Communications in Information and Systems
Volume 14, Number 4, 1–27, 2014

Continuous Collision Detection with Medial Axis Transform for Rigid Body Simulation

Shibo Song, Lei Lan , Junfeng Yao* and Xiaohu Guo*

Continuous Collision Detection (CCD) is a fundamental problem for physically based simulation, such as rigid motion, elastic deformation, cloth animation, etc. CCD has been widely studied in the past decades, and most proposed algorithms are performed on the level of triangle mesh by using some culling methods to reduce the number of tested triangle pairs. However, the efficiency of these algorithms is very sensitive to the resolution of the triangular mesh. In this paper, we present an effective and efficient CCD method based on Medial Axis Transform (MAT) for simulating rigid motion. The simplified MAT, represented as a medial mesh composed of medial primitives like cones and slabs can provide a high-quality tight enclosure of the original 3D shape, while its number of primitives is smaller than that of the original triangle mesh by several orders of magnitude. With this key observation, our colliding elementary tests are only performed on the medial mesh, and the first time-of-impact in CCD can be computed by solving a quadratic optimization problem. The experiments show our algorithm can accurately and efficiently handle CCD of multi-bodies with thin features, without any penetration or tunnelling artifacts. Compare to the standard Bounding Volume Hierarchy (BVH) methods, our method achieves great improvement in efficiency.

1. Introduction

Most physically-based simulations have to effectively and efficiently handle collision problems to prevent invalid penetration or overlap between geometric primitives. The fidelity and efficiency of these simulators depends largely on the collision detection algorithms they choose. For real-time or interactive simulators, collision detection is their major performance bottleneck. In general, an exact collision detection not only needs to determine whether two objects are intersecting on a given configuration, but also needs to query all overlapping geometric primitives. Nowadays, simulated objects with hundreds of thousands or even millions of surface triangles are ubiquitous, and it is unacceptable to perform pair-wise primitive intersection test in a brute-force way. In the past decades, a number of collision detection algorithms have been proposed to speed up computation by culling out the non-colliding primitives as much as possible. These algorithms can be classified into high-level culling techniques like Bounding Volume Hierarchy (BVH) [14, 34], spatial subdivision [9], and low-level culling techniques like normal cone [35, 37], orphan sets [30], and representative triangles [8, 32].

However, efficiency of collision detection is still a serious challenge for most real-time applications. To compromise on efficiency, these applications have to use Discrete Collision Detection (DCD), which only checks for collisions at a given time step and omit the collisions between the time interval. It is well known that DCD may suffer from missing collisions when objects move at a high speed or simulators run on a large time step, and could lead to visual artifacts in simulation. In contrast, Continues Collision Detection (CCD) [7, 28, 36] is a more accurate alternative. CCD considers trajectories of objects in a time interval by interpolation, and checks for the first collision event, including the first time-of-contact and location information, that happens along the trajectories. At the triangle-level detection, CCD reduces the overlapping test of a triangle pair to 15 elementary tests, including 6 vertex-face tests and 9 edge-edge tests [6, 28]. These elementary tests require using cubic polynomial root solvers to obtain the first time-of-contact. Therefore, CCD is more time-consuming than DCD. Besides physically-based simulations, CCD has been widely used in robot motion planning [31, 32], haptic rendering [10, 40], etc.

In this paper, we propose a new CCD algorithm for rigid body simulation. Our algorithm accelerates CCD by directly performing elementary tests on Medial Axis Transform (MAT) of input objects, rather than on their triangle mesh. MAT of a 3D object is composed of a set of medial spheres, which are the maximally inscribed spheres touching the boundary with at least two contacts [5]. MAT preserves the topology and geometric features of the 3D objects while containing their local thickness information, so MAT has long been considered as an effective shape approximation representation [11, 29, 41]. Medial mesh is a compact expression of MAT, and it is a non-manifold triangle mesh that discretizes MAT with piecewise linear elements. Vertices on the medial mesh are medial spheres, and the connections between vertices form the volumetric medial primitives by linearly interpolating two (or three) medial spheres along edge (or faces). Lan et

al. [23] demonstrated that medial primitives can be used in DCD as bounding primitives to perform pair-wise primitive intersection tests by solving an analytically Quadratically Constrained Quadratic Program (QCQP) problem. Since the simplified medial mesh only contains a modest number (n) of medial primitives while maintaining a high-quality approximation for the 3D object with a large number (N) of triangles, the simulator only needs $\mathbf{O}(n)$ ($n \ll N$) calculations and culls more non-colliding triangle pairs in DCD to achieve interactive simulation of elastic bodies with high-resolution triangle meshes in collision-rich scenes. The algorithm we propose extends DCD algorithm of MAT in [23] to CCD. In addition, due to the non-deformable nature of rigid bodies, we directly perform CCD on MAT rather than using it as bounding primitives for triangular meshes.
Main Contributions: Inspired by Lan et al.'s work [23], we present a CCD algorithm with MAT for rigid body simulation in this paper. Only medial primitives are detected for elementary tests, and the global first time-of-contact and colliding location are found by solving a quadratic optimization problem about the spherical distance between medial primitives in the given time interval. We use alternating iteration to solve this optimization problem: when the time instance is fixed as a constant, this problem is equal to the QCQP in [23]; when the nearest sphere pair is fixed, this problem becomes a standard quadratic optimization problem and can be solved by computing its derivative. This iteration strategy can converge quickly in the rigid body simulation. The results show our algorithm can significantly improve the efficiency of CCD without any penetration or tunnelling artifacts.

2. Related Work

In this section, we briefly introduce the prior works about CCD algorithms, including collision culling, cubic solver, 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 primary adopted high-level culling algorithm for CCD and DCD. Various types of bounding volume have been explored such as Axis-Aligned Bounding Box (AABB) [34], Object-aligned Bounding Box (OBB) [12], Bounding Sphere [14, 16], Box-tree [44], Spherical Shell [19], etc. For deformable body simulation, these BVHs have to be update to satisfy shape changes in each frame. Many algorithms are presented to quickly refit or reconstruct BVHs, including linear time refitting [20], selective restructuring [26, 43], and parallel reconstruction [18, 21]. Most of Low-level culling algorithms are designed for self-collision in CCD by removing duplicate elementary tests based on connectivity information of mesh. Orphan Sets [30] and Representative Triangles [8, 32] are the low-level culling algorithms for many CCD applications with self-collisions, especially for cloth simulation. Orphan Sets [30] 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 [8, 32] 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 [4] 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 [45] proposed an energy-based metric to improve the effectiveness of self-collision culling. Wong et al. [39] 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 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 avoiding these errors, and there is an excellent review of the topic [42] recommended to readers. Brochu et al. [7] proposes an exact CCD algorithm by calculating non-constructive predicates for parity of the number of collisions. Wang [36] performed forward error analysis to check the existence of exact vertex-triangle or edge-edge intersection to reduce false positives. Tang et al. [33] presented a geometrically exact CCD algorithm based on the exact geometric computation paradigm to perform reliable Boolean collision queries.

In order to increase the performance of CD, a number of acceleration technologies have been proposed by exploiting the parallel capability of GPUs [22, 23, 27, 38]. Since we only need a very low number of medial primitives to tightly bound each 3D object, our algorithm simply checks all elementary tests of medial primitive pairs parallelly on GPU. Recently, some simulators of elastoplastic contact models are presented to handle challenging contact scenarios with CCD [13, 17, 24]. Although these simulators can produce high-quality animations, they run in a slow rate far from real-time.

3. Medial Mesh Representation

An initial MAT usually is computed by the Voronoi diagram of a set of sampled points on the shape boundary of 3D object [1]. However, the initial MAT has many undesirable spikes, making them unsuitable for any practical application. To obtain a structurally simple and compact medial axis, the initial MAT has to be simplified by identifying and pruning the spikes. Li et al. [25] proposed a MAT simplification algorithm that using a quadratic error metric to measure approximation errors in MAT simplification. This algorithm efficiently and effectively removes a lot of meaningless medial primitives, while producing a compact and accurate approximation of input object. Once the radii of MAT is slightly dilated, a high-quality tight enclosure of the original object can be obtained. Therefore, the original 3D objects can be enclosed by the dilated MAT to perform the overlapping test, by detecting if there is any medial spheres overlapping.

Given a medial vertex $\mathbf{m}$ on medial mesh, and it is denoted as a 4D vector by $\mathbf{m} = \{\mathbf{c}^\top, r\}^\top$, where $\mathbf{c}$ is the center of medial sphere and r is its radius. Each edge and face of medial mesh is associated with a volumetric primitive, which is called *medial primitive*, as shown in Fig. 1(a). All medial primitives are combined to be the enveloping primitives of the medial mesh, which is an approximation of the original 3D object, as shown in Fig. 1(b). An edge is denoted $\mathcal{C}_{ij} = \{\mathbf{m}_i, \mathbf{m}_j\}$, which connects two vertices $\mathbf{m}_i$ and $\mathbf{m}_j$. By interpolating both the sphere centers and radii information across $\mathcal{C}_{ij}$, the resulting volumetric primitive is a *medial cone*: $\{\mathbf{m}|\mathbf{m} = \alpha\mathbf{m}_i + (1-\alpha)\mathbf{m}_j,\ \alpha \in [0,1]\}$. Similarly, a triangle of medial mesh is denoted $\mathcal{C}_{ijk} = \{\mathbf{m}_i, \mathbf{m}_j, \mathbf{m}_k\}$. The volumetric primitive associated with it is a *medial slab*, which is obtained by linearly interpolating the three medial spheres: $\{\mathbf{m}|\mathbf{m} = \beta_i\mathbf{m}_i + \beta_j\mathbf{m}_j + (1-\beta_i-\beta_j)\mathbf{m}_k, \beta_i \in [0,1], \beta_j \in [0, 1-\beta_i]\}$.

For a 3D object, its medial mesh is not only an inner skeleton, but also provides a volumetric approximation of the object with a small number of medial primitives. Lei et al. [23] used the medial primitive as the bounding volume to cull unnecessary collisions (DCD only) in elastic body animation. The distance between two medial primitives is formulated as a signed spherical distance that is determined by the two closest medial spheres on each other. If the distance is negative, the two medial primitives are overlapping. Otherwise, they are separating. Since the medial mesh provides an excellent

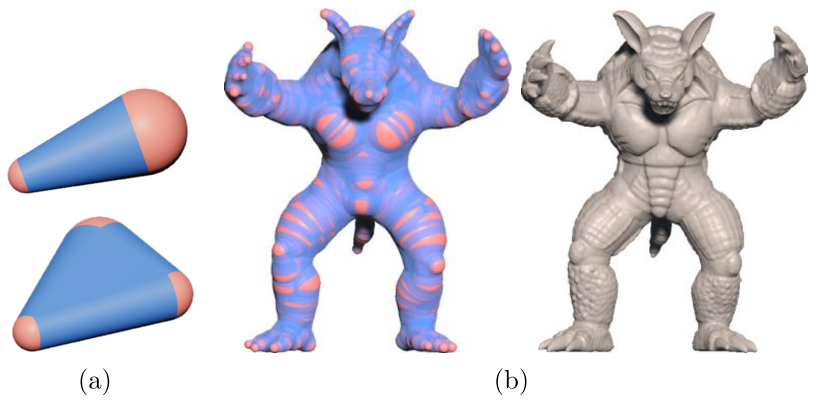

Figure 1: (a) Medial primitives: medial cone (upper) and medial slab (bottom). (b) The enveloping primitives of its medial mesh (left) and the original 3D object (right).

shape approximation with only a small number of medial primitives, the method shows at least two orders of magnitude higher culling effectiveness than other widely used bounding primitives, such as OBB and bounding spheres. Its culling strategy does not depend on any tree-like multi-level data structure. All overlapping tests on the pairs of medial primitives are parallelly performed on GPUs. However, Lei et al. [23] ran the overlapping tests of medial primitives on the pipeline of discrete collision detection, some collision events may be missed when adopting a large time step.

In this paper, we will generalize the MAT-based bounding volume to continues collision detection for rigid body simulation. The radii of medial spheres do not need to be updated because of the rigid motion. It means that the bounding quality of rest-shape MAT is maintained, and high collision accuracy can be obtained if a high-resolution medial mesh is configured. Unlike [23], our algorithm will directly perform the low-level collision detection on MAT, instead of on triangle mesh.

4. Continue Collision Detection on Medial Mesh

4.1. Continuous Collision Detection

For CCD on triangle mesh, detecting collisions between two triangles in motion reduces to performing pairwise vertex-face and edge-edge elementary tests. The motion of triangles usually is linearly interpolated within a time interval. For two arbitrary triangles, there are 6 vertex-face and 9 edge-edge elementary tests to be performed to compute the first time-of-contact. For two arbitrary medial primitives, detecting continuous collisions can also be reduced to performing two types of pairwise elementary tests: cone-cone and sphere-slab. For two medial cones we need to perform a cone-cone test; for a medial cone and a medial slab we need to perform 3 cone-cone and 2 sphere-slab tests; and for two medial slabs we need to perform 9 cone-cone and 6 sphere-slab tests. A list of collision tests including all cone-cone and sphere-slab tests between two medial meshes will be built up in the precomputation stage. During animating, all tests on the list will be parallelly performed on GPUs if the bounding boxes of two medial meshes are overlapping.

4.2. Elementary Tests of Medial Primitive Pairs

Let us introduce our algorithm by taking the cone-cone test as an example. Its routine can be readily generalized to sphere-slab tests, since the problem structure is unchanged. Consider two medial cones $\mathcal{C}_1 = \{\mathbf{m}|\mathbf{m} = \alpha\mathbf{m}_{11} + (1-\alpha)\mathbf{m}_{12}, \alpha \in [0,1]\}$ and $\mathcal{C}_2 = \{\mathbf{m}|\mathbf{m} = \beta\mathbf{m}_{21} + (1-\beta)\mathbf{m}_{22}, \beta \in [0,1]\}$ in rigid motion. Let $\mathbf{v}_{11}$, $\mathbf{v}_{12}$, $\mathbf{v}_{21}$, $\mathbf{v}_{22}$ be the constant velocities of the four centers of medial spheres in the time interval $[t_0, t_1]$. The centers and radii of arbitrary spheres on these two cones at time $t \in [t_0, t_1]$ can be defined as:

$$\begin{aligned} \mathbf{c}_1(\alpha,t) &= \alpha(\mathbf{c}_{11} + \mathbf{v}_{11}t) + (1-\alpha)(\mathbf{c}_{12} + \mathbf{v}_{12}t), \\ r_1(\alpha) &= \alpha r_{11} + (1-\alpha)r_{12}, \\ \mathbf{c}_2(\beta,t) &= \beta(\mathbf{c}_{21} + \mathbf{v}_{21}t) + (1-\beta)(\mathbf{c}_{22} + \mathbf{v}_{22}t), \\ r_2(\alpha) &= \beta r_{21} + (1-\beta)r_{22}. \end{aligned} \tag{1}$$

The signed distance between these two spheres can be formulated as:

$$\begin{aligned} S(\alpha,\beta,t) &= \|\mathbf{c}_1(\alpha,t) - \mathbf{c}_2(\beta,t)\| - (r_1(\alpha) + r_2(\beta)) \\ &= \sqrt{P_1(\alpha,\beta)t^2 + P_2(\alpha,\beta)t + P_3(\alpha,\beta)} - R(\alpha,\beta). \end{aligned} \tag{2}$$

Here,

$$\begin{aligned}
P_1(\alpha,\beta) \quad &= \|\mathbf{v}_{11}-\mathbf{v}_{12}\|^2\alpha^2 + \|\mathbf{v}_{21}-\mathbf{v}_{22}\|^2\beta^2\\
&-2(\mathbf{v}_{11}-\mathbf{v}_{12})^\top(\mathbf{v}_{21}-\mathbf{v}_{22})\alpha\beta\\
&+2(\mathbf{v}_{11}-\mathbf{v}_{12})^\top(\mathbf{v}_{12}-\mathbf{v}_{22})\alpha\\
&-2(\mathbf{v}_{21}-\mathbf{v}_{22})^\top(\mathbf{v}_{12}-\mathbf{v}_{22})\beta\\
&+\|\mathbf{v}_{12}-\mathbf{v}_{22}\|^2,
\end{aligned}$$

$$\begin{aligned}
P_2(\alpha,\beta) \quad &= 2(\mathbf{v}_{11}-\mathbf{v}_{12})^\top(\mathbf{c}_{11}-\mathbf{c}_{12})\alpha^2\\
&+2(\mathbf{v}_{21}-\mathbf{v}_{22})^\top(\mathbf{c}_{21}-\mathbf{c}_{22})\beta^2\\
&-2[(\mathbf{v}_{11}-\mathbf{v}_{12})^\top(\mathbf{c}_{21}-\mathbf{c}_{22})+(\mathbf{v}_{21}-\mathbf{v}_{22})^\top(\mathbf{c}_{11}-\mathbf{c}_{12})]\alpha\beta\\
&+2[(\mathbf{v}_{11}-\mathbf{v}_{12})^\top(\mathbf{c}_{12}-\mathbf{c}_{22})+(\mathbf{v}_{12}-\mathbf{v}_{22})^\top(\mathbf{c}_{11}-\mathbf{c}_{12})]\alpha\\
&-2[(\mathbf{v}_{21}-\mathbf{v}_{22})^\top(\mathbf{c}_{12}-\mathbf{c}_{22})+(\mathbf{v}_{12}-\mathbf{v}_{22})^\top(\mathbf{c}_{21}-\mathbf{c}_{22})]\beta\\
&+2(\mathbf{v}_{12}-\mathbf{v}_{22})^\top(\mathbf{c}_{12}-\mathbf{c}_{22}),
\end{aligned}$$

$$\begin{aligned}
P_3(\alpha,\beta) \quad &= \|\mathbf{c}_{11}-\mathbf{c}_{12}\|^2\alpha^2 + \|\mathbf{c}_{21}-\mathbf{c}_{22}\|^2\beta^2\\
&-2(\mathbf{c}_{11}-\mathbf{c}_{12})^\top(\mathbf{c}_{21}-\mathbf{c}_{22})\alpha\beta\\
&+2(\mathbf{c}_{11}-\mathbf{c}_{12})^\top(\mathbf{c}_{12}-\mathbf{c}_{22})\alpha\\
&-2(\mathbf{c}_{21}-\mathbf{c}_{22})^\top(\mathbf{c}_{12}-\mathbf{c}_{22})\beta\\
&+\|\mathbf{c}_{12}-\mathbf{c}_{22}\|^2,
\end{aligned}$$

$$R(\alpha,\beta) \quad = (r_{11}-r_{12})\alpha + (r_{21}-r_{22})\beta + (r_{12}+r_{22}).$$

The goal of CCD is to determine whether collisions occur within the time interval, and obtain the first time-of-contact, which is equivalent to solving the minimum root of $S(\alpha,\beta,t)=0$. However, it is difficult to find the analytic solution of $S(\alpha,\beta,t)=0$. We re-formulate it as the following minimization problem:

$$\begin{aligned}
&\min \quad f(\alpha,\beta,t) = S(\alpha,\beta,t)^2,\\
&\text{s.t.} \quad 0\le\alpha\le 1, \quad 0\le\beta\le 1, \quad t_0\le t\le t_1.
\end{aligned} \tag{3}$$

Obviously, $f(\alpha,\beta,t)=0$ is the minimum value. If collisions happen between $\mathcal{C}_1$ and $\mathcal{C}_2$ during the time interval $[t_0,t_1]$, there must be at least one root making $f(\alpha,\beta,t)=0$. We just need to obtain the root that is first time-of-contact. Otherwise, $f(\alpha,\beta,t)>0$ at the entire span of time interval.

We propose to minimize $f(\alpha,\beta,t)$ and solve for $\{\alpha,\beta\}$ and $\{t\}$ in an iterative manner: (1) fix the time t, and solve for the linear interpolation parameters $\{\alpha,\beta\}$ of the nearest sphere pair; (2) fix $\{\alpha,\beta\}$, and solve for the closest time instance t. We iterate these 2 steps until convergence or no-collision being confirmed. In addition, after the QCQP[23] is solved in the first step, the quadratic optimization problem in second step is actually to search the closest moment of the fixed sphere pair in the continuous domain $[t_0, t_1]$. Therefore, the whole iteration process of searching for the First Time-of-Contact is continuous and consistent with the idea of CCD.

From geometric perspective, the whole iteration process is to find the nearest sphere pair $\{\alpha^j, \beta^j\}$ at a time t^j firstly (j is the iteration number), then find the first time t^{j+1} when the spheres α^j and β^j are in contact (or closest if no-collision). We keep updating the nearest sphere pair $\{\alpha, \beta\}$ and contact time t until they converge, which means we find the first time-of-contact and the colliding sphere pair.

4.3. The First Time-of-Contact

When t is fixed, the problem is equivalent to the discrete collision detection for MAT. The nearest sphere pair can be obtained by the method proposed by Lan et al. [23]. When $\{\alpha, \beta\}$ is fixed, we set the first-order derivatives of $f(t)$ to be zero:

$$\frac{\mathbf{d}f(t)}{\mathbf{d}t} = \frac{(\sqrt{P_1t^2 + P_2t + P_3} - R)(2P_1t + P_2)}{\sqrt{P_1t^2 + P_2t + P_3}} = 0. \tag{4}$$

Due to the existence of $\frac{1}{\sqrt{P_1t^2+P_2t+P_3}}$, Eq. (4) has a singular point when $P_1t^2 + P_2t + P_3 = 0$. It will happen if and only if the centers of two spheres $\{\alpha, \beta\}$ coincide. So, we can ignore it in general situations. It is noticed that $\sqrt{P_1t^2 + P_2t + P_3} - R = 0$ is equivalent to a parabola $P_1t^2 + P_2t + P_3 - R^2 = 0$, and it has two roots $t^{(1)}$, $t^{(2)}$ ($t^{(1)} < t^{(2)}$). The third solution comes from $2P_1t + P_2 = 0$: $t^{(3)} = -\frac{P_2}{2P_1}$, which is exactly the symmetry axis of the parabola. Note that we use the notation $t^{(i)}$ to denote the i-th root of t for Eq. (4), and use the notation t^j to denote the j-th iteration of t throughout the optimization of Eq. (3). We can discuss the following two cases for the parabola:

Case 1: $t^{(1)}, t^{(2)} \in \mathbb{R}$. This is the most general case. If $t^{(1)} \geq t_0$ and $f(t^{(1)}) = 0$, $t^{(1)}$ is the first time-of-contact of the two spheres. Then, the two spheres will penetrated each other in the deepest distance at $t^{(3)}$, and contact again for separation on $t^{(2)}$, as shown in Fig. 2(a). However, if $t^{(2)} < t_0$, the two spheres do not have any contact during $[t_0, t_1]$.

Case 2: $t^{(1)}, t^{(2)} \notin \mathbb{R}$. If $t^{(3)} \in [t_0, t_1]$, $t^{(3)}$ is the moment that the two spheres are closest, as shown in Fig. 2(b). If $f(t^{(3)}) = 0$, $t^{(3)}$ is the first time-of-contact of the two spheres. Otherwise, the two spheres are in separation during $[t_0, t_1]$.

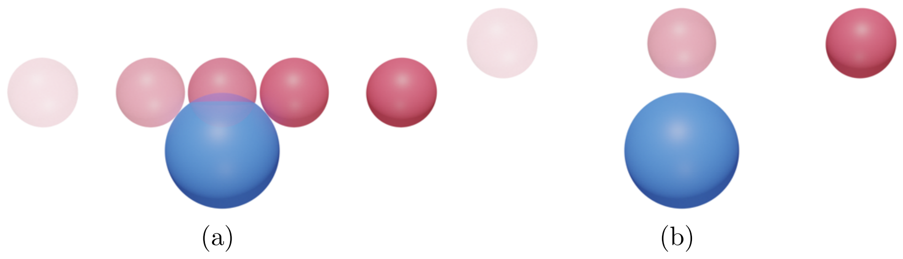

Figure 2: (a) **Case 1**: two spheres contact each other at first time at $t^{(1)}$, if $t^{(1)} \geq t_0$ and $f(t^{(1)}) = 0$. From left to right, red spheres represent its locations at t_0, $t^{(1)}$, $t^{(3)}$, $t^{(2)}$, and t_1 respectively. (b) **Case 2**: $t^{(3)}$ is the moment that the two spheres are closest. From left to right, red spheres represent its locations at t_0, $t^{(3)}$, and t_1 respectively.

Besides the two cases, there is one special situation when the parabola is degenerated at $P_1 = P_2 = 0$. It indicates that relative velocity between two medial cones equals zero, and they contact on their rest positions. This situation will prevent the termination of the iteration. We will discuss how to handle it in section 4.4.

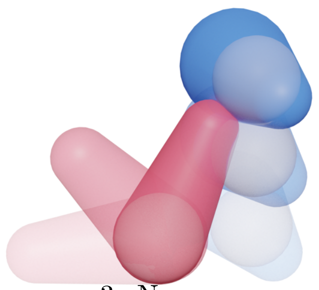

Figure 3: Non-convergence case when using t_0 as initial value. Primitives are more transparent when t is closer to t_0. White spheres are the nearest sphere pair $\{\alpha^0, \beta^0\}$ at t_0.

In our experiments, we have found that the convergence of iteration is influenced by the selection of initial values. In the situation shown in Fig. 3, the distance of the nearest sphere pair $\{\alpha^0, \beta^0\}$ at t_0 will increase monotonically over $[t_0, t_1]$. This situation makes the optimal t rollback to t_0, and the iteration will become an endless loop between $\{\alpha^0, \beta^0\}$ and t_0. To solve this problem, t_1 is choose as initial value since the nearest sphere pair $\{\alpha^0, \beta^0\}$ at t_1 will be updated once t is changed.

4.4. Termination Conditions

Obviously, the distance between the nearest sphere pair equals zero should be the primary condition of termination. However, this condition is not enough to determine whether a collision will occur. Given the result of j-th iteration

t^j. if the nearest sphere pair at $t = t^j$ is $\{\alpha^{j+1}, \beta^{j+1}\}$ and $f(\alpha^{j+1}, \beta^{j+1}, t^j) = 0$ has been met in the next iteration. To prevent false-positive error, the three roots $t^{(1)}, t^{(2)}, t^{(3)}$ have to be updated from $f(\alpha^{j+1}, \beta^{j+1}, t) = 0$, then check whether the medial cones will collide or move away in the next time-step. The termination conditions are summarized as following:
Condition 1: if two primitives are in contact at t_0 and separate away on $[t_0, t_1]$. An illustrating example is shown in Fig. 4(a). In this case, either root $t^{(2)}$ or $t^{(3)}$ are equal to t_0. We return no-collision to terminate the iteration.
Condition 2: if two primitives intersect on $[t_0, t_1]$ and are in contact at t_1. An illustrating example is shown in Fig. 4(b). This case is caused by using t_1 as initial value and belongs to the tunnelling effect. In this case, the root $t^{(2)}$ will be equal to t_1, and we should update t by root $t^{(1)}$, then go on iterations.
Condition 3: if the distance keeps greater than zero, which means the two medial cones never collide. t^j and t^{j+1} are used to terminate the iteration. If t^j is equal to t^{j+1}, we return non-collision directly.

The schematic figures of **Condition 1** and **Condition 2** are shown in Fig. 4. Since **Condition 3** represents all general no-collision situations, we will not elaborate on it.

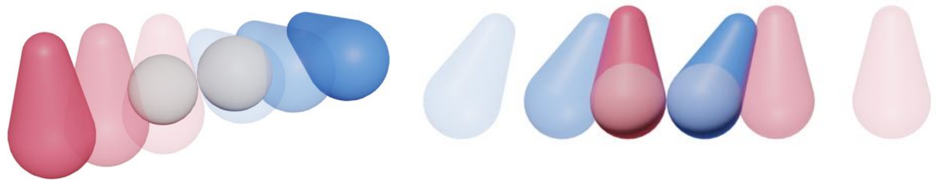

Figure 4: The trajectory of two medial primitives (using two medial cone as an example). The closer t is to t_0, the higher the transparency of cones. The white spheres represent the contact sphere pair. (a) **Condition 1**: two cones are in contact at t_0 and separate away. (b) **Condition 2**: two cones are in contact at t_1 after passing through each other.

4.5. Collision Response

After collision detection, we can confirm the two rigid bodies are colliding if we can get a valid solution of Eq. (3) making $f(\alpha, \beta, t) = 0$.Once collisions happen, we need to update all rigid objects by the global minimum first time-of-contact t_{min}. Then, all nearest sphere pairs will be discarded if their time-of-contact are greater than t_{min}. We use the non-penetration constraint method [2] to perform collision response. For colliding impulse, we can easily compute contact point and contact normal through the nearest sphere pair, as shown in Fig. 9.

5. Experimental Results

In this section, we describe our implementation and highlight the performance of our algorithm on several collision-rich scenes. We have implemented our algorithm with Unity 2019.4.0f1 and run the experiments on a Windows desktop computer with Intel i7 5960X CPU (3.0 GHz) and an NVidia RTX 2060 SUPER GPU. The source codes will be made publicly available upon the paper's publication. Note our CCD algorithm runs parallelly on the GPU, with high accuracy and real-time rate. Medial axis transform is extracted and simplified using Q-MAT [25]. Table 1 summarizes 3D models' MAT information. Table 2 records the time spent on every stage of the algorithm and FPS of animation. Fig. 8 records the number of active elementary pair and the detailed timing statistics of continuous collision. We build three benchmarks to show the advantages of our algorithm in Fig. 5, Fig. 6 and Fig. 7. We refer the readers to the accompanying video for the detailed animation effects.

	#Tris.	#MS.	#MPs.	# EPs.
Multi-bodies	276,643	5,214	2,263	8,581,733
T-rex	200,004	5878	2,472	74,755,481
Falling Rings	19,115	876	433	101,192

Table 1: Statistics of 3D models' MAT information. # **Tris.** is the total number of triangles. # **MS.** and # **MPs.** are total numbers of medial spheres and medial primitives on medial mesh. # **EPs.** are total numbers of elementary pairs.

5.1. Accuracy of Continuous Collision Detection

In order to verify the accuracy of our algorithm, we test multiple bodies dropping on some extremely thin pillars, as shown in Fig. 5(a). Each pillar is enclosed in a medial cone. During the animation, multi-bodies with difference shapes and volumes will collide with pillars or each other, and no

	Init	CD	CR	Upd	#FPS.
Multi-bodies	13.781	6.30	0.016	1.489	56
T-rex	10.343	9.57	0.038	0.829	35
Falling Rings	0.438	3.28	0.035	0.137	75

Table 2: Statistics of running time in simulation. **Init** is the time for generating elementary pairs and rendered MATs (s). **CD** is the average collision detection time (ms). **CR** is the average collision response time(ms). **Upd** is the average time to update rigid bodies(ms). # **FPS** is the average frame-per-second for the overall animation.

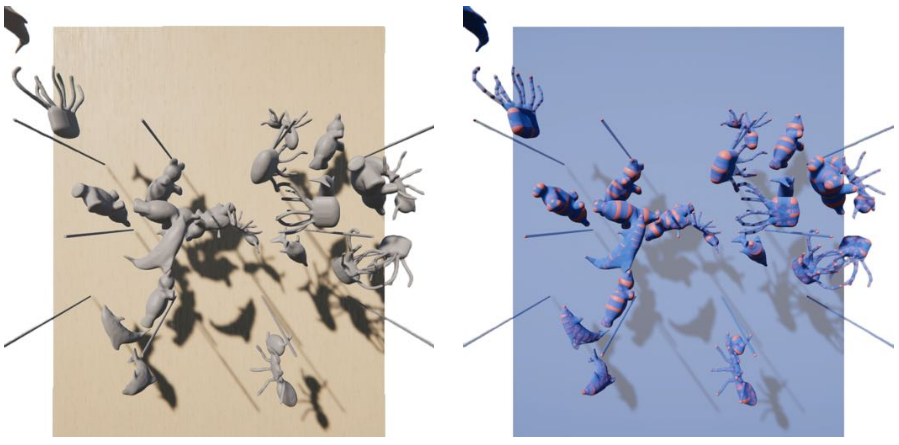
(a) (b)

Figure 5: Multi-bodies benchmark: (a) Multiple rigid objects collide with each other and the thin pillars. This scene has over 550K vertices and 270K triangles. (b) The interpolated MAT. Each MAT retains the shape features of its original triangle mesh.

collision is missed. Some collisions on thin geometric primitives are highlighted in Fig. 9.

To guarantee high bounding quality, we choose the proper number of medial primitives (MPs) for different models based on Hausdorff error, while retaining as much model details as possible. The specific statistics are shown in Table 3. Besides, we also compare the Hausdorff error of OBBs and Bounding Spheres with MAT. When BVHs have the same number of leaf primitives

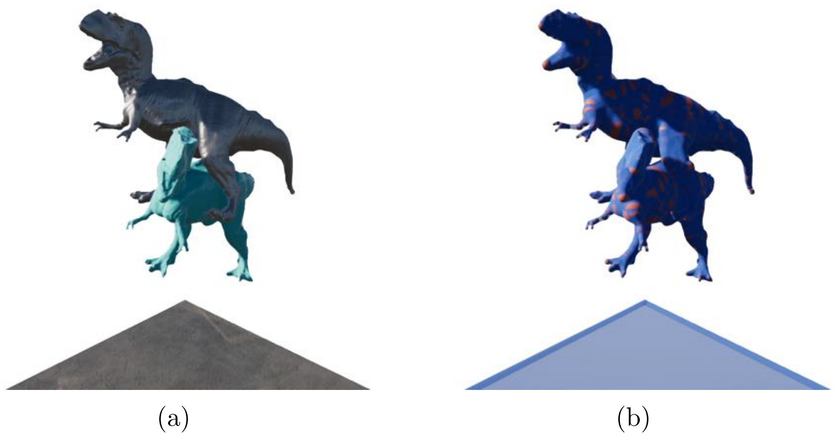

Figure 6: T-rex benchmark: (a) Two T-rexs collide with each other. This scene has over 200K vertices and 400K triangles. (b) The interpolated MAT. Each T-rex embeds 1,236 medial primitives, which retains more model details.

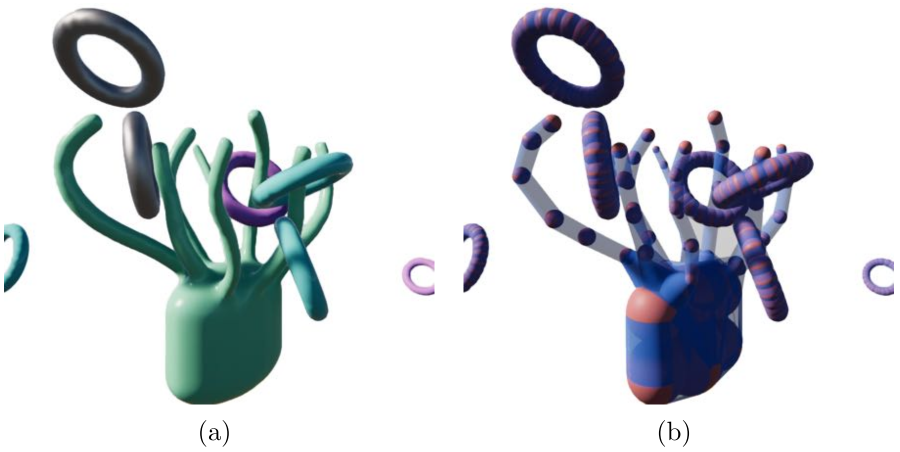

Figure 7: Falling rings benchmark: (a) The original triangle mesh. (b) The interpolated MAT. Only 27 medial primitives are needed to accurately approximate each ring without losing concavity.

as our MPs, MAT visually provides a much more accurate approximation

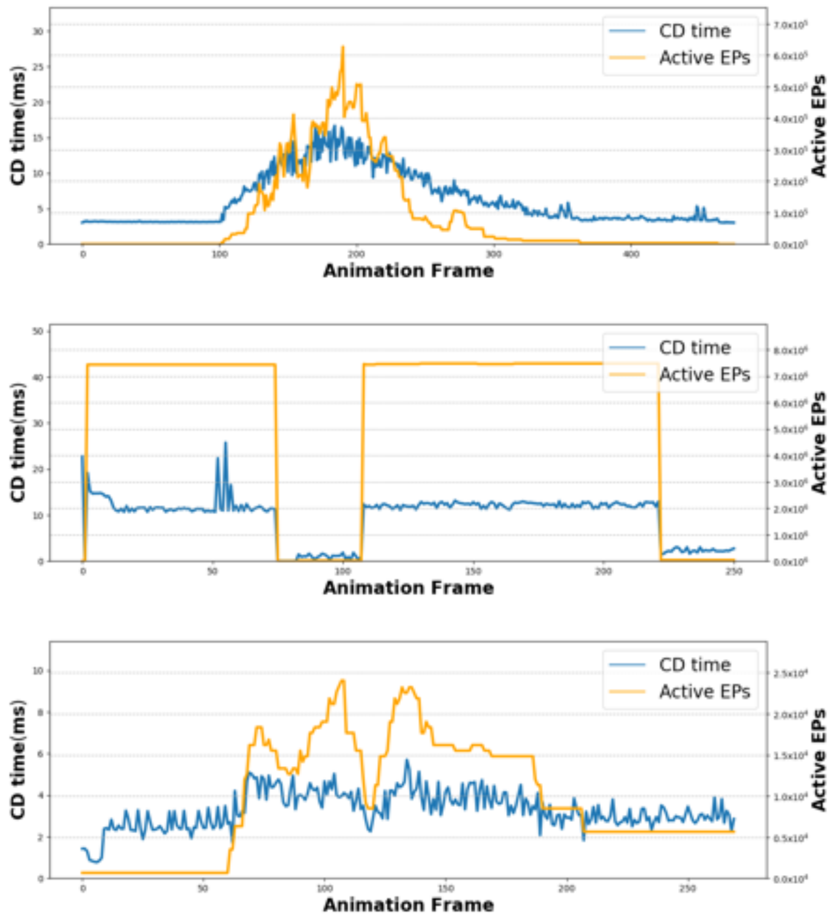

Figure 8: The statistics of detailed timing of CCD and number of active EPs. Muti-bodies benchmark(upper), T-rex benchmark(middle) and Falling rings benchmark(bottom).

to the original triangular mesh than OBBs and Bounding Spheres (shown in Fig. 10), which is the motivation that we perform CCD on MAT directly.

In order to prove the completeness of our algorithm and the accuracy of First Time-of-Contact(TOC), we compare our algorithm with triangle-level CCD in a colliding spider and bunny scene with only CD performed. In this scene, we generate a series of motions according to different linear and angular velocities to cover collisions in various situations. Then, we simulate each motion with CCD on high-resolution triangle meshes(Ground truth)

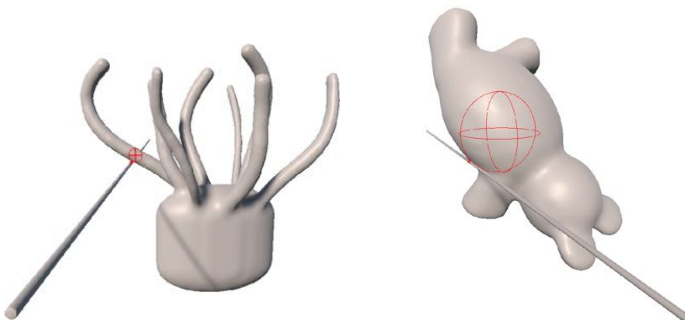

Figure 9: Colliding sphere pairs between models and thin pillars from Multi-bodies benchmark. Our algorithm can handle these collisions without penetration or tunnelling.

#MPs	50	500	1k	2k	5k
Bug	0.05	0.044	0.011	0.01	0.009
Bear	0.055	0.03	0.0136	0.135	0.0083
Dolphin	0.06	0.0233	0.023	0.0132	0.01
Spider	0.065	0.0114	0.008	0.007	0.007
Armadillo	0.138	0.059	0.051	0.0383	0.0279
T-rex	0.121	0.045	0.029	0.020	0.017

Table 3: The Hausdorff error of each model with different number of MPs. All models have been normalized into unit bounding boxes.

and CCD on MAT. As shown in Fig. 11, the TOC errors of each motion are extremely low, which indicates that our algorithm can handle collisions accurately and without omission.

5.2. Comparison with Bounding Volume Hierarchy

To highlight the efficiency of our algorithm, we compare our algorithm with BVHs of OBBs and Bounding Spheres in a Colliding T-rex and Armadillo scene, as shown in Fig. 12. We are able to approximate the T-rex model and Armadillo model with 691 MPs and 617 MPs respectively. In such setting, we only produce about 3 million elementary pairs. More importantly, our algorithm avoids performing CCD on triangles, while OBBs and Bounding Spheres still needs to compute CCD at the triangle-level because of their

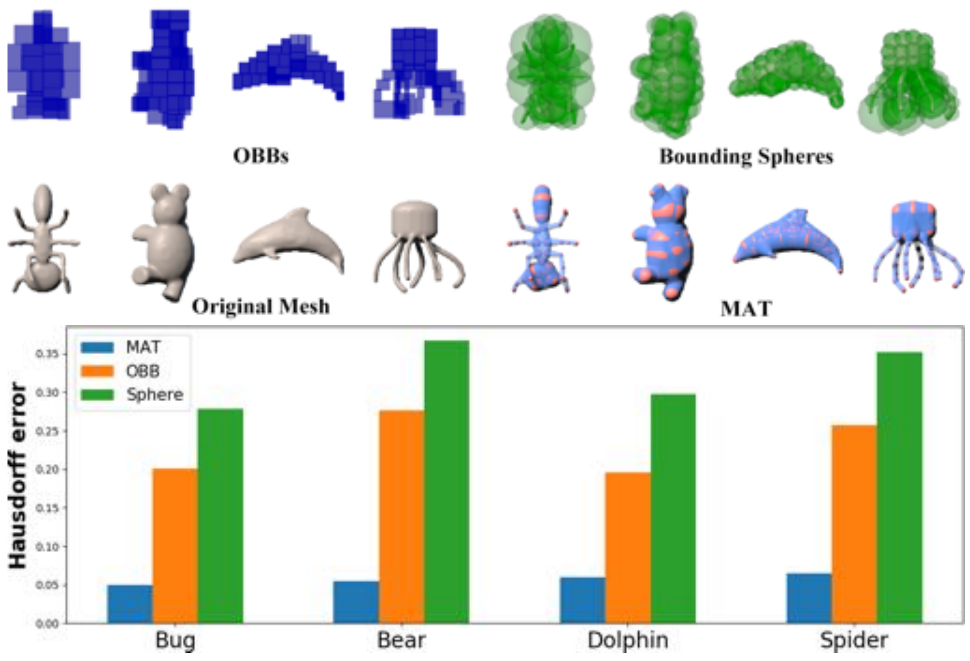

Figure 10: The bounding quality of MAT, OBBs, and Bounding Spheres with their numbers of leaf primitives equal the number of MPs (Bug: 80MPs; Bear: 100MPs; Dolphin: 114MPs; Spider: 69MPs). OBBs and Bounding Spheres have poor visual bounding qualities and high Hausdorff error, while our MAT approximates the original mesh accurately.

poor bounding capacity. As the result, our algorithm can be much faster than common BVHs on CCD. We also compare our algorithm with OBBs and Bounding Sphere on different resolution triangle meshes which are remeshed using instant field-aligned meshing [15], and report the timing statistics in Fig. 13. When the number of vertices of the triangle mesh is reduced, the performance of OBB/BS can be equal to our algorithm. But MAT can provide a better approximation of the original mesh while low resolution mesh will cause more visual artifacts as shown in Fig. 14.

5.3. Comparison with Bounding Convex Hull

We also compare our algorithm with the bounding convex hull algorithm, which is integrated in most popular game engines. The convex hull is computed from the triangle mesh based on Quickhull algorithm [3]. Although

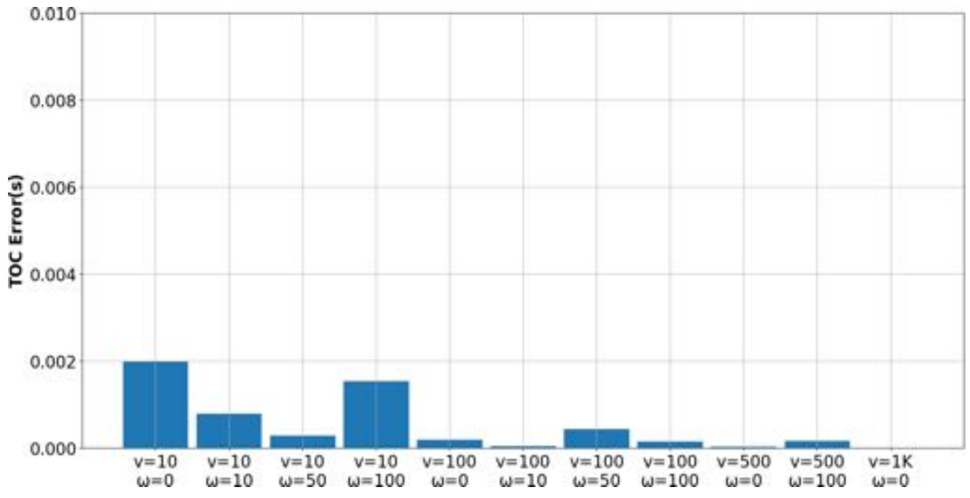

Figure 11: Comparison of First Time-of-Contact with triangle-level CCD. The time step of each simulation is fixed to 0.01s. With $\mathbf{t_{tri}}$ and $\mathbf{t_{mat}}$ representing the TOC of triangle-level CCD and TOC of MAT-level CCD respectively, **TOC Error** is defined as $|\mathbf{t_{tri}} - \mathbf{t_{mat}}|$. $\mathbf{v}$ and ω represent the magnitude of the relative linear velocity and angular velocity.

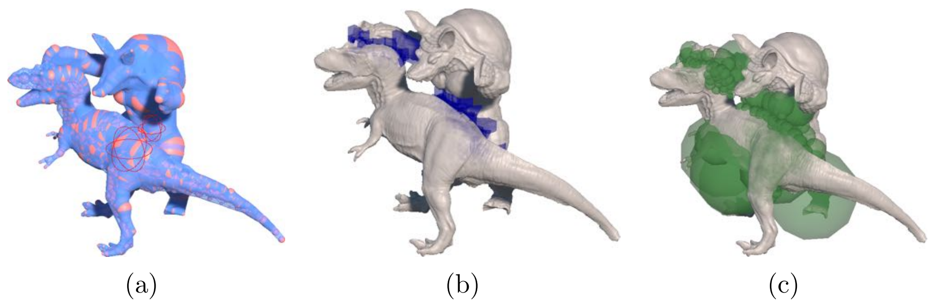

Figure 12: MAT, OBBs and Bounding Spheres. (a) Interpolated MAT. (b) and (c) BVHs of OBBs and Bounding Spheres. Only the intersected leaf-level bounding boxes/spheres are highlighted.

convex hull can simplify object’s shape, it will lose the concave properties of triangle mesh and lead to “ghost” collision artifacts. The comparison is shown in Fig. 15. In this animation, fifteen rings randomly drop on top

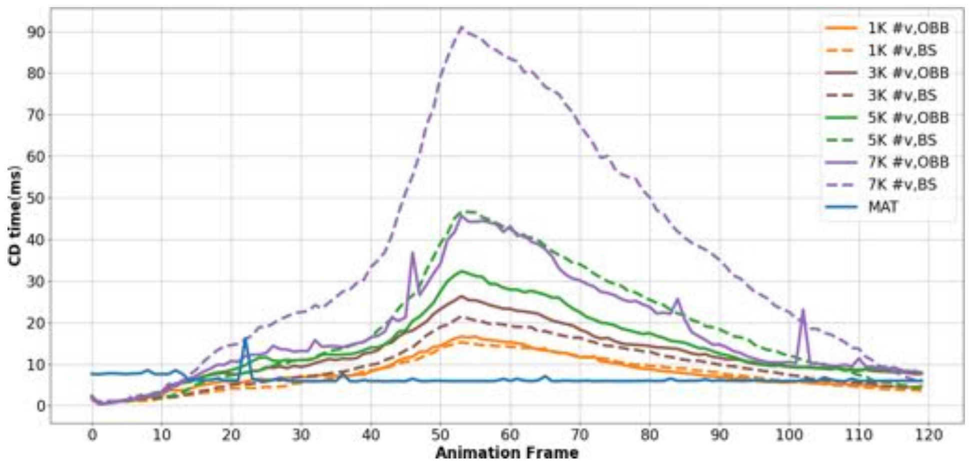

Figure 13: Timing statistics of MAT, OBBs and Bounding Spheres on the colliding T-rex and Armadillo scene. #v is the number of vertices of the triangle mesh after remeshing. Average CCD time of MAT: 6.72ms. From #v=1k to #v=7k, average CCD time of OBBs: 7.85ms, 12.24ms, 14.16ms, 18.45ms. Average CCD time of Bounding Spheres: 7.45ms, 9.83ms, 18.71ms, 36.53ms.

of a spider model. Many "ghost" collision artifacts happen when using the bounding convex hull algorithm.

In all these benchmarks, we are computing the continuous collision detection directly on simplified medial mesh instead of the original triangle mesh, due to its excellent bounding capacity as shown in Fig. 10. Since the medial mesh will produce much less elementary pairs for testing than triangle mesh, our algorithm improves the performance of collision detection significantly while preventing penetration and tunnelling effects.

5.4. Comparison of MATs with different qualities

Different MAT generation methods will produce MATs of different quality, which is reflected in the number of vertices of the medial mesh and the accuracy of the original mesh approximation. To illustrate the influence of these methods on our algorithm, we use Q-MAT to generate a series of MATs of different quality and simulate under the same motion. The simulation results and detailed statistics are shown in Fig. 16 and Table. 4 respectively. There are noticeable visual artifacts like intersection of original triangle mesh or

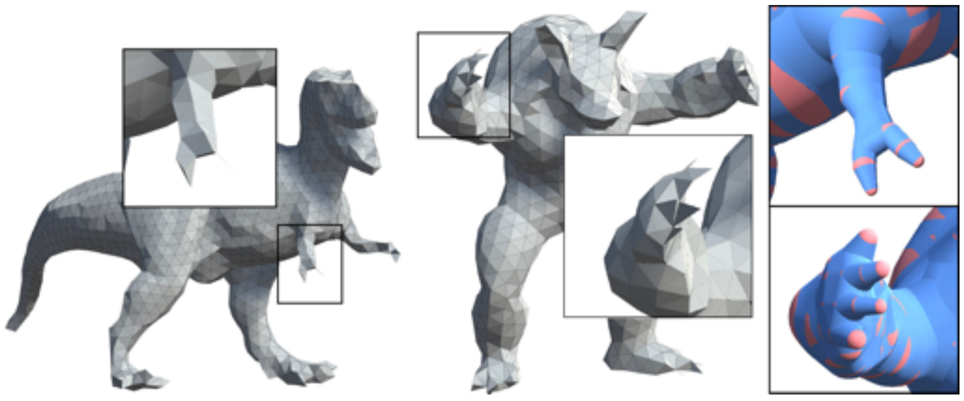

Figure 14: With both triangle mesh and medial mesh having same number of vertices(1k), low resolution triangle mesh will collapse in some sharp areas while MAT offers high quality approximation.

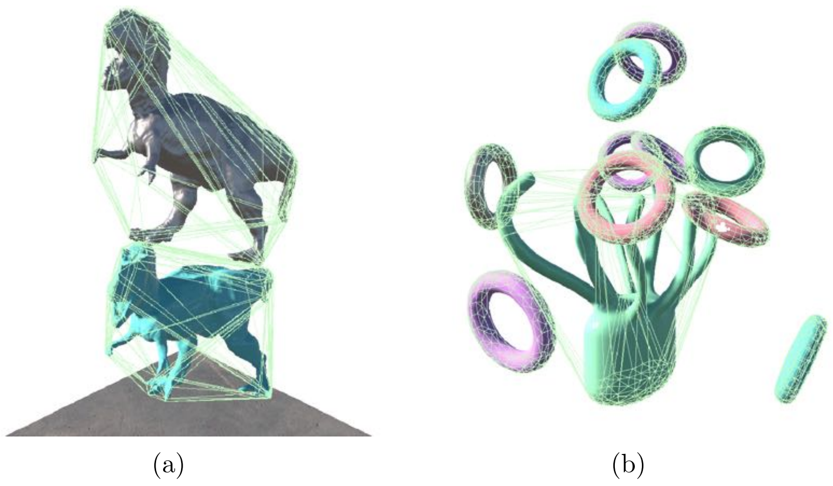

Figure 15: Simulations based on the bounding convex hull. The concave properties of T-rex, ring, and spider are basically lost.

"ghost" collision when medial mesh has low resolution. But while high resolution offers better approximation, it will also increase the time-consumption of the collision detection phase. Like triangle-level CCD, we have to trade off between quality and speed.

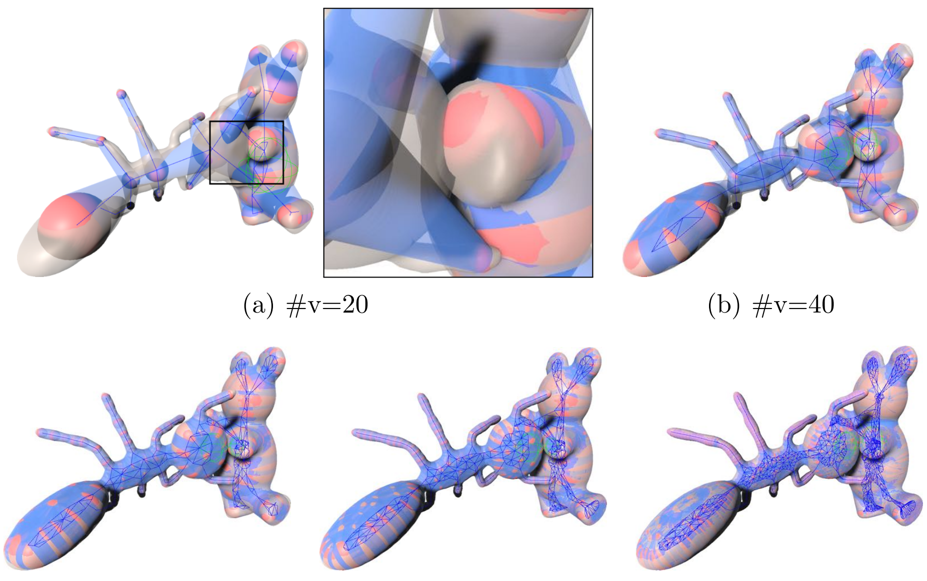

Figure 16: Simulation with MAT of different quality. Both interpolated MAT and original triangle mesh are rendered to illustrate results visually. Medial meshes are highlighted in blue and colliding sphere pairs are highlighted in green. **#v** is the number of vertices of the medial mesh and 2 models share the same number of **#v**.

6. Limitation and Future Work

In this paper, we present a MAT-based continuous collision detection algorithm for rigid body simulation. Our algorithm directly performs MAT-level collision detection, instead of using MAT as a culling method on triangle mesh [23]. With small amount of medial primitives we can approximate the original mesh accurately while producing much less pairs of elementary tests than triangle-level CCD. Our algorithm significantly improves the efficiency of CCD, and also avoids losing concave properties like convex hull method.

However, our algorithm also has two main limitations. On the one hand, since we assume the motion of triangles is linearly interpolated with constant velocity within a time interval, numerical errors due to rotational motion are sensitive to the size of time step, especially when the rigid bodies rotate quickly. It is a common problem for most CCD algorithms. On the other hand, our first time-of-contact does not have an analytical solution. The

	#EPs.	CD	HE		# v.
			Bug	Bear	
Fig. 16(a)	1,273	0.949	0.168	0.079	20
Fig. 16(b)	21,801	1.525	0.047	0.056	40
Fig. 16(c)	254,749	1.837	0.026	0.038	100
Fig. 16(d)	1,516,775	4.003	0.015	0.018	200
Fig. 16(e)	21,821,570	17.814	0.01	0.013	500

Table 4: Statistics of the simulation with MAT of different quality. **#EPs.** is the number of elementary pair. **CD** is the average collision detection time (ms). **HE** is the Hausdorff error. #**v.** is the vertex number of medial mesh. When #**v.** is small, its increase will greatly reduce the Hausdorff error without much performance degradation. But with the continuous increase of # **v.**, its impact on Hausdorff error decreases and leads to a large drop in efficiency performance.

use of iteration-based numerical computation for each elementary test is not friendly for parallel computation on GPUs. In addition to addressing these limitations, we plan to generalize the MAT-based CCD algorithm to deformable model simulation in the future.

7. Acknowledgements

The project is supported by the science technology bureau of Xiamen municipal government in 2018 (No. 3502Z20184058), the natural science foundation of science technology bureau of Fujian province in 2019 (No. 2019J01601), the foreign cooperation project of science technology bureau of Fujian province in 2018 (No. 2018I0015) and the creation fund project of science technology bureau of Fujian province in 2018 (No. 2019C0021).

References

[1] Nina Amenta and Marshall Bern, *Surface reconstruction by voronoi filtering*. Discrete & Computational Geometry, **22**(4):481–504, 1999.

[2] David Baraff, *An introduction to physically based modeling: rigid body simulation II—nonpenetration constraints*. *SIGGRAPH course notes*, pages D31–D68, 1997.

[3] C Bradford Barber, David P Dobkin, and Hannu Huhdanpaa, *The quickhull algorithm for convex hulls. ACM Transactions on Mathematical Software (TOMS)*, 22(4):469–483, 1996.

[4] Jernej Barbič and Doug L James, *Subspace self-collision culling. ACM Trans. Graph. (TOG)*, **29**(4):81, 2010.

[5] Harry Blum, *A transformation for extracting new descriptors of shape. Models for Perception of Speech and Visual Forms, 1967*, pages 362–380, 1967.

[6] Robert Bridson, Ronald Fedkiw, and John Anderson, *Robust treatment of collisions, contact and friction for cloth animation. ACM Trans. Graph.*, **21**(3):594–603, July 2002.

[7] Tyson Brochu, Essex Edwards, and Robert Bridson, *Efficient geometrically exact continuous collision detection. ACM Trans. Graph.*, **31**(4), July 2012.

[8] Sean Curtis, Rasmus Tamstorf, and Dinesh 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.

[9] Mark de Berg, Joao Comba, and Leonidas 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.

[10] H. Ding, H. Mitake, and S. Hasegawa, *Continuous collision detection for virtual proxy haptic rendering of deformable triangular mesh models. IEEE Transactions on Haptics*, **12**(4):624–634, 2019.

[11] Noura Faraj, Jean-Marc Thiery, and Tamy Boubekeur, *Progressive medial axis filtration.* In *SIGGRAPH Asia 2013 Technical Briefs*, page 3. ACM, 2013.

[12] Stefan Gottschalk, Ming C Lin, and Dinesh Manocha, *Obbtree: A hierarchical structure for rapid interference detection.* In *Computer graphics and interactive techniques*, pages 171–180. ACM, 1996.

[13] Qi Guo, Xuchen Han, Chuyuan Fu, Theodore Gast, Rasmus Tamstorf, and Joseph Teran, *A material point method for thin shells with frictional contact. ACM Trans. Graph.*, **37**(4), July 2018.

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

[15] Wenzel Jakob, Marco Tarini, Daniele Panozzo, and Olga Sorkine-Hornung, *Instant field-aligned meshes. ACM Transactions on Graphics (Proceedings of SIGGRAPH ASIA)*, **34**(6), November 2015.

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

[17] Chenfanfu Jiang, Theodore Gast, and Joseph Teran, *Anisotropic elastoplasticity for cloth, knit and hair frictional contact. ACM Trans. Graph.*, **36**(4), July 2017.

[18] Daniel Kopta, Thiago Ize, Josef Spjut, Erik Brunvand, Al Davis, and Andrew 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.

[19] Shankar Krishnan, M Gopi, M Lin, Dinesh 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.

[20] Thomas Larsson and Tomas Akenine-Möller, *A dynamic bounding volume hierarchy for generalized collision detection. Comput. Graph.*, **30**(3):450–459, June 2006.

[21] Christian Lauterbach, Michael Garland, Shubhabrata Sengupta, David P. Luebke, and Dinesh Manocha, *Fast bvh construction on gpus. Computer Graphics Forum*, **28**(2):375–384, 2009.

[22] Christian Lauterbach, Qi Mo, and Dinesh Manocha, *gproximity: Hierarchical gpu-based operations for collision and distance queries. Comput. Graph. Forum*, **29**:419–428, 05 2010.

[23] Lan Lei, Luo Ran, Fratarcangeli Marco, Xu Weiwei, Wang Huamin, Guo Xiaohu, Yao Junfeng, and Yang Yin, *Medial elastics: Efficient and collision-ready deformation via medial axis transform. ACM Trans. Graph.*, **39**(3), April 2020.

[24] Minchen Li, Zachary Ferguson, Teseo Schneider, Timothy Langlois, Denis Zorin, Daniele Panozzo, Chenfanfu Jiang, and Danny Kaufman, *Incremental potential contact: Intersection-and inversion-free, large-deformation dynamics. ACM Transactions on Graphics*, **39**:20, 07 2020.

[25] Pan Li, Bin Wang, Feng Sun, Xiaohu Guo, Caiming Zhang, and Wen-ping Wang, *Q-Mat: Computing medial axis transform by quadratic error minimization. ACM Trans. Graph. (TOG)*, **35**(1):8, 2015.

[26] 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.

[27] Simon Pabst, Artur Koch, and Wolfgang Straßer, *Fast and scalable cpu/gpu collision detection for rigid and deformable surfaces. Comput. Graph. Forum*, **29**:1605–1612, 07 2010.

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

[29] Svetlana Stolpner, Paul Kry, and Kaleem Siddiqi, *Medial spheres for shape approximation. IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(6):1234–1240, 2012.

[30] Min Tang, Sean Curtis, Sung-Eui Yoon, and Dinesh 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.

[31] Min Tang, Young J. Kim, and Dinesh Manocha, *Ccq: Efficient local planning using connection collision query.* In *In 9th Workshop on the Algorithmic Foundations of Robotics (WAFR'10*, pages 229–247, 2010.

[32] Min Tang, Dinesh Manocha, Sung-Eui Yoon, Peng Du, Jae-Pil Heo, and Ruofeng Tong, *VolCCD: Fast continuous collision culling between deforming volume meshes. ACM Trans. Graph.*, **30**:111:1–111:15, May 2011.

[33] Min Tang, Ruofeng Tong, Zhendong Wang, and Dinesh Manocha, *Fast and exact continuous collision detection with bernstein sign classification. ACM Trans. Graph.*, **33**(6), November 2014.

[34] Gino van den Bergen, *Efficient collision detection of complex deformable models using aabb trees. J. Graph. Tools*, **2**(4):1–13, January 1998.

[35] Pascal VOLINO and Nadia Magnenat THALMANN, *Efficient self-collision detection on smoothly discretized surface animations using geometrical shape regularity. Computer Graphics Forum*, 1994.

[36] Huamin Wang, *Defending continuous collision detection against errors. ACM Trans. Graph.*, **33**(4), July 2014.

[37] Tongtong Wang, Zhihua Liu, Min Tang, Ruofeng Tong, and Dinesh Manocha, *Efficient and reliable self-collision culling using unprojected normal cones. Computer Graphics Forum*, **36**(8), 2017.

[38] Xinlei Wang, Min Tang, Dinesh Manocha, and Ruofeng Tong, *Efficient bvh-based collision detection scheme with ordering and restructuring. Computer Graphics Forum*, **37**:227–237, 05 2018.

[39] Sai-Keung Wong, Wen-Chieh Lin, Chun-Hung Hung, Yi-Jheng Huang, and Shing-Yeu Lii, *Radial view based culling for continuous self-collision detection of skeletal models. ACM Trans. Graph.*, **32**(4), July 2013.

[40] Hongyi Xu and Jernej Barbič, *6-dof haptic rendering using continuous collision detection between points and signed distance fields. IEEE Transactions on Haptics*, **10**(2):151–161, 2017.

[41] Baorong Yang, Junfeng Yao, and Xiaohu Guo, *DMAT: Deformable Medial Axis Transform for Animated Mesh Approximation. Computer Graphics Forum*, 2018.

[42] Chee K. Yap and Vikram Sharma, *Robust Geometric Computation*, pages 1860–1863. Springer New York, New York, NY, 2016.

[43] Sung-Eui Yoon, Sean Curtis, and Dinesh 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.

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

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