Efficient Collision Detection using Hybrid Medial Axis Transform and BVH for Rigid Body Simulation

Xingxin Li [1,2] Junfeng Yao [1,2] * Rongzhou Zhou [1,2] Qingqi Hong [1,2]

[1]Xiamen University, China
[2]Key Laboratory of Digital Protection and Intelligent Processing of Intangible Cultural Heritage of Fujian and Taiwan Ministry of Culture and Tourism

Abstract

Medial Axis Transform(MAT) has been recently adopted as the acceleration structure of broad-phase collision detection. Compared to traditional BVH-based methods, MAT can provide a high-fidelity volumetric approximation of 3D complex objects, resulting in higher collision culling efficiency. However, due to MAT's non-hierarchical structure, it may be outperformed in collision-light scenarios because several cullings at the top level of a BVH may take a large number of cullings with MAT. We propose a collision detection method that combines MAT and BVH to address the above problem. Our technique efficiently culls collisions between dynamic and static objects. Experimental results show that our method has higher culling efficiency than pure BVH or MAT methods.

Keywords: Collision detection, Physical simulation.

1. Introduction

Most physically-based animations have to efficiently handle collision problems to prevent visual artifacts like penetration or overlap between 3D meshes. For both real-time and offline simulators, collision detection(CD) is their major performance bottleneck. Especially in scenes where objects generally have tens of thousands or even hundreds of thousands of triangle primitives, it is infeasible to perform pair-wise primitive overlapping tests in a brute-force way. A large research effort has been invested in speeding up collision detection by culling out non-colliding primitives. These collision culling algorithms can be classified into broad-phase culling techniques like Bounding Volume Hierarchy(BVH) [2, 7], spatial subdivision [5], medial elastics [11], and narrow-phase culling techniques like normal cone [21, 23], orphan set [18], and representative triangles [4, 19].

*Corresponding author: yao0010@xmu.edu.cn

Another factor that affects performance comes from the type of collision detection algorithm chosen. To ensure high efficiency, interactive applications often employ discrete collision detection (DCD). However, it only detects collisions at discrete moments, which often leads to problems of penetration or tunneling, which hinders its applicability. In contrast, continuous collision detection [16, 3, 22] is a more accurate alternative. CCD considers trajectories of objects or primitives in a time interval by interpolation and checks for the first collision event, including collision time and location. The problem with CCD is that it is more time-consuming and can fail due to float-point rounding errors.

Inspired by Lan *et al.*'s [11] and Song *et al.*'s [17] works, we present an efficient collision culling and detection method using medial axis transform and BVH in rigid body simulation. For dynamic objects in the simulation, we interpolate the bounding sphere along its moving trajectory to obtain a medial cone and convert collision culling between dynamic objects into cone-cone overlap tests. For static objects, we use oriented bounding box(OBB) as leaf primitive to build a BVH structure to ensure higher culling efficiency. The overlapping test between a medial cone and OBB is formulated as an intersection test between a 4D conic section and the parametric domain, which can be solved following a similar routine of cone-cone collision culling. For narrow phase collision detection, our method simply performs pair-wise medial primitive level CD instead of triangle level CD. Experiments show that our method can produce realistic results in collision-rich scenes at interactive speed and achieve higher performance than traditional BVH methods.

2. Related Work

In this section, we briefly introduce the previous works on continuous collision detection algorithms, including collision culling, parallel collision detection, and collision handling.

Collision culling is a critical process for existing collision detection algorithms because of limited computing resources. BVH is a primarily adopted high-level culling algorithm for CCD and DCD. Various types of bounding volume have been explored such as Axis-Aligned Bounding Box (AABB) [2], Object-aligned Bounding Box (OBB) [6], Bounding Sphere [7, 8], Boxtree [27], Spherical Shell [10], etc. For deformable body simulation, these BVHs have to be updated to satisfy shape changes in each frame. Many algorithms are presented to quickly refit or reconstruct BVHs, including linear time refitting [13], selective restructuring [15, 26], and parallel reconstruction [14, 9]. Most Low-level culling algorithms are designed for self-collision in CCD by removing duplicate elementary tests based on connectivity information of mesh. Orphan Sets [18] and Representative Triangles [4, 19] are the low-level culling algorithms for many CCD applications with self-collisions, especially for cloth simulation. Orphan Sets [18] precomputes an orphan set from adjacent collision pairs and removes the primitives among all adjacent pairs that are not in the orphan sets. Representative Triangles [4, 19] assigns each primitive to a unique triangle to guarantee no duplicate elementary test would be checked. Besides, many techniques handle deformable objects by computing some bounds related to deformation and using them for self-collision culling. Barbič and James [1] computed self-collision culling *certificates* in subspace to accelerate self-collision culling. Based on the observation that a self-collision occurs under large local deformation, Zheng and James [28] proposed an energy-based metric to improve the effectiveness of self-collision culling. Wong *et al*. [24] proposed a technique that accelerates continuous collision detection by performing radial view-based culling based on the skeleton structure, but the skeleton needs to be precomputed and the overhead for models undergoing topology changes can be high.

Elementary tests of CCD are performed by finding the roots of a cubic polynomial equation, which is derived from coplanar conditions. These elementary tests are typically implemented using finite-precision or floating-point arithmetic and use error tolerances, resulting in the tests being prone to error, such as false negatives. Many exact cubic solvers are proposed to avoid these errors, and there is an excellent review of the topic [25] recommended to readers. Brochu *et al*. [3] proposes an exact CCD algorithm by calculating non-constructive predicates for parity of the number of collisions. Wang [22] performed forward error analysis to check the existence of exact vertex-triangle or edge-edge intersection to reduce false positives. Tang *et al*. [20] presented a geometrically exact CCD algorithm based on the exact geometric computation paradigm to perform reliable Boolean collision queries.

Recently, Medial Axis Transform(MAT) has been introduced as the acceleration structure of collision detection. Medial elastics [11] is the first work to use MAT for collision detection, mainly proposing a semi-reduced projective dynamics framework that seamlessly combines collision detection and model reduction using MAT. On this basis, Song *et al*. [17] proposed a continuous collision detection algorithm at the medial primitive level for rigid body simulation. It divides the higher-order problem of computing the first time of contact between the medial primitives into finding the nearest sphere pair between primitives and the CCD problem of two moving spheres and solves it quickly in an alternate iterative manner. Lan and colleagues give the medial primitive level CCD problem analytic formulation and a numerical solution in Medial-IPC[12].

3. MAT Preprocessing

Existing MAT approximation approaches generate medial mesh with quite some degenerated cases which will have a performance impact on subsequent collision detection method. In this section, we will cover the detailed algorithm to detect and filter out these degenerated medial primitives before using them.

3.1. Degenerated Medial Cone

The degenerated case of the medial cone is quite simple. A medial cone $\mathcal{C} = \{m_1, m_2\}$ degenerates into a sphere if and only if one medial sphere encapsulates the another, as shown in Fig.1. This degeneration can be trivially identified by checking whether the distance of 2 spheres' centers plus the smaller radius is less than the larger radius.

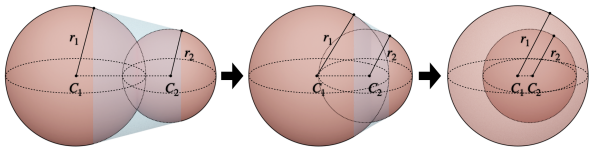


Figure 1: Degeneration of medial cone

3.2. Degenerated Medial Slab

Similar to medial cone, the main reason for the degeneration of medial slab can be summarized as the severe overlap of the volumes of medial spheres or cones. It can be categorized into the following two cases according to the cause of degeneration.

- **Case 1** Medial spheres overlapping. In this case, medial sphere with maximum radius encapsulates one or both other two spheres. Based on the number of encapsulated spheres, slab may become medial cone (1 sphere) or medial sphere (2 spheres).
- **Case 2** Medial cones overlapping. As Fig.2 shows, the conical surface of a medial cone divides the space into

two parts. If the smallest sphere falls into the internal region, the slab will degenerate into cone.

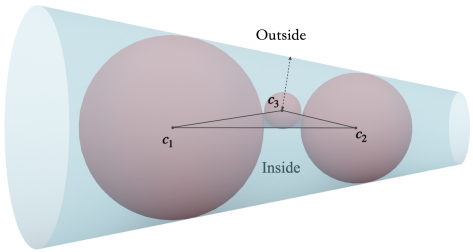


Figure 2: Degeneration of medial slab **Case 2**

Detecting **Case 1** is simply performing previous degenerated medial cone tests for each edge of a slab. On the contrary, **Case 2** is much more complicated, we will elaborate on how to detect this type of degeneration.

Consider a medial slab $\mathcal{S}$ and let c_1, c_2, c_3 be the centers of medial spheres at three vertices of $\mathcal{S}$, whose radii are r_1, r_2 and r_3 respectively. Without loss of generality, assume that $r_1 \geq r_2 \geq r_3$. We use Γ_{ij} to donate the conical surface formed by medial sphere m_i and m_j. The most intuitive way to detect whether sphere m_3 falls in the internal region is computing the surface distance between m_3 and Γ_{12}. However, suffered from numerical errors, this method will produce false positive or false negative errors. To avoid these errors, the distance threshold needs to be adjusted specifically for some slabs, which means it is hard to tune.

Let C^i_{ij} be the intersection circle of Γ_{ij} and m_i. we notice that if m_3 falls in the internal region, circle C^1_{12} and C^1_{13} will have at most one intersection point on m_1, as shown in Fig.3. With this key observation, the detection of **Case 2** can be simplified to check the number of intersections of C^1_{12} and C^1_{13}.Computing the intersection point of the intersection circles can be converted into the calculation of the intersection point of two conical surfaces.

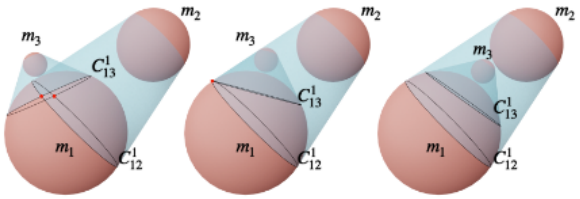


Figure 3: Illustration of different cases of C^1_{12} and C^1_{13}. The intersection points are highlighted in red.

We take Γ^1_{12} and Γ^1_{13} as an example to show how to compute their intersection point p. Fig.4 gives an illustration of the computation process. Here, $\vec{n}$ represents the normal vector of plane $c_1c_2c_3$. c_p is the projection point of p on the plane. v_{12} and v_{13} are the vertical feet of c_p on edge c_1c_2 and c_1c_3 respectively. Since we can simply rotate $\vec{n}$ by 90 degrees around c_1c_2 with v_{12} as pivot to obtain $v_{12}c_p$, c_p can be calculated by performing an intersection test of plane $v_{13}c_pp$ and line $v_{12}c_p$. p can be calculated from c_p and $\vec{n}$ based on the geometric relationship. In the implementation, once we compute the intersection point p, checking the existence of p is equivalent to checking whether p is on sphere m_1.If there is a point p not on m_1, it means that m_3 falls in the internal region. After preprocessing, degenerated medial primitives will be simplified into spheres or medial cones based on their degeneration types.

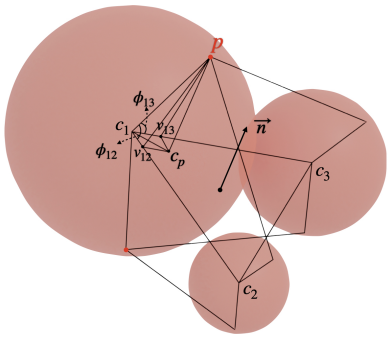


Figure 4: The computation process of intersection point of conical surfaces Γ^1_{12} and Γ^1_{13}.

4. Collision Culling

Since the simplified MAT can provide a high-fidelity volumetric approximation of the original 3D shape with orders of magnitude fewer primitives than triangle mesh, we use it instead of triangle mesh in collision detection. Due to MAT's non-hierarchical structure, pair-wise medial primitive intersection tests can be executed entirely in parallel on the GPU. But it would be a waste of performance to perform all tests indiscriminately for those time steps where no or very few collisions occur. Especially as the number of objects increases, the simulation performance will get worse.

To avoid this problem, common implementations use a two-phase approach: broad-phase and narrow-phase. In broad-phase, collision tests are usually based on bounding volume only to quickly prune away pairs of objects that do not collide with each other and output the potential colliding pairs. In our simulation framework, we use bounding sphere and bounding box for dynamic objects and static objects respectively in broad-phase. Assume the trajectory is linearized in each time step, bounding spheres at the initial position and end position form a medial cone that has constant radius variation, as shown in Fig.5. In this way, the broad-phase collision detection between dynamic objects is converted into a cone-cone intersection test. As both cones

have constant radius variation, the problem can be further simplified. In particular, when the time step is large, the trajectory can be approximated by a chain of medial cones in the form of piece-wise interpolation, which is equivalent to approximating curves with discrete linear segments.

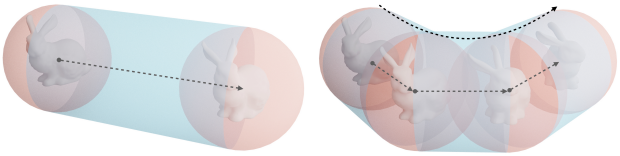


Figure 5: Bounding medial cone(s) formed along trajectory

4.1. Cone-OBB collision culling

For static objects, we use BVH of OBBs in broad-phase collision culling. Consider an oriented bounding box $\mathcal{B}$ defined by a corner point o and orthogonal set of basis vectors $\{\hat{u}, \hat{v}, \hat{w}\}$ and a medial cone $\mathcal{C}$. Let c_1, c_2 be the centers of medial spheres at two vertices of $\mathcal{C}$, whose radii are r_1 and r_2 respectively. On $\mathcal{B}$, arbitrary point v can be defined through trilinear interpolation as:

$$v = o + \hat{u}t_1 + \hat{v}t_2 + \hat{w}t_3, \quad t_1, t_2, t_3 \in [0, 1]. \tag{1}$$

The center c and radius r of an arbitrary sphere on $\mathcal{C}$ can be defined though linear interpolation as:

$$c = t_4 c_1 + (1 - t_4)c_2,\ r = t_4 r_1 + (1 - t_4)r_2,\ 0 \le t_4 \le 1. \tag{2}$$

The minimum surface distance between $\mathcal{B}$ and $\mathcal{C}$ can be formulated as:

$$\begin{aligned} \min \quad & f(t_1, t_2, t_3, t_4) = \|v - c\| - r = \sqrt{S} - (r_1 - r_2) \\ & t_4 - r_2, \\ \text{s.t.} \quad & t_1, t_2, t_3, t_4 \in [0, 1]. \end{aligned} \tag{3}$$

where

$$\begin{aligned} & S = At_1^2 + Bt_2^2 + Ct_3^2 + Dt_4^2 + Et_1t_2 + Ft_1t_3 + Gt_1t_4 \\ & + Ht_2t_3 + It_2t_4 + Jt_3t_4 + Kt_1 + Lt_2 + Mt_3 + Nt_4 \\ & + O, \\ & A = \|\hat{u}\|^2, B = \|\hat{v}\|^2, C = \|\hat{w}\|^2, D = \|c_2 - c_1\|^2, \\ & E = 2\hat{u} \cdot \hat{v}, F = 2\hat{u} \cdot \hat{w}, G = 2\hat{u} \cdot (c_2 - c_1), \\ & H = 2\hat{v} \cdot \hat{w}, I = 2\hat{v} \cdot (c_2 - c_1), J = 2\hat{w} \cdot (c_2 - c_1), \\ & K = 2\hat{u} \cdot (o - c_2), L = 2\hat{v} \cdot (o - c_2), M = 2\hat{w} \cdot (o - c_2), \\ & N = 2(c_2 - c_1) \cdot (o - c_2), O = \|o - c_2\|^2. \end{aligned} \tag{4}$$

Once $f(t_1, t_2, t_3, t_4)$ is less or equal to zero, we know that $\mathcal{B}$ and $\mathcal{C}$ do overlap. So, instead of the actual minimum value, our only interest is if there exists at least one real solution for $f(t_1, t_2, t_3, t_4) = 0$. Since both $\sqrt{S}$ and r are positive, $f(t_1, t_2, t_3, t_4) = 0$ is equivalent to:

$$\begin{aligned} & g(t_1, t_2, t_3, t_4) = S - ((r_1 - r_2)t_4 + r_2)^2 \\ & = At_1^2 + Bt_2^2 + Ct_3 + (D - (r_1 - r_2)^2)t_4^2 + Et_1t_2 \\ & + Ft_1t_3 + Gt_1t_4 + Ht_2t_3 + It_2t_4 + Jt_3t_4 + Kt_1 \\ & + Lt_2 + Mt_3 + (N - 2r_2(r_1 - r_2))t_4 + O - r_2^2 = 0. \end{aligned} \tag{5}$$

$g(t_1, t_2, t_3, t_4)$ is essentially a quadratic function of t_1, t_2, t_3 and t_4. Setting $g(t_1, t_2, t_3, t_4)$ equal to zero describes a 4D conic section that divides the 4D space into two regions. $g = 0$ will have at least one real solution if and only if the conic section intersects with the parametric domain which is a tesseract. By setting one or more parameters to 0 or 1, we can reduce the dimensionality of the problem which eventually yields a quadratic equation. We can verify if the quadratic equation has at least one solution between 0 and 1 to secure the collision. If $g = 0$ does not overlap with any boundary of the tesseract region, we still need to check the situation when $g = 0$ is a 4D hyperellipsoid that encloses the parametric domain. Specifically, if the value of g at the center of the hyperellipsoid is negative, we can secure the collision. In practice, if the dynamic object remains static or only has rotational motion at the current time step, $\mathcal{C}$ will degenerate into a sphere. For this case, we can simply perform a sphere-OBB intersection test.

After broad-phase collision culling, we will perform medial primitive level continuous collision detection on GPU for pairs of objects that may collide.

4.2. Implementation Details

We approximate the 3D shape of MAT by a medial mesh M_s. Each point M_i in M_s is denoted as a medial sphere and is represented by a 4-dimensional point $m = (c, r)$, where $c \in R^3$ is the center of the medial sphere and r is its radius. The edges of Ms are represented by two medial spheres M_i and M_j as $e_{ij} = \{M_i, M_j\}$. Similarly, the triangular faces of Ms are denoted as $f_{ijk} = \{M_i, M_j, M_k\}$. Each edge or face of the medial mesh defines a simple composite volume primitive. The envelope volume element obtained by linear interpolation of the medial spheres M_i and M_j, that is, $(1-t)M_i + tM_j, t \in [0, 1]$, is called the medial cone, which consists of two medial spheres and the cone surface they form. The primitive given by the surface $\{M_i, M_j, M_k\}$ is called the medial slab, which is obtained by linearly interpolating the three intermediate spheres M_i, M_j, and M_k, that is,$a_1M_i + a_2M_j + a_3M_k$, where $a_i (i = 1, 2, 3) >= 0$, $a_1 + a_2 + a_3 = 1$, and it consists of three medial spheres, three conic surfaces, and two triangles.

We store broad-phase bounding primitives and simplified MAT in a generalized medial mesh structure that contains a list of medial spheres and a list of primitive indices. The first index of each primitive is used to identify its type(-2 for sphere, -1 for medial cone, and valid index for medial

slab). We add the bounding sphere at the end of the sphere list and the corresponding indices at the start of the primitive list for dynamic objects. For static objects, we reorganize the basis vectors and corner point of its OBB into three 4D vectors and store them at the end of sphere list, and the indices of these vectors form a primitive. The detailed structure is shown in Fig.6.

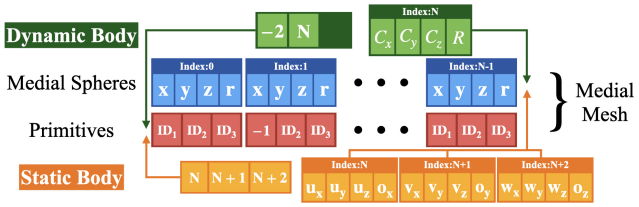


Figure 6: Structure of the generalized medial mesh.

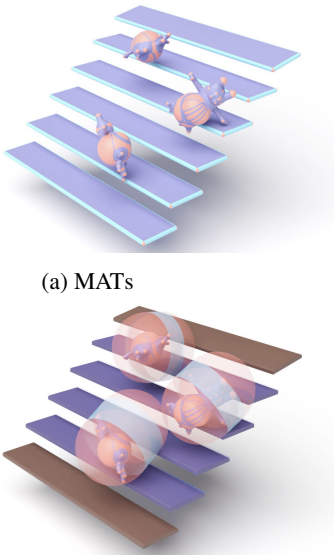
5. Experimental Results

We implement our algorithm in python, using Taichi(CUDA backend) for GPU parallel computation. All experiments are run on a Windows desktop computer with a 3.0GHz Intel Core i7-5960X CPU and an NVIDIA RTX 2060S(8GB memory). We tested our framework on a variety of complex models in collision-rich scenes. In the preparation stage, the medial meshes of all models used in the experiments are calculated and simplified using Q-MAT. Then, we construct the broad-phase bounding primitives and generate the generalized medial mesh for each object. Tab.1 reports the detailed MAT and geometry information for all models used in experiments.

Model	# Tri.	# MC	# MS	ϵ
Deer	16,932	100	42	2.02%
Cow	19,324	163	80	1.03%
Armadillo	86,382	210	116	1.21%
Bear	20,278	119	60	0.61%
Bug	17,276	61	17	0.47%
Dolphin	10,506	121	62	0.55%
Spider	22,098	56	7	0.71%
Bunny	69,630	201	125	1.10%
Dinosaur	30,000	318	170	0.78%
Torus	1,152	26	0	0.03%

Table 1: MATs and geometry information of models. # **Tri** is the total number of triangle faces in the original mesh. # **MC** and # **MS** represent the total number of medial cones and slabs in the simplified medial mesh. ϵ records the one-side Hausdorff distance error from the original surface to MAT.

Culling performance Fig.7 shows a typical example, where several deers and cows collide with some static stairs. From Tab.1, if there is no broad-phase collision culling, we need to perform about 330,000 times medial primitive level CCD tests per frame. But in the vast majority of frames,



(a) MATs

(b) Broad-phase bounding primitives

Figure 7: Structure of the generalized medial mesh.

these deer and cow models do not collide with each other, so there is no need to perform all tests. In the particular frame that Fig.7 shows, after collision culling, there are only about 1,500 tests left, resulting in a significant performance boost. Detailed time statistics are given in Fig.8. It is very clear that after applying broad-phase collision culling, the efficiency of collision culling and detection has been significantly improved.

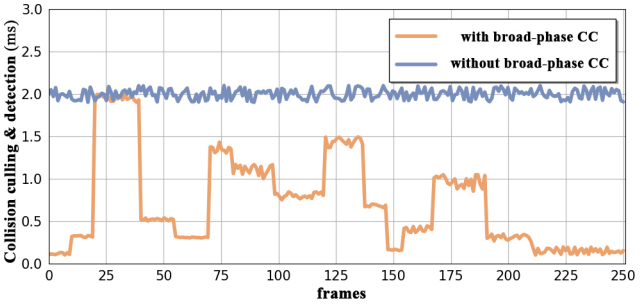


Figure 8: Detailed time statistics of applying broad-phase collision culling.

Comparison with BVHs To quantify the collision performance, we compare our algorithm with BVHs of different bounding primitives in several scenes. In Fig.9, a bunch of models falls over a few glassy rods. Each rod's medial mesh contains only a single medial cone. The total number of primitive pairs in this scene is about 2.1 million, which means that brute force execution of all tests is exceedingly

time-consuming even on GPU(∼25ms). To quantify the collision performance, we set the total number of leaves in BVH to be the same as the number of medial primitives. Fig.10(**top**) records the detailed time statistics of collision detection with BVHs of different bounding primitives and MAT. It is clear that MAT-based CC/CD is much more efficient than traditional BVHs. However, since we use the medial cone as the broad-phase bounding primitive for dynamic objects, the collision culling efficiency between dynamic objects is similar to that of BVH of Bounding Sphere but lower than that of BVH of OBB. The efficiency improvement of our method is mainly due to the fact that the MAT does not contain tree-like data structures, and the broad-phase collision culling only requires a modest number of cone-cone and cone-obb overlapping tests.

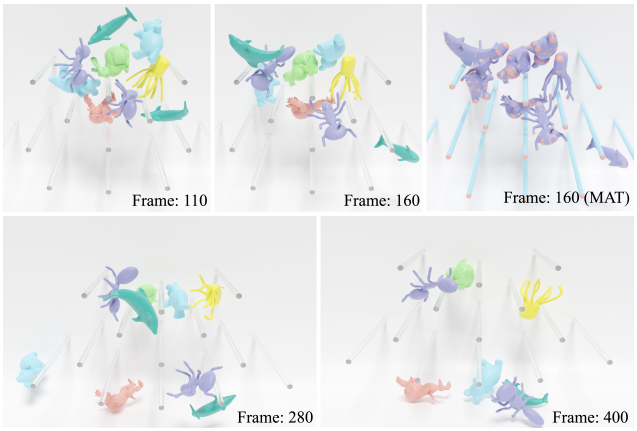


Figure 9: Nine different models fall over a few glassy rods. The average times for collision culling and collision detection are 0.15 ms and 6.78ms, respectively.

More examples More examples are shown in Fig.11 with per-frame time statistics in Fig.10. To quantify the collision performance, we also set the total number of leaves in BVH to be the same as the number of medial primitives. In Fig.11(**top**), a dinosaur model(in yellow) flies into a group of objects at speed of 100m/s. Medial-primitive level CCD can detect all those collisions without any penetration or tunneling artifacts. Fig.11(**bottom**) shows an array of 51 torus falls to the ground. Since the overall number of primitive pairs in this scene is smaller than in the first two scenes, the collision performance remains high even in those collision-rich frames (100-110 frames) where culling efficiency is very low.

6. Conclusion and Discussion

In this paper, we present a hybrid MAT and BVH method for collision culling and detection in rigid body simulation. For broad-phase collision culling, we use the medial cone as the bounding primitive to enclose the trajectory of dynamic

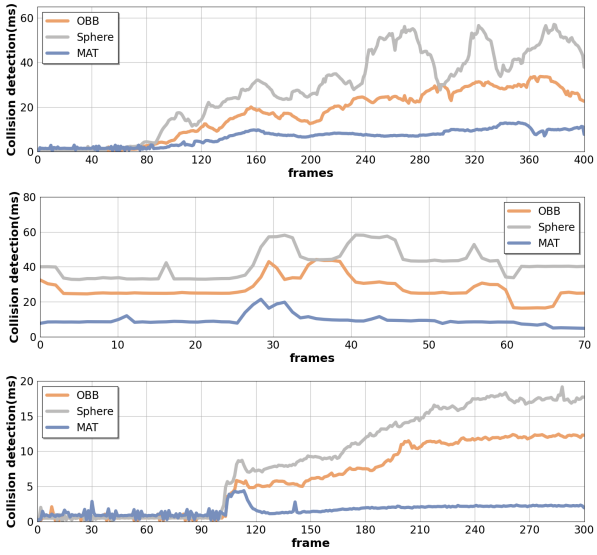


Figure 10: **From top to bottom**: Detailed timing statistics of MAT, OBB, and Bounding Sphere for scenes in Fig.9, Fig.11(**top**) and Fig.11(**bottom**). The average CC/CD time of MAT: 6.48ms,10.42ms,1.67ms;The average CC/CD time of OBB: 16.76ms,27.81ms, 6.15ms;The average CC/CD time of bounding sphere:27.76ms,41.75ms, 8.84ms.

objects and BVH of OBB for static objects to ensure high culling efficiency. As each object's bounding primitive is combined with its input medial mesh, the broad-phase CC and narrow-phase CD can be executed seamlessly on GPU. Also, since we only construct the BVH of the static object, there is no overhead of updating the BVH during the simulation.

However, our hybrid MAT and BVH collision detection method also have several limitations. For dynamic objects, we directly use the bounding sphere along the object trajectory to form a medial cone as the broad-phase bounding primitive, which can only provide poor bounding quality. Especially in those collision-rich animations, low bounding quality results in low culling efficiency, which in turn leads to an increase in the number of CCD tests that need to be performed in the narrow phase. In the future, we would like to try to generate hierarchical MAT to address this limitation. Using multiple layers of MAT, collision culling between dynamic objects can be done in the same way as BVH, improving culling efficiency while maintaining a modest number of overlapping tests.

7. Acknowledgment

The paper is supported by the Natural Science Foundation of China (No. 62072388), Col-laborative Project fund of Fuzhou-Xiamen-Quanzhou Innovation Zone(No.3502ZCQXT202001),the industry guidance project foundation of science tech-nology bureau of

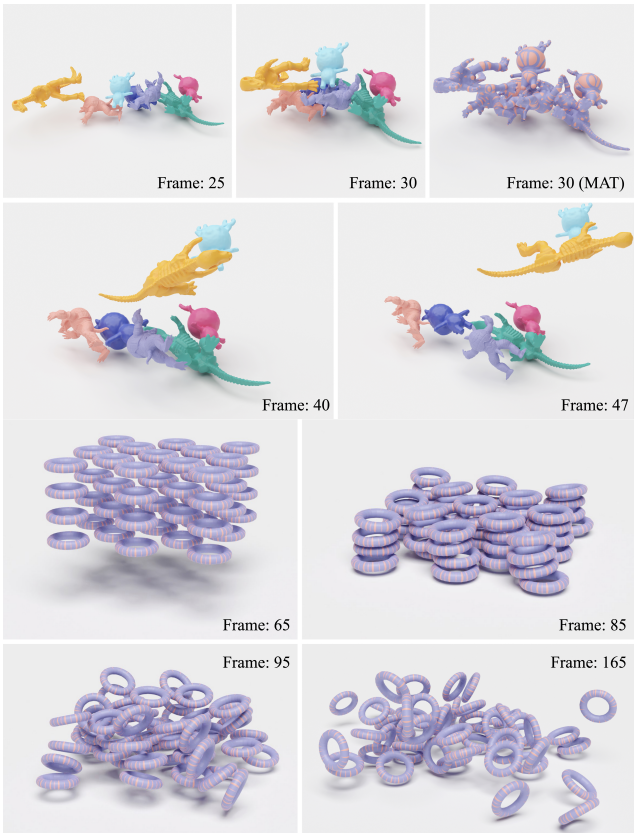


Figure 11: **Top**: A dinosaur flies into a group of objects at high speed. The total number of primitive pairs in this scene is around 3.8 million. **Bottom**: An array of 51 torus falls to the ground. Each torus only contains 26 medial cones, resulting 66,300 primitive pairs in total.

Fujian province in 2020(No.2020H0047), the natural science foundation of science technology bureau of Fujian province in 2019 (No.2019J01601), the creation fund project of science technology bureau of Fujian province in 2019(No.2019C0021), and Fujian Sunshine Charity Foundation.

References

[1] J. Barbiĉ and D. L. James. Subspace self-collision culling. *ACM Trans. Graph. (TOG)*, 29(4):81, 2010. 2

[2] G. v. d. Bergen. Efficient collision detection of complex deformable models using aabb trees. *Journal of graphics tools*, 2(4):1–13, 1997. 1, 2

[3] T. Brochu, E. Edwards, and R. Bridson. Efficient geometrically exact continuous collision detection. *ACM Transactions on Graphics (TOG)*, 31(4):1–7, 2012. 1, 2

[4] S. Curtis, R. Tamstorf, and D. Manocha. Fast collision detection for deformable models using representative-triangles. In *Proceedings of the 2008 Symposium on Interactive 3D Graphics and Games*, I3D '08, page 61–69, New York, NY, USA, 2008. Association for Computing Machinery. 1, 2

[5] M. de Berg, J. Comba, and L. J. Guibas. A segment-tree based kinetic bsp. In *Proceedings of the Seventeenth Annual Symposium on Computational Geometry*, SCG '01, page 134–140, New York, NY, USA, 2001. Association for Computing Machinery. 1

[6] S. Gottschalk, M. C. Lin, and D. Manocha. Obbtree: A hierarchical structure for rapid interference detection. In *Computer graphics and interactive techniques*, pages 171–180. ACM, 1996. 2

[7] P. M. Hubbard. Collision detection for interactive graphics applications. *IEEE Transactions on Visualization and Computer Graphics*, 1(3):218–230, 1995. 1, 2

[8] D. L. James and D. K. Pai. Bd-tree: output-sensitive collision detection for reduced deformable models. *ACM Trans. Graph. (TOG)*, 23(3):393–398, 2004. 2

[9] D. Kopta, T. Ize, J. Spjut, E. Brunvand, A. Davis, and A. Kensler. Fast, effective bvh updates for animated scenes. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '12, page 197–204, New York, NY, USA, 2012. Association for Computing Machinery. 2

[10] S. Krishnan, M. Gopi, M. Lin, D. Manocha, and A. Pattekar. Rapid and accurate contact determination between spline models using shelltrees. In *Computer Graphics Forum*, volume 17, pages 315–326. Wiley Online Library, 1998. 2

[11] L. Lan, R. Luo, M. Fratarcangeli, W. Xu, H. Wang, X. Guo, J. Yao, and Y. Yang. Medial elastics: Efficient and collision-ready deformation via medial axis transform. *ACM Transactions on Graphics (TOG)*, 39(3):1–17, 2020. 1, 2

[12] L. Lan, Y. Yang, D. Kaufman, J. Yao, M. Li, and C. Jiang. Medial ipc: Accelerated incremental potential contact with medial elastics. *ACM Trans. Graph.*, 40(4), jul 2021. 2

[13] T. Larsson and T. Akenine-Möller. A dynamic bounding volume hierarchy for generalized collision detection. *Computers & Graphics*, 30(3):450–459, 2006. 2

[14] C. Lauterbach, M. Garland, S. Sengupta, D. P. Luebke, and D. Manocha. Fast bvh construction on gpus. *Computer Graphics Forum*, 28(2):375–384, 2009. 2

[15] M. A. Otaduy, O. Chassot, D. Steinemann, and M. Gross. Balanced hierarchies for collision detection between fracturing objects. In *2007 IEEE Virtual Reality Conference*, pages 83–90, 2007. 2

[16] X. Provot. Collision and self-collision handling in cloth model dedicated to design garments. In D. Thalmann and M. van de Panne, editors, *Computer Animation and Simulation '97*, pages 177–189, Vienna, 1997. Springer Vienna. 1

[17] S. Song, L. Lan, J. Yao, and X. Guo. Continuous collision detection with medial axis transform for rigid body simulation. *Commun. Inf. Syst.*, 22(1):53–78, 2022. 1, 2

[18] M. Tang, S. Curtis, S.-E. Yoon, and D. Manocha. Interactive continuous collision detection between deformable models using connectivity-based culling. In *SPM '08: Proceedings of the 2008 ACM symposium on Solid and physical modeling*, pages 25–36, New York, NY, USA, 2008. ACM. 1, 2

[19] M. Tang, D. Manocha, S.-E. Yoon, P. Du, J.-P. Heo, and R. Tong. VolCCD: Fast continuous collision culling between

deforming volume meshes. *ACM Trans. Graph.*, 30:111:1–111:15, May 2011. 1, 2

[20] M. Tang, R. Tong, Z. Wang, and D. Manocha. Fast and exact continuous collision detection with bernstein sign classification. *ACM Transactions on Graphics (TOG)*, 33(6):1–8, 2014. 2

[21] P. VOLINO and N. M. THALMANN. Efficient self-collision detection on smoothly discretized surface animations using geometrical shape regularity. *Computer Graphics Forum*, 1994. 1

[22] H. Wang. Defending continuous collision detection against errors. *ACM Transactions on Graphics (TOG)*, 33(4):1–10, 2014. 1, 2

[23] T. Wang, Z. Liu, M. Tang, R. Tong, and D. Manocha. Efficient and reliable self-collision culling using unprojected normal cones. *Computer Graphics Forum*, 36(8), 2017. 1

[24] S.-K. Wong, W.-C. Lin, C.-H. Hung, Y.-J. Huang, and S.-Y. Lii. Radial view based culling for continuous self-collision detection of skeletal models. *ACM Transactions on Graphics (TOG)*, 32(4):1–10, 2013. 2

[25] C. K. Yap and V. Sharma. *Robust Geometric Computation*, pages 1860–1863. Springer New York, New York, NY, 2016. 2

[26] S.-E. Yoon, S. Curtis, and D. Manocha. Ray tracing dynamic scenes using selective restructuring. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, EGSR'07, page 73–84, Goslar, DEU, 2007. Eurographics Association. 2

[27] G. Zachmann. Minimal hierarchical collision detection. In *ACM symposium on Virtual reality software and technology*, pages 121–128. ACM, 2002. 2

[28] C. Zheng and D. L. James. Energy-based self-collision culling for arbitrary mesh deformations. *ACM Trans. Graph. (TOG)*, 31(4):98, 2012. 2