spatial hashing introduces hash collision and non-overlapping cells are mapped in the same voxel. Traditionally a collision-detection check is performed between all triangles in a voxel. Employing a perfect spatial hash function can reduce this problem to some degree. But the different resolutions of objects still cause irregular voxels. More importantly, a perfect spatial hash function reduces the overall performance of the spatial hashing method due to overhead costs of memory management (see the result from Teschner et al [2]).

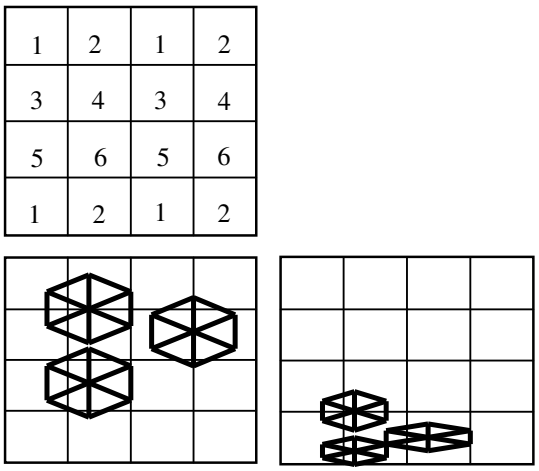


Figure 1. Spatial subdivision cells in a regular grid: the number is the index of the voxel and the total voxels are six (top), a well distributed case (bottom left), and a squeezed case (bottom right).

These problematic assumptions can be removed by using non-uniform spatial subdivision, but this decays the overall performance of the spatial subdivision method. For example, we implemented a spatial hashing method that uniformly distributes cells among voxels using sorting, but this requires O(nlgn) asymptotic run time. Using an r-tree structure might be an option, but it cannot keep O(n) performance steady. These problems are particularly apparent when objects are under a heavy load with a prolonged large deformation and when a large number of objects are concurrently engaged in compound collisions.

We maximize the spatial hashing method for spatial subdivision method by abbreviating these problems with a simple group hashing. Our method can be easily implemented into general spatial hashing techniques and it achieves optimal performance by balancing between the dispersion of cells and the cost of subdivision through changing the group size.

## 2. RELATED WORKS

### 2.1 Collision Detection for Deformable Objects

Collision detection among deformable objects is well studied in survey by Jimenez et al [3] and Teschner et al [4]. Collision detection can be divided into broad phase detection and narrow phase detection. While the computational cost is the main issue in broad phase, the accuracy is the main topic in narrow phase. Our method is categorized as broad phase detection.

### 2.2 Spatial Subdivision Using Hashing

Spatial hashing has been successfully employed in the spatial subdivision method for collision detection of rigid body simulation and deformable object simulation many researchers [1], [2], [5], [10]. A spatial hashing for deformable objects proposed by Teschner et al. [2] used a limited size of data structure and a simple spatial hash function for tetrahedron. They computed only static collision contact triangle and barycentric coordinates. Recently Eitz and Lixu [11] introduced a hierarchical spatial hashing using one dimensional array with XOR hash function. Their method ran in a consistent for different cell sizes with a slight computational overhead, but hashing collision that reduces the overall performance in hashing was not considered seriously. A perfect spatial hashing was tried using preprocessing and applied to

collision detection between objects by Lefebvre and Hoppe [6]. A perfect hash function could not resolve the clods of triangles in a voxel because although each voxel will take one spatial cell, it requires a large number of voxels and the number of voxels unknown a priori. Moreover, the irregularity of the number of triangles in voxels is inevitable in regular voxel grids because the mesh has different resolutions. All previous methods decomposed the problem for static collision detection without considering the clods of triangles. However, this is a significant problem for deformable object collision handling because the dispersion of triangles of an object can be altered continuously. We extend spatial subdivision to continuous collision detection by solving the clods of triangles with group spatial hashing to abbreviate the hash collision effect.

Our contribution is maximizing the spatial subdivision method for deformable objects using group hashing, which reduces the irregularity of the number of triangles in voxels.

## 3. COLLISION DETECTION USING GROUP SPATIAL HASHING

Our algorithm consists of following steps: updating each triangle's AABB, voxelization (mapping AABB of triangles into the fixed number of voxels), and performing an exact collision test (continuous collision detection and distance calculation) for the possible collision pair triangles.

### 3.1 AABB Update

Since our method performs exact collision detection for both continuous collision detection and distance calculation, the AABB of each triangle covers the entire trace of the triangle in a given period and the threshold of distance calculation is added. The result AABB is slightly larger than for static collision detection on it, yet overall an AABB update is quick because there is no hierarchy.

### 3.2 Group Spatial Hashing

In order to voxelize objects, the size of the cell and the number of voxels must be known before mapping the triangles into the voxels. The number of voxels and the size of the cell can be computed while updating AABB of triangles. In general, a hash function maps one cell to one voxel in perfect hashing and multiple cells to one voxel in hashing collision. In our case, the size of cell and the number of voxels are computed in the beginning of each simulation once and keep it constant during the simulation. Figure 2 illustrates the group mapping algorithm. The optimal cell size is the length of the largest axis of AABB and the optimal number of voxels is the number of triangles divided by five.

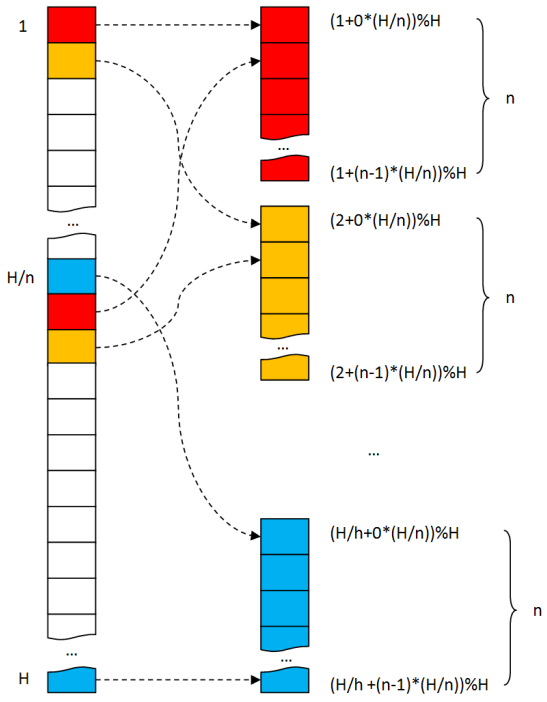


Figure 2. Group mapping: H voxels are grouped into n groups using a simple index calculation.

We compared the performance with a different number of voxels and the result shown in Figure 3. When objects in the scene do not move, the objects can be voxelized once and keep the information for each voxelization.

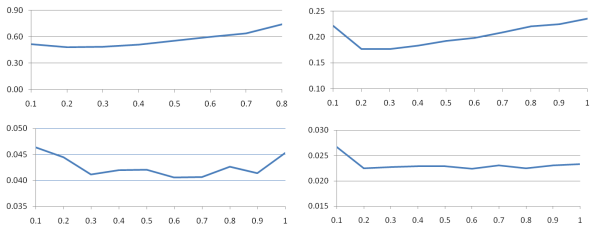

Figure 3. Performance comparison according to the number of voxels: the computing time for the whole process of collision detection is recorded according to the number of voxels (x axis is the number of voxels proportional to the number of triangles and y axis is second). Different numbers of triangles in a scene are tested: **(left top)** cloth and bunny simulation example with 90K triangles, **(right top)** cloth example with 20K triangles, **(left bottom)** 14 tori example with 11,200 triangles, and **(right bottom)** a 7 tori example with 5,600 triangles. The result confirms that the optimal number of voxels is about 20 percent of the number of triangles. **(left top)** and **(right top)** are the result on 1 node and **(left bottom)** and **(right bottom)** are shown here are results on 4 nodes.

To reduce the dispersion of the number of triangles in voxels, a hash function collision resolution method can be applied such as chaining and open addressing [7], [8]. Chaining requires extra dynamic storages and cannot keep up $O(n)$ performance. The open addressing scheme can be expensive too when the number of voxels is too small compared the number of primitives because the open addressing scheme searches throughout the voxels for an empty voxel. Our goal is not resolving hash function collision, but distributing the triangles evenly in the voxels. One voxel can contain triangles from more than one cell, yet the numbers of triangles should be evenly distributed. Each voxel contains the mapped cell information after it is mapped once.

Group hashing consists of three steps. First, each triangle is divided into the size of the cell and is found the index of each cell using a spatial hash function. Second, a set of possible indices of voxels for the triangle is calculated. If one of the selected voxels is empty or contains the cell, add the triangle into that voxel. Third, if the triangle is not added yet, the numbers of triangles in the three voxels are compared and the triangle is added into the voxel with the smallest number of triangles and the cell is mapped to the voxel. As n is increasing, the computational cost for spatial hashing is growing but the more evenly distributed problem sets are produced. Since the computational cost for voxelization increases along with the number of groups, we selected a small number for n (e.g., 4), which optimize the overall collision detection process. The group mapping algorithm is in algorithm 1.

Algorithm 1 GROUP MAPPING (*cell*)

```
calculate index using hash function for cell
calculate group indices
smallest ← index
isMapped ←false
for all i in group indices and isMapped ≠ true do
    if cell is mapped to VoxelList[i] then
        map cell to VoxelList[i]
        isMapped ← true
    end if
    if VoxelList[smallest].length> VoxelList[i].length
then
        smallest ← i
    end if
end for
if isMapped ≠ true then
   map cell to VoxelList[smallest]
end if
```

### 3.4 Exact Collision Detection

The possible triangle pairs are checked to find

the exact collision time and the collision location (barycentric coordinates). Fifteen continuous collision detections and 15 static collision detections are performed 9 for edge/edge and 6 for triangle/point. For continuous collision detections, coplanar condition is used with optimized numerical computation and removed from the redundant collision detection pairs. The collision detection result (e.g., collision time: dt and collision location: barycentric coordinates) is returned for collision resolution.

## 4. EXPERIMENTS

Our method was applied to various models and simulation scenarios. Our experiments conformed that when the number of voxels is large enough, our method produces similar results with the perfect hash function. The size of the group for group spatial hashing was 3 for all example simulations. Even a small size of the group removes the extreme case of voxels efficiently and improves overall performance.

The numbers of triangles in voxels are compared in terms of their dispersion and maximum number.

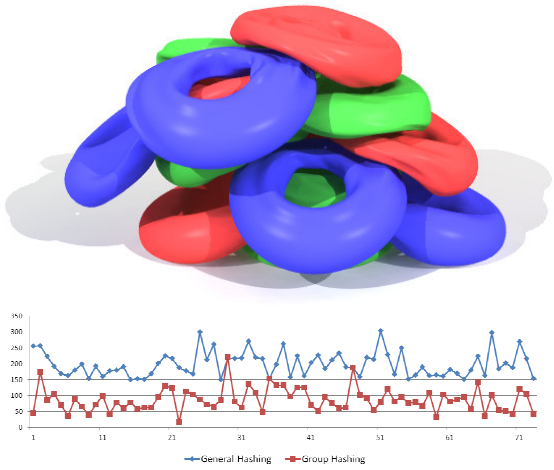


Fig. 4. 14 Tori simulation: snapshot (top), the comparison of the numbers of triangles in 73 highly dense voxels that have more than 150 triangles (bottom).

In this example, a torus model contains 800 triangles. Since the size of triangles in a torus varies, clods of triangles are inevitable within a model. This simulation contains 11.2K triangles. Figure 4 shows squeezed tori under prolonged contact and deformation and the comparison of the number of triangles in highly dense voxels. The maximum number of triangles in a voxel with an imperfect hashing method is 304, but it is reduced to 268 with our group mapping method. The group mapping method does not reduce the dispersion significantly because the clod of triangles is caused by resolution variation, however, inner culling works efficiently to remove unnecessary possible collision pairs for exact collision detection.

## 5. CONCLUSION

We present a fast and efficient voxelization method to reduce the irregularity of collision with a group mapping spatial hashing. Our method improves the overall performance of voxel-based collision detection by one order of magnitude. Even though we performed continuous collision detection and distance query at the same time, the performance of our method is comparable to the discrete Voronoi method by Sud et al [9] and our method has no limitations in topology and accuracy. Our approach not only substantially improves the performance but it is quite easy to implement and integrate into existing dynamic simulation environments. The benefit of our method enlarges the applicable deformable objects simulation from animation to medical simulation. Currently, our method does not consider topology changes such as object fracturing while processing, and considering this would be a natural extension for the next step.

ACKNOWLEDGMENT

This research was partially supported by National Science Foundation Career Award ACI-0238521.

## REFERENCES


[1] J. Dingliana and C. O'Sullivan, "A voxel-based approach to approximate collision handling," *J. Graphics Tools*, vol. 10, no. 4, pp. 33–48, 2005.

[2] M. Teschner, B. Heidelberger, M. Mueller, D. Pomeranets, and M. Gross, "Optimized spatial hashing for collision detection of deformable objects," *Proc. VMV'03*, pp. 47–54, 2003.

[3] P. Jimenez, F. Thomas, and C. Torras, "3D collision detection: A survey," *Computers and Graphics*, vol. 25, no. 2, pp. 269–285, 2001.

[4] M. Teschner, S. Kimmerle, G. Zachmann, B. Heidelberger, L. Raghupathi, A. Fuhrmann, M.-P. Cani, F. Faure, N. Magnetat-Thalmann, and W. Strasser, "Collision detection for deformable objects," *Computer Graphics Forum*, vol. 24(1), pp. 61–81, 2005.

[5] F. Ganovelli, J. Dingliana, and C. O'Sullivan, "Buckettree: Improving collision detection between deformable objects," *Proc. Summer Conf. Computer Graphics (SCCG'00)*, 2000

[6] S. Lefebvre and H. Hoppe, "Perfect spatial hashing," *ACM Trans. Graphics*, vol. 25, no. 3, pp. 579–588, 2006.

[7] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to algorithms*, Cambridge, MIT Press, 2001.

[8] J. I. Munro and P. Celis, "Techniques for collision resolution in hash tables with open addressing," *Proc. 1986 ACM Fall Joint Computer Conf. (ACM '86)*, pp. 601–610, 1986.

[9] A. Sud, N. Govindaraju, R. Gayle, I. Kabul, and D. Manocha, "Fast proximity computation among deformable models using discrete Voronoi diagrams," *ACM Trans. Graphics*, vol. 25, no. 3, pp. 1144–1153, 2006.

[10] A. D. Gregory, M. C. Lin, S. Gottschalk, and R. Taylor, "A framework for fast and accurate collision detection for haptic interaction," in *IEEE Virtual Reality Conf.*, pp. 38–45, 1999.

[11] M. Eitz and G. Lixu, "Hierarchical spatial hashing for real-time collision detection", *Proc. the IEEE Int'l Conf. on Shape Modeling and Applications (SMI'07)*. pp. 61–70, 2007.


## ABOUT THE AUTHORS


1. **Sunhwa Jung** is a PhD student in computer science and information systems at the University of Colorado Denver. His research interests are in physically-based simulation and collision detection and resolution. Contact him at sunhwa.jung@cudenver.edu.

2. **Min-Hyung Choi** is the director of the Computer Graphics and Virtual Environments Laboratory, and an associate professor of Computer Science and Engineering at the University of Colorado Denver. Contact him at min.choi@cudenver.edu.