Interactive Collision Detection for Large-Scale Deformable Objects

Sung-Eui Yoon
Scalable Graphics Lab.
KAIST

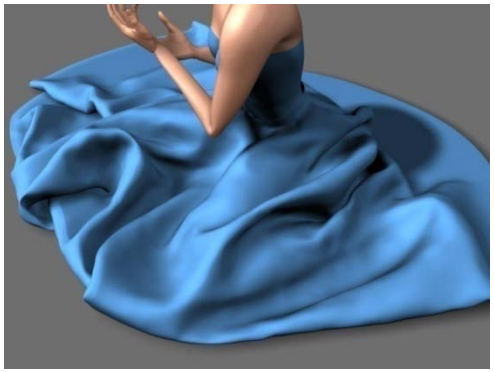
http://sglab.kaist.ac.kr/~sungeui/

KAIST

Collision Detection

- **Collision detection is used in various fields**
  - **Game, movie, scientific simulation and robotics**

<Figure from PIXAR>



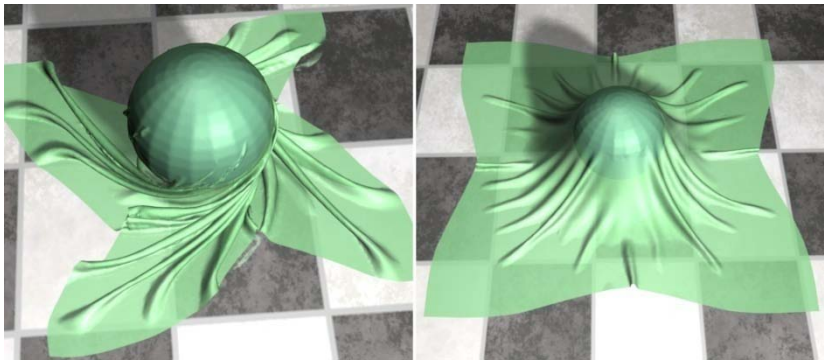
<Figure from C. Lauterbach >

<Figure from AION >

KAIST

Goals

- **Achieve interactive performance for exact collision detection between large-scale deformable models**
  - **E.g., deforming models consisting of tens or hundreds of thousand triangles**



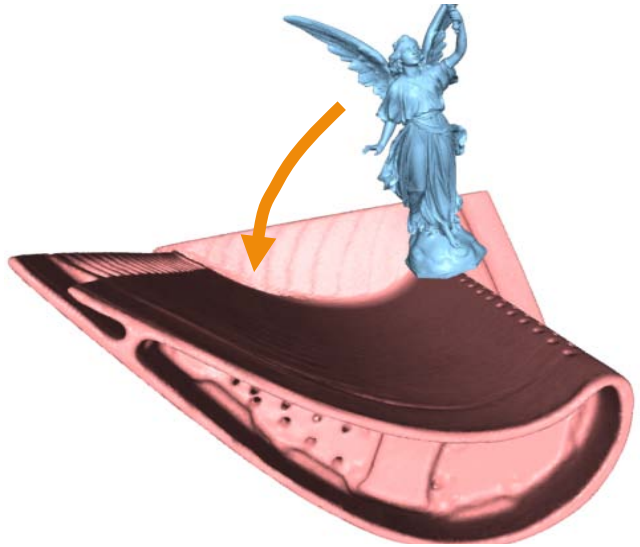
<Cloth-ball, 94K triangles>

<Breaking dragon, 252K triangles>

KAIST

Large-Scale Applications

- Entertainment (games/movies)
- E-heritage applications
- Virtual reality, etc



KAIST

Overview

- **HPCCD: Hybrid parallel continuous collision detection**
- **FASTCD: Fracturing-Aware Stable Collision Detection**
- **HCCMeshes: Hierarchical-Culling oriented Compact Meshes**

KAIST

Overview

- **HPCCD: Hybrid parallel continuous collision detection**
- **FASTCD: Fracturing-Aware Stable Collision Detection**
- **HCCMeshes: Hierarchical-Culling oriented Compact Meshes**

KAIST

Discrete vs. Continuous

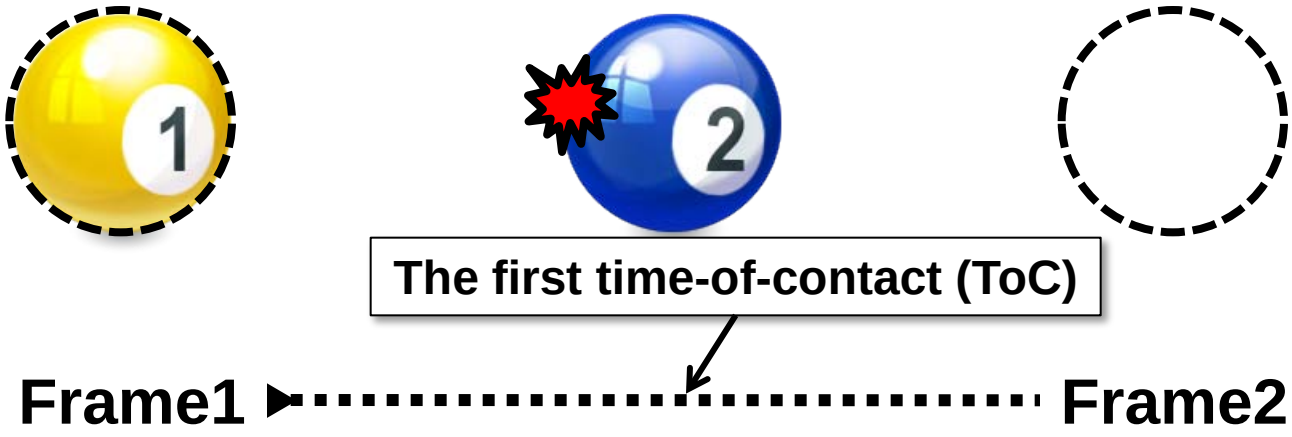
- **Discrete collision detection (DCD)**
  - **Detect collisions at each frame**
  - **Fast, but can miss collisions**

Miss collisions

Frame1 **Frame2**

Discrete vs. Continuous

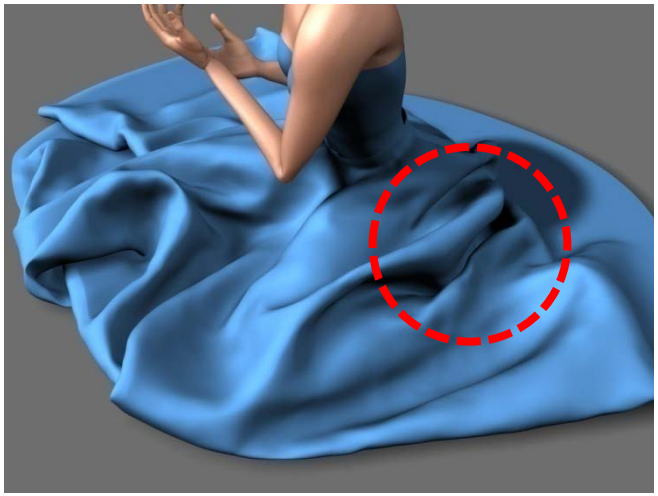
- Discrete collision detection (DCD)
- **Continuous collision detection (CCD)**
  - **Identify the first time-of-contact (ToC)**
  - **Accurate, but requires a long computation time**
  - **Not widely used in interactive applications**



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

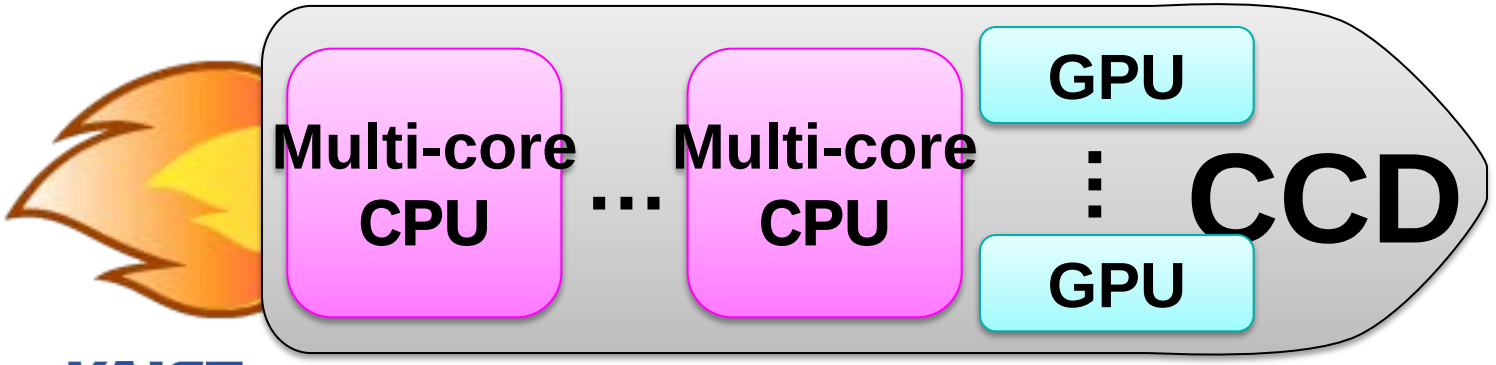
Parallel Computing Trends

- 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

Main Contributions

- **A novel, hybrid parallel CCD method**
  - **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**

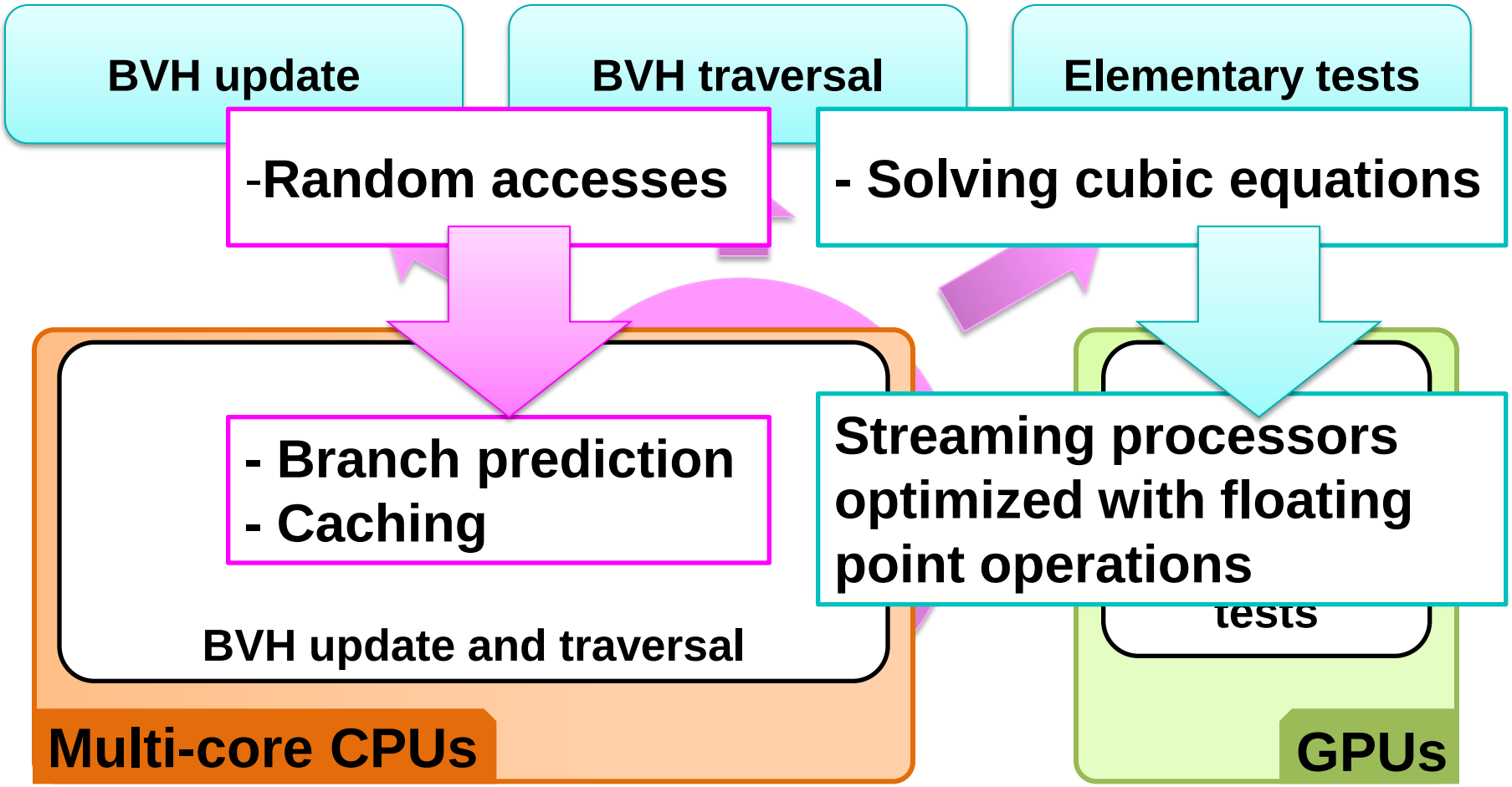
- Kim et al. PG 09
- Received a distinguished paper award



KAIST

Pacific Graphics 2009

Task Distribution



Testing Environment

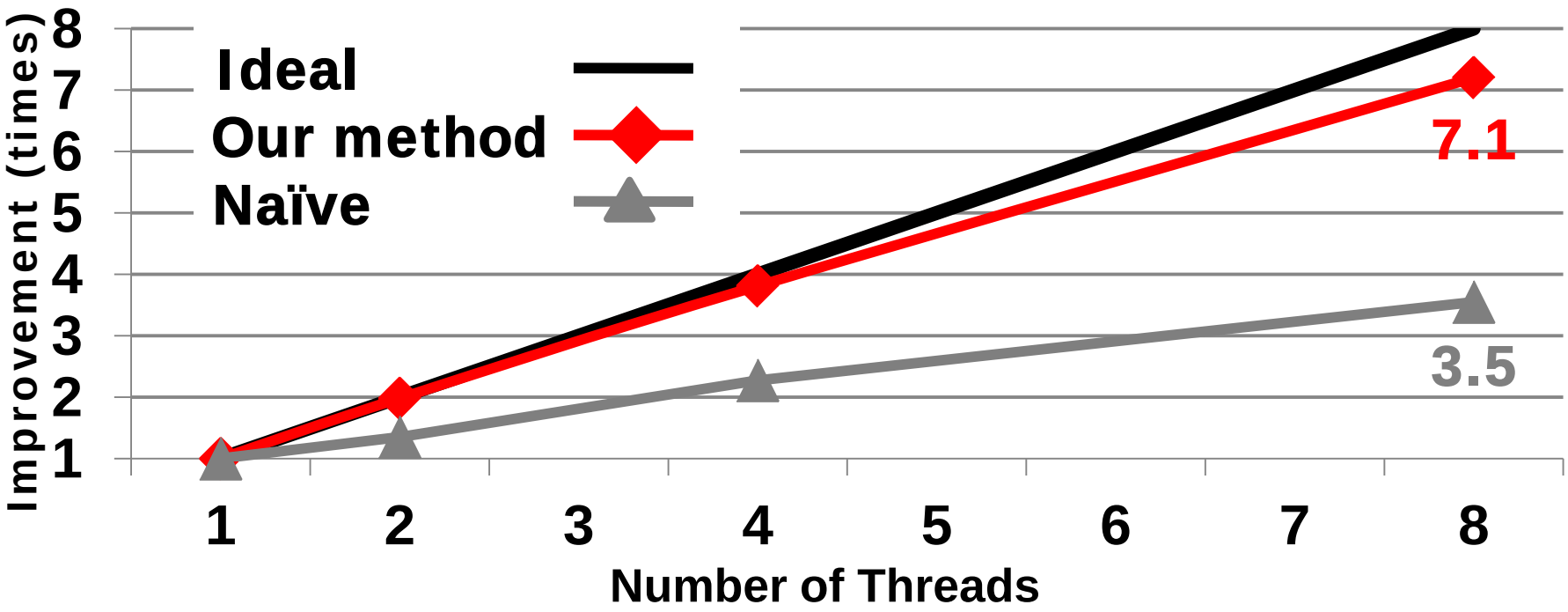
- **Machine**
  - **One quad-core CPU (Intel i7 CPU, 3.2 GHz)**
  - **Two GPUs (Nvidia Geforce GTX285)**
- **Run eight CPU threads by using Intel's hyper threading technology**

KAIST

Results

KAIST

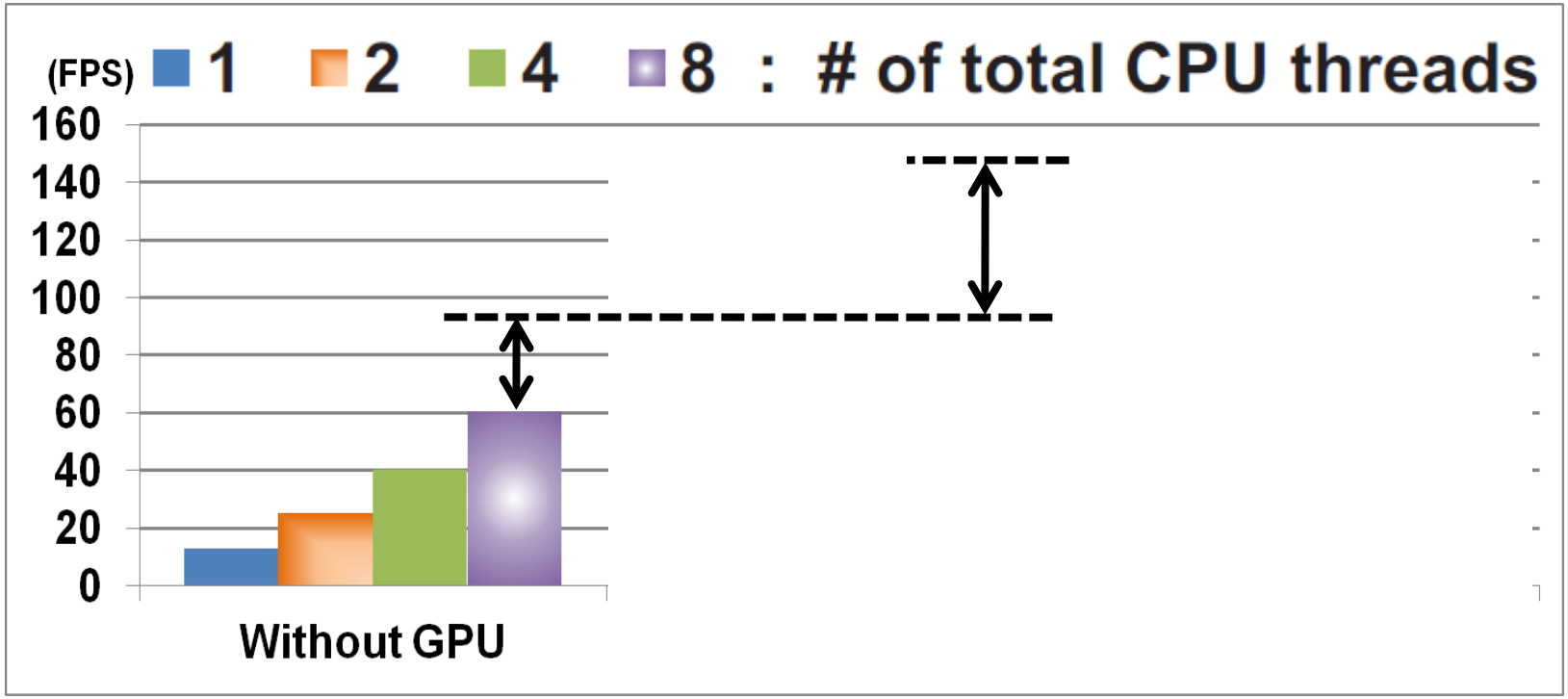
Results of Our CPU-based Parallel CCD



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

KAIST

Results of HPCCD



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

KAIST

Limitation

- **Low scalability for small rigid models**

KAIST

Summary

- A novel, hybrid parallel algorithm
  - Utilize both multi-core CPUs and GPUs
- High scalability

The implementation code will be available as

OpenCCD library

(http://sglab.kaist.ac.kr/OpenCCD)

  - Show 19-140 FPS for various deformable models consisting of tens or hundreds of thousand triangles

Overview

- **HPCCD: Hybrid parallel continuous collision detection**
- **FASTCD: Fracturing-Aware Stable Collision Detection**
- **HCCMeshes: Hierarchical-Culling oriented Compact Meshes**

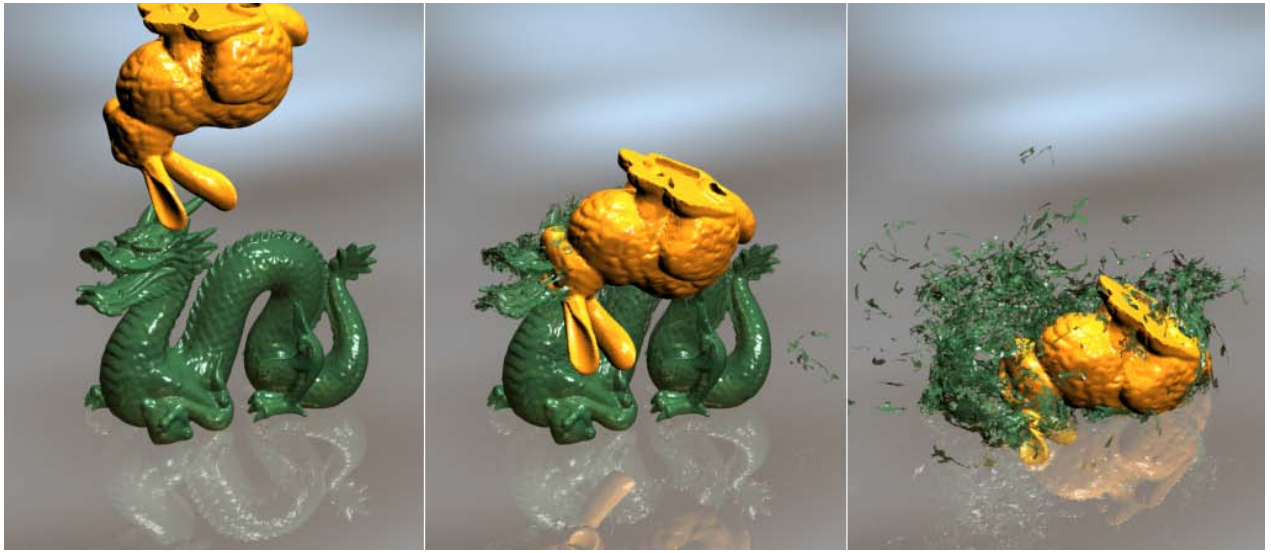
KAIST

Fracturing in Simulations

- More widely used in various applications to create more realistic interactions
- Presents significant challenges to CD
- Self-CD can take more expensive time with fracturing models
  - Fractured segments can be located in a near proximity

KAIST

Examples (Video)



252K

142K

KAIST

Research Challenges with Fracturing Models

- **Most prior works assume fixed topologies**
  - **Can take prohibitive times when models change their topologies**
- **Self-CD can take more expensive time with fracturing models**
  - **Fractured segments can be located in a near proximity**

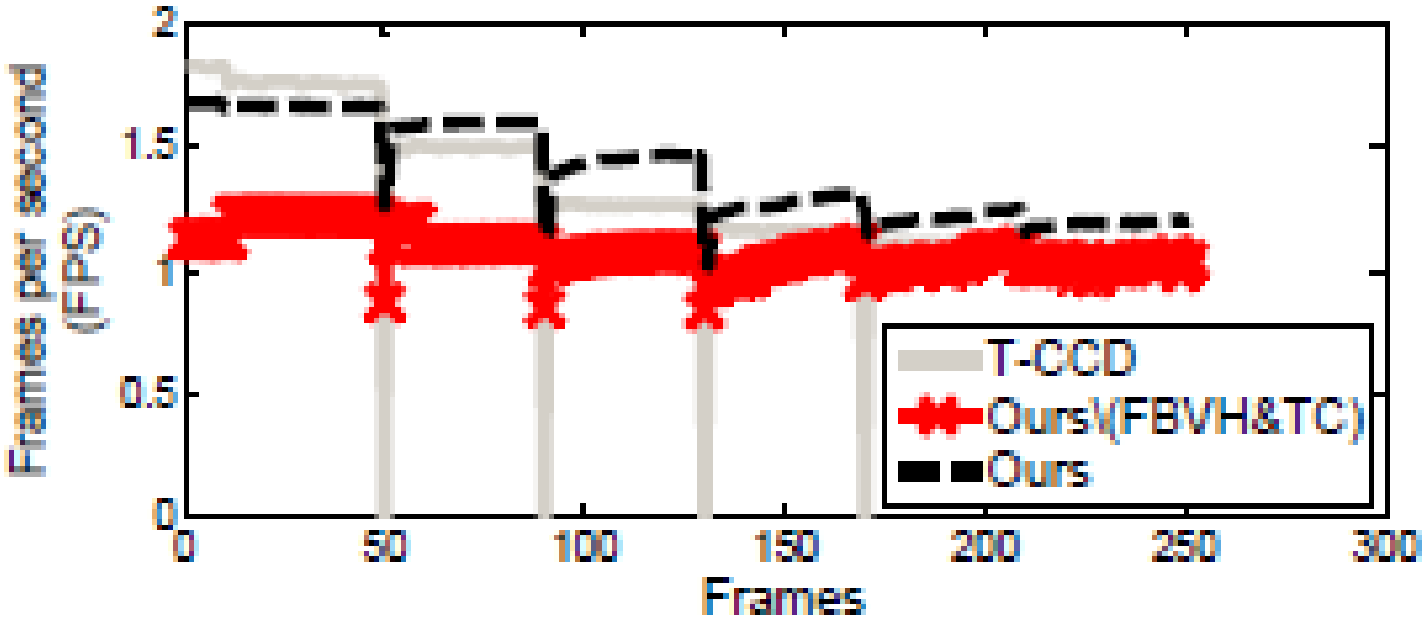
KAIST

Our Approach

- **Fracturing-aware stable CD**
  - **Incrementally update meshes and BVHs by utilizing topological changes of models**
  - **Design a simple self-CD culling method without much pre-computations**

KAIST

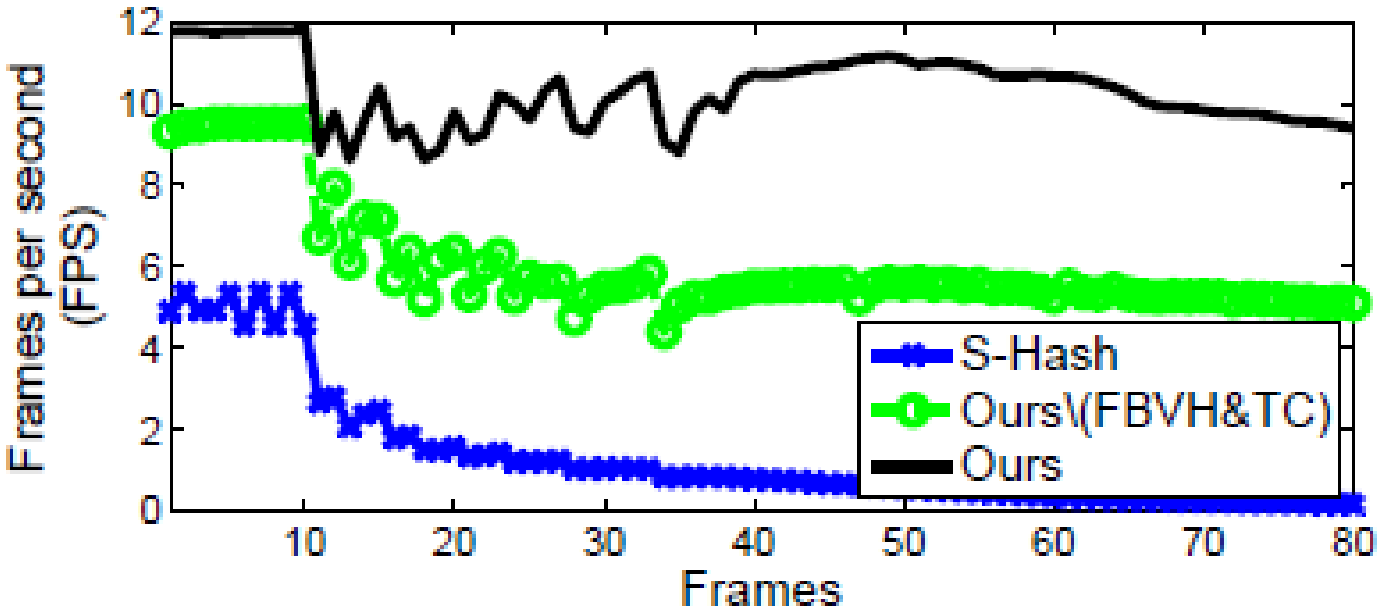
Results with the Exploding Dragon Model



T-CCD: Tang et al. [2008]

KAIST

Results with the Breaking-Wall Model



S-Hash: Optimized spatial hashing (Teschner et al. 03]

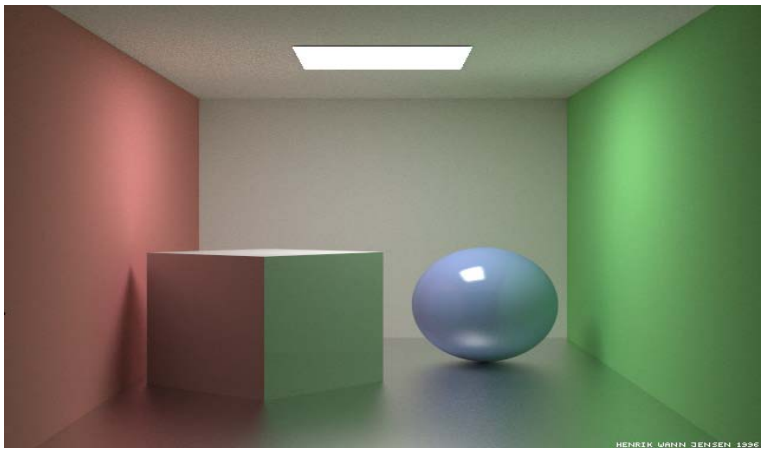
KAIST

Overview

- **HPCCD: Hybrid parallel continuous collision detection**
- **FASTCD: Fracturing-Aware Stable Collision Detection**
- **HCCMeshes: Hierarchical-Culling oriented Compact Meshes**

Hierarchical Culling and Traversal

- **Widely used in many search based algorithms**
  - **Various proximity queries**
  - **Ray tracing**



Path tracing

Photon mapping

[Images are from Jensen's homepage]

KAIST

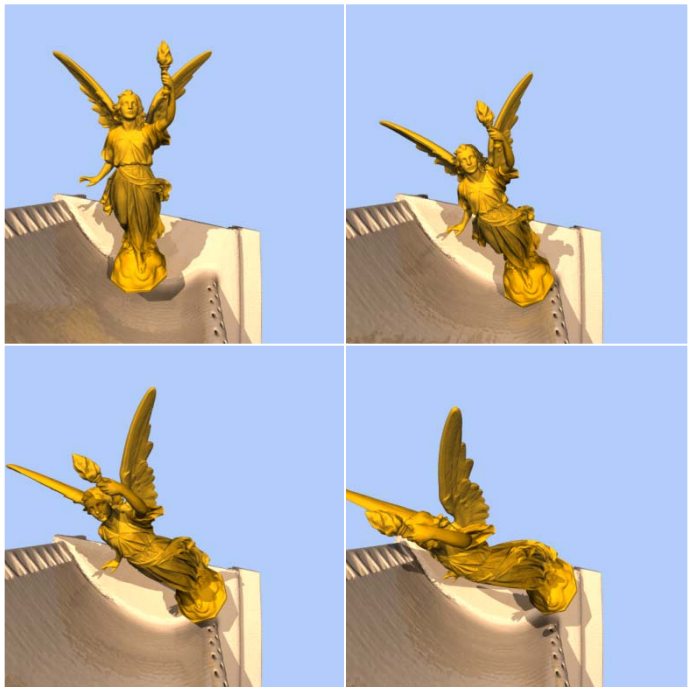
Ray Tracing – Massive Model

St. Matthew

- **128M triangles**
- **256M BVH nodes**
- **14GB data**
- **Size of memory < 4GB**
- **Rendering time : 40 min ↑**

KAIST

Collision Detection– Massive Model



- **30M triangles**
- **60M BVH nodes**
- **3.2GB data**

Lucy & CAD turbine

KAIST

Random-Accessible Compressed Data

- **Compression methods of meshes and hierarchies**
  - **Reduce the memory requirements**
  - **Supports random accesses on meshes and hierarchies**
  - **Can be useful to many different applications**

[Kim et al. Eurographics 2010 (to appear);
Kim et al., TVCG 09;
Yoon and Lindstrom, VIS 07]

→ 1st rank at ACM SIGGRAPH Student Competition

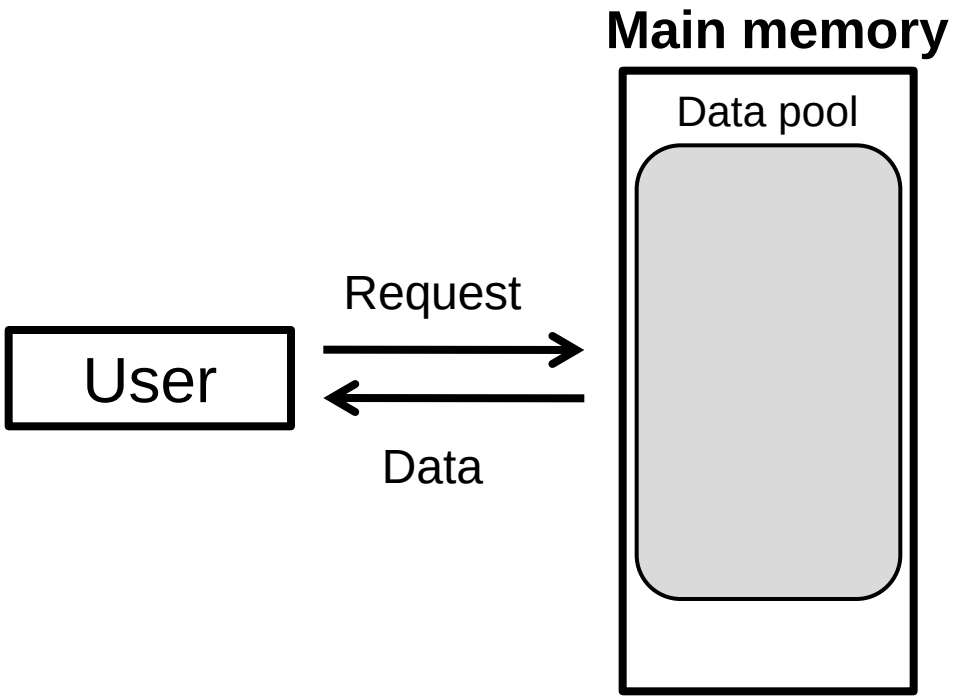
KAIST

Hierarchical-Culling oriented Compact Meshes (HCCMeshes)

- **Consists of two parts:**
  - **i-HCCMeshes (in-core representation)**
  - **o-HCCMeshes (out-of-core representation)**

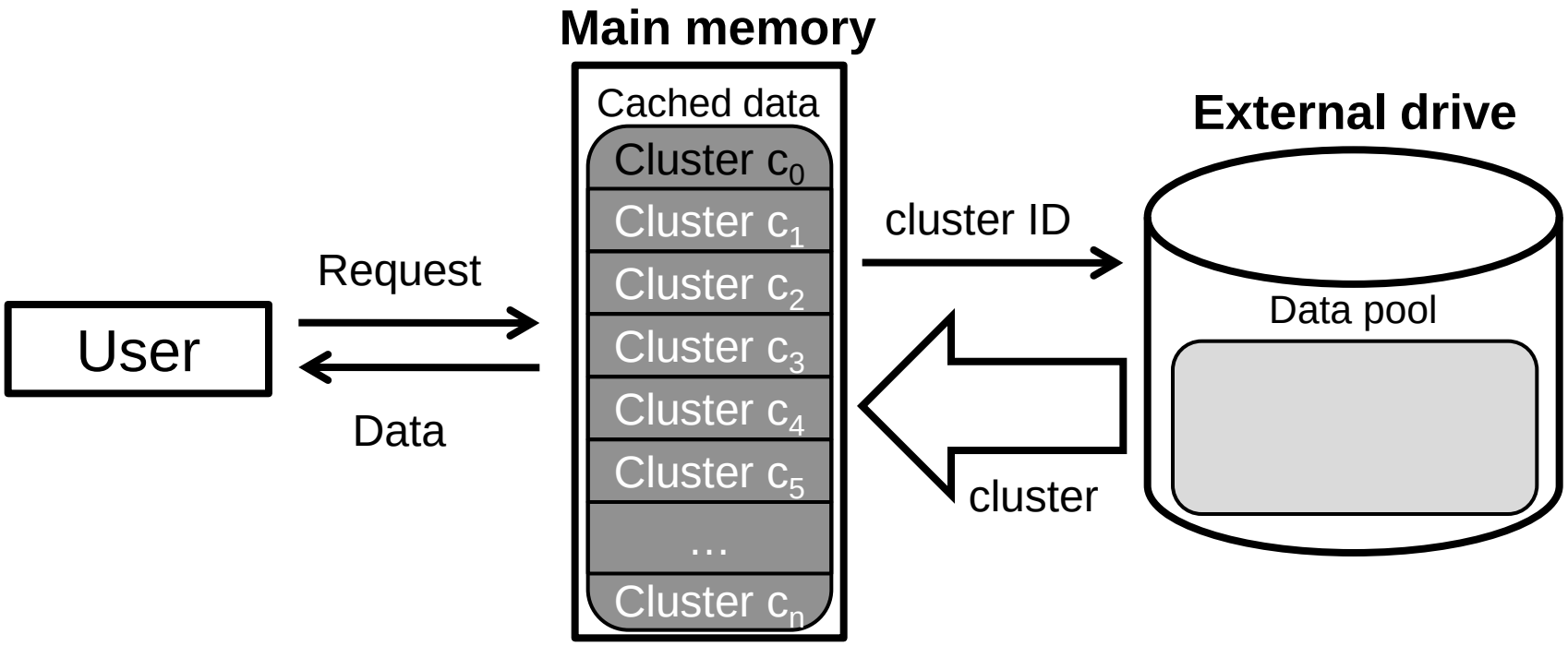
KAIST

Data Access Framework



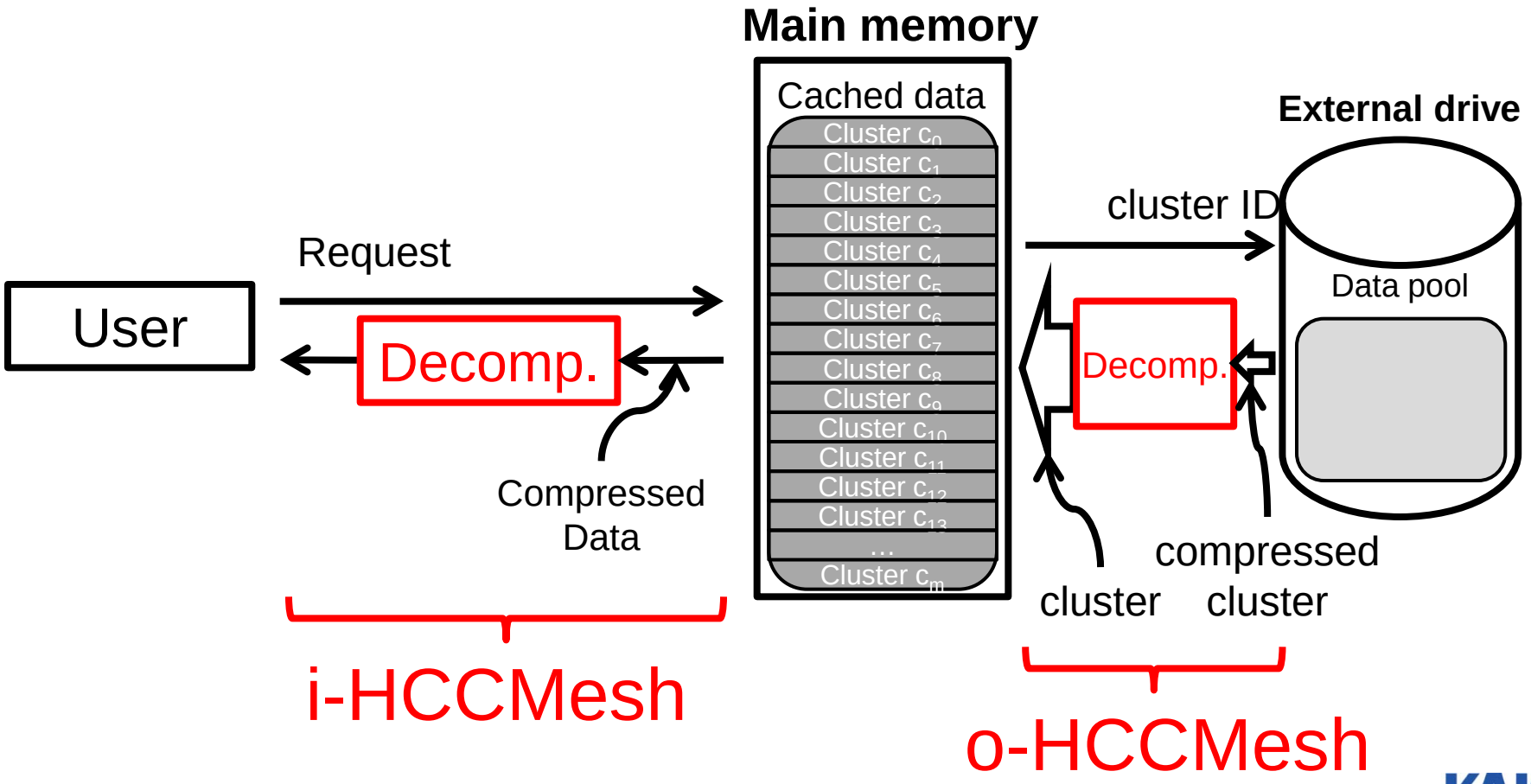
KAIST

Data Access Framework - Out-of-Core Technique



HCCMeshes

Support hierarchical random access!



Main Benefits

- **Use a lower memory space and working set size**
  - **o-HCCMeshes have 20:1 compression ratios**
  - **i-HCCMeshes have 6:1 compression ratios**
- **Improve runtime performance**

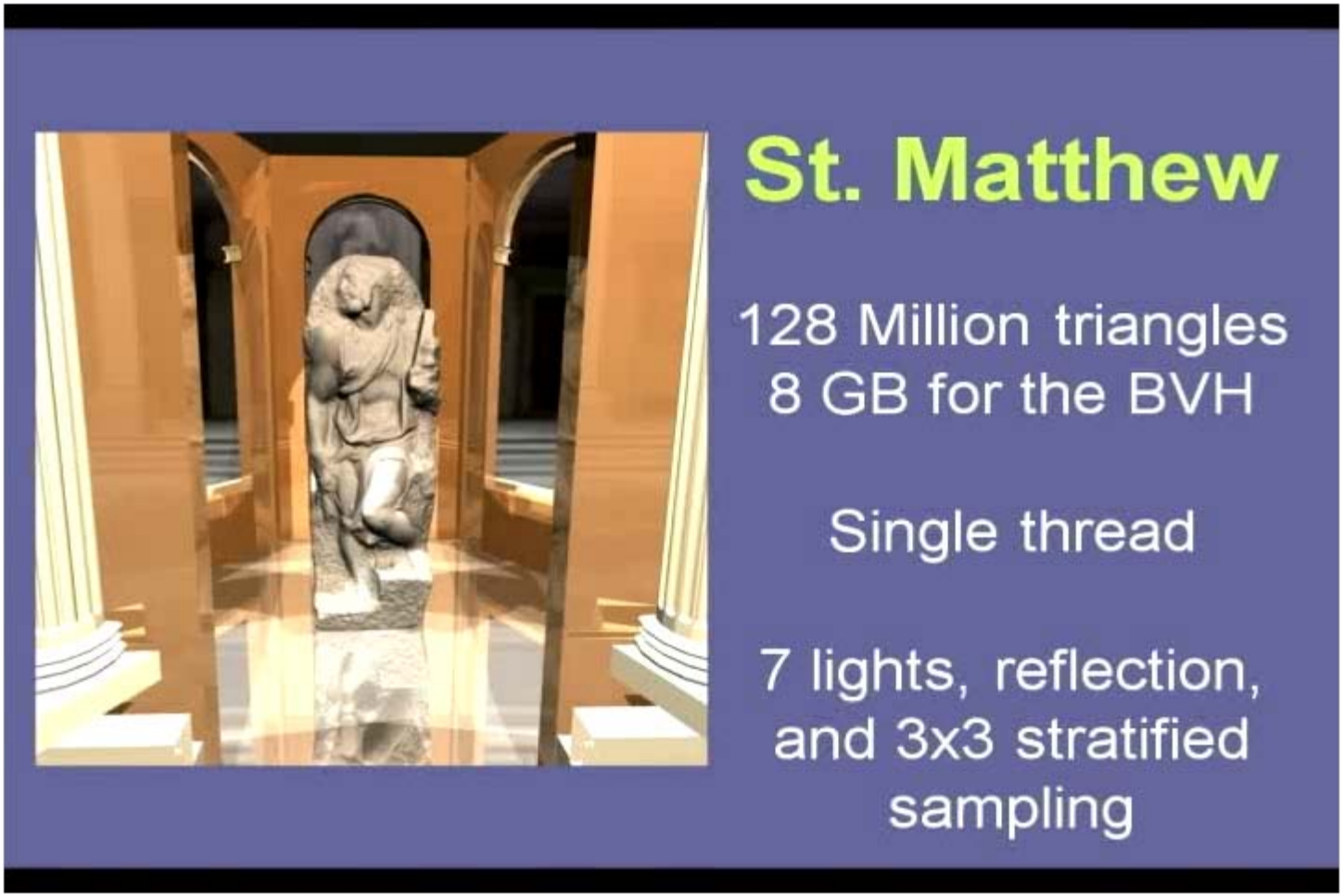
KAIST

Applications

- Whitted-style ray tracing
- LOD-based ray tracing
- Collision detection
- Photon mapping
- Non-photorealistic rendering

KAIST

Results



KAIST

Conclusions

- **Discussed three different techniques for interactive CD among large-scale deforming models**
  - **HPCCD: Hybrid parallel continuous collision detection**
  - **FASTCD: Fracturing-Aware Stable Collision Detection**
  - **HCCMeshes: Hierarchical-Culling oriented Compact Meshes**

KAIST

Acknowledgements

- **Research collaborators**
  - **DukSu Kim, JaePil Heo, TaeJoon Kim, Pio Claudio, BooChang Moon, YongYoung Byun, SeungYong Lee, YongJin Kim, JaeHyuk Heo, John Kim**

- **Funding sources**
  - **Ministry of Knowledge Economy**
  - **Microsoft Research Asia**
  - **KAIST seed grant**
  - **Samsung**
  - **Korea Research Foundation**

KAIST