COMPUTER-AIDED DESIGN & APPLICATIONS, 2018
VOL. 15, NO. 4, 585–600
https://doi.org/10.1080/16864360.2017.1419646

Fast computation of local minimal distances between CAD models for dynamics simulation

Sébastien Crozet [a], Jean-Claude Léon [b] and Xavier Merlhiot [a]

[a] CEA, LIST, Interactive Simulation Laboratory, Gif-sur-Yvette, F-91191; [b] INRIA, Grenoble University

ABSTRACT

A complete framework for the fast computation of multiple local minimal distances between complex CAD models represented by their smooth boundaries without any tessellation is presented. It is used for collision detection as part of a rigid body dynamics simulation engine for interactive virtual manipulations and telerobotics applications. Computing contact points on the smooth model representations naturally prevents numerical artifacts introduced by discontinuous normals of tessellated models. Typical use-cases are the simulation of rolling and sliding motions with small mechanical clearance where shocks introduced by geometric approximations may induce significant and sudden changes in the objects trajectories. Our method is based on the observation that industrial CAD models are mostly composed of simple curves and surfaces (segments, circles, cylinders, planes, cones, etc.), allowing several efficient surface-type-dependent special-cases. We improve existing off-line geometric analysis for grouping similar features, and propose a new, tighter, bounding volume for normal cones of faces, edges, and vertices. Both contributions bring significant performance gains for interactive applications as they prevent many useless or redundant distance computations. A comparison with an existing similar distance-based tessellation-based collision detection method is presented to prove the advantages of our method regarding the balance between accuracy and performances.

KEYWORDS

Collision Detection; B-Rep; Smooth Constraints; Local Minimal Distances

1. Introduction

With the development of advanced robotic systems and complex teleoperation tasks, the need to perform simulations before operating on physical systems becomes of increasing interest for feasibility tests, training of the human operators, motion planning, etc. Such simulations are to be performed with great accuracy of physical phenomena if, e.g., the operator is expected to face the same ones in the real world and in the virtual scene. Typical tasks to be simulated interactively are the virtual assembly or disassembly of a product composed of multiple rigid body parts. These simulations may involve monitoring a robotic arm to perform maintenance operations on a remote location. Other needs for real time mechanical simulations appear during the product design phase with usage scenarios where the user has a key impact to manipulate the product and evaluate its adequacy. In such scenarios, the manipulated objects are commonly subject to rolling and sliding motions that must be rendered accurately. At the core of these simulations is a physical simulation engine that computes the trajectory of objects subjected to user interactions, kinematic constraints, and intermittent contacts, acting also as nonlinear constraints. Collision detection (CD) is a significant and time-consuming part of such an engine when it computes contact points (and contact normals) between interacting rigid bodies. The accuracy and smoothness of such contact information is of primary importance to produce a realistic behavior of the simulated objects.

However, the quality of the computed contact information strongly depends on the geometric representation of the virtual scene and the parts directly involved in the mechanical simulation. Currently, two of the major approaches are volumetric representations [17, 29] where the interior of each object is represented explicitly, and boundary representations [27] where only the boundary of each object is explicitly represented, e.g., with a surface mesh, or with smooth surfaces.

Volumetric representations are typically preferred for low-accuracy simulations requiring high refresh rates while boundary representations yield more accurate results but at a higher computational cost. Boundary

CONTACT Sébastien Crozet sebastien.crozet@cea.fr; Jean-Claude Léon jean-claude.leon@grenoble-inp.fr; Xavier Merlhiot xavier.merlhiot@cea.fr

representations themselves can be split into two families: tessellation-based and smooth surfaces-based. While the latter corresponds to whatever smooth output of a CAD modeler yields in term of piecewise polynomial surfaces, the former is an approximation of smooth surfaces obtained with triangles, segments, and points. Tessellation-based algorithms are by far the most popular choice and have been subjected to a significant amount of research [15]. However, because of the inaccuracies of the contact points locations and the discontinuity (with regard to the objects motion parameters) of the contact normals induced by this discrete approximation of smooth shapes, tessellated objects are not always applicable in practice to interactive simulations requiring a high level of accuracy for rolling and sliding motions performed with small mechanical clearance between objects. Therefore, using smooth surfaces is of major interest since they *natively* get rid of the previously mentioned discontinuity problems. However, existing robust and accurate methods readily operating on smooth surface representations still suffer from performance limitations. In order to speed up the smooth surface processing, [6] have recently introduced new surface features and bounding volume hierarchy that can produce real-time mechanical simulations. Here, the contribution is twofold: the introduction of a generic data structure structuring the surface features previously validated and a tight approximation of tangent cone polars that further speed up the local minimal distances computation.

This paper describes a fast algorithm for computing closest points between two complete industrial CAD models (not only individual surfaces) described through their smooth Boundary Representation (B-Rep) issued by CAD modelers. No tessellation process is involved. The proposed approach takes advantage of key features of mechanical components because they are often modeled with surfaces describing functional contacts with canonical surfaces (cylinder, sphere, cone, plane, torus) while contacts over free-form surfaces like B-Splines are very rare and punctual. A comparison is performed with a similar distance-based method that relies on polyhedral approximations of the original B-Rep model. In the context of telerobotics or real-time mechanical simulations, our method features both better performances and better accuracy for an insertion task with small mechanical clearance.

2. Related works

Within the context of the simulation of non-smooth mechanical systems, local minimal distances and penetration depths are the most common concepts to define contact points and their associated normals. In our applications, the contact information generated have to comply with the requirements of a non-smooth contact dynamics solver based on a time-stepping scheme [1]. As shown in Fig. 1, these information are used to formulate contact constraints that should be regular, i.e., with normal at contact points having continuous first order derivatives with respect to the contact location, in order to ensure the subsequent constraints resolution algorithms behave accurately and efficiently. From the user point of view, failing to have contact normals continuous with respect to the motion parameters may affect the simulation of rolling and sliding motions significantly by generating unrealistic interactions. This sometimes changes the object's behavior in sudden and unexpected ways with spurious impacts and jamming.

There are three main approaches to overcome this limitation. At first comes an increased accuracy of the discretization. This reduces the adverse effects of impacts on added edges and vertices at the cost of a higher number of contact points. Such an increase of the number of contacts raises significant performance issues because the number of constraints passed on to the dynamics solver increases as well. Therefore, this method is not applicable to simulations where a very high level of accuracy is expected, e.g., sliding and rolling motions.

A second alternative relies on other measures like the *growth distance* [8] or the *continuous penetration depths* [30]. While both are continuous wrt. motion parameters even if the objects have discrete representations, the first one is limited to convex objects only. The second one is made possible by approximating smoothly the *contact space*, i.e., the set of positions for which two objects just touch (without penetration), and using it to detect contacts and compute the penetration depth at run-time. However, those penetration depth approximations of the contact space are applicable to translations only and don't handle rotational motions yet.

The third category of approaches computes distances or penetration depths using the smooth curves and surfaces of the B-Rep models. While such approaches have all the benefits in terms of accuracy (continuity of normals, accurate and smooth contact kinematics [21]) and output a low number of contact points, most existing methods that are efficient enough to handle complete B-Rep models interactively come with restrictions or approximations that may significantly affect their results. [16] and [32] restrict the models to sets of convex components only so that they may track the unique contact point between two convex parts in real-time. [5] locates approximately contact areas of concave components by computing intersections between their dilated polyhedral approximations. The central points of those intersections

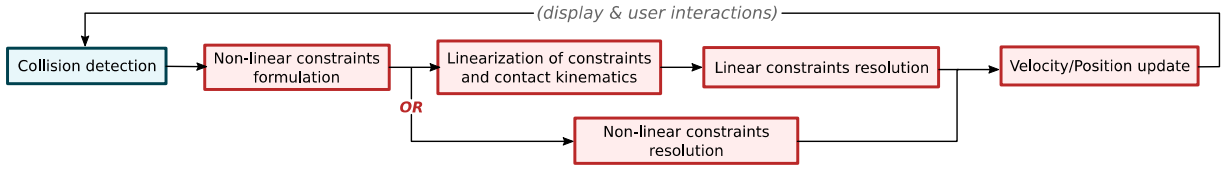

Figure 1. Flowchart of a simulator based on a time-stepping integration scheme for the simulation of a non-smooth mechanical system. Typical real-time physics engines use linearized constraints and contact kinematics. Non-linear solvers exist [1] and achieve better realism but at a higher computational cost.

are used to initialize an iterative minimization method extracting one contact point per identified contact area. Being limited to one contact per intersection of the polyhedral approximations, many of them may be missed over non-convex areas if some non-convex faces of the objects are close to one-another, or if the chosen dilation is too large. [24] describes a method for collision detection of a B-Spline-shaped tool with a CAD model composed of deformable B-Spline surfaces. After a coarse proximity filtering based on convex hulls, it searches for intersecting surface areas using sampled areas of each solid. Bounding spheres are computed from those samples to cover these areas and refine the search. Once the search areas are sufficiently small, a local triangulation is used to compute the intersecting areas. Because bounding spheres are computed from samples over the surfaces, small surface patches with high curvatures may not be detected by this algorithm. Edges and vertices are not processed either.

Using our previous work [6] as a basis, we compute Local Minimal Distances (LMD) between the smooth B-Reps obtained from industrial CAD models. We focused on improving performances in order to perform interactive rigid body dynamics simulations in the context of, e.g., virtual teleoperation and robotics. We bring contributions to both the pre-computation phase and the run-time phase of the collision detection process. In section 5, we extend the notions of *maximal faces* and *maximal edges* introduced by [4, 14] to include faces or edges with non-adjacent parts and use them to reduce the number of individual entities and parametric domains to be handled during the run-time phase. While we first introduced this concept under the name of *supermaximal features* in [6], this paper improves the underlying data structures with the notion *supermaximal domains*. In section 6, we design a new, tighter bounding volume for normals. Given the usual characterization of LMD (see section 3.2), bounding normals of subsets of the B-Rep features is invaluable to accelerate the search for those that may contain a contact point. Such bounding volume already exist in the form of revolution cones and have been successfully applied to distance and penetration depth computation [11, 28]. However, even if these cones are suited for distance computation between free-form surfaces, they can be tightened for edges, and patches of canonical faces which are the areas where sliding and rolling motions can take place. The run-time phase of our method is identical to the one we presented in [6] except for modifications to take into account our new supermaximal domains and polyhedral normal cones. As most existing methods do for real-time or interactive simulations, it is based on spatial bounding volumes to search for potential parts of the B-Rep model that may touch [9, 13, 22, 31] another object. Those spatial volumes, together with our polyhedral normal cones, are arranged into Bounding Volume Trees (BVT). Note that our extension to supermaximal domains allows us to filter out not only individual pairs of BVT nodes based on their bounding volumes, but also pairs of features for which computing new contact points would be redundant. Moreover, some new and very simple criteria on the objects types and orientations provide extra filtering particularly useful when bounding volumes fail to discriminate. Finally, we show in section 8 an example of interactive simulation for teleorobotics compared to an existing similar distance-based, tessellation-based, method [19].

3. Solids models and characterization of contact points

Industrial solid models have complex shapes mostly built from rather simple primitives, i.e., extrusions and revolutions generated using 2D contours made up from line segments and arcs of circles. Often, surfaces forming the geometry of faces of a solid are subjected to functional contacts. Surfaces bounding these primitives, even if they happen to belong to functional contacts, usually end up being canonical faces [25], i.e., planes, cylinders, cones, spheres or tori, possibly with internal contours. Thus, our work is optimized for models with many canonical faces and few faces with a more general geometry (revolution surface, Bézier, etc.)

Contact normals and contact points generated by our method must comply with the requirements of a non-smooth contact dynamics solver based on a time-stepping scheme as described in [1]. Such a scheme relies

on the definition of differentiable constraints that will be enforced by the solver to prevent penetration, enable friction, and generate impacts. Contact constraints are typically modeled by gap functions g (see Fig. 3) between two solids, namely $\mathcal{A}$ and $\mathcal{B}$ seen as sets of points that include the solids' interiors. Their boundaries represented by the B-Rep models are noted$\partial\mathcal{A}$, and $\partial\mathcal{B}$. Natural candidates for such gap functions are the local minima (seen as functions of the motion parameters $q_\mathcal{A}$ and $q_\mathcal{B}$ of $\mathcal{A}$ and $\mathcal{B}$) of the Euclidean distance between $\mathcal{A}$ and $\mathcal{B}$. The corresponding distance values are called Local Minimal Distances (LMD) [19] and the points generating them are *contact points*. Note that real-time physics engines (see Fig. 1) are often based on a linear constraints solver which introduces numerical errors. Moreover, even non-linear iterative solvers may generate intermediate results that violate some constraints before reaching convergence. These are the reasons why our method must be able to output negative gap values nonetheless. Therefore, each solid is *virtually* dilated using an envelope of spheres of radius $\varepsilon \in \mathbb{R}^{+*}$ (see Fig. 2c). *Virtually* means that the boundaries of the objects are not actually modified but 2ε is systematically subtracted from any distance computation result. Note that this completes our definition of gap function that now tolerates negative values down to -2ε.

Our method is based on the characterization of contact points between two solids $\mathcal{A}$ and $\mathcal{B}$ as stated by [6, 19]. Indeed, in addition to Eqn. (3.2) expressing the LMD between $\mathcal{A}$ and $\mathcal{B}$, the direction of the LMD appears also as a key feature that can be characterized using the concepts of tangents, tangent cones, and cone polars, as defined in the field of convex analysis [3]. We recall some definitions from [3]. Let $\mathcal{S}$ designates points of either $\mathcal{A}$ or $\mathcal{B}$. Given a point $\mathbf{x}_\mathcal{S} \in \mathcal{S}$ (which can be either on the interior of the boundary of $\mathcal{S}$), its *tangents* are the vectors $\mathbf{y} \in \mathbb{R}^3$ such that there exists a sequence of vectors $\{\mathbf{x}^k\} \subset \mathcal{S}, \mathbf{x}^k \neq \mathbf{x}_\mathcal{S}$, and:

$$\mathbf{x}^k \to \mathbf{x}_\mathcal{S}, \frac{\mathbf{x}^k - \mathbf{x}_\mathcal{S}}{||\mathbf{x}^k - \mathbf{x}_\mathcal{S}||} \to \frac{\mathbf{y}}{||\mathbf{y}||} \tag{3.3}$$

The set of all tangents at $\mathbf{x}_\mathcal{S}$ form a cone (in the sense of convex analysis) called *tangent cone* $T_\mathcal{S}(\mathbf{x}_\mathcal{S})$. The *tangent cone polar*:

$$T_\mathcal{S}(\mathbf{x}_\mathcal{S})^* = \{\mathbf{d} \in \mathbb{R}^3 | \forall \mathbf{v} \in T_\mathcal{S}(\mathbf{x}_\mathcal{S}), \langle \mathbf{d}, \mathbf{v} \rangle \leq 0\} \tag{3.4}$$

is the set of vectors opposite to all the tangents at $\mathbf{x}_\mathcal{S}$. Some tangent cones and their polars are depicted in Fig. 3 to illustrate common configurations. For sake of simplicity, a 2D domain is used rather than a 3D one. The tangent cone in Fig. 3(c) is larger than a half-space, therefore its polar degenerates to the singleton $\{\mathbf{0}\}$.

We now have the tools to characterize *critical points* of the squared distance function between $\mathcal{A}$ and $\mathcal{B}$. Indeed,

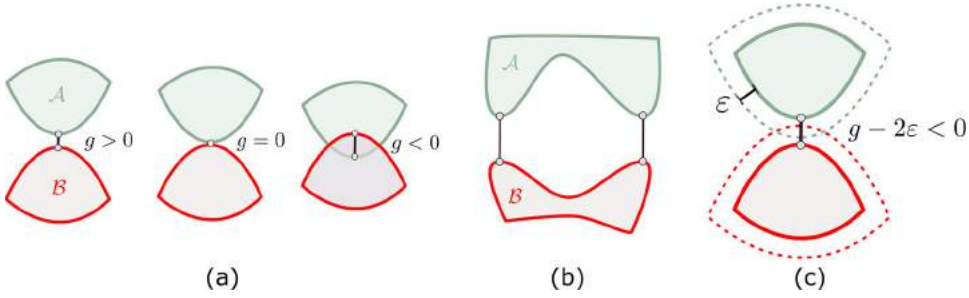

Figure 2. (a) Gap g between two solids $\mathcal{A}$ and $\mathcal{B}$ that are, from left to right, separated, touching, or penetrating. (b) Multiple gaps can be defined between non-convex solids. (c) The virtually dilated solids (dashed lines) in a configuration leading to a negative value of the modified gap function.

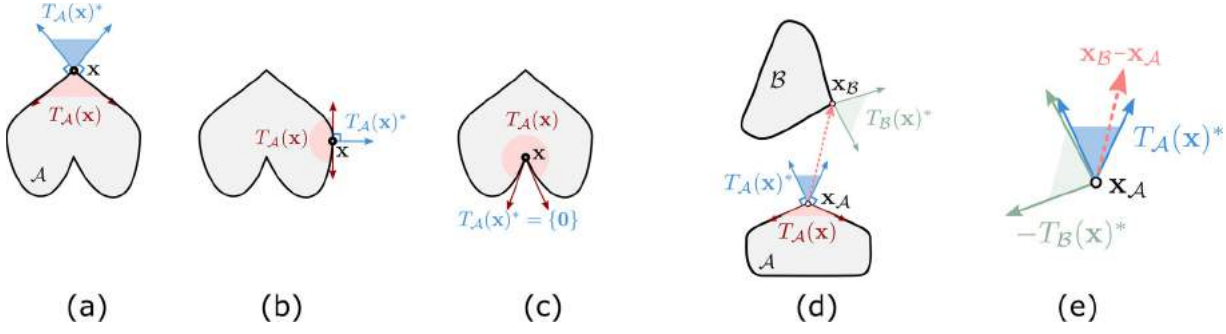

Figure 3. A tangent cone $T_\mathcal{A}(\mathbf{x})$ and its polar $T^*_\mathcal{A}(\mathbf{x})$ at a point $\mathbf{x}$ which is: (a) G^0 only and convex; (b) regular on an edge; (c) G^0 only and concave. (d) Two tangent cones polar where $\mathbf{x}_\mathcal{A}$ and $\mathbf{x}_\mathcal{B}$ are not critical points and (e) the corresponding illustration of Eqn. (3.5).

two points $\mathbf{x}_\mathcal{A} \in \mathcal{A}$ and $\mathbf{x}_\mathcal{B} \in \mathcal{B}$ are critical points iff. the following condition holds [6, 19]:

$$\mathbf{x}_\mathcal{B} - \mathbf{x}_\mathcal{A} \in T_\mathcal{A}(\mathbf{x}_\mathcal{A})^* \cap -T_\mathcal{B}(\mathbf{x}_\mathcal{B})^* \tag{3.5}$$

The goal of our method being the computation of contact points (which are also critical points), we use Eqn. (3.5) to distinguish parts of features that may or may not contain some closest points in section 6. Note that this definition allows non-isolated critical points (i.e. conformal contacts configurations) which are handled by the contact-area sampling method from [6].

As a conclusion, B-Rep solid boundaries and time-stepping schemes with ϵ-dilation are consistent wrt. each other and robustly define the spatial domain of existence of contact points between $\mathcal{A}$ and $\mathcal{B}$, i.e., $\partial\mathcal{A}$ and $\partial\mathcal{B}$. The same properties do not hold for contact computation based on penetration distances.

4. Overview of the proposed method

Our method is split into two stages depicted on Fig. 4. Firstly, the preprocessing stage (executed only once) creates a tree-based acceleration structure (see section 7) on top of the original B-Rep structure (which is left unchanged) in order to accelerate future LMD computations at simulation run-time. Then, the run-time stage (executed at each time-step) uses both the original B-Rep and the precomputed acceleration structure to find features that may contain contact points and actually computes them.

The different steps performed during the preprocessing stage (supermaximal features grouping and BVT construction) are similar to [6] except for three contributions brought by this paper:

- The data structure produced by supermaximal feature grouping has been improved with the introduction of the concept of *supermaximal domain*, which results into a unified representation of the supermaximal feature parts' parametric domains. This is detailed in section 5.
- The introduction, in section 6, of a new type of normal bounding volume that is much tighter than the revolution cones used by [6]. Consequently, the run-time stage performs much more efficiently, as shown by section 8.
- The notion of compatibility between different types of boundary features, introduced in section 7, showing that some combinations never hold an LMD because of their orientations.

Note that all NURBS curves and surfaces are decomposed into their Bézier components, which can be evaluated more efficiently and used to derive their bounding volumes more easily. This decomposition is always possible by repeated knot-insertion [23].

Finally, the major difference between the run-time phase of our method and of [6] resides in the use of the intersection test presented in section 6.2 that performs on polyhedral normal cones.

5. Supermaximal features and supermaximal domains

First of all, let us recall the definitions of supermaximal features introduced in [6] and state the grouping criteria. Then, follows the improvement of the data structures used to represent supermaximal faces using the concept of *supermaximal domain*.

Because of topological restrictions prescribed by their CW-complex structure, B-Rep models often contain disjoint areas of the same surface or curve that appear as several independent geometrical entities (see faces $F^7_\mathcal{A}$ and $F^8_\mathcal{A}$ in Fig. 5). These areas coincide with faces or edges of the B-Rep data structure. On the one hand, we say that the i-th set ${}^sF^i_\mathcal{A}$ of possibly disjoint areas of canonical surfaces, $\{F^l_\mathcal{A}, F^m_\mathcal{A}, \ldots, F^n_\mathcal{A}\}$, sharing the same orientation such that all entities of ${}^sF^i_\mathcal{A}$ share the same intrinsic geometric properties, forms a *supermaximal face*. In other words, these areas are embedded into the same untrimmed surface. On the other hand, the j-th set ${}^sE^j_\mathcal{A}$ is a *supermaximal edge* and is composed of the edges (with possible disjoint curves) $\{E^p_\mathcal{A}, E^q_\mathcal{A}, \ldots, E^r_\mathcal{A}\}$ at the intersection of two supermaximal faces which are part of the same connected component of the intersection between the underlying untrimmed surfaces. In other words, these curves are embedded into a single

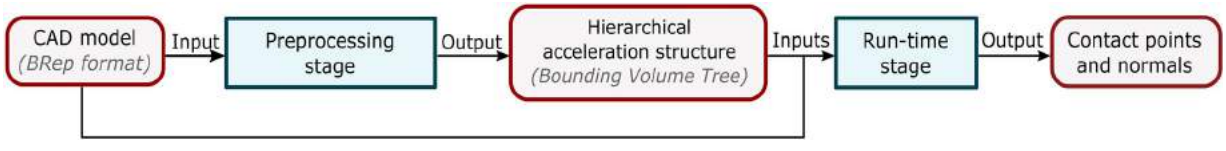

Figure 4. Different inputs and outputs at both stages of the proposed method. Data structures are in rounded red boxes while algorithmic treatments are in blue rectangles.

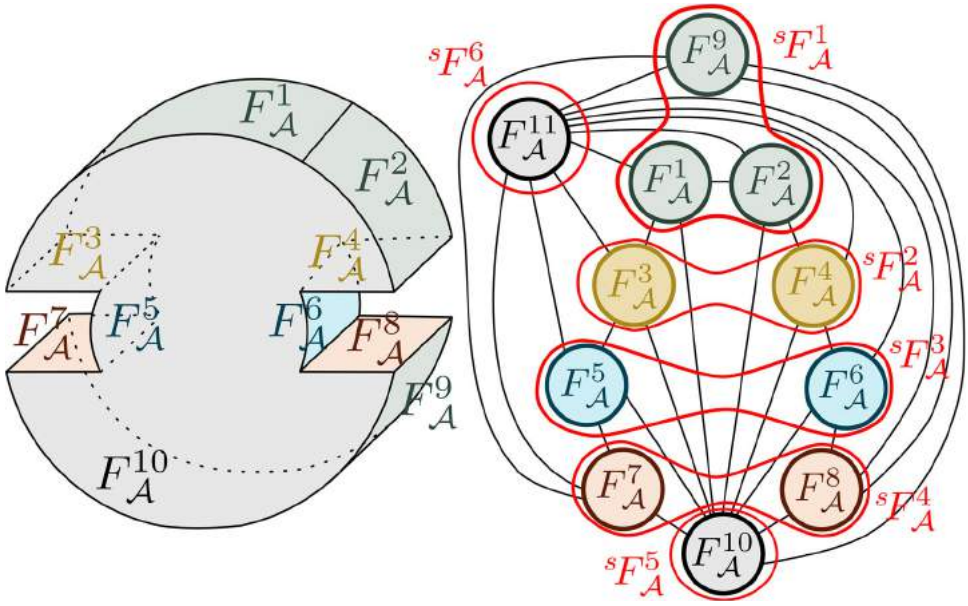

Figure 5. A solid and its B-Rep face-edge adjacency graph structure. F_A^{11} designates the hidden planar face parallel to F_A^{10}. All the supermaximal faces are circled on the graph.

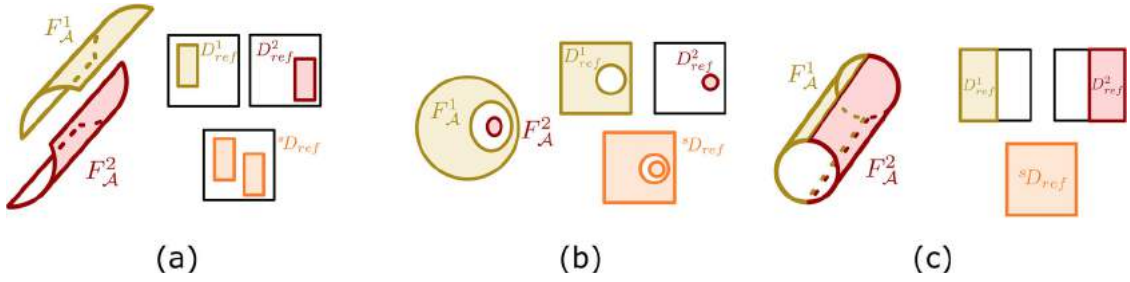

Figure 6. Three configurations for merging two parametric domains D^1_{ref} and D^2_{ref}. The resulting supermaximal domain ${}^sD_{ref}$ may have (a) two non-nested loops delimiting disjoint faces; (b) nested loops which alternately delimit boundary areas and holes; (c) only one loop: the two coinciding vertical trimming curves are no longer needed and have been removed.

(continuous) intersection curve between two untrimmed surfaces. Note that *supermaximal faces* and *supermaximal edges* are somewhat similar to the *maximal edges* and *maximal surfaces* introduced by [4, 14].

An example is given in Fig. 5. One of the supermaximal faces of $\partial\mathcal{A}$ is ${}^sF^1_{\mathcal{A}} = \{F^1_{\mathcal{A}}, F^2_{\mathcal{A}}, F^9_{\mathcal{A}}\}$ because these faces are areas of the same unbounded cylinder. Moreover, ${}^sE^1_{\mathcal{A}} = \{E^{5,10}_{\mathcal{A}}, E^{6,10}_{\mathcal{A}}\}$ (with $E^{i,j}_{\mathcal{A}}$ denoting the edges at the intersection of $F^i_{\mathcal{A}}$ and $F^j_{\mathcal{A}}$) is a supermaximal edge (a circle).

Finally, note that this identification of supermaximal edges and faces allows us to apply run-time optimizations as in [6]. Next, we define the notion supermaximal domain that simplifies the data structure of the supermaximal faces domain.

B-Rep faces domains $\mathcal{D}_i \in F^i_{\mathcal{A}}$ are subsets of $\mathbb{R}^2$ and bounded by trimming curves that form a set of simple loops. Those loops are such that the parametrization functions $\Phi_i : \mathcal{D}_i \to \mathbb{R}^3$ are global homeomorphisms, even for periodic surfaces. One of those loops is identified as the *exterior loop*, which delimits the contour of $F^i_{\mathcal{A}}$. There cannot be several exterior loops since $F^i_{\mathcal{A}}$ contains only one connected component. Let $\mathcal{D}_i$ be the parametrization domain of the i^{th} surface of the face $F^i_{\mathcal{A}}$. A supermaximal face ${}^sF^j_{\mathcal{A}}$ is defined in [6] as the set $\{(\Phi_{ref}, \mathcal{D}^i_{ref}, \Xi_i) | i \in I\}$, where Φ_{ref} is a fixed reference parametrization over the largest (untrimmed) domain noted $\mathcal{D}_{ref}$. The domains $\mathcal{D}^i_{ref} \subset \mathcal{D}_{ref}$, are such that $\forall\alpha \in \mathbb{R}^2, \alpha \in \mathcal{D}^i_{ref} \Rightarrow \Phi_i^{-1}(\Phi_{ref}(\alpha)) \in \mathcal{D}_i$. Ξ_i are sets of modeler-dependent per-feature attributes and metadata. We improve this definition by observing that all $\mathcal{D}^i_{ref}$ delimit subsets of $\mathcal{D}_{ref}$, therefore we can completely characterize ${}^sF^j_{\mathcal{A}}$ with a single triplet $(\Phi_{ref}, {}^s\mathcal{D}_{ref}, \coprod_{i\in I} \Xi_i)$ where ${}^s\mathcal{D}_{ref} = \bigcup_{i\in I} \mathcal{D}^i_{ref}$ is the *supermaximal domain* (see Fig. 10) and $\coprod_{I\in I} \Xi_i$ is the disjoint union of all attributes and metadata.

6. Polyhedral normal cones bounding LMD directions

This section describes the construction of tight bounds for the tangent cone polar of a vertex, a curve segment, or a surface patch. Following the literature [19], we call those bounds *normal bounds* or *normal cones* even if they do not necessarily bound the normal cone as defined in convex analysis [3] which are supersets of tangent cones polars. Besides [19], other similar bounds exist [11, 12, 28] but they are both less tight and not always based on rigorous mathematical characterizations of LMDs. Additionally, normal bounds in related works are based on revolution cones.

Let us first recall the concept of *Gauss Map*, $\mathcal{G}$, of a surface which is a function $\mathcal{G} : \mathbb{R}^3 \to \mathcal{S}^2$ that maps each point of the surface to its (unit) normal $\mathbf{n}$ seen as a point on the unit sphere $\mathcal{S}^2$. We extend this concept to faces, edges, and vertices, when considering that $\mathcal{G}$ maps a point $\mathbf{x}$ of any of those features to the set of unit vectors on its tangent cone polar. The *Gauss Map Image* of a subset of a given feature is the union of the Gauss maps of all its points. To address the construction methods of the polyhedral normal cone introduced in section 6.1, $\mathcal{S}^2$ is embedded into a Euclidean space and given a local direct coordinate system centered at the origin. Its two poles are located at $+\mathbf{y}$ and $-\mathbf{y}$. The intersection curve of any plane passing through these poles (and the origin) with $\mathcal{S}^2$ is called a meridian (see Fig. 7(a)). Planes orthogonal to the $\mathbf{y}$ axis (but not necessarily containing the origin) intersect $\mathcal{S}^2$ along lines of latitude (see Fig. 7(b)).

While revolution cones [11] can be satisfyingly tight to bound normals of complex surfaces like Bézier surfaces, they can be extremely loose for edges and canonical faces for which tangent cone polars are often simply an arc on $\mathcal{S}^2$. We significantly improve the accuracy of those bounds with polyhedral cones instead of revolution cones. Within the following subsections, some proofs have been omitted for conciseness, but are illustrated with figures when possible.

6.1. Definition

We follow the usual definition of polyhedral cones from convex analysis [3]. Let $G = \{\mathbf{g}_1, \ldots, \mathbf{g}_n\}$ be a finite set of n vectors of $\mathbb{R}^3$. We call:

$$C = \text{cone}(G) = \text{cone}(\mathbf{g}_1, \ldots, \mathbf{g}_n) = \left\{ \mathbf{x} | \mathbf{x} = \sum_{j=1}^{n} \mu_j \mathbf{g}_j, \mu_j \geq 0, j = 1, \ldots, n \right\} \quad (6.1)$$

a *finitely generated (polyhedral) cone* and the $\mathbf{g}_j$ are its *generators*. In particular, we say that the set $G = \{\mathbf{g}_1, \ldots, \mathbf{g}_n\}$ of unit vectors is a *minimal set of unit generators* iff. none of its proper subsets generate the same cone: $\forall G' \subset G, G' \neq G \Rightarrow \text{cone}(G) \neq \text{cone}(G')$. We note $-C = \{-\mathbf{c} | \mathbf{c} \in C\}$ the cone opposite to C.

According to the Minkowski-Weil theorem [3], polyhedral cones can equivalently be defined as the intersection of a set of half-spaces whose boundary pass through the origin: $C = \{\mathbf{x} | \forall \mathbf{n}_j \in N, \langle \mathbf{n}_j, \mathbf{x} \rangle \leq 0\}$, where the finite set of vectors $N = \{\mathbf{n}_1, \ldots, \mathbf{n}_m\}$ are the (outward) normals of the planes of each half-space. Note that because polyhedral cone edges are necessarily multiples of a generator on C, each face normal is necessarily equal to the cross product of at least two generators. In particular, if G is a minimal set of unit generators, we say that G is *sorted* if:

$$\forall i \in [0, n] \mathbf{n}_i = \mathbf{g}_{i+1 \,(\bmod)\, n} \times \mathbf{g}_i \quad (6.4)$$

Fig. 8 shows a 3D polyhedral cone, its normals, and various possible sets of generators.

6.2. Intersection test

Let $C_a = \text{cone}(\mathbf{g}_1^a, \mathbf{g}_2^a, \ldots, \mathbf{g}_m^a)$ and $C_b = \text{cone}(\mathbf{g}_1^b, \mathbf{g}_2^b, \ldots, \mathbf{g}_n^b)$ be two polyhedral cones with nonempty interiors and m and n generators, respectively. The existence of LMDs given by Eqn. (3.5) and Fig. 3 is satisfied iff. their tangent cone polars contain antipodal directions, which we call *antipodal normals* here. Thus, given the two

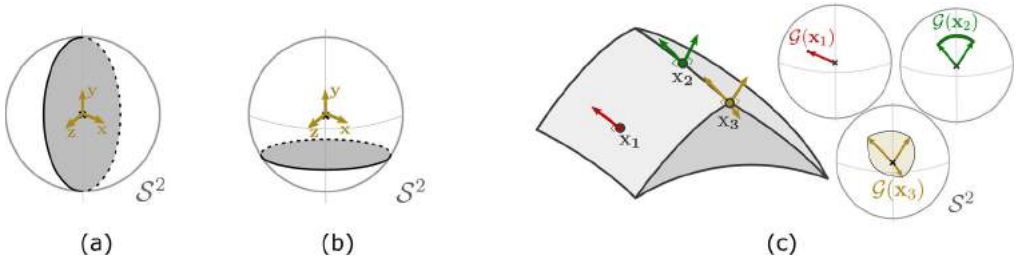

Figure 7. The unit sphere with its local coordinate system. An example of meridian (a), and line of latitude (b). (c) Left: points x_1, x_2, and x_3 respectively on a face, an edge, and a vertex. The surfaces normals at those points are shown as arrows. Right: their Gauss map images, i.e., tangent cones polar.

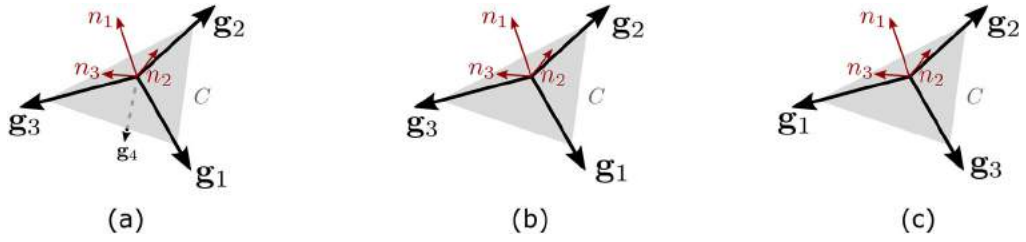

Figure 8. A polyhedral cone with its (outward) faces normal $N = \{n_1, n_2, n_3\}$. Its generators set: (a) $G = \{g_1, \ldots, g_4\}$ is not minimal because of g_4; (b) $G = \{g_1, g_2, g_3\}$ is minimal but not sorted because, e.g. $n_1 \neq g_2 \times g_1$; (c) $G = \{g_1, g_2, g_3\}$ is minimal and sorted.

bounds C_a and C_b of those tangent cone polars, it is useful to identify feature pairs such that $C_a \cap -C_b = \emptyset$ because they are guaranteed not to contain any LMD. First, we show that:

$$\begin{aligned} &\text{Int}(C_a) \cap \text{Int}(-C_b) \neq \emptyset \Rightarrow C_a \oplus C_b : \\ &= \text{cone}(\mathbf{g}_1^a, \ldots, \mathbf{g}_m^a, \mathbf{g}_1^b, \ldots, \mathbf{g}_n^b) = \mathbb{R}^3 \end{aligned} \tag{6.5}$$

where $\text{Int}(C_a)$ and $\text{Int}(-C_b)$ designate respectively the interior of C_a and $-C_b$. Polyhedral cones with empty interiors are not handled here. This limitation is avoided by dilating cones with empty interiors using the method described in section 6.3.3.

Proof: Let $\mathbf{v}$ be any vector of $\mathbb{R}^3$. If $\text{Int}(C_a) \cap \text{Int}(-C_b) \neq \emptyset$, then there exists a unit vector $\mathbf{u} \in \text{Int}(C_a)$ such that $-\mathbf{u} \in \text{Int}(C_b)$. Let $\mathbf{p}$ be a vector defining a point on the half-line starting at the origin with direction vector $\mathbf{u}$, i.e., $\mathbf{p} = \lambda\mathbf{u}$ with $\lambda \in \mathbb{R}_+^\star$. Let α be the angle between the vector $(\mathbf{v} - \mathbf{p})$ and the direction $-\mathbf{u}$. It comes:

$$\cos(\alpha) = \left\langle \frac{\mathbf{v} - \lambda\mathbf{u}}{||\mathbf{v} - \lambda\mathbf{u}||}, -\mathbf{u} \right\rangle$$

$$\cos(\alpha) = \frac{\lambda - \langle \mathbf{v}, \mathbf{u} \rangle}{||\mathbf{v} - \lambda\mathbf{u}||}$$

Then, it is possible to obtain any value of $\cos(\alpha) \in [\cos(\beta), 1[$ where $\cos(\beta) = \frac{\langle -\mathbf{v}, \mathbf{u} \rangle}{||\mathbf{v}||}$ is the cosine of the angle between $\mathbf{v}$ and $-\mathbf{u}$. Thus, λ can be chosen such that α takes any value in $]0, \beta]$. The Fig. 9(a) illustrates those elements.

In addition, because $\text{Int}(C_b)$ is open there exists an $\epsilon \in [0, \pi]$ such that$\forall \mathbf{d} \in \mathbb{R}^3$, $\arccos(\langle -\mathbf{u}, \mathbf{d} \rangle) \leq \epsilon \Rightarrow \mathbf{d} \in \text{Int}(C_b)$. By choosing λ such that $\alpha = \min(\epsilon, \beta)$ (as shown in Fig. 9(b)), it comes that:

$$(\mathbf{v} - \mathbf{p}) \in C_b \Rightarrow \mathbf{v} \in \{\mathbf{c}_b + \lambda\mathbf{u} | \mathbf{c}_b \in C_b\} \Rightarrow \mathbf{v} \in C_a \oplus C_b$$

because $\forall \lambda \in \mathbb{R}_+^\star, \lambda\mathbf{u} \in C_b$ by definition of a cone. Thus, $\mathbb{R}^3 \subset C_a \oplus C_b$. Because $C_a \oplus C_b$ is clearly a subset of $\mathbb{R}^3$ by definition, we conclude that $C_a \oplus C_b = \mathbb{R}^3$.

The contrapositive of this property provides an efficient and conservative intersection test: any pair of BVT

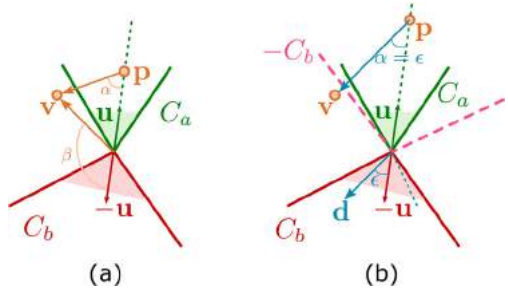

Figure 9. (a) 2D example of configuration of the elements of this proof. (b) A choice of ϵ and the corresponding location of $\mathbf{p} = \lambda\mathbf{u}$.

node for which the associated normal cones are such that $C_a \oplus C_b \neq \mathbb{R}^3$ can be rejected as their underlying feature pieces do not have any antipodal normals. Because $C_a \oplus C_b$ is itself a finitely-generated polyhedral cone, it is convex and $C_a \oplus C_b$ is clearly either equal to $\mathbb{R}^3$ or has a support plane passing through the origin. Note that to find a support plane of any cone, it is sufficient to find a plane that contains one of its faces and all its generators in a single half-space. Because of the relationship between the normals $\mathbf{n}_i \in N$ and the cone generators $\mathbf{g}_j^a$ (Eqn. (6.4)), the normals to faces of $C_a \oplus C_b$ can be obtained from the set of cross products of all combinations of generator pairs, i.e., $\mathbf{g}_i^a \times \mathbf{g}_j^a$, $\mathbf{g}_i^a \times \mathbf{g}_k^b$, and $\mathbf{g}_k^b \times \mathbf{g}_l^b$ for all $i, j \in [0..m], k, l \in [0..n]$. The intersection algorithm is summarized by Alg. 1. For each potential face normal of $C_a \oplus C_b$ computed at line 5, we have to check that all the generators of $C_a \oplus C_b$ lie on the same half-space. If they do, then we found a separating plane passing through the origin and can conclude at line 7 that $\text{Int}(C_a) \cap \text{Int}(-C_b) = \emptyset$ because of Eqn. (6.5). If this test fails for all face normals of $C_a \oplus C_b$, then we conclude at line 11 that C_a and C_b contain antipodal directions.

Assume C_a and C_b have approximately the same number of generators n. The test at line 6 performs $O(n)$ scalar products and comparisons. This is repeated for all potential normals of $C_a \oplus C_b$. Therefore, the overall complexity of the intersection test is $O(n^3)$. This is not adequate to improve or preserve the performances of LMD computation. Consequently, it is mandatory to set up a compromise between tightness of the polyhedral

normal cones and test efficiency by setting an upper limit on their number of sides (see section 6.3).

Alg. 1: Polyhedral Normal Cone intersection test.

1: **Inputs:** *the generators G_a, G_b of two polyhedral cones C_a and C_b .*
2: ***Output:****whether or not* $\text{Int}(C_a) \cap \text{Int}(-C_b) \neq \emptyset$.
3: **For all** $(\mathbf{g}_1, \mathbf{g}_2) \in (G_a \times G_a) \cup (G_b \times G_b) \cup (G_a \times G_b)$ **do**
4: **if** $\mathbf{g}_1$ and $\mathbf{g}_2$ are not collinear **then**
5: $\mathbf{n} \leftarrow \mathbf{g}_1 \times \mathbf{g}_2$ *// A normal of a potential face of* $C_a \oplus C_b$.
6: **if** all the scalar products $\langle \mathbf{g}, \mathbf{n} \rangle$ have the same sign for all $\mathbf{g} \in G_a \cup G_b$ **then**
7: **Return** FALSE *// All generators are on the same half-space.*
8: **end if**
9: **end if**
10: **end for**
11: **return** TRUE *// No separating plane found.*

6.3. Generating polyhedral normal cones

The computation of polyhedral normal cones for surfaces, edges, and vertices is carried out on a case-by-case basis in order to obtain tight bounds. It is sufficient to represent almost-minimally normal cones of all canonical surfaces, and provide bounds that are not too loose for complex surfaces, edges, and vertices. Note that our intersection test presented in section 6.2 requires polyhedral cones to have nonempty interiors. The following sections only computes closed polyhedral normal bounds. Then, they are dilated in section 6.3.3 to satisfy the non-empty interior property.

6.3.1. Polyhedral cone of a meridian or a line of latitude on $\mathcal{S}^2$

Polyhedral normal cones are necessarily convex, hence we assume that any arc of meridian or line of latitude to be bounded, describes an arc of $\mathcal{S}^2$ with a spanning angle $\alpha < \pi$. Otherwise, the resulting cone would contain all $\mathbb{R}^3$. If $\alpha > \pi$, the corresponding surface must be subdivided into smaller pieces to produce smaller Gauss Map images.

On the one hand, arcs of meridians are straightforward to bound because they are at the intersection of $\mathcal{S}^2$ with a plane passing through its poles (see Fig. 10(a)). On the other hand, because arcs of lines of latitude are the result of the intersection of $\mathcal{S}^2$ with a plane orthogonal to its revolution axis that may not contain the origin, it is not straightforward to bound it tightly (see Fig. 10(b,c,d)). Fig. 10(e) shows the resulting polyhedral cone in comparison to a revolution cone with ends up being much looser.

6.3.2. Polyhedral normal cone of a BRep feature

The polyhedral normal cone of a plane contains only one generator coinciding with the plane normal. Moreover, given the procedure presented above for a meridian or a line of latitude on $\mathcal{S}^2$, the computation of the polyhedral cone becomes straightforward for all canonical surfaces as well (see Fig. 11).

Computing the polyhedral normal cone of edges, vertices, and free-form surfaces is more difficult because the contour of their Gauss Map images neither may coincide with a meridian nor a line of latitude. We default to a more general approach in three steps. Firstly, we collect enough generators to form a polyhedral cone that bounds, not necessarily minimally, the polyhedral normal cone polar of each feature area. The method depends on the feature:

- Bézier surfaces: we compute explicitly the scaled normal surface [12] which is a Bézier surface too. Its control points benefit from a convex hull property [23] so they can be used as generators;
- Edges: a polyhedral normal cone is generated using the subsets of each surface adjacent to the given edge. Then, the sets of generators of both cones are merged. See the appendix of [6] for a proof that this is sufficient in order to bound the edge's tangent cone polar required by Eqn. (3.5);
- Vertices: the generators are given by the normals of the surfaces meeting at this point. Note that G^1 edges and vertices can be processed using a polyhedral cone without interior that can be dilated as described in section 6.3.3.

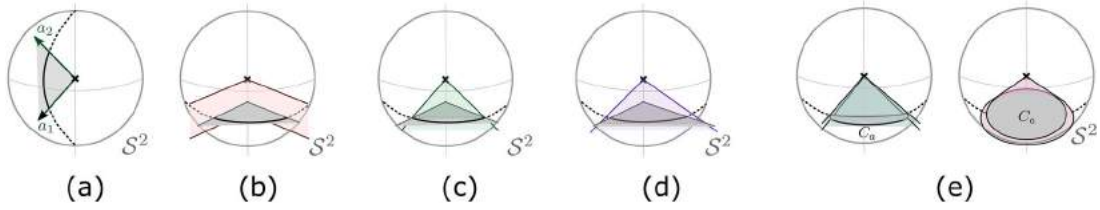

Figure 10. (a) Bounding an arc of meridian requires only the two generators $\{g_1, g_2\}$. Bounding arc of line of latitude is achieved using four planes: (b) two at its extremities, (c) one passing through the origin and both extremities, and (d) one passing though the origin and tangent to the curve (e.g. tangent to its middle point). (e) The resulting polyhedral cone (left) and revolution cone (right) of the same line of latitude. The revolution cone is much looser.

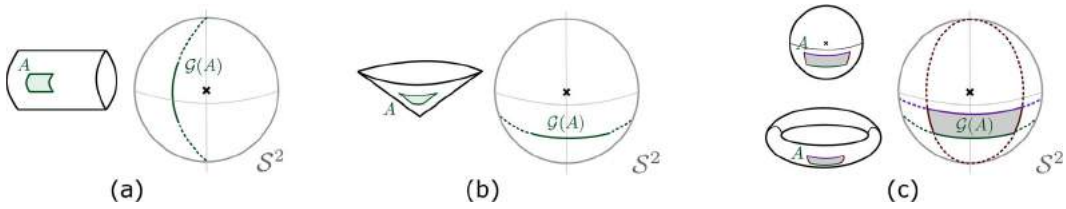

Figure 11. Areas$A$ of canonical surfaces and their respective Gauss map images $\mathcal{G}(A)$ on $\mathcal{S}^2$. This Gauss map image coincides with (a) a meridian for the cylinder, (b) a line of latitude for the cone, (c) an area delimited by two meridians and two lines of latitude for the torus.

Secondly, a minimal set of generators bounding the area of the feature's tangent cone polar is computed. Because the polyhedral cone is necessarily convex, this can be achieved by a convex hull algorithm performed on the unit sphere. Such an approach has been used by [19] with a method based on the Gift-wrapping convex hull algorithm. We designed a faster algorithm based on the 2D QuickHull method [2]. Given a set of unit generators $G \subset \mathcal{S}^2$, a direction $\mathbf{d} \in \mathcal{S}^2$, and an axis $\mathbf{a} \in \mathcal{S}^2$ orthogonal to $\mathbf{d}$, we define the *extremal generator toward* $\mathbf{d}$ *wrt.* $\mathbf{a}$ as the generator $\mathbf{g}^\star \in G$ that maximizes the angle $\mathbf{g}^\star = \text{argmax}_{\mathbf{g}\in G} \arccos(\langle \mathbf{g}, \mathbf{a} \times \mathbf{d}\rangle)$ and $\langle \mathbf{g}, \mathbf{d}\rangle \geq 0$. Our method is then the same as a 2D QuickHull algorithm where the concept of *extremal points* is replaced by our notion of *extremal generators*.

Because of the high complexity of the polyhedral cone intersection tests (see section 6.2), we limit the maximum number of sides of a polyhedral normal cone. We set this number to four as a compromise. Limiting the number of sides of the polyhedral normal cones amounts to bounding the cone computed by the spherical convex hull described by **Error! Reference source not found.** with another, larger, cone having only four sides, i.e., its minimal generator set has four elements. Computing the smallest four-sided cone is difficult and left to future works. Instead, the simplified cone is obtained as the intersection of four half-spaces, each containing the original polyhedral cone. For example each of those half-spaces can be chosen such that its border contains one face of the original polyhedral cone.

6.3.3. Polyhedral cone dilation

To ensure the intersection test shown in section 6.2 is applicable, and to support the feature of *angular regularization* introduced by [19], polyhedral cones computed in section 6.3 are *dilated*. Given a *dilation angle* $\alpha \in \mathbb{R}^{+\star}$ and a polyhedral cone C, this amounts to computing a new polyhedral cone, called *dilated polyhedral cone*, that bounds the set:

$$\tilde{C} = \left\{ \tilde{\mathbf{c}} | \exists \mathbf{c} \in C \text{ such that } \arccos\left(\frac{\langle \mathbf{c}, \tilde{\mathbf{c}}\rangle}{||\mathbf{c}||||\tilde{\mathbf{c}}||}\right) \leq \alpha \right\} \tag{6.6}$$

Fig. 12(a,b,c) shows the three configurations that may arise.

When $\text{Int}(C) \neq \emptyset$ (see Fig. 12(b,c)) and given the cone's minimal sorted set of n unit generators $G = \{\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_n\}$, the face normals of the dilated polyhedral cone of C with a dilation angle α are computed by rotating the original cone faces normals $(\mathbf{g}_{i+1} \times \mathbf{g}_i)$ along the axis $(\mathbf{g}_{i+1} - \mathbf{g}_i)$. When C has an empty interior as in Fig. 12(d,e), the cone is first enlarged on the plane containing the two generators, then the obtained auxiliary vectors are rotated to generate the four generators of the dilated cone.

7. Bounding volume tree construction

The data structure at the core of our method is a binary bounding volume tree very similar to the Spatialized Normal Cone Hierarchy introduced in [11] and applied to non-convex polyhedrons in [19]. Instead of working with bounding spheres and normal cones of revolution

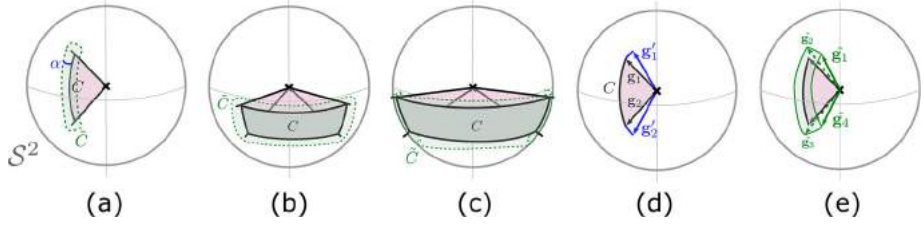

Figure 12. Polyhedral cones C and the set $\tilde{C}$ that must be bounded by another polyhedral cone to account for the dilation of angle α. (a) C has an empty interior. (b) C has a non-empty interior. (c) C has a non-empty interior but $\tilde{C}$ is so large it cannot be bounded by a polyhedral cone other than $\mathbb{R}^3$. (d) Auxiliary vectors for the dilation of an arc. (e) In green, the arc's dilated polyhedral cone.

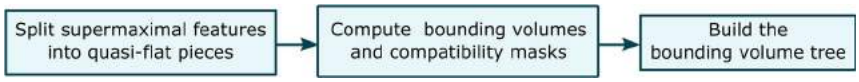

Figure 13. steps performed to build the bounding volume tree.

as [11], we use Oriented Bounding Boxes (OBB) as spatial bounding volumes and polyhedral cones as bounding volumes for normals. In practice, they offer a better balance between tightness and efficiency of intersection tests. The construction steps to generate the bounding volume tree are given in Fig. 13.

7.1. Supermaximal features splitting

Likewise [6] each supermaximal face and edge is broken down into small areas in order to improve the tightness of the chosen bounding volumes. In particular:

1. Surfaces often have holes because of some of their trimming curves (forming internal loops) or because of the topology of the complete B-Rep model. Because an OBB is convex it cannot fit tightly to a shape with holes. It should be subdivided into subsets without large holes that can be more tightly bounded. See Fig. 14 for an example.
2. We need each area to be *sufficiently flat*. As discussed in [6], some distance computations require the resolution of a high-degree polynomial equation. An iterative scheme extracts at most one solution for one given iteration starting point. Thus, the algorithm should ensure that concave features of the B-Rep model are split into subsets where each of them may contain at most one solution for the root-finding process. Splitting the model into almost-flat pieces is a common way to reduce the risks to encounter this issue. A first method is to split surface areas (resp. edges) until their revolution normal cone (resp. polyhedral normal cone) has a small half-angle [10, 26]. Because this generates good results in practice, we chose to use this approach. Other splitting criteria exist [7, 18, 26] but fail to capture areas of surfaces with high curvatures.

In practice, after choosing a sufficiently small angular upper bound, we first subdivide the features with respect to this second criterion (flatness requirement), and then remove any piece that lies completely on a hole. However, using this method, highly curved freeform surfaces may produce a very large number of subdivisions. Thus, we stop the subdivision as soon as the OBBs become too small. Both the angular upper bound and the minimal OBB size limit have to be chosen by the user, depending on the complexity of the simulated scene.

Note that splitting a curve or a surface is obtained in two different ways, depending on their types: canonical curves and surfaces are split using new bounds of their parametric domain whereas Bézier curves and surfaces are effectively subdivided along isoparametric curves to obtain new surfaces with new control points to simplify greatly the construction of their bounding volumes.

7.2. Bounding volumes and compatibility masks computation

Next is the computation of the bounding volume of each feature area. While the computation of OBBs for a canonical surface or curve is straightforward, computing the OBB of a Bézier surface, (resp. curve) relies on its *positivity* property, i.e., the fact that the whole surface (resp. curve) is contained by the convex hull of its control points [23]. Thus, an OBB bounding the control points will bound the Bézier surface itself as in [22]. The construction of polyhedral normal cones for vertices, edges and surfaces were described in detail in section 6.3.

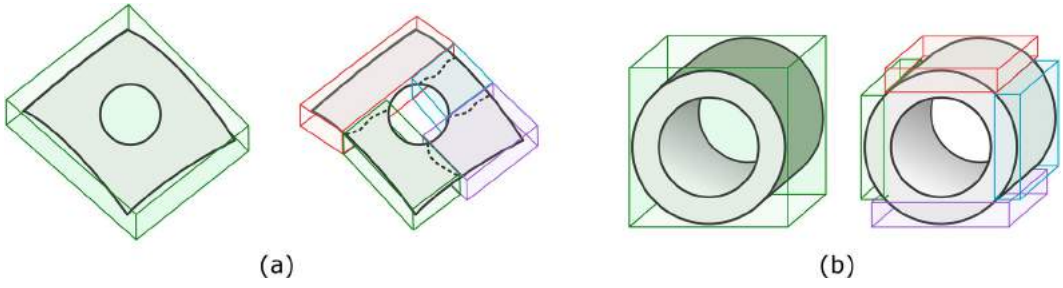

Figure 14. (a) A surface trimmed such that it has one circular hole. A single large OBB (left) is not tight since it contains the hole. Splitting the surface into smaller areas (right) produces tighter bounds. (b) A cylindrical tube with a cylindrical hole. Bounding the outermost convex cylinder with one OBB (left) is much less tight than using several OBB (right).

Table 1. Compatibilities between different types of surfaces and curves. A symbol × indicates an incompatibility. A symbol + indicates a compatibility and any pair missing from this table are compatible. *Concave sp/cy/co* designates concave spheres, concave cylinders, and concave cones.

	Plane	Cone/ Cylinder	Concave/ sp/cy/co	Concave torus	Line
Planes	×	×	×	×	×
Cone/Cylinder	×	+	×	+	+
Concave sp/cy/co	×	×	×	×	×
Concave torus	×	+	×	×	+
Line	×	+	×	+	+

In addition to bounding volumes, each feature area of a shape is given *compatibility attributes*. Those attributes indicate whether or not a given pair of surfaces or curves can contain any LMD at all. For example, a line cannot have an isolated closest point with a concave cylinder without intersecting it, which is forbidden for our method to be usable. Tab. 1 summarizes the pairs of features that are incompatible for similar reasons.

This concept of compatibility can be implemented very efficiently with the help of bit masks. Testing for compatibility has the runtime cost of a bitwise *and*.

7.3. Tree construction

Finally, all the bounding volumes and masks computed so far are aggregated into a binary tree using a top-down strategy. Each node has either zero or two children, i.e., this is a *proper* binary tree. Each leaf contains: an area of a surface or curve or a vertex, a reference to the corresponding supermaximal feature, an OBB, a polyhedral cone, and compatibility attributes. Internal nodes neither contain a reference to any geometric area nor supermaximal feature but contain:

- An OBB constructed by bounding all the OBB of the leaves of the sub-trees rooted by its children;
- A polyhedral normal cone constructed by bounding the polyhedral cones C_a and C_b of its two children. As shown in Fig. 15, if $C_a \cap -C_b = \emptyset$ (which can be tested with Alg. 1) then they are merged using the union of their generators as input to the procedure described in section 0. Otherwise, $C_a \cap -C_b \neq \emptyset$ meaning that they contain antipodal directions and can only be bounded with the polyhedral cone containing $\mathbb{R}^3$ completely.
- Compatibility attributes computed by combining the two type masks and two compatibility masks of its two children with a bitwise *or*.

8. Comparison and performance

Our collision detection method is implemented within the interactive multibody physics simulation framework XDE [20] developed by the CEA. Simulations are performed with no friction, no penetration, and are based on a time-stepping scheme. Contact constraints follow the Signorini contact law and are solved using a Gauss-Seidel based iterative solver similar to the algorithms described in [1].

In this section, we compare our method with our previous work [6] and to a state-of-the art distance-based collision detection method that operates tessellated models only [19] (using the original authors' code). This benchmark is performed on two scenarios involving industrial models mostly composed of canonical surfaces. All manipulations are performed using a SpaceMouse.

8.1. First scenario: telerobotics insertion task

Our first scenario simulates the slave arm side of a force-feedback teleoperation system. A robotic arm has to be manipulated by an operator in order to insert the conical mobile solid (Fig. 16(e) bottom) into a static hole that has a similar shape (Fig. 16(e) top). The mechanical clearance between the mobile solid and the hole after insertion is very small, i.e., equal to 1% of the hole's largest radius. There is no friction.

We distinguish four phases of the simulation because they all show significantly different performance characteristics. The *free flight* phase corresponds to Fig. 16(a) where the operator adjusts the position of the conical

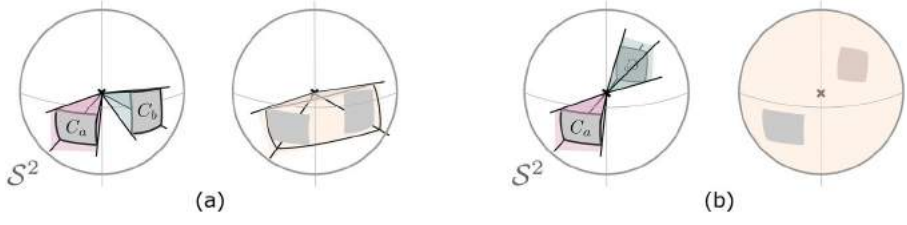

Figure 15. Computation of the polyhedral cone of an internal node (right, orange) by merging the cones C_a (left, red) and C_b (left, green) of its two children. (a) $C_a \cap -C_b = \emptyset$ so the resulting cone is non-degenerated. (b) C_a and C_b contain antipodal directions so the resulting cone contains $\mathbb{R}^3$ completely.

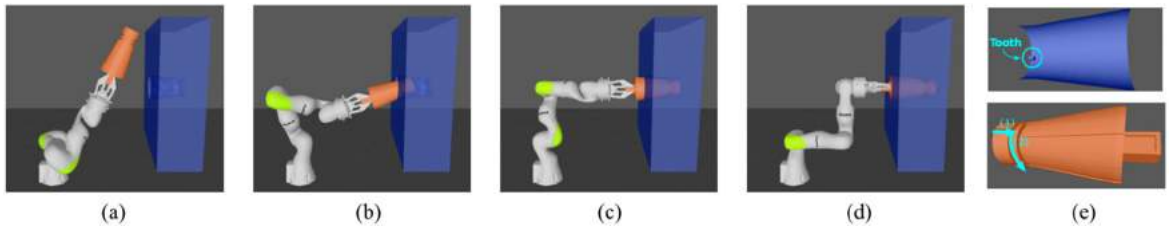

Figure 16. (a-d) Interactive insertion of a conical solid (with a cylindrical end) into a hole with a similar shape. (e) Cut view of the static object hole and a close view of the mobile conical solid. The manipulated object has to be translated then rotated such that the tooth attached to the hole follows the arrows depicted on the solid.

solid so that the insertion process can begin as in Fig. 16(b). This involved almost no collisions. Then, the *insertion and locking* phase slides (see Fig. 16(c)) and rotates the solid into the conical hole in such a way that it becomes fully inserted: objects start being in contact. Note that some precise manipulations are necessary in order to account for the tooth shown Fig. 16(e). Once fully inserted as in Fig. 16(d), small rotations are applied to the solid. The number of contacts is significantly increased since many faces of both solids are almost in conformal contact. Finally, the *extraction* phase removes the solid from the hole.

Comparison of collision detection times during the whole simulation is given in Fig. 17(a). This accounts only for collision detection. The contact constraints resolution time is not included. We observe that our method provides an almost constant speedup of 15% to 20% compared to our previous work. Thus, the performance gap when compared to the tessellation-based approach [19] is further increased in favor of our method based on smooth models because the tessellation-based approach requires a high number of triangles to overcome numerical artifacts due to the discretization. We used two discretization levels: one which we call *coarse* because it is the coarsest discretization that allows the insertion to be performed despite the discretization errors. Indeed, the discretization's maximal chord error is chosen equal to the gap between both objects. The second *thin* tessellation is 30% thinner than the coarse one: it has been empirically determined as a discretization that allows the manipulation to be done satisfyingly smoothly, i.e., without numerical artifacts that are noticeable enough to make the insertion more difficult.

Moreover, note that the number of generated contacts has a significant impact on the performance of the overall simulation. Indeed, each contact point represents at least one contact constraint that has to be handled by the dynamics solver so the higher the number of contacts, the larger the time to solve the constraints. Fig. 17(b) shows that the number of contacts our method generates is more than 40% lower than with the tessellation-based approach [19].

8.2. Second scenario: ROV insertion task

Our second scenario depicted in Fig. 18 stages the underwater insertion of a hotstab on a receptacle by a Remotely Operated Vehicle (ROV). This requires a great level of simulation accuracy as the smallest mechanical clearance is equal to 0.1mm where the hotstab diameter is four hundred times larger. Handling such scenarios with tessellated models requires a prohibitively thin tessellation. Similarly to the previous scenario, we distinguish different steps. First the hotstab is initialized outside of the receptacle and is inserted. Then, once fully inserted, we apply small rotations to the hotstab wrt. its symmetry

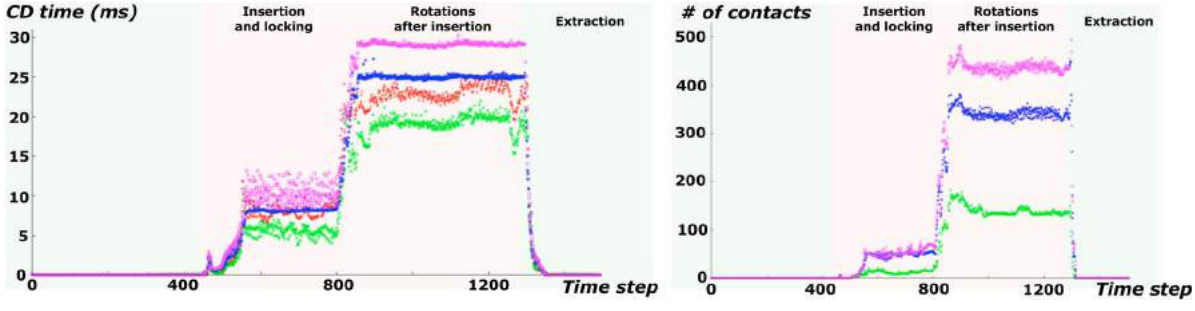

Figure 17. (a) Time to compute contact points using the polyhedra-based method [19] with thin (purple) or coarse (blue) tessellations. Our previous work (red) [6]. And after addition of our new improvements (green). (b) Comparison of number of contact points.

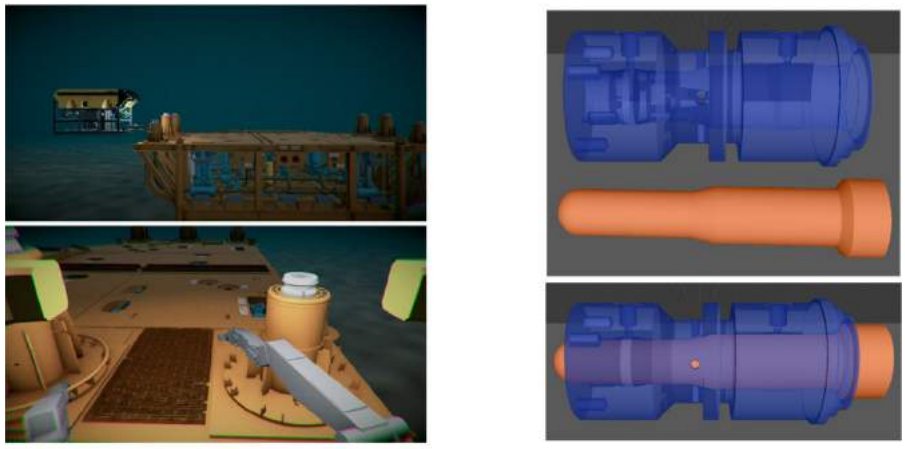

Figure 18. (a) Global view of the simulation. The (yellow) ROV is equipped with two (gray) arms that allow it to grab the (orange) manifold and tools necessary to manipulate the hotstab. The bottom image shows the view available during the teleoperation. (b) Our simulated scene focusing on the (orange) hotstab and the (blue) receptacle shown before and after insertion. (Images and CAD models courtesy of TechnipFMC)

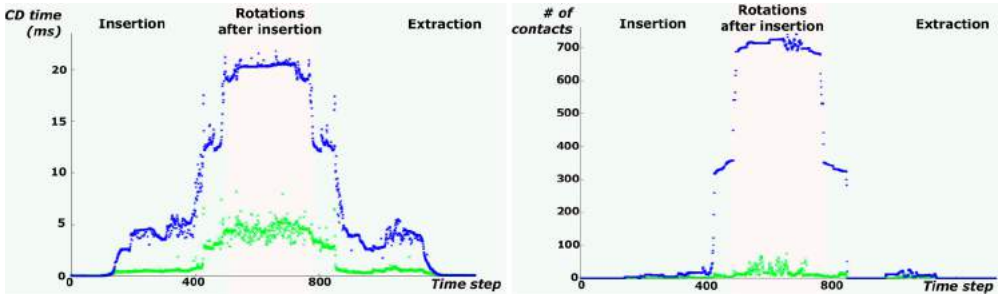

Figure 19. (a) Time to compute contact points using our method (green) compared to the polyhedra-based one [19] with the coarsest possible tessellation (blue). (b) Comparison of the number of contact points.

axis. Finally, it is extracted from the receptacle to recover its initial position.

Comparison between our method and the polyhedra-based method [19] is given by Fig. 19. For the second method, we used a tessellation with a chordal error (for each object) equal to half the smallest mechanical clearance allowed for the insertion to be feasible despite the geometric approximations. Because the insertion is tighter than for our previous scenario, and the contact areas larger, the performance difference between our method and the polyhedra-based one is more significant than in our first scenario. Here, using our method reduces the computation times by a factor of four and the number of actual contacts by a factor of ten to fifty. The overall simulation time (including constraints resolution) is thus significantly improved by a factor of almost 7.

9. Conclusion and future works

We presented in this article a method for distance-based collision detection between two solids modeled by their boundary representations (B-Rep). It is based on a bounding volume tree and on optimizations made possible by the observation that most industrial CAD models are composed mostly of canonical surfaces (Plane, Sphere, Cylinder, Cone, and Torus). We described two significant improvements over our previous works [6]. The first contribution unifies the data structures of supermaximal faces by merging the domains of its parts into what we call a *supermaximal domain*. Our second contribution improves significantly the tightness of normal bounding volumes by using polyhedral cones instead of revolution cones. This allows us to achieve real-time performances for simulations involving complex industrial CAD models. Indeed, computation times are improved by over 20% compared to our previous works, and by over 30% compared to polyhedra-based approaches (using a discretization that is just as thin as necessary to make the simulation work).

Our next step is to lower even more computation times of our approach in order to make our method applicable to haptic simulations. We intend to achieve that

by parallelizing most steps of our method. Moreover, as we use our method as part of a dynamics simulation framework, we intend to increase further the simulation accuracy by improving the constraints solver to handle non-linearities of contact constraints and contact kinematics.

Acknowledgements

The authors thank TechnipFMC for providing CAD models and use cases for the software benchmarks.

ORCID

Sébastien Crozet http://orcid.org/0000-0002-6011-1701
Jean-Claude Léon http://orcid.org/0000-0003-3337-081X
Xavier Merlhiot http://orcid.org/0000-0002-3788-2671

References

[1] Acary, V.; Brogliato, B.: Numerical Methods for Nonsmooth Dynamical Systems, Springer, 2008. https://doi.org/10.1017/CBO9781107415324.004
[2] Barber, C.B.; Dobkin, D.P.; Huhdanpaa, H.: The quickhull algorithm for convex hulls, *ACM Transactions on Mathematical Software* 22(4), 1996, 469–483. https://doi.org/10.1145/235815.235821
[3] Bertsekas, D.; Nedić, A.; Ozdaglar, A.: Convex analysis and optimization, 2003.
[4] Boussuge, F.; Léon, J.-C.; Hahmann, S.; Fine, L.: Extraction of generative processes from B-Rep shapes and application to idealization transformations, *Computer-Aided Design* 46, 2014, 79–89. https://doi.org/10.1016/j.cad.2013.08.020
[5] Chou, W.; Xiao, J.: Real-time and Accurate Multiple Contact Detection between General Curved Objects, International Conference on Intelligent Robots and Systems, IEEE, 2006, 556–561. https://doi.org/10.1109/IROS.2006.282364
[6] Crozet, S.; Léon, J.-C.; Merlhiot, X.: Fast computation of contact points for robotic simulations based on CAD models without tessellation, 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2016, 937–944. https://doi.org/10.1109/IROS.2016.7759162
[7] Dyllong, E.; Luther, W.: Distance calculation between a point and a NURBS surface, 2000.
[8] Gilbert, E.G.; Chong Jin Ong.: New distances for the separation and penetration of objects, Proceedings of the 1994 IEEE International Conference on Robotics and Automation, IEEE Comput. Soc. Press, 1994, 579–586. https://doi.org/10.1109/ROBOT.1994.351237
[9] Von Herzen, B.; Barr, A.H.; Zatz, H.R.: Geometric collisions for time-dependent parametric surfaces, Proceedings of the 17th annual conference on Computer graphics and interactive techniques - SIGGRAPH '90, ACM Press, 1990, 39–48. https://doi.org/10.1145/97879.97883
[10] Johnson, D.; Cohen, E.: Distance extrema for spline models using tangent cones, Graphics Interface, 2005.
[11] Johnson, D.E.; Cohen, E.: Spatialized normal come hierarchies, Proceedings of the 2001 symposium on Interactive 3D graphics - SI3D '01, ACM Press, 2001, 129–134. https://doi.org/10.1145/364338.364380
[12] Kim, D.-S.; Papalambros, P.Y.; Woo, T.C.: Tangent, normal, and visibility cones on Bézier surfaces, *Computer Aided Geometric Design* 12(3), 1995, 305–320. https://doi.org/10.1016/0167-8396(94)00016-L
[13] Krishnan, S.; Gopi, M.; Lin, M.; Manocha, D.; Pattekar, A.: Rapid and Accurate Contact Determination between Spline Models using ShellTrees, *Computer Graphics Forum* 17(3), 1998, 315–326. https://doi.org/10.1111/1467-8659.00278
[14] Li, K.; Foucault, G.; Léon, J.-C.; Trlin, M.: Fast global and partial reflective symmetry analyses using boundary surfaces of mechanical components, *Computer-Aided Design* 53, 2014, 70–89. https://doi.org/10.1016/j.cad.2014.03.005
[15] Lin, M.; Gottschalk, S.: Collision detection between geometric models: A survey, In *Proc. of IMA Conference on Mathematics of Surfaces*, 1998, 37–56. https://doi.org/10.1.1.45.3926
[16] Lin, M.C.; Manocha, D.: Fast interference detection between geometric models, *The Visual Computer* 11(10), 1995, 542–561. https://doi.org/10.1007/BF02434040
[17] Lock, S.M.; Wills, D.P.M.: VoxColliDe: Voxel collision detection for virtual environments, *Virtual Reality* 5(1), 2000, 8–22. https://doi.org/10.1007/BF01418972
[18] Ma, Y.L.; Hewitt, W.T.: Point inversion and projection for NURBS curve and surface: Control polygon approach, *Computer Aided Geometric Design* 20(2), 2003, 79–99. https://doi.org/10.1016/S0167-8396(03)00021-9
[19] Merlhiot, X.: A robust, efficient and time-stepping compatible collision detection method for non-smooth contact between rigid bodies of arbitrary shape, Multibody Dynamics, Eccomas Thematic Conference, 2007.
[20] Merlhiot, X.; Garrec, J. Le; Saupin, G.; Andriot, C.: The XDE mechanical kernel: Efficient and robust simulation of multibody dynamics with intermittent nonsmooth contacts, International Conference on Multibody System Dynamics, 2012, 5–6.
[21] Montana, D.J.: The Kinematics of Contact and Grasp, *The International Journal of Robotics Research* 7(3), 1988, 17–32. https://doi.org/10.1177/027836498800700302
[22] Page, F.; Guibault, F.: Collision detection algorithm for NURBS surfaces in interactive applications, Canadian Conference on Electrical and Computer Engineering. Toward a Caring and Humane Technology, IEEE, 2003, 1417–1420. https://doi.org/10.1109/CCECE.2003.1226166
[23] Piegl, L.; Tiller, W.: *The NURBS Book*, Springer Berlin Heidelberg, Berlin, Heidelberg, 1997. https://doi.org/10.1007/978-3-642-59223-2
[24] Pungotra, H.; Knopf, G.K.; Canas, R.: Novel Collision Detection Algorithm for Physics-based Simulation of Deformable B-spline Shapes, *Computer-Aided Design and Applications* 6(1), 2009, 43–54. https://doi.org/10.3722/cadaps.2009.43-54
[25] Samuel, N.M.; Requicha, A.A.G.; Elkind, S.A.: Methodology and Results of an Industrial Part Survey, 1979.
[26] Selimovic, I.: Improved algorithms for the projection of points on NURBS curves and surfaces, *Computer Aided*

Geometric Design 23(5), 2006, 439–445. https://doi.org/10.1016/j.cagd.2006.01.007

[27] Stroud, I.: *Boundary Representation Modelling Techniques*, Springer London, London, 2006. https://doi.org/10.1007/978-1-84628-616-2

[28] Thomas, F.; Turnbull, C.; Ros, L.; Cameron, S.: Computing signed distances between free-form objects, International Conference on Robotics and Automation, IEEE, 2000, 3713–3718. https://doi.org/10.1109/ROBOT.2000.845310

[29] Weller, R.; Zachmann, G.: Inner Sphere Trees and Their Application to Collision Detection, In *Virtual Realities*. Springer Vienna, Vienna, 2011, 181–201. https://doi.org/10.1007/978-3-211-99178-7_10

[30] Zhang, X.; Kim, Y.J.; Manocha, D.: Continuous penetration depth, *Computer Aided Design* 46(1), 2014, 3–13. https://doi.org/10.1016/j.cad.2013.08.013

[31] Zhao, W.; Lan, Y.: A Fast Collision Detection Algorithm Based on Distance Calculations between NURBS Surfaces, 2012 International Conference on Computer Science and Electronics Engineering, IEEE, 2012, 534–537. https://doi.org/10.1109/ICCSEE.2012.2

[32] Zou, Z.; Xiao, J.: Tracking minimum distances between curved objects with parametric surfaces in real time, Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453), IEEE, 2003, 2692–2698. https://doi.org/10.1109/IROS.2003.1249277