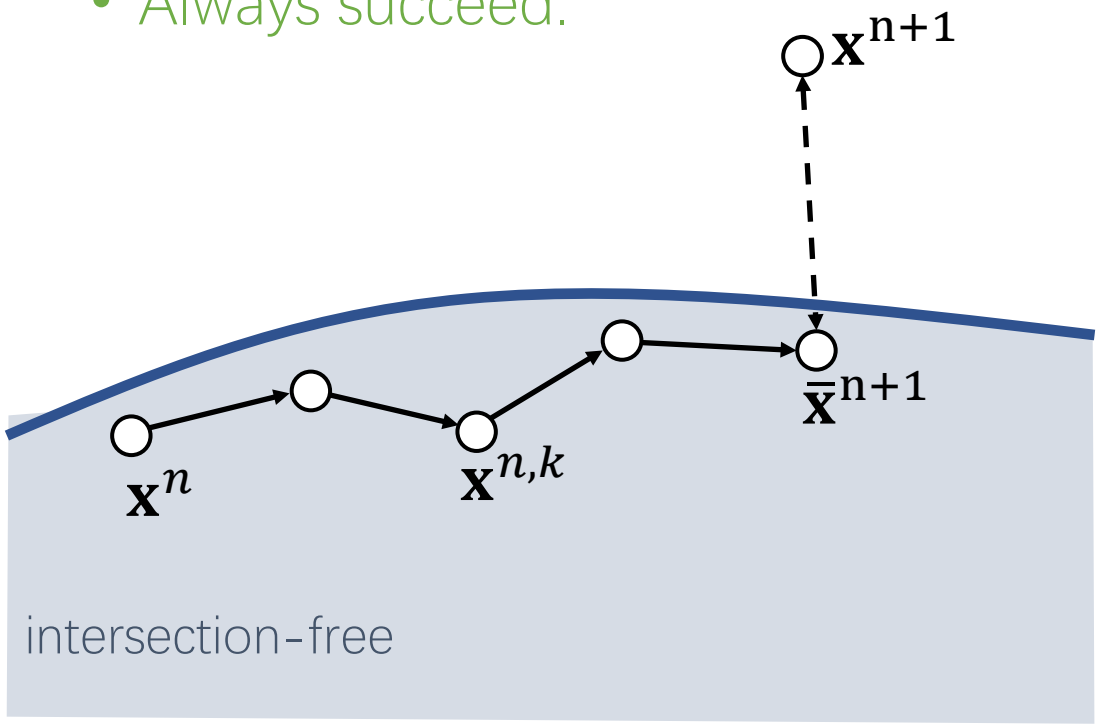
Vertex-Face Pair



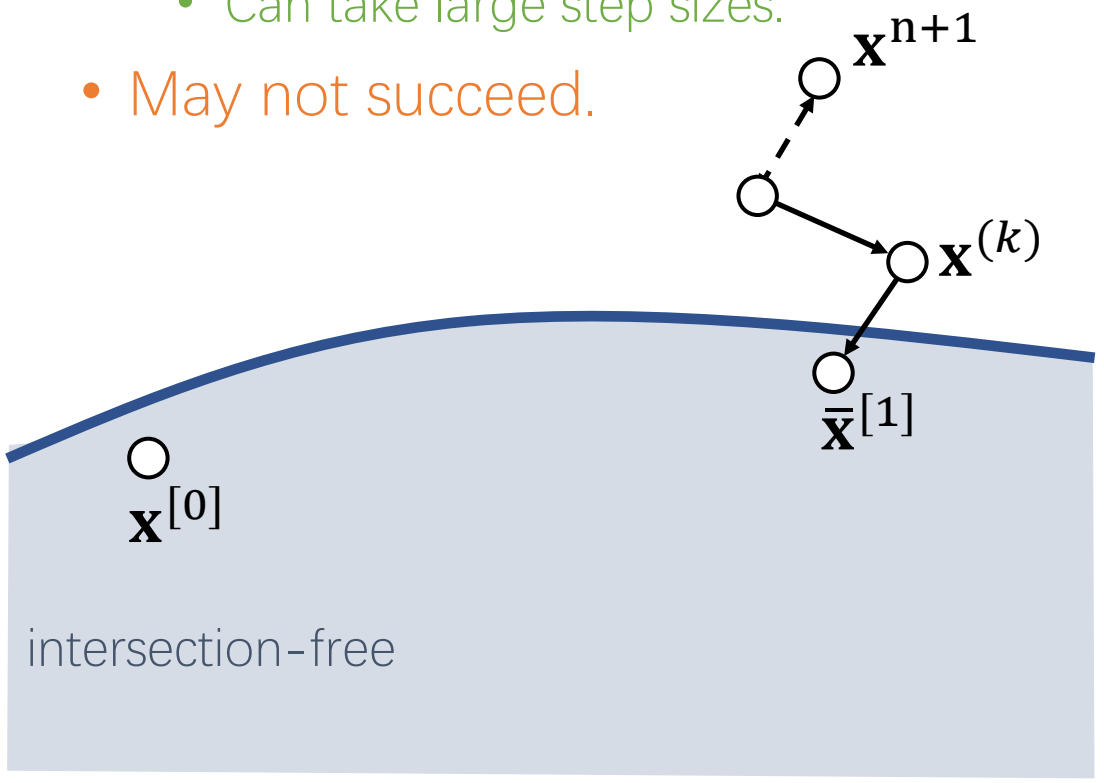
Edge-Edge Pair

Pros and Cons

- Slow.
  - Far from solution.
  - All of the vertices.
  - Cautiously by small step sizes.
- Always succeed.

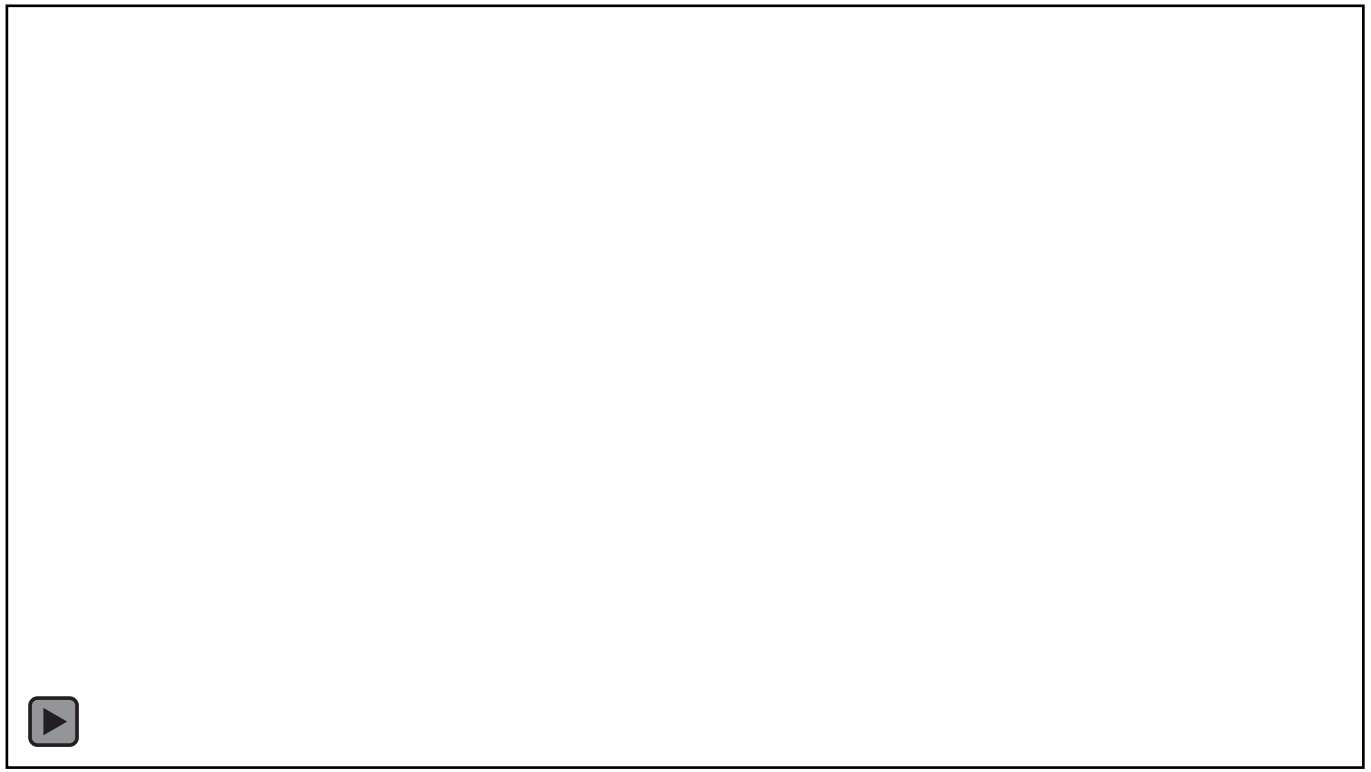


- Fast.
  - Close to solution.
  - Only vertices in collision (impact zones).
  - Can take large step sizes.
- May not succeed.

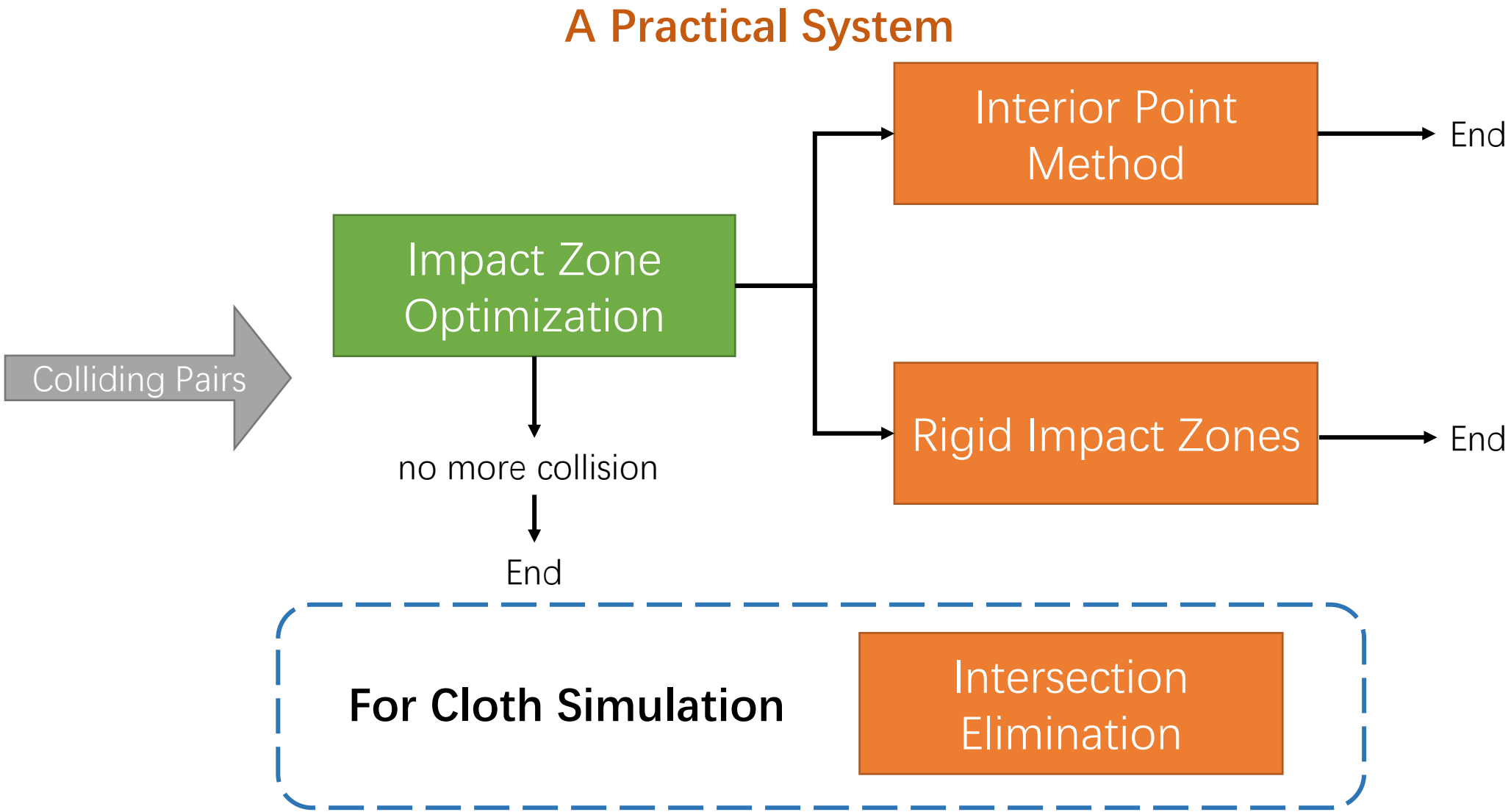


Rigid Impact Zones

The rigid impact zone method simply freezes vertices in collision from moving in their pre-collision state. It's simple and safe, but has noticeable artifacts.



Outline Today



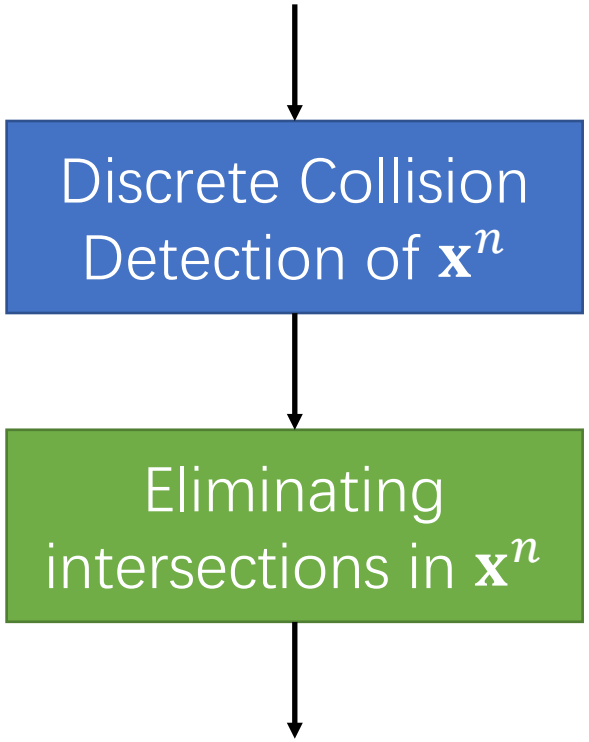
Intersection Elimination

Intersection Elimination

- Let's consider how to eliminate existing intersections, but without using any collision history.

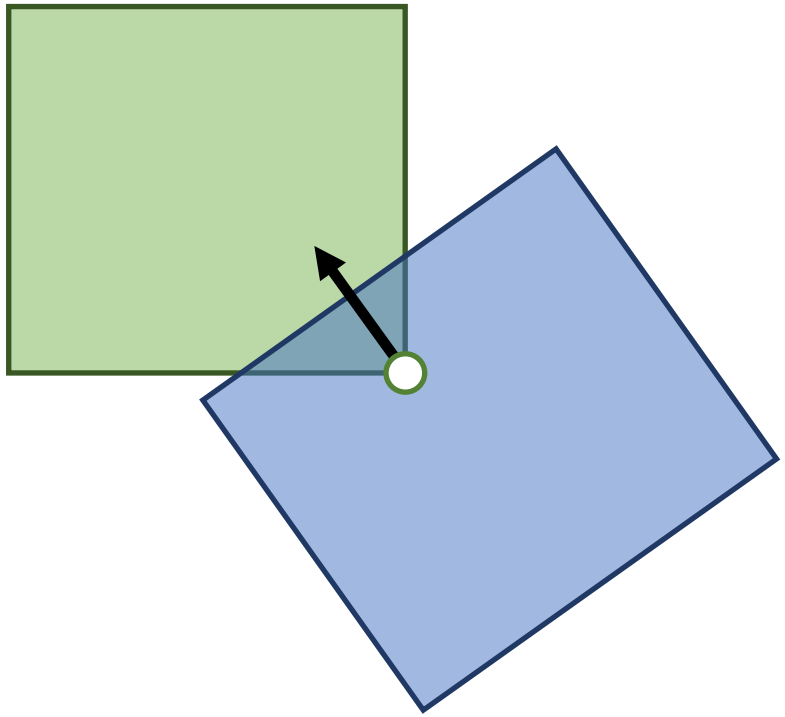
- Such a method is useful when there are already intersections in simulation, due to:
  - Past collision handling failures
  - Intense user interaction

- In this case, we don't require the simulation is to always intersection-free.

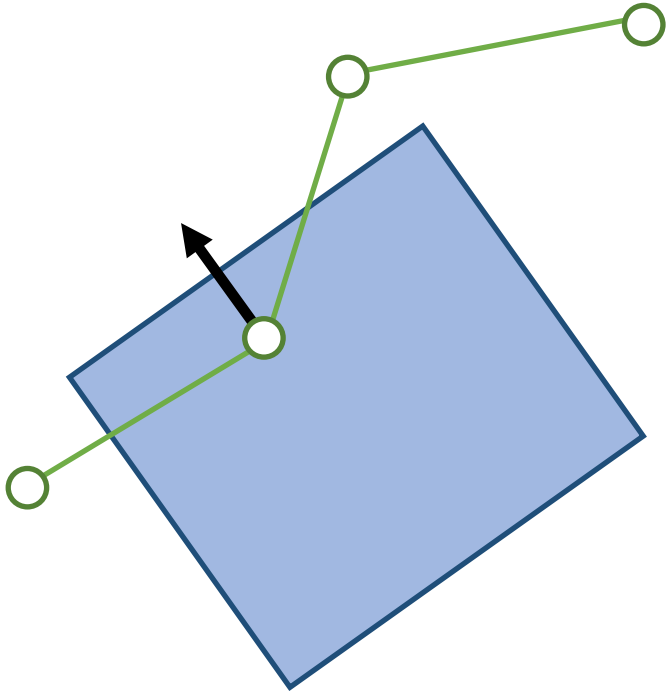


Intersection Elimination

Eliminating cloth-volume and volume-volume intersections is straightforward: simply pushing vertices/edges in the volume out.



Cloth-volume intersection

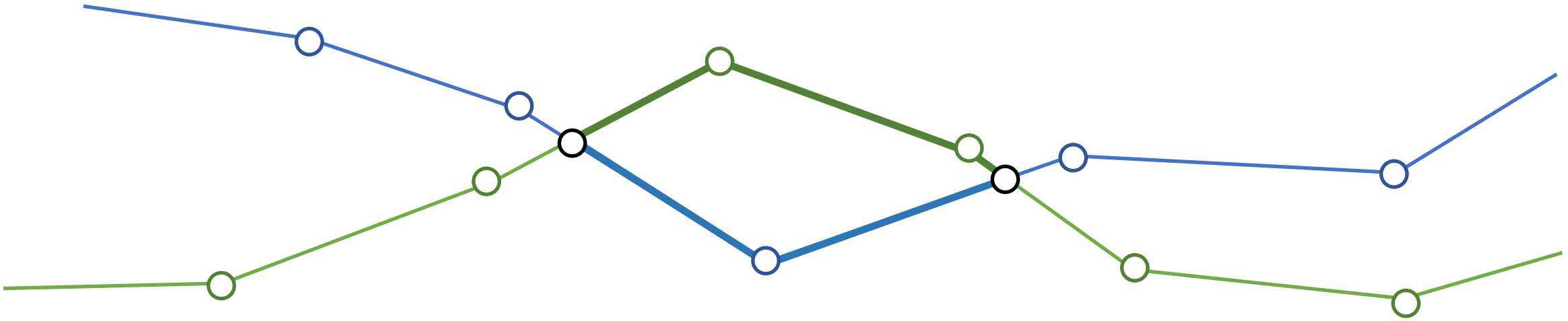


Volume-volume intersection

Intersection Elimination

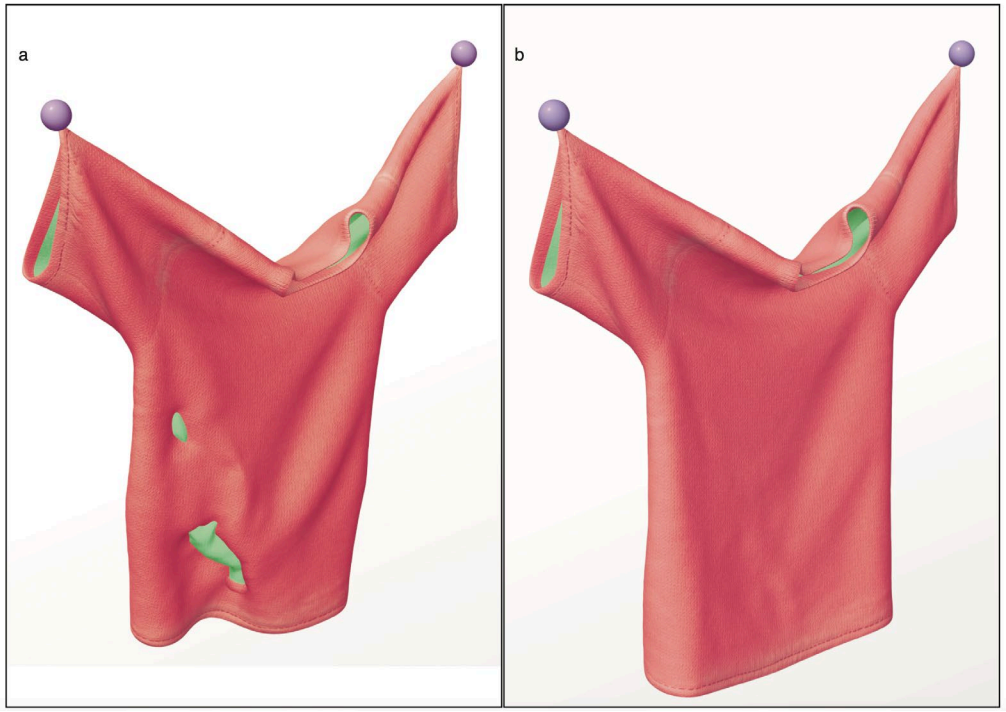
The situation is complicated in cloth-cloth intersection, since we don't have a clear definition of inside and outside.

Baraff et al. used flood-fill to segment cloth into regions and decided which region is in intersection. (Cannot handle boundary well.)



Intersection Elimination

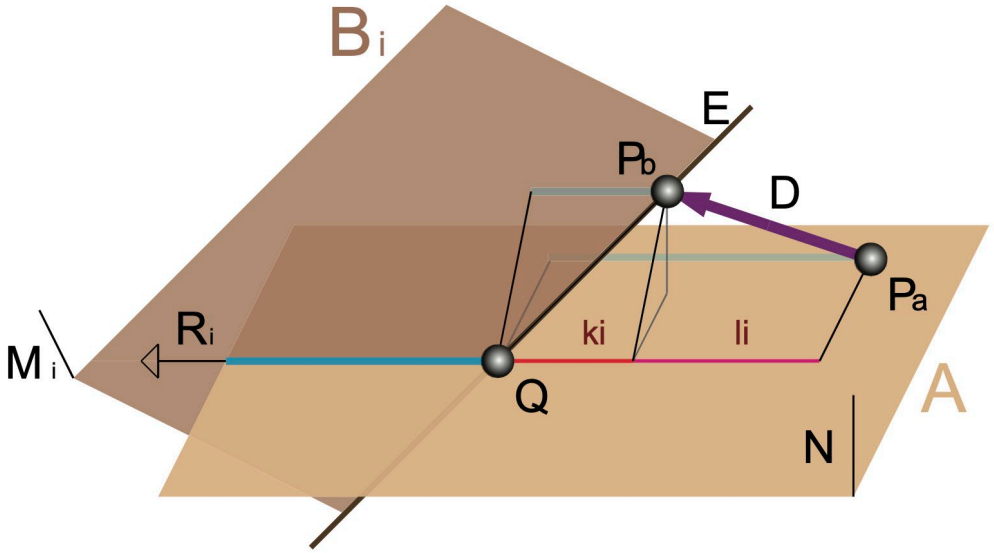
Baraff et al. used flood-fill to segment cloth into regions and decided which region is in intersection. (Cannot handle boundary well.)



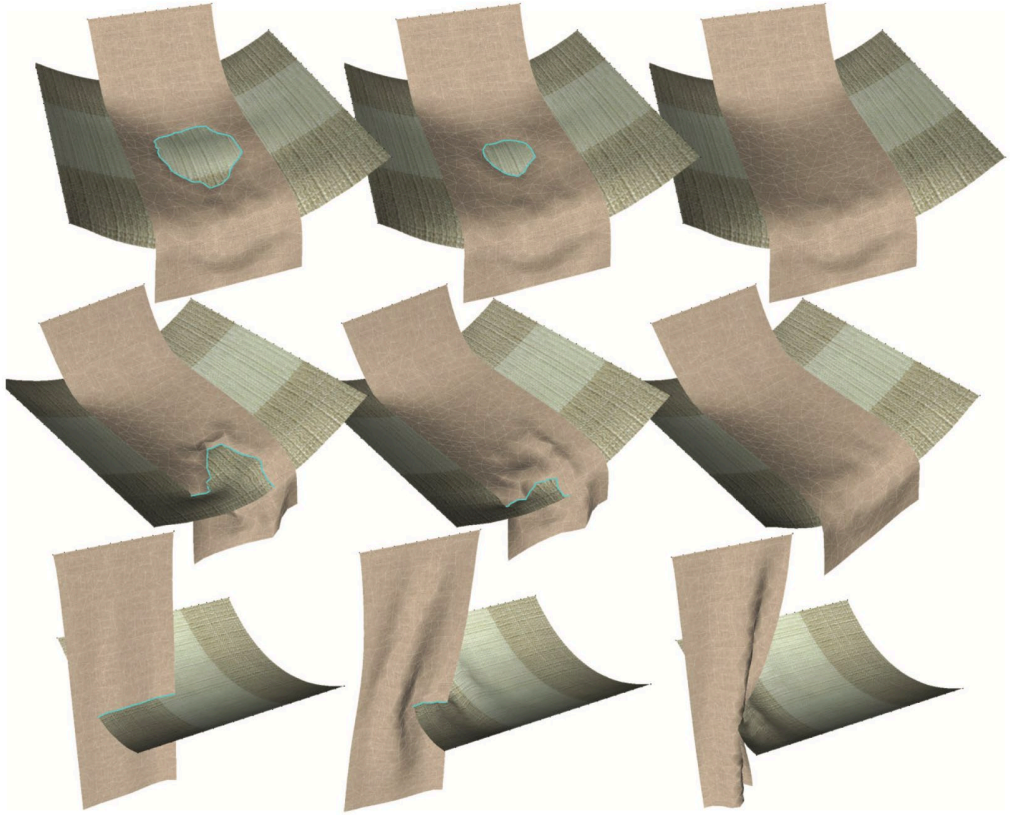
Intersection Elimination

Volino and Magnenat-Thalmann proposed to untangle cloth by reducing the intersection contour.

Their method can handle boundaries, but it doesn't always work.



$$G = \sum_i \left(R_i - 2\frac{E \cdot R_i}{E \cdot N} N \right)$$



Reference and Futher Readings

- GAMES103-基于物理的计算机动画入门 [Link]
- Incremental Potential Contact: Intersection- and Inversion-free Large Deformation Dynamics [Link]
- Position Based Dynamics [Link]
- Untangling Cloth [Link]
- Resolving surface collisions through intersection contour minimization [Link]

All relevant files (ppt, pdf, image, code, video):
https://github.com/LILKOTYO/Lab-Presentation/tree/master/20230522_Collision_Handling