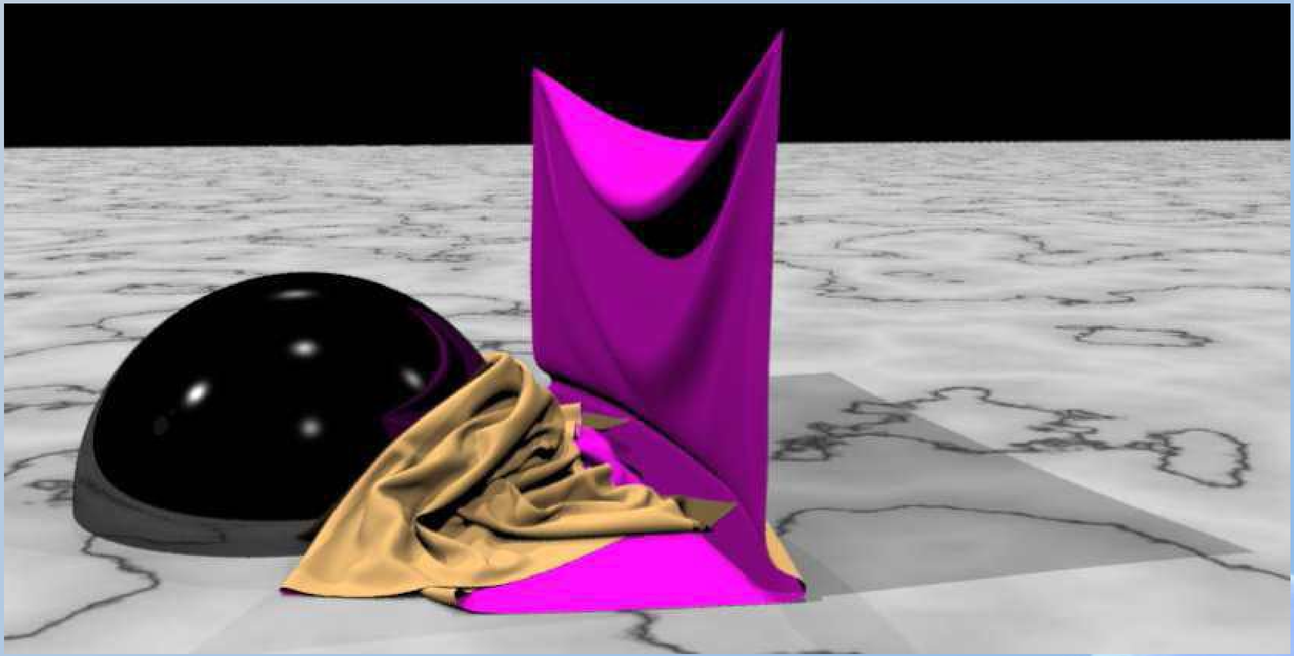
Optimized Spatial Hashing for Collision Detection of Deformable Objects

Teschner, Heidelberg, Muller, Pomeranets, Gross

2003



Introduction

- collision and self-collision detection of dynamically deforming objects
- generated hash table using hash function
- works with tetrahedrals meshes
- easily adapted to other primitives, such as triangles

Optimized Spatial Hashing for Collision Detection of Deformable Objects

Teschner, Heidelberg, Muller, Pomeranets, Gross

2003

Usage

- Cloth modeling
- Game engines
- Surgical simulators
- other physically based environments with up to 20k tetrahedrons in real-time

Optimized Spatial Hashing for Collision Detection of Deformable Objects

Teschner, Heidelberg, Muller, Pomeranets, Gross

2003

Collision detection algorithm

- all objects are classified to small 3D cells
- all tetrahedrons are classified with respect to these cells
- discretize minimum and maximum of all AABBs
- hash table of vertices and tetrahedrons
- intersection tests for vertices and tetrahedrons

Spatial hashing of vertices

- computed in first pass
- coordinates of vertex (x,y,z) are divided by the given grid cell size l and divided down to next integer
  - $(i=[x/l], j=[y/l], k=[z/l])$
- hash function maps discretized positions (i,j,k) to 1D index h
- Vertex and object information is stored in hash table with indexes $h=hash(i,j,k)$

Hash function

- gets three values describing vertex position
- return hash value

$$hash(x,y,z) = (x\,p1 \textbf{ xor } y\,p2 \textbf{ xor } z\,p3) \textbf{ mod } n$$

  - $p1,p2,p3$ are large prime numbers
  - n is the hash table size
- the quality of the hash function is less important for larger hash tables

Spatial hashing of tetrahedrons

- discretize minimum and maximum values describing the AABB of tetrahedron
- values are divided by cell size and rounded down to integer
- hash values are computed for all cells affected by the AABB of a tetrahedron
- all vertices found at the according hash table index are tested for intersection

Intersection tests

- using barycentric coordinates
- if Vertex penetrates Tetrahedron
  - detect Collision
- if Vertex penetrates Tetrahedron and both belong to same object
  - detect Self-Collision
- if Vertex is part of Tetrahedron
  - test is omitted

Actual Intersection tests

- if Vertex p and Tetrahedron t are mapped to the same hash index and p is not part of t
  - perform Penetration test
- check p against AABB of t whether p is inside t with vertices at positions (x0,x1,x2,x3)
- Barycentric-coordinate test is slightly faster than the half-space test

Optimized Spatial Hashing for Collision Detection of Deformable Objects

Teschner, Heidelberg, Muller, Pomeranets, Gross

2003

Barycentric coordinates test

- express p with new coordinates $\beta = (\beta_1, \beta_2, \beta_3)^T$
  - with respect to x_0 axis coincide with the edges of t adjacent to x_0
  - $p = x_0 + A\beta$
  - $A = [x_1 - x_0, x_2 - x_0, x_3 - x_0]$
  - $\beta = \mathbf{A}^{-1}(\mathbf{p} - \mathbf{x}_0)$
- if $\beta_1 \geq 0, \beta_2 \geq 0, \beta_3 \geq 0$ and $\beta_1 + \beta_2 + \beta_3 \leq 1$
  - p lies inside tetrahedron t

Grid cell size

- larger cells increase number of primitives in hash index, slows down intersection test
- cell size should have size of the average length off all tetrahedrons
- grid cell size has a bigger effect on the performance than hash function or hash table size

Hash table size

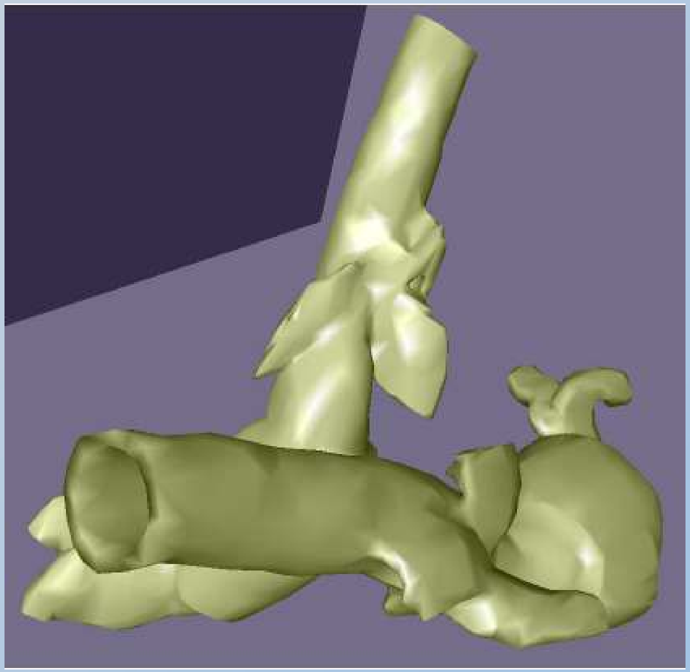
- larger table size
  - reduce the risk of mapping different 3D positions to the same hash index
  - algorithm works faster
  - the performance slightly decreases
- larger hash table size than number of object primitives minimalize the hash collisions risk
- not require re-initialization in each step, using time stamps in hash table cells

Optimized Spatial Hashing for Collision Detection of Deformable Objects

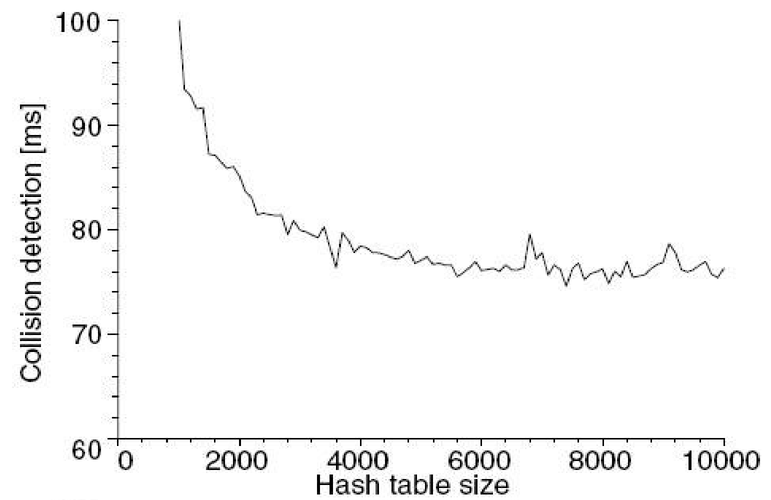
Teschner, Heidelberg, Muller, Pomeranets, Gross

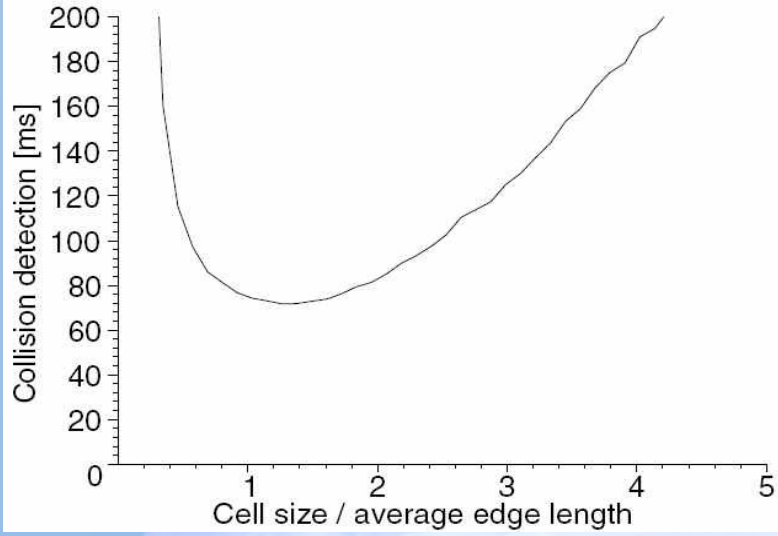
2003

Example 1



two deformable objects with an overall number of 5898 vertices and 20514 tetrahedrons



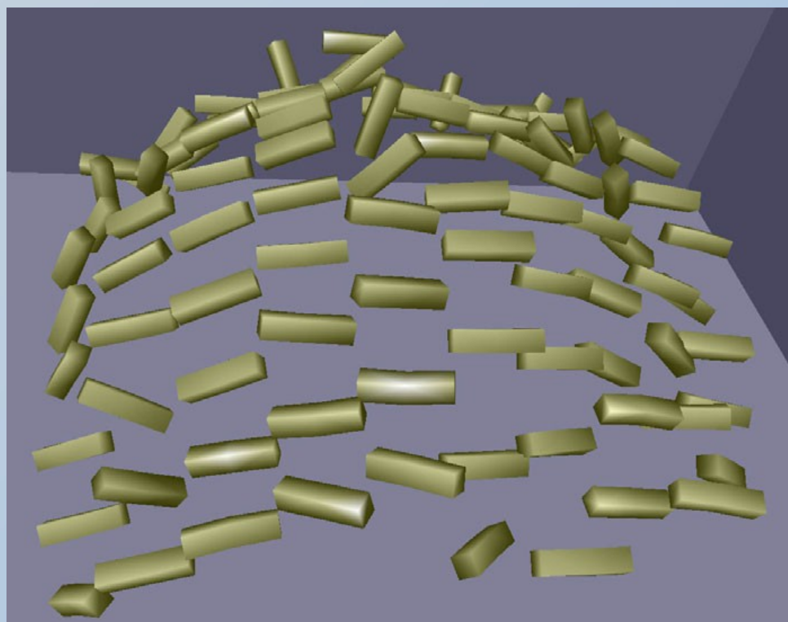


Optimized Spatial Hashing for Collision Detection of Deformable Objects

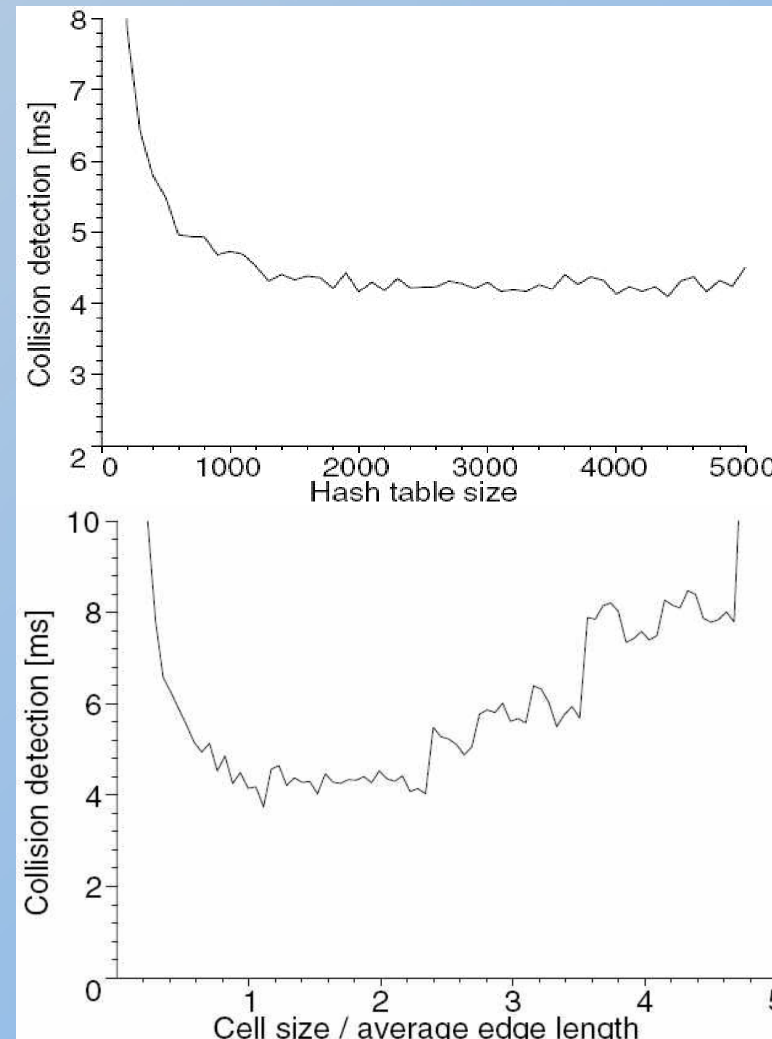
Teschner, Heidelberg, Muller, Pomeranets, Gross

2003

Example 2



100 deformable objects with an overall number of 1200 vertices and 1000 tetrahedrons



Time complexity

- Time complexity: $O(n^2)$, goal: $O(n)$
  - n *is number of primitives*
- first pass - insert vertices into hash table: $O(n)$
- *second pass* - $O(n.p.q)$
  - p *is* the average number of cells intersected by a tetrahedron
  - q is the average number of vertices per cell
  - *choose cell size to by proportional to average tetrahedron size =* p *is constant*
  - *no hash collisions =* q *is constant too*

Optimized Spatial Hashing for Collision Detection of Deformable Objects

Teschner, Heidelberg, Muller, Pomeranets, Gross

2003

Results of the algorithm

- Performance of the collision detection algorithm
- The performance is independent from the number of objects. It only depends on the number of object primitives.
- Average collision detection time, minimum, maximum, and standard deviation for 1000 simulation step

setup	objects	tetras	vertices
A	100	1000	1200
B	8	4000	1936
C	20	10000	4840
D	2	20514	5898
E	100	50000	24200

setup	ave [ms]	min [ms]	max [ms]	dev [ms]
A	4.3	4.1	6.5	0.24
B	12.6	11.3	15.0	0.59
C	30.4	28.9	34.4	1.25
D	70.0	68.5	72.1	0.86
E	172.5	170.5	174.6	1.08

Defect of the algorithm

- presented algorithm does not detect, whether an edge intersects with tetrahedron due to two reasons
  - first: the relevance of an edge test is unclear in case of densely sampled objects
  - second: it is rather uncommon and costly to implement collision response in case of penetrating edges

Optimized Spatial Hashing for Collision Detection of Deformable Objects

Teschner, Heidelberg, Muller, Pomeranets, Gross

2003

The End

©Stanislav Miklóšik, 2011