



## PSCC: Parallel Self-Collision Culling with Spatial Hashing on GPUs

<https://min-tang.github.io/home/PSCC/>

Min Tang<sup>1,2</sup>, Zhongyuan Liu<sup>1</sup>, Ruofeng Tong<sup>1</sup>, Dinesh Manocha<sup>1,3</sup>

<sup>1</sup>*Zhejiang University*

<sup>2</sup>*Alibaba-Zhejiang University Joint Institute of Frontier Technologies*

<sup>3</sup>*University of Maryland at College Park*

# Outline

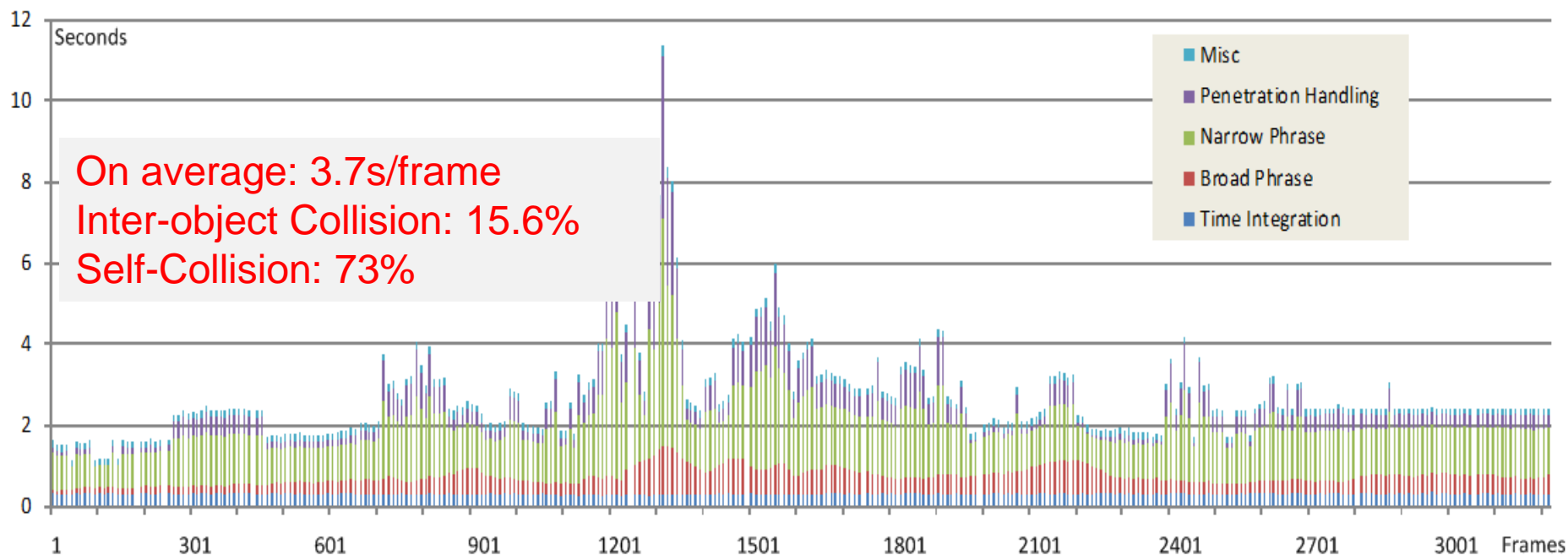
- Motivation & Challenges
- Related Work
- Main Results
- Algorithms
  - Parallel Self-collision Culling
  - Extended Spatial Hashing
  - Optimized Cloth Simulation Pipeline
- Result & Benchmarks
- Conclusions

# Outline

- Motivation & Challenges
- Related Work
- Main Results
- Algorithms
  - Parallel Self-collision Culling
  - Extended Spatial Hashing
  - Optimized Cloth Simulation Pipeline
- Result & Benchmarks
- Conclusions

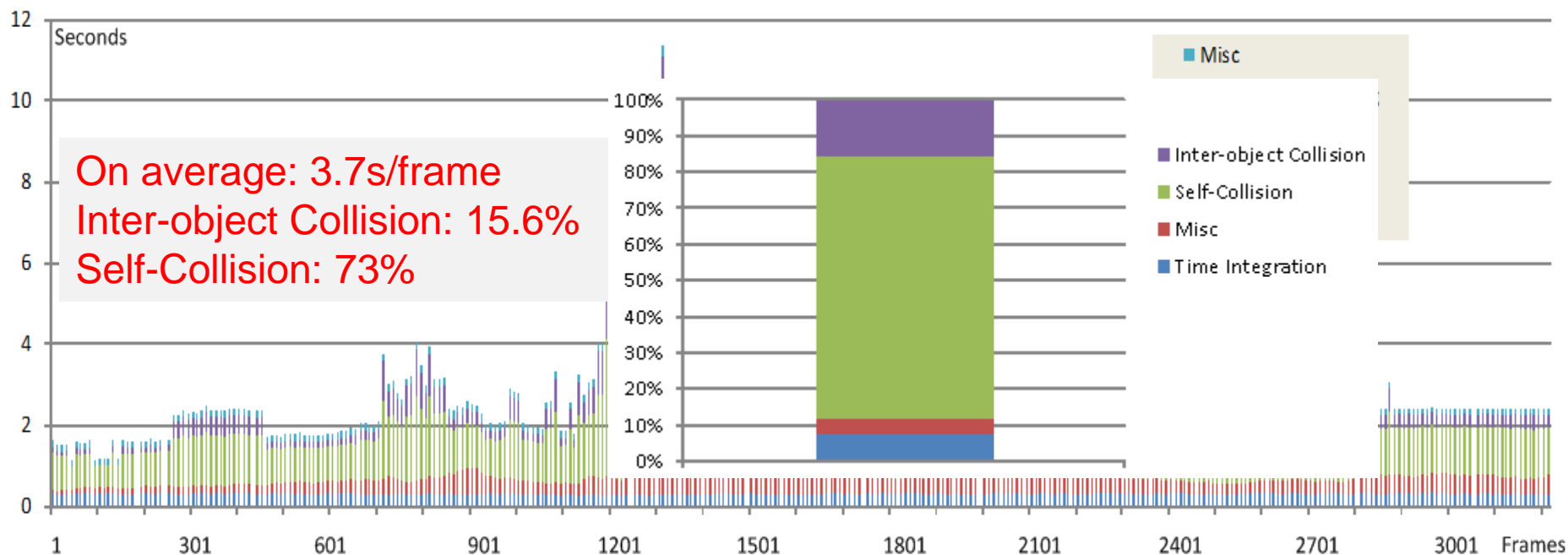
# Challenges

- Collision handling remains a major bottleneck in deformable simulation
- Major bottleneck in cloth simulation [Tang et al. 2016]
- Most parallel GPU-base collision detection algorithms do not perform self-collision culling [Tang et al. 2016; Weller et al. 2017]



# Challenges

- Collision handling remains a major bottleneck in deformable simulation
- Major bottleneck in cloth simulation [Tang et al. 2016]
- Most parallel GPU-base collision detection algorithms do not perform self-collision culling [Tang et al. 2016; Weller et al. 2017]



# Motivation

- We want to design an optimized collision handling scheme with following capabilities:
  - **Lower memory overhead:** most commodity GPUs have less than 6GB memory (e.g., NVIDIA GeForce GTX 1060)
    - CAMA runs on Tesla K40c with 12G memory [Tang et al. 2016]
  - **Faster collision detection:** A key bottleneck in interactive performance
    - CAMA needs 4-5s/frames for its benchmarks [Tang et al. 2016]
  - **Parallel cloth simulation:** should integrate with parallel, GPU-friendly deformable simulation algorithms

# Outline

- Motivation & Challenges
- Related Work
- Main Results
- Algorithms
  - Parallel Self-collision Culling
  - Extended Spatial Hashing
  - Optimized Cloth Simulation Pipeline
- Result & Benchmarks
- Conclusion

# Related Work

- Self-collision Culling
- Spatial Hashing on GPUs
- Parallel Cloth Simulation on Multi-core / Many-core Processors



# Related Work

- Self-collision Culling
  - Normal cone culling [Provot 1997, Schvartzman et al. 2010, Tang et al. 2009, Wang et al. 2017]
  - Energy-based culling [Barbic and James 2010, Zheng and James 2012]
  - Radial-based culling [Wong et al. 2013; Wong and Cheng 2014]
  - Most of them are serial algorithm running on single CPU core

# Related Work

- Spatial Hashing on GPU
  - Used for collision detection [Lefebvre and Hoppe 2006]
  - Uniform grids [Pabst et al. 2010] or two-layer grids [Faure et al. 2012]
  - Hierarchical grids [Weller et al. 2017]
  - No self-collision culling
  - Can be used for rigid and deformable simulation

# Related Work

- Parallel Cloth Simulation on Multi-core / Many-core Processors
  - Multi-core algorithms [Selle et al. 2009]
  - GPU parallelization for regular-shaped cloth [Tang et al. 2013]
  - CAMA: GPU streaming + Arbitray topology + robust collision handling [Tang et al. 2016]
  - Large memory overhead
  - Takes a few seconds per frame on a Tesla GPU

# Outline

- Motivation & Challenge
- Related Work
- **Main Results**
- Algorithms
  - Parallel Self-collision Culling
  - Extended Spatial Hashing
  - Optimized Cloth Simulation Pipeline
- Result & Benchmarks
- Conclusion

# Main Results

A GPU-based self-collision culling method; combines normal cone culling and spatial hashing:

1. Parallel self-collision culling based on normal cone test front;
2. Extended spatial-hashing for inter-object collisions and self-collisions;
3. New, optimized collision handling pipeline for cloth simulation.

# Benefits

1. Lower memory overhead: 5-7X reduction than prior methods
2. Faster GPU-based collision detection between deformable models: 6-8X faster
3. Faster cloth simulation algorithm on GPUs: 4-6X faster

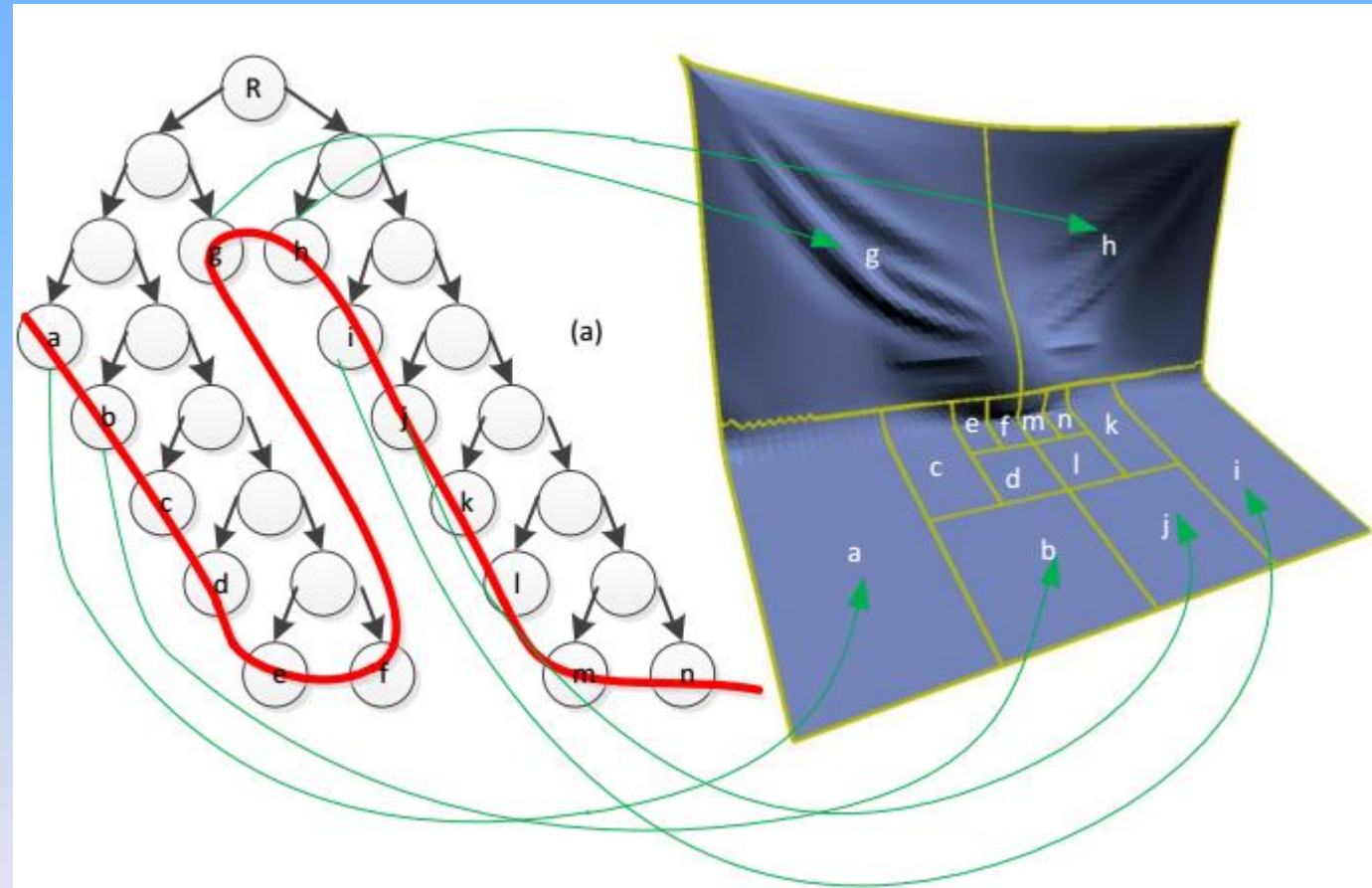
# Outline

- Motivation & Challenge
- Related Work
- Main Results
- Algorithms
  - Parallel Self-collision Culling
  - Extended Spatial Hashing
  - Optimized Cloth Simulation Pipeline
- Result & Benchmarks
- Conclusion

# Parallel Normal Cone Culling

Conventional top-down culling is replaced by parallel culling;

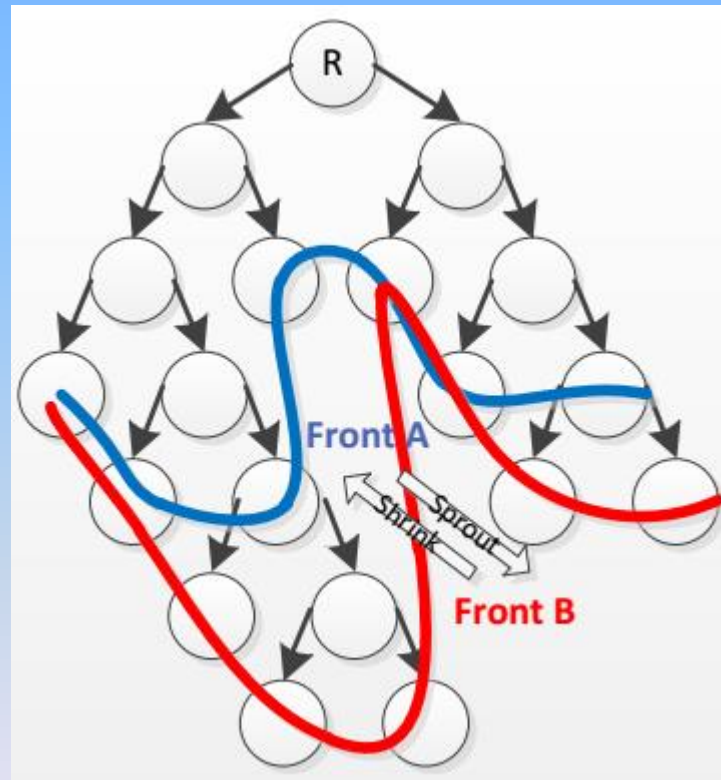
Maintain a Normal Cone Test Front (NCTF).



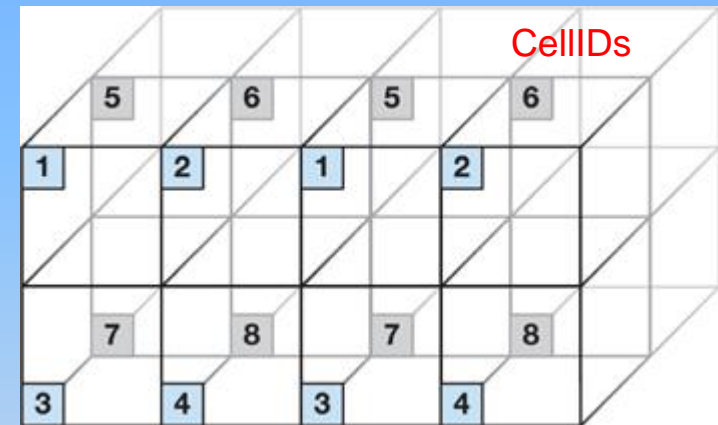
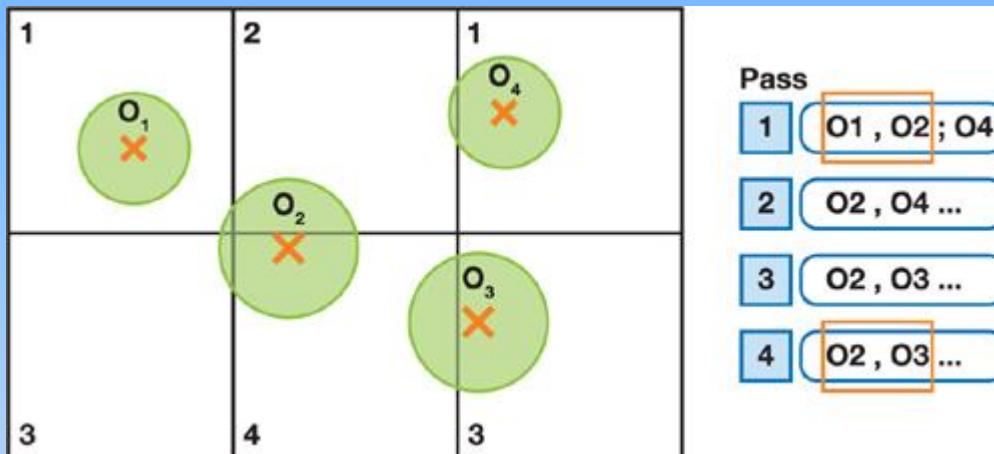


# Parallel Normal Cone Culling

Front update using  
sprouting and  
shrinking operators



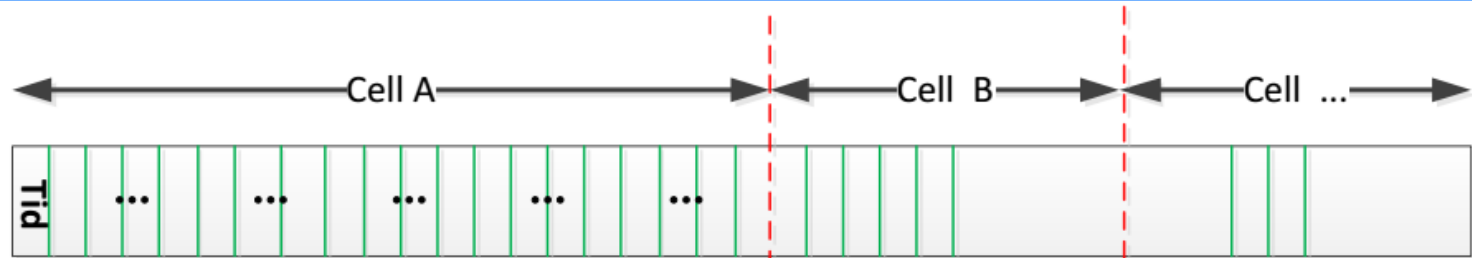
# Conventional Spatial Hashing



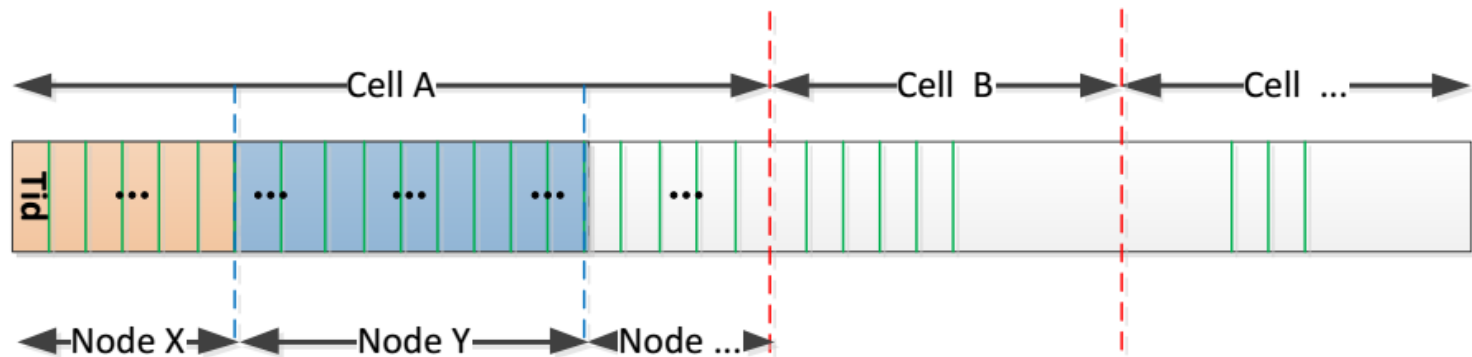
- Distribute all the objects into cells based on a hash function
- Intersection tests for all the objects in the same cell
- No self-collision culling between deformable objects

**GPU Gems 3: Chapter 32. Broad-Phase Collision Detection with CUDA,** *Scott Le Grand, NVIDIA.*

# Extended Spatial Hashing



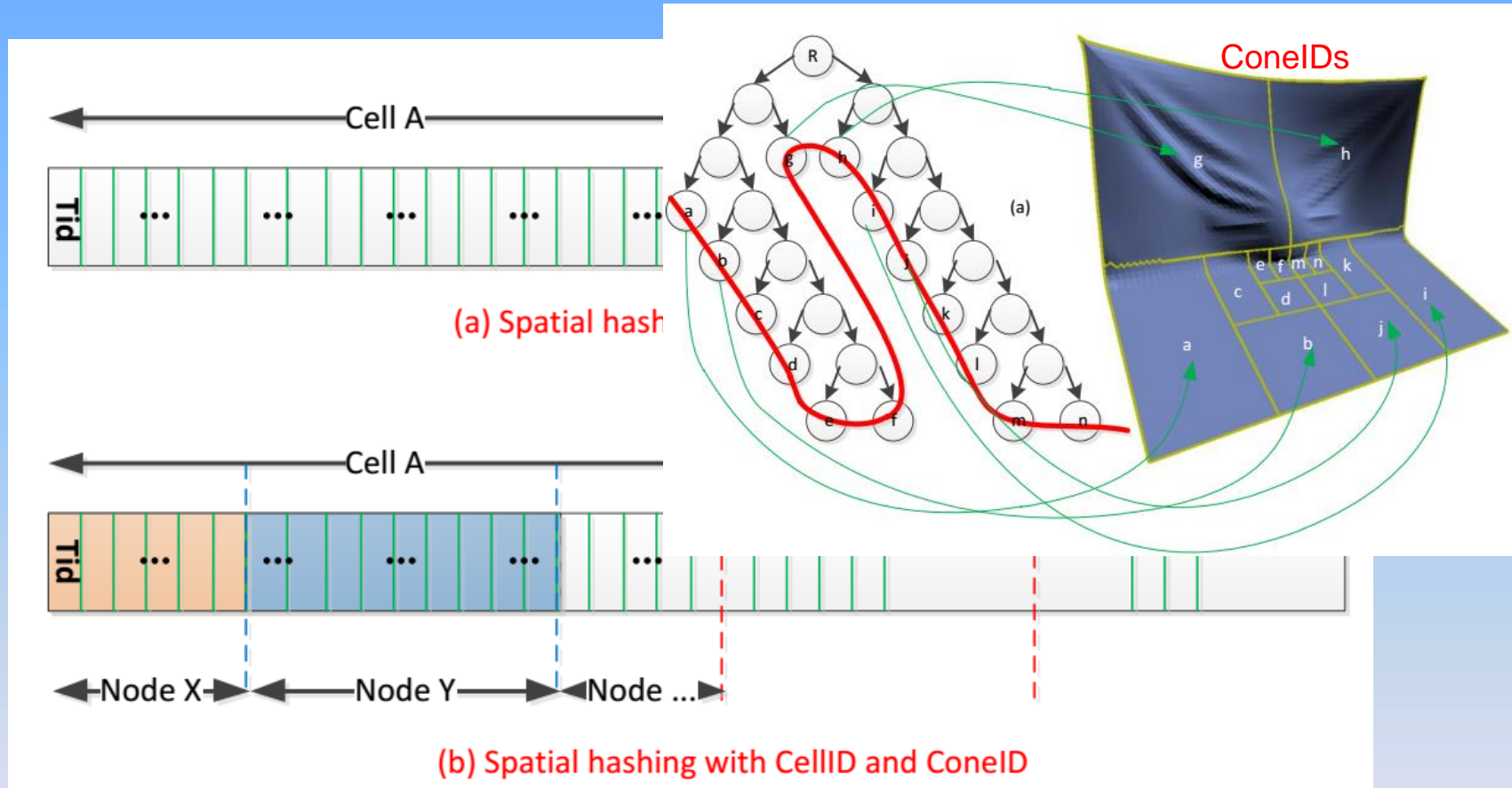
(a) Spatial hashing with CellID only



(b) Spatial hashing with CellID and ConeID

To perform both inter-object and intra-object collision culling, CellID (spatial information) and ConeID (normal cone information) are used as hash keys

# Extended Spatial Hashing



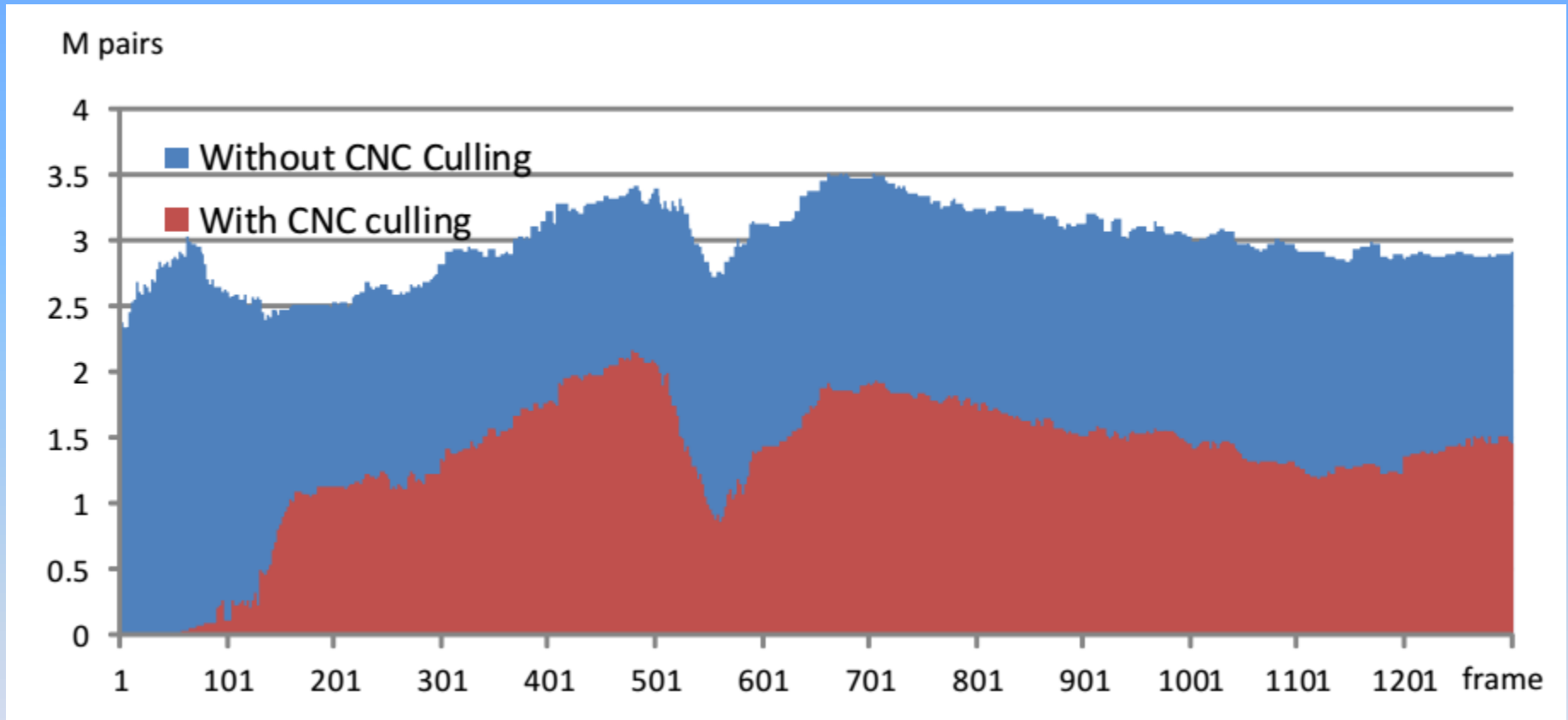
To perform both inter-object and intra-object collision culling, CellID (spatial information) and ConeID (normal cone information) are used as hash keys

# Extended Spatial Hashing

Fewer triangle pairs are tested for collisions: due to self-collision culling:

$$\begin{aligned} N_{inter} - N'_{inter} &= n_t * (n_t - 1)/2 - \sum_{i \neq j} n_i * n_j \\ &= [(\sum n_i)^2 - \sum n_i - 2 * \sum_{i \neq j} n_i * n_j]/2 \\ &= (\sum n_i^2 - \sum n_i)/2 \geq 0 \end{aligned}$$

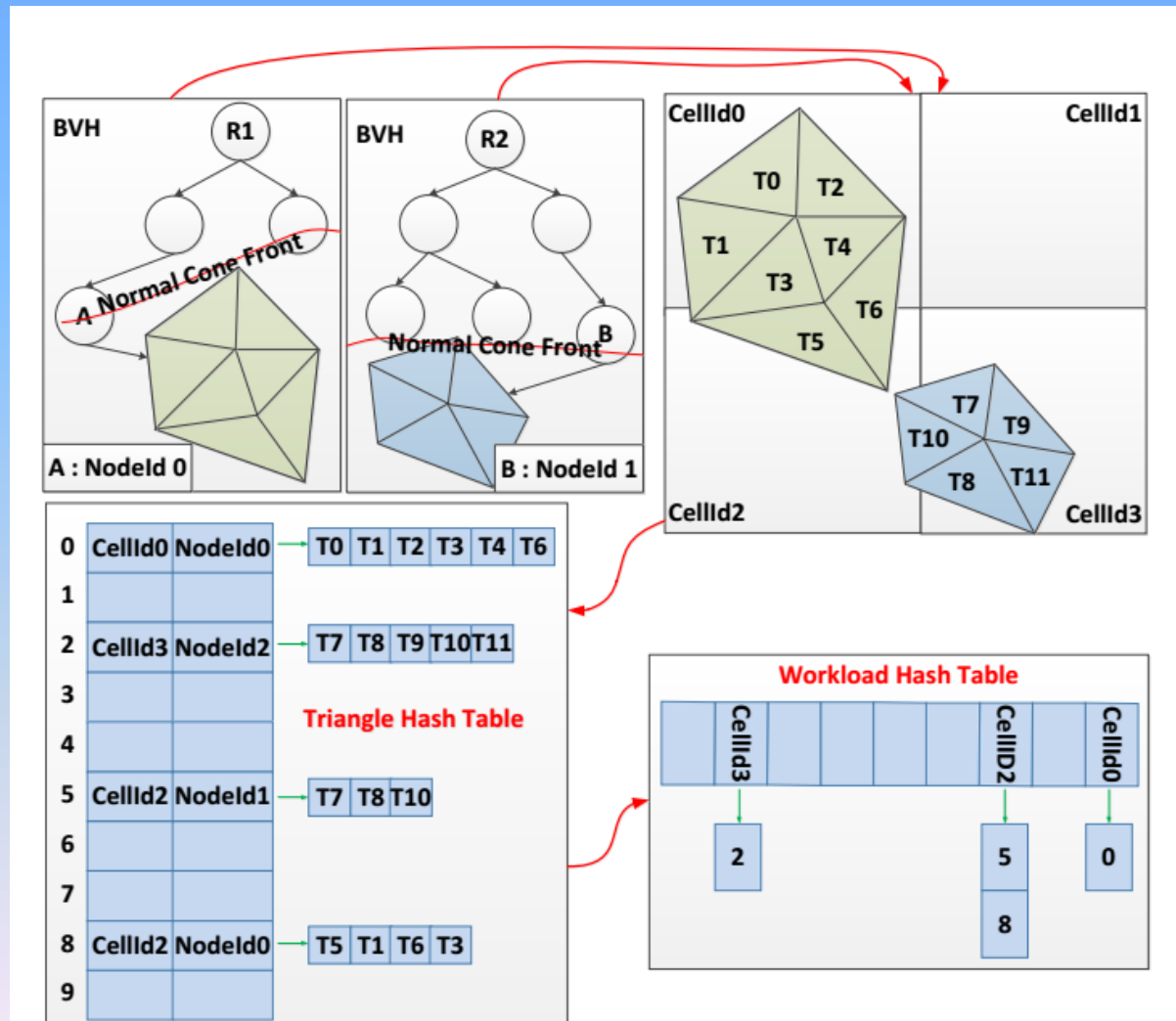
# Extended Spatial Hashing



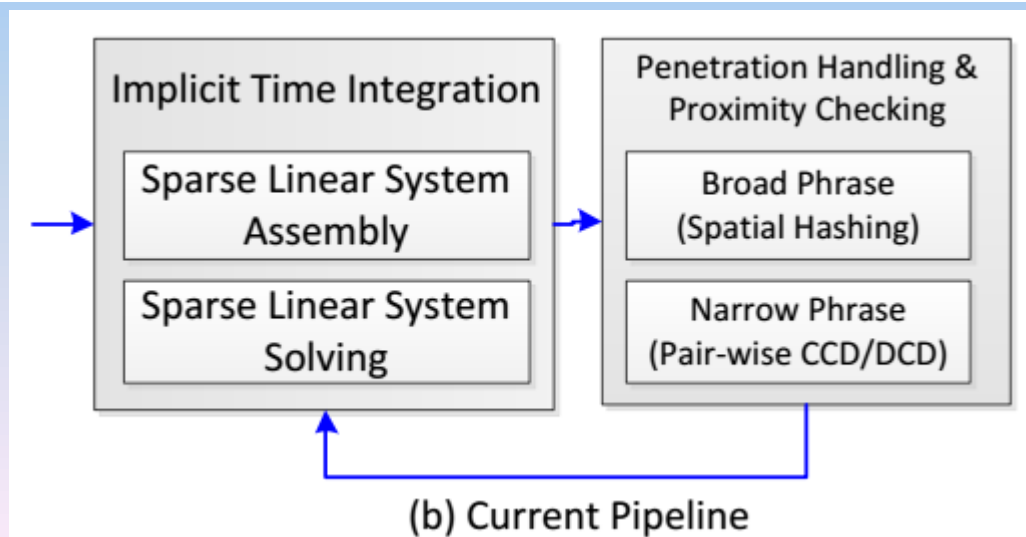
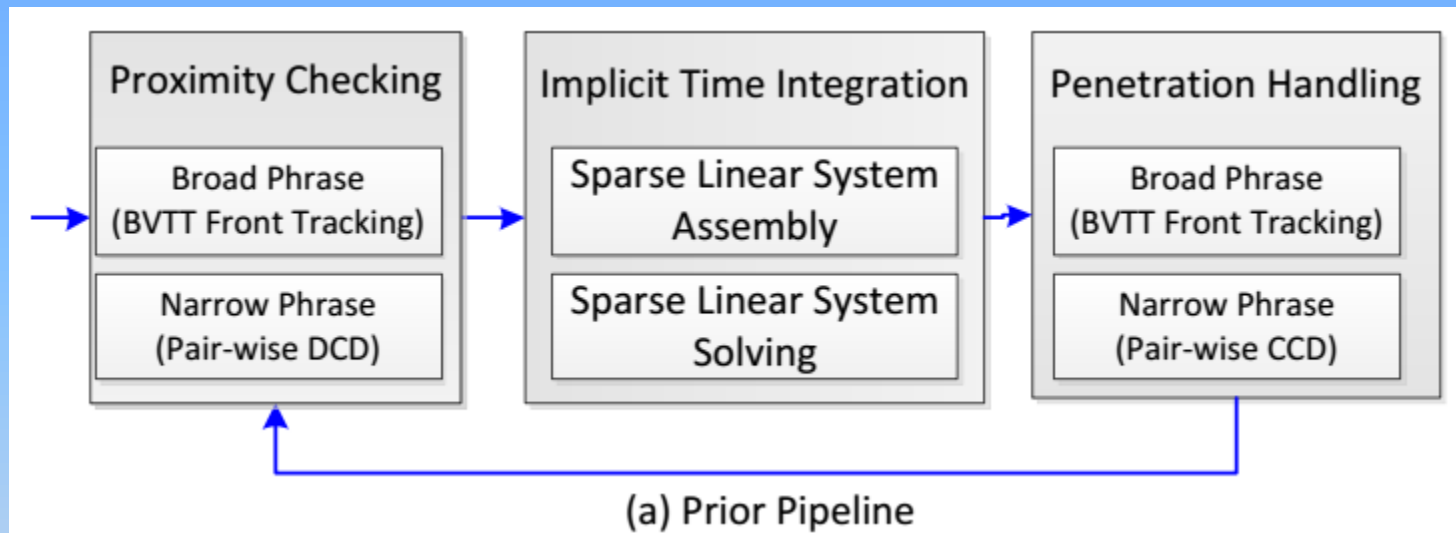
- Triangle pairs from broad phrase culling
- With and without CNC culling
- Fewer false positives with CNC culling

# Extended Spatial Hashing

- Building Workload Hash Table on GPU
- GPU-based sparse matrix assembly [Tang et al. 2016]

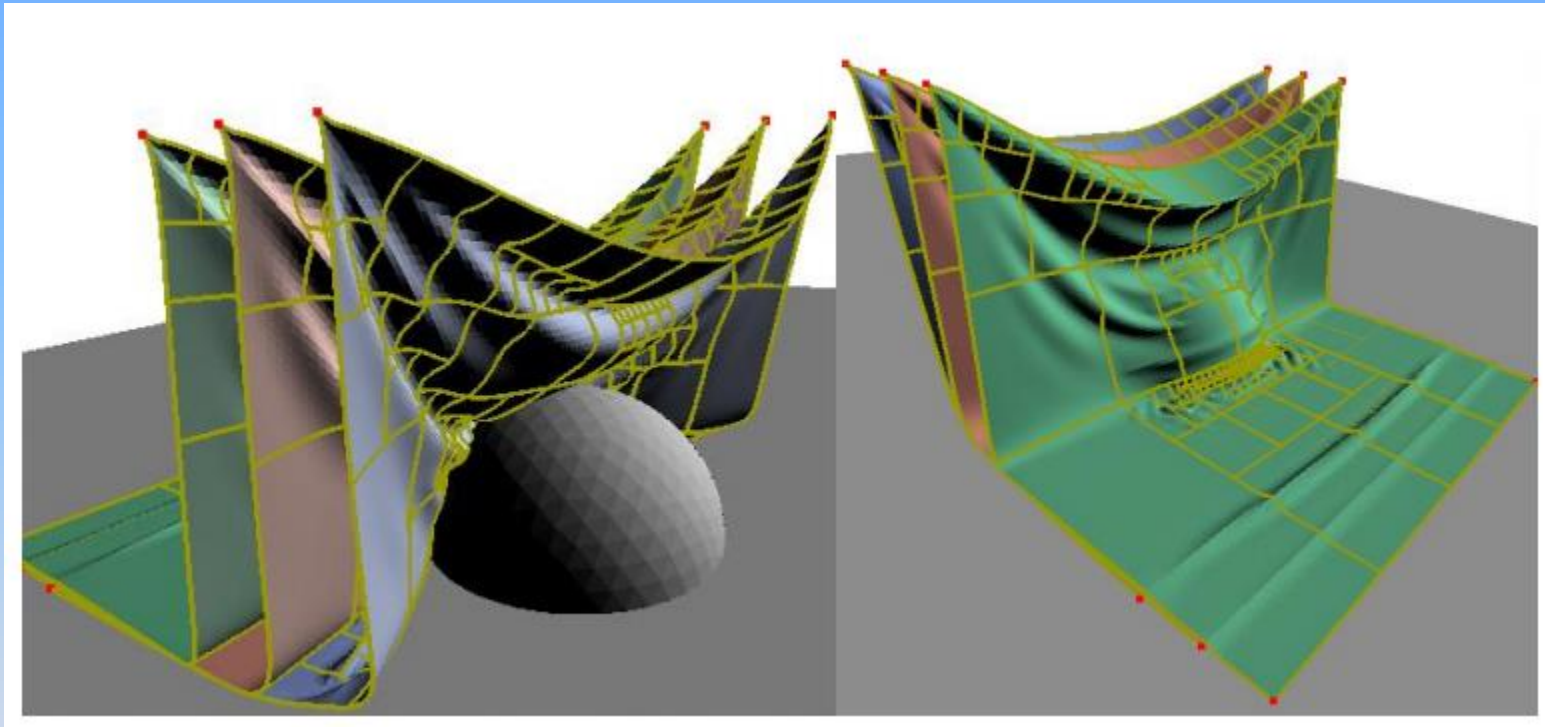


# New Collision Handling Pipeline





# New Collision Handling Pipeline



In this benchmark, the number of BV tests and running time of broad phrase culling is reduced by 51.1% and 53.3%, respectively.

# Outline

- Motivation & Challenge
- Related Work
- Main Results
- Algorithms
  - Parallel Self-collision Culling
  - Extended Spatial Hashing
  - Optimized Cloth Simulation Pipeline
- Results & Benchmarks
- Conclusions

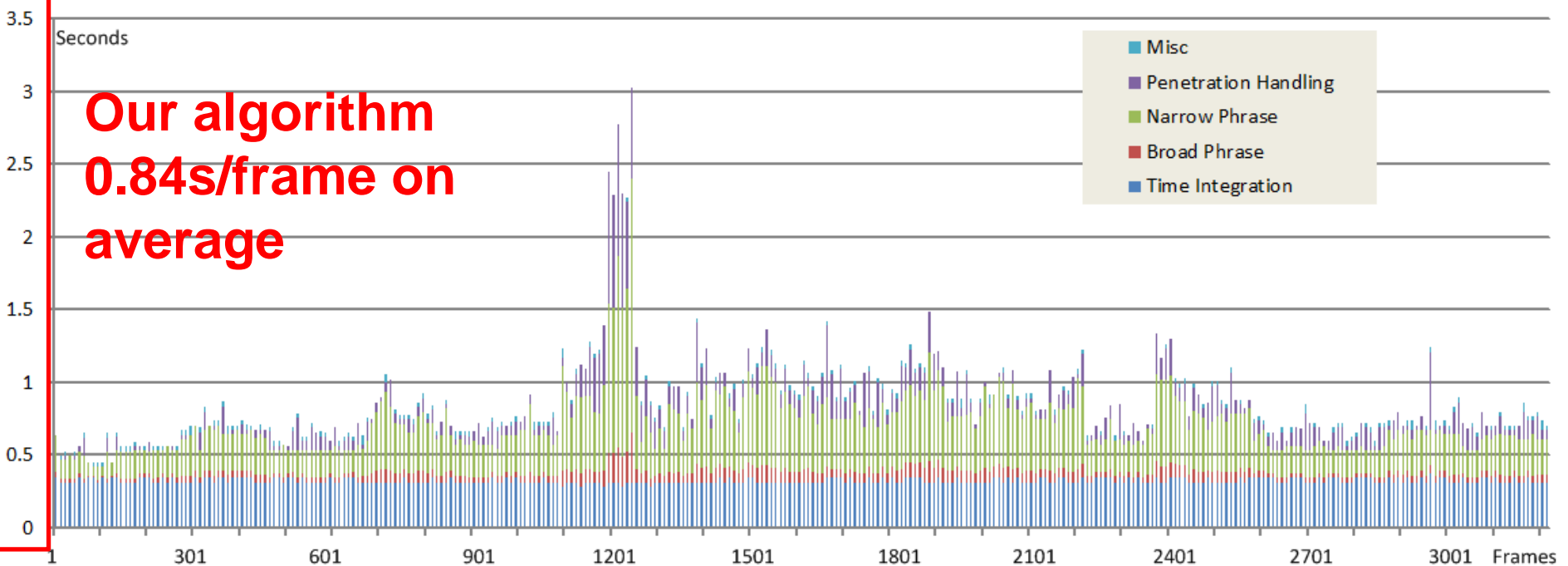
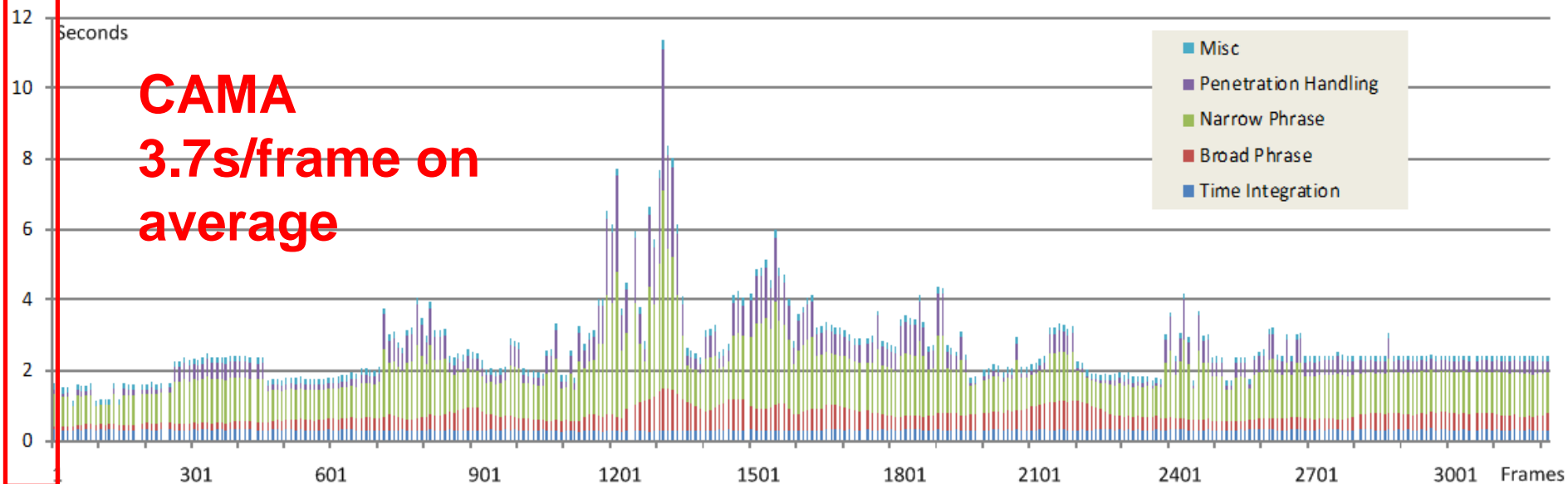
# Performance

- Evaluated on NVIDIA Tesla K40c, GeForce GTX 1080, and GeForce GTX 1080 Ti;
- Complex benchmarks: 80K-200K triangles
  - High number of inter-object and intra-object collisions & folds
- Less than 1 second per frame for cloth simulation on GTX 1080 and 1080 Ti
- Considerable speedups over prior algorithms

# Performance Comparison

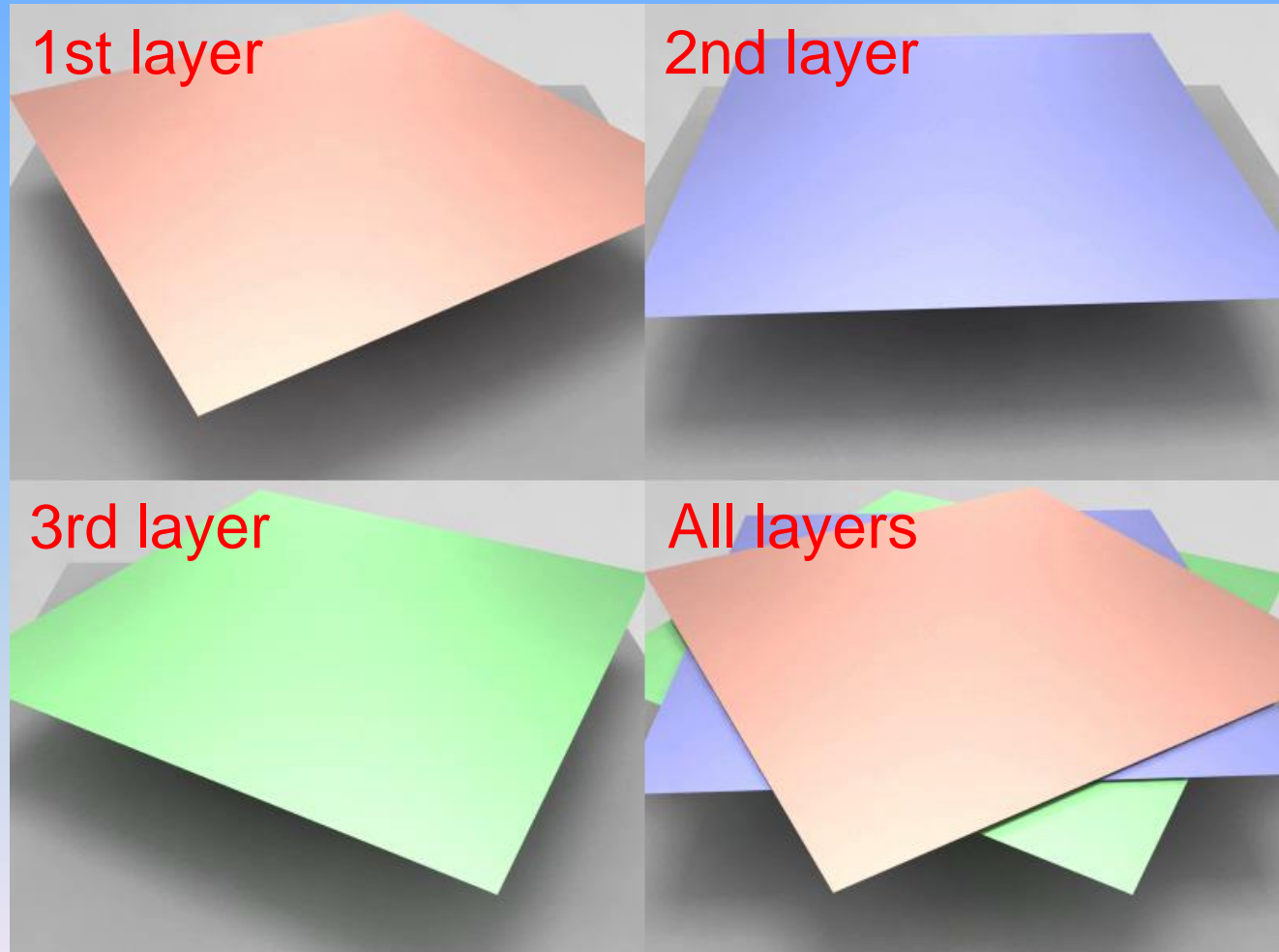
- Benchmark  
Andy
- 127K triangles
- Time step:  
 $1/25\text{s}$
- NVIDIA  
GeForce GTX  
1080
- Average cloth  
simulation time:  
 $0.84\text{s/frame}$
- Played at 24x  
speed





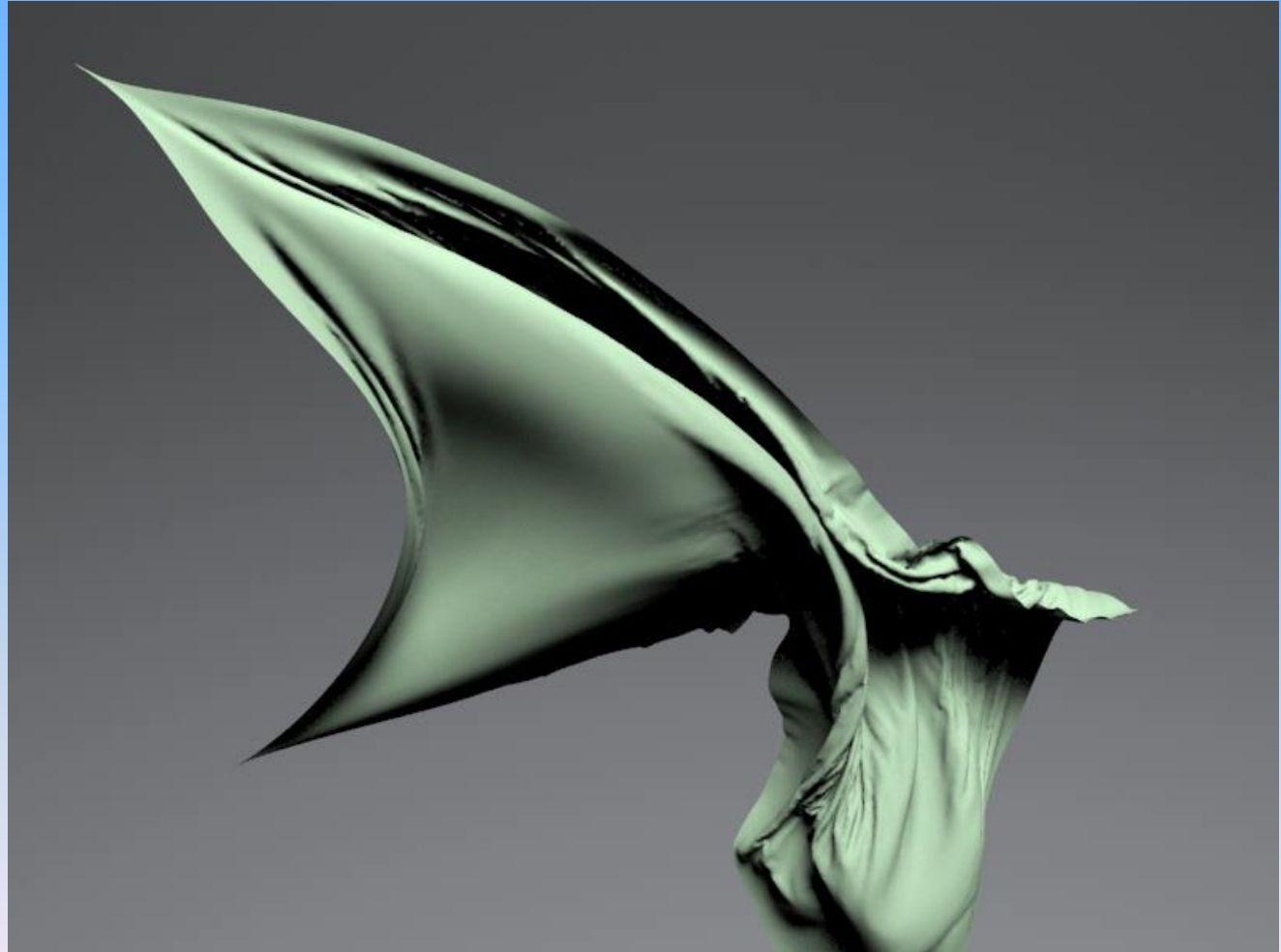
# Benchmark: Twisting

- 200K triangles
- Time step:  $1/200s$
- Multiple layers and contacts
- Average cloth simulation time: 0.97s/frame
- Played at 28x speed



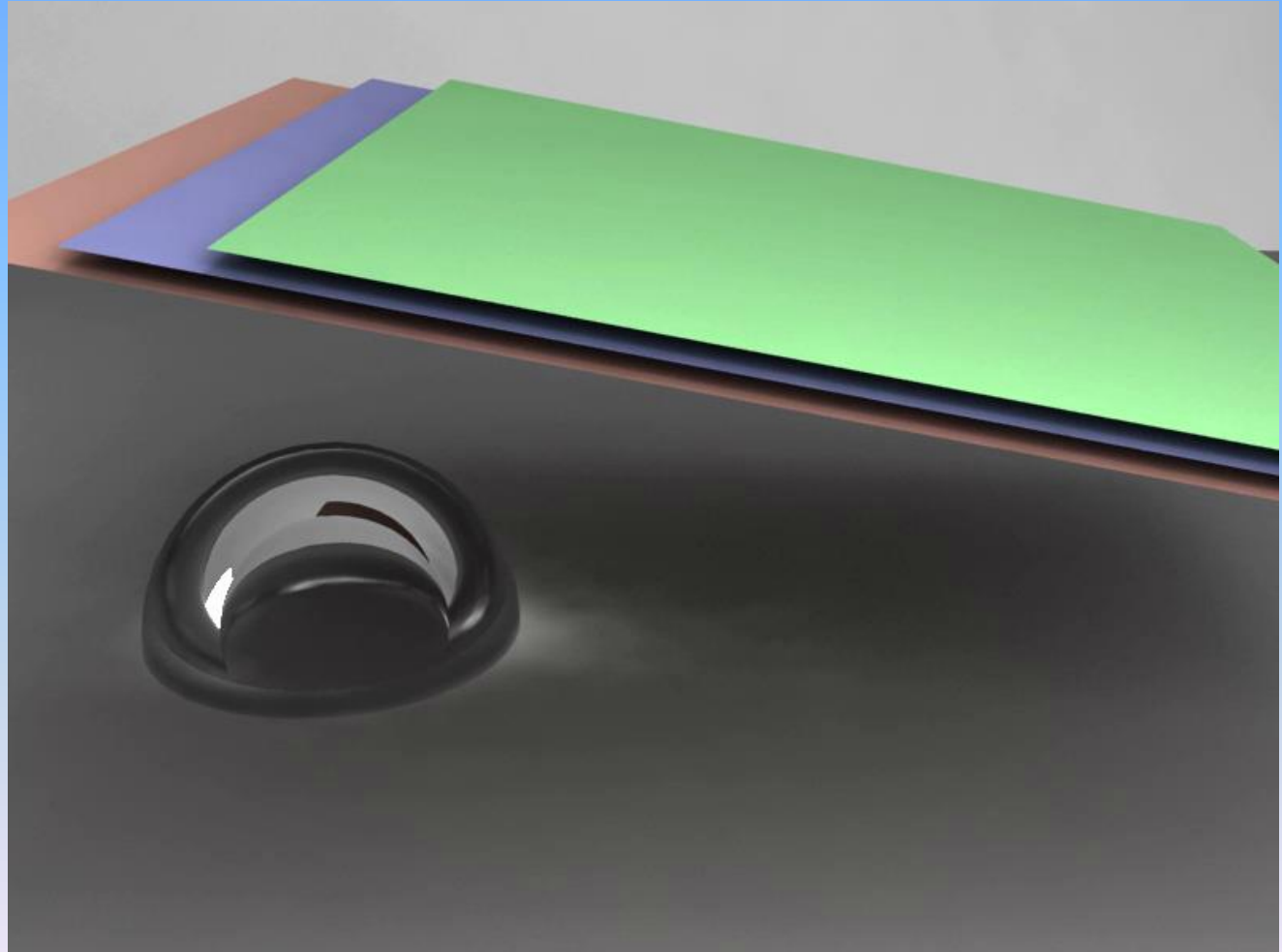
# Benchmark: Flag

- 80K triangles
- Time step:  
1/100s
- Multiple self-collisions
- Average cloth simulation time:  
0.35s/frame
- Played at 10x speed



# Benchmark: Sphere

- 200K triangles
- Time step:  
1/300s
- Multiple layers  
and contacts
- Average cloth  
simulation time:  
0.94s/frame
- Played at 26x  
speed





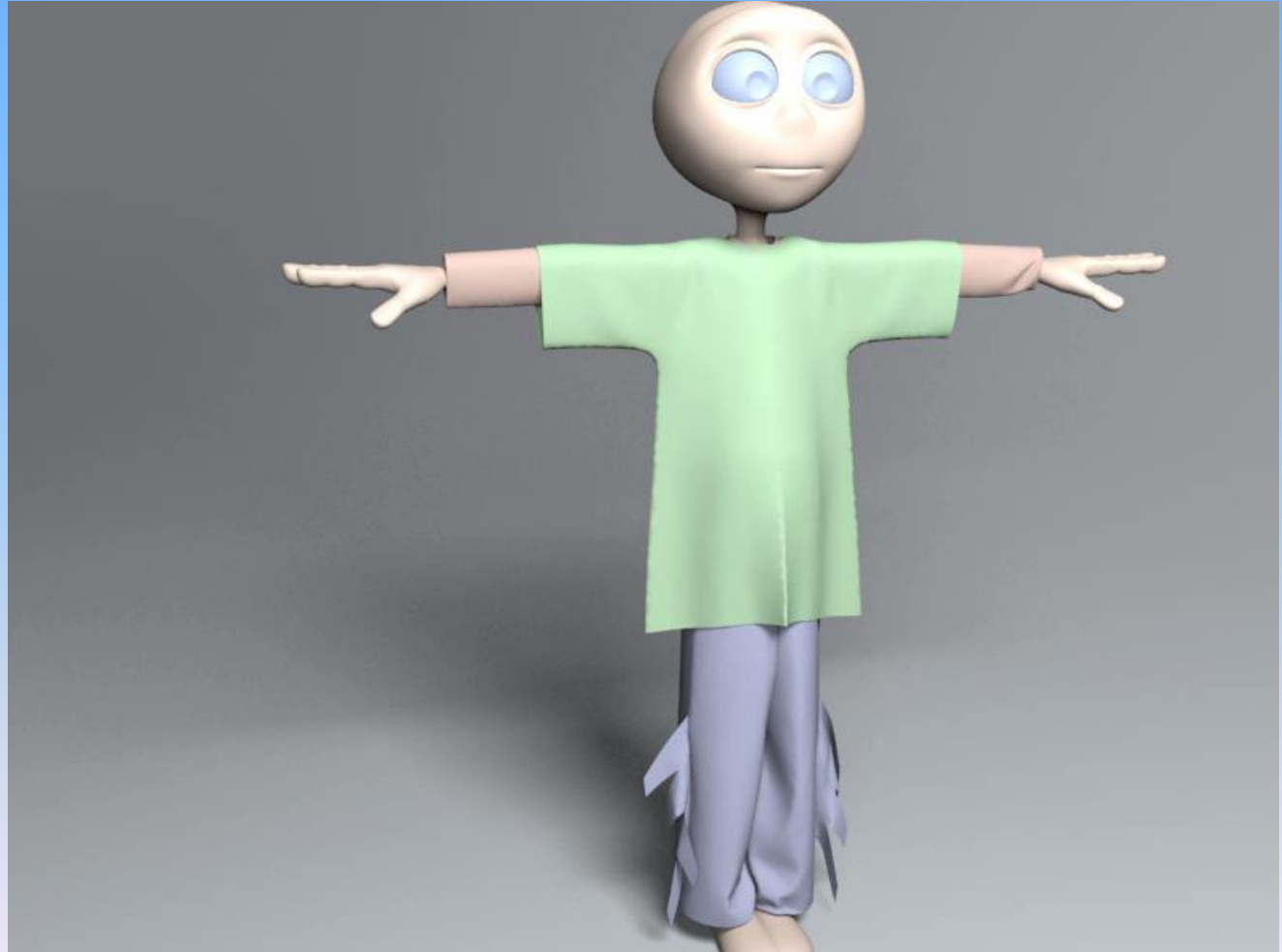
# Benchmark: Falling

- 172K triangles
- Time step:  $1/30$ s
- Multiple inter-object and intra-object collisions
- Average cloth simulation time: 0.51s/frame
- Played at 14x speed



# Benchmark: Bishop

- 124K triangles
- Time step:  
1/30s
- Multiple layers  
and contacts
- Average cloth  
simulation time:  
0.94s/frame
- Played at 26x  
speed



# Outline

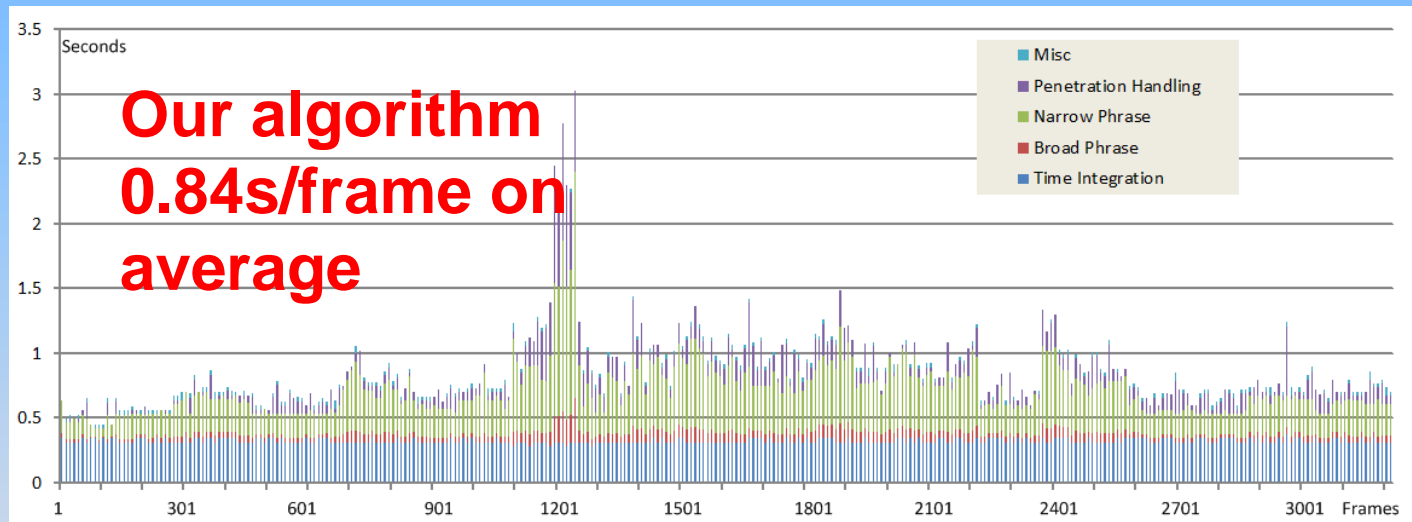
- Motivation & Challenge
- Related Work
- Main Results
- Algorithms
  - Parallel Self-collision Culling
  - Extended Spatial Hashing
  - Optimized Cloth Simulation Pipeline
- Results & Benchmarks
- Conclusions

# Main Results

- Novel parallel GPU-based self-collision culling algorithm;
- Considerable speedups over prior GPU-based algorithms;
- Almost real-time cloth simulation on complex benchmarks on a commodity GPU

# Limitations

- For tangled cloth, collision detection and penetration handling still remain a major efficiency bottleneck;



- For meshes undergoing topological changes, the normal cones and their associated contour edges need to be updated on-the-fly.

# Future work

- Faster collision handling
- Distance-field based collision handling
- Integration with cloth design and VR systems

# Acknowledgements

- National Key R&D Program of China (2017YFB1002703), NSFC (61732015, 61572423, 61572424), the Science and Technology Project of Zhejiang Province (2018C01080), and Zhejiang Provincial NSFC (LZ16F020003).
- 1000 National Scholar Program of China
- NVIDIA for hardware donation (NVIDIA Tesla K40c)
- Providers of all animation data

Q&A

Thanks!