Recent Advances in Real-Time Collision and Proximity Computations for Games and Simulations

Organizer:

- Sung-eui Yoon and Young J. Kim

Lecturers (alphabetic order):

- Takahiro Harada, Young J. Kim, and Sung-eui Yoon

Abstract:

This course is intended for instructing students and practitioners on recent developments related to collision and proximity computations for interactive games and simulations.

There have been significant advances in various physics-based simulation techniques for movies, interactive games, and virtual environments. Most recent work has been on achieving realistic simulations of rigid, articulated, deforming, and fracturing models. However, many complex and challenging simulations (e.g., fracturing simulation) are not widely used in interactive games because of their computational requirements, although the hardware capability of current CPUs and GPUs has considerably improving. It is well known that one of the main performance bottlenecks in most simulations lies in proximity queries including collision detection, minimum separation distance, and penetration depth computations.

As a result, there has been significant recent research on developing real-time proximity computation algorithms for interactive games and high-quality simulations. Some of recent advanced techniques are able to achieve interactive performance even for most challenging simulations such as fracturing or large-scale cloth simulations. However, these techniques are quite complicated. Moreover, they require in-depth geometric background and sophisticated optimizations on multi-core architectures. These techniques, therefore, have not been easily accessible to students and practitioners who work on real-time simulation methods.

Our objective is to introduce and teach students and practitioners about efficient proximity computation methods and their practical implementations. By doing so, we can expose the attendees to the latest developments, to bridge the gap between the two different fields: proximity computation and simulation. At a broad level, this course will cover the following topics:

- Basic algorithms for various proximity queries including collision detection, minimum separation distances, penetration depth, etc.
- Discrete and continuous algorithms for rigid, articulated, deforming, and fracturing models.
- Parallel algorithms that utilize many cores of CPUs, GPUs, or CPUs/GPUs.
- Applications of various proximity queries in Havoc, a widely used Physics simulation package
- Optimized proximity data structures for many-core architectures including GPU
- Integrating proximity computation algorithms into physically-based simulation systems.

We have three instructors from academia and industry, each of who has significant experiences in designing and implementing different aspects of the aforementioned teaching materials. Since each instructor is a world-class expert in his field, students will receive the best instruction. Moreover, students and practitioners can learn how the industry-leading physics middleware, Havok, benefits from efficient proximity queries, in addition to getting the overall understanding of these libraries.

Relate courses:
To the best of our knowledge, there have been no courses on topics similar to ours, at least within the last three years at SIGGRAPH. However, we will offer a similar course at the SIGGRAPH this year. The main difference between this proposal and the one for SIGGRAPH 2010 is the speaker list; Prof. Dinesh Manocha, Erwin Coumans (Sony Entertainment), and Richard Tonge (Nvidia) do not participate in this course, but Dr. Takahiro Harada (currently at AMD and previously at Havok) will talk about GPU- and Havok-related issues.

Proposed Schedule:

8:30am: Introduction (Yoon)
9:15am: Real-time continuous collision detection and penetration depth computation for rigid and articulated characters (Kim)

10:15: Break

10:15am: Collision detection for deformable and fracturing models (Yoon)
11:00am: Recent parallel techniques for collision detection and its application to physics simulation on the GPU (Harada)

12:00pm: Q&A session (All)
12:15pm: Close

Talk Syllabus:

1. Introduction (Yoon)

Summary: We explain a basic collision detection method based on a bounding volume hierarchy. We explain how to perform inter- and intra-collision detection methods in an efficient manner, followed by discussing the difference between discrete and continuous collision detection. We also introduce other related proximity queries (e.g., minimum separation distance).

Syllabus:

- Basic collision detection methods
- Bounding volume hierarchies and their operations
- Self- and inter-collision detection methods.
- Discrete and continuous detection methods.

2. Real-time Continuous Collision Detection and Penetration Depth Computation for Rigid and Articulated Characters (Kim)

Summary: Maintaining the non-penetration constraint is a crucial problem for realistic simulations of rigid and articulated bodies. Recent researches show that the two seemingly conflicting approaches based on penetration avoidance and recovery turn out to be complementary to each other and are both useful for imposing the non-penetration constraint. In this talk, we present a penetration avoidance technique based on continuous collision detection (CCD) by taking into account of the underlying character motion. We also explain a penetration recovery technique based on penetration depth (PD) computation. Finally, we describe a practical method to implement these collision and penetration queries in a relatively simple manner, which can guarantee interactive performance at run-time, and also demonstrate how these queries can be integrated into real-time physics simulation.

Syllabus:

- Continuous collision detection algorithms for rigid models
- Continuous collision detection algorithms for articulated models
- Penetration depth computation for rigid models
- Application of continuous collision detection and penetration depth to real-time physics simulation

3. Collision Detection for Large-Scale Deforming and Fracturing Models (Yoon)

Summary: Many physically-based simulations used in games and movies are heavily using different types of deforming models. Unlike handling rigid models, deforming models can have self-collisions. Therefore, the performance of collision detection for deforming models has been known to be significantly slower than that for rigid bodies. In this talk, we describe various optimization methods that have been designed recently to improve the performance of discrete and continuous collision detection for deforming models. We also cover most recent techniques designed for fracturing simulations, which are one of the most challenging simulations in terms of collision detection.

Syllabus:

- BVH update and reconstruction methods for deforming and fracturing models
- Efficient self-collision detection method
- Various culling techniques for intra- and self-collisions
- Hybrid parallel algorithms

4. GPU-based collision detection and proximity computations (Harada)

Summary: The trend of processor technology is to increase the number of cores on a processor. On these processors, technologies developed for a single core processor cannot be used to achieve high performance. Thus, developing technologies for these parallel processors are getting more and more important. This session starts with an introduction of parallel computation on the GPU. Later, it presents recent techniques for collision detection and its application to physics simulation on the GPU.

Syllabus:

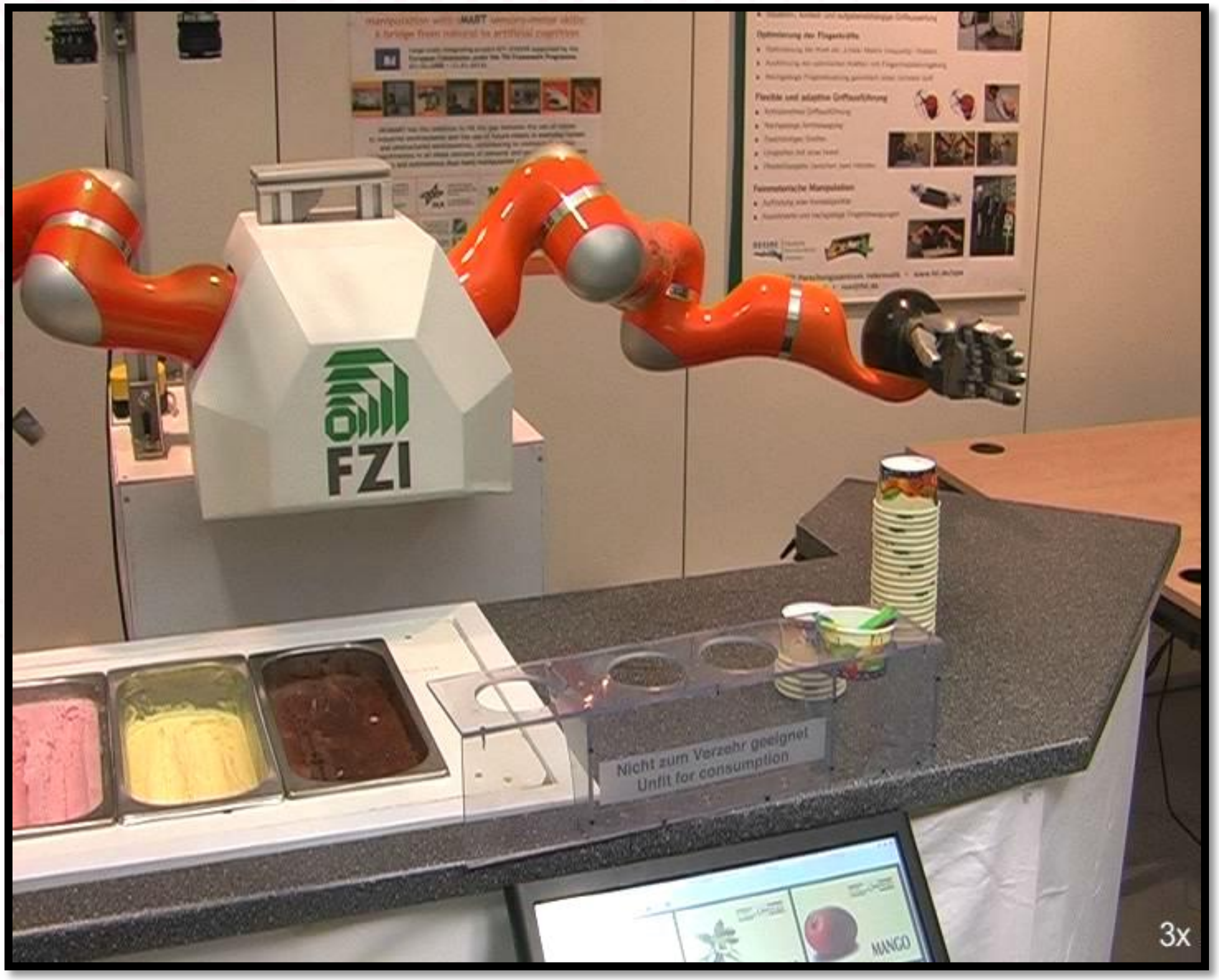
- GPU-friendly rigid and paticle-based simulations
- GPU-friendly collision detection methods
- Techniques using multiple GPUs
- Parallel contact handling

Real-time Continuous Collision Detection and Penetration Depth Computation

Young J. Kim

http://graphics.ewha.ac.kr
Ewha Womans University

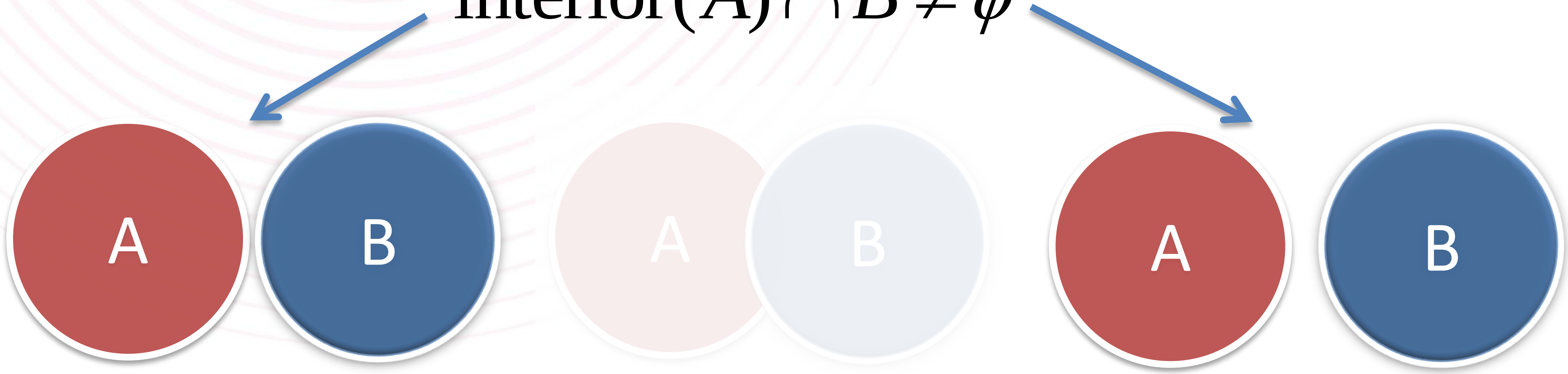
Autonomous Ice Serving Robot
(with Zhixing Xue at FZI)



Non-penetration Constraint

- No overlap between geometric objects

$$\text{interior}(A) \cap B \neq \phi$$



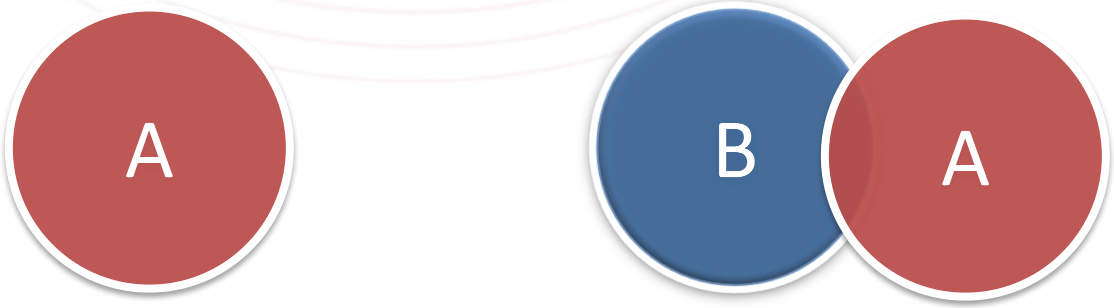
- Crucial for mimicking the physical presence

Earlier Research

- Focused on checking for whether there is any overlap between A and B, *fixed in space*
  - Tons of papers published in the area of *collision detection*
  - Well-studied and matured technology
- Not clear how to resolve such overlap

Recent Research Trends

- Avoid inter-penetration
  - Continuous collision detection (CCD)
- Allow inter-penetration but backtrack
  - Penetration depth (PD)

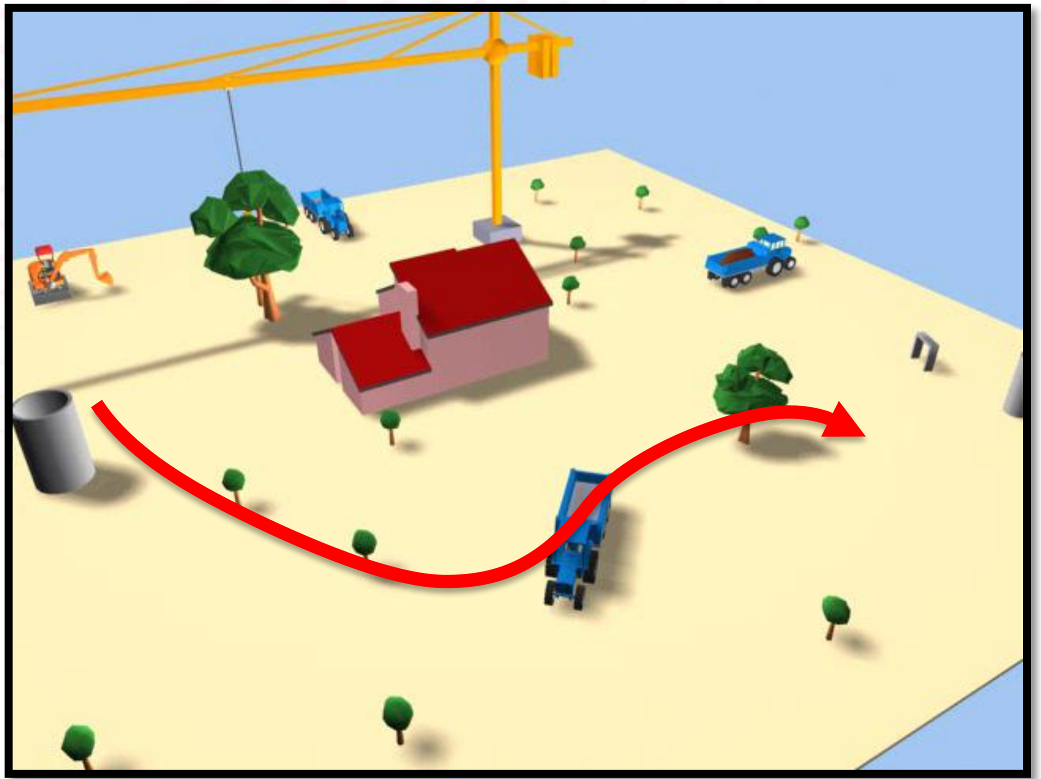


Applications of Continuous CD

- Rigid body dynamics
  - Find the time of contact (ToC) to apply forces

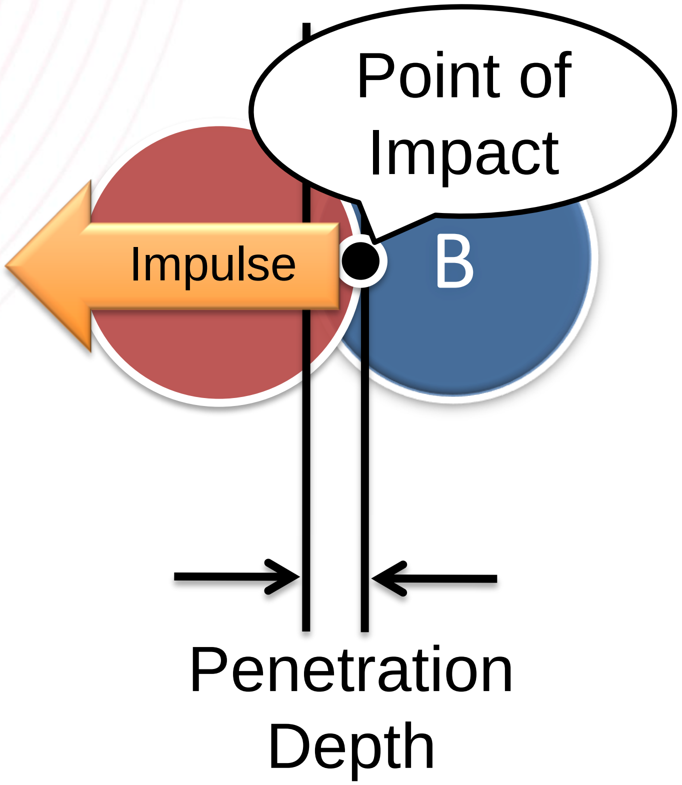
Applications of Continuous CD

- Motion planning
  - Check whether a path is collision-free



Applications of Penetration Depth

- Dynamics simulation
  - Penalty-based
  - Impulse-based



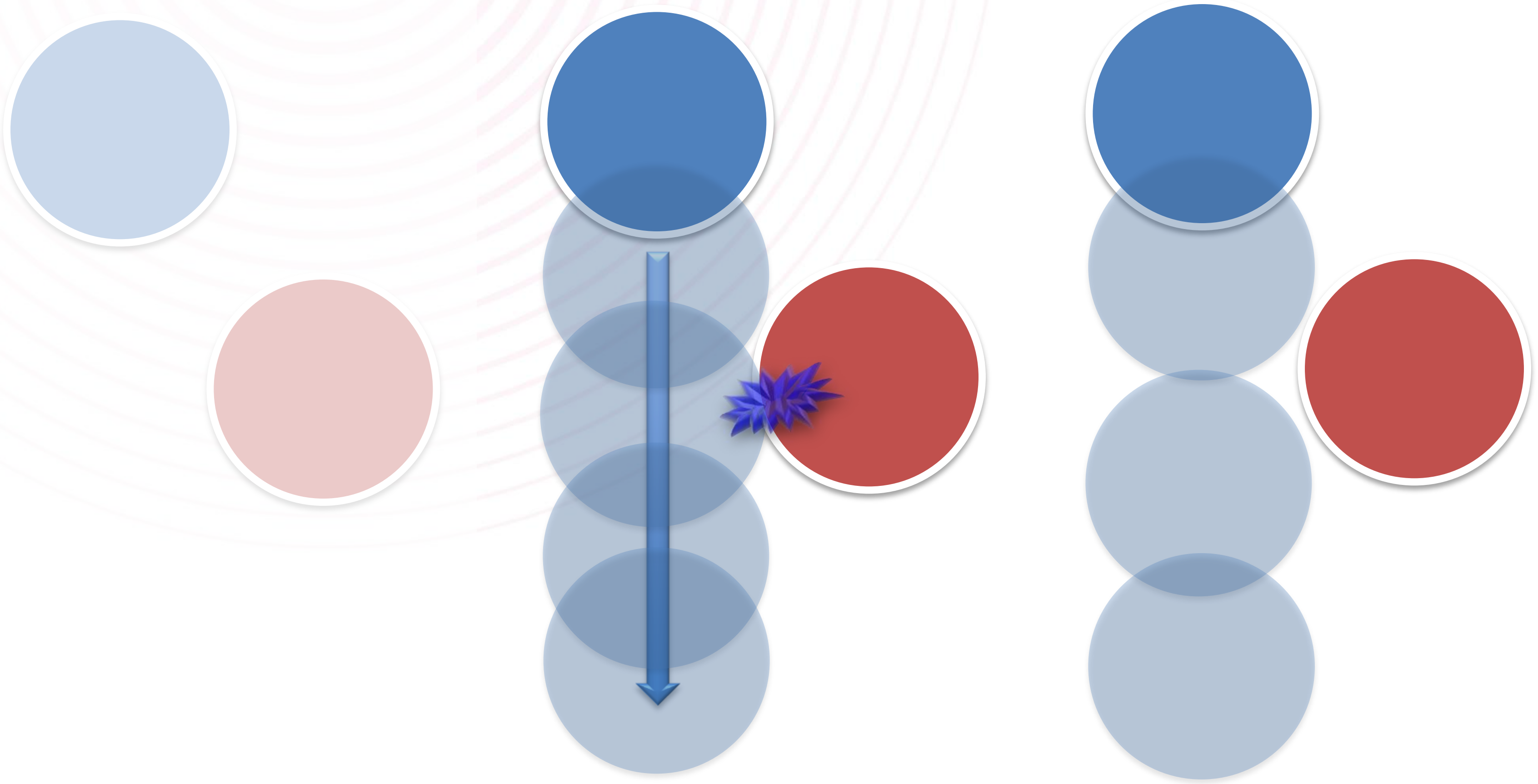
Goal

- Recent research results
  - CCD (rigid, polygon-soups, articulated)
  - PD (pointwise, translational)

- Applications
  - Real-time rigid body dynamics
  - Robotic grasping
  - CAD disassembly

CONTINUOUS COLLISION DETECTION

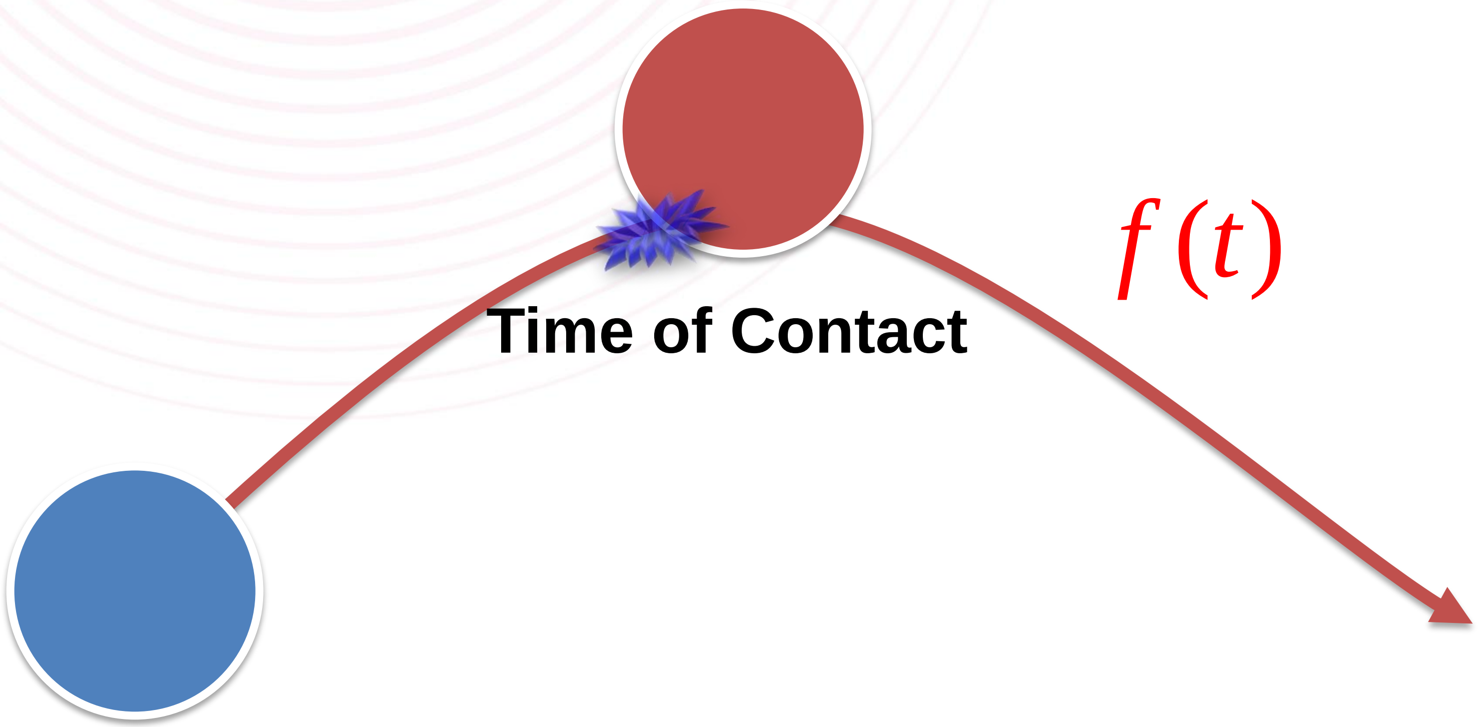
Discrete Collision Detection



Collision Missing

Continuous Collision Detection

- Motion trajectory ***f(t)*** is known in advance

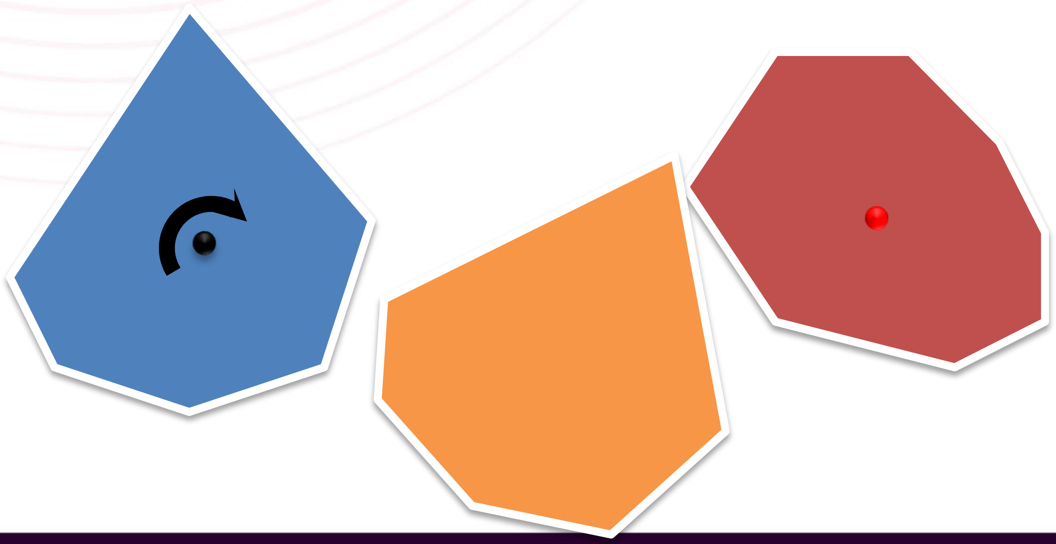


Previous Work on CCD

- Algebraic solution -[Canny86], [Redon00], [Kim03], [Choi06]
- Swept volume -[Abdel-Malek02],[Hubbard93], [Redon04a,b]
- Bisection -[Redon02], [Schwarzer02]
- Kinetic data structures -[Kim98], [Kirkpatrick00], [Agarwal01]
- Minkowski sum -[Bergen04]
- Conservative advancement
  - [Mirtich96], [Mirtich00],[Coumans 2006],[Zhang06], [Tang09]

Conservative Advancement (CA)

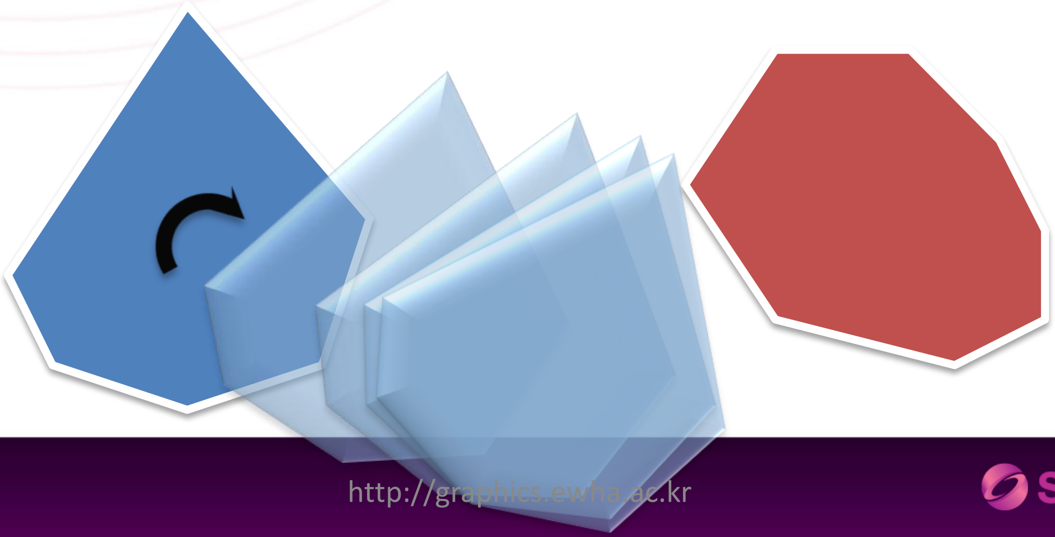
- Assume objects are *convex*
- Find the 1st time of contact (ToC) of a moving object



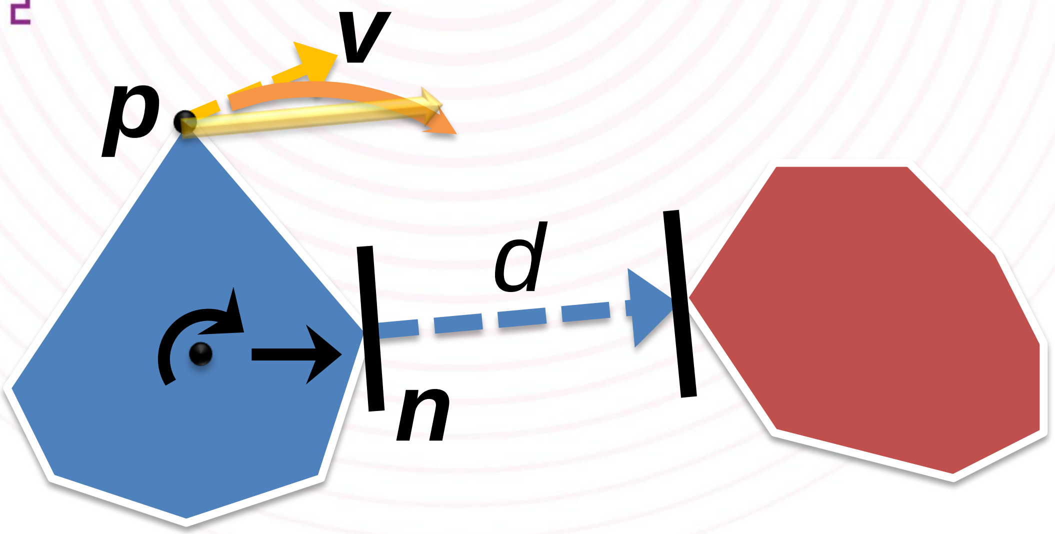
Conservative Advancement (CA)

1. Find a step size Δt_i to conservatively advance the object until collision occurs
2. Repeat until inter-distance < ε

$$ToC = \Delta t_1 + \Delta t_2 + \Delta t_3 + \Delta t_4$$



Calculating *Δt* in CA



d, ***n***: closest distance, direction vector

v: velocity

$$\int_0^{\Delta t} |v(t)\cdot n(t)|\,dt \le \int_0^{\Delta t} \boxed{\max |v(t)\cdot n(t)|}\,dt$$

$$= \mu$$

Calculating Δt in CA

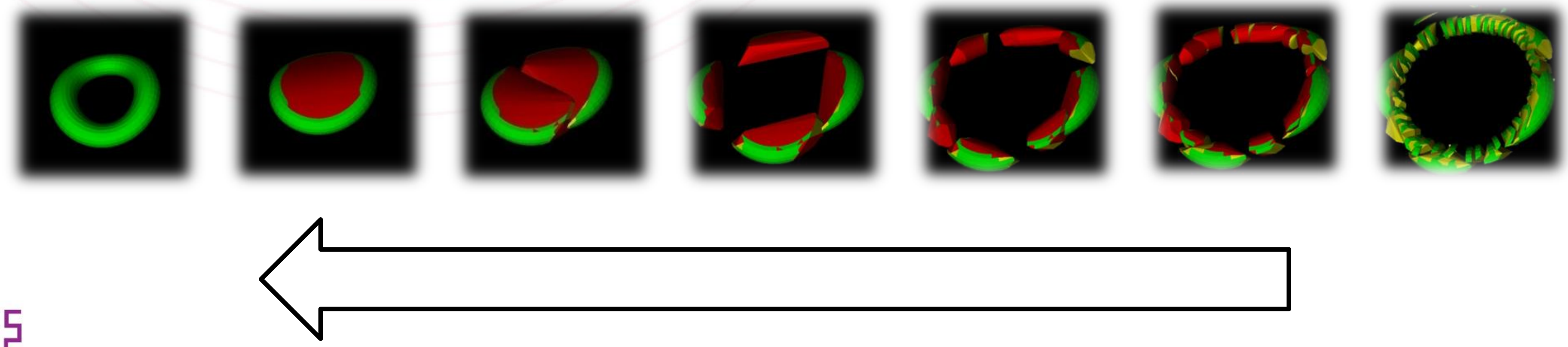
$$\mu \Delta t \le d \qquad \therefore \Delta t \le \frac{d}{\mu}$$

$$\int_0^{\Delta t} |v(t) \bullet n(t)|\, dt \le \int_0^{\Delta t} \overset{=\mu}{\boxed{\max(|v(t) \bullet n(t)|)}}\, dt \le d$$

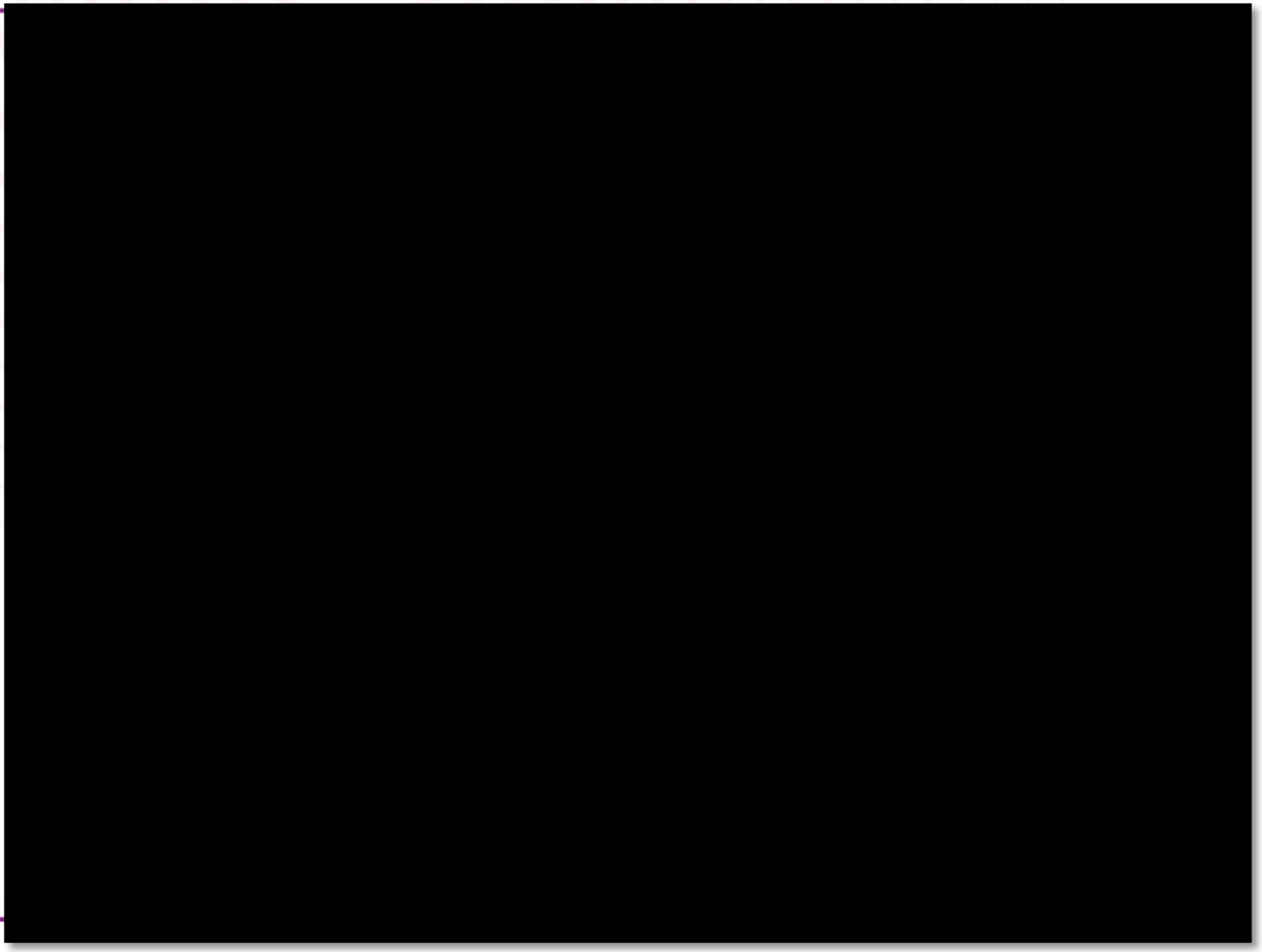
Extension to Non-convex

[Zhang et. al PG 06]

- Use of convex decomposition
- Build a hierarchy of decomposed convex pieces and perform CA *hierarchically*



Santa vs. Thin Board

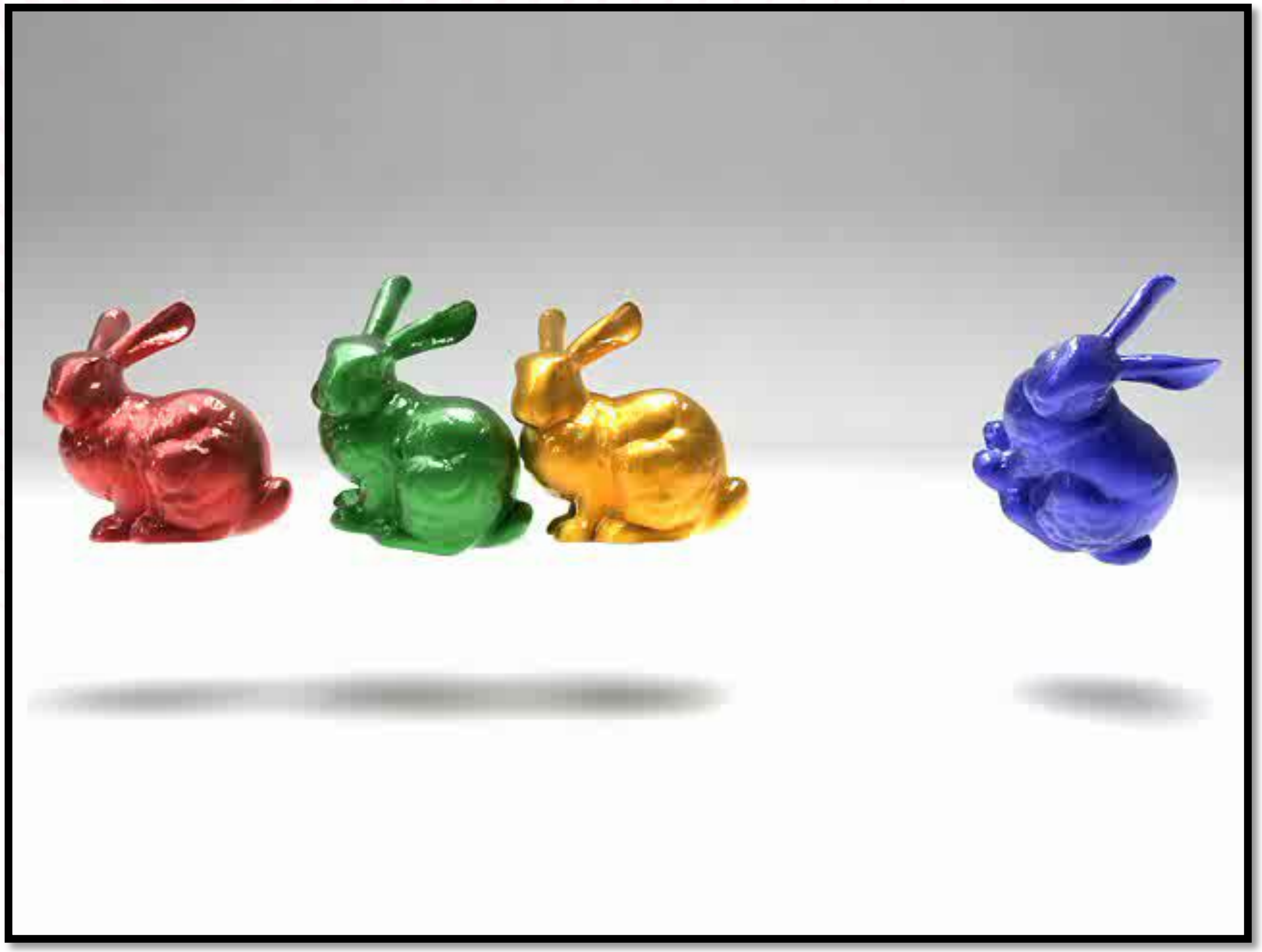


37K triangles

51,546 FPS

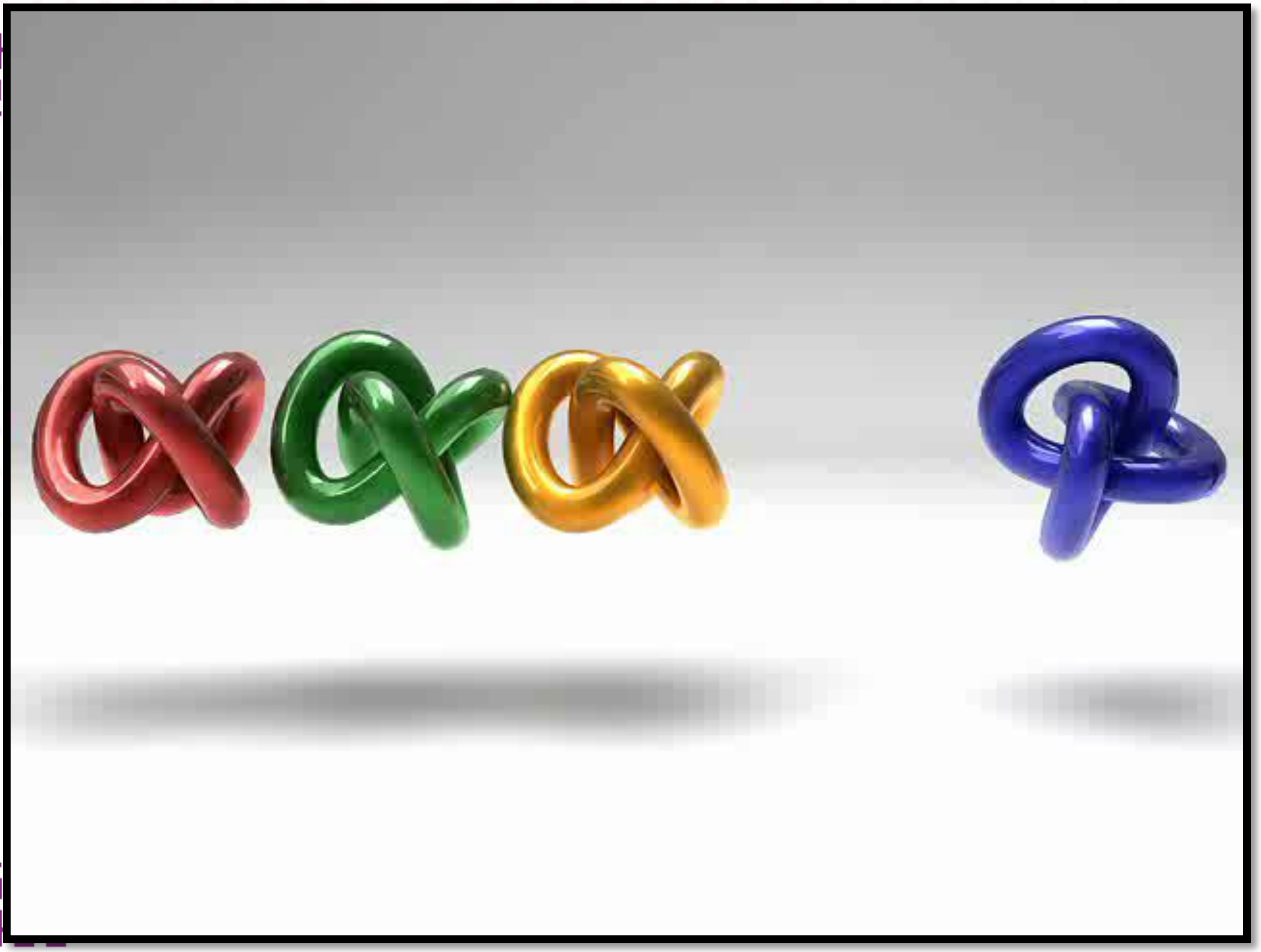
# of iterations
3.68

Bunny vs. Bunny



70K triangles/bunny 110 FPS # of iterations 4.7

Torusknot vs. Torusknot

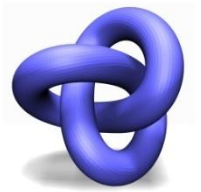


2.8K
1067 FPS

# of iterations 4.49

11K
400 FPS

# of iterations 4.49



34K
186 FPS

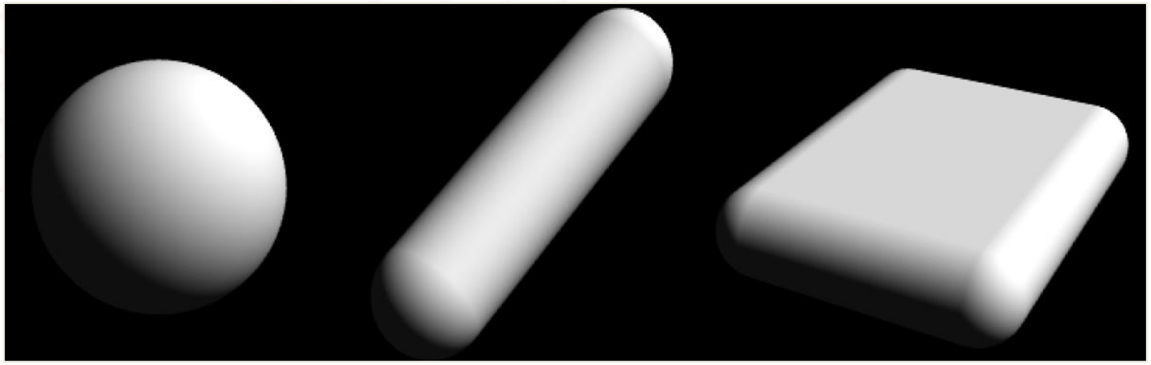
# of iterations 4.46

Extension to Polygon-Soups
[Tang et. al IEEE ICRA 09]

- Construct the bounding volume hierarchy of polygons
- Motion bound calculation $\Delta t \leq \dfrac{d}{\mu}$
  - Bounding volume
  - Triangles

Extension to Polygon-Soups
[Tang et. al IEEE ICRA 09]

- We use swept sphere volumes


[Larsen et. al IEEE ICRA 1999]

Motion Bound Calculation

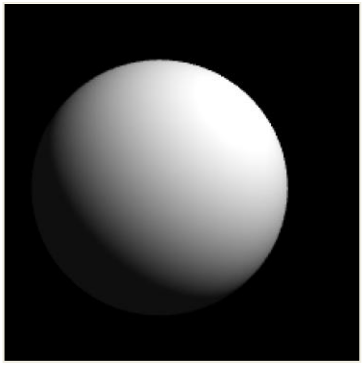
- Motion bound of SSV (e.g. PSS)

$$\mu \le \left\|\boldsymbol{\omega} \times \mathbf{n}\right\| \left(\left\| \mathbf{c}_1^{\perp} \right\| + \mathbf{r} \right)$$

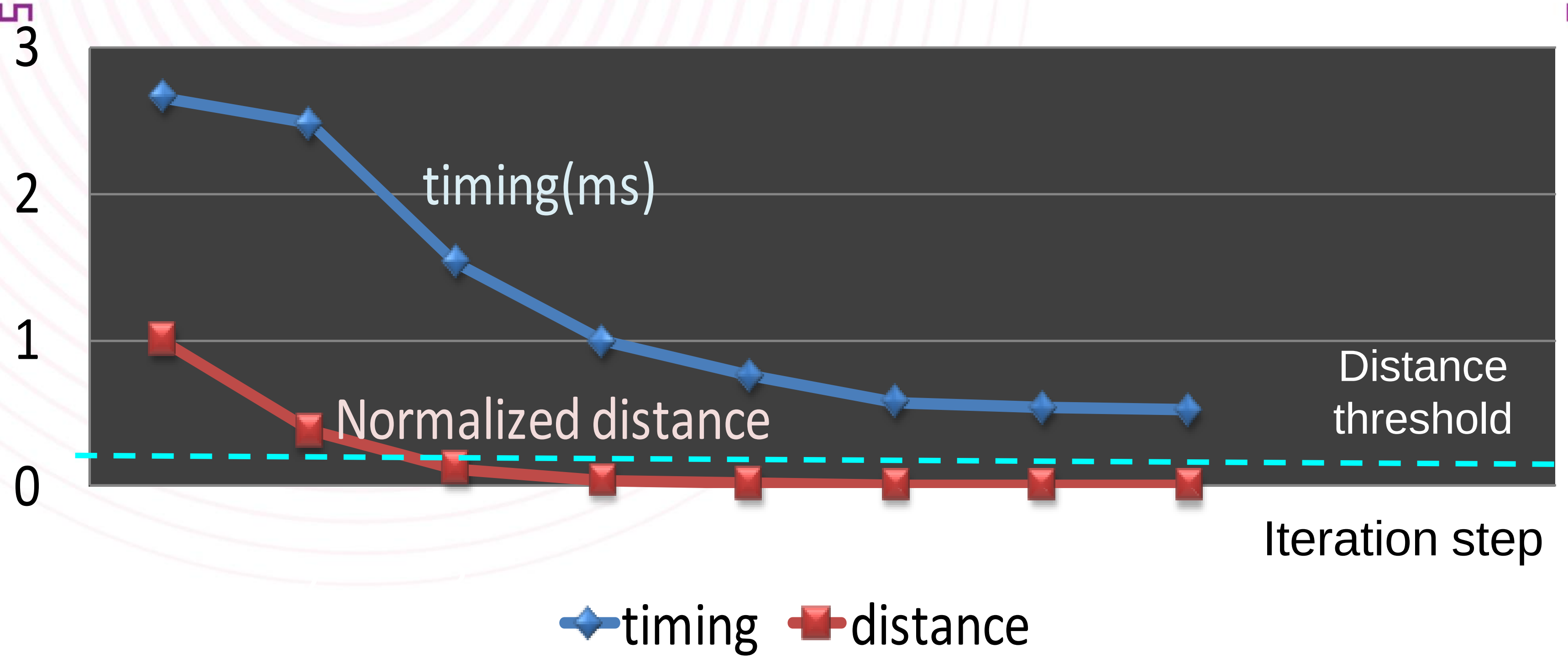
$\boldsymbol{\omega}$: rotational velocity

$\mathbf{n}$: closest direction

$\mathbf{r}$: radius of PSS

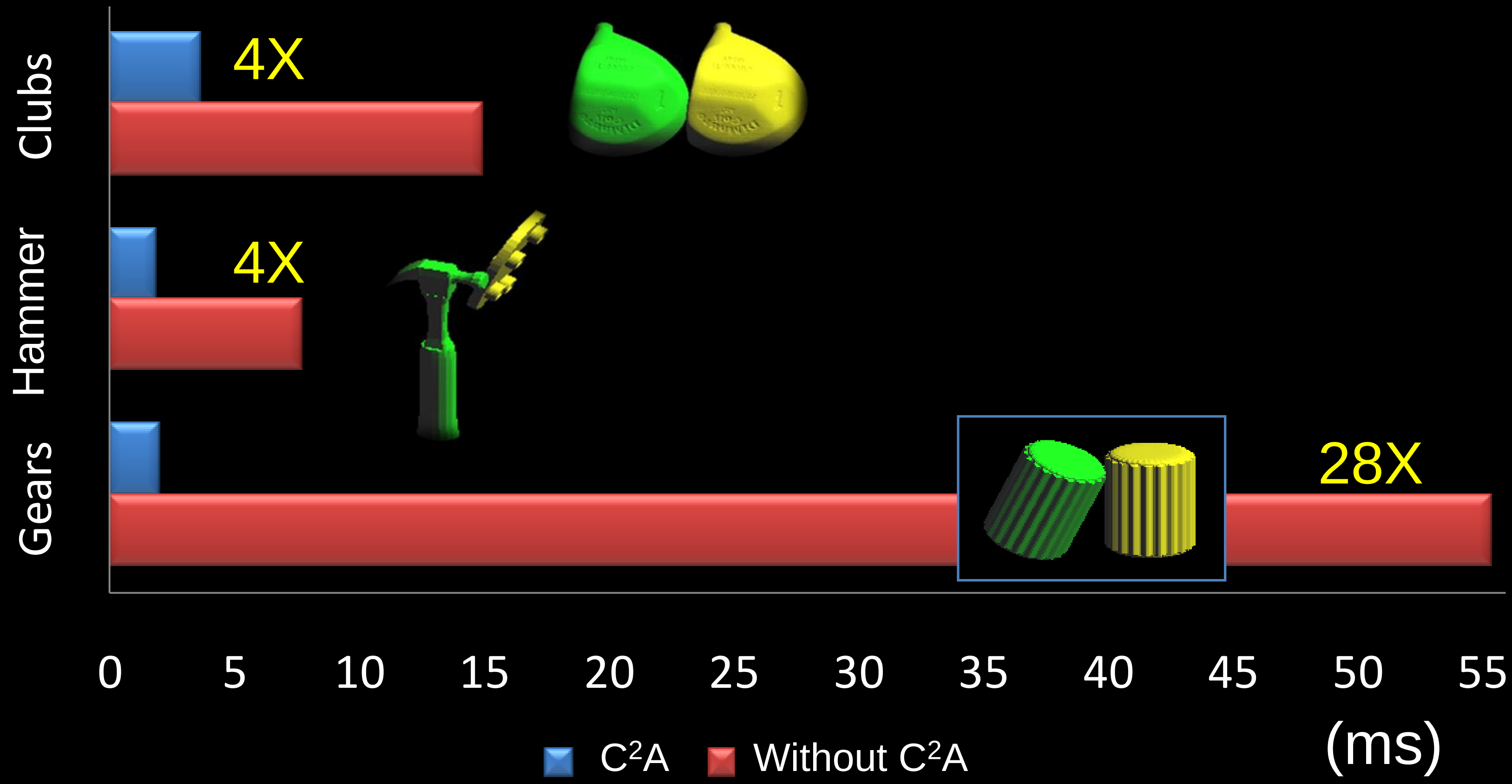


Controlling the CA Iterations

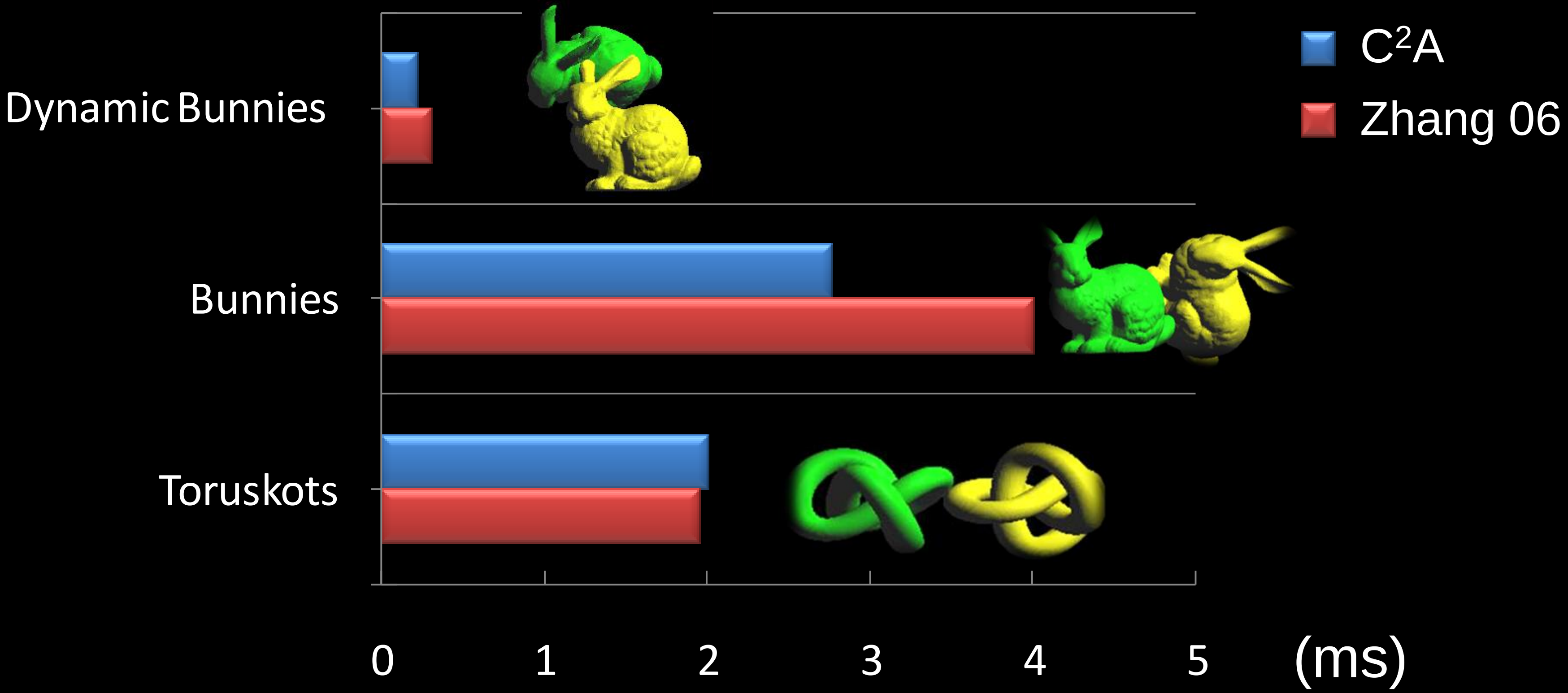


1. Compute approximate distance in the beginning
2. Compute exact distance toward the end

Results - Timings



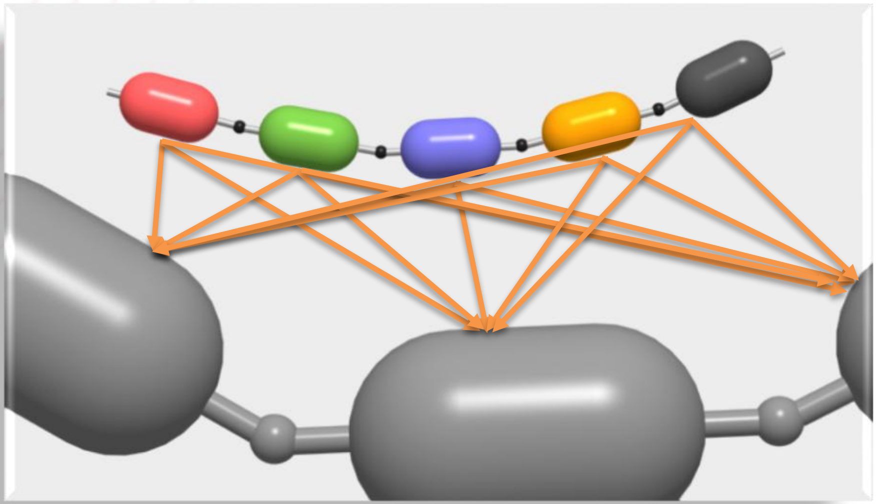
Comparisons against [Zhang 06]



- [Zhang 06] can handle only manifold surfaces

Extension to Articulated Models
[Zhang et. al SIGGRAPH 07]

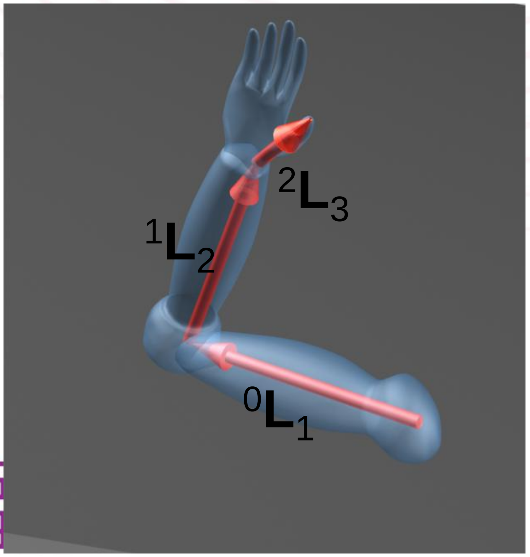
- Treat each link as a rigid body
- Apply CA to each link independently
- Taking the minimum of CA results

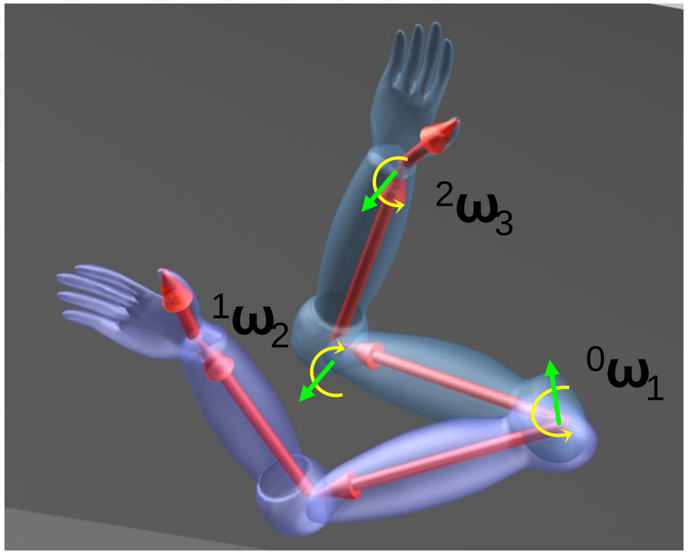


Motion Bound Calculation $\Delta t \le \frac{d}{\mu}$

μ

$$\le \left| {}^{0}\boldsymbol{\upsilon}_1 \cdot \mathbf{n} \right| + \left| \mathbf{n} \times {}^{0}\boldsymbol{\omega}_1 \right| \left| {}^{0}\mathbf{L}_1 \right|_{\mu} + \sum_{j=2}^{i-1} \left(\left| {}^{j-1}\boldsymbol{\upsilon}_j \right| + \left(\left| \mathbf{n} \times {}^{0}\boldsymbol{\omega}_1 \right| + \sum_{k=2}^{j} \left| {}^{k-1}\boldsymbol{\omega}_k \right| \right) \left| {}^{j-1}\mathbf{L}_j \right|_{\mu} \right)$$





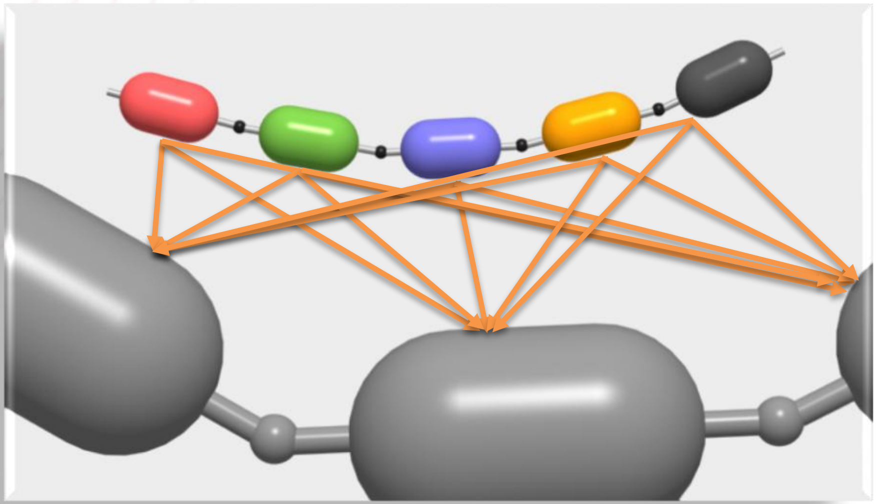
${}^{0}v_i$: velocity of link i

${}^{i-1}\omega_i$: rotational velocity of link i w.r.t. link i-1

${}^{i-1}L_i$: difference vector btwn the links

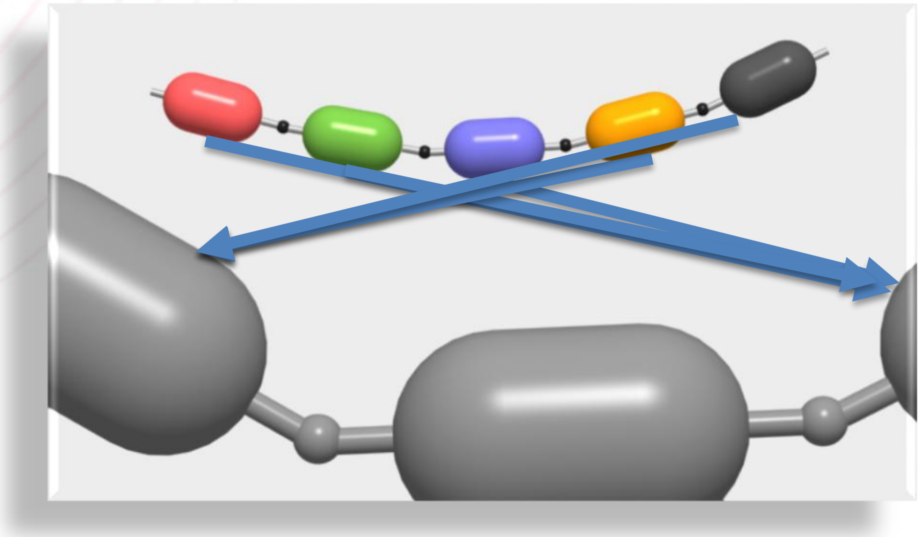
Problems in Straightforward CA

- Problems
  - O(n^2) checking between individual links



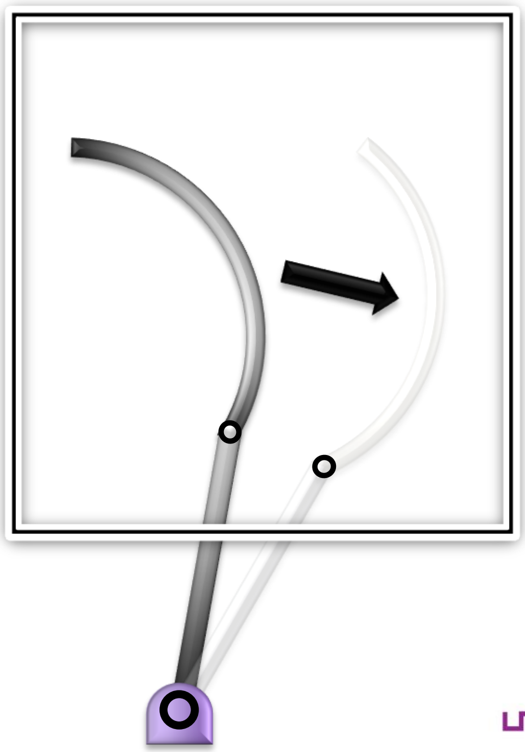
Spatial Culling

- Cull the link pairs that are far apart
- Use bounding volume-based collision-culling

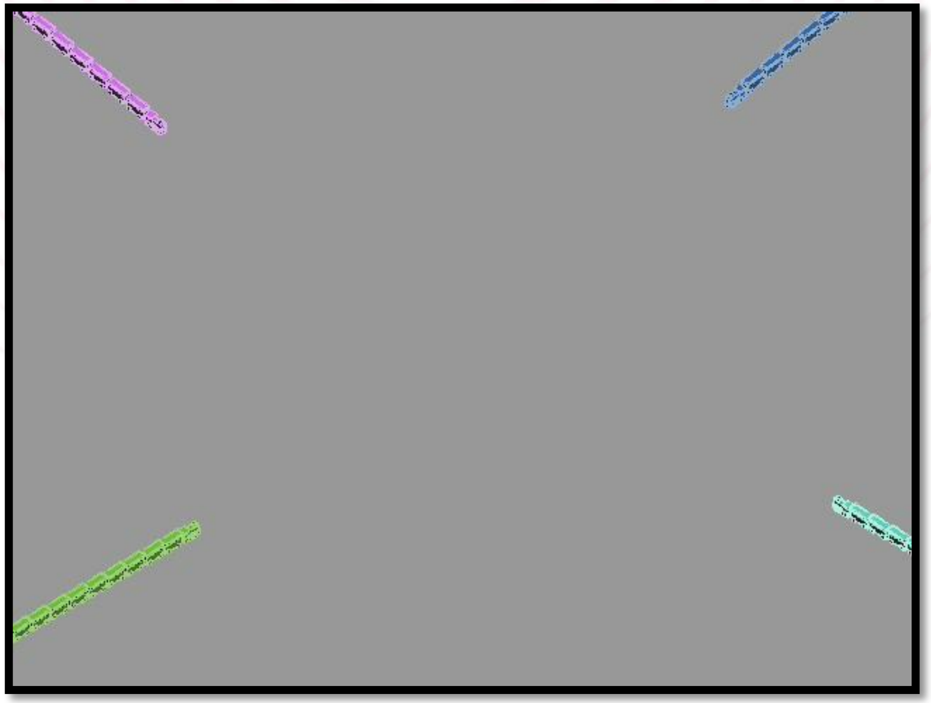


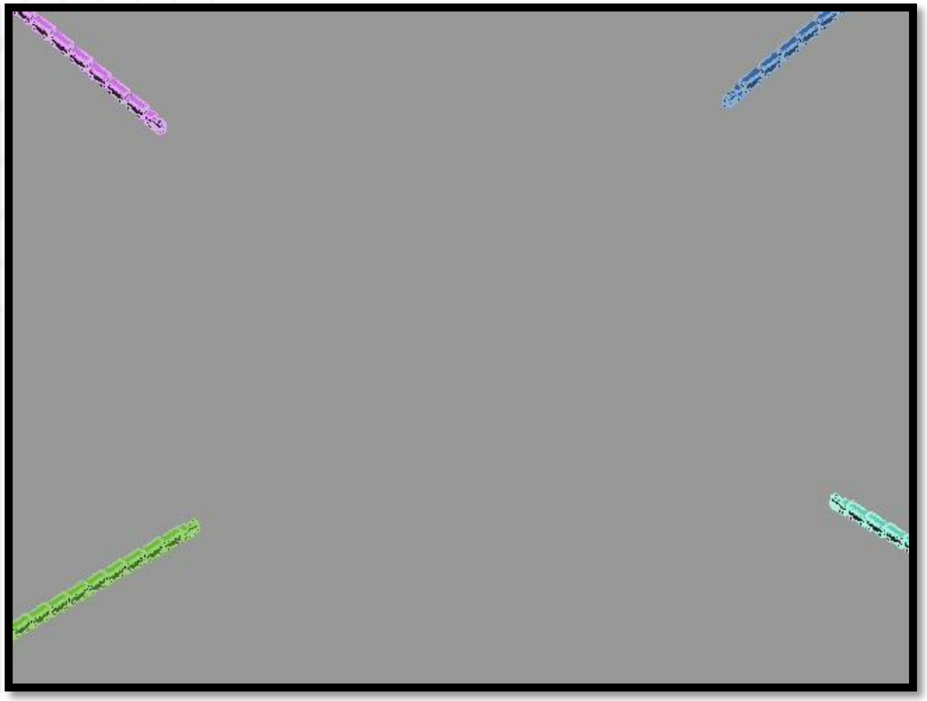
Spatial Culling using Dynamic AABB

- Goal
  - Compute an axis-aligned bounding box (AABB) that bounds the motion of a moving link



Bounding Volume Culling


Interval Arithmetic


Taylor Models

Locomotion Benchmark

- **CCD performance**
  - **1.22 msec**
- Mannequin
  - 15 links, 20K tri
- Obstacles
  - 101K tri
- Locomotion SW
  - Footstep™

Exercise Benchmark

- Mannequin
  - 15 links, 20K triangles
- **Self-CCD performance**
  - **0.38 msec**

Motion Planning Benchmark 1

- Excavator
  - 52 links, 19K tri
- Obstacles
  - 0.4M tri

- **CCD performance**
  - **100~700 msec**

Motion Planning Benchmark 2

- Tower crane
  - 14 links, 1288 tri
- **CCD performance**
  - **5.66~15.1 msec**

Articulated Body Dynamics Benchmark

- Four trains
  - 10 links, 23K tri (each)

- **CCD performance**
  - **535 msec**

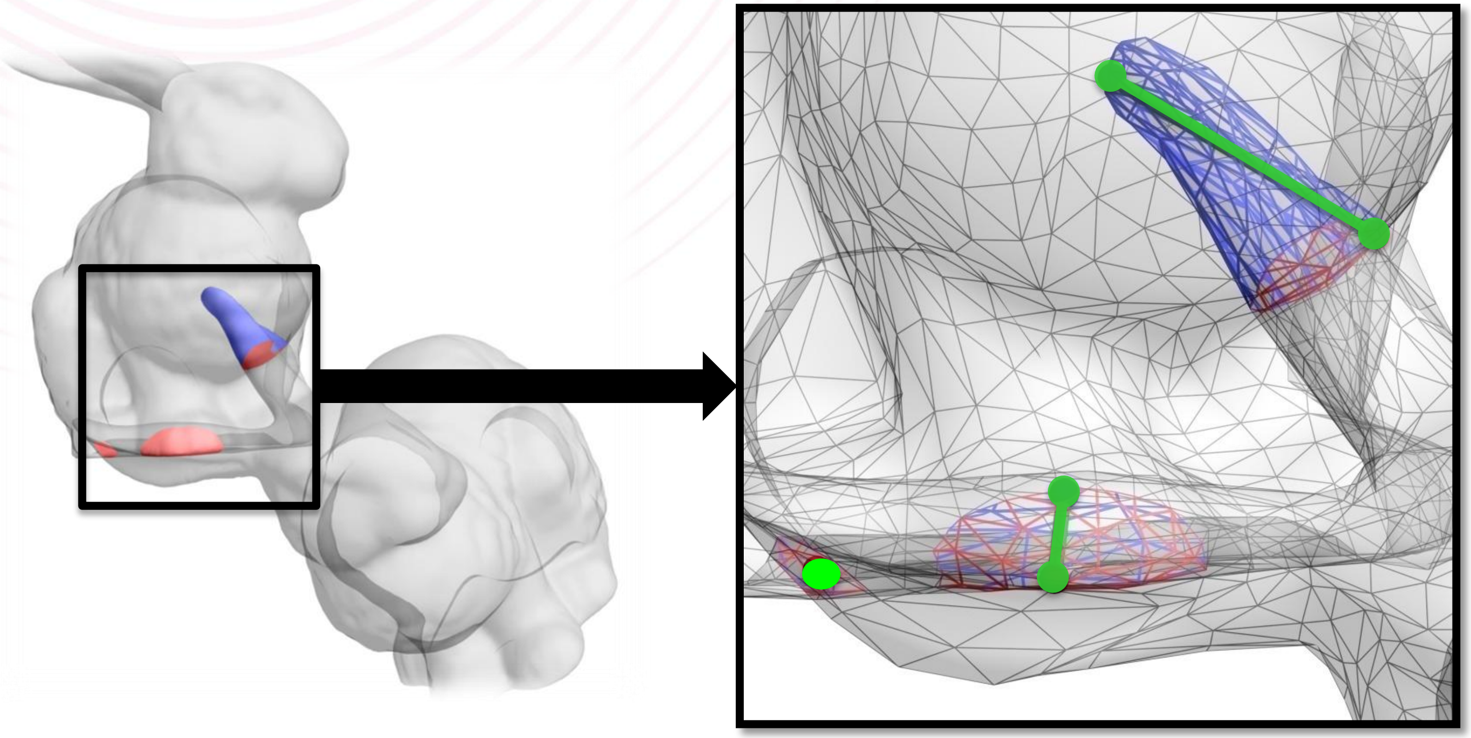
Software Implementations

- Source codes are available
  - http://graphics.ewha.ac.kr/FAST (2-manifold)
  - http://graphics.ewha.ac.kr/C2A (polygon-soups)
  - http://graphics.ewha.ac.kr/CATCH (articulated)

PENETRATION DEPTH COMPUTATION

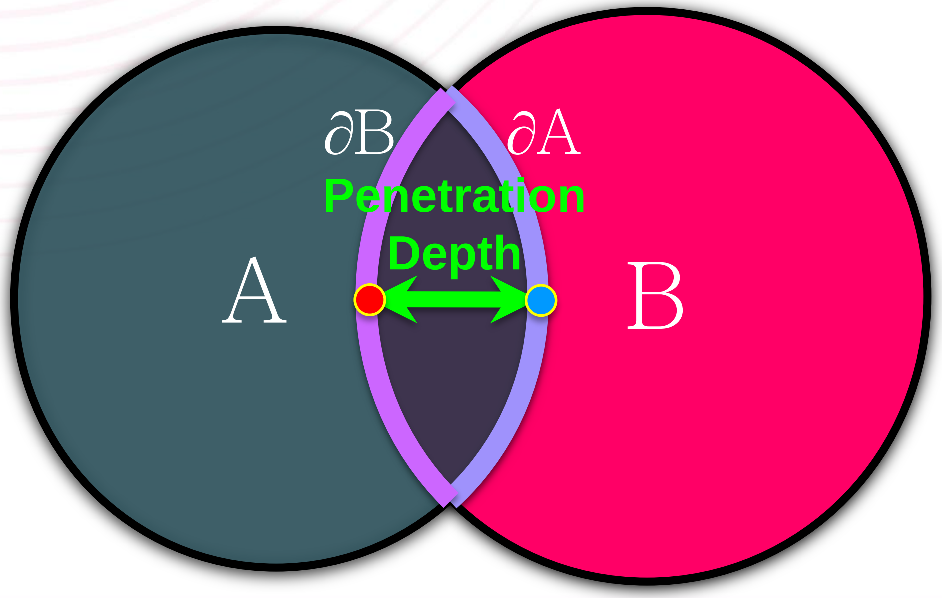
Pointwise Penetration Depth
[Tang et. al SIGGRAPH 09]

- Defined as deepest interpenetrating points



Pointwise Penetration Depth

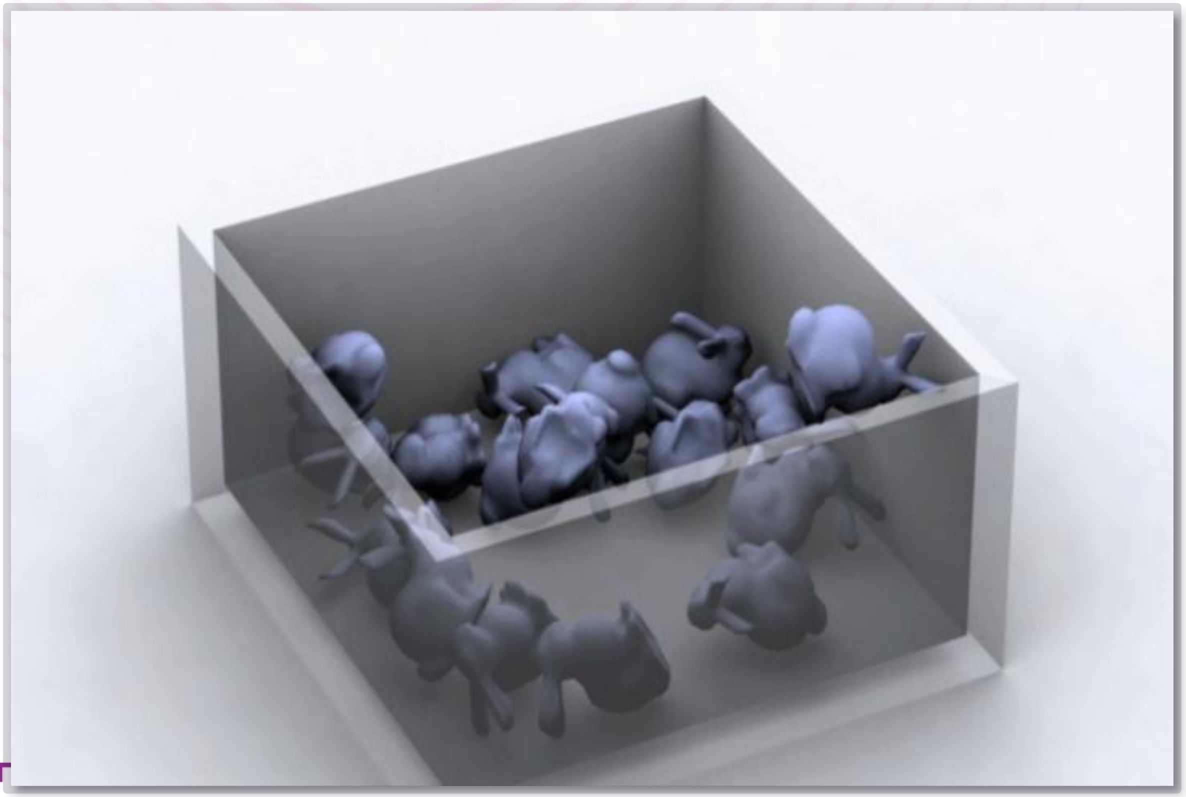
1. Find intersection surfaces $\partial\mathcal{A}$ and $\partial\mathcal{B}$
2. Penetration depth = $H(\partial\mathcal{A}, \partial\mathcal{B})$



Pointwise Penetration Depth

Demo (40K Bunny vs 40K Bunny)

Benchmark: Pointwise PD



Model complexity

– 50K tri

Avg. Performance

– 3.88ms/pair

Benchmark: Pointwise PD

SIGGRAPH2009

Model complexity

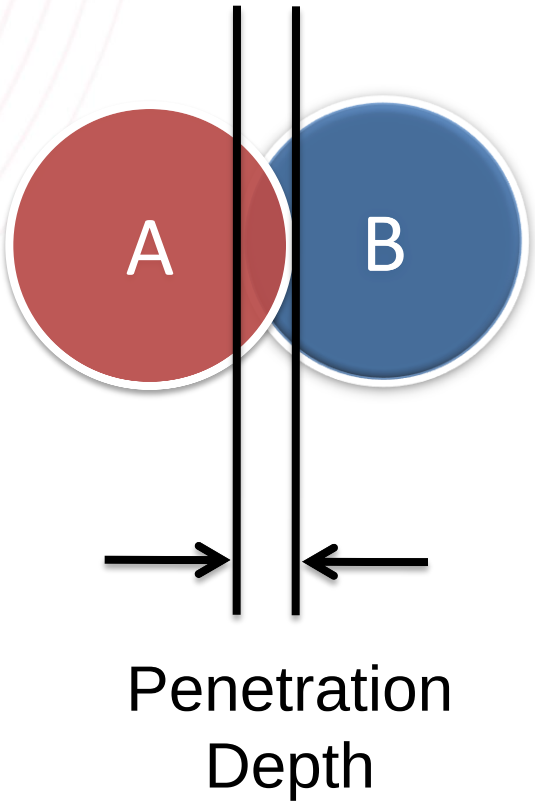
– 3.5K tri

Avg. performance

– 0.95ms/pair

Penetration Depth
[Dobkin 93]

- Minimum translational distance to separate overlapping objects



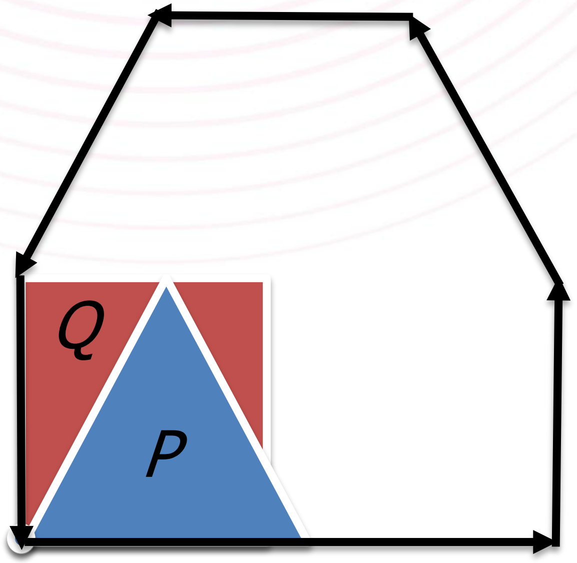
Previous Work on PD

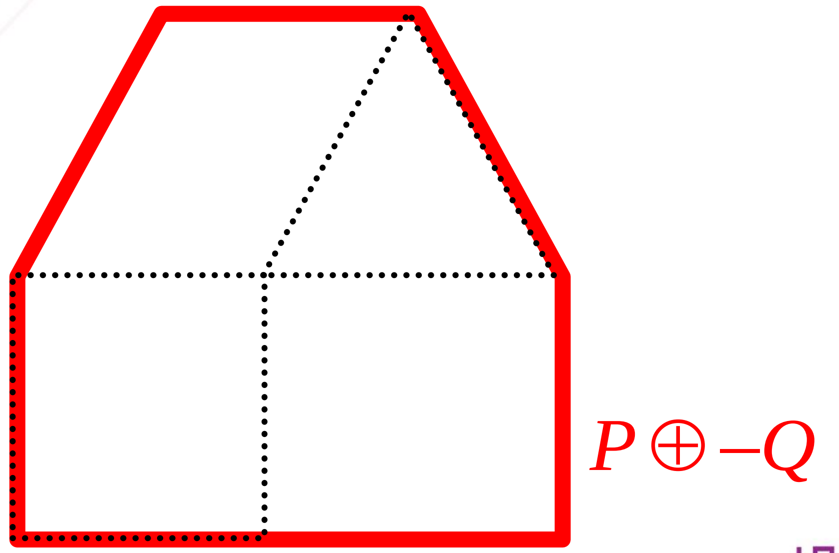
- Convex polytopes -[Cameron and Culley86], [Dobkin93], [Agarwal00], [Bergen01], [Kim04]
- Non-convex polyhedra -[Kim02],[Redon and Lin06], [Lien08a,b], [Hachenberger09]
- Distance fields -[Fisher and Lin01], [Hoff02], [Sud06]
- Pointwise PD -[Tang09]
- Generalized PD – [Ong and Gilbert96], [Ong96], [Zhang07]
- Volumetric PD - [Wellner and Zachmann09]

Minkowski Sum

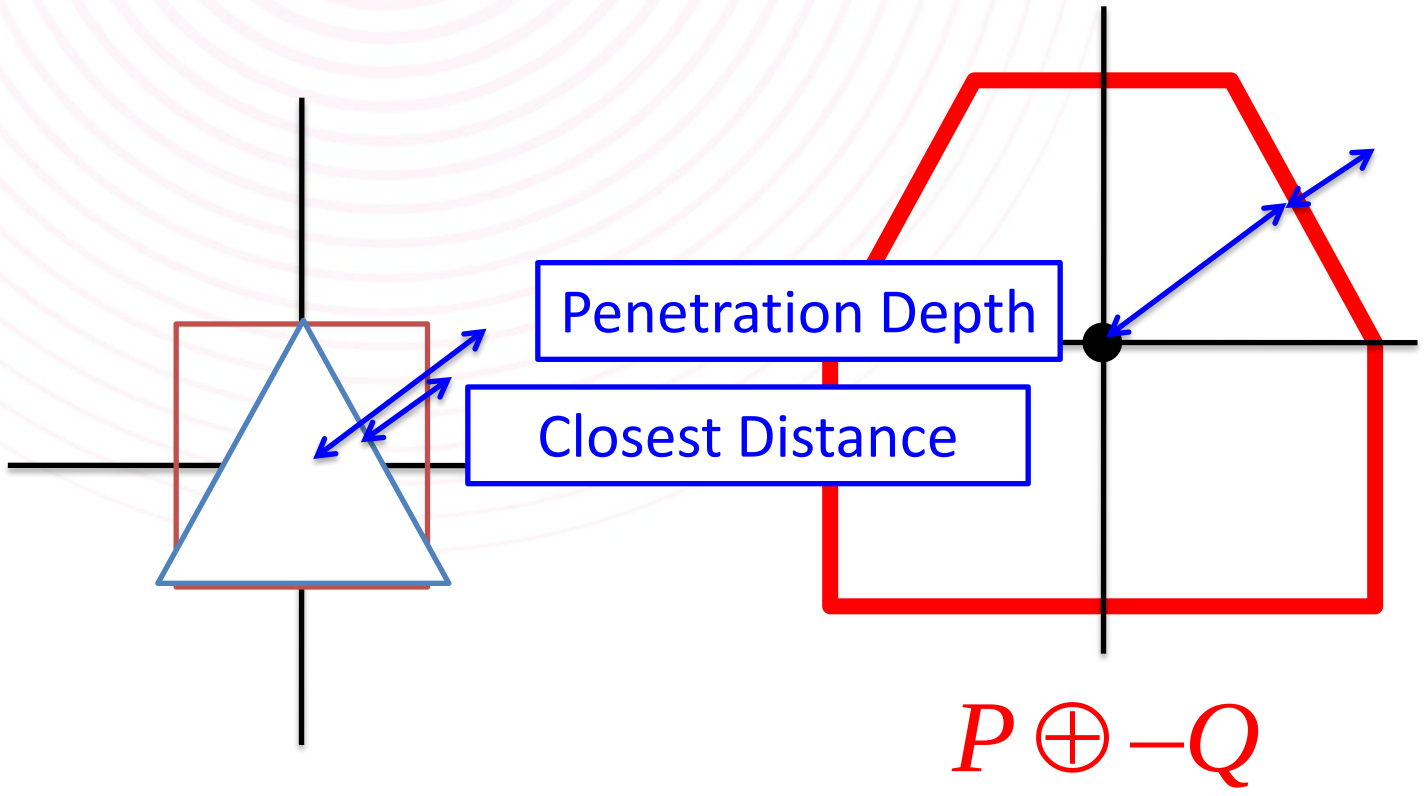
$$P \oplus Q = \{\mathbf{p} + \mathbf{q} \mid \mathbf{p} \in P, \mathbf{q} \in Q\}$$

$$P \oplus -Q = \{\mathbf{p} - \mathbf{q} \mid \mathbf{p} \in P, \mathbf{q} \in Q\}$$



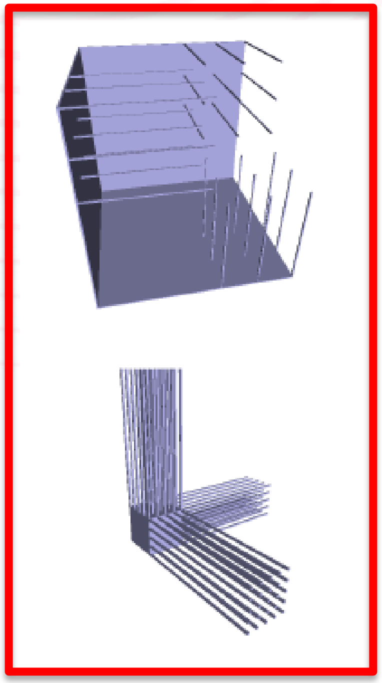


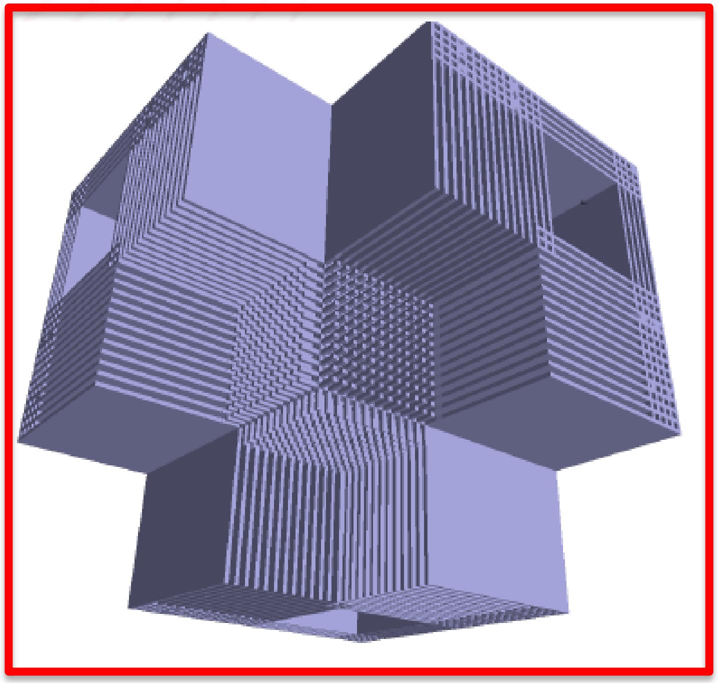
Proximity VS Minkowski Sum



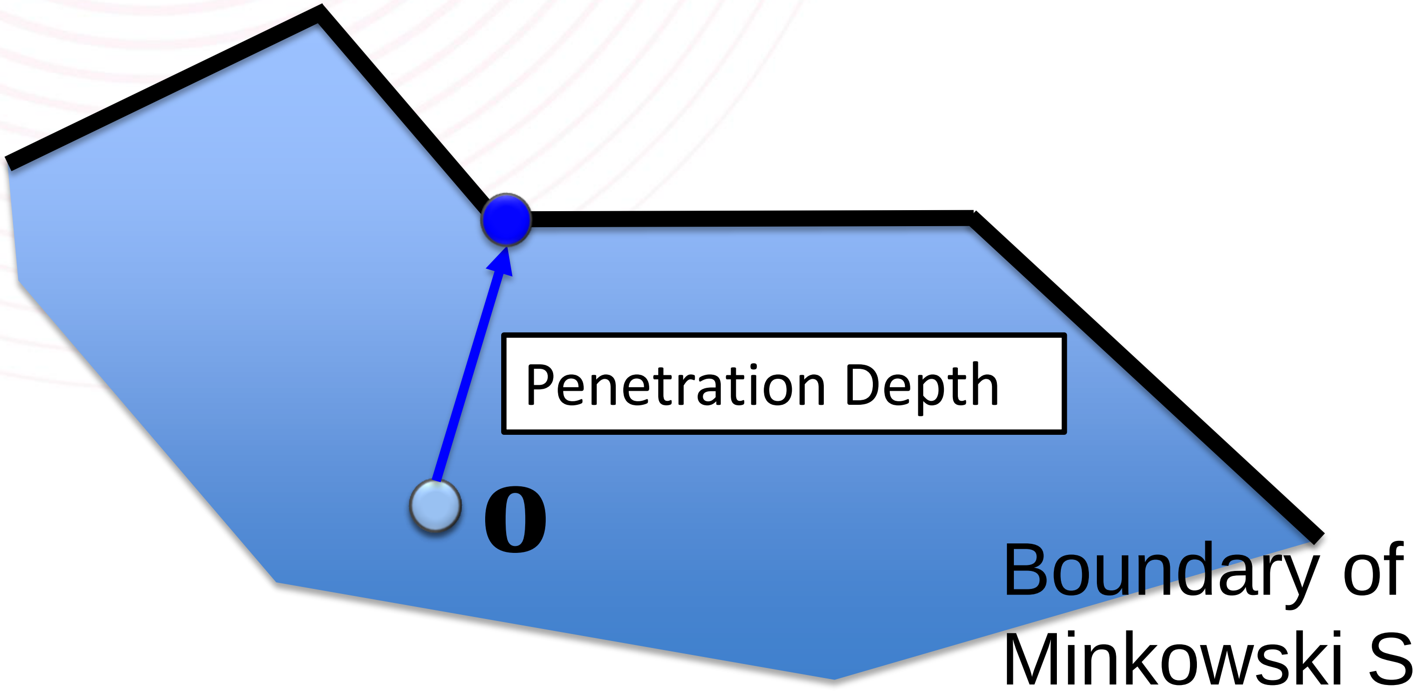
Combinatorial Explosion

- Complexity of Minkowski Sum
  - $O(m^3n^3)$ with m and n triangles

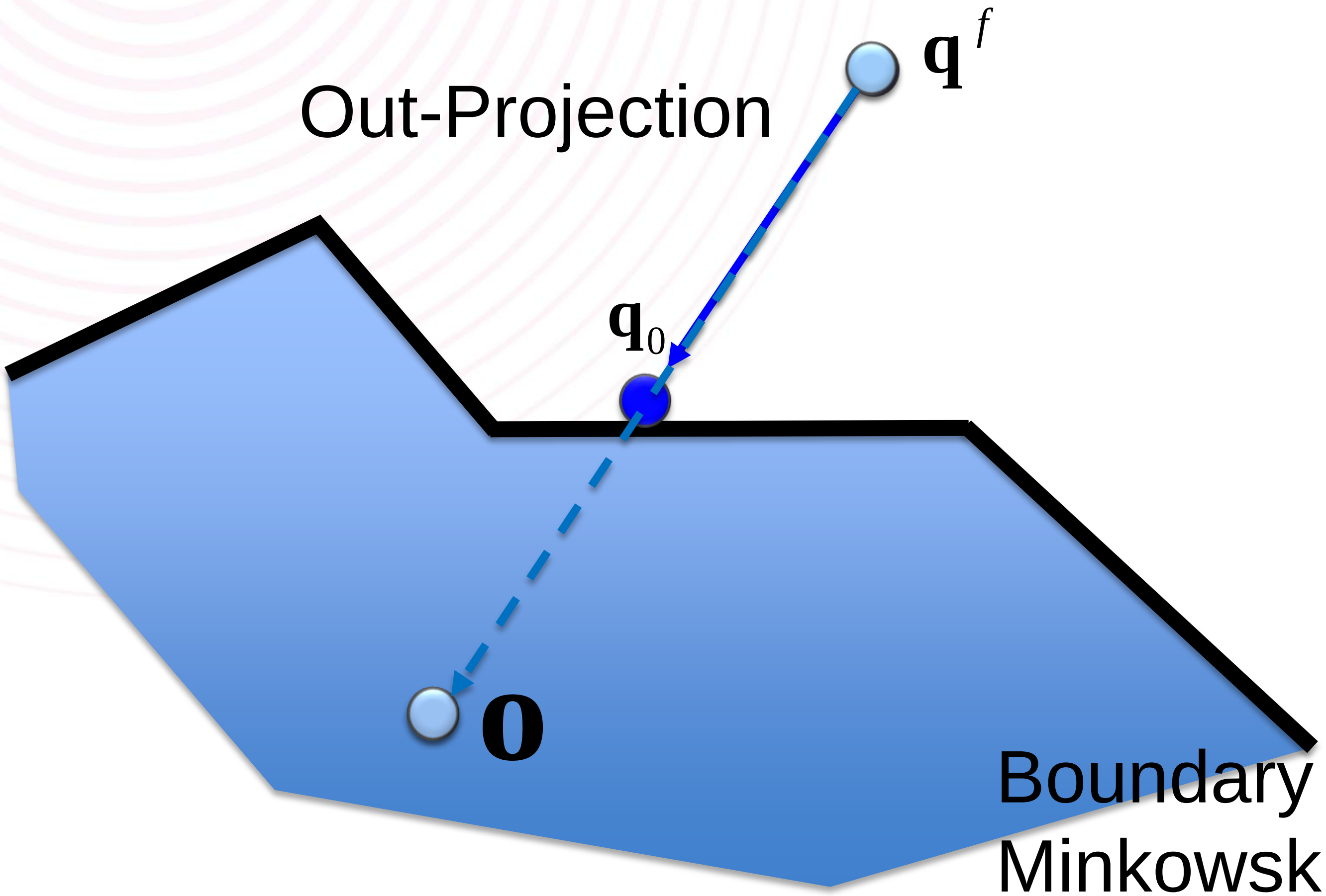




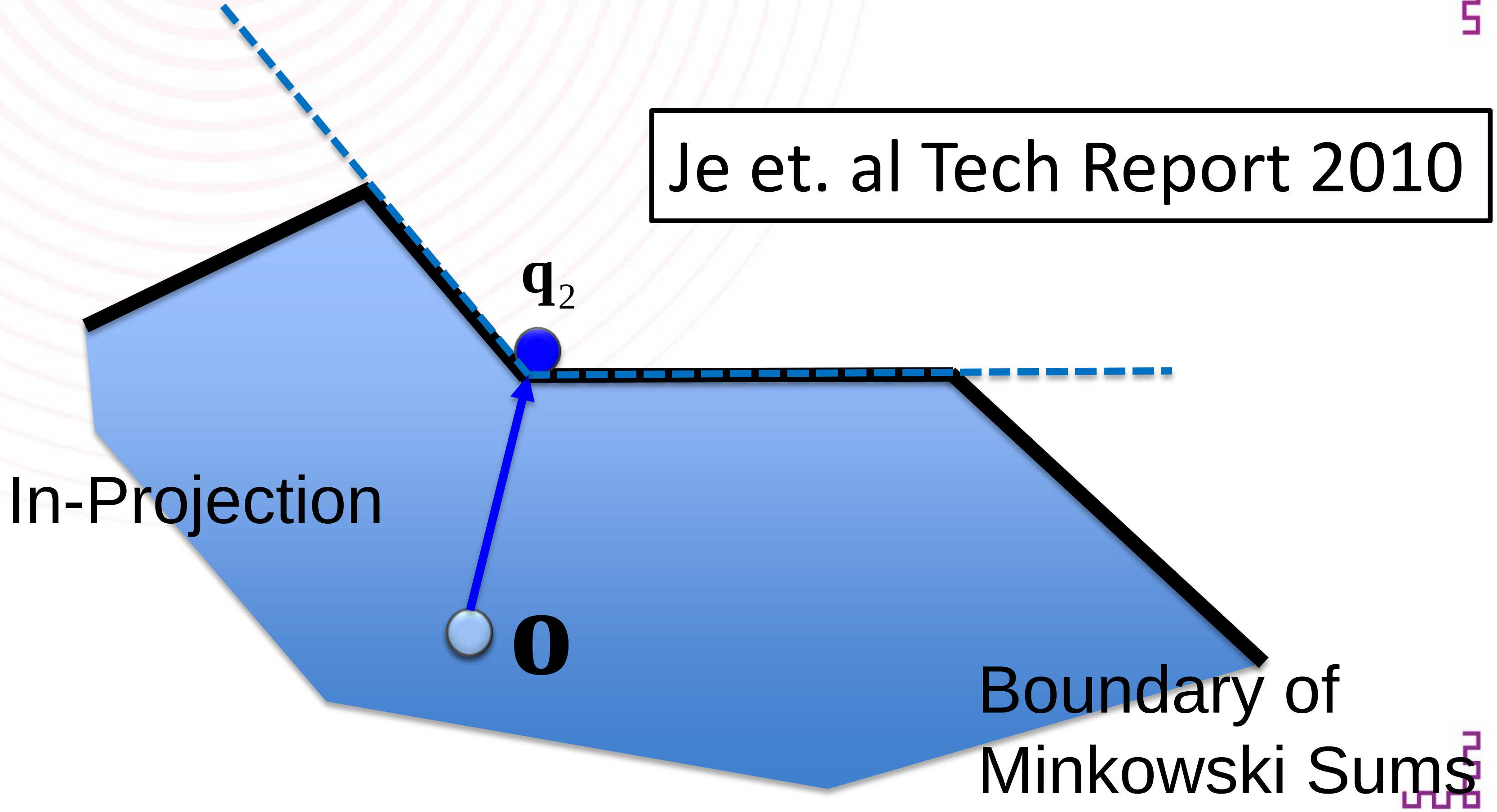
PD Estimation



Out-Projection = CCD

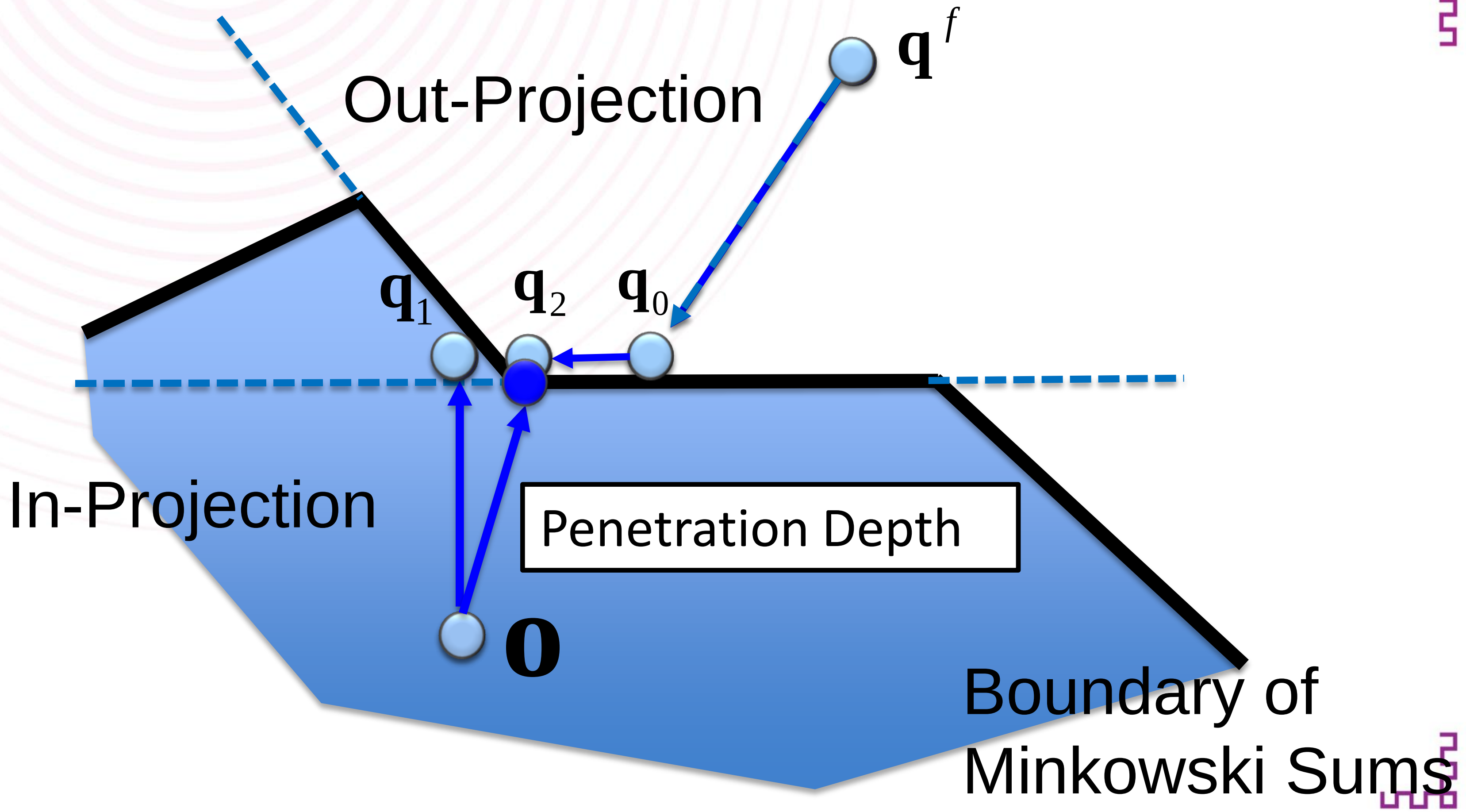


In-Projection = LCP
(Linear Complementarity Problem)

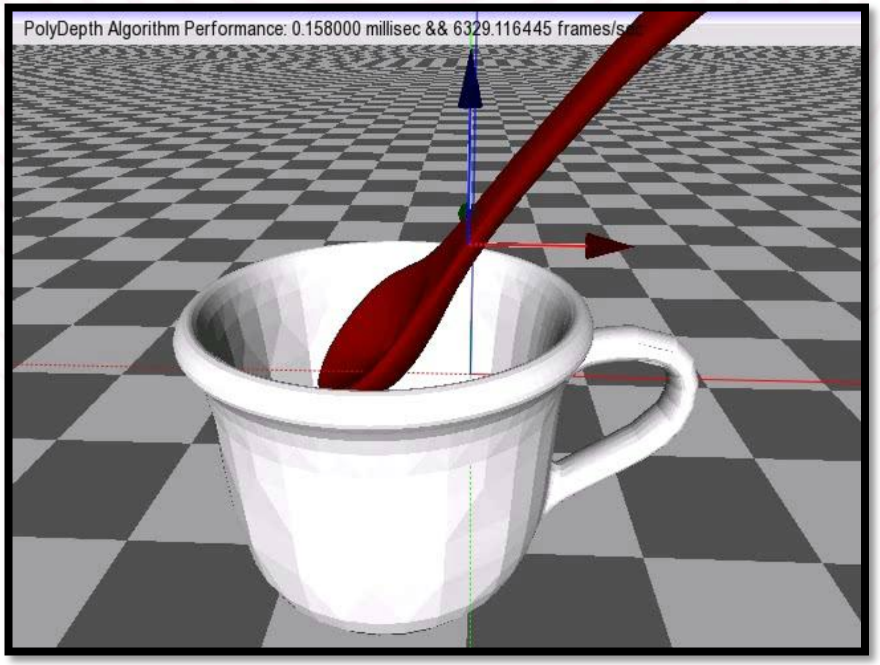


Je et. al Tech Report 2010

PolyDepth: Iterative Optimization



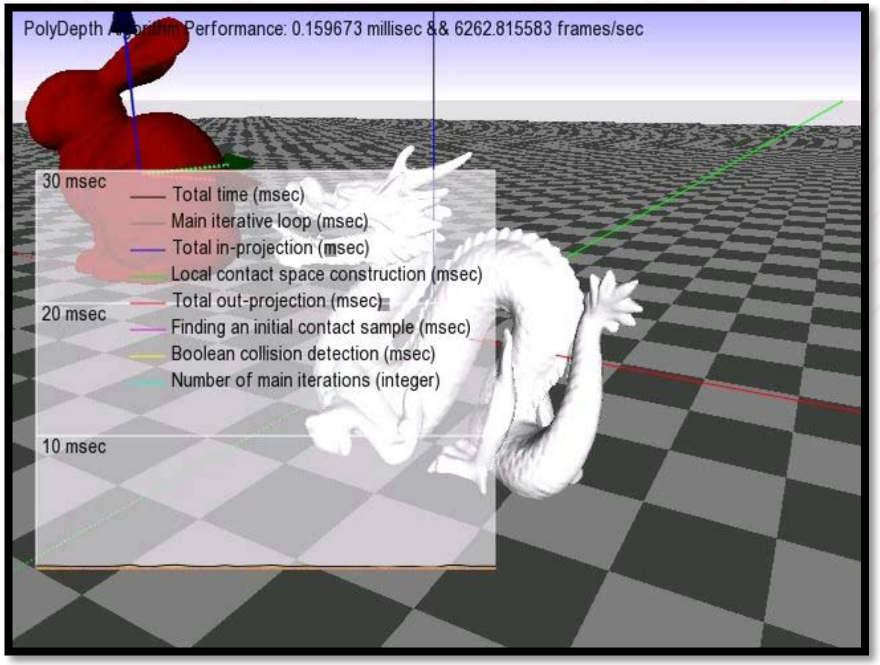
PolyDepth Performance


PolyDepth Algorithm Performance: 0.158000 millisec && 6329.116445 frames/sec

- Spoon: 1.3K triangles
- Cup: 8.4K triangles

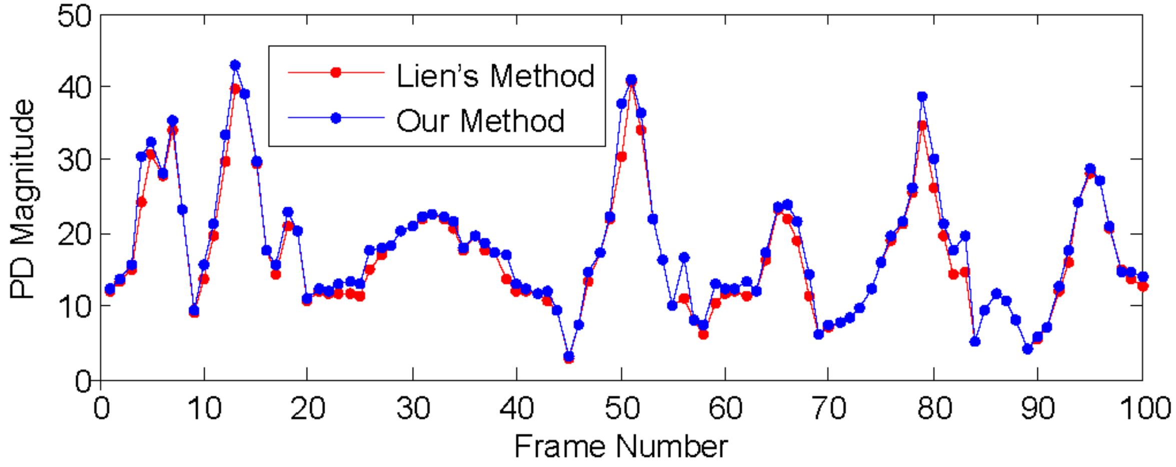
- Time: 1~7 msec

PolyDepth Performance

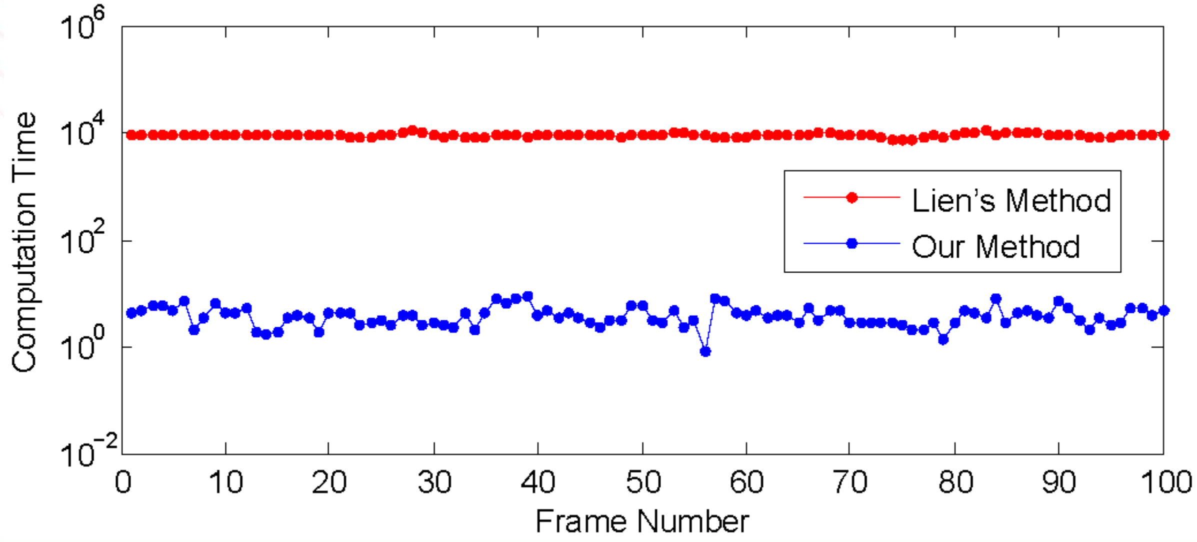

PolyDepth Algorithm Performance: 0.159673 millisec && 6262.815583 frames/sec
30 msec
20 msec
10 msec
Total time (msec)
Main iterative loop (msec)
Total in-projection (msec)
Local contact space construction (msec)
Total out-projection (msec)
Finding an initial contact sample (msec)
Boolean collision detection (msec)
Number of main iterations (integer)

- Bunny: 40K triangles
- Dragon: 174K triangles
- Time: 2~15 msec

Comparison against Exact Solution



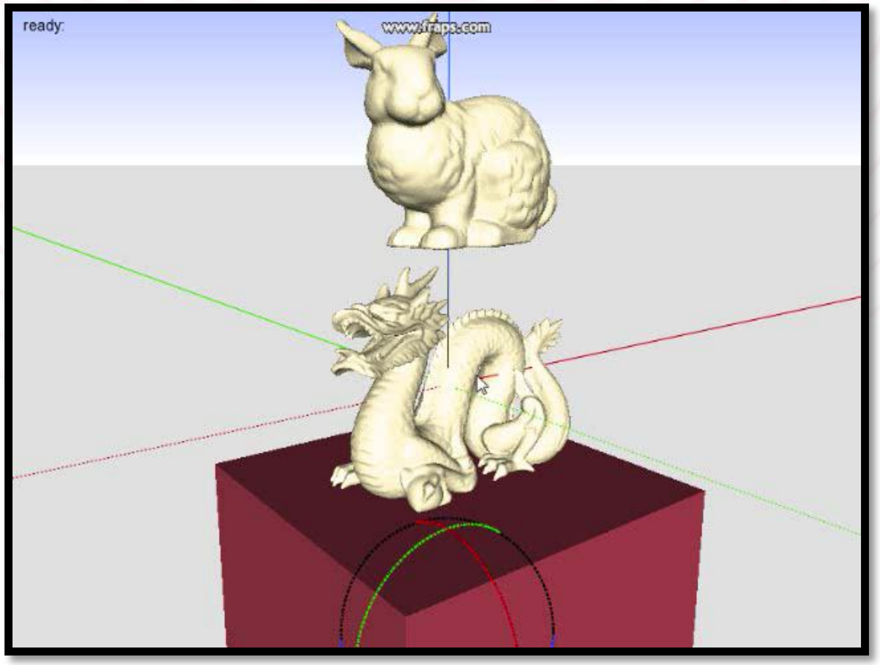
Accuracy

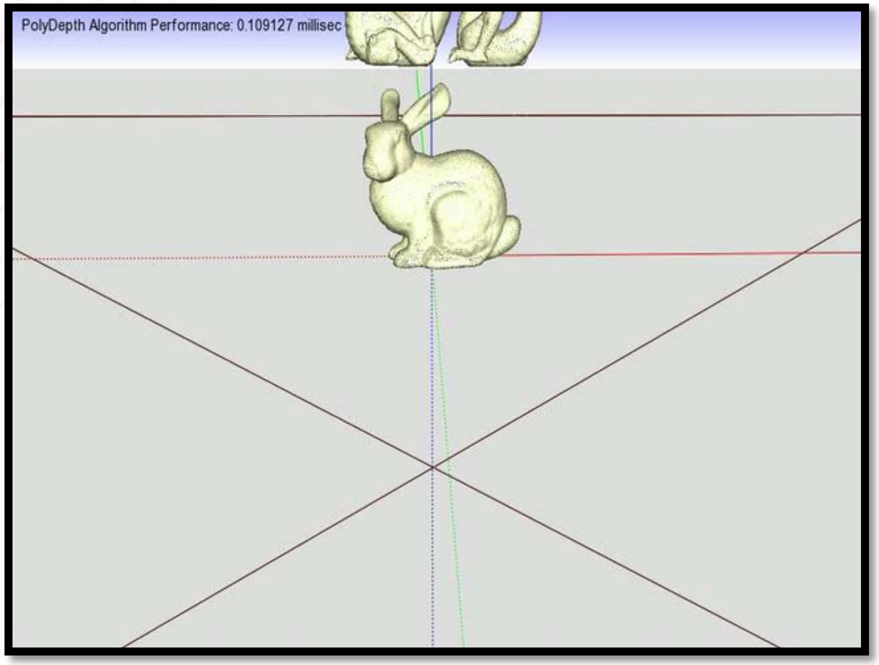


Performance

APPLICATIONS

Real-time Physics Simulation using PD





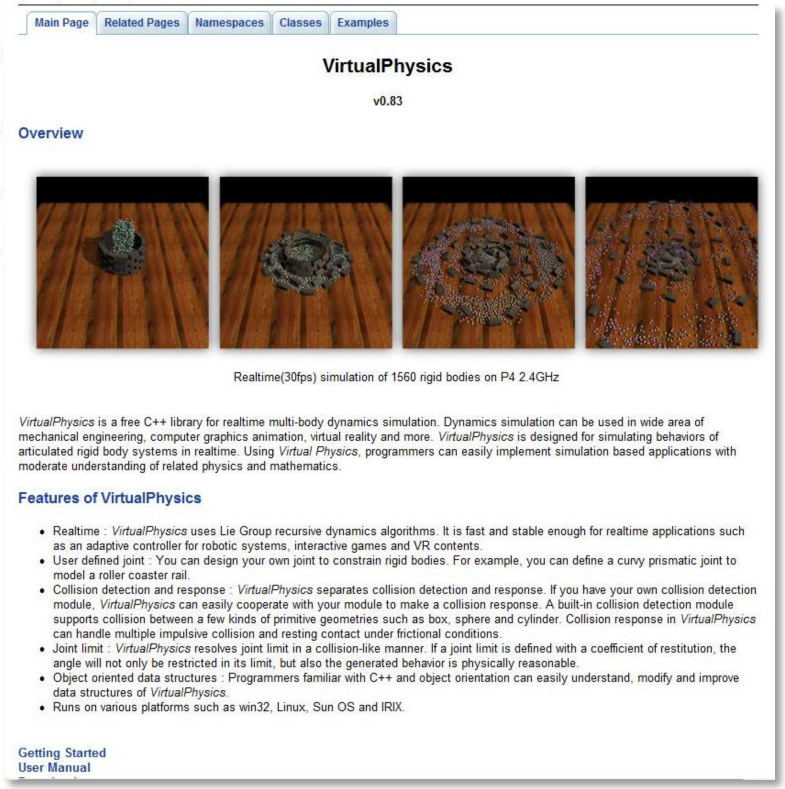
214K triangles in total

802K triangles in total

Integration with Physics Engine

http://virtualphysics.kr

VIRTUAL PHYSICS
REALTIME DYNAMICS SIMULATION LIBRARY



VirtualPhysics

v0.83

Overview

Realtime(30fps) simulation of 1560 rigid bodies on P4 2.4GHz

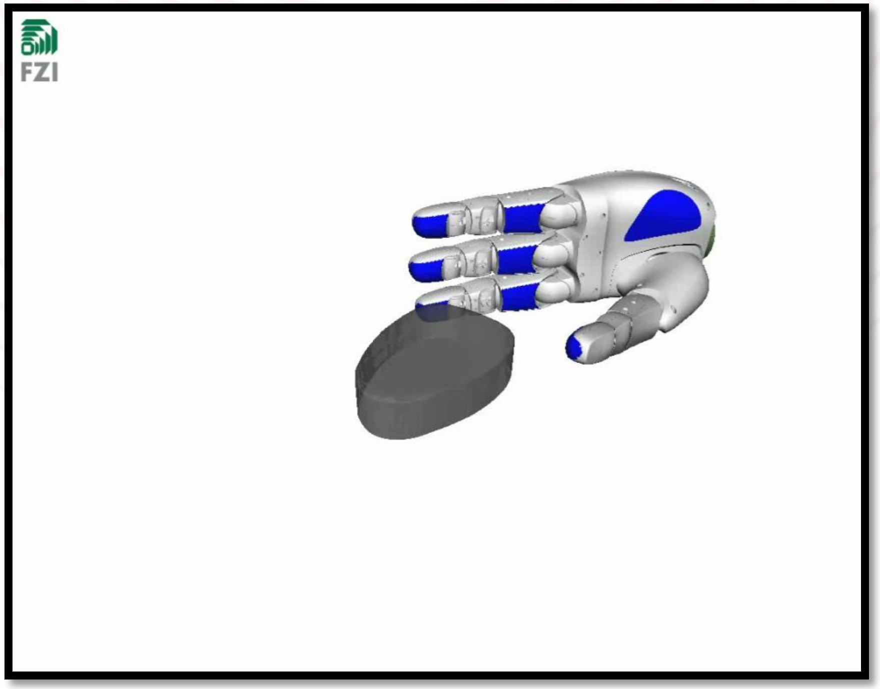
VirtualPhysics is a free C++ library for realtime multi-body dynamics simulation. Dynamics simulation can be used in wide area of mechanical engineering, computer graphics animation, virtual reality and more. *VirtualPhysics* is designed for simulating behaviors of articulated rigid body systems in realtime. Using *Virtual Physics*, programmers can easily implement simulation based applications with moderate understanding of related physics and mathematics.

Features of VirtualPhysics

- Realtime : *VirtualPhysics* uses Lie Group recursive dynamics algorithms. It is fast and stable enough for realtime applications such as an adaptive controller for robotic systems, interactive games and VR contents.
- User defined joint : You can design your own joint to constrain rigid bodies. For example, you can define a curvy prismatic joint to model a roller coaster rail.
- Collision detection and response : *VirtualPhysics* separates collision detection and response. If you have your own collision detection module, *VirtualPhysics* can easily cooperate with your module to make a collision response. A built-in collision detection module supports collision between a few kinds of primitive geometries such as box, sphere and cylinder. Collision response in *VirtualPhysics* can handle multiple impulsive collision and resting contact under frictional conditions.
- Joint limit : *VirtualPhysics* resolves joint limit in a collision-like manner. If a joint limit is defined with a coefficient of restitution, the angle will not only be restricted in its limit, but also the generated behavior is physically reasonable.
- Object oriented data structures : Programmers familiar with C++ and object orientation can easily understand, modify and improve data structures of *VirtualPhysics*.
- Runs on various platforms such as win32, Linux, Sun OS and IRIX.

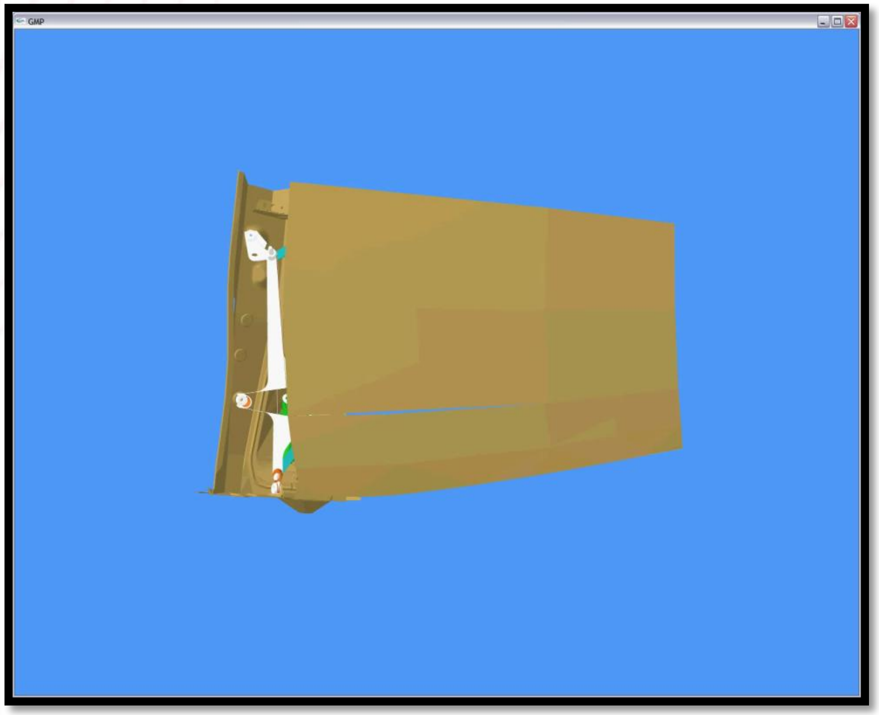
Getting Started
User Manual

Robotic Grasping using CCD



With Zhixing Xue @ FZI

CAD Disassembly using CCD



With Liangjun Zhang @ Stanford/UNC

Summary

	CCD	PD
Concept	Collision avoidance	Collision correction
Usages	1. Constraint-based dynamics 2. Exact motion planning 3. Grasping	1. Penalty-, impulse-based dynamics 2. Retraction-based motion planning
Complexities	O(mn)	$O(m^3n^3)$

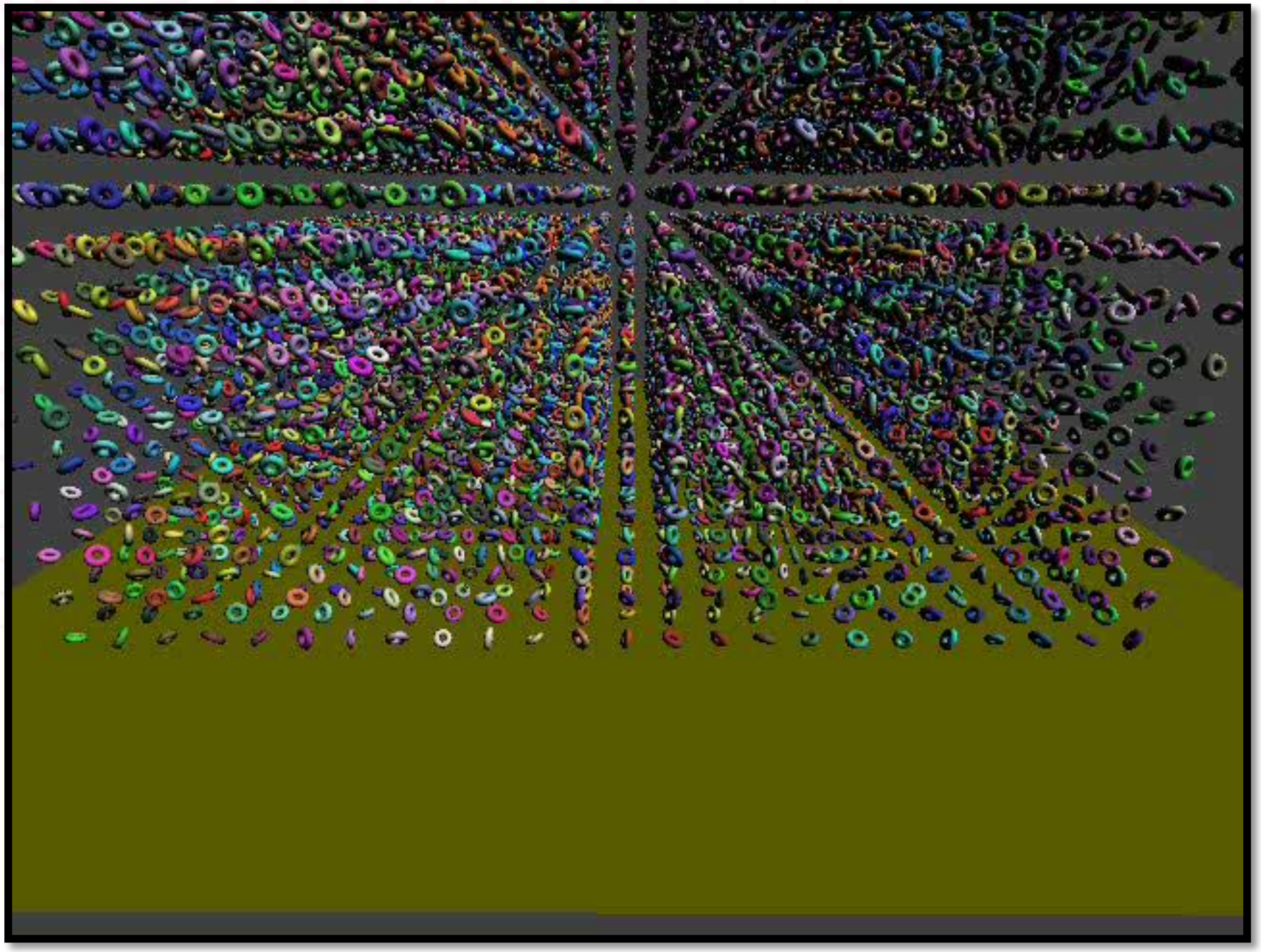
Future Work

- Continuous collision detection
  - N-body
  - Non-linear motion

- PD
  - Articulated body
  - Deformable
  - N-body

Collision Culling of a Million Bodies on GPUs

Friday, 11:00 AM, Room E1-E4



Real-time Dynamics Simulation of 16,000 Rigid Bodies

Acknowledgements

- Min Tang, Xinyu Zhang, Minkyoung Lee, Youngeun Lee (Ewha)
- Stephane Redon (INRIA)
- Dinesh Manocha (UNC)
- Liangjun Zhang (Stanford)
- Zhixing Xue (FZI)

- KEIT/MKE (IT core research)
- KRF (Young investigator award)

Main References

- X. Zhang, M. Lee, Y. Kim, **Interactive Continuous Collision Detection for Non-convex Polyhedra**, Pacific Graphics 2006 http://graphics.ewha.ac.kr/FAST

- X. Zhang, S. Redon, M. Lee, Y. Kim, **Continuous Collision Detection for Articulated Models using Taylor Models and Temporal Culling**, SIGGRAPH 2007 http://graphics.ewha.ac.kr/CATCH

- M. Tang, Y. Kim, D. Manocha, **C^2A: Controlled Conservative Advancement for Interactive Continuous Collision Detection**, IEEE ICRA 2009 http://graphics.ewha.ac.kr/C2A

Main References

- M. Tang, M. Lee, Y. Kim, **Interactive Hausdorff Distance Computation for General Polygonal Models**, SIGGRAPH 2009
http://graphics.ewha.ac.kr/HDIST

- C. Je, M. Tang, Y. Lee, M. Lee, Y. Kim, **PolyDepth: Real-time Penetration Depth Computation using Iterative Contact-space Projection**, Ewha Technical Report 2010
http://graphics.ewha.ac.kr/PolyDepth

- M. Tang, Y. Kim, D. Manocha, **Efficient Local Planning using Connection Collision Query**, Workshop on Algorithmic Foundations of Robotics 2010
http://graphics.ewha.ac.kr/CCQ

Thank you for listening!

http://graphics.ewha.ac.kr

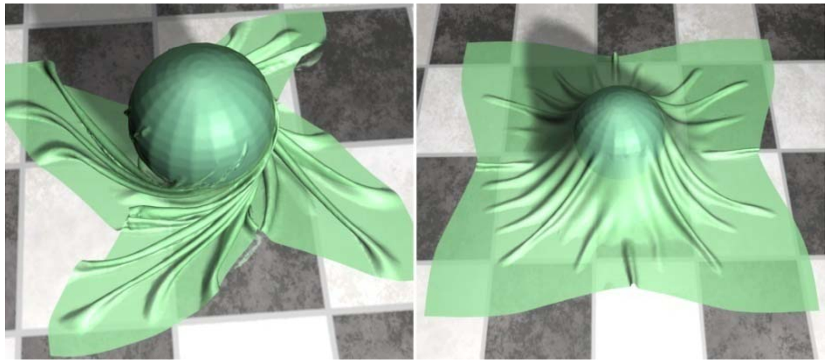
Interactive Collision Detection for Deformable and Fracturing Objects

Sung-Eui Yoon
Scalable Graphics Lab.
KAIST

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

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
- HPCCD: Hybrid parallel continuous collision detection
- FASTCD: Fracturing-Aware Stable Collision Detection

Overview

- Background
- HPCCD: Hybrid parallel continuous collision detection
- FASTCD: Fracturing-Aware Stable Collision Detection

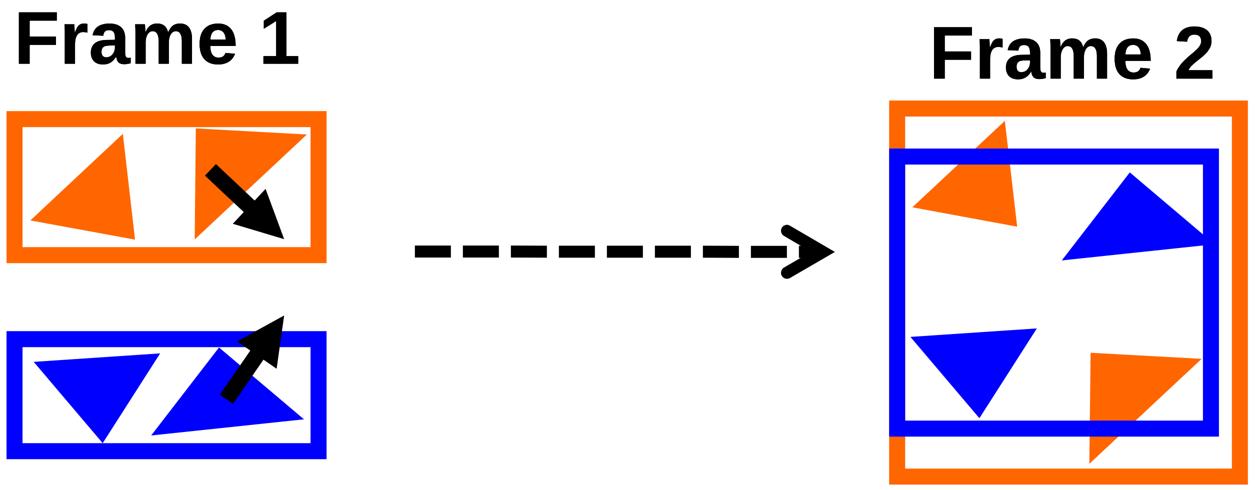
Background

- BVH-based collision detection
- BVH construction

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

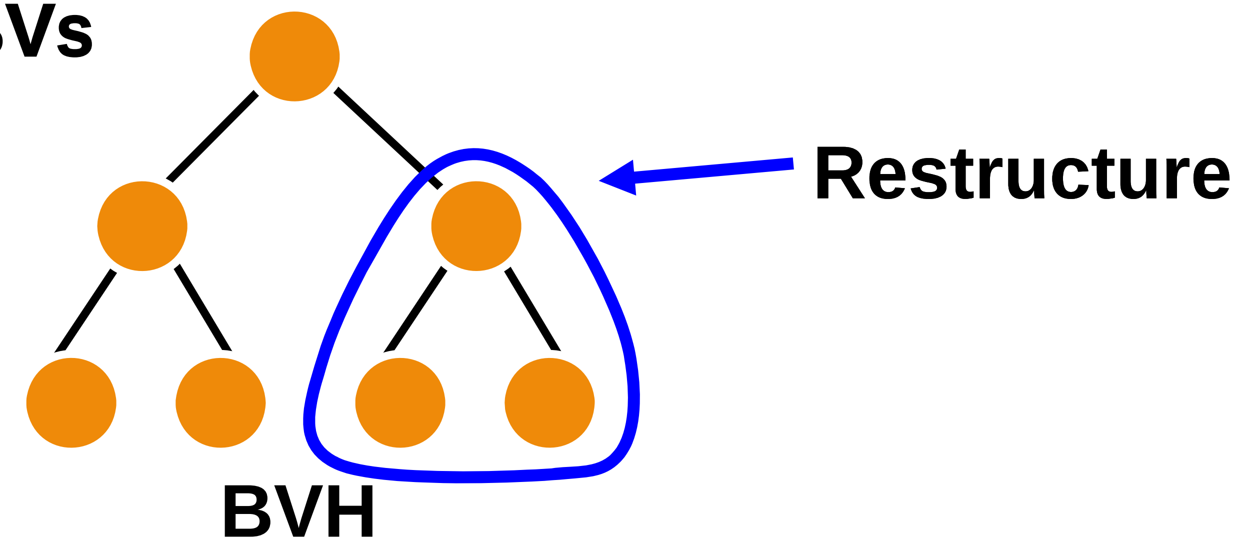
BVH Refitting

- 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



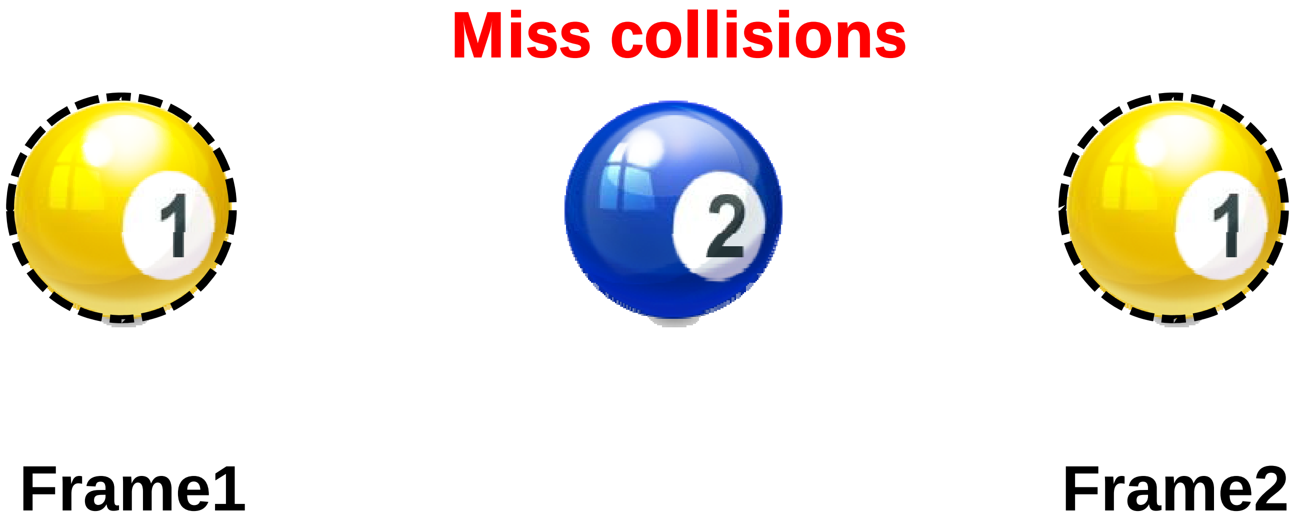
- Requires a metric indentifying such subsets
  - Volume ratios of BVs of parent and child BVs [Zachmann 02, Larsson et al. 06, Yoon et al. 06]

Overview

- Background
- HPCCD: Hybrid parallel continuous collision detection
- FASTCD: Fracturing-Aware Stable Collision Detection

Discrete vs. Continuous

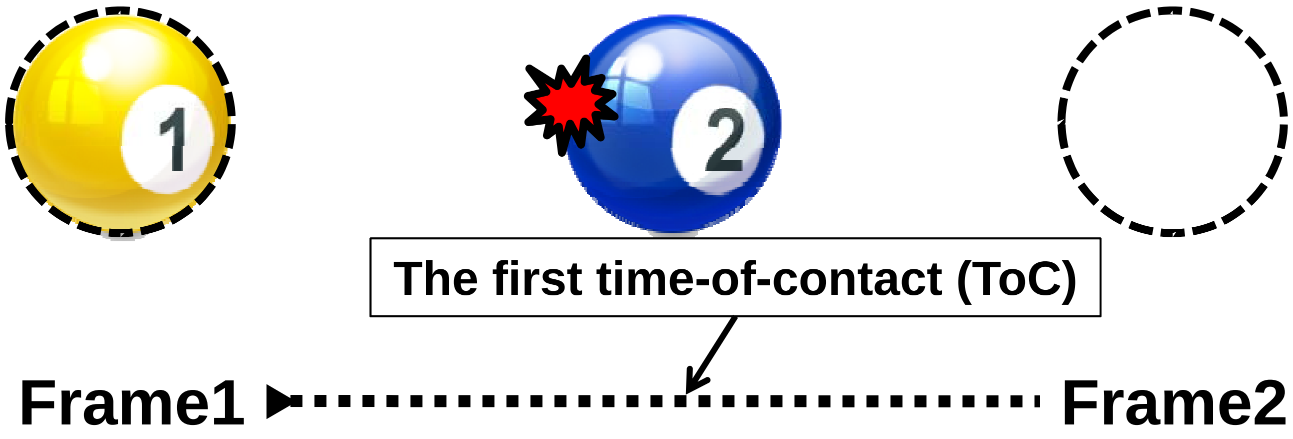
- Discrete collision detection (DCD)
  - Detect collisions at each frame
  - Fast, but can miss collisions



Pacific Graphics 2009

Discrete vs. Continuous

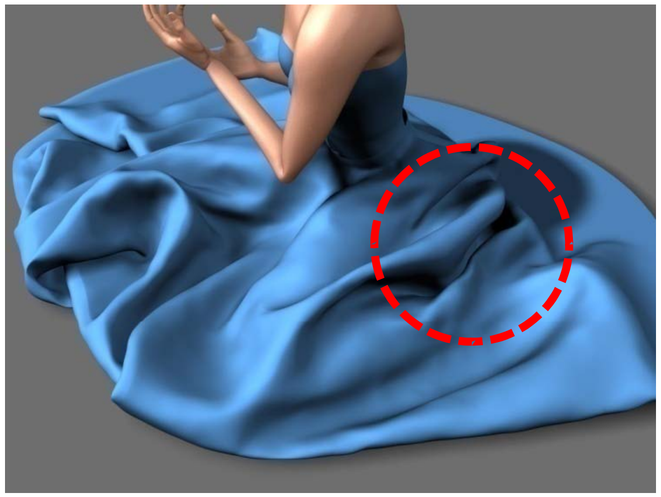
- Discrete collision detection (DCD)
- Continuous collision detection (CCD)
  - Identify the first time-of-contact (ToC)
  - Accurate, but requires a long computation time
  - Not widely used in interactive applications



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

Recent Parallel Collision Detection Methods

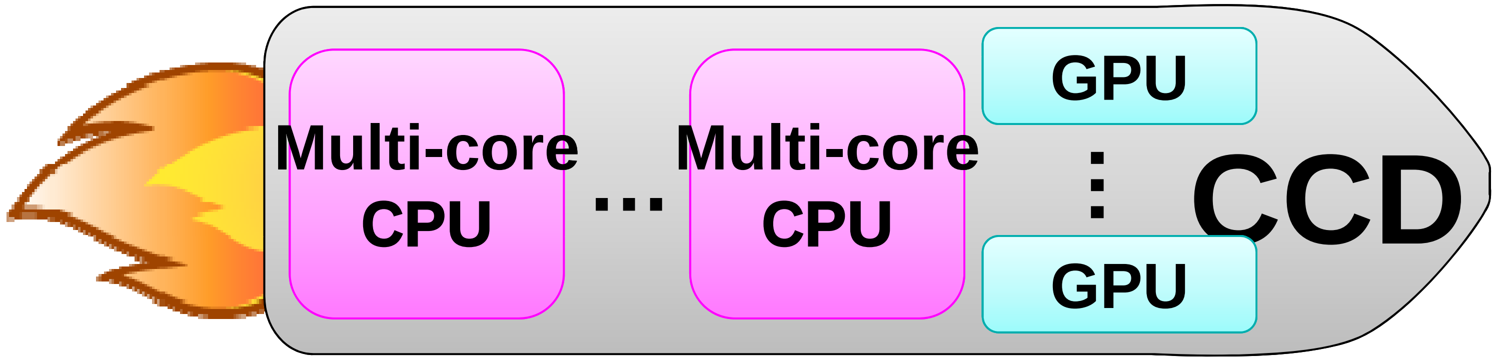
- **CPU-based CD method**
  - **Tang et al., Solid and Physical Modeling, 2009**
- **gProximity**
  - **Laterbach et al., Eurographics 2010**
- **Hybrid parallel CD method**
  - **Kim et al., Pacific Graphics 2009**

Recent Parallel Collision Detection Methods

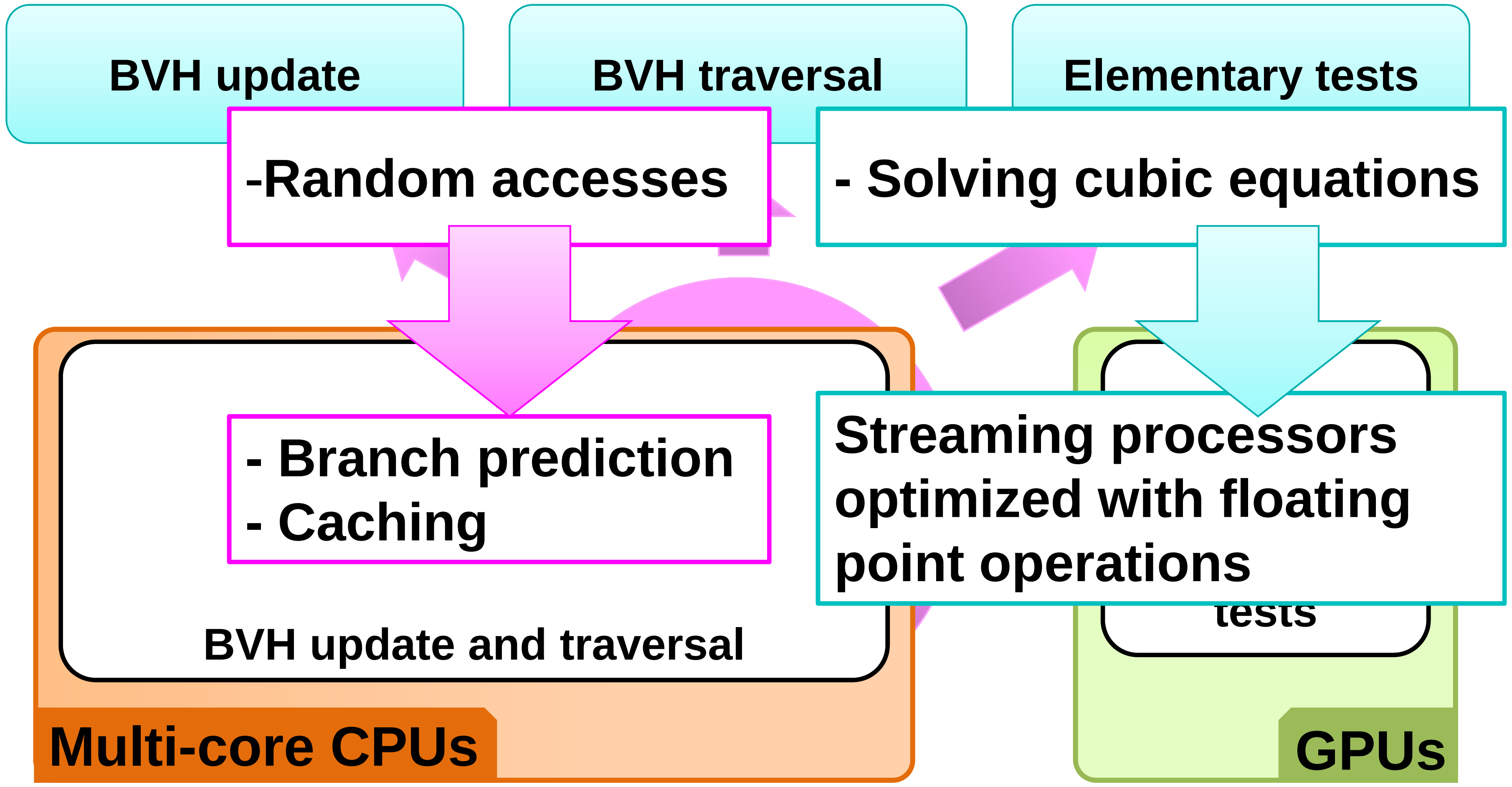
- CPU-based CD method
  - Tang et al., Solid and Physical Modeling, 2009
- gProximity
  - Laterbach et al., Eurographics 2010
- Hybrid parallel CD method
  - Kim et al., Pacific Graphics 2009

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



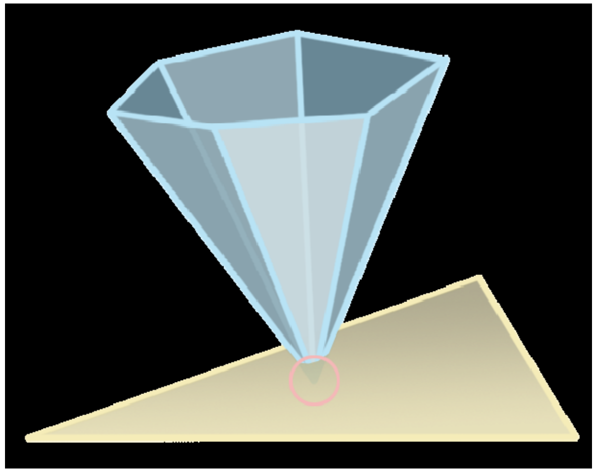
Testing Environment

- **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**

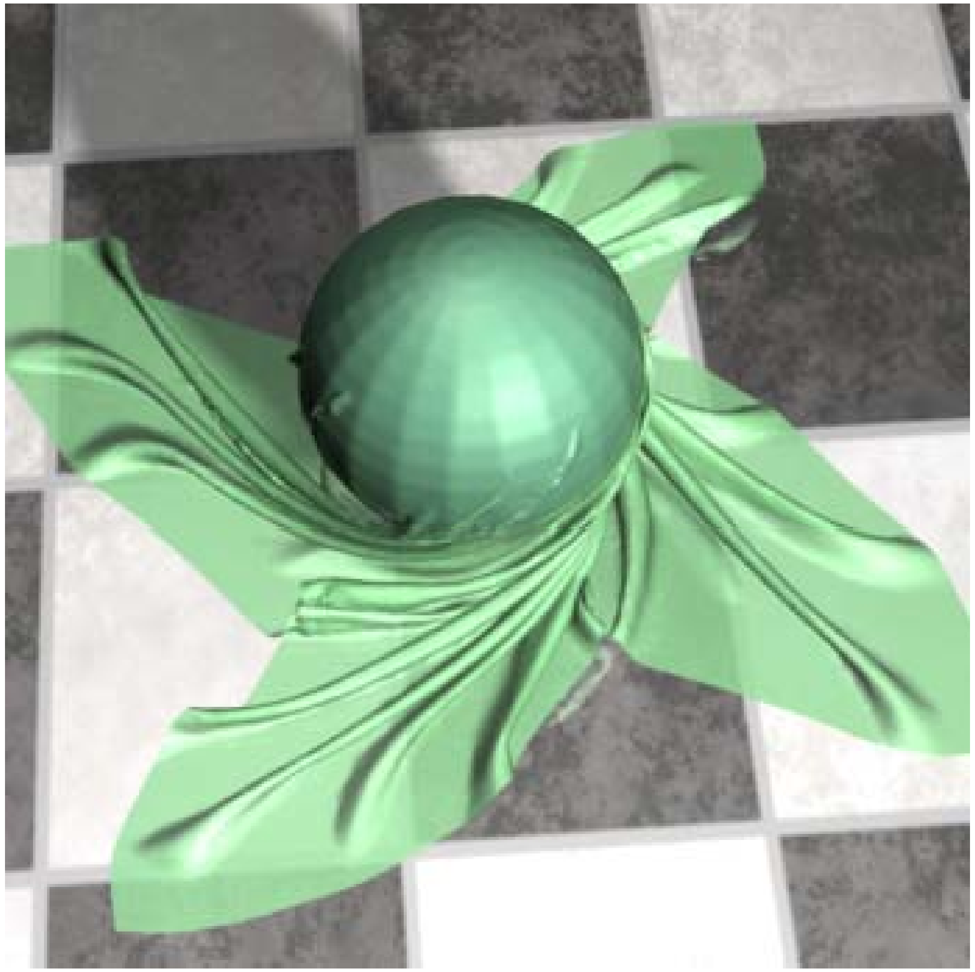
KAIST

BVH-based CCD

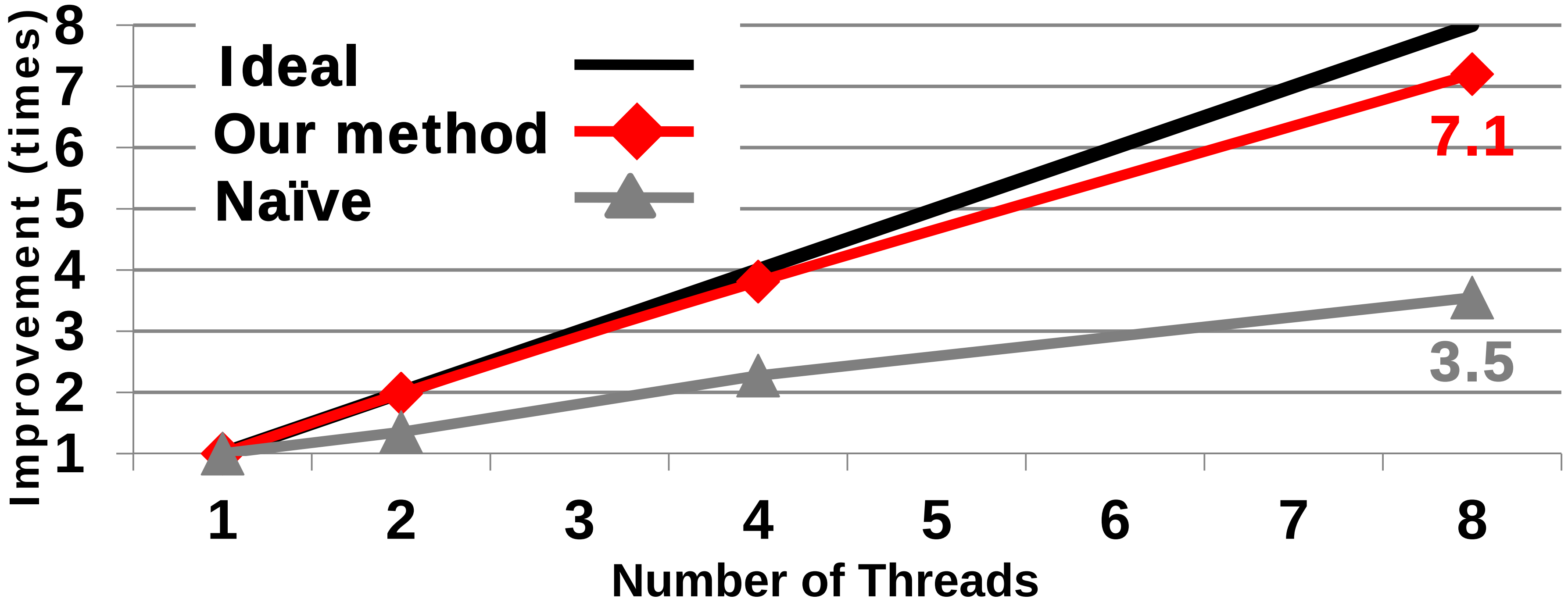
- Axis-aligned bounding boxes
- Linear interpolations on vertices for the continuous motion
  - Vertex-face and edge-edge tests for CCD [Provot 96]
- Feature based BVHs [Curtis et al., I3D 08]
  - Assign each features (e.g., vertex and edge) to each triangle



Results

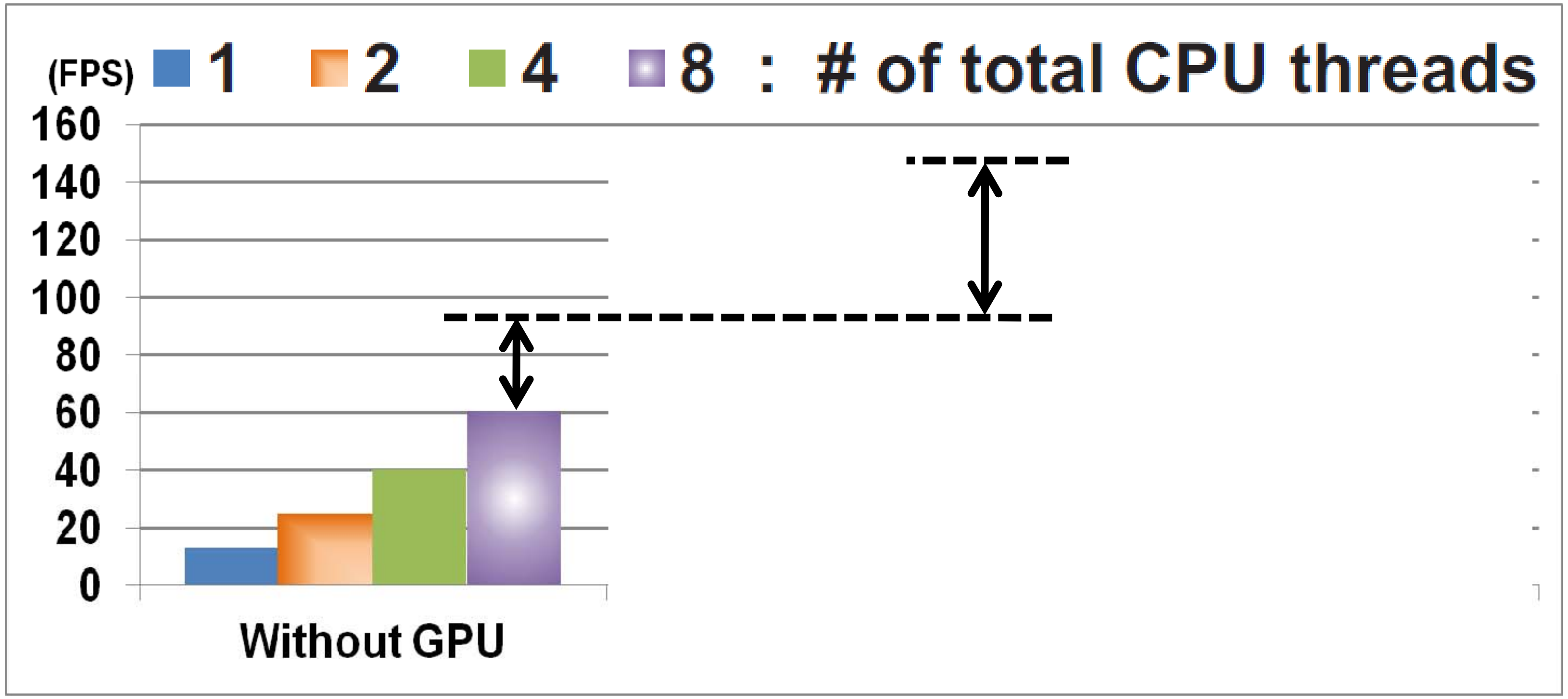


Results of a CPU-based Parallel CCD



- 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

Summary

- A **hybrid parallel** algorithm
    - Utilize both multi-core CPUs and GPUs

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

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

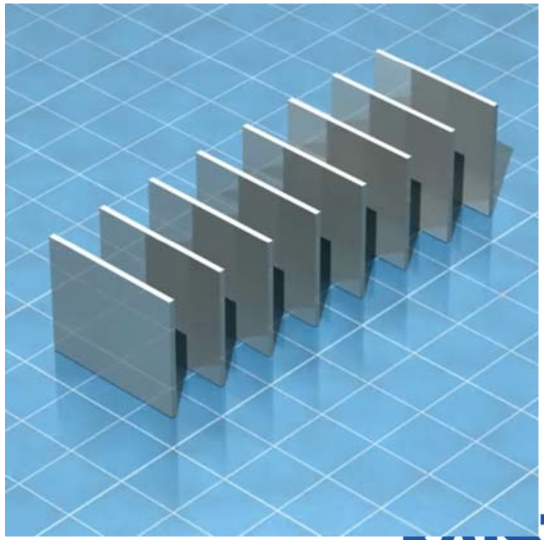
Overview

- Background
- HPCCD: Hybrid parallel continuous collision detection
- FASTCD: Fracturing-Aware Stable Collision Detection

CD for Fracturing Models

- More widely used in various applications to create more realistic interactions
- Fracturing
  - Changes the connectivity of a mesh: pre-computed hierarchies show low culling ratios
  - Places many objects in close proximity: CD cost is increasing

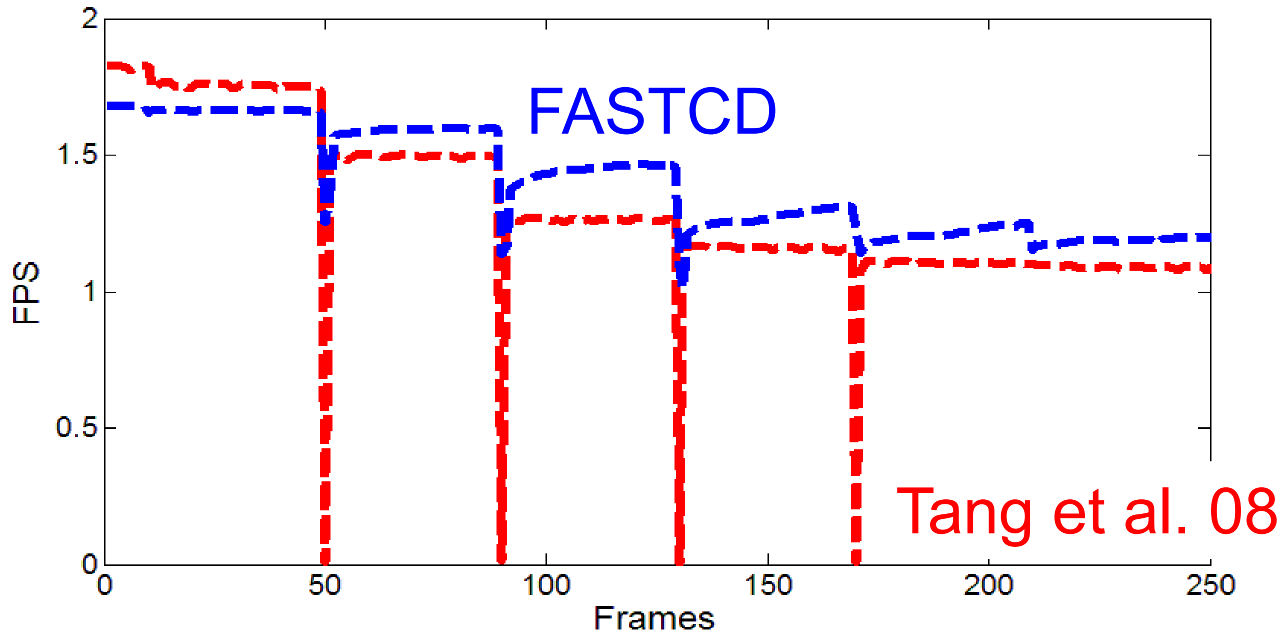
- Fracturing is one of the most challenging scenarios of collision detection



Our Approach

- FASTCD: 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

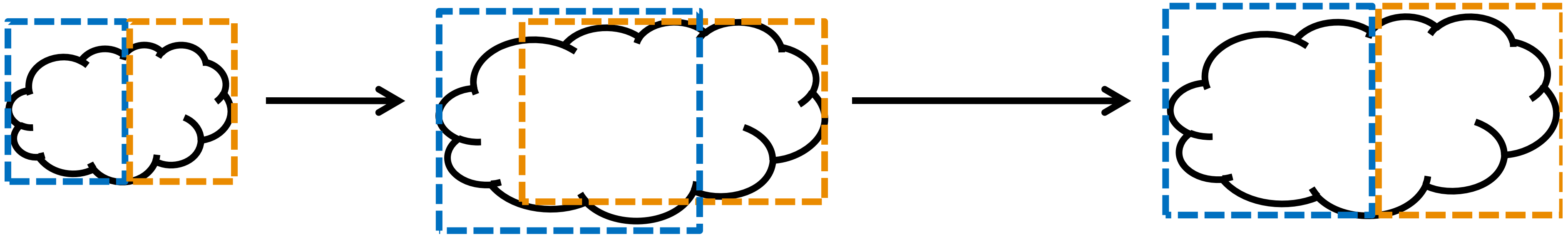
CCD Performance with the Breaking Dragon Model



- **FASTCD shows stable performance**

Selective Restructuring of BVHs

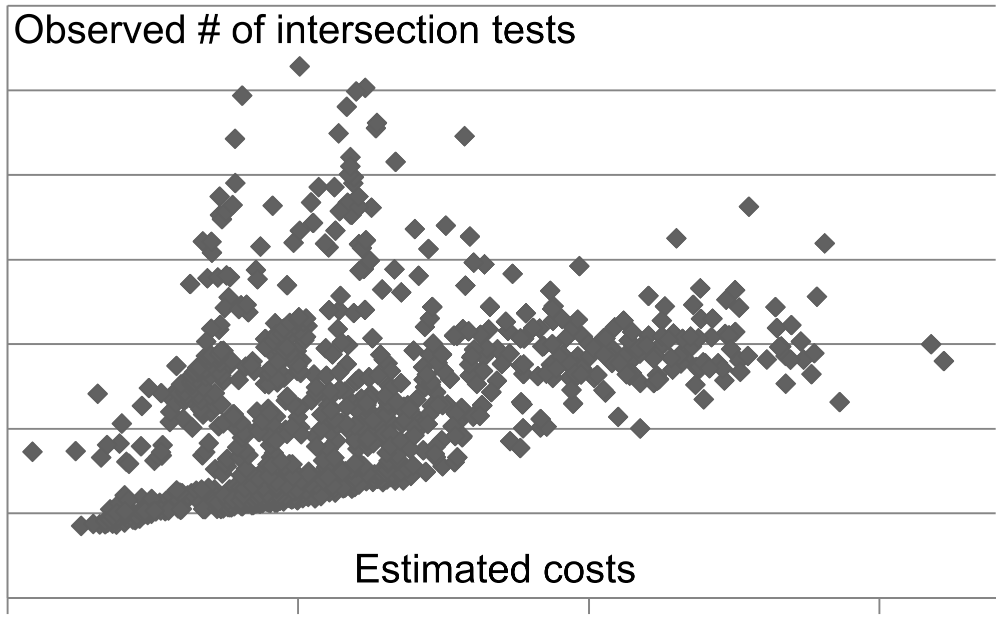
- **As models deform, culling efficiency of their BVHs can be getting lower**
  - **Selective restructuring can address the problem**

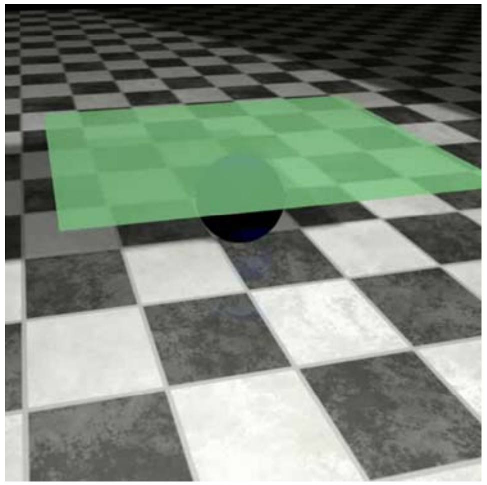


- **How to determine a culling efficiency of a BVH?**
  - **Heuristic metrics have been proposed**
- **A cost metric that measures the expected number of intersection tests is proposed**

Metric Validation

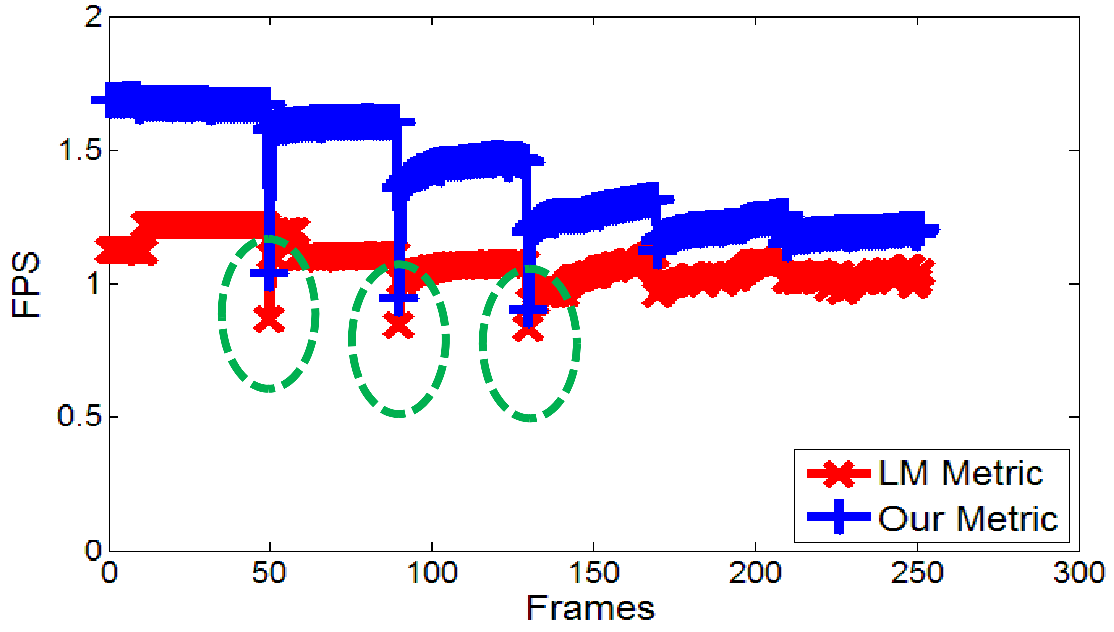
- **Estimated # of tests vs. Observed # of tests**





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

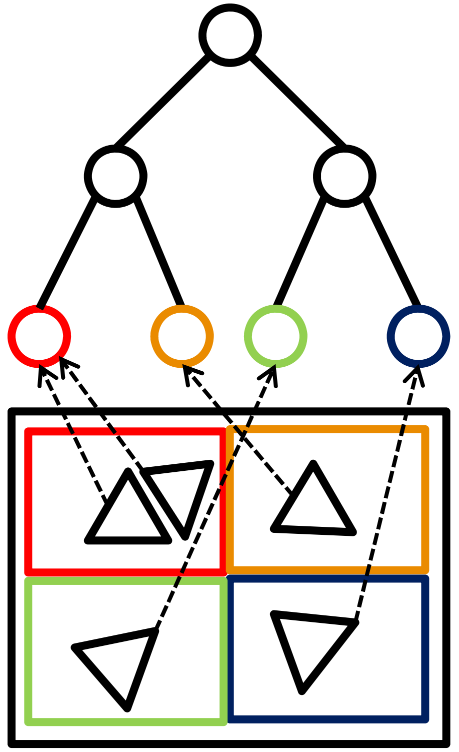
Result of Selective Restructuring



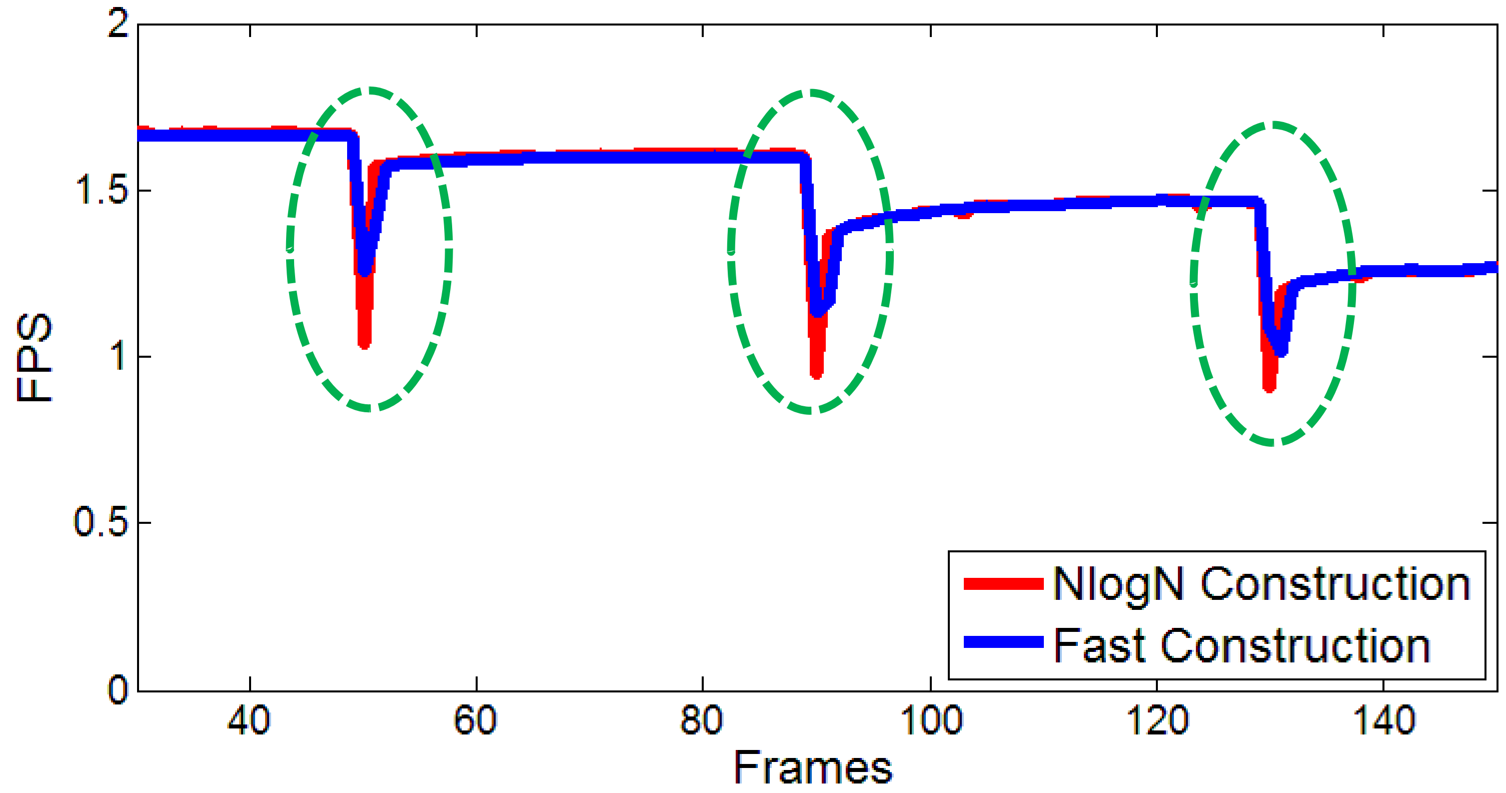
- LM metric : [Larsson and Akenine-Möller 2006]
- Performance degradations at topological changes
  → unstable

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

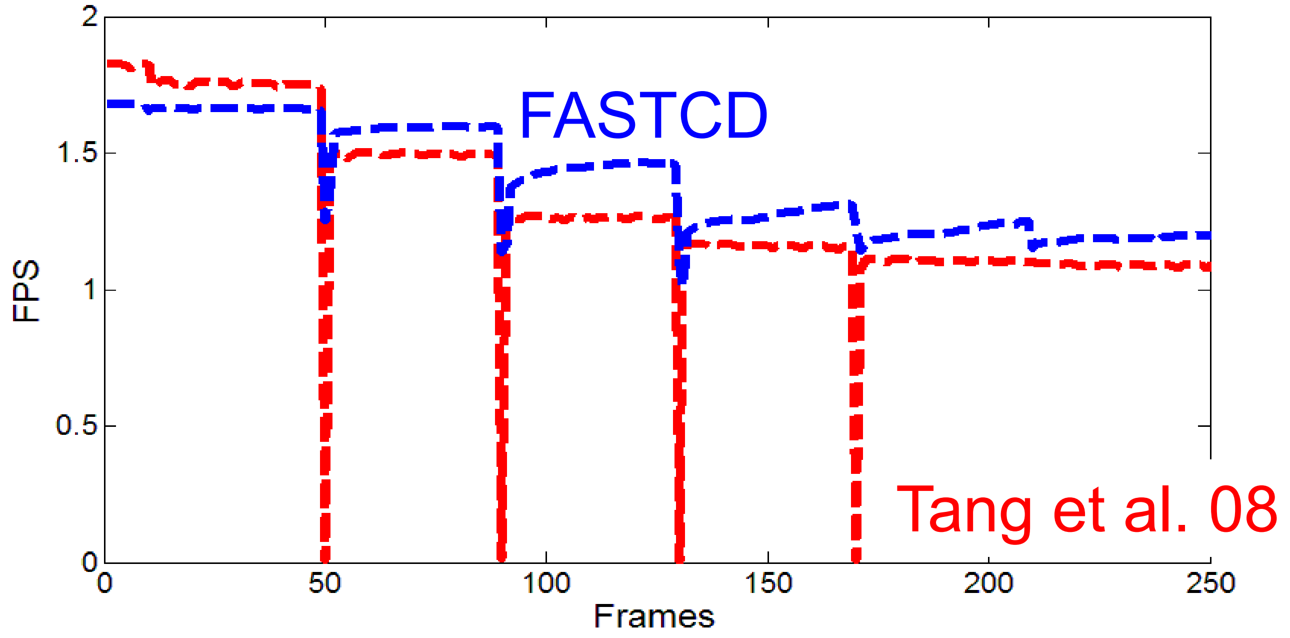


Result of Fast BVH Construction



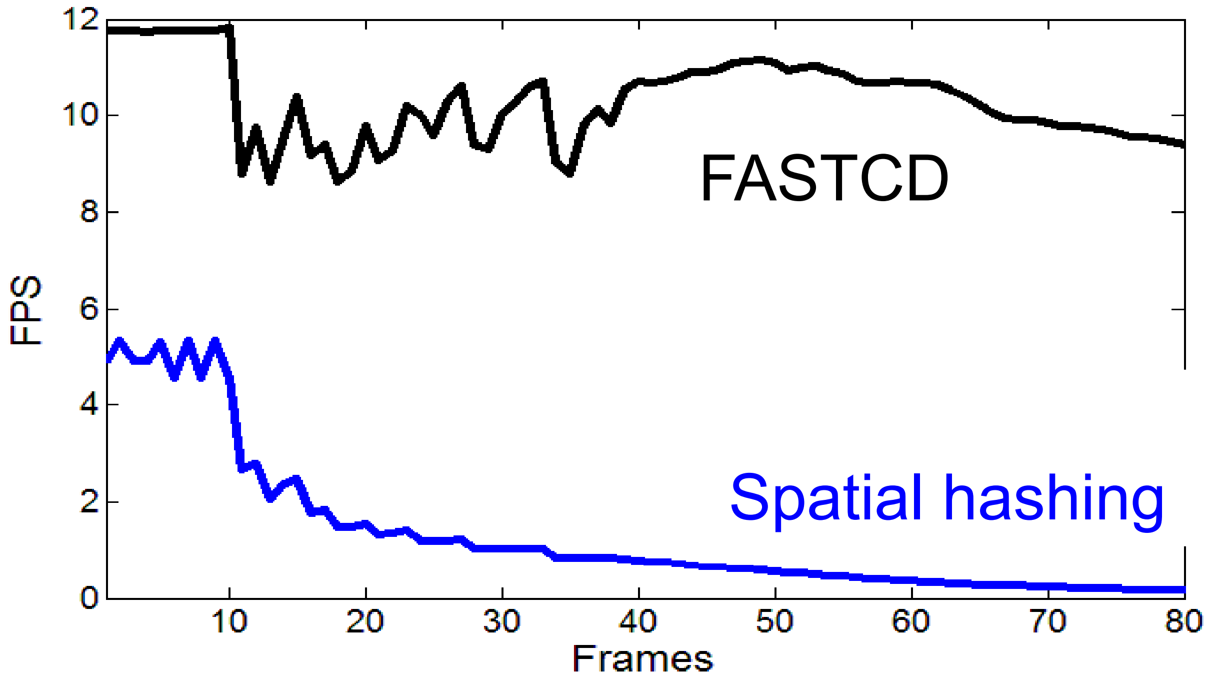
- **Performance degradations at fracturing events are reduced**

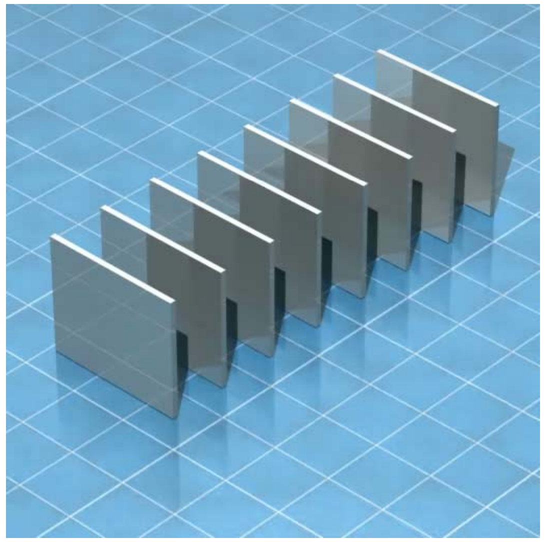
Comparison on CCD



- **FASTCD shows more stable performance**

Comparison (Discrete CD)





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

Summary

- Presented two recent BVH-based methods for interactive CD among large-scale deforming models
  - HPCCD: Hybrid Parallel Continuous Collision Detection

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

◆ Two fracturing models are available (http://sglab.kaist.ac.kr/FASTCD/)

Future Directions

- Various parallel proximity queries and their applications
  - g-Planner (GPU-based motion planner), AAAI 10
  - Hybrid proximity queries, under progress
  - Their applications to time-critical applications (e.g., robot motion planning)
- Volumetric representations
  - VolCCD, Tang et al. 2010, under progress

Acknowledgements

- Research collaborators
  - DukSu Kim, JaePil Heo, John Kim, Miguel Otaduy, Tang Min, JeongMo Hong, JoonKyung Seong, JaeHyuk Heo

- Funding sources
  - Ministry of Knowledge Economy
  - Microsoft Research Asia
  - Samsung
  - Korea Research Foundation

Reference

- FASTCD: Fracturing-Aware Stable Collision Detection
  - ACM SIGGRAPH/Eurographics Symp. on Computer Animation (SCA), 2010
  - http://sglab.kaist.ac.kr/FASTCD/
- HPCCD: Hybrid Parallel Continuous Collision Detection using CPUs and GPUs
  - Computer Graphics Forum (Pacific Graphics) 2009 http://sglab.kaist.ac.kr/HPCCD/