Penetration-free Deformable Simulation on the GPU

Lei Lan[1,2], Guanqun Ma [1,2], Yin Yang [1,2,6], Changxi Zheng [3,4], Minchen Li [5,6], Chenfanfu Jiang [5,6]

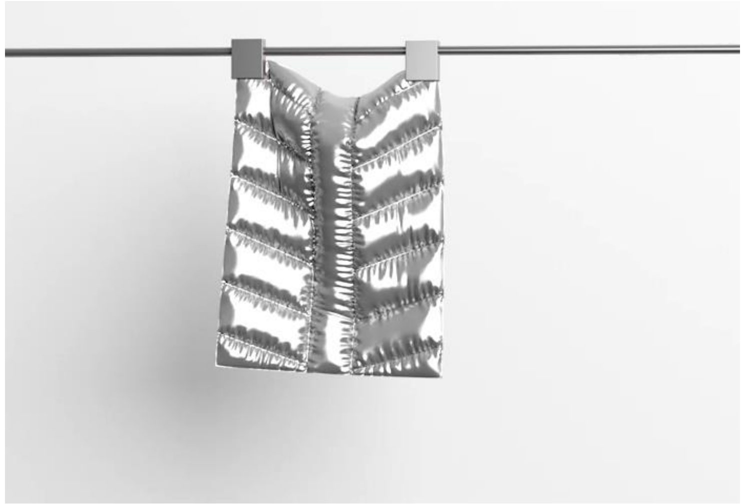
1 Clemson University 2 The University of Utah 3 Columbia University 4 Tencent Pixel Lab 5 UCLA 6 TimeStep Inc.

Deformable Simulation

[Li .et al 2019]

[Ruo .et al 2019]

[Smith .et al 2018]

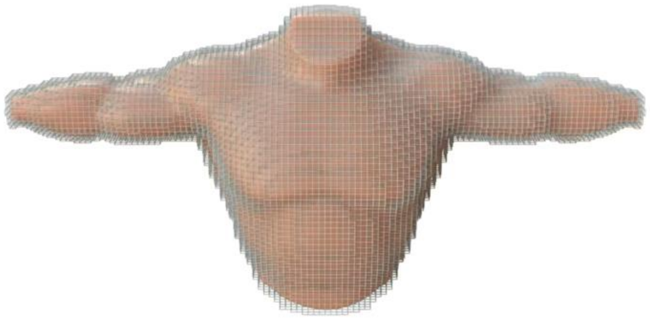


[Wang .et al 2021]

Deformable Simulation

❖ Equation of Motion:

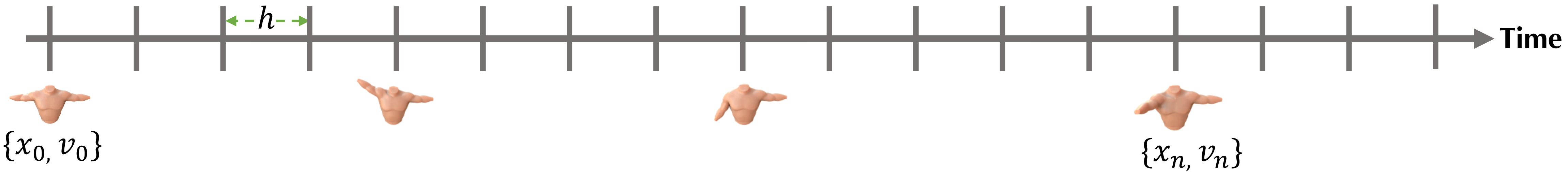
Continuum model



Spatial discretization

Sequence of poses

$$M\frac{\mathrm{d}^2x(t)}{\mathrm{d}^2t^2} = f_e + f_i(x(t))$$



Pictures and video come from [Smith .et al 2018].

Deformable Simulation

❖ Implicit Time Integration:

$$\begin{cases} v_{n+1} = v_n + hM^{-1}(f_i(x_{n+1}) + f_e) \\ x_{n+1} = x_n + hv_{n+1} \end{cases}$$

⬇

$$\begin{cases} x^*_{n+1} = x_n + hv_n + h^2 M^{-1} f_e \\ x_{n+1} = x^*_{n+1} + hM^{-1} f_i(x_{n+1}) \end{cases}$$

❖ Minimized Optimization:

$$\arg\min_x \; E(x) = \underbrace{\frac{1}{2}\|x - x^*\|^2_{\mathrm{M}}}_{\textit{inertial potential}} + \underbrace{h^2\Psi(x)}_{\textit{elastic potential}}$$

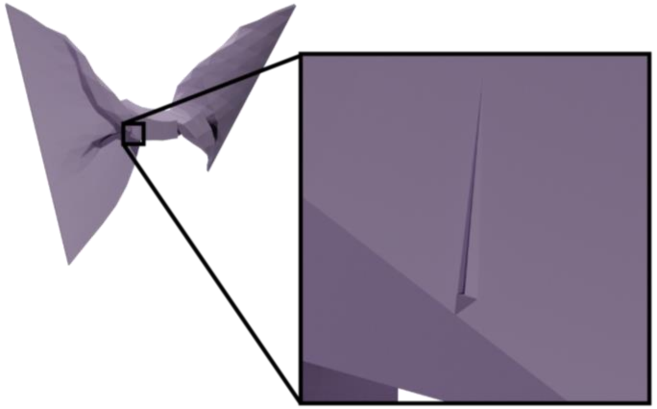
 # Solving a large nonlinear system is extremely slow.

- Reduced Simulation
- Position Based Dynamic (PBD)
- Projective Dynamic (PD)
- Multi-core CPUs/GPU
- Deep Learning
- …

Contact

❖ Collisions introduce extra difficulties to system solver

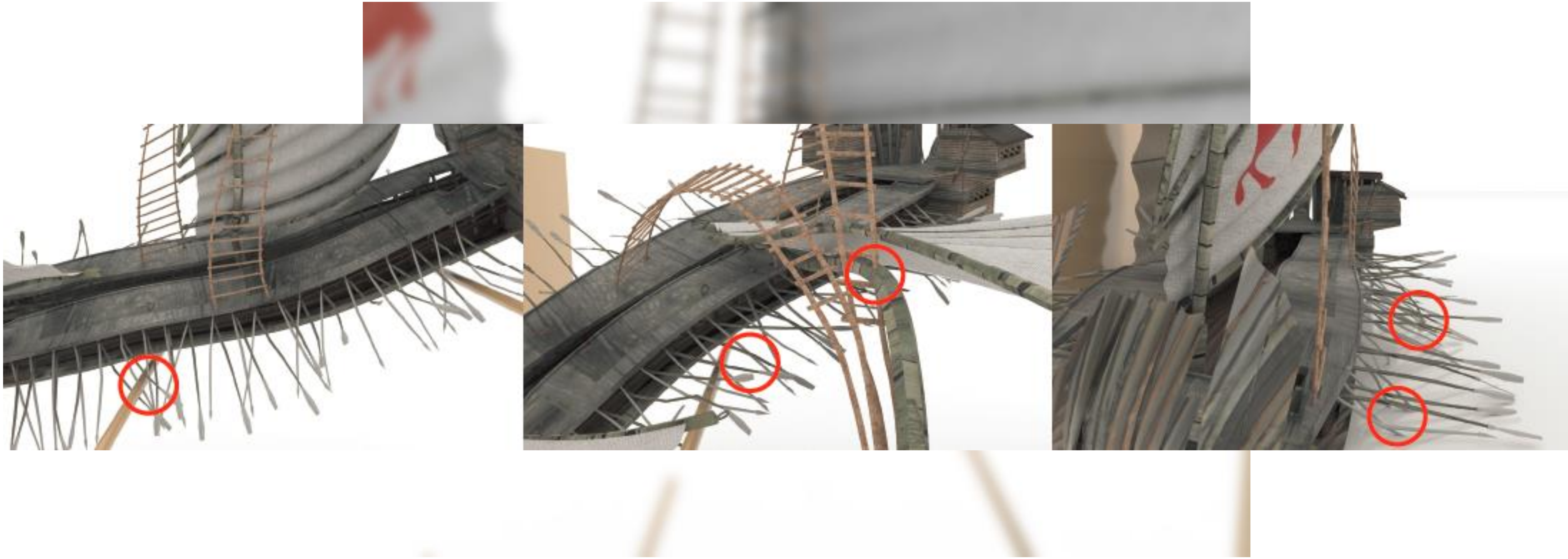
- A lot of intersection tests (DCD or CCD)
- Resolving using penalty force/impulse
  - Artifacts and instability
  - Non-guaranteed
- Resolving using Inequality constraints
  - Convert to LCP



Contact

DCD + Penalty force   dt = 0.02s

Contact



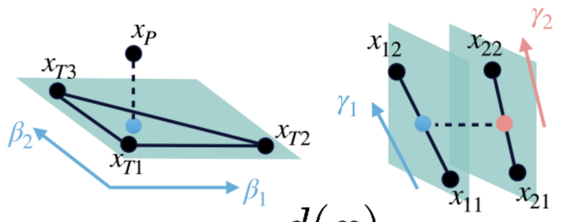
DCD + Penalty force   dt = 0.02s

Contact

❖ Incremental Potential Contact (IPC)

$$\arg\min_{x} \; E(x) = \frac{1}{2}\|x - x^*\|_{\mathrm{M}}^2 + h^2\Psi(x) + \kappa \sum_{\mathrm{k}\in C} b(d_{\mathrm{k}}(x))$$

$$\text{s.t.} \quad d(x) \geq 0$$

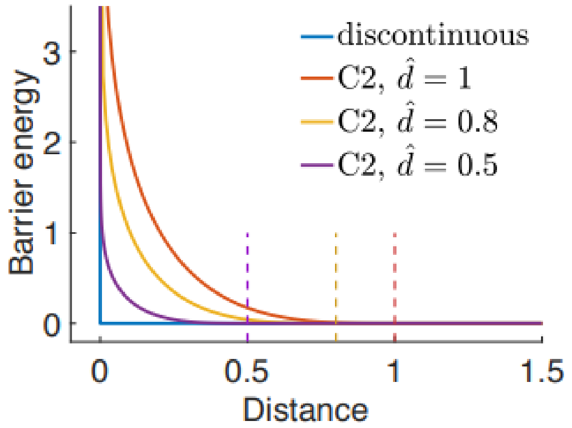


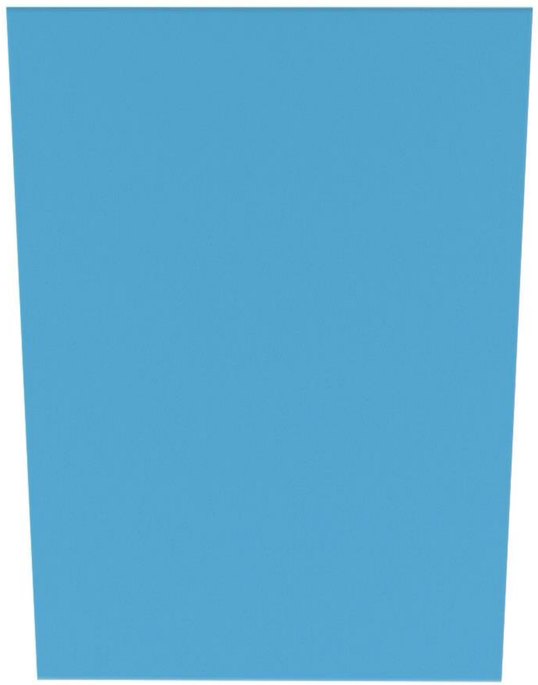
$d(x)$

- Barrier function

$$b(d_k(x)) = \begin{cases} -(d-\hat{d})^2 \ln(\frac{d}{\hat{d}}), & 0 < d < \hat{d} \\ 0, & d \geq \hat{d} \end{cases}$$

  - Unconstrained optimization problem
  - Impose a large internal force
  - Smooth





Projective Dynamic

❖ Projective Dynamics (PD)

- Fast simulation method

$$\arg\min_{x} \; E(x) = \frac{1}{2}\|x - x^*\|_{\mathrm{M}}^2 + h^2 \sum W(x)$$

quadratic optimization

local step

$$\arg\min_{y_i} \; \frac{\omega_i}{2}\|A_i S_i x - B_i y_i\|_{\mathrm{F}}^2 , \quad \text{s.t.} \quad C_i(y_i) = 0$$

global step

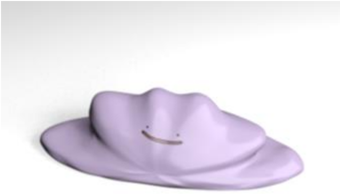
$$\left(\frac{M}{h^2} + \sum_i \omega_i S_i^{\mathrm{T}} A_i^{\mathrm{T}} A_i S_i\right) x = \frac{M}{h^2} x^* + \sum_i \omega_i S_i^{\mathrm{T}} A_i^{\mathrm{T}} B_i y_i$$

- Local-global iterative solver
- Local step is parallel friendly
- Global step is a (fixed) linear system
- DCD-based collision constraint

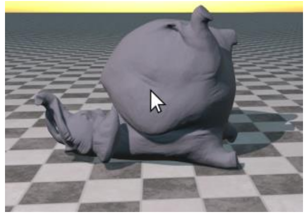
[Bouaziz .et al 2014]

Chebyshev method
[Wang .et al 2015]

Parallel Gauss-Seidel
[Fratarcangeli .et al 2015]



Quasi Newton Solver
[Liu .et al 2017]



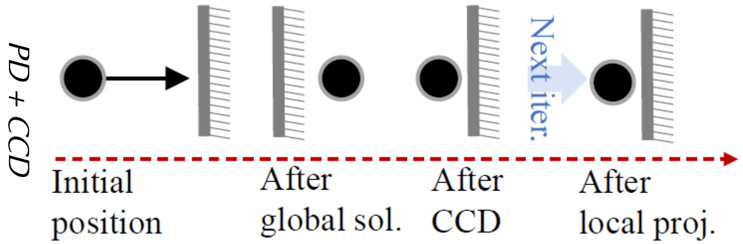
Reduced PD
[Brandt .et al 2018]

Semi-Reduced PD
[Lan .et al 2020]

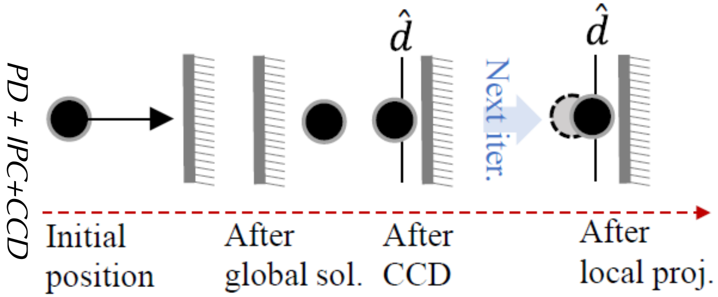
Our Method

❖ How to offer the non-intersection in PD?

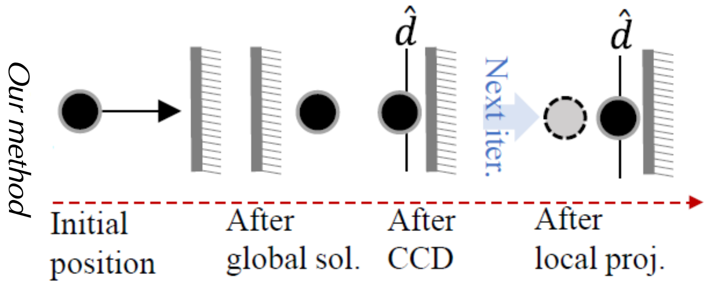
- PD + CCD, or PD + IPC+CCD?



- CCD pruning
- Target is current position



- CCD pruning
- Target is $\hat{d}$-away position



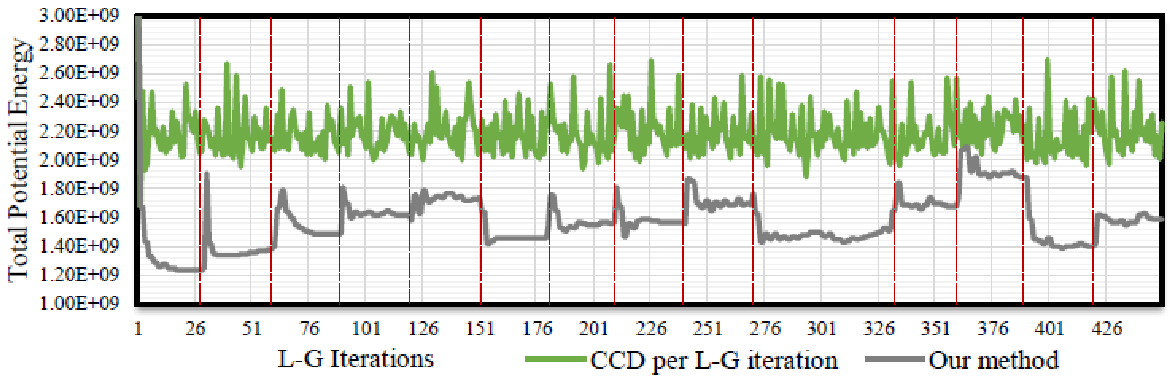
- CCD pruning
- $x + t\dot{x} + (1-t)(\mathrm{I} - (1+\epsilon)nn^{\mathrm{T}})\dot{x}$

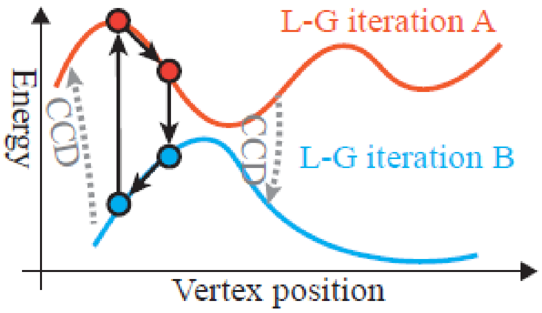
Rebounded

Our method

❖ Projective IPC with Tow-level Iteration Strategy

- Energy oscillating appear in single-level iteration





ALGORITHM 1: Projective IPC solver.

$z \leftarrow x^* + h\dot{x}^* + h^2 \mathrm{M}^{-1} f_{ext}$;
$\tilde{x} \leftarrow x^* + h\dot{x}^* + \frac{h^2}{4}\ddot{x}^*$; // $\tilde{x}$ now is a predicted position
$x \leftarrow x^*$, $\Delta x \leftarrow \tilde{x} - x$; ***# Required many outer iterations.***
while $\|\Delta x\|^2 > \varepsilon_{outer}$ **do**
  $B \leftarrow$ BarrierProjection(x); // barrier projection (§ 4.1)
  $\delta E \leftarrow +\infty$; // δE is per-iteration potential change rate
  while $\delta E > \varepsilon_{inner}$ **do**
    $E \leftarrow \frac{1}{2h^2}\|\mathrm{M}^{-1}(\tilde{x} - z)\|_F^2$; // update momentum potential
    $\Psi \leftarrow$ ElasticProjection$(\tilde{x})$;
    $\tilde{x} \leftarrow$ GlobalSolve;
    update δE;
  end
  CollisionCulling$(\tilde{x})$; // patch-based GPU culling (§ 7.2)
  $t_I \leftarrow$ CCD$(x, \tilde{x})$; // minimum-gradient Newton method (§ 6)
  $\tilde{x} \leftarrow x + \frac{t_I}{h} \cdot (\tilde{x} - x)$; // per outer loop CCD pruning (§ 4.2)
  $\Delta x \leftarrow \tilde{x} - x$, $x \leftarrow \tilde{x}$; // update x and Δx
end
$\dot{x} \leftarrow \frac{x - x^*}{h}$, $\ddot{x} \leftarrow \frac{\dot{x} - \dot{x}^*}{h}$; // velocity and acceleration update

Aggregated Jacobi Solver

$$\underbrace{\left(\frac{M}{h^2} + \sum_i \omega_i S_i^{\mathrm{T}} A_i^{\mathrm{T}} A_i S_i\right)}_{\mathrm{D}-\mathrm{B}} x = \underbrace{\frac{M}{h^2} x^* + \sum_i \omega_i S_i^{\mathrm{T}} A_i^{\mathrm{T}} B_i y_i}_{b}$$

❖ Jacobi Solver

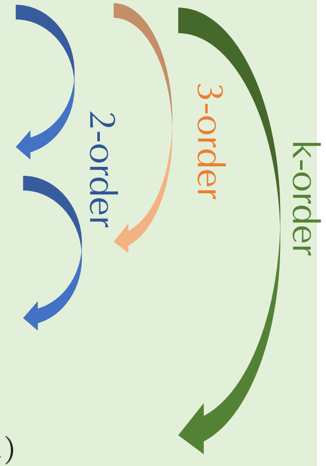
$$x^{(k)} = \mathrm{D}^{-1} b + \mathrm{R} x^{(k-1)},\ \mathrm{R} = \mathrm{D}^{-1}\mathrm{B}$$

- Slow convergence
- Per-thread computation is too light
- Waste the overhead of GPU scheduling

 # Better harvests the capacity of GPUs

 # Speed up convergence

$$
\begin{aligned}
&x^{(0)},\\
&x^{(1)} = \mathrm{D}^{-1} b + \mathrm{R} x^{(0)},\\
&x^{(2)} = \mathrm{D}^{-1} b + \mathrm{R} x^{(1)},\\
&x^{(3)} = \mathrm{D}^{-1} b + \mathrm{R} x^{(2)},\\
&x^{(4)} = \mathrm{D}^{-1} b + \mathrm{R} x^{(3)},\\
&\qquad \ldots,\\
&x^{(k)} = \mathrm{D}^{-1} b + \mathrm{R} x^{(k-1)},
\end{aligned}
$$

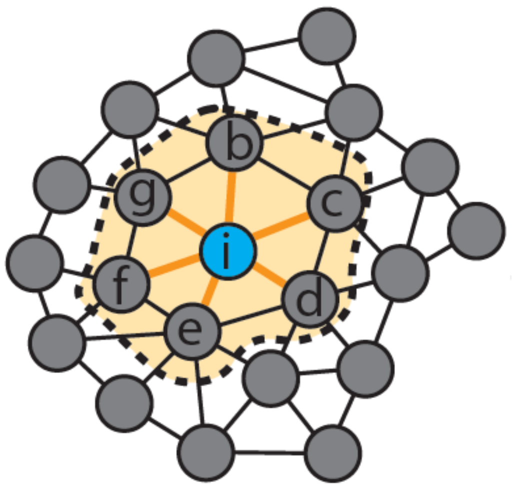


Aggregated Jacobi Solver

❖ A-Jacobi Solver

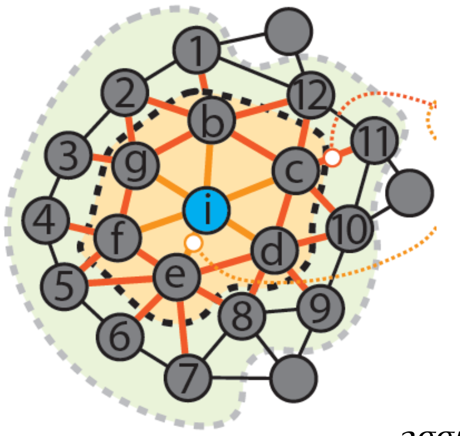
$$x^{(k)} = \sum_{j=0}^{l-1} \mathrm{R}^j \mathrm{D}^{-1} b + \mathrm{R}^l x^{(k-1)}$$

- Compatible with Chebyshev
- Sparsity suppression
- Flexible



$[\mathrm{R}x]_i = \mathrm{D}_{ii}^{-1}\mathrm{B}_{ij}x_j$

traverse 1-ring neighbors

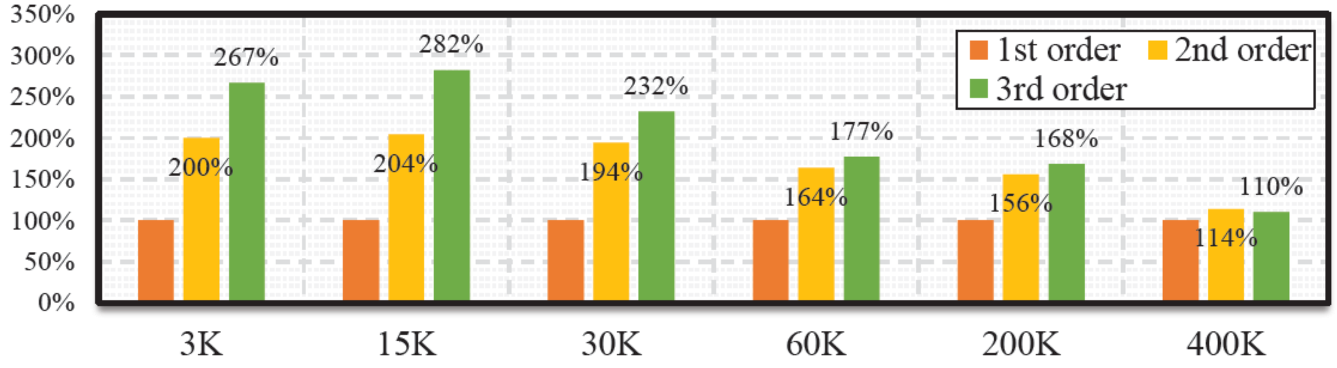


aggregated product

$[\mathrm{R}^2x]_i = \mathrm{D}_{ii}^{-1}\mathrm{D}_{ss}^{-1}\mathrm{B}_{is}\mathrm{B}_{sj}x_j$

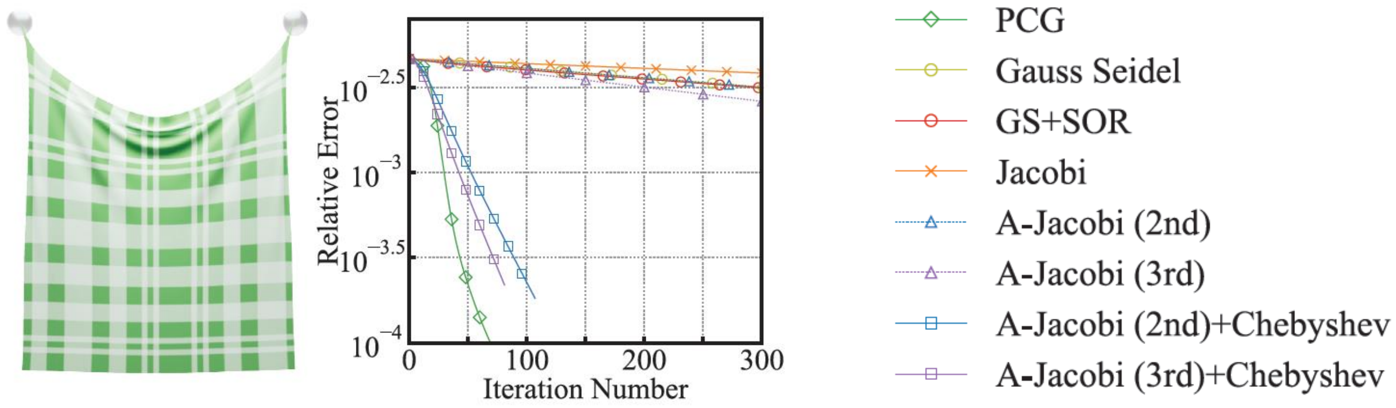
traverse 2-ring neighbors

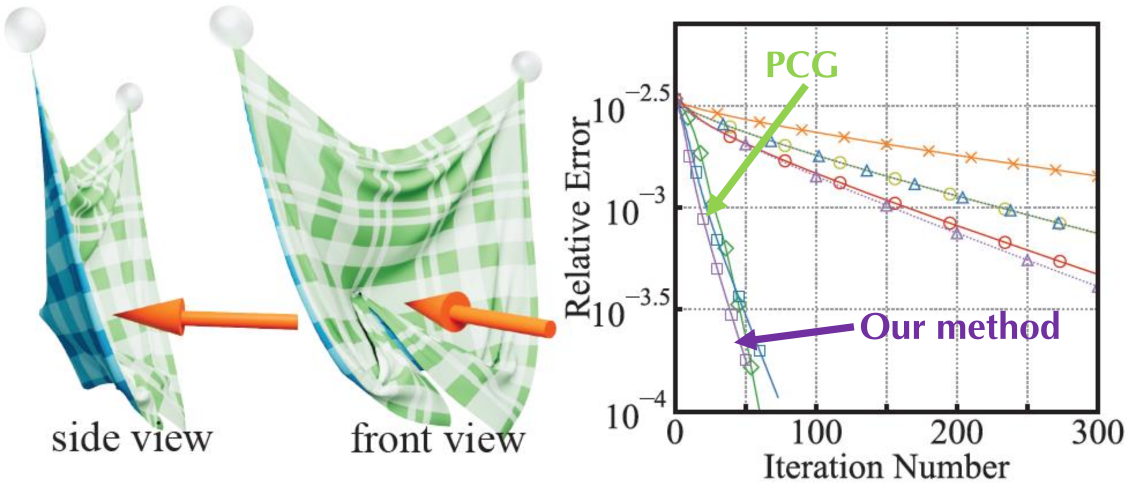
- A-Jacobi Performance



Aggregated Jacobi Solver

- A-Jacobi Convergence



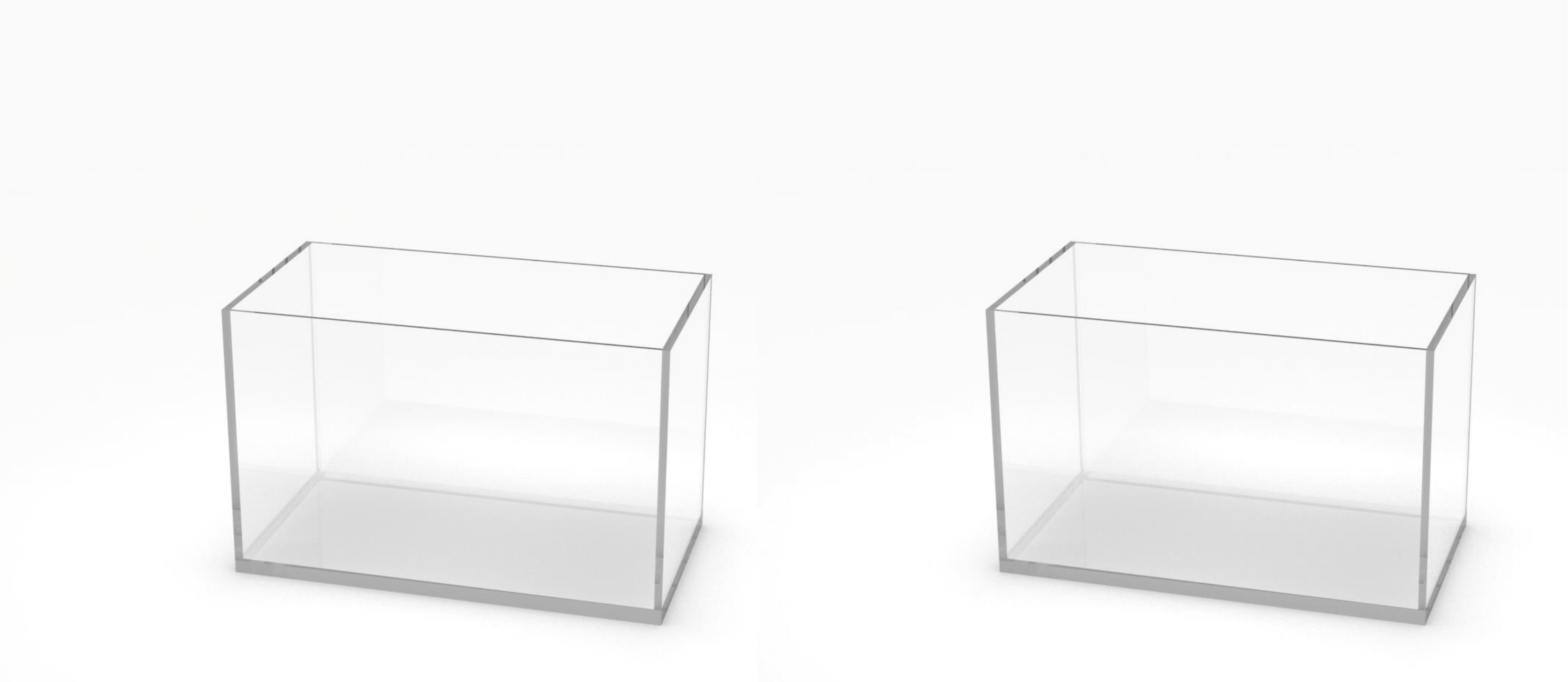


Collision Detection

❖ Patch-based GPU Collision Culling

- Binary AABB tree
- Each leaf is a patch of geometry

Comparison to IPC: Rubber Helicopters



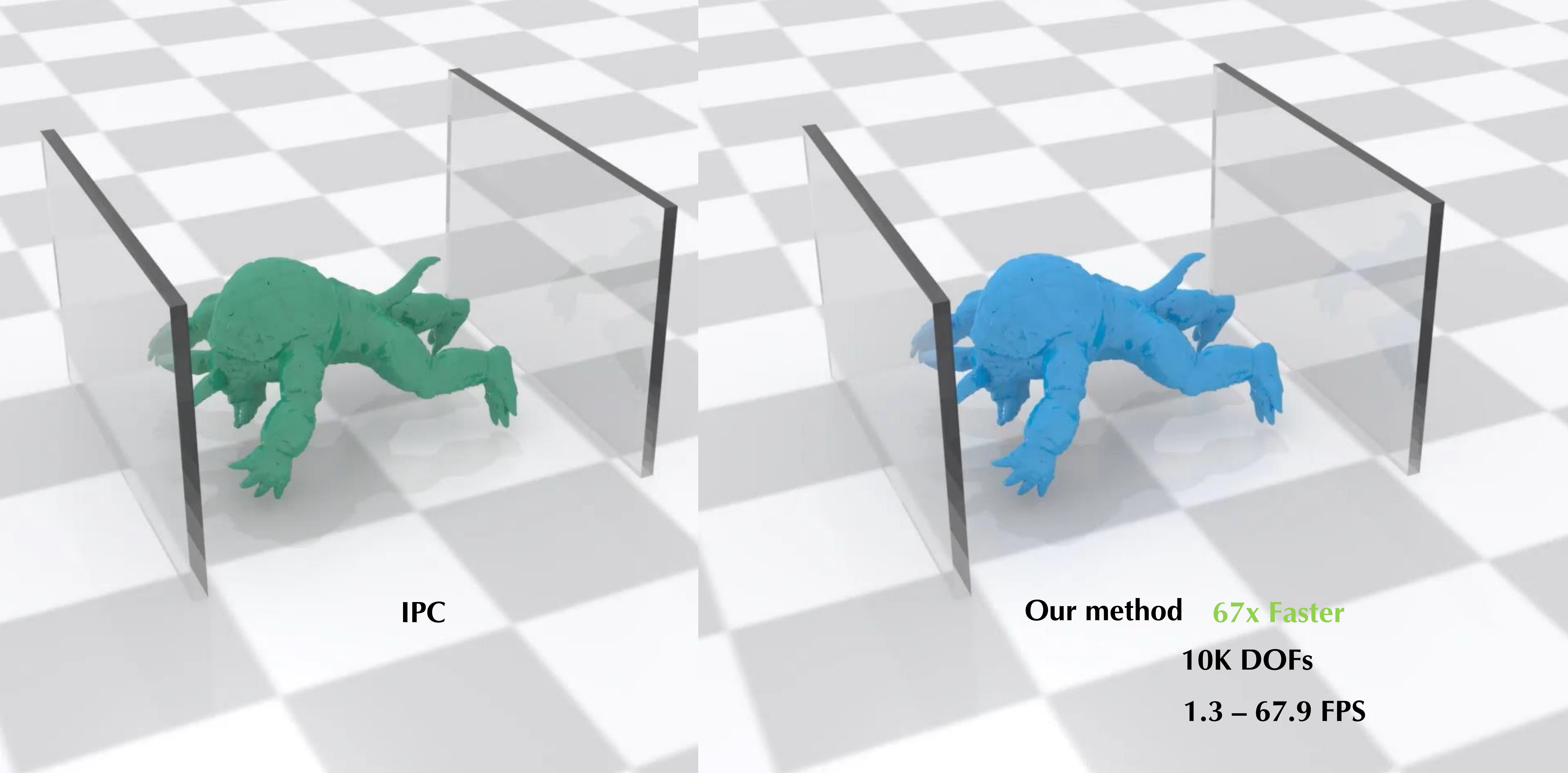
IPC

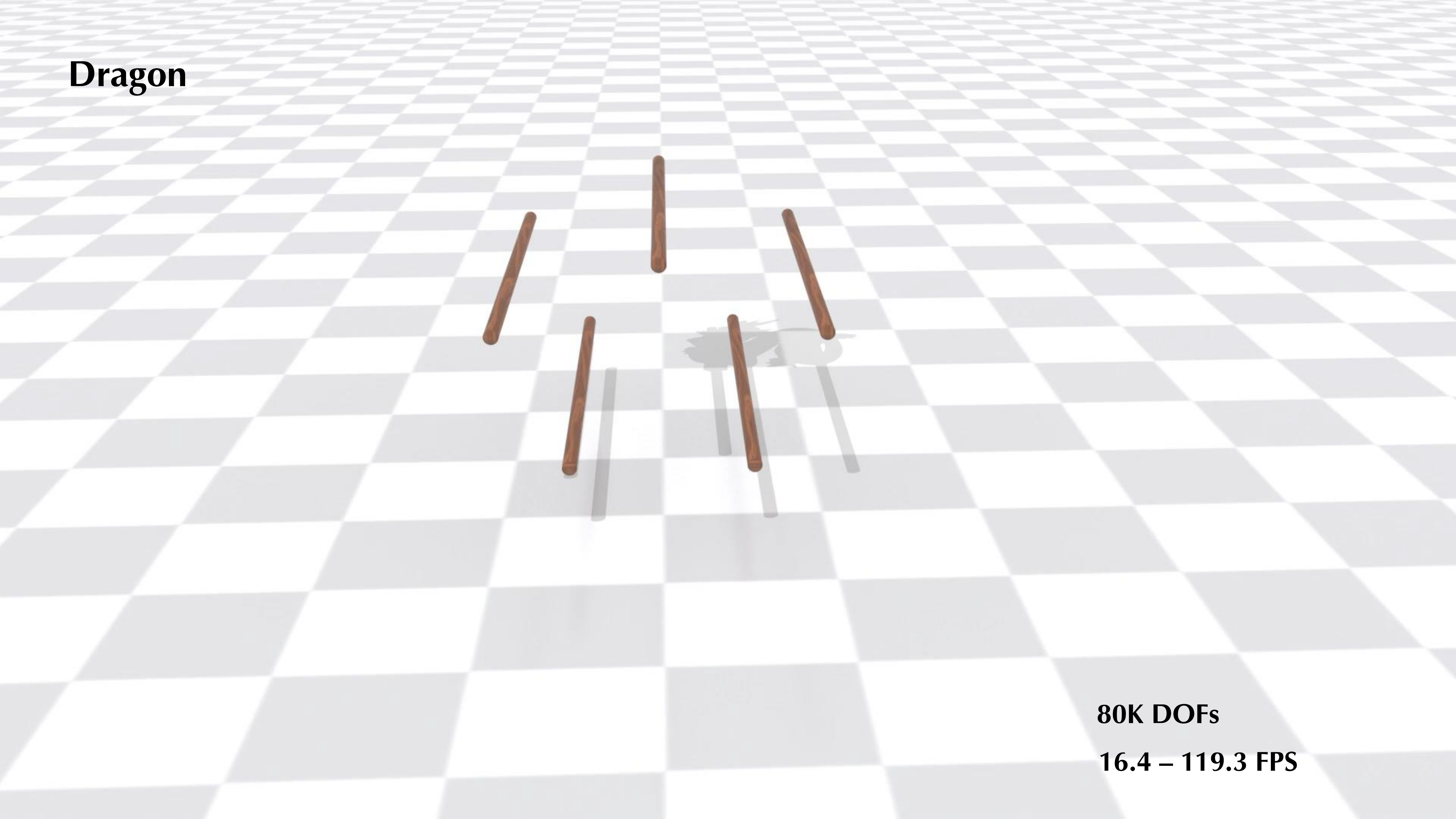
Our method 2000x Faster

150K DOFs

11.2 – 87.1 FPS

Comparison to IPC: Flatten Armadillo



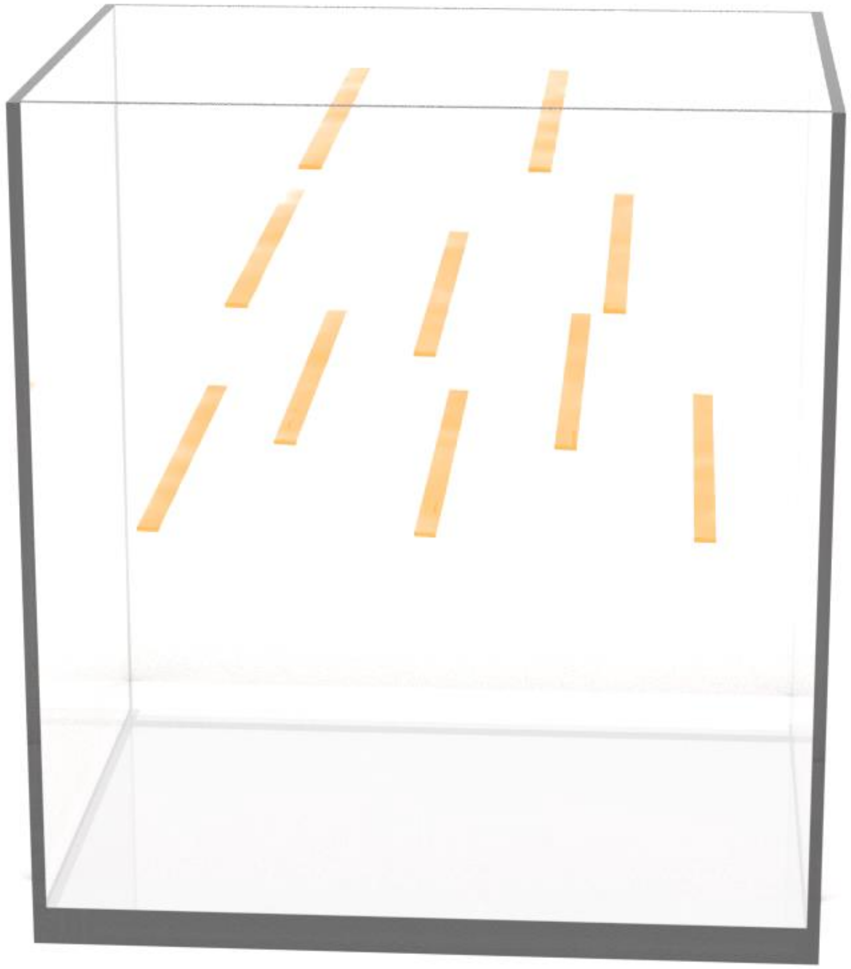

Dragon
80K DOFs
16.4 – 119.3 FPS

Tiered skirt

91K DOFs

12.2 – 29.1 FPS

“Animal Crossing”



218K DOFs

13.4 – 46.3 FPS


Halloween Party
249K DOFs
7.7 – 26.8 FPS