Interactive Collision Detection for Deformable and Fracturing Objects

Sung-Eui Yoon (윤성의)
IWON associate professor
KAIST

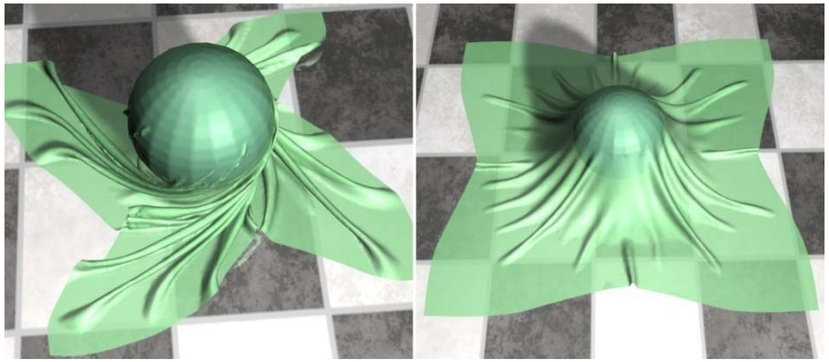
http://sglab.kaist.ac.kr/~sungeui/

Acknowledgements

- Research collaborators
  - DukSu Kim, JaePil Heo, John Kim, Miguel Otaduy, Tang Min, JeongMo Hong, JoonKyung Seong, Dinesh Manocha, JaeHyuk Heo
- Funding sources
  - Ministry of Knowledge Economy
  - Microsoft Research Asia
  - Samsung
  - Korea Research Foundation

Goals

- **Achieve interactive performance for collision detection among deformable and fracturing models**
  - **E.g., deforming models consisting of tens or hundreds of thousand triangles**



<Cloth-ball, 94K triangles>

<Breaking dragon, 252K triangles>

Overview

- Background
- Hybrid parallel proximity computation
- Fracturing-aware collision detection
- CD for volumetric representations
- Multi-resolution cloth simulation

Overview

- Background
- Hybrid parallel proximity computation
- Fracturing-aware collision detection
- CD for volumetric representations
- Multi-resolution cloth simulation

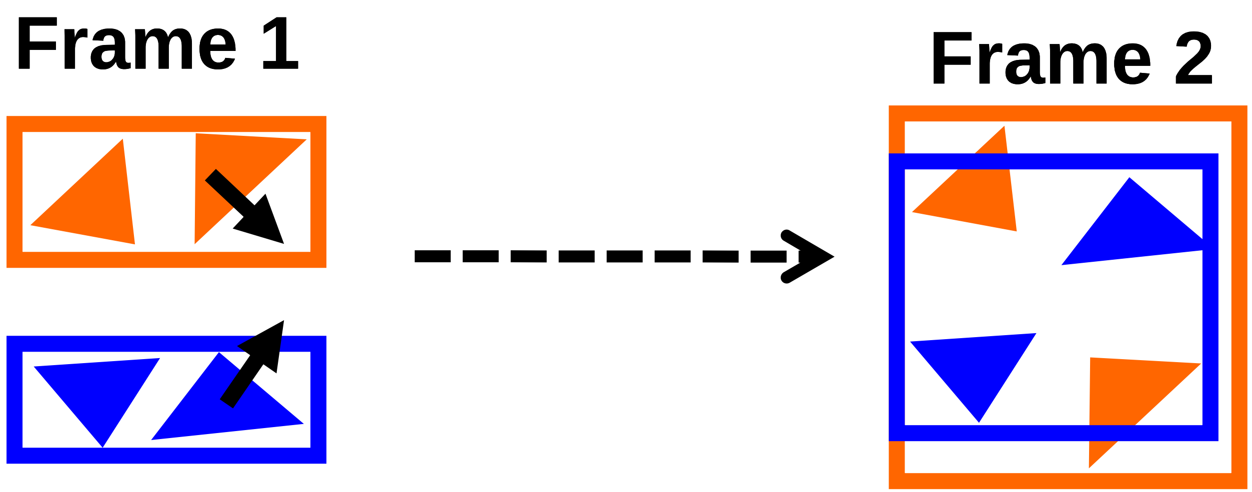
Background

- BVH-based collision detection
- BVH construction

- Updates BVHs as models deforms
  - Reconstruction from scratch
  - Refitting
  - Selective reconstruction

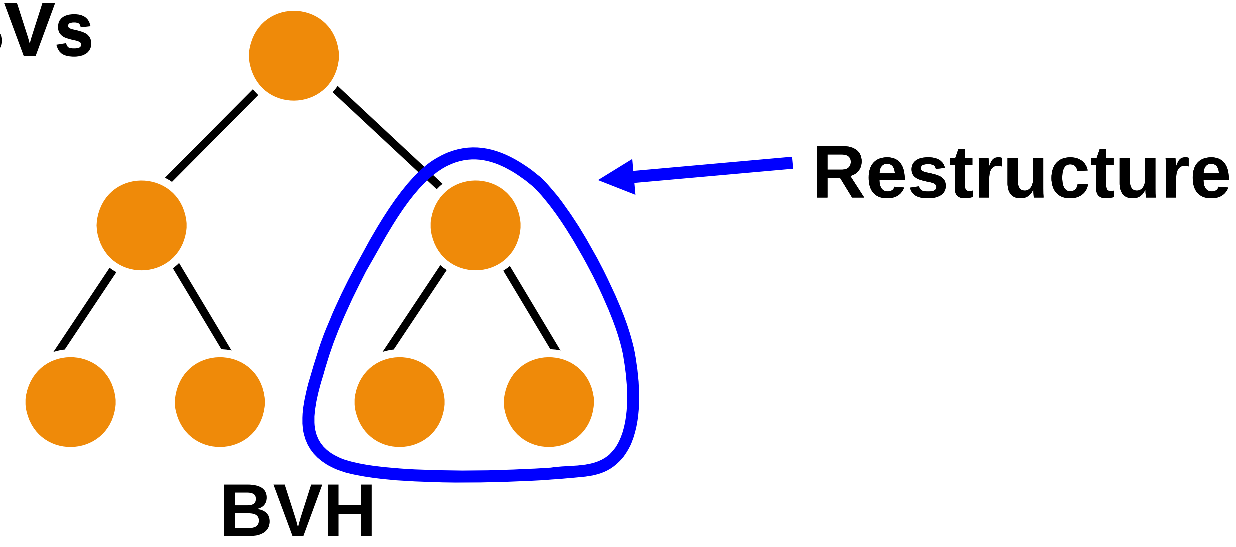
BVH Refitting

- Refit BVs with deformed vertices
  - Performed efficiently in a bottom-up traversal
  - Can have loose BVs when deformation levels are high



BVH Selective Restructuring

- **Restructure only subsets of BVHs after refitting BVs**



- **Requires a metric indentifying such subsets**
  - **Volume ratios of BVs of parent and child BVs [Zachmann 02, Larsson et al. 06, Yoon et al. 06]**

Discrete vs. Continuous

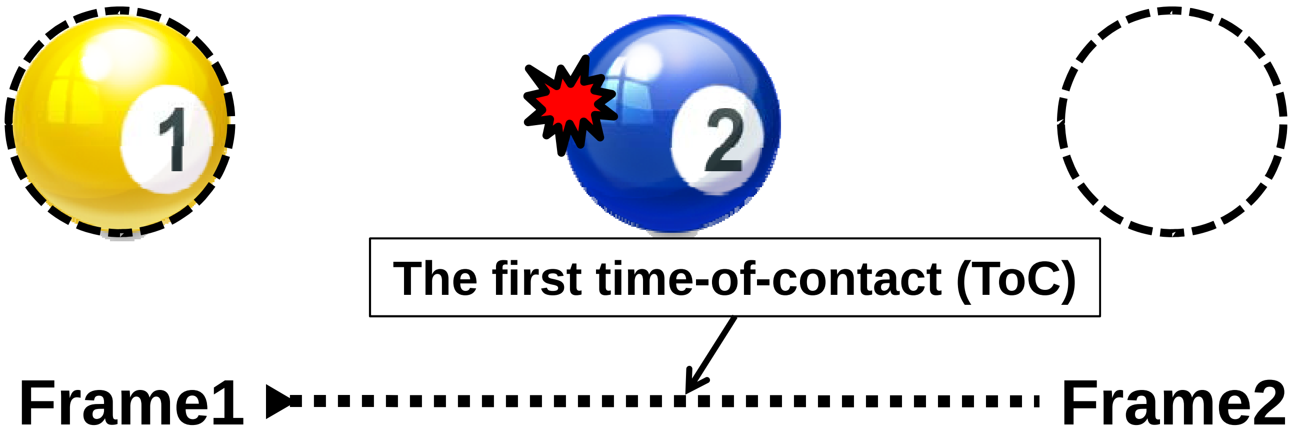
- **Discrete collision detection (DCD)**
  - **Detect collisions at each frame**
  - **Fast, but can miss collisions**

Miss collisions

Frame1 **Frame2**

Discrete vs. Continuous

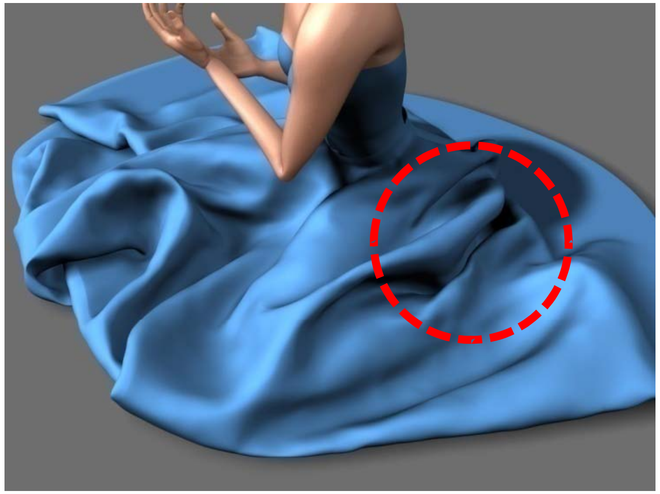
- Discrete collision detection (DCD)
- Continuous collision detection (CCD)
  - Identify the first time-of-contact (ToC)
  - Accurate, but requires a long computation time
  - Not widely used in interactive applications



Inter- and Self-Collisions

- **Inter-collisions**
  - **Collisions between two objects**

- **Self-collisions**
  - **Collisions between two regions of a deformable object**
  - **Takes a long computation time to detect**



From Govindaraju's paper

Overview

- Background
- Hybrid parallel proximity computation
- Fracturing-aware collision detection
- CD for volumetric representations
- Multi-resolution cloth simulation

Parallel Computing Trends

- Many core architectures
  - Multi-core CPU architectures
  - GPU architectures
- Heterogeneous architectures
  - Intel's Larabee and AMD's Fusion
- Designing parallel algorithms is important to utilize these parallel architectures

Recent Parallel Collision Detection Methods

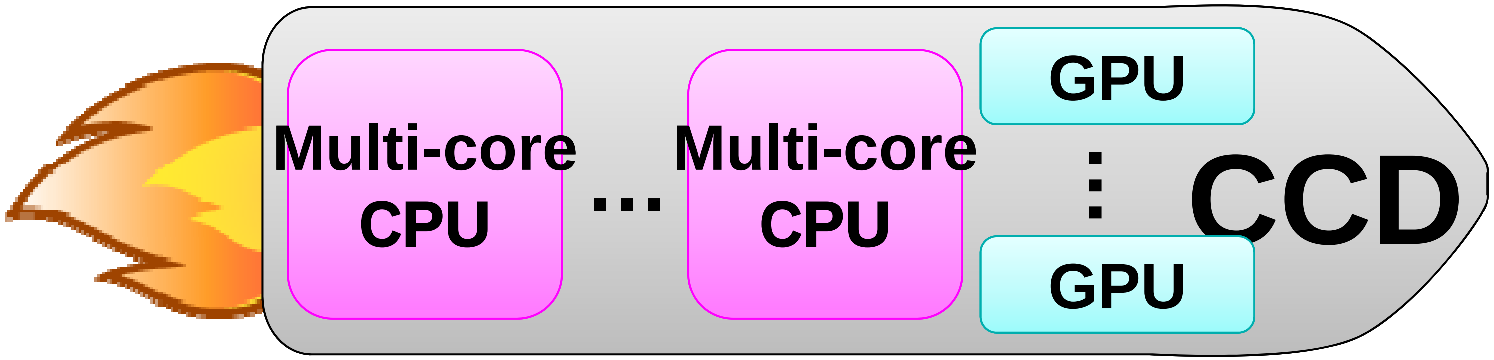
- **CPU-based CD method**
  - **Tang et al., Solid and Physical Modeling, 2009**
- **gProximity**
  - **Laterbach et al., Eurographics 2010**
- **Hybrid parallel CD method**
  - **Kim et al., Pacific Graphics 2009**

Recent Parallel Collision Detection Methods

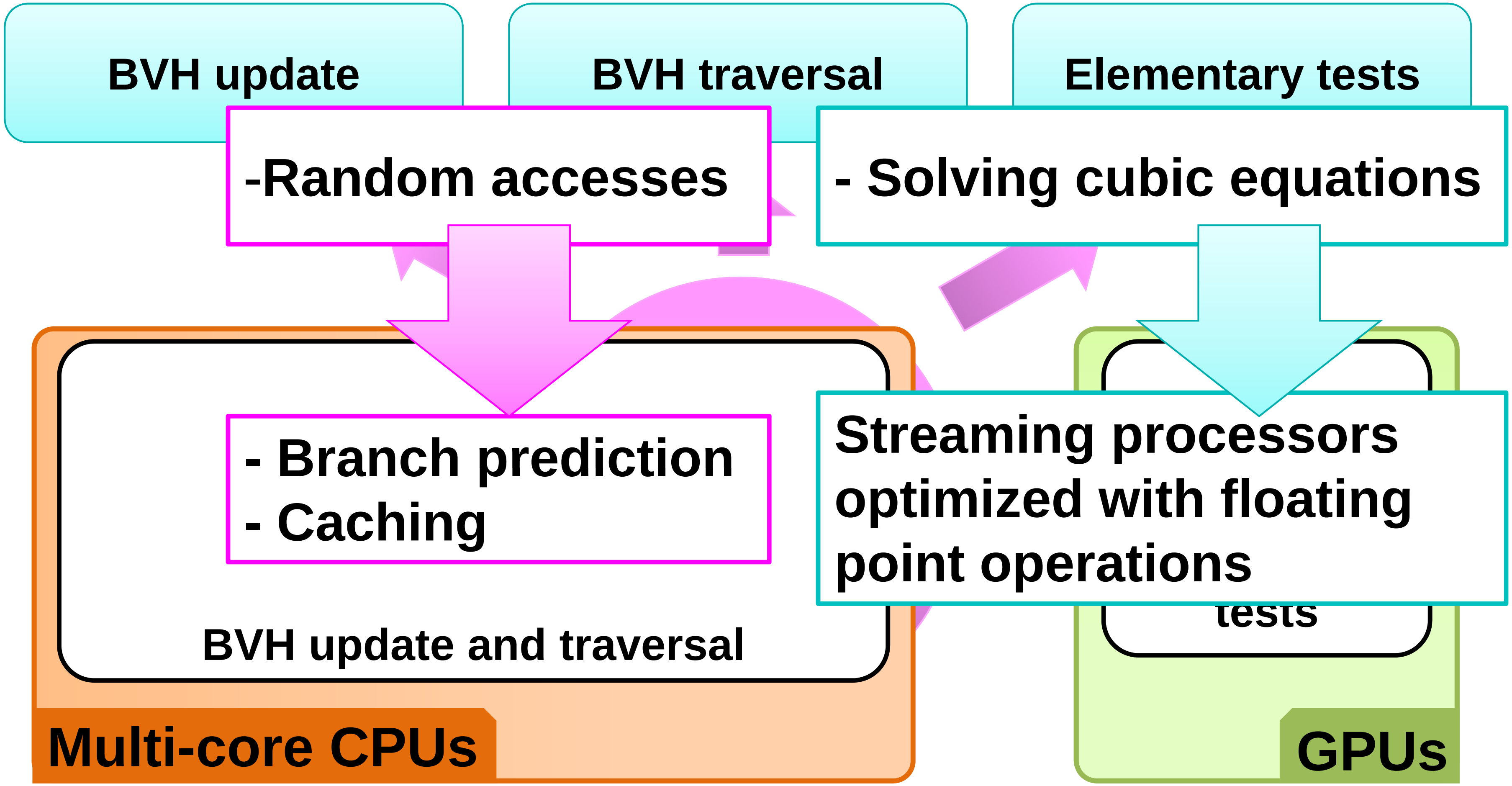
- CPU-based CD method
  - Tang et al., Solid and Physical Modeling, 2009
- gProximity
  - Laterbach et al., Eurographics 2010
- Hybrid parallel CD method
  - Kim et al., Pacific Graphics 2009
  - Received a distinguished paper award

HPCCD: Hybrid Parallel CCD

- Utilize both multi-core CPUs and GPUs
  - No locking in the main loop of CD
  - GPU-based exact CD between two triangles
- High scalability & interactive performance



Task Distribution



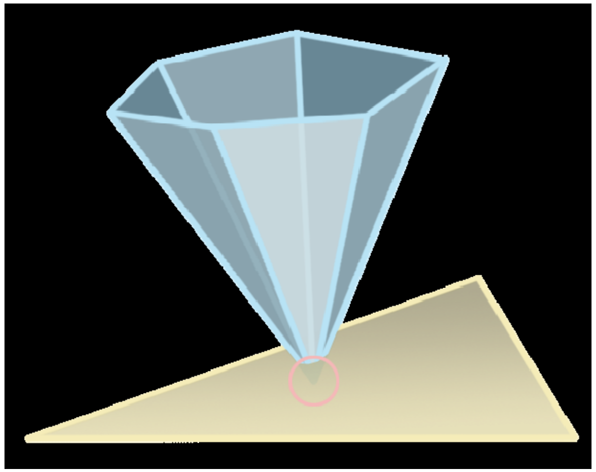
Testing Environment

- Machine
  - One quad-core CPU (Intel i7 CPU, 3.2 GHz)
  - Two GPUs (Nvidia Geforce GTX285)
- Run eight CPU threads by using Intel's hyper threading technology

KAIST

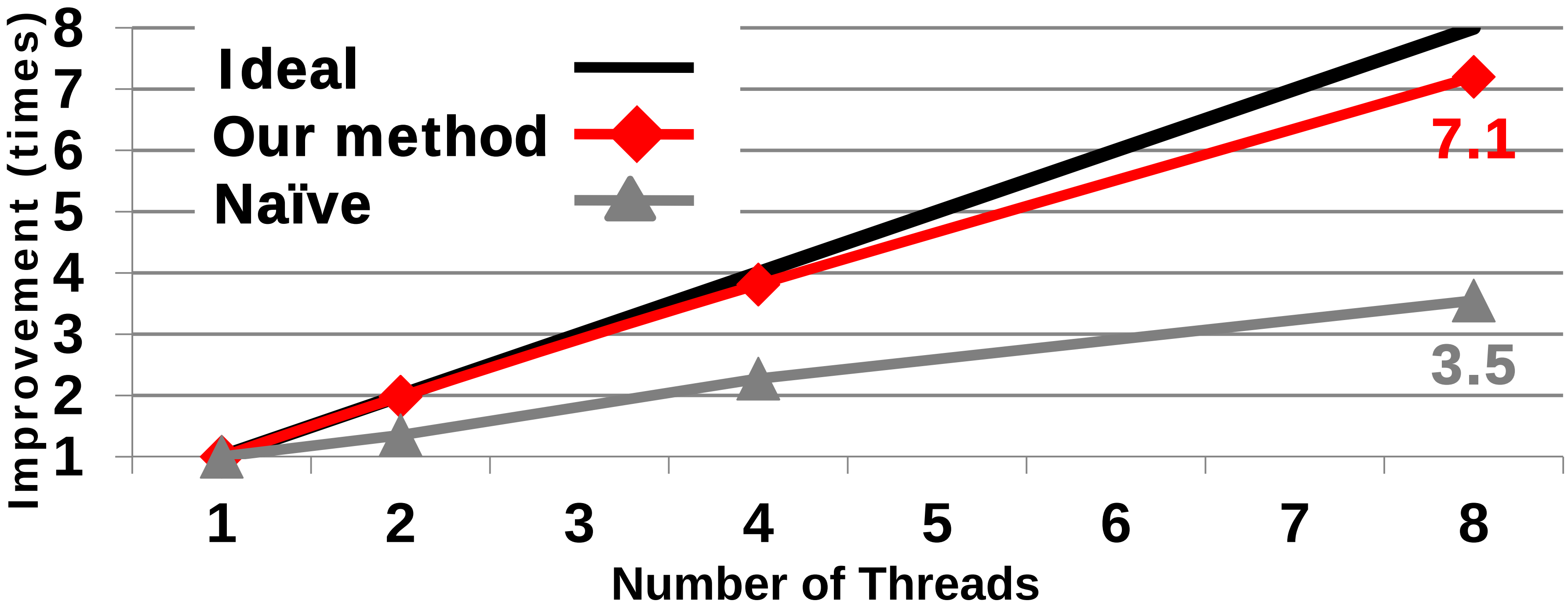
BVH-based CCD

- Axis-aligned bounding boxes
- Linear interpolations on vertices for the continuous motion
  - Vertex-face and edge-edge tests for CCD [Provot 96]
- Feature based BVHs [Curtis et al., I3D 08]
  - Assign each features (e.g., vertex and edge) to each triangle



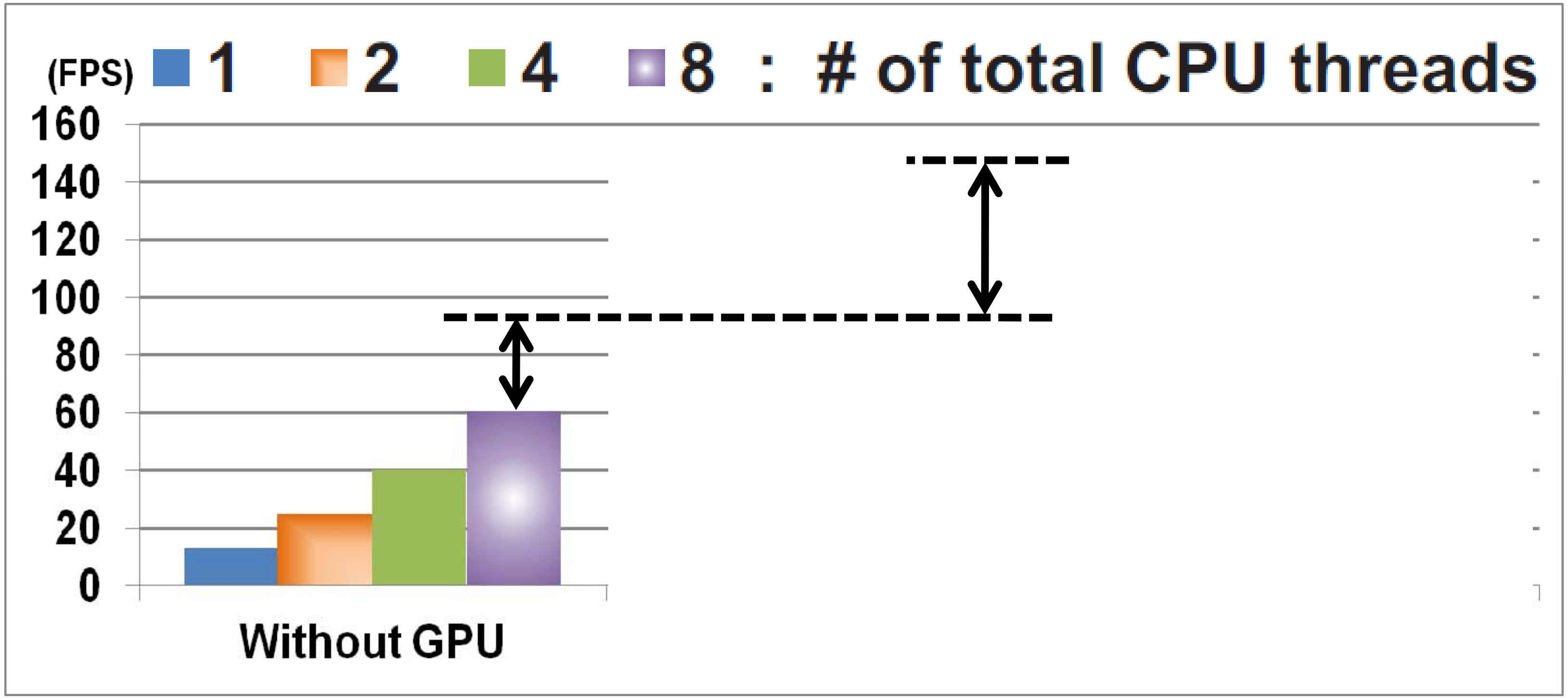
Results

Results of a CPU-based Parallel CCD



- **Remove locking in the main loop of CD**
- **Employ efficient dynamic load-balancing based on inter-CD task units**

Results of HPCCD



- As the number of GPUs is increased, we get higher performances

Limitation

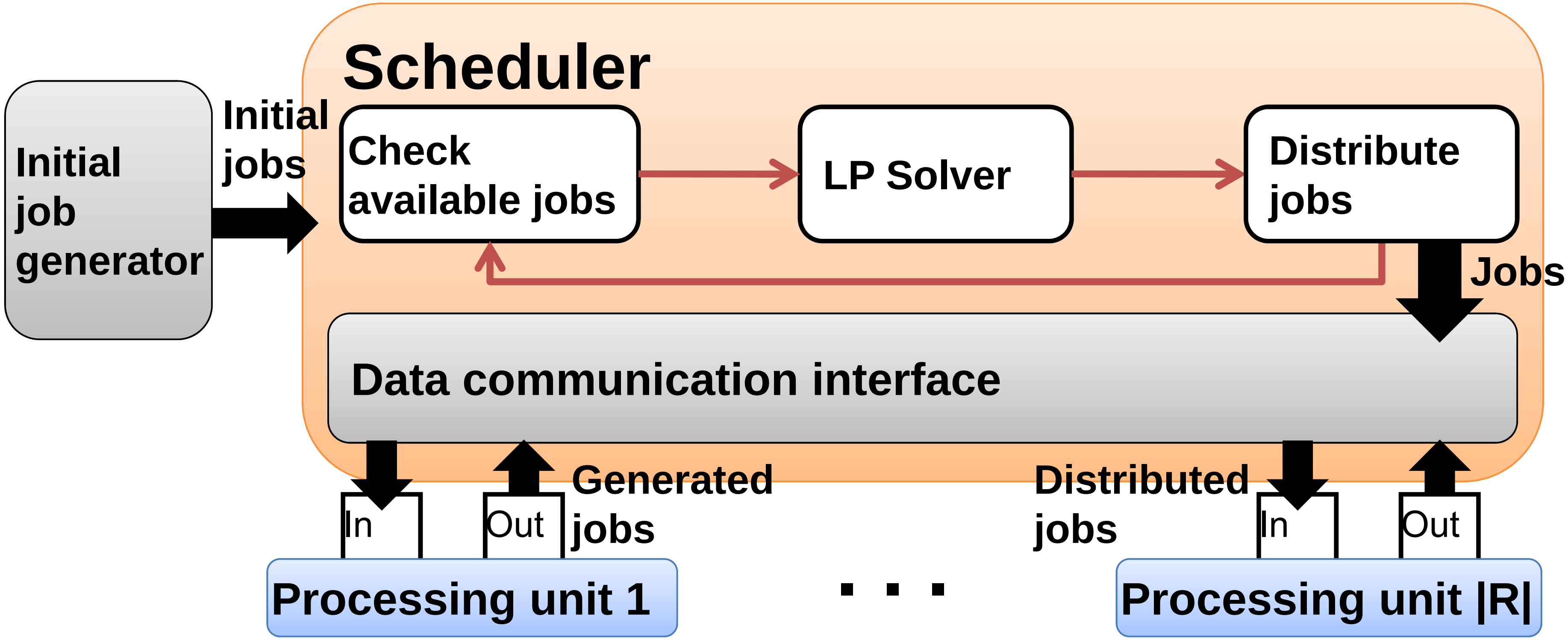
- Low scalability for small rigid models
- Specialized technique for continuous collision detection

Related Work

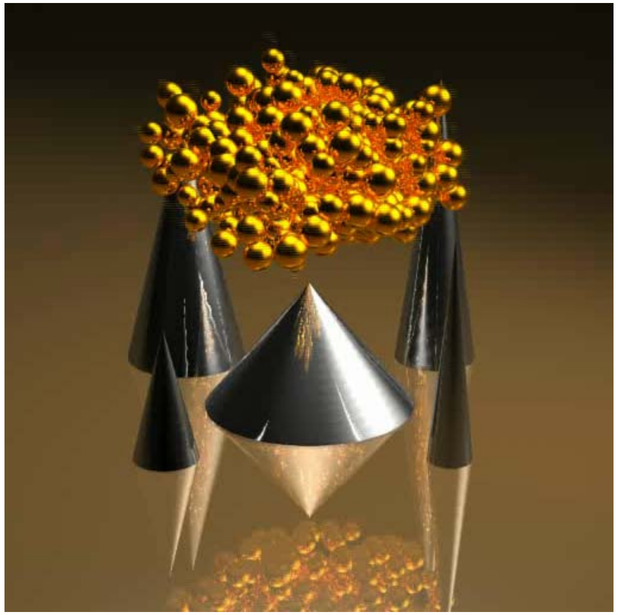
- **Hybrid parallel computation for proximity queries**
  - **A general, job distribution algorithm for CPUs and GPUs [Under submission]**

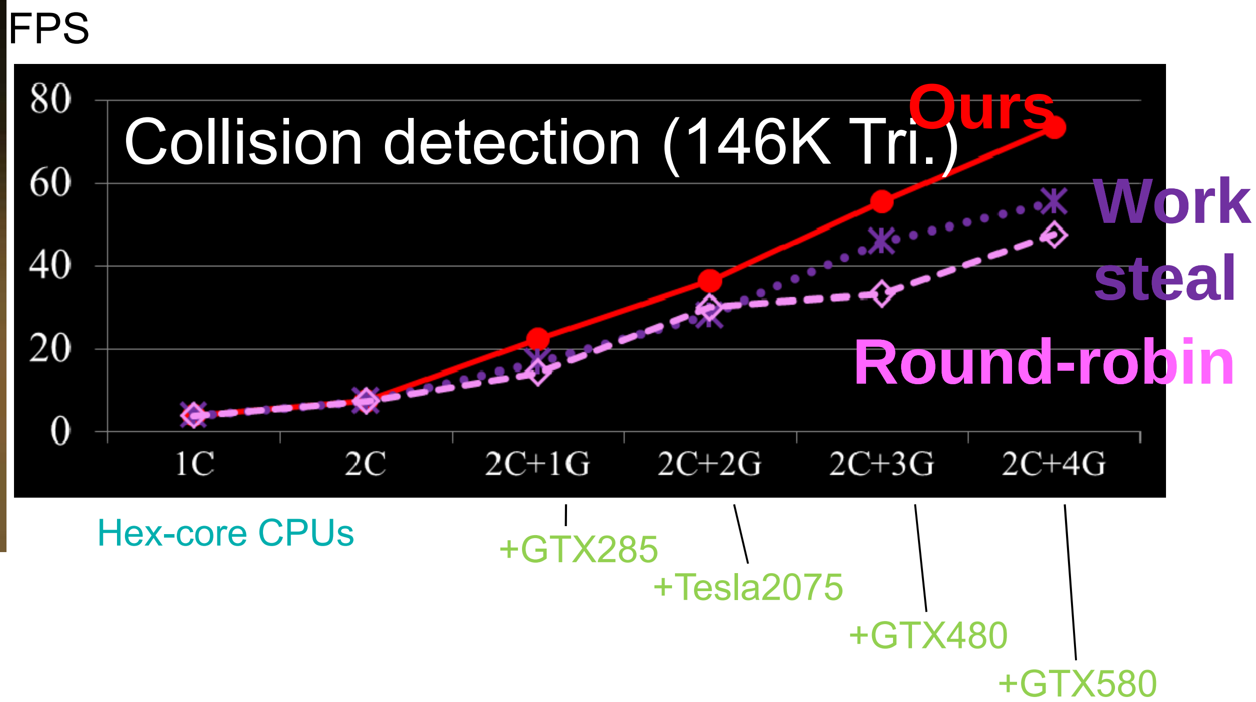
Motion planning **[Lee et al., ICRA 12]**

Overview of Hybrid Parallel Framework

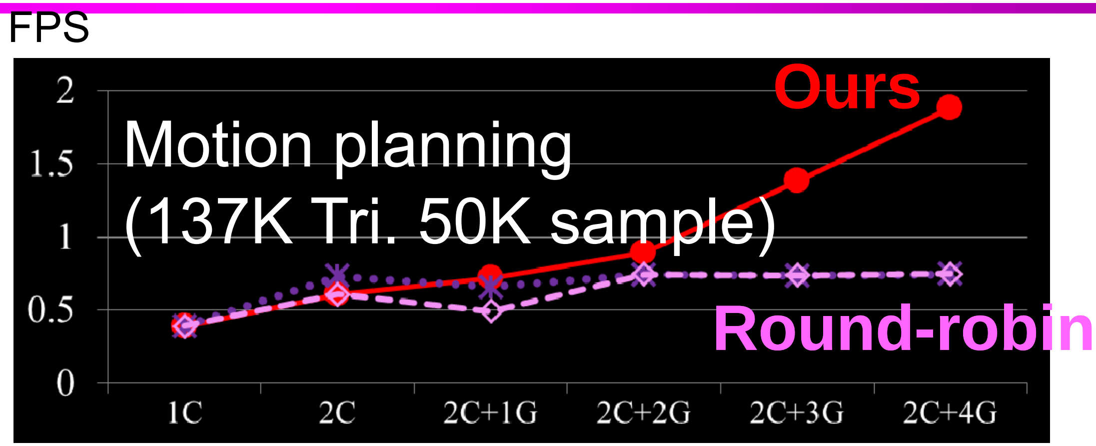


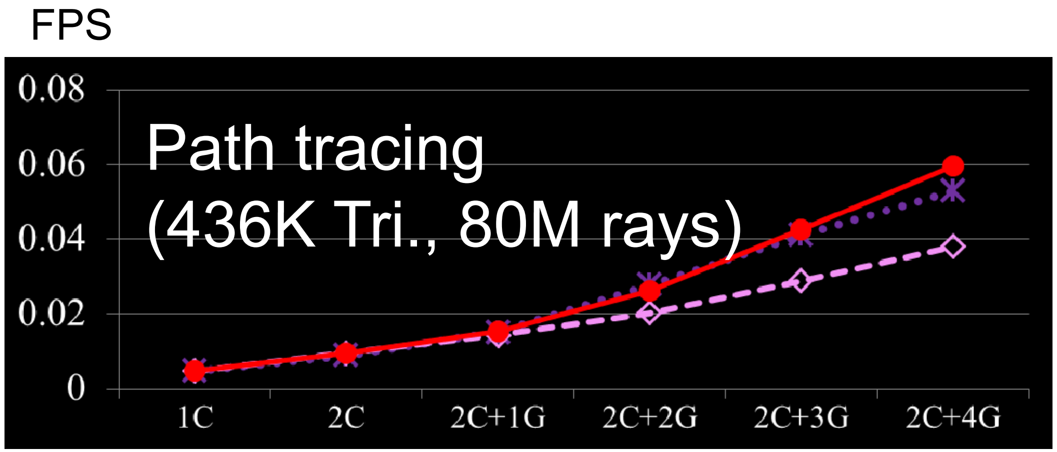
Results




FPS
Collision detection (146K Tri.)
80
60
40
20
0
1C
2C
2C+1G
2C+2G
2C+3G
2C+4G
Ours
Work steal
Round-robin
Hex-core CPUs
+GTX285
+Tesla2075
+GTX480
+GTX580

Results

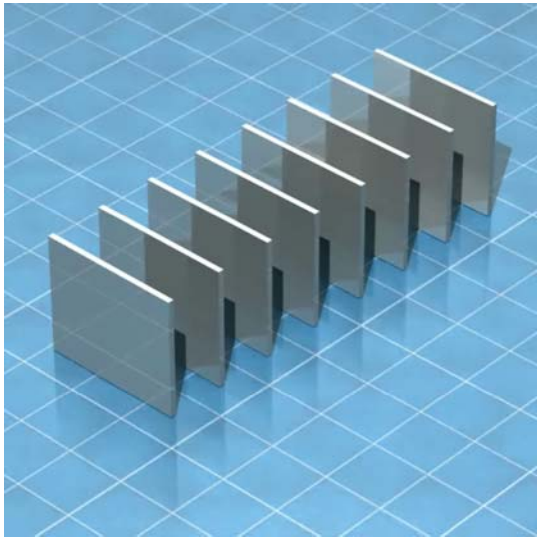

FPS
Ours
Motion planning
(137K Tri. 50K sample)
Round-robin
2
1.5
1
0.5
0
1C
2C
2C+1G
2C+2G
2C+3G
2C+4G


FPS
Path tracing
(436K Tri., 80M rays)
0.08
0.06
0.04
0.02
0
1C
2C
2C+1G
2C+2G
2C+3G
2C+4G

Overview

- Background
- Hybrid parallel proximity computation
- Fracturing-aware collision detection
- CD for volumetric representations
- Multi-resolution cloth simulation

CD for Fracturing Models

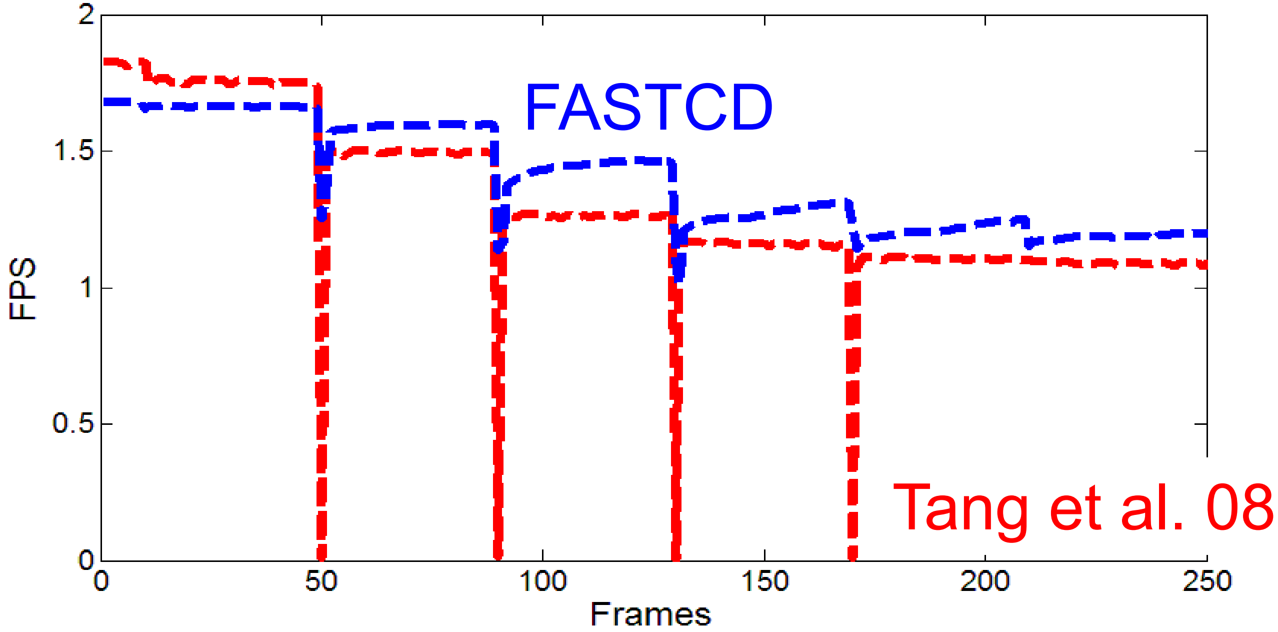


- Fracturing becomes to be widely used, but is one of the most challenging scenarios of collision detection
- Fracturing
  - Changes the connectivity of a mesh: pre-computed hierarchies show low culling ratios
  - Places many objects in close proximity: CD cost is increasing

Our Approach

- FASTCD: Fracturing-Aware Stable CD [Heo et al., SCA 10]
  - Incrementally update meshes and BVHs by utilizing topological changes of models
  - Design a simple self-CD culling method without much pre-computations

CCD Performance with the Breaking Dragon Model



- **FASTCD shows stable performance**

Selective Restructuring of BVHs

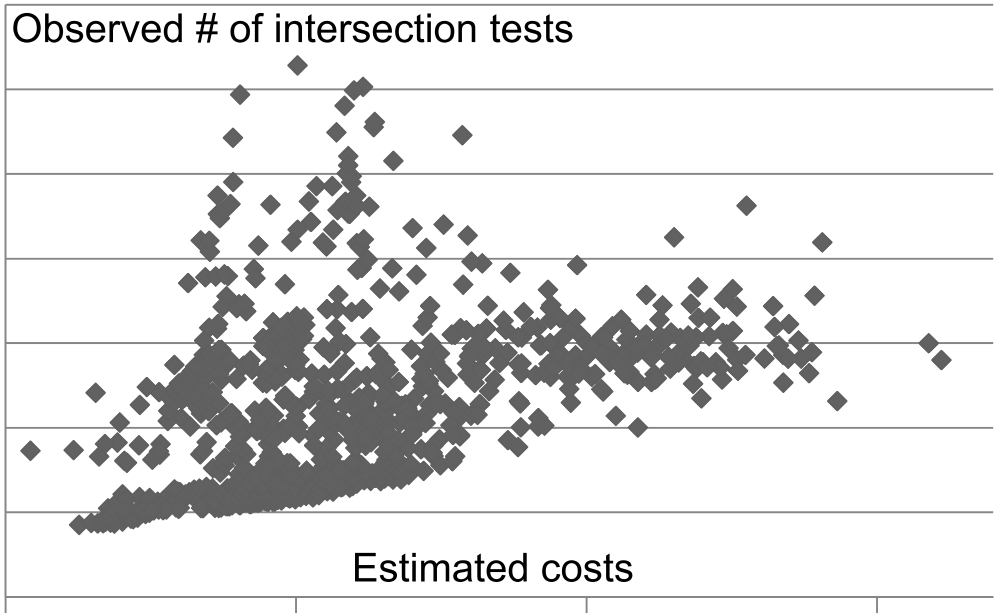
- As models deform, culling efficiency of their BVHs can be getting lower
  - Selective restructuring can address the problem

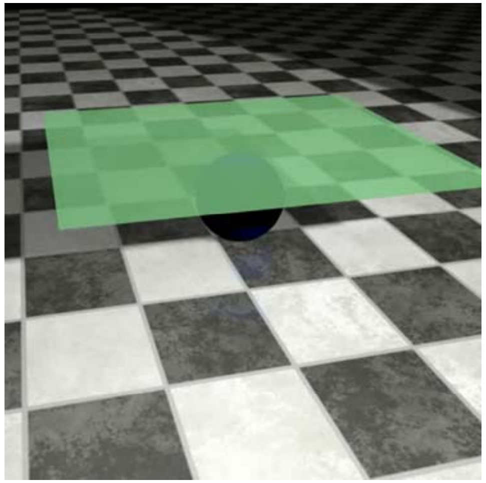
- How to determine a culling efficiency of a BVH?
  - Heuristic metrics have been proposed
- **Propose a cost metric that measures the expected number of intersection tests**

KAIST

Metric Validation

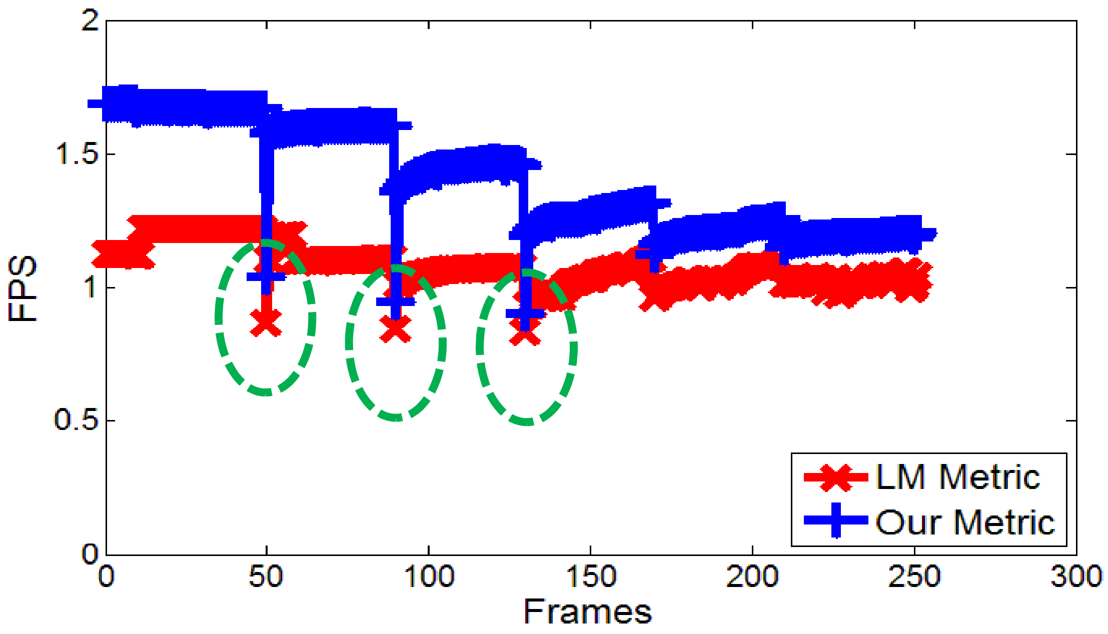
- **Estimated # of tests vs. Observed # of tests**





- **Linear Correlation : 0.71**
  - **Tested with various models (0.28 ~ 0.76 , average 0.48)**

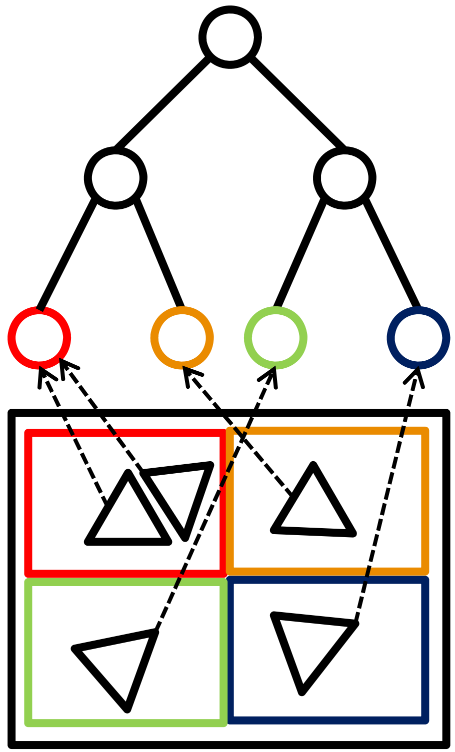
Result of Selective Restructuring



- **LM metric : [Larsson and Akenine-Möller 2006]**
- **Performance degradations at topological changes → unstable**

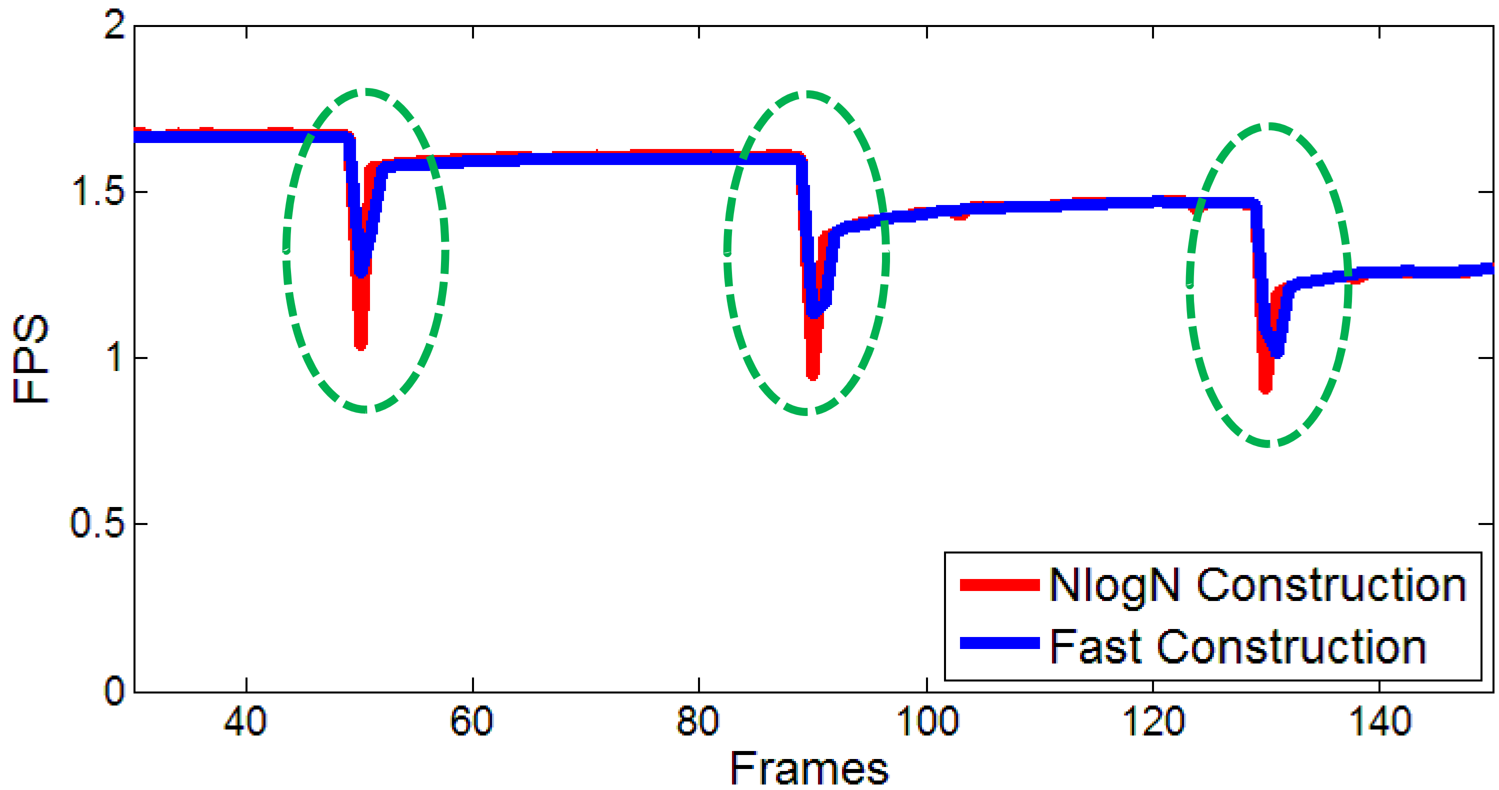
Fast BVH Construction Method

- At a fracturing event, BVHs for fractured parts should be updated for high culling efficiencies
  - Causes noticeable performance degradations
- Propose a BVH construction method based on grid and hashing, instead of typical NlogN methods



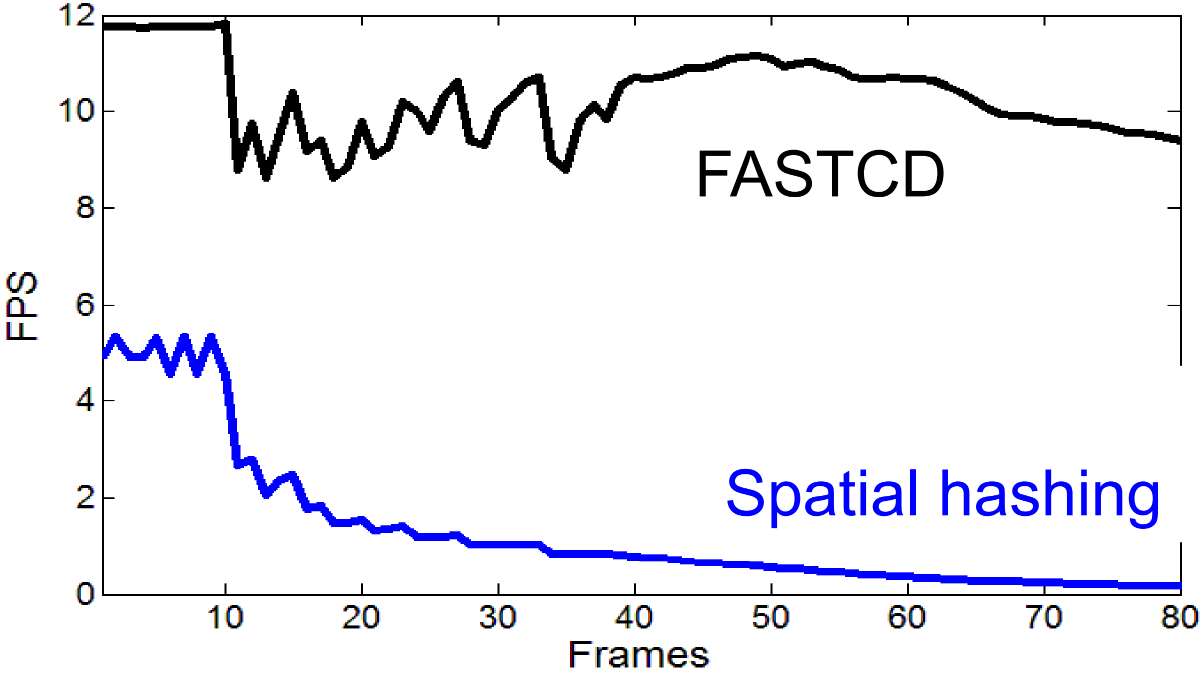
- Constructed hierarchies have low culling efficiencies, but use less construction times
  - Improve the overall performance at fracturing events

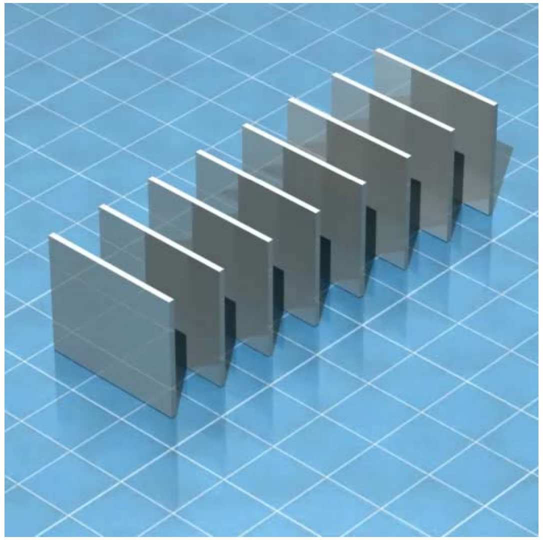
Result of Fast BVH Construction



- **Performance degradations at fracturing events are reduced**

Comparison (Discrete CD)





- **20x faster than optimized spatial hashing [Teschner et al, 2003]**
- **Stable performance**

KAIST

Overview

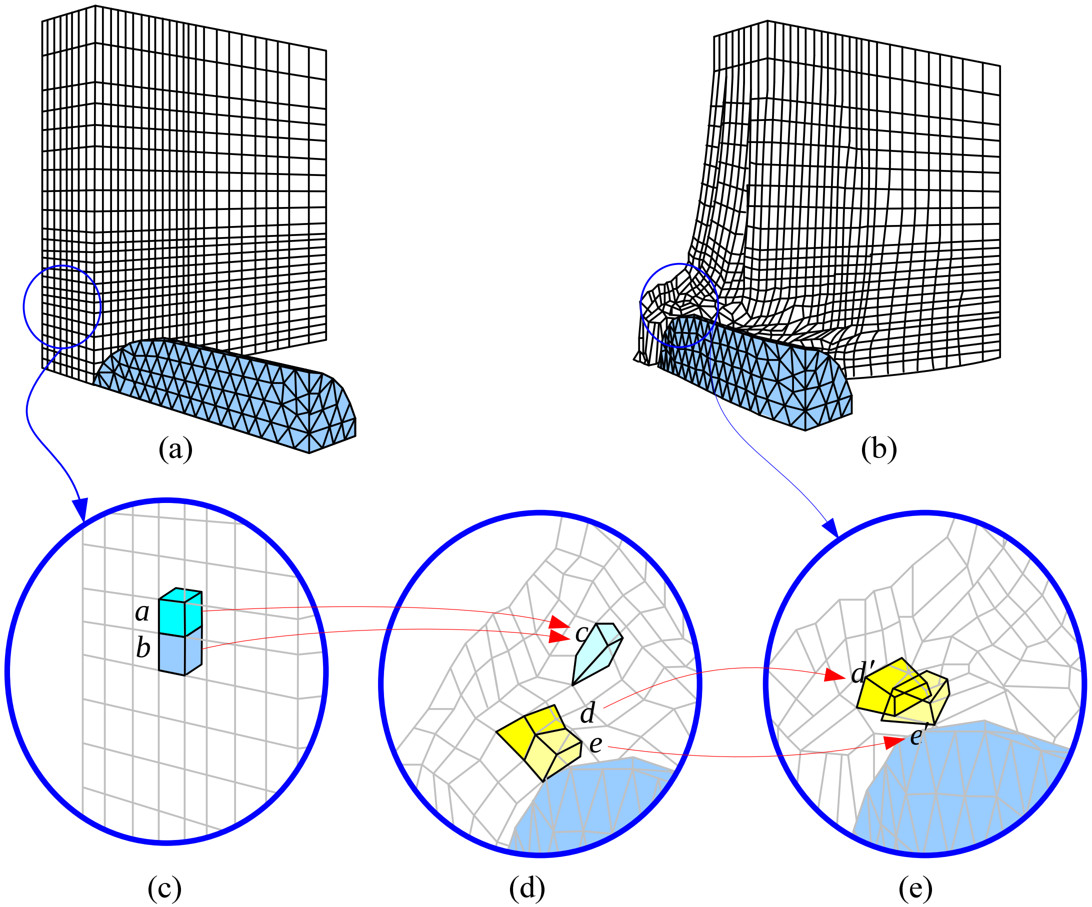
- Background
- Hybrid parallel proximity computation
- Fracturing-aware collision detection
- CD for volumetric representations
- Multi-resolution cloth simulation

Contributions

- **Novel culling algorithms to perform fast and robust continuous collision detection between volume meshes [Tang et al. 11, ToG]**
  - **A continuous separating axis test (CSAT) between volume meshes**
  - **Feature-level culling to eliminate redundant tests**

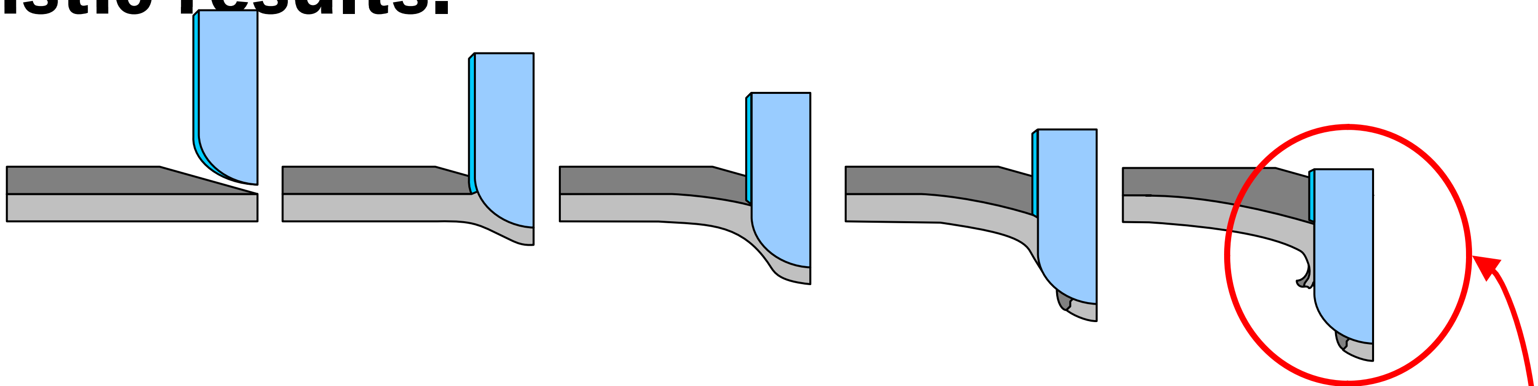
Motivation

- **Topological changes make interior elements colliding with each other**

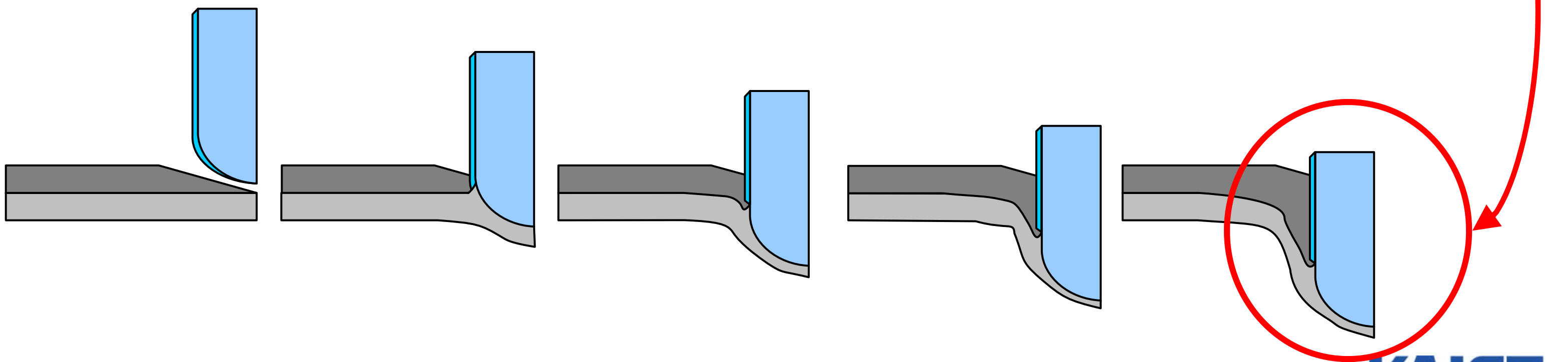


Motivation

- **Check both contacts between external and internal elements to compute realistic results.**



(a) Simulation with volume-based collision detection

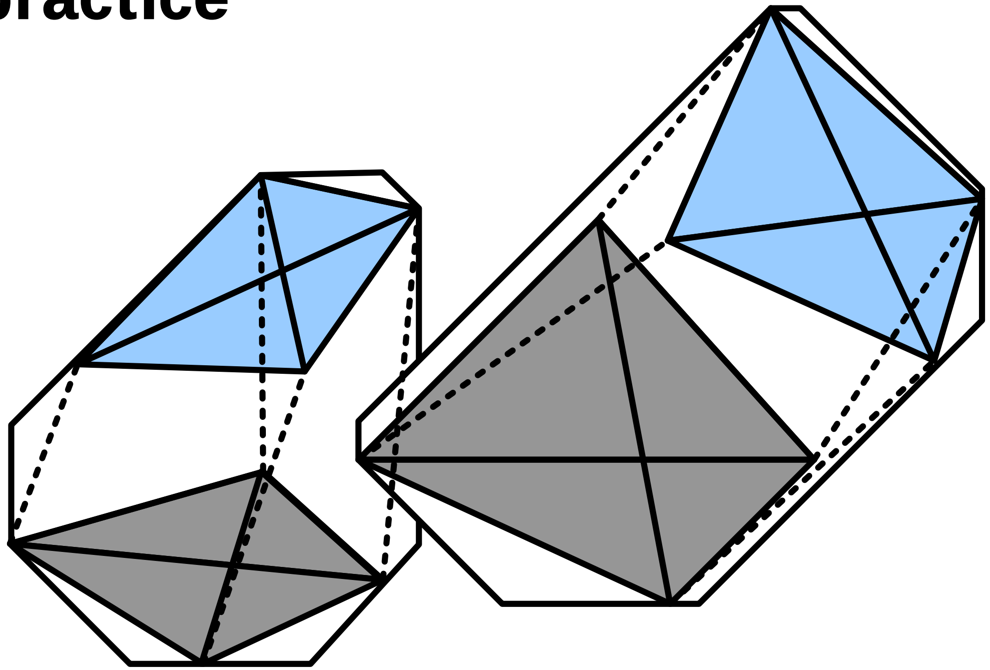


(b) Simulation with surface-based collision detection

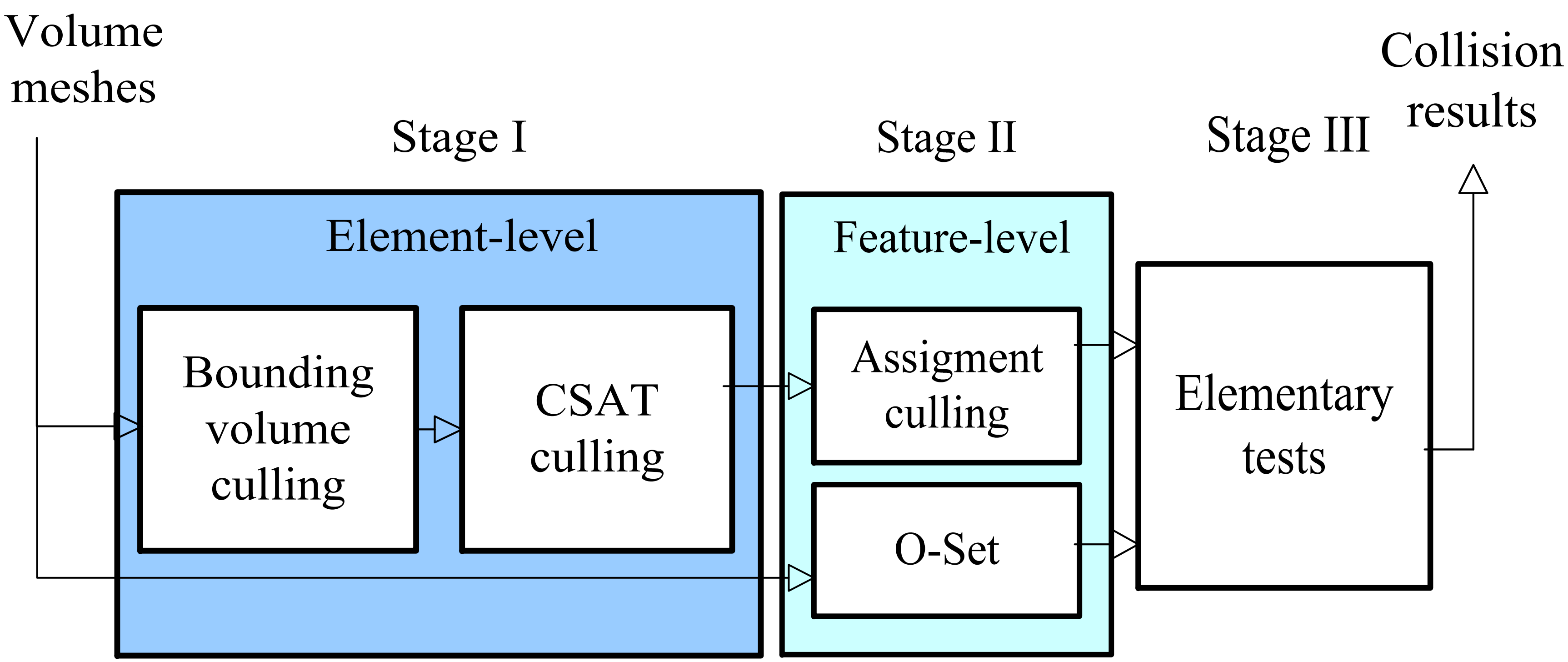
KAIST

Bounding Volumes

- We use tight fitting k-DOPs to enclose the swept volume of each deforming volumetric element
  - Can result in a high number of false positives in practice

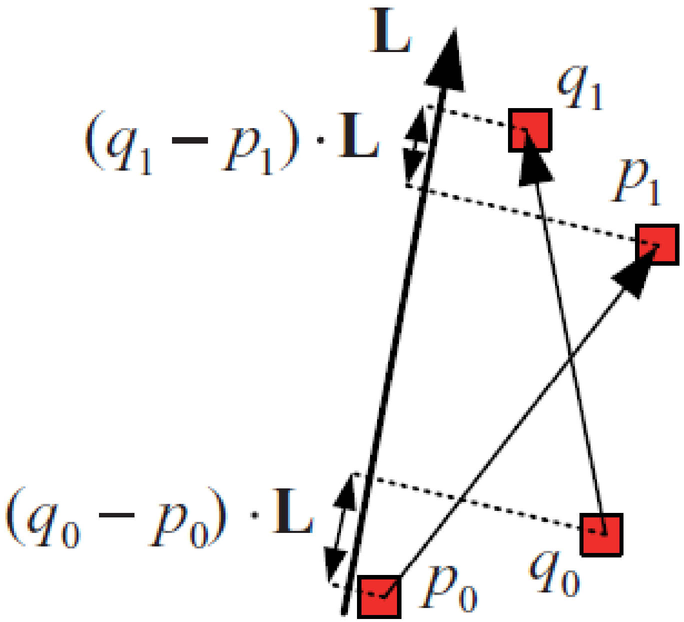


Our CCD Algorithm



Continuous Separating Axis Theorem

- An example of two moving vertices
- Look at the distance between two vertices
  - Consider this in a line, L



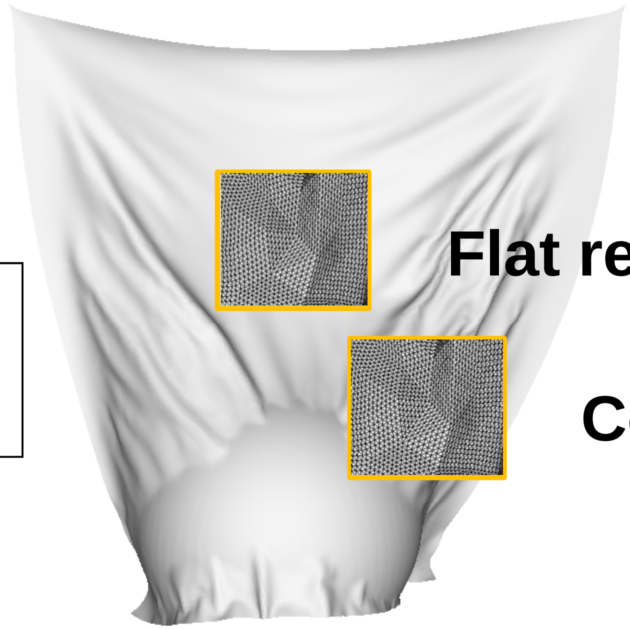
Results

Overview

- Background
- Hybrid parallel proximity computation
- Fracturing-aware collision detection
- CD for volumetric representations
- Multi-resolution cloth simulation

Common single resolution cloth simulation

$$\mathbf{A}\Delta\mathbf{x} = \mathbf{b}$$

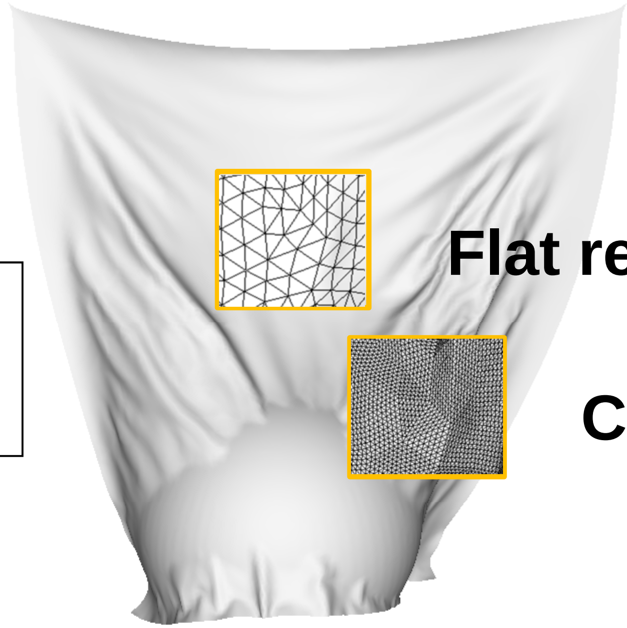


Very high resolution enough to capture detailed features

Significantly slow simulation performance

Multi-resolution cloth simulation

$$\mathbf{A}\Delta\mathbf{x} = \mathbf{b}$$



Flat region

Complex region

Improve speed with less vertices

maintaining simulation quality

Multi-Resolution Cloth Simulation [Lee et al., PG 10]

Summary

- Presented recent BVH-based methods for interactive CD among large-scale deforming models

◆ The code of HPCCD is available as OpenCCD library (http://sglab.kaist.ac.kr/OpenCCD)

◆ Various models are available at KAIST Model Benchmark:

http://sglab.kaist.ac.kr/models/