Comput. Methods Appl. Mech. Engrg. 268 (2014) 574–588

Contents lists available at ScienceDirect

Comput. Methods Appl. Mech. Engrg.

journal homepage: www.elsevier.com/locate/cma

A hierarchical detection framework for computational contact mechanics

Alejandro M. Aragón [a,*], Jean-François Molinari [a,b]

[a] School of Architecture, Civil and Environmental Engineering (ENAC), École polytechnique fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland
[b] School of Engineering (STI), École polytechnique fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland

ARTICLE INFO

Article history:
Received 8 November 2012
Received in revised form 24 September 2013
Accepted 3 October 2013
Available online 22 October 2013

Keywords:
Contact mechanics
Contact detection
Collision detection
Bounding volume hierarchies
Finite element method
Closest-point projection

ABSTRACT

A novel methodology consisting of three hierarchical levels is proposed for the detection phase of contact mechanics simulations. The top level of the hierarchy uses kinematic information from the objects involved in the simulation to determine approximate collision times. These instants then determine when the engine resumes operation for further detection. By using bounding volume hierarchies, the second level of detection precludes contact by computing simple exclusion tests on bounding volumes of increasing tightness. When contact cannot be ruled out by using simple tests, the final level of detection comes into effect by using thorough checks on finite element primitives. To that purpose, a robust optimization-based formulation that does not rely on orthogonal projections is outlined. The detection framework can be used to predict the exact collision time among finite element discretizations. The performance of the proposed methodology is investigated with a set of examples.

1. Introduction

The science of *contact mechanics* originated in the early 1880s with the work of Hertz [1], who solved the problem of two elastic bodies with convex boundaries in contact. The advent of computers, and the development of the finite element method (FEM) for the simulation of a wide range of engineering problems, gave rise to the field of *computational contact mechanics*. The latter aims at providing the computational means for the prediction of the physical processes that occur when simulated solid bodies come into contact with one another. Diverging from a purely theoretical framework, the latter has found applications in a wide range of areas, including the analysis of crashworthiness [2], metal forming [3,4], shell structures [5], and impact and penetration [6]. The implementation of a contact mechanics algorithm can be roughly subdivided into two major components: (i) *contact detection*, and (ii) *contact resolution*. This article discusses the first major component, as the detection phase can encompass a major portion of the CPU time in the simulation of contact, specially with explicit integration time stepping algorithms [7]. As an example, Attaway et al. [8] allot 30% to 60% of the computational time to the contact detection phase on the Cray Y-MP vector supercomputer.

In computer science, *collision detection* describes the procedure by which the intersection between at least two objects is determined. Collision detection has found applications in computer graphics visualization [9,10], virtual reality [11], computer-aided design (CAD) [12,13], game development [14], and robotics [15–17]. Collision detection checks are typically carried out following a space decomposition or on hierarchies of bounding volumes, or even a combination thereof. The objective of the former is to subdivide the search space so that the number of collision checks is minimized. Choices for space

* Corresponding author. Tel.: +41 21 693 24 50.
E-mail addresses: alejandro.aragon@fulbrightmail.org (A.M. Aragón), jean-francois.molinari@epfl.ch (J.-F. Molinari).

0045-7825/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.cma.2013.10.001

decomposition can consist of tree data structures, including binary space partition (BSP) trees [18,19], k-d trees [20,21,11,22], octrees [23–25], R*-trees [26], tetrahedral meshing [11], and grids [27]. Bounding volumes typically chosen for building hierarchies include bounding spheres (BSs) [9], axis-aligned bounding boxes (AABBs) [27], discrete orientation polytopes (k-DOPs) [10], oriented bounding boxes (OBBs) [28], and convex hulls [29]. For survey articles on collision detection, the reader is referred to [30–32] and the references therein. Also, a thorough treatment on collision detection can be found in the book by Ericson [33].

The *contact resolution* is the phase of the simulation that determines the response of colliding objects. Under the terminology of *collision response*, the computer science literature lists a substantial body of work, starting from the early works on the response between rigid bodies [24,34–36], to the study of deformable bodies [37–39], cloth animation [40–44], fragmentation [45,46] and even hair assemblies [47,48]. Even though most of the works on collision response in the computer graphics literature are simplistic in regards to the physics, some newer articles address the response in a more rigorous manner, e.g., by including adhesive contact [49] and friction [50,51,47,48,44]. The finite element method has also been used for more accurate representations of the response [39,44]. Note that this is by no means an exhaustive list.

This article emphasizes on the contact detection phase, and as expressed above, there is a wealth of knowledge from the computer science community that could successfully be applied to the field of computational contact mechanics. We will focus mainly on collision checks on bounding volume hierarchies, as this procedure can be combined with any aforementioned space decomposition technique. Gottschalk et al. [28] employed hierarchies of OBBs and showed that they outperform the use of BS and AABB hierarchies in situations of close proximity. Klosowski et al. [10] studied the use of bounding volume trees based on k-DOPs and showed that their algorithm can perform the collision detection of hundreds of thousands of polygons at interactive rates. Otaduy and Lin [52] introduced the idea of *contact levels of detail* (CLOD), by which they use a dual hierarchy of the model for accelerating the collision detection phase. One hierarchy is a multiresolution representation of the model, while the other one uses bounding volumes. In a recent publication, Larsson and Akenine-Möller [53] combine the BSs, OBBs, and k-DOPs into *slab cut balls*, which are hybrid bounding volumes that outperform both BS and OBB hierarchies.

In the computational contact mechanics literature there is only a handful of works that adopt some of this knowledge. Grids are the space decomposition methodology of choice [5,54]. The straightforward implementation of this technique, that is also referred to as *bucket search*, may be the culprit for its prevailing adoption among computational mechanicians. A global contact search methodology based on the sorting of slave nodes is also described by Heinstein et al. [54]. Yang and Laursen [7] have recently implemented the detection based on hierarchies of k-DOPs in the context of contact simulations based on mortar formulations. Nevertheless, it is the view of the authors that there is a considerable time lag between the two fields of study. This manuscript is the authors' attempt on using some of the newer knowledge gathered by the computer science community and apply it to the field of computational contact mechanics.

In this article we propose a novel three-level detection phase for contact mechanics. At the upper level, a *trajectory based* approach is used to compute the approximate time the detection engine initiates to operate. In the literature, a fixed time step is normally used to perform the detection, and collisions can be missed depending on the choice of this parameter. By determining an approximate time of a collision, the proposed approach is robust as it guarantees the detection of all collisions. A second level of detection consists of carrying out intersection tests on hierarchies of bounding volumes. Initially, the simplest bounding volumes are used, and adaptivity is applied to the bounding volumes as more precise detection tests are needed. The final stage of the detection phase consists of thorough checks for actual contact, by computing distances from nodes to finite elements using an optimization-based approach. An appropriate contact detection engine has to be both efficient and accurate [15], for it is important that precise contact detections are determined with the least amount of time. In the proposed scheme, the upper two layers are concerned with efficiency, while the last layer deals with accuracy. The proposed strategy can prove invaluable in speeding up this computationally-intensive stage of physics simulations involving contact.

The article is organized as follows: the description of the contact problem, together with mathematical definitions of bounding volumes and their corresponding hierarchies, is provided in Section 2. Section 3 outlines the proposed hierarchical approach to contact detection. Implementation details of the framework are given in Section 4. Finally, Section 5 presents some examples of the use of the proposed methodology.

2. Problem description

Let $\mathbb{R}^d$ represent the d-dimensional Euclidean space, where a coordinate is represented by $\boldsymbol{x} = x_i \boldsymbol{e}_i$, and $\boldsymbol{e}_i$ is a chosen orthogonal basis. The space is equipped with inner product $\langle \cdot, \cdot \rangle$ that induces the norm $\|\boldsymbol{x}\| = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle}$. Let $\Omega_i \subset \mathbb{R}^d$ be the mathematical representation of the ith object involved in the simulation, as shown in Fig. 1. The boundary of the domain $\partial\Omega_i := \overline{\Omega}_i \setminus \Omega_i$, with unit normal $\boldsymbol{n}_i$, can be decomposed into mutually exclusive regions $\partial\Omega_i^u \cup \partial\Omega_i^t \cup \partial\Omega_i^c$. The regions $\partial\Omega_i^u$ and $\partial\Omega_i^t$ correspond to those where Dirichlet and Neumann boundary conditions are prescribed, respectively. The contact constraints are enforced in $\partial\Omega_i^c$, that depending on the problem, may be equal to $\emptyset$ most of the simulation time.

For each individual domain Ω_i, we are interested in obtaining its displacement $\boldsymbol{u}(\boldsymbol{x}, t) : \Omega_i \times T \to \mathbb{R}^d$, and corresponding velocity $\dot{\boldsymbol{u}}$ and acceleration $\ddot{\boldsymbol{u}}$ fields, for a time period $T \subset \mathbb{R}$. The initial elasto-dynamics boundary value problem is stated as: given the body density $\rho_i : \Omega_i \to \mathbb{R}$, body force $\boldsymbol{b}_i : \Omega_i \times T \to \mathbb{R}^d$, initial displacement and velocity $\boldsymbol{u}_i^0, \dot{\boldsymbol{u}}_i^0 : \Omega_i \to \mathbb{R}^d$,

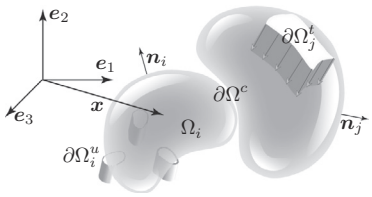

Fig. 1. Schematic of two solid bodies in contact.

prescribed displacement $\tilde{\boldsymbol{u}}_i : \partial\Omega^u_i \times T \to \mathbb{R}^d$, and prescribed traction $\tilde{\boldsymbol{t}}_i : \partial\Omega^t_i \times T \to \mathbb{R}^d$, find the displacement field $\boldsymbol{u}_i : \overline{\Omega}_i \times T \to \mathbb{R}^d$, such that $\forall\Omega_i$:

$$\begin{aligned}
&\rho_i\ddot{\boldsymbol{u}}_i = \nabla\cdot\boldsymbol{\tau}_i + \boldsymbol{b}_i && \text{in } \Omega_i \times T,\\
&\boldsymbol{u}_i = \tilde{\boldsymbol{u}}_i && \text{on } \partial\Omega^u_i \times T,\\
&\boldsymbol{\tau}_i\boldsymbol{n}_i = \tilde{\boldsymbol{t}}_i && \text{on } \partial\Omega^t_i \times T,\\
&\boldsymbol{u}_i(\boldsymbol{x},0) = \boldsymbol{u}^0_i && \text{in } \Omega_i,\\
&\dot{\boldsymbol{u}}_i(\boldsymbol{x},0) = \dot{\boldsymbol{u}}^0_i && \text{in } \Omega_i,
\end{aligned} \tag{1}$$

with Hertz-Signorini-Moreau conditions for frictionless contact [55]

$$\begin{aligned}
&\gamma_i \geqslant 0\\
&\boldsymbol{\tau}_i\cdot\boldsymbol{n}_i \leqslant 0 && \text{on } \partial\Omega^c_i \times T,\\
&\gamma_i\boldsymbol{\tau}_i\cdot\boldsymbol{n}_i = 0
\end{aligned} \tag{2}$$

where $\nabla\cdot\boldsymbol{\tau}_i$ designates the divergence of the stress tensor $\boldsymbol{\tau}_i : \overline{\Omega}_i \to \mathbb{R}^d \times \mathbb{R}^d$, and γ_i the normal gap function associated with the contact boundary.

The subscript i that has been used so far to represent the ith body is removed henceforth in order to ease the formulation, and thus the following definitions apply to a single object. A finite element discretization Ω^h, that approximates Ω as closely as possible, is defined by its corresponding sets of finite elements $\mathscr{E} = \{\omega_i\}$ and nodes $\mathscr{N} = \{v_i\}$. Note that there is a coordinate $\boldsymbol{x}_i$ associated with each node v_i, so from this point forward they are considered equivalent and used interchangeably in the article. The discretization has boundary $\partial\Omega^h := \overline{\Omega}^h \setminus \Omega^h$, where the closure of the discretization is $\overline{\Omega}^h \equiv \cup_i\overline{\omega}_i$. Analogously, given an element ω, its boundary $\partial\omega$ can be decomposed into faces φ_i such that $\overline{\partial\omega} = \cup_i\overline{\varphi}_i$. It is the boundary $\partial\Omega^h$ that can come in contact with itself or with other object boundaries, so the most fundamental components used in the contact detection phase are the faces of the finite elements that intersect the boundary. Element faces are therefore the lower dimensional manifolds that are used as primitives for the computation of bounding volumes. For a given discretization Ω^h, let the contact surface be defined as $\sigma := \left\{\varphi_i | \varphi_i \cap \partial\Omega^h \neq \emptyset\right\}$. Let n be the number of nodes of the discretization that belong to the contact surface, *i.e.*, $n = |\mathscr{N} \cap \sigma|$.

In this manuscript emphasis is placed on the contact detection phase, so a standard finite element formulation for the elasto-dynamics problem is used. Thus, without entering into the details of the discretization of the initial boundary value problem given by Eqs. (1) and (2), the matrix problem is stated as: find the global displacement vector $\mathbf{U}$ such that

$$\begin{aligned}
&\mathbf{M}\ddot{\mathbf{U}} + \mathbf{F}^I(\mathbf{U}) + \mathbf{F}^K(\mathbf{U}) = \mathbf{F}^E \quad t \in T,\\
&\mathbf{U} = \mathbf{U}^0,\\
&\dot{\mathbf{U}} = \dot{\mathbf{U}}^0,
\end{aligned} \tag{3}$$

where $\mathbf{M}$ is the mass matrix, and $\mathbf{F}^I, \mathbf{F}^E, \mathbf{F}^K$ are the internal, the external, and the contact force vectors, respectively. These are obtained by means of the finite element assembly operator $\mathbb{A}$ of element local matrices and vectors. The displacement vector $\mathbf{U}$ is obtained here by the Newmark explicit predictor–corrector scheme, even though the detection methodology is general and can be used with any time-integration scheme. For additional details on the mathematical formulation of the FEM, the reader is advised to consult classical textbooks on the subject [56,57].

Bounding volumes and their hierarchical representations

A sphere $s \subset \mathbb{R}^d$, with radius r and center $\boldsymbol{x}^c$, is defined as

$$s(r,\boldsymbol{x}^c) := \left\{\boldsymbol{x} \in \mathbb{R}^d \,\middle|\, \|\boldsymbol{x} - \boldsymbol{x}^c\| \leqslant r\right\}. \tag{4}$$

Thus, given a discretization Ω^h, there exists a bounding sphere that encloses all nodes in the boundary with minimum volume

$$V^\circ := s(\bar{r},\bar{\boldsymbol{x}}^c) \supset \Omega^h, \tag{5}$$

where

$$\{\bar{r}, \bar{\boldsymbol{x}}^c\} = \underset{r, \boldsymbol{x}^c}{\operatorname{argmin}}\ V(r) = 4\pi r^3/3, \tag{6}$$

$$\text{such that } \|v_i - \boldsymbol{x}^c\| \leqslant r, \quad \forall v_i \in \sigma. \tag{7}$$

Given a set of n points, efficient algorithms have been proposed to obtain tightly fitted bounding spheres with time complexity $\mathcal{O}(n)$ [58,59].

Given a basis $\boldsymbol{e}_i \in \mathbb{R}^d$, the vector $\boldsymbol{p} = p_i \boldsymbol{e}_i$ represents an axis over which we would like to project a discretization Ω^h. Thus, defining the projection operator

$$\operatorname{proj}_{\boldsymbol{p}} \boldsymbol{x} := \frac{\langle \boldsymbol{x}, \boldsymbol{p} \rangle}{\|\boldsymbol{p}\|^2} \boldsymbol{p}, \tag{8}$$

the portion of the Euclidian space where there is a non-empty orthogonal projection of the body to the vector is defined as

$$S^{\boldsymbol{p}}\left(\Omega^h\right) := \left\{ \boldsymbol{x} \in \mathbb{R}^d \middle| a^{\boldsymbol{p}} \leqslant \left\| \operatorname{proj}_{\boldsymbol{p}} \boldsymbol{x} \right\| \leqslant b^{\boldsymbol{p}} \right\} \subset \mathbb{R}^d, \tag{9}$$

where the bounds $a^{\boldsymbol{p}}, b^{\boldsymbol{p}} \in \mathbb{R}$ are given by

$$a^{\boldsymbol{p}} := \inf \left\{ \bigcup_{v_i \subset \sigma} \|\operatorname{proj}_{\boldsymbol{p}} v_i\| \right\}, \quad b^{\boldsymbol{p}} := \sup \left\{ \bigcup_{v_i \subset \sigma} \|\operatorname{proj}_{\boldsymbol{p}} v_i\| \right\}. \tag{10}$$

In other words, $S^{\boldsymbol{p}}$ defines a slab in Euclidean space that has a non-empty orthogonal projection between the vectors $a^{\boldsymbol{p}} \bar{\boldsymbol{p}}$ and $b^{\boldsymbol{p}} \bar{\boldsymbol{p}}$, with $\bar{\boldsymbol{p}} = \boldsymbol{p}/\|\boldsymbol{p}\|$. With the definitions given by Eqs. (9) and (10), and given a set of vectors $\boldsymbol{P} = \{\boldsymbol{p}_j, j = 1, k/2\}$, a discrete orientation polytope representation of Ω^h, of degree k, is defined as

$$V^k := \prod_{\boldsymbol{p}_j \in \boldsymbol{P}} S^{\boldsymbol{p}_j}\left(\Omega^h\right) = \left\{ \boldsymbol{x} \in \mathbb{R}^d \middle| \boldsymbol{x} \in \bigcap_{\boldsymbol{p}_j \in \boldsymbol{P}} S^{\boldsymbol{p}_j}\left(\Omega^h\right) \right\} \supseteq \Omega^h, \tag{11}$$

where $\prod$ denotes the cartesian product. In other words, a k-DOP represents the subset of the Euclidean space that contains the intersection of $k/2$ slabs.

An axis-aligned bounding box in d-dimensional space can be represented as V^{2d}, where the projection axes are chosen as the Cartesian basis $\boldsymbol{e}_i$. Both bounding spheres and axis-aligned bounding boxes may not fit the discretization tightly, so higher-order k-DOPs can be defined to reduce the enclosing volume. These can be defined by adding projection vectors on discrete directions. In $\mathbb{R}^2$, a 8-DOP represents the volume enclosed by the projections on the basis vectors $\boldsymbol{e}_\alpha, \alpha = 1, 2$, and the two vectors $\{1, \pm 1\}^{\mathsf{T}}$. Similarly, a 14-DOP in $\mathbb{R}^3$ uses the basis $\boldsymbol{e}_i$ and the four vectors $\{1, \pm 1, \pm 1\}^{\mathsf{T}}$. The latter two, denoted V^8 and V^{14} depending on the dimension, can be considered as a first order improvement from an AABB, for a tighter enclosure is obtained by eliminating volume from the corners. Klosowski et al. [10] studied even higher order k-DOPs in $\mathbb{R}^3$, namely V^{18} and V^{26}. An oriented bounding box, denoted as $V^{2d}_{\boldsymbol{g}}$ henceforth, is equivalent to a $2d$-DOP whose basis $\boldsymbol{g}_i$ has been chosen carefully as to reduce the enclosing volume of the discretization as much as possible. The basis of the OBB can be related to the main basis $\boldsymbol{e}_j$ through the orthogonal transformation $[\mathbf{Q}]_{ij} = \langle \boldsymbol{g}_i, \boldsymbol{e}_j \rangle$.

Let $\mathscr{T}$ denote the graph that represents a rooted K-ary tree data structure that is used to define the bounding volume hierarchy of a discretization Ω^h. K denotes the maximum number of children per vertex of the tree, e.g., $K = 2$, for a binary tree, $K = 3$ for a ternary tree, and so forth. Formally, $\mathscr{T} := \{\mathscr{V}, \mathscr{E}\}$, where $\mathscr{V}$ and $\mathscr{E}$ designate sets of vertices and edges, respectively. Each vertex $v_j \in \mathscr{V}(\mathscr{T})$ denotes a bounding volume, with the one for the entire discretization corresponding to the root of the tree. The hierarchy extends down to the tree leaves v^k, which correspond to the bounding volumes of primitive objects used in the discretization, *i.e.*, those of finite element faces. The height of a balanced tree is $h(\mathscr{T}) := \lceil \log_K |\sigma| \rceil$, where $\lceil \cdot \rceil$ denotes the ceiling function. The bounding volume at a depth less than $h(\mathscr{T})$ thus contains the bounding volumes of its children, *i.e.*, $v^k_1, \ldots, v^k_K \subset v^k_{1\ldots K} \subset \ldots \subset V^k$.

3. A three-level contact detection engine

As mentioned in the introduction, a three-level contact detection phase is proposed in this work. This section describes in detail each level and concludes with an analysis on the total cost of the detection engine.

3.1. High-level: trajectory-based detection

The top-most level uses a trajectory-based approach to determine the time $t^\star \in T$, that is a prediction of the collision between the roots of bounding volume hierarchies. The methodology is illustrated schematically in Fig. 2, where the bounding spheres of bodies Ω_i and Ω_j intersect at time $t^\star$. Bounding spheres are chosen because they loosely fit the objects and can therefore accommodate their rotations without changing the volume. Given a bounding sphere V°_i of radius r_i, let its center

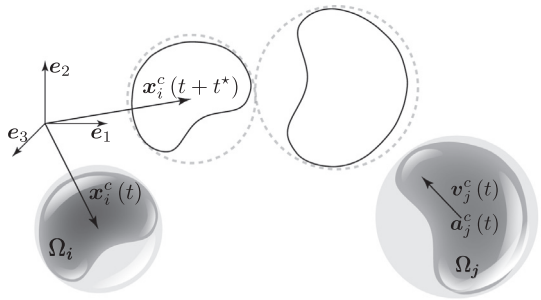

Fig. 2. Schematic of the high-level trajectory based detection, showing the collision between the bounding spheres of two objects Ω_i and Ω_j at time $t^\star$ from their initial positions at time t.

position, velocity, and acceleration be denoted by $\boldsymbol{x}_i^c$, $\boldsymbol{v}_i^c$ and $\boldsymbol{a}_i^c$, respectively. Note that these kinematic quantities are determined from the finite element results, but in general they do not necessarily correspond to those of the object's barycenter. The position of the ith sphere in time is thus determined by $\boldsymbol{x}_i^c(t) = \boldsymbol{x}_i^c + \boldsymbol{v}_i^c t + \boldsymbol{a}_i^c t^2/2$. At any point in time, the relative distance between two such spheres is given by $\boldsymbol{d}_{ij}(t) = \boldsymbol{x}_i^c(t) - \boldsymbol{x}_j^c(t)$ and their intersection (if any) is obtained at time $t^\star$ by solving the quartic equation in time

$$\boldsymbol{d}_{ij}(t^\star) \cdot \boldsymbol{d}_{ij}(t^\star) = (r_i + r_j)^2. \tag{12}$$

In the case that the bodies move with constant velocity, Eq. (12) reduces to a quadratic equation. Once the approximate collision time is found for every pair of bodies, the detection engine ceases operation until time min $(t^\star)$. As a result, the high-level detection has direct implications on the efficiency of the final framework, for no further work is needed until then.

The idea of space–time bounds in the context of collision detection is not new. While discussing an algorithm to find the minimum distance between two convex polyhedra, Schwartz [60] describes the idea of collision-free paths given the position of bodies as a function of time. Culley and Kempf [15] describe a methodology for detecting collisions using *velocity–distance* bounds, and thus adaptively advance the simulation time safely. Cameron [17] used the intersection of four-dimensional (space–time) structures to determine the collision between objects undergoing linear motion. The methodology presented here has some similarities with the *space–time bounds* employed by Hubbard in the collision detection for interactive graphics applications [9].

3.2. Middle-level: bounding volume hierarchical detection

The bounding spheres used in the trajectory-based detection level correspond to the roots of bounding volume hierarchies. At a given time, if any two of these spheres overlap, their hierarchies are traversed down from the root and collision is tested at deeper levels due to the better representation of the volumes in the collision zone. Intuitively, the cost of testing a pair of bounding volume hierarchies for overlap increases with the tightness of the bounding volumes that are used to represent the objects. The intersection of bounding spheres can be done rapidly, so they are chosen as a first approximation of the bounding volumes in the entire hierarchy. Furthermore, spheres can completely bound any significant translational and rotational changes in the objects without changing volume, which makes them suitable for when the objects are far from each other. When the traversal finds leaves in both hierarchies, collision is imminent and the leaves' bounding volumes are *refined* by increasing their degree of tightness for a more accurate collision test. At this point the interacting bodies are not expected to translate and rotate as before so their positions change slightly between time steps.

For the creation of the bounding volume hierarchies, this work adopts a bottom-up approach. Omohundro [61] conveys that this procedure constructs bounding sphere hierarchies with the least amount of volume. In a bottom-up strategy, bounding spheres are determined from all finite element primitives and then placed into an *online insertion tree*, which would later hold the entire hierarchy. In this dynamic tree data structure, nodes are inserted into a location that tries to minimize the volume of the entire arrangement. A priority queue is also used in aiding the construction of the final hierarchy. The *best pair* for each sphere is determined, defined as the one that minimizes their joint volume, and the resulting tuples are stored in the priority queue. An iterative procedure follows, checking the tuple from the priority queue with the highest priority, *i.e.*, that with the least amount of joint volume. If the first sphere of the tuple has not already been paired, its corresponding best pair is obtained with the aid of the online tree. If the best pair matches the stored best in the tuple, the sphere and its best pair are removed from the online tree, and linked into a parent node having their joint volume that is reinserted into the tree. The procedure continues until the entire bounding sphere hierarchy is built. Details about the implementation of the algorithm can be found in [61].

The methodology for the construction of the bounding volume hierarchy is illustrated in Fig. 3 for two meshes. Figs. 3a and 3b show the meshes that are used for the construction of the bounding sphere hierarchies displayed in the subsequent

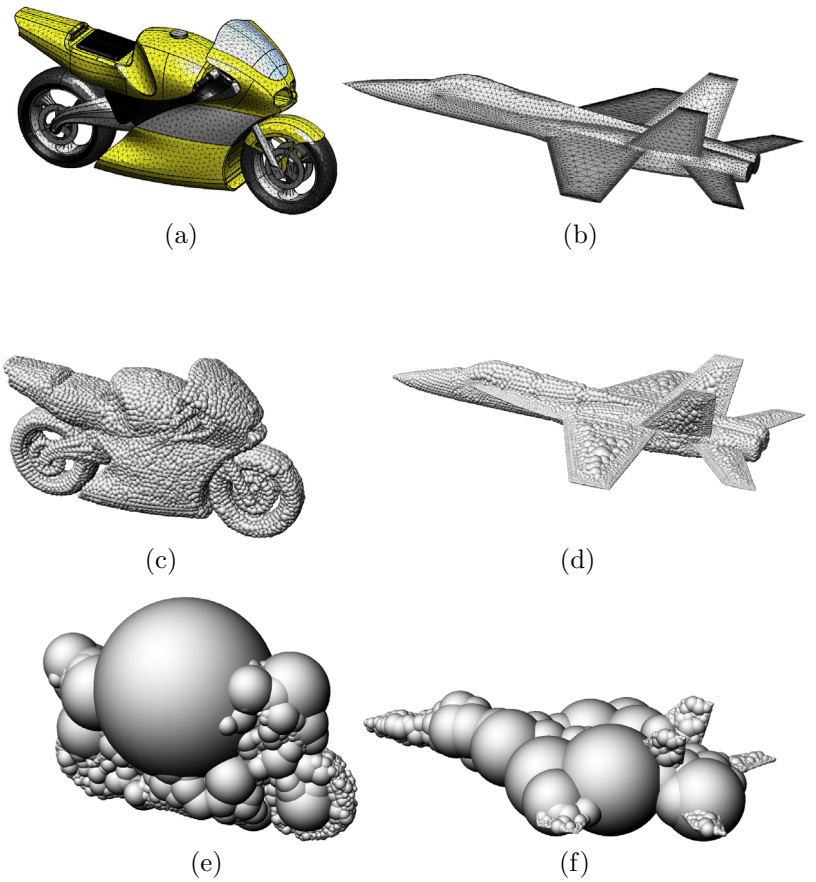

Fig. 3. Motorcycle (a) and jet (b) finite element meshes; (c), (d) bounding spheres at hierarchy leaves; (e), (f) twelfth level of their corresponding bounding sphere hierarchies.

figures. The resulting bounding sphere hierarchy for the mesh of the motorcycle is not balanced, resulting in an average height of 18, with minimum and maximum depths of 8 and 38, respectively. The hierarchy of the jet has average, minimum and maximum heights of 18, 8 and 43, respectively. The leaves of the hierarchies, which enclose the finite element primitives in the model, are displayed in Figs. 3c and 3d. Similarly, Figs. 3e and 3f show the bounding spheres at a level 12 below the root. The methodology using axis-aligned bounding boxes is again applied to the meshes of Fig. 3a and 3b, and the corresponding leaves of the resulting bounding volume hierarchies are displayed in Figs. 4a and 4b, respectively. Hierarchies of more complex bounding volumes can be constructed analogously. Nevertheless, the simpler the bounding volume used in the hierarchy, the fastest the computation of its corresponding intersection exclusion test. Some of the bounding volumes that can be used, together with their corresponding contact exclusion tests are described below.

3.2.1. Bounding sphere

This bounding volume encloses with a sphere all points of a particular discretization. The procedure of choice for the computation of bounding spheres in this work is due to Ritter [58], who describes a two-pass $\mathcal{O}(n)$ algorithm to compute an

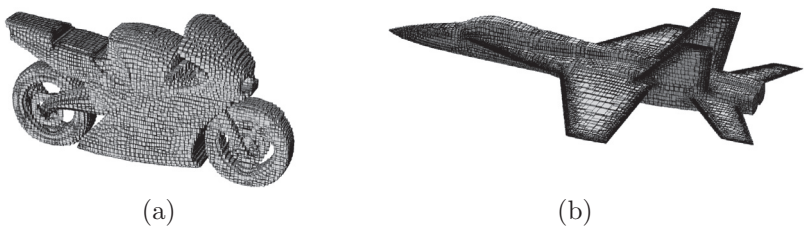

Fig. 4. Leaves of the axis-aligned bounding box hierarchies.

approximate bounding sphere. In a recent article by Larsson [59], a new methodology is proposed that produces tighter bounding spheres at roughly the same time as Ritter's algorithm. Larsson's *extremal points optimal sphere* algorithm has complexity $\mathcal{O}(kn)$, where k denotes the number of directions used for projections. A very small number of directions is needed to provide a tightly fitted sphere, leading to an overall $\mathcal{O}(n)$ complexity.

Contact between two bounding spheres V_i° and V_j° can be ruled out iff

$$\left\| \boldsymbol{x}_i^c - \boldsymbol{x}_j^c \right\| > r_i + r_j.$$

3.2.2. Axis-aligned bounding box

This method uses the Cartesian axes as the directions for the determination of the bounding volume. Its computation involves finding minimum and maximum coordinate values along all coordinate axes, with an overall complexity $\mathcal{O}(n)$.

Given two AABBs V_i^{2d}, V_j^{2d}, with corresponding bounds $a_{i,j}^{\boldsymbol{e}_k}, b_{i,j}^{\boldsymbol{e}_k}$ along the basis vector $\boldsymbol{e}_k$, contact is precluded iff

$$a_i^{\boldsymbol{e}_k} > b_j^{\boldsymbol{e}_k} \bigwedge a_j^{\boldsymbol{e}_k} > b_i^{\boldsymbol{e}_k},$$

totaling $2d$ tests.

3.2.3. Oriented bounding box

The technique is similar to the one presented in Section 3.2.2 once the rotated basis $\boldsymbol{g}_i$ is known. Gottschalk et al. [28] suggest computing the mean position and the covariance matrix, and then obtaining the eigenvectors of the latter to find this basis. Instead, a similar physics-based approach is proposed here using the concepts of barycenter and moment of inertia. Given a finite element discretization Ω^h, let $\imath(\mathcal{N})$ represent the index set of its nodes. The discretization barycenter is computed as

$$\boldsymbol{x}^g = \frac{1}{M} \sum_{i \in \imath(\mathcal{N})} m_i v_i, \tag{13}$$

where M is the mass of the entire domain, and m_i refers to the mass associated with node v_i. The moment of inertia tensor $\boldsymbol{I} : \Omega^h \to \mathbb{R}^d \times \mathbb{R}^d$ is computed as

$$[\boldsymbol{I}]_{jk} = \sum_{i \in \imath(\mathcal{N})} m_i \left(\delta_{jk} \|v_i\|^2 - v_{i,j} v_{i,k} \right), \tag{14}$$

where δ_{jk} is the Kronecker delta, and $v_{i,j}$ represents the jth coordinate of the ith node. Through the Huygens–Steiner theorem, the moment of inertia tensor with respect to barycentric axes is

$$[\boldsymbol{I}^g]_{jk} = [\boldsymbol{I}]_{jk} - M \left(\delta_{jk} \|\boldsymbol{x}^g\|^2 - x_j^c x_k^c \right). \tag{15}$$

The eigenvalue problem given by $\boldsymbol{I}^g \boldsymbol{w} = \lambda \boldsymbol{w}$ determines the principal moments of inertia λ_i, and their corresponding angular velocity vectors $\boldsymbol{w}_i$. The latter eigenvectors form the basis $\boldsymbol{g}_i = \boldsymbol{w}_i / \|\boldsymbol{w}_i\|$ that is used to compute the OBB of the discretization. Computing the barycenter and the moment of inertia tensor have both complexity $\mathcal{O}(n)$. Because of the dimensions of the inertia tensor, the determination of the eigenvectors has complexity $\mathcal{O}(1)$. Then, as it is the case with an AABB, finding the enclosing oriented bounding volume has complexity $\mathcal{O}(n)$. As a result, the entire operation has complexity $\mathcal{O}(n)$.

With two OBBs $V_{\boldsymbol{g}}^{2d}$ and $V_{\boldsymbol{h}}^{2d}$, obtained in basis $\boldsymbol{g}_i$ and $\boldsymbol{h}_i$, respectively, a sufficient condition for a contact rejection test is

$$a_i^{\boldsymbol{g}_k} > b_j^{\boldsymbol{g}_k} \bigwedge a_j^{\boldsymbol{g}_k} > b_i^{\boldsymbol{g}_k},$$

$$a_i^{\boldsymbol{h}_k} > b_j^{\boldsymbol{h}_k} \bigwedge a_j^{\boldsymbol{h}_k} > b_i^{\boldsymbol{h}_k},$$

for a total of $4d$ tests (over $2d$ directions) in the worst case. Note however that considering only these tests is conservative, for a rigorous contact rejection test would require projections over $2d + d^2$ directions due to the separation axis theorem [28]. The $4d$ tests above fail to report, e.g., that there is no collision when two OBBs approach one another edge to edge in $\mathbb{R}^3$. Nevertheless, not taking into account the additional directions is a conservative measure, as no collisions go undetected and it avoids extra computation due to cases that are uncommon in practice. Through numerical experiments, van den Bergen [62] reports that not considering the extra d^2 directions results in incorrect rejection tests only in about 6% of the time.

3.2.4. Higher order polytopes

With an overall complexity of $\mathcal{O}(kn)$, k-DOPs are obtained by carrying out projections over $k/2$ directions. For two polytopes V_i^k, V_j^k, the test for contact preclusion is

$$a_i^{\boldsymbol{p}_k} > b_j^{\boldsymbol{p}_k} \bigwedge a_j^{\boldsymbol{p}_k} > b_i^{\boldsymbol{p}_k},$$

for all $k/2$ discrete directions $\boldsymbol{p}_k$.

3.3. Low-level: optimization-based detection

When contact cannot be excluded from the tests on bounding volumes of the previous detection level, a more elaborate test is needed to discern if finite element primitives intersect with one another. The cost of testing primitives for contact is high because a thorough test comprises the need to compute distances between two contact surfaces. This computation may involve several nodes and finite elements, and a robust approach is required as the usual node-to-node approach can fail to detect the right contact location [54]. Consequently, distance computations that rely on orthogonal projections (often not uniquely defined) are usually carried out. Here, we propose a robust formulation for the computation of nodal distances to contact surfaces that can be used irrespective of the polynomial order of the underlying finite element mesh used for the discretization. The methodology does not rely on the existence of an orthogonal projection to the surface and can thus be used in the modeling of both smooth and non-smooth contact.

Consider in Fig. 5 a finite element discretization Ω^h of the domain, with contact surface σ. Let the point $\boldsymbol{y} \equiv v_i$ represent a node that can potentially enter in contact with the domain surface σ. The node represented by $\boldsymbol{y}$ does not necessarily need to belong to a different discretization, in which case the foregoing discussion is also relevant in the context of self-contact.

For a given face $\varphi \subset \sigma$, the coordinate $\boldsymbol{x} \subset \varphi$ is a function of the natural (parent) domain coordinate $\boldsymbol{\xi}$, as displayed in Fig. 5. In other words, the coordinate $\boldsymbol{x} : \mathbb{R}^{d-1} \to \mathbb{R}^d$ is represented by $\boldsymbol{x}(\boldsymbol{\xi}) = \sum N_i(\boldsymbol{\xi}) v_i$, where N_i represents the Lagrangian shape function associated with node v_i. Note that due to the mapping from a lower-order manifold, the Jacobian matrix of the transformation $[\boldsymbol{J}]_{i\alpha} = \partial x_i / \partial \xi_\alpha$ is not square.

The closest point $\boldsymbol{x}^* := \boldsymbol{x}(\boldsymbol{\xi}^*) \subset \varphi$ to $\boldsymbol{y}$ is obtained after solving the following optimization problem

$$\boldsymbol{\xi}^* = \underset{\boldsymbol{\xi}}{\operatorname{argmin}}\, f(\boldsymbol{y}, \boldsymbol{x}(\boldsymbol{\xi})) = \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{x}(\boldsymbol{\xi})\|^2, \quad \text{such that} \quad c_i(\boldsymbol{\xi}) \geqslant 0, \ \forall i \in \mathscr{I}, \tag{16}$$

where $\mathscr{I}$ represents the index set of the inequality constraints associated with the face. For example, for the quadrilateral face shown in Fig. 5, the vector of constraints is $\boldsymbol{c}^{\mathsf{T}}(\boldsymbol{\xi}) = \{\xi_1 + 1, -\xi_1 + 1, \xi_2 + 1, -\xi_2 + 1\}$. It is worth noting that the solution to Eq. (16) may not be unique. Denoting the set of optimal coordinates $\mathscr{X} = \cup_i \boldsymbol{x}_i^*$, which results from solving Eq. 16 for every face considered $\varphi_i \in \sigma$, the global optimum $\boldsymbol{x}^\star$ can be found through

$$\boldsymbol{x}^\star = \underset{\boldsymbol{x}_i^* \in \mathscr{X}}{\operatorname{argmin}} \ \{\|\boldsymbol{y} - \boldsymbol{x}_i^*\|\}. \tag{17}$$

The inequality-constrained optimization problem stated by Eq. (16) can be solved by defining the Lagrangian functional [63]

$$\mathscr{L}(\boldsymbol{\xi}, \boldsymbol{\lambda}) = f(\boldsymbol{y}, \boldsymbol{x}(\boldsymbol{\xi})) + \boldsymbol{\lambda}^{\mathsf{T}} \boldsymbol{c}(\boldsymbol{\xi}), \tag{18}$$

where $\boldsymbol{\lambda}$ is a vector of Lagrange multipliers. The pair $(\boldsymbol{\xi}^*, \boldsymbol{\lambda}^*)$ is said to be a local solution to the optimization problem if it satisfies the Karush–Kuhn–Tucker (KKT) conditions

$$\begin{aligned}
&\nabla \mathscr{L}(\boldsymbol{\xi}^*, \boldsymbol{\lambda}^*) = \boldsymbol{0}, \\
&c_i(\boldsymbol{\xi}^*) \geqslant 0, \quad \forall i \in \mathscr{I}, \\
&\lambda_i^* \geqslant 0, \quad \forall i \in \mathscr{I}, \\
&\lambda_i^* c_i(\boldsymbol{\xi}^*) = 0, \quad \forall i \in \mathscr{I}.
\end{aligned}$$

For the current discussion, the minimization problem is solved by sequential quadratic programming (SQP) [63]. The starting guess of the parent domain coordinate ξ^0 is obtained as the closest point within the element to point $\boldsymbol{y}$, chosen between the center of the face and each of its vertices. A comprehensive study of the proposed formulation was published elsewhere [64]. That work presents a thorough comparison with other approaches used in computational geometry. Furthermore, SQP is also compared to other methodologies used in the field of mathematical optimization, including interior-point and active-set methods.

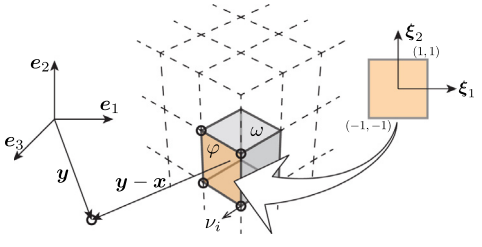

Fig. 5. Schematic of the problem of finding the distance from a point $\boldsymbol{y}$ to a face φ of the contact surface σ.

3.4. Detection cost

The total contact detection cost function of the proposed methodology, which extends that proposed by Klosowski [10] by adding the cost of the highest level of detection, is given by

$$\mathrm{T} = \underbrace{N_t C_t}_{\text{high}} + \underbrace{N_u C_u + N_v C_v}_{\text{middle}} + \underbrace{N_p C_p}_{\text{low}}, \tag{19}$$

where N_t, N_u, N_v and N_p refer to the number of trajectory-based tests, the number of hierarchy updates, the number of pairs of bounding volumes overlap tests, and the number of primitive overlap tests, respectively. C_t, C_u, C_v, and C_p refer to their corresponding costs.

In the beginning of the simulation, given that no bodies initially overlap, time is spent only at the highest level of detection, *i.e.*, the first term in Eq. (19). Approximate collision times are computed by solving Eq. (12) for every pair of objects. Note that the solution of this equation relies on velocities and accelerations of the hierarchies' roots, which are computed from updated positions obtained from the finite element approximation. Velocities and accelerations are assumed constant after their initial determination, a valid assumption for it is likely that these kinematic quantities change only after collision. As a result, the detection engine ceases operation completely saving on computational costs until time $t = \min(t^\star)$. The first term in Eq. (19) continues to contribute to the total cost after a contact event, as if the entire simulation is restarted after this point. It is worth nothing that it is not necessary to update entire bounding volume hierarchies in order to obtain approximate collision times. By exploiting the concept of temporal and geometric coherence [65], the time complexity of updating the hierarchies' roots could be obtained in constant time. The coherence property states that the position of the objects do not change significantly between time steps, an assumption valid for small time steps that allows us to search locally for bounding volume updates. Nevertheless, updating the positions of the roots has worst time complexity $\mathcal{O}(n)$.

The second and third terms in Eq. (19), which correspond to the middle level of detection, start contributing to the overall detection cost when hierarchies' roots overlap, *i.e.*, when $t \geqslant \min(t^\star)$. The first of these terms deals with the update of bounding volume hierarchies. At each time increment of the physics simulation, the locations of the primitives change, and therefore the entire bounding volume hierarchies need to be updated. After updating all the leaves of a hierarchy $\mathscr{T}$ corresponding to the finite element primitives, the complexity of updating the entire hierarchy is linear in the number of tree vertices $\mathscr{V}(\mathscr{T})$. A post-order traversal update of the tree vertices is chosen for this work. The cost of testing a pair of bounding volume hierarchies for overlap, C_v, increases as the tightness of the bounding volume that is used to represent the objects. Inversely, tightly-fitted volumes decrease the value of N_v. When a pair of bounding volume hierarchies overlaps, they are traversed and tested at deeper levels due to the better representation of the volumes in the collision zone. When leaves are found in both hierarchies, their bounding volumes are *refined* by increasing their degree of tightness. In this way, by using *volume adaptivity* the number of overlap tests N_v is decreased, and the costly contact detection test between finite element primitives is deferred to the latest possible time.

The low-level detection contributes to the overall cost when bounding volume hierarchies are traversed down to leaves and contact cannot be ruled out by using bounding volumes even after these are refined. However, the cost spent at the lowest level of the detection engine is kept to a minimum by the use of the other two levels.

Table 1 reports the results of numerical simulations, showing the exclusion test average times and their corresponding standard deviations for three types of bounding volumes. Each average time reported, which exclude any time spent in creating the bounding volume, is obtained over 1,000 numerical tests running sequentially on a 2.6 GHz Intel Core i7 processor. Bounding volumes were generated randomly with a seed chosen as to have roughly 50% of positive collision tests. The table also lists the average time and standard deviation of computing the distance from a 3D point to a triangular element using the optimization-based methodology outlined above. As apparent from the table, exclusion tests for bounding volumes require roughly the same time, and they are considerably faster than carrying out an optimization-based distance computation. The distance results were obtained by an equivalent C++ algorithm of the Fortran SQP implementation described by Kraft [66]. The algorithm uses BFGS updates for the approximation of the Hessian inverse matrix used in the search, requiring an average of 12 iterations (within the range $[5, 29]$) for the results reported in the table. The variability in the number of

Table 1
Numerical experiments of collision exclusion tests for different types of bounding volumes (up), and for the optimization-based distance computation (down). Results report the average time in nanoseconds of 1000 simulations, and their corresponding standard deviations.

	Average time [ns]	Standard deviation
Bounding volume		
BS	39	0.7
AABB	37	6
14-DOP	42	7
Method		
SQP	20457	7276

iterations has a direct impact on the computational time, as manifested by the standard deviation for the distance computation. These results support our motivation of defering the distance computation to the very last time.

4. Implementation

This section provides guidelines for the implementation of the contact detection framework in finite element codes. The methodology has been implemented in `Akantu`,[1] an open-source object-oriented C++ FE library developed at the École Polytechnique Fédérale de Lausanne (EPFL), Switzerland. The entire detection engine is contained in a **ContactManager** object, that is responsible for storing the state of the bounding volume hierarchies. Algorithm 1 provides pseudo-code for the main functions that determine contact at a given time t. Note that the algorithm provides procedural code and it conceals obvious details that would otherwise obscure the main points of the implementation. Therefore, it is expected that an efficient implementation of the algorithm be quite different.

The algorithm requires as input the bounding volume hierarchies at the current simulation time t, and a priority queue that is used to store the times when the detection engine has to operate, together with their corresponding working hierarchies. In other words, the queue data structure contains time-hierarchy-hierarchy triplets where the highest-priority object is the triplet with the minimum time. These are then parameters to the function DetectContact. Lines 2–3 in Algorithm 1 summarize the initialization stage. The first steps in the simulation are required to determine the kinematic quantities of the objects involved, *i.e.*, if they are stationary, if they have constant velocity, or if they accelerate. After this step, the priority queue is filled with approximate collision times and their corresponding volume hierarchies. From this point forward the detection engine only works if t is greater than the top item in the queue with time min $(t^\star)$. The hierarchies are then updated and the queue is examined for collisions.

Algorithm 1. ContactManager functions

```
    function DetectContact(t, {T_i}, q)                    ▷ time, hierarchies, priority queue
2:    if q = ∅ then                                        ▷ initialization stage
        return Setup (q, {T_i})
4:    while q ≠ ∅ do
        {t*, v_i, v_j} ← Top(q)                            ▷ get priority queue top item
6:      if t < t* then return                              ▷ contact detection disabled
        else
8:        {T_i, T_j} ← Update(v_i, v_j)                    ▷ update hierarchies
          Collision(q, v_i, v_j)                           ▷ check for collisions
10: function Collision (q, v_i, v_j)
    if Leaf(v_i) ∧ Leaf(v_j) then                          ▷ hierarchy leaves found
12:       if v_i ⋂ v_j ≠ ∅ then                            ▷ bounding volumes intersect
            DetailedCheck(v_i, v_j)
14:   else
        if (not Leaf(v_i) ∧ (Leaf(v_j) ∨ v_i ⩾ v_j)) then
16:       Traverse(q, Left(v_i), Right(v_i), v_j) then   ▷ traverse hierarchy rooted at v_i
        if (not Leaf(v_j) ∧ (Leaf(v_i) ∨ v_i ⩽ v_j)) then
18:       Traverse(q, Left(v_j), Right(v_j), v_i)        ▷ traverse hierarchy rooted at v_j
    function Traverse(q, v_i, v_j, v_k)
20:   q ← Push ( CollisionTime(v_i, v_k), v_i, v_k)      ▷ solve Eq. (12)
      q ← Push ( CollisionTime(v_j, v_k), v_j, v_k)      ▷ and add triplets to queue
```

For the top item in the priority queue, the function Collision is called. This function is responsible for modifying the queue with new triples that are obtained after traversing the hierarchies that are in potential contact. Whenever two leaves are found whose intersection volume is non-empty, a more refined detection check is performed by using better fitting volumes. When no predicates on bounding volumes exclude the possibility of contact, the actual finite element primitives are used to determine contact. Both the detection using refined bounding volumes and the thorough optimization-based check are implicitly contained within the DetailedCheck routine. If any of the two volumes is not a leaf, one or both hierarchies are traversed down to add new triplets to the queue by solving Eq. (12). The choice of which hierarchy to traverse depends on whether a bounding volume is a leaf of its hierarchy and on the actual volume values.

[1] http://lsms.epfl.ch/akantu.

Implementation remarks

- Bounding volume hierarchies are assumed not to overlap at time $t = 0$. An additional test could be added to abort the execution of the program if overlapping hierarchies are given as input.
- Once the initial kinematic quantities are determined in the initialization stage of the simulation, these are assumed to be constant henceforth.
- Even though the tree hierarchies could store polymorphic bounding volume objects, we have opted to store only bounding spheres and create on the fly higher-order volume representations as needed. Storing polymorphic objects and accessing the type of an object by using pointers increased considerably the execution time due to memory hierarchy effects.
- The priority queue actually stores pointers (or more precisely iterators) to the tree vertices that need to be checked for contact. The first triplet added will then contain pointers to the roots of the hierarchies involved. When roots overlap and the hierarchies are traversed down, then the priority queue also stores iterators to inner volumes.
- The time increment used in the simulation can be changed when collision is imminent, and therefore the exact collision time can be resolved using the proposed methodology. Section 5.1 presents an example showing this capability.

5. Examples

This section illustrates the use of the proposed detection engine with a set of examples.

5.1. Impact time prediction example

This example aims at determining the exact time of collision by using the three-level detection engine described in Section 3. Consider two spherical projectiles of radius $r = 1$ m, separated 1500 m from one another at time $t = 0$. An initial velocity $\dot{\boldsymbol{u}}^0 = 50\ \text{m/s}\left(\pm\boldsymbol{e}_1 + \sqrt{3}\boldsymbol{e}_3\right)$ is imposed to the spheres, after which they move towards one another with gravitational acceleration $\ddot{\boldsymbol{u}} = -g\boldsymbol{e}_3$. Explicit time integration is used with increment $\Delta t = 0.05$ s. In this example, the time increment is reduced when the objects are close enough as to detect the exact time of collision. From particle kinematics, it can be determined that the sphere centers overlap at time $t = 15$ s, so this time can be regarded as an upper bound on the actual collision time.

Fig. 6 displays snapshots of the two spheres at different times prior to the collision, and their corresponding hierarchies' bounding volumes with the highest priority. After the initial steps, the highest-level detection system predicts a first

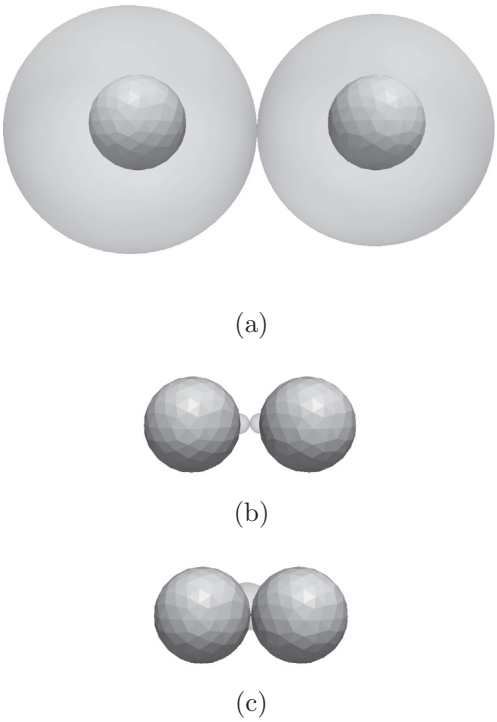

Fig. 6. Exact collision time prediction. Snapshots of the spherical projectiles at times (a) $t = 14.9503$ s, (b) $t = 14.9763$ s, and (c) $t = 14.98$ s. The figures also show the hierarchies' bounding volumes with the highest priority at the corresponding time.

collision between the hierarchies' roots at time $t_1^\star = 14.9503$ s. It is worth mentioning that even thought the objects involved in the collision are spherical, the resulting hierarchies have roots whose volumes are higher than those of the objects they bound. The detection system thus is idle up to time $t_1^\star$, where the predicted collision of the hierarchies' roots is shown in Fig. 6a. Fig. 6b shows one of the subsequent steps, where the bounding spheres with the highest priority correspond to the volumes of the elements that will come in contact. The detection system determines that the spheres collide at time $t = 14.98$ s, a step shown in Fig. 6c.

5.2. Motorcycle collision example

Consider the motorcycle mesh presented in Fig. 3a, moving with a constant velocity of 120 km/h. A wall is located approximately 100 m in front of the motorcycle and thus collision happens at $t = 3$ s. The motorcycle mesh is composed of 9,981 nodes and 17,955 finite elements, whereas the wall mesh contains 71 nodes and 144 elements. A time increment $\Delta t = 0.05$ s is used for the explicit time integration.

Fig. 7 shows several steps during the simulation. In each figure shown, a red sphere shows the contact location of the motorcycle bounding sphere that is determined by solving Eq. (12). At time $t = 0.1$ s, the detection engine has enough information to determine that the motorcycle travels at constant speed and that the wall is stationary. At this point, the first level of detection uses the hierarchies' top-level bounding spheres to determine the contact time $t_1^\star = 2.8$ s. The entire detection engine ceases operation until this time, illustrated in Fig. 7a. At this point, the bounding volume hierarchy of higher volume is traversed down and a new time $t_2^\star = 2.88$ s is determined. Subsequent traversal on the wall hierarchy encounters times $t_3^\star = 2.9$ s, $t_4^\star = 2.95$ s, and $t_5^\star = 3$ s.

Due to the comminuted mesh of the motorcycle, its corresponding bounding sphere hierarchy construction accounted for roughly 99% of the simulation's computational time up to the point of contact detection at $t = 3$ s. The remaining 1% is therefore spent during the contact detection phase. It is worth noting that the construction of the hierarchies is made once in the beginning of the simulation, and they are only updated later on during the contact detection phase. The detection engine operated only at times $t = 0.05, 0.1, 2.8, 2.9, 2.95, 3$ s. Considering only this stage, their corresponding share of computational times are 17.3%, 16.2%, 16.6%, 16.4%, 16.2%, and 17.3%, respectively. Note that the computational time is roughly uniform and thus it does not depend on how deep in the bounding volume hierarchies the detection engine works on. As a matter of fact, the computational time spent in this step is attributed almost entirely to updating the hierarchies. When the bounding sphere hierarchies are traversed down such that two leaves are encountered, the final detection mechanism enters in place for the most accurate determination of contact.

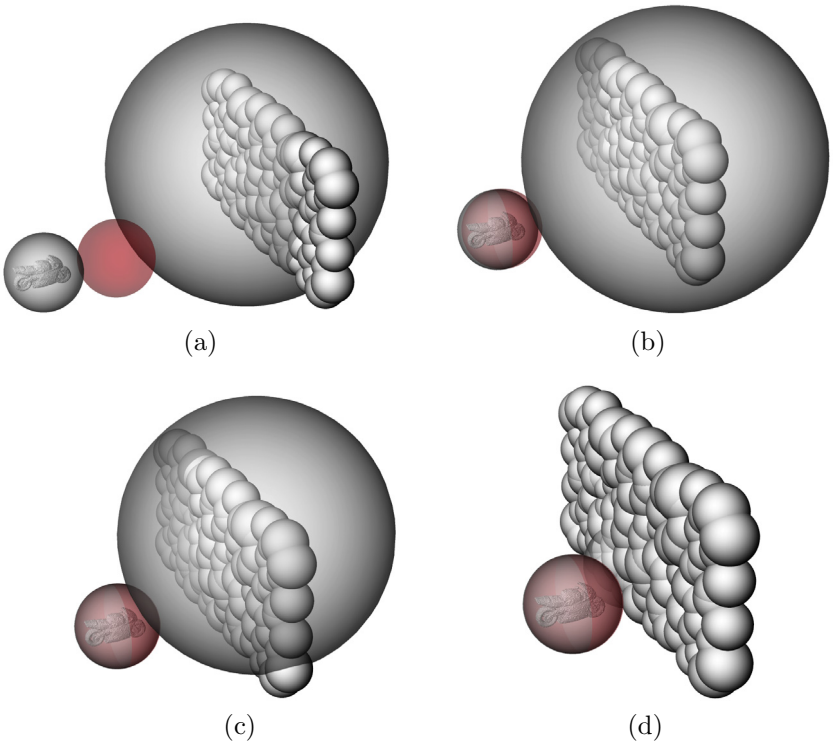

Fig. 7. Motorcycle-wall collision simulation snapshots at times (a) $t = 2.8$ s, (b) $t = 2.9$ s, (c) $t = 2.95$ s, and (d) $t = 3$ s. In each figure, a red sphere denotes the next collision time of the bounding volume hierarchies, as determined by solving Eq. (12).

6. Conclusions

In this manuscript, a three-level contact detection framework based on hierarchies of bounding volumes has been presented. At the highest level, the engine determines the collision times based on the trajectories of the bounding volumes of the solids participating in the simulation. The detection engine ceases operation until reaching the predicted contact times, thus saving on computational time. Once the top-level bounding volumes intersect, their hierarchies are traversed down to determine new collision times. When two hierarchies have been traversed down to the leaves, the finite element primitives involved in the contact are determined and a more refined level of detection enters in place to find out if the meshes have interpenetrated. An algorithm has been presented to ease the implementation of the framework in existing computational contact mechanics finite element codes. Comments on implementation details were also given by the authors, and the framework was tested with a couple of examples. It was demonstrated how the framework can be used to predict the exact time of collision.

Acknowledgments

The authors gratefully acknowledge Till Junge and Dr. Nicolas Richart from the Laboratoire de simulation en mécanique des solides (LSMS) at the EPFL for the valuable discussions. The authors also gratefully acknowledge Prof. Christophe Geuzaine from the University of Liège in Belgium and ir. Henk Krüs for providing the finite element meshes used in this manuscript. The research presented in this article is supported by the European Research Council (ERCstg UFO-240332).

References

[1] H. Hertz, ber die berührung fester elastischer körper, Journal für die reine und angewandte Mathematik 92 (92) (1881) 156–171. <http://www.reference-global.com/doi/abs/10.1515/crll.1882.92.156> .
[2] K. Schweizerhof, L. Nilsson, J. Hallquist, Crashworthiness analysis in the automotive industry, International Journal of Computer Applications in Technology, Special Issue on the Industrial Use of Finite-element Analysis 5 (2/3/4) (1992) 134–156.
[3] I.S. Doltsinis, Aspects of modelling and computation in the analysis of metal forming, Engineering Computations 7 (1) (1990) 2–20. <http://dx.doi.org/10.1108/eb02378> .
[4] S. Stupkiewicz, Z. Mróz, A model of third body abrasive friction and wear in hot metal forming, Wear 231 (1) (1999) 124–138. <http://dx.doi.org/10.1016/S0043-1648(99)00124-6> .
[5] D.J. Benson, J.O. Hallquist, A single surface contact algorithm for the post-buckling analysis of shell structures, Computer Methods in Applied Mechanics and Engineering 78 (2) (1990) 141–163. <http://doi.acm.org/10.1016/0045-7825(90)90098-7> .
[6] G. Camacho, M. Ortiz, Adaptive lagrangian modelling of ballistic penetration of metallic targets, Computer Methods in Applied Mechanics and Engineering 142 (3–4) (1997) 269–301. <http://doi.acm.org/10.1016/S0045-7825(96)01134-6> .
[7] B. Yang, T. Laursen, A contact searching algorithm including bounding volume trees applied to finite sliding mortar formulations, Computational Mechanics 41 (2008) 189–205, http://dx.doi.org/10.1007/s00466-006-0116-z.
[8] S.W. Attaway, B.A. Hendrickson, S.J. Plimpton, D.R. Gardner, C.T. Vaughan, K.H. Brown, M.W. Heinstein, A parallel contact detection algorithm for transient solid dynamics simulations using PRONTO3D, Computational Mechanics 22 (1998) 143–159, http://dx.doi.org/10.1007/s004660050348.
[9] P. Hubbard, Collision detection for interactive graphics applications, IEEE Transactions on Visualization and Computer Graphics 1 (3) (1995) 218–230, http://dx.doi.org/10.1109/2945.466717.
[10] J. Klosowski, M. Held, J. Mitchell, H. Sowizral, K. Zikan, Efficient collision detection using bounding volume hierarchies of k-dops, IEEE Transactions on Visualization and Computer Graphics 4 (1) (1998) 21–36, http://dx.doi.org/10.1109/2945.675649.
[11] M. Held, J.T. Klosowski, J.S. Mitchell, Evaluation of collision detection methods for virtual reality fly-throughs, in: Canadian Conference on Computational Geometry, 1995, pp. 205–210.
[12] J.W. Boyse, Interference detection among solids and surfaces, Communications of the ACM 22 (1) (1979) 3–9, http://dx.doi.org/10.1145/359046.359048.
[13] S. Cameron, Approximation hierarchies and s-bounds, in: Proceedings of the First ACM Symposium on Solid Modeling Foundations and CAD/CAM Applications, SMA '91, ACM, New York, NY, USA, 1991, pp. 129–137, http://dx.doi.org/10.1145/112515.112537.
[14] D.H. Eberly, 3D game engine design, in: A Practical Approach to Real-Time Computer Graphics, second ed., The Morgan Kaufmann Series in Interactive 3D Technology, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2006.
[15] R. Culley, K. Kempf, A collision detection algorithm based on velocity and distance bounds, in: Proceedings of 1986 IEEE International Conference on Robotics and Automation, vol. 3, 1986, pp. 1064–1069. http://dx.doi.org/10.1109/ROBOT.1986.1087575.
[16] E. Gilbert, D. Johnson, S. Keerthi, A fast procedure for computing the distance between complex objects in three-dimensional space, IEEE Journal of Robotics and Automation 4 (2) (1988) 193–203, http://dx.doi.org/10.1109/56.2083.
[17] S. Cameron, Collision detection by four-dimensional intersection testing, IEEE Transactions on Robotics and Automation 6 (3) (1990) 291–302, http://dx.doi.org/10.1109/70.56661.
[18] W.C. Thibault, B.F. Naylor, Set operations on polyhedra using binary space partitioning trees, in: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '87, ACM, New York, NY, USA, 1987, pp. 153–162, http://dx.doi.org/10.1145/37401.37421.
[19] B. Naylor, J. Amanatides, W. Thibault, Merging bsp trees yields polyhedral set operations, in: Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '90, ACM, New York, NY, USA, 1990, pp. 115–124, http://dx.doi.org/10.1145/97879.97892.
[20] J.L. Bentley, Multidimensional binary search trees used for associative searching, Communications of the ACM 18 (9) (1975) 509–517, http://dx.doi.org/10.1145/361002.361007.
[21] J.H. Friedman, J.L. Bentley, R.A. Finkel, An algorithm for finding best matches in logarithmic expected time, ACM Transactions on Mathematical Software 3 (3) (1977) 209–226, http://dx.doi.org/10.1145/355744.355745.
[22] I. Wald, V. Havran, On building fast kd-trees for ray tracing, and on doing that in O(N log N), in: IEEE Symposium on Interactive Ray Tracing 2006, pp. 61–69. http://dx.doi.org/10.1109/RT.2006.280216.
[23] H. Samet, The quadtree and related hierarchical data structures, ACM Computer Survey 16 (2) (1984) 187–260, http://dx.doi.org/10.1145/356924.356930.
[24] M. Moore, J. Wilhelms, Collision detection and response for computer animation, in: Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '88, ACM, New York, NY, USA, 1988, pp. 289–298, http://dx.doi.org/10.1145/54852.378528.
[25] H. Noborio, S. Fukuda, S. Arimoto, Fast interference check method using octree representation, Advanced Robotics 3 (3) (1988) 193–212, http://dx.doi.org/10.1163/156855389X00091. <http://www.ingentaconnect.com/content/vsp/arb/1988/00000003/00000003/art00003> .

[26] N. Beckmann, H.-P. Kriegel, R. Schneider, B. Seeger, The R*-tree: an efficient and robust access method for points and rectangles, in: Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data, SIGMOD '90, ACM, New York, NY, USA, 1990, pp. 322–331, http://dx.doi.org/10.1145/93597.98741.
[27] A. Garcia-Alonso, N. Serrano, J. Flaquer, Solving the collision detection problem, IEEE Computer Graphics and Applications 14 (3) (1994) 36–43, http://dx.doi.org/10.1109/38.279041.
[28] S. Gottschalk, M.C. Lin, D. Manocha, Obbtree: a hierarchical structure for rapid interference detection, in: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '96, ACM, New York, NY, USA, 1996, pp. 171–180, http://dx.doi.org/10.1145/237170.237244.
[29] S.A. Ehmann, M.C. Lin, Accurate and fast proximity queries between polyhedra using convex surface decomposition, Computer Graphics Forum 20 (3) (2001) 500–511, http://dx.doi.org/10.1111/1467-8659.00543.
[30] M.C. Lin, S. Gottschalk, Collision detection between geometric models: a survey, in: Proc. of IMA Conference on Mathematics of Surfaces, 1998, pp. 37–56.
[31] P. Jiménez, F. Thomas, C. Torras, 3d collision detection: a survey, Computers and Graphics 25 (2000) 269–285.
[32] M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M.-P. Cani, F. Faure, N. Magnenat-Thalmann, W. Strasser, P. Volino, Collision detection for deformable objects, Computer Graphics Forum 24 (1) (2005) 61–81, http://dx.doi.org/10.1111/j.1467-8659.2005.00829.x.
[33] C. Ericson, in: Real-Time Collision Detection, The Morgan Kaufmann Series in Interactive 3-D Technology, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.
[34] D. Baraff, Analytical methods for dynamic simulation of non-penetrating rigid bodies, in: Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '89, ACM, New York, NY, USA, 1989, pp. 223–232, http://dx.doi.org/10.1145/74333.74356.
[35] D. Baraff, Fast contact force computation for nonpenetrating rigid bodies, in: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '94, ACM, New York, NY, USA, 1994, pp. 23–34, http://dx.doi.org/10.1145/192161.192168.
[36] D.M. Kaufman, T. Edmunds, D.K. Pai, Fast frictional dynamics for rigid bodies, in: ACM SIGGRAPH 2005 Papers, SIGGRAPH '05, ACM, New York, NY, USA, 2005, pp. 946–956, http://dx.doi.org/10.1145/1186822.1073295.
[37] D. Baraff, A. Witkin, Dynamic simulation of non-penetrating flexible bodies, in: Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '92, ACM, New York, NY, USA, 1992, pp. 303–308, http://dx.doi.org/10.1145/133994.134084.
[38] M. Pauly, D.K. Pai, L.J. Guibas, Quasi-rigid objects in contact, in: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '04, Eurographics Association, Aire-la-Ville, Switzerland, 2004, pp. 109–119, http://dx.doi.org/10.1145/1028523.1028539.
[39] N. Galoppo, M.A. Otaduy, P. Mecklenburg, M. Gross, M.C. Lin, Fast simulation of deformable models in contact using dynamic deformation textures, in: Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '06, Eurographics Association, Aire-la-Ville, Switzerland, 2006, pp. 73–82. <http://dl.acm.org/citation.cfm?id=1218064.1218074>.
[40] D. Baraff, A. Witkin, Large steps in cloth simulation, in: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '98, ACM, New York, NY, USA, 1998, pp. 43–54, http://dx.doi.org/10.1145/280814.280821.
[41] P. Volino, N. Thalmann, Implementing fast cloth simulation with collision response, in: Proceedings Computer Graphics International, 2000, pp. 257–266. http://dx.doi.org/10.1109/CGI.2000.852341.
[42] R. Bridson, R. Fedkiw, J. Anderson, Robust treatment of collisions, contact and friction for cloth animation, in: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '02, ACM, New York, NY, USA, 2002, pp. 594–603, http://dx.doi.org/10.1145/566570.566623.
[43] J.M. Kaldor, D.L. James, S. Marschner, Efficient yarn-based cloth with adaptive contact linearization, in: ACM SIGGRAPH 2010 Papers, SIGGRAPH '10, ACM, New York, NY, USA, 2010, pp. 105:1–105:10, http://dx.doi.org/10.1145/1833349.1778842.
[44] Z. Chen, R. Feng, H. Wang, Modeling friction and air effects between cloth and deformable bodies, ACM Transactions on Graphics 32 (4) (2013). 88(1–88), pp. 8.
[45] J.-P. Heo, J.-K. Seong, D. Kim, M.A. Otaduy, J.-M. Hong, M. Tang, S.-E. Yoon, Fastcd: fracturing-aware stable collision detection, in: Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '10, Eurographics Association, Aire-la-Ville, Switzerland, 2010, pp. 149–158. <http://dl.acm.org/citation.cfm?id=1921427.1921450>.
[46] L. Glondu, S.C. Schvartzman, M. Marchal, G. Dumont, M.A. Otaduy, Efficient collision detection for brittle fracture, in: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '12, Eurographics Association, Aire-la-Ville, Switzerland, 2012, pp. 285–294. <http://dl.acm.org/citation.cfm?id=2422356.2422397>.
[47] F. Bertails-Descoubes, F. Cadoux, G. Daviet, V. Acary, A nonsmooth newton solver for capturing exact coulomb friction in fiber assemblies, ACM Transactions on Graphics 30 (1) (2011). 6(1–6), pp. 14.
[48] G. Daviet, F. Bertails-Descoubes, L. Boissieux, A hybrid iterative solver for robustly capturing coulomb friction in hair dynamics, in: Proceedings of the 2011 SIGGRAPH Asia Conference, SA '11, ACM, New York, NY, USA, 2011, pp. 139:1–139:12, http://dx.doi.org/10.1145/2024156.2024173.
[49] J. Gascón, J.S. Zurdo, M.A. Otaduy, Constraint-based simulation of adhesive contact, in: Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '10, Eurographics Association, Aire-la-Ville, Switzerland, 2010, pp. 39–44. <http://dl.acm.org/citation.cfm?id=1921427.1921434>.
[50] D.M. Kaufman, S. Sueda, D.L. James, D.K. Pai, Staggered projections for frictional contact in multibody systems, in: ACM SIGGRAPH Asia Papers, SIGGRAPH Asia '08, ACM., ACM, New York, NY, USA, 2008, pp. 164:1–164:11, http://dx.doi.org/10.1145/1457515.1409117.
[51] D. Harmon, E. Vouga, B. Smith, R. Tamstorf, E. Grinspun, Asynchronous contact mechanics, in: ACM SIGGRAPH 2009 Papers, SIGGRAPH '09, ACM, New York, NY, USA, 2009, pp. 87:1–87:12, http://dx.doi.org/10.1145/1576246.1531393.
[52] M.A. Otaduy, M.C. Lin, Clods: dual hierarchies for multiresolution collision detection, in: Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, SGP '03, Eurographics Association, Aire-la-Ville, Switzerland, 2003, pp. 94–101. <http://dl.acm.org/citation.cfm?id=882370.882382>.
[53] T. Larsson, T. Akenine-Möller, Bounding volume hierarchies of slab cut balls, Computer Graphics Forum 28 (8) (2009) 2379–2395, http://dx.doi.org/10.1111/j.1467-8659.2009.01548.x.
[54] M.W. Heinstein, F.J. Mello, S.W. Attaway, T.A. Laursen, Contact–impact modeling in explicit transient dynamics, Computer Methods in Applied Mechanics and Engineering 187 (3–4) (2000) 621–640. <http://doi.acm.org/10.1016/S0045-7825(99)00342-4> .
[55] P. Wriggers, Computational Contact Mechanics, second ed., Springer, 2006.
[56] T. Hughes, The finite element method: linear static and dynamic finite element analysis, Dover Civil and Mechanical Engineering Series, Dover Publications, 2000.
[57] O.C. Zienkiewicz, R.L. Taylor, J.Z. Zhu, The Finite Element Method: Its Basis and Fundamentals, Butterworth-Heinemann, 2005.
[58] J. Ritter, An efficient bounding sphere, in: A.S. Glassner (Ed.), Graphics Gems, Academic Press Professional Inc., San Diego, CA, USA, 1990, pp. 301–303.
[59] T. Larsson, Fast and tight fitting bounding spheres, in: Proceedings of The Annual SIGRAD Conference, Linköping University Electronic Press, 2008, pp. 27–30.
[60] J.T. Schwartz, Fin ding the minimum distance between two convex polygons, Information Processing Letters 13 (4/5) (1981) 168–170, http://dx.doi.org/10.1016/0020-0190(81)90051-X. <http://www.sciencedirect.com/science/article/pii/002001908190051X> .
[61] S.M. Omohundro, Five balltree construction algorithms, Technical Report TR-89-063, International Computer Science Institute, University of California at Berkeley, 1989. <http://ftp.icsi.berkeley.edu/ftp/pub/techreports/1989/tr-89-063.pdf>
[62] G.v.d. Bergen, Efficient collision detection of complex deformable models using aabb trees, Journal of Graphics Tools 2 (4) (1997) 1–13, http://dx.doi.org/10.1080/10867651.1997.10487480.

[63] J. Nocedal, S.J. Wright, Numerical Optimization, Springer, 2000.
[64] A.M. Aragón, V.A. Yastrebov, J.-F. Molinari, A constrained-optimization methodology for the detection phase in contact mechanics simulations, International Journal for Numerical Methods in Engineering 96 (5) (2013) 323–338, http://dx.doi.org/10.1002/nme.4561.
[65] J.D. Cohen, M.C. Lin, D. Manocha, M. Ponamgi, I-collide: an interactive and exact collision detection system for large-scale environments, in: Proceedings of the 1995 Symposium on Interactive 3D Graphics, I3D '95, ACM, New York, NY, USA, 1995, p. 189, http://dx.doi.org/10.1145/199404.199437.
[66] D. Kraft, Algorithm 733: tomp–fortran modules for optimal control calculations, ACM Transactions on Mathematical Software 20 (3) (1994) 262–281, http://dx.doi.org/10.1145/192115.192124.