Efficient contact determination between solids with boundary representations (B-Rep)

Sébastien Crozet

Communauté
UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTE UNIVERSITE GRENOBLE ALPES

Spécialité : **Mathématiques & Informatique**

Arrêté ministériel : 25 mai 2016

Présentée par

Sébastien Crozet

Thèse dirigée par **Jean-Claude Léon, Professeur, Laboratoire Jean Kuntzmann - INRIA équipe IMAGINE**, et co-encadrée par **Xavier Merlhiot, Ingénieur Chercheur, Commissariat à l'Énergie Atomique et Énergies Alternatives**

préparée au sein du **Laboratoire Jean Kuntzmann, et Commissariat à l'Énergie Atomique et Énergies Alternatives** dans **l'École Doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique**

Efficient contact determination between solids with boundary representations (B-Rep)

Détermination efficace des contacts entre solides représentés par modélisation surfacique (BRep)

Thèse soutenue publiquement le **08 décembre 2017**, devant le jury composé de :

Prof. Laurent Grisoni
Professeur, Université de Lille 1, Rapporteur
Dr. Paul Kry
Professeur associé, McGill University, Rapporteur
Dr. Bernard Brogliato
Directeur de Recherches, INRIA, Président du jury
Dr. Frédéric Dubois
Ingénieur de Recherche, CNRS, Examinateur
Prof. Jean-Claude Léon
Professeur, Grenoble INP, Directeur de thèse
Dr. Xavier Merlhiot
Ingénieur Chercheur, CEA List, Co-encadrant de thèse

Acknowledgements

First, I would like to thank all the members of the jury, Laurent Grisoni and Paul Kry for accepting the heavy task of being reviewers of this manuscript, as well as Bernard Brogliato for accepting to be the president of the jury.

I thank my outstanding thesis director Jean-Claude Léon as well as my supervisor Xavier Merlhiot for their trust during those three years, for their very complementary contributions to the various scientific discussions, as well as their great support and guidance.

I thank my colleagues from the team of the Laboratoire de Simulation Interactive of the CEA. In particular I thank Laurent Chodorge for hosting me in his laboratory and providing resources for completing this thesis. A special thank to Francois Keith and Bruno Bodin who helped integrating my work into the Unity environment. This allowed the setup of nice demos as well as the use of haptic devices to interact with simulations involving my work.

I thank my colleagues from the team Imagine from Inria for their support and logistical assistance.

Finally, I would like to thank deeply my family, Lucile, and Viola, for their everlasting support and encouragements during the whole thesis.

Of course, I thank as well any person not mentioned here whom I had the pleasure to interact with during this thesis.

Thank you!

Résumé

Avec le développement de systèmes robotiques avancés et de tâches de téléopération complexes, le besoin pour la réalisation de simulations en amont des opérations sur les systèmes réels se fait de plus en plus ressentir. Cela concerne en particulier les tests de faisabilité, d'entraînement d'opérateurs humains, de planification de mouvements, etc. Ces simulations doivent généralement être réalisées avec une représentation précise des phénomènes physiques, notamment si l'opérateur humain est supposé faire face aux mêmes comportements mécaniques dans le monde réel qu'avec la scène virtuelle. La détection de collisions, c'est-à-dire le calcul de points de contact et des normales de contact entre des objets rigides en mouvement et susceptibles d'interagir entre eux, occupe une portion significative des temps de calcul pour ce type de simulations. La précision, ainsi que l'ordre de continuité de ces informations de contact, sont de première importance afin de produire des comportements réalistes des objets simulés. Cependant, la qualité des informations de contact ainsi calculées dépend fortement de la représentation géométrique des parties de la scène virtuelle directement impliquées dans la simulation mécanique. D'une part, les représentations géométriques basées sur des volumes discrets (voxels, arbres de sphères, etc.) ou des tessellations permettent une génération de contacts extrêmement rapide mais, en contrepartie, peuvent introduire des artefacts numériques dûs à l'approximation des formes en contact. D'autre part, l'utilisation de représentations surfaciques lisses (composées de courbes et surfaces lisses) produites par les modeleurs CAO permet d'éliminer ce problème d'approximation. Cependant, ces approches sont actuellement considérées trop lentes en pratique pour des applications en temps réel.

Cette thèse est dédiée au développement d'un premier module informatique de détection de collisions entre solides représentés par une modélisation surfacique lisse et suffisamment efficace pour offrir des performances temps-réel pour certaines applications industrielles nécessitant un niveau de précision élevé. Ces applications prennent typiquement la forme de la simulation d'opérations d'insertion de composants en présence de jeu faible. L'approche proposée est basée sur une hiérarchie de volumes englobants et tire profit de caractéristiques clefs des composants mécaniques industriels. En particulier, les surfaces sujettes à des contacts fonctionnels sont généralement modélisées par des surfaces canoniques (cylindres, sphères, cônes, plans, tores). Les contacts sur des surfaces gauches telles que les NURBS sont généralement accidentels et rencontrés lors d'opérations de maintenance et d'assemblage. Notre hiérarchie de volumes englobants est améliorée par l'identification d'*entités supermaximales* afin d'éviter la localisation redondante de points de contacts entre surfaces canoniques parfois découpées en plusieurs entités distinctes par le modeleur CAO. De plus, le concept de *cônes polyédriques de normales* est défini afin d'établir des bornes de normales plus précises que les cônes de normales de révolution existants. De plus, le module ainsi développé est étendu afin de supporter des configurations incluant des câbles modélisés par des courbes de Bézier dilatées. Enfin, l'exploitation de la cohérence temporelle, ainsi que la parallélisation de l'ensemble des traitements clefs permet l'exécution en temps réel de certains scénarios industriels.

Mots-clefs:

BRep, CAO, Détermination du contact, Distance minimal locale, Détection de collision, Dynamique multi-corps, Moteur physique

Abstract

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 human operators, motion planning, etc. Such simulations usually need to be performed with great accuracy of physical phenomena if the operator is expected to face the same ones in the real world as well as in the virtual scene. Collision detection, i.e., the computation of contact points and contact normals between interacting rigid bodies, occupies a time-consuming part of such a mechanical simulation. 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 bodies of the virtual scene directly involved in the mechanical simulation. On the one hand, discrete volumes-based (voxels, sphere trees, etc.) and tessellation-based geometric representations allow very fast contact generation at the cost of the potential introduction of numerical artifacts due to the approximation of the interacting geometric shapes. On the other hand, the use of boundary representations (as issued by CAD modelers) composed of smooth curves and surfaces removes this approximation problem but is practically considered too slow for real-time applications.

This Ph.D focuses on developing a first complete collision detection framework on solids with smooth boundary representations that achieves real-time performances. Our goal is to reach the real-time simulation of industrial scenarios that require a high level of accuracy. Typical applications are insertion tasks with small mechanical clearances. The proposed approach is based on a bounding-volume hierarchy and takes advantage of key features of industrial mechanical components. Indeed, they are often modeled using surfaces describing functional contacts with canonical surfaces (cylinder, sphere, cone, plane, torus) while contacts over free-form surfaces like B-Splines are mostly accidental and encountered during approaching movements prior to effective insertions. We augment our bounding volume hierarchy with the identification of *super-maximal features* in order to avoid redundant exact localization of contact points on canonical surfaces that may be represented as distinct features of the CAD model. In addition, we define *polyhedral normal cones* that offer tighter bounds of normals than existing normal cones of revolution. Moreover, we extend our method to handle configurations that involve deformable beams modeled as dilated Bézier curves. Finally, the parallelization of the full approach enables the processing of industrial scenarios to be simulated in real-time.

Keywords:

BRep, CAD, Boundary representation, Contact determination, Local minimal distance, Collision detection, Multibody dynamics, Physics engine

Contents

Introduction

Context of collision detection and multibody dynamics simulation

Collision Detection (CD) encompasses several geometric queries of varying complexities ranging from simple binary interference tests to the computation of all the contact points between virtual 3D objects evolving in complex environments. The choice of a geometric query type and the required accuracy of their output strongly depend on the targeted application. For example, applications like accessibility tests and motion planning [113] rely mostly on binary interference tests. Thus, the task of the collision framework is simply the detection of intersections between multiple solids without necessarily producing any additional information regarding the geometric description of the intersection itself. Haptic simulations, animation, video games, and virtual training require more complex geometric informations whenever two solids touch each other or are about to collide. Such information is at the root of the definition of contact constraints that can be simulated by a dynamics simulation software. However, contact information can take several forms depending on the geometric representation of the objects being simulated, and on the desired accuracy and performance of the whole application:

- Applications like haptic simulations require the physical phenomena to be simulated in real-time and at a very high update rate (∼1000Hz). Those generally require fast responses of the collision detection engine in order to output force feedbacks at a high frequency. Thus, accuracy is typically sacrificed for performance using coarse approximations of the objects' geometric models (see Section 1.4.1) and approximate computations to estimate their penetration (see Section 1.3.2). That way, they benefit from the efficiency of penalty-based methods [115];

- Applications like video games rely on a lower update rate (∼60Hz) and require physical simulations to be efficient enough to allow real-time interactions while staying accurate enough to be visually appealing and plausible. Often, the geometric queries include the computation of penetrations, contact normals and tangents (for friction) and, sometimes, the localization of closest points. The accuracy of objects' geometric models is generally sacrificed in favor of approximate ones so that CD can be fast enough. Popular simulation packages include Bullet Physics [31], the Open Dynamics Engine [123] and PhysX [101]. A more exhaustive list and comparison can be found in [18];

- The real-time simulation of high-accuracy applications like interactive assembly and virtual training in real-time often allows smaller update rates (30 to 60Hz). This lets the application allocate more time for the simulation of the physical phenomena which must be more realistic as well. Update rates down to 30Hz remain acceptable because this gives the user an acceptable power of interaction with the simulated components. The executed geometric queries are similar to those required by video games though a higher level of accuracy is necessary so that penetrations can be avoided as much as possible. Thus, they may rely on distance informations (see Section 1.3.1) and polyhedral approximations

(see Section 1.4.2) or smooth representations (BRep, see Section 1.4.3) of their boundaries, depending on the specification of the application. Popular simulation packages include the eXtended Dynamics Engine (XDE) [91], Vortex [30], and AGX [6].

- Finally, some applications like accessibility test for maintenance operations require a high level of accuracy but tolerate non real-time execution of the simulation as long as the capabilities of the user to interact with the physical scene is not too inconvenient. In those cases the simulated physical phenomena appear as if they are slowed down since the time evolves more slowly in the simulated scene than in the real world.

This thesis aims at contributing to the third category, i.e., applications where accuracy is more important than computation times as long as the simulation remains real-time and with a reasonable update rate. Ideally, an update rate of 60Hz is targeted, implying that at most 16ms can be allocated to update the physical scene (including CD). Indeed, the multibody systems of the targeted industrial applications become increasingly complex with geometric models that are highly curved and mechanical clearances between assembly components that become of the order of tenths of millimeters or smaller. As an example, the scenario depicted by Figures 1 and 2 involves a simulated underwater Remotely Operated Vehicle (ROV) performing a docking procedure[1]. This application can be seen as the simulation of an insertion task with small mechanical clearances. Additionally, once the insertion is complete as shown in Figure 2a, the fingers are deployed thanks to an hydraulic actuator (see Figure 2b) and have to be simulated as well in order to complete the docking. The original 3D models of the tips being inserted and the hole are mostly represented as smooth pieces of cylinders and cones. Other examples, with benchmarks of our methods, are presented in Chapter 5.

The real-time and haptic simulation of this type of applications with solids fitting together with small mechanical clearances and involving rolling and sliding motions that occur frequently, is still a challenge. Indeed, efficient methods based on geometric approximations (such as triangle meshes) are hardly applicable without using an excessively high number of approximating elements. For example, tessellation-based solutions produce fast and accurate results for most common industrial use-cases, but fail for such insertion tasks since high accuracy of the contact geometry is necessary to achieve the required level of realism while keeping acceptable performances. Indeed, as discussed in Section 1.4.2, polyhedral approximations induce numerical errors with noticeable consequences from the user point of view like, e.g., unrealistic shocks during the simulation causing "jumps" or an inconsistent loss of kinetic energy, or even the impossibility to insert an object completely into another one containing its imprinted cavity or socket.

Limitations of existing approaches

Existing real-time simulation frameworks for industrial real-time multibody simulations rely on approximate geometric representations of the input models of their constitutive objects in order to keep the computational cost of CD sufficiently low. One of the most commonly used approximations, the polyhedral representation which is a piecewise-linear approximation of the object models boundaries, introduces numerical errors into the simulations that may affect significantly the trajectory of the simulated objects. Indeed, the set of positions reachable by the dynamic objects cannot be represented exactly due to the fact that non-penetration constraints inherit their accuracy and smoothness from the conformity of the geometric models used for CD with regard to the ones being manufactured. This issue can be worked around, e.g., when using polyhedral representations with an increasing the number of polygons and hence, refining the

[1]The structure on which the ROV is docking is hidden on the pictures due to confidentiality restrictions.

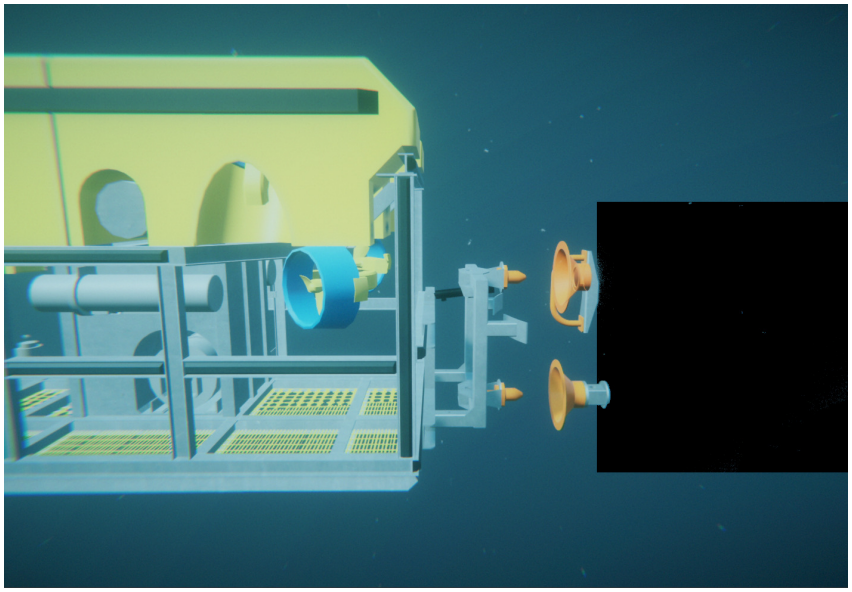


Figure 1: Underwater ROV about to dock to a structure (hidden in black) by inserting two orange components into the orange sockets. *Image courtesy of TechnipFMC.*

approximation. But this is achieved at a great performance cost since more polygons imply an increased computation time dedicated to collision detection, a greater memory footprint, and more contact constraints to be solved (see Figure 3), which slows down the dynamics solver as well. Consequently, there is a point where switching to smooth geometric representations becomes necessary for some application in order to meet the required performances while maintaining an optimal accuracy.

Indeed, a drastic solution for dealing with those inaccuracy issues consists of getting rid of all kinds of geometric approximations of the input geometric models and use the actual smooth geometries produced by CAD[2] modelers. For example, most CAD modelers output 3D models represented as a smooth BRep solids. Roughly speaking, the boundary of a BRep solid is composed of points, curves, and surfaces, connected together (see Section 1.4.3). Using this kind of smooth representation has the immediate benefit of allowing the non-penetration constraints as well as the set of reachable configurations, to be represented without approximation. However, while this setting produces optimal accuracy, the corresponding existing CD methods still suffer from performance or accuracy issues. Moreover, only few approaches in the literature attempt to compute collisions between full CAD models rather than isolated curves and surfaces (see Section 1.5). Attempting to fill this gap is the purpose of our CD framework.

Structure of the manuscript

The goal of this manuscript is to describe a new framework for the computation of Local Minimal Distances (LMD) between smooth industrial BRep models (representing the geometrical shape of rigid bodies) and deformable Bézier curves (representing the geometrical shape of deformable beams with circular cross-sections). It is organized into five chapters:

- Chapter 1 states formally the context and objectives through the mathematical description and interpretation of non-penetration constraints and LMDs. The questions about the choice of a geometric contact model as well as the geometric representations of the

[2]Computer-Aided Design

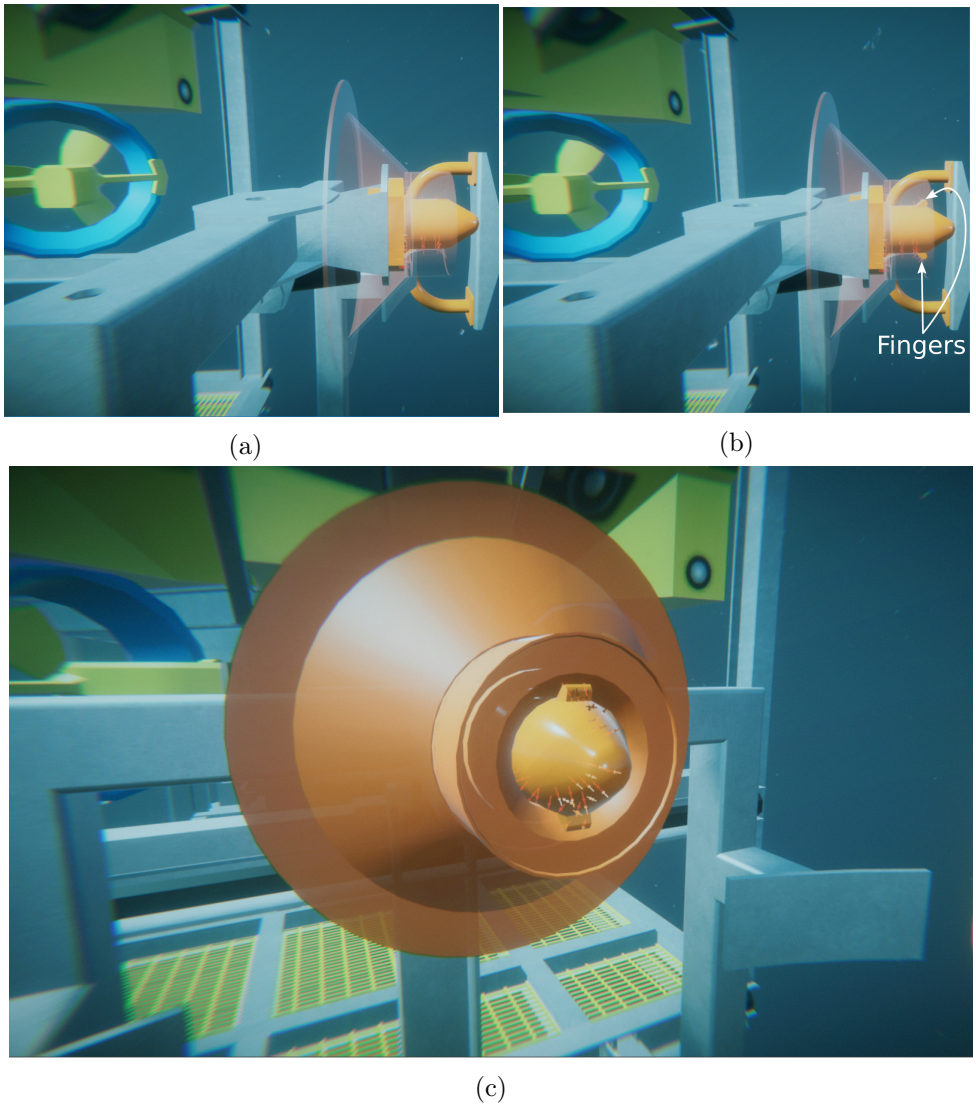
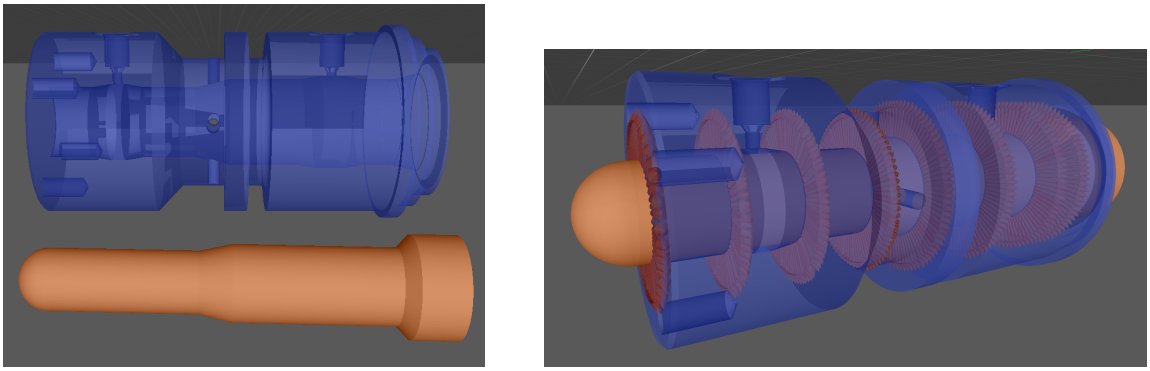


Figure 2: Side view of the docking procedure. (a) After insertion. (b) After locking by deploying two fingers. (c) Close view where some contact points are visible is small red arrows. *Images courtesy of TechnipFMC.*

simulated objects are addressed and contextualized for the applications targeted by the proposed framework in order to motivate its objectives. The latter are the fast and efficient computation of LMDs between smooth BRep models to accurately describe rolling and sliding motions between components. Existing methods for the computation of distances between smooth curves and surfaces are detailed;

- Chapter 2 describes all the operations performed only once as an initialization step, i.e., before the execution of any actual geometric query. The goal of this offline phase is to build a bounding volume hierarchy for each BRep model and deformable curve. This hierarchy is filled up with attributes that are widely used in the literature like oriented bounding boxes and cones of revolution. Additionally, we add logical informations resulting from an analysis of the shape structure of the input model. This analysis includes the grouping of similar surfaces and curves into new types of entities called *supermaximal faces* and *supermaximal surfaces*. As a complement, the curvatures of various surfaces are studied to



(a) (b)

Figure 3: Simulation of the insertion of an hotstab (orange) operated by a ROV into a container (blue). The mechanical clearance is small ($1/100^{th}$ of the hotstab outer diameter). All the red marks in (b) identify a contact constraint. There are approximately 800 linear contact constraints while this work output around 10 non-linear contact constraints. Refer to Section 5.2 for more details on this scenario.

determine *compatibility attributes* that identify some pair of surface types that can never contain closest points;

- Chapter 3 defines a new bounding volume dedicated to LMD directions. This bounding volume, defined as a mathematical cone with a polygonal base, is used to bound normals, tangents, and directions of projection. As a tight bounding volume, it leads to a more discriminative culling test in order to identify more efficiently the areas from two objects that may contain closest points. Additionally, methods for computing those bounding volumes for vertices, edges, and faces areas depending on their actual geometric representation, are provided;

- Chapter 4 describes how the pre-computed data generated by the offline phase is exploited at run-time to compute the LMDs between two smooth BRep models or deformable Bézier curves. The common tree-traversal approach is detailed and adapted to take into account our new bounding volume as well as the supermaximal entities and compatibility attributes. An hybrid method that combines the bounding volume hierarchy with dynamic subdivisions is proposed for deformable curves. Computation times are then significantly improved by the parallelization of the whole CD pipeline and the incorporation of temporal coherence to avoid repeated computations from one time-step of the dynamics simulation to the next;

- Chapter 5 analyzes the performance of the proposed framework applied to various tasks that require smooth geometric representations to be properly simulated in real-time. Performance comparison is provided between our framework and an alternative distance-based CD framework handling polyhedral representations during simulations.

Finally, the conclusion summarizes the contributions brought by our work and highlights some limitations of our framework. Potential directions for research leading to improvements overcoming those limitations and increasing further the simulation accuracy when our framework is integrated to a dynamics simulation engine, are proposed.

Chapter 1

Computing contact points between industrial CAD models: representation-dependent approaches

This first chapter describes the context of the work presented in this manuscript together with some existing methods that partially address the various steps required for fast and real-time computations of closest points between industrial CAD[1] models. The goals of a collision detection framework, seen as part of a multibody dynamics simulation, is first presented in order to identify its objectives. After clarifying that the main goal of this work is the computation of geometric informations as required by the dynamics integrator in order to generate constraints and avoid penetration between the different simulated solids, two of the main geometric contact models for real-time simulations are presented. Various geometric representations are detailed as well and compared from the point of view of the accuracy of their resulting non-penetration constraints. Those discussions lead to the identification of our objectives: the real-time computation of local minimal distances between smooth boundary representations of the objects' shape geometries. This removes inconsistencies that would be introduced into the non-penetration constraints by approximations of the component shapes. Several existing methods addressing the distance computation between smooth curves and surfaces are then presented and categorized. Overall, the complexity of closest point computation between smooth industrial models yield performance issues that will be under focus in the subsequent chapters.

1.1 Collision detection throughout the communities

Collision Detection (CD) is a broad field of research with applications throughout a wide range of communities. Depending on their applications, various communities have been motivated to discover methods with extremely different characteristics depending on the required computational efficiency and geometric accuracy. Three of those communities that played a significant role through the contributions to CD methods used for dynamics simulations can be characterized as follows:

1. The mechanics community is mostly concerned by the realistic modeling and analysis of the equations of motion and its related constraints, including non-penetration constraints,

[1] Computer-Aided Design

frictional 3D contacts and impacts, kinematic constraints, etc. In particular, substantial research has been made regarding the definition of contact models and the well-posedness, conditions of existence, and uniqueness of solutions for the constrained equations of motion [142, 10, 20, 37, 4]. Because this community studies the mechanical systems from motions and forces points of view, the problems related to collision detection and of representation of the geometric models are most often not addressed explicitly. Indeed, it is generally considered that CD provides whichever geometric information necessary to define contact constraints that are often assumed to be smooth functions (see Section 1.2.1). Such a smoothness naturally derives from the smooth shapes of the components;

2. The computer graphics (CG) community relies on dynamics simulation in order to generate interactions between solids, such that their behaviors seem either credible for the user or follow custom 'physical' laws in order to generate specific artistic effects. Video games, animations, and generation of special effects for movies are typical applications where the use of physical simulations is common. Because of the low requirement regarding the accuracy of the simulation compared to the prominence given to highly efficient user interactions, a focal point of the CG community is the development of fast algorithms for the simulation of the dynamics of rigid body systems. This motivated the creation of fast CD algorithms relying on geometric approximations of the shape of the rigid bodies. These approximations are mostly nonsmooth (e.g. triangle meshes which have discontinuous normals) but lead to the design of efficient CD algorithms. Particularly, the design of fast data structures including bounding volume hierarchies like sphere trees [111, 104, 64], OBB[2] trees [52, 34], and trees of k-DOPs[3] [75] made several CD queries like interference tests, distance computations, and intersection computation, applicable within real-time applications.

   As discussed into more details in Section 1.4, methods based on polyhedral approximations may affect significantly the behavior of the simulated objects that departs from the real ones. Therefore, the generation of smooth contact constraints with continuous normals is of increasing interest. For example Kry et al. [80] combine contact kinematics equations with the equations of dynamics to propose a reduced-coordinates approach for the simulation of exactly one contact between two smooth surfaces. Zhang et al. [144] propose a method to approximate smoothly the contact space while using polyhedral models;

3. Other communities like robotics and digital manufacturing have needs that range from simple geometric queries, e.g., accessibility and clearance tests [78] or motion planing [113], to full-featured dynamics simulation for the simulation of tele-operations, the training of human operators using virtual equipments, etc. In addition, haptic rendering would significantly benefit from CD between smooth objects because discontinuous contacts normals with respect to the motion parameters may generate unrealistic force feedbacks causing sensations of 'bumps' while the touched surface is actually smooth. In particular, Johnson et al. [68] did perform some preliminary works for the haptic rendering of contacts between a probe and a smooth curve or surface modeled as using splines.

Several surveys about CD methods bring in-depth information that can be synthesized as follows. Lin et al. [84] sort methods in accordance with the geometric representations while they are sorted by algorithmic schemes in [66]. Teschner et al. [131] present various methods for deformable objects and Kockara et al. [77] describe common approaches for CD frameworks based on two phases: the broad-phase that determines which pairs of objects are about to

[2]Oriented Bounding Box
[3]k-Discrete Oriented Polytopes

interact and the narrow-phase that generates detailed geometric informations about each pair of objects. Those two phases have been introduced by Hubbard [63] and are standard throughout all efficient CD framework for multibody dynamics simulation.

Let us note that this manuscript describes only the narrow-phase of our CD framework. Indeed, as discussed in Section 1.2.1, the focus is placed on the computation of contact informations between two solids, only. Thus, it is assumed that a broad phase has already identified the pairs of solids that may interact. Because existing broad phases like the sweep-and-prune [11] one or the hierarchical hash table [92] do not depend on the representation of the geometric models (but solely on bounding volumes that contain them), they are applicable as-is before executing our methods that are devoted to two objects. More details regarding the typical CD pipeline and various common algorithms for real-time collision detection can be found in the reference book [40].

This chapter introduces fundamental concepts motivating the choices made within the proposed framework as well as fundamental properties of closest points that can be exploited to design efficient algorithms:

- Section 1.2 details the equations of motion using the Lagrangian formalism applied to a system of rigid bodies as well as a system involving deformable solids. Non-penetration constraints are presented abstractly and combined with the motion equations. The two main numerical approaches for integrating those unilaterally constrained equations of motion in order to compute new velocities and position of the system across time, are presented as well;

- Section 1.3 discusses the first choice that must be made before designing our CD framework: the choice of a geometric contact model used to define the geometric information that must be computed by the CD framework to provide enough information to build the contact kinematics needed by the envisioned mechanical contact models (e.g. Signorini [122], Signorini-Coulomb, contact with rolling resistance, etc.) The distance-based and penetration-based geometric contact models are presented and their respective advantages and limitations are highlighted;

- Section 1.4 discusses the second choice that must be made before designing our CD framework: the choice of a geometric representation of the shapes of the bodies. Discrete-volume, polyhedral, and smooth boundary representations (BReps) are discussed and compared;

- Finally, Section 1.5 justifies our choices: a distance-based geometric contact model combined with smooth BReps. The general algorithmic schemes and existing methods for distance computation between two smooth trimmed surfaces or curves are presented as well.

1.2 Equations of motion and definition of contact constraints

This section provides an overview of the development of the equations of motion and non-penetration constraints governing multibody dynamics systems. Reference books [105, 44, 4, 129] give in-depth treatments of this topic. Here, only the mandatory subsets required to introduce the context where the proposed CD framework will be developed, are detailed:

- The equations of motion derived from the Lagrangian formalism are presented in Section 1.2.1. Perfect unilateral constraints that prevent objects from penetrating are defined abstractly and their addition to the equations of motion highlighted;

- Methods for integrating the constrained equations of motion are introduced in Section 1.2.2 with a reference of the fact that the proposed work is integrated into a time-stepping integration scheme;
- And Section 1.2.3 discusses the extension of motion equations presented so far to the case where some of the solids are deformable.

1.2.1 Lagrangian formalism for constrained equations of motion

Let us consider a mechanical assembly as a multibody system "MS" containing a set of N bodies B_k, $k \in \{1, \ldots, N\}$. The work of this thesis falls within the scope of nonsmooth dynamics simulations, i.e., the dynamic simulation of solids subjected to constraints that generate discontinuous jumps of the bodies velocities.

In a first place, let us assume the B_k are rigid solids with motions subjected to kinematic constraints. Those kinematic constraints interact with MS in such a way that the overall system has n degrees of freedom. The spatial configuration of MS can be described by a set of generalized coordinates $\mathbf{q}(t) \in \mathbb{R}^n$. Following the developments and notations from Acary et al. [4], the equations of motion governing MS can be obtained from Lagrange's equations:

$$\frac{d}{dt}\left(\frac{\partial L(\mathbf{q}(t), \mathbf{v}(t))}{\partial \mathbf{v}_i}\right) - \frac{\partial L(\mathbf{q}(t), \mathbf{v}(t))}{\partial \mathbf{q}_i} = Q_i(\mathbf{q}(t), t), \qquad i \in [1, n], \tag{1.1}$$

where $\mathbf{v}(t) = \dot{\mathbf{q}}(t)$ are the derivative of the generalized coordinates with respect to time, i.e., they are the generalized velocities. The quantity Q_i designates the generalized forces applied to the the body B_k. The Lagrangian L is obtained as the difference of the kinetic energy T and potential energy V of the system, i.e., $L(q, v) = T(q, v) - V(q)$ where V is provided as an additional input to the system. Given M, the inertia matrix of the system, the kinetic energy is defined as:

$$T(q, v) = \frac{1}{2}\mathbf{v}^t M(\mathbf{q})\mathbf{v}. \tag{1.2}$$

Developing Equation (1.1) yields the usual equations of motion:

$$M(\mathbf{q}(t))\frac{d\mathbf{v}}{dt}(t) = Q(\mathbf{q}(t), t) - \nabla V(\mathbf{q}(t)) - N(\mathbf{q}(t), \mathbf{v}(t)), \tag{1.3}$$

where N denotes the gyroscopic accelerations.

Equation (1.3) does not account for contacts between the B_k and thus, may lead to values of $\mathbf{q}(t) \in \mathbb{R}^n$ corresponding to geometric configurations of MS that are not feasible due to non-interpenetration constraints. In particular, some of them correspond to configurations where the space occupied by some B_k overlap with others, i.e., $\exists B_i, B_j / B_i \cap^* {}^4 B_j \neq \emptyset$. As a consequence, Equation (1.3) is augmented with perfect unilateral constraints called *non-penetration constraints*[5] to avoid the generation of configurations where penetrations occur. Those constraints are expressed through m inequalities:

$$g_i(\mathbf{q}) \geq 0, \qquad i \in [1, m], \tag{1.4}$$

where each $g_i : \mathbb{R}^n \to \mathbb{R}$ is referred to as a *gap function*. The subset of $\mathbb{R}^n$ where $\mathbf{q}$ may evolve without generating any penetration is called the *feasible space* $\mathcal{C}$:

$$\mathcal{C} = \{\mathbf{q} \in \mathbb{R}^n \mid g_i(\mathbf{q}) \geq 0, i \in [1, m]\}. \tag{1.5}$$

[4] $\cap^*$ designates the regularized Boolean operator of intersection.

[5] Note that we are only interested in (time-independent) non-penetration constraints in this manuscript. Typical dynamics simulations also include bilateral constraints to model, e.g., joints between two B_k, and other unilateral constraints to model, e.g., joint limits. All those are not included in our derivations though they could be added without affecting choices made for the design of our CD framework.

These constraints can be taken into account by the equations of motion Equation (1.3) with the addition of Lagrange multipliers $\lambda \in \mathbb{R}^n$:

$$\begin{cases} M(\mathbf{q}(t))\frac{d\mathbf{v}}{dt}(t) = Q(\mathbf{q}(t),t) - \nabla V(\mathbf{q}(t)) - N(\mathbf{q}(t),\mathbf{v}(t)) + \sum_{i=0}^{m} \nabla g_i(\mathbf{q})\lambda_i, \\ 0 \leq g_i(\mathbf{q}) \perp \lambda_i \geq 0, i \in [1,m], \end{cases} \tag{1.6}$$

where the last complementarity condition $0 \leq g_i(\mathbf{q}) \perp \lambda_i \geq 0$ is the Signorini condition [122] that forces both λ_i and g_i to remain positive and to be such that only one is non-zero at a time, i.e., $\lambda_i g_i(\mathbf{q}) = 0$. Moreover, whenever a gap function g_i originates from the geometric definition of a punctual contact (between two solids) its gradient ∇g_i (seen here as a column vector) is intrinsically linked with the geometrical notion of contact normal as discussed in Section 1.2.4.

Considering only two bodies B_i and B_j, let $\mathcal{A}(\mathbf{q}) \subset \mathbb{R}^3$ and $\mathcal{B}(\mathbf{q}) \subset \mathbb{R}^3$ be two compact sets expressing their respective shapes as well as their spatial location at the configuration $\mathbf{q}$. Within this manuscript the explicit mention of the parameter $\mathbf{q}$ is generally omitted for conciseness. Figure 1.1 shows an example of a simple planar rigid body system with $\mathcal{A}$ a mobile disk and $\mathcal{B}$ a static obstacle. Note that the boundary of $\mathcal{C}$ noted $\partial\mathcal{C}$ corresponds to the generalized coordinates for which $\mathcal{A}$ and $\mathcal{B}$ touch each other exactly (without penetration), i.e., their boundaries intersect but not their interiors. Some of those configurations are shown in Figure 1.2b using dotted lines.

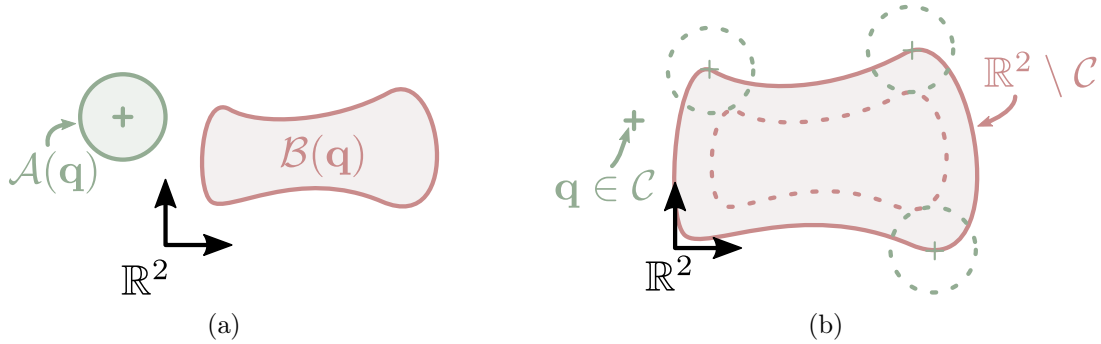


Figure 1.1: (a) A rigid body system including a disk $\mathcal{A}$ with two degrees of freedom and a non-convex solid $\mathcal{B}$ with zero degree of freedom. (b) The space of generalized coordinates $\mathbb{R}^2$ and the subset that is not feasible $\mathbb{R}^2 \setminus \mathcal{C}$ due to non-penetration constraints between the shapes of $\mathcal{A}$ and $\mathcal{B}$.

Figure 1.2 shows a similar example where the disk is constrained to rotate around a fixed point $\mathbf{p}$ in the plane. Thus, it has only one degree of freedom, which is the rotation angle around $\mathbf{p}$. Therefore, $\mathcal{C}$ is the set of rotation angles where the disk does not intersect the rectangle.

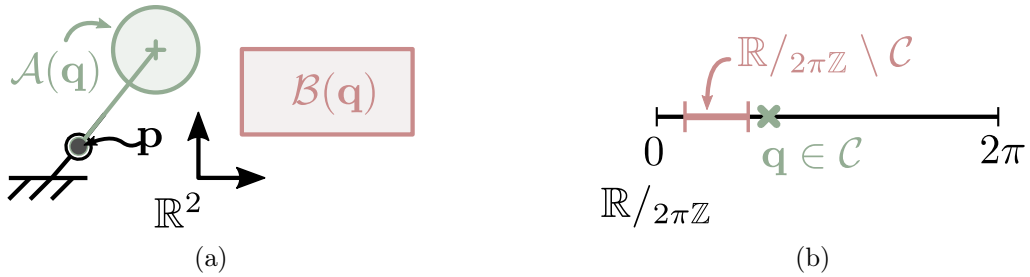


Figure 1.2: (a) A rigid body system including a 2D disk $\mathcal{A}$ with only one (rotational) degree of freedom because it is attached to the ground with a revolute joint. The rectangular obstacle is static. (b) The space of generalized coordinates is the interval I_q, $q \in [0, 2\pi]$ and non-penetration constraints between the disk and the rectangle removes a real interval from I_q to form $\mathcal{C}$.

Let us note that g_i being arbitrary functions, only one constraint is sufficient to describe $\mathcal{C}$ completely. However, such a constraint could contain singular points, i.e. points where a

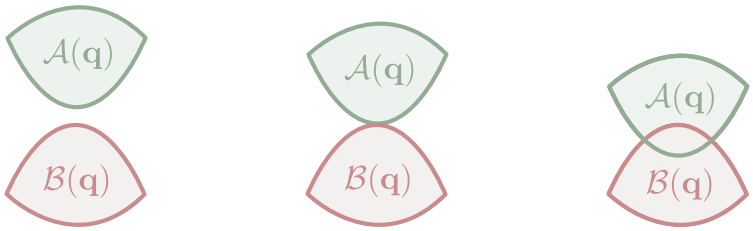
discontinuity of ∇g_i occurs, if $\partial\mathcal{C}$ has corners. Such singularities generally cause issues regarding the convergence of the constraint solver. Therefore, it is often preferable to rely on a set of g_i which are regular, and restrict our applications to configurations where only a finite number of g_i, defined on a domain $\mathcal{D}_i \subset \mathbb{R}^n$, is sufficient to represent $\mathcal{C}$. As discussed in [4], this is not too restrictive for most applications with practical interest[6]. Moreover, each g_i is restricted to describe a non-penetration constraint of only two B_k such that their values can be interpreted geometrically as follows:

$$g_i(\mathbf{q}) > 0 \Rightarrow \mathcal{A}(\mathbf{q}) \cap \mathcal{B}(\mathbf{q}) = \emptyset, \tag{1.7}$$

$$g_i(\mathbf{q}) = 0 \Rightarrow \begin{cases} \mathrm{Int}(\mathcal{A}(\mathbf{q})) \cap \mathrm{Int}(\mathcal{B}(\mathbf{q})) = \emptyset, \\ \partial\mathcal{A}(\mathbf{q}) \cap \partial\mathcal{B}(\mathbf{q}) \neq \emptyset, \end{cases} \tag{1.8}$$

$$g_i(\mathbf{q}) < 0 \Rightarrow \mathrm{Int}(\mathcal{A}(\mathbf{q})) \cap \mathrm{Int}(\mathcal{B}(\mathbf{q})) \neq \emptyset, \tag{1.9}$$

where $\partial\mathcal{A}$ and $\partial\mathcal{B}$ designate the boundary of $\mathcal{A}$ and $\mathcal{B}$, respectively. The operator Int computes the topological interior of a set, e.g., $\mathrm{Int}(\mathcal{A}) = \mathcal{A} \setminus \partial\mathcal{A}$. Those three configurations are illustrated in Figure 1.3. Note that multiple g_i can be assigned to the same pair of solids when any of them is not convex.



(a) $g_i(\mathbf{q}) > 0$ (b) $g_i(\mathbf{q}) = 0$ (c) $g_i(\mathbf{q}) < 0$

Figure 1.3: Gap function g_i between two solids $\mathcal{A}$ and $\mathcal{B}$ that are (a) separated, (b) touching, or (c) penetrating.

Typical applications simulate *MS* with $N > 2$. In that case, all the computations are performed pairwise and combined by collecting all the resulting non-penetration constraints using Equation (1.5). Consequently, all the concepts and algorithms described throughout this manuscript apply to a pair of sets $\mathcal{A}$ and $\mathcal{B}$ only, since it is the core configuration of any simulation of multibody systems. Moreover, in practice, the g_i can take different forms depending on their chosen *geometric contact model* as detailed in Section 1.3. We will go back to more details about gap functions and contact kinematics in Section 1.2.4.

1.2.2 Integration schemes of the equation of motion

Given an initial configuration of *MS* at the time $t_0 \in \mathbb{R}$, the numerical computation of an approximation the velocities and positions of each B_k at the time $t^* > t_0$, taking into account the constrained equations of motion described by Equation (1.6), is the task of a so-called *integration scheme*, generally taking one of two forms:

1. Starting at the time t_0, an *event-driven* integration scheme first detects (on the basis of a smooth ODE integration trajectory estimate) the next time t_1 where a nonsmooth evolution of the accelerations or velocities should occur, e.g., because of a closing contact. Those occurrences are called *events*. Then, the smooth equations of motion (see Equation (1.1)) are integrated on the interval $[t_0, t_1]$ using an ordinary differential equation (ODE) or differential algebraic equation (DAE) solver (refer to [55, 56] for in-depth treatments of the

[6] The simulation of a cylinder with its circular base resting on a table is a counter-example.

resolution each of ODEs and DAEs). Then, all the nonsmooth velocity changes at the time t_1 are determined using dedicated equations, yielding new values of the velocities of each B_k. Then, this procedure is repeated until the simulation is advanced to the time t^*. This iterative procedure is illustrated by Figure 1.4a;

2. Starting at the time t_0, a *time-stepping* integration scheme computes the evolution of the positions and velocities of each B_k using a subdivision of the interval $[t, t^*]$ into several sub-intervals. For simplicity, let us assume that every sub-interval have an equal size Δt. Each times $t_i = t + i\Delta t$ is referred to as a *time-step*. At each successive time-step, the equations of motion together with the smooth functions modeling the constraints generating nonsmooth velocity changes (see Equation (1.6)) are discretized on the time interval $[t_i, t_{i+1}]$. This can usually be formulated as a Nonlinear Complementarity problem (NCP) or a Linear Complementarity Problem (LCP) (see [4] for definitions), which are solved to determine new velocities and positions. This process is illustrated in Figure 1.4b.

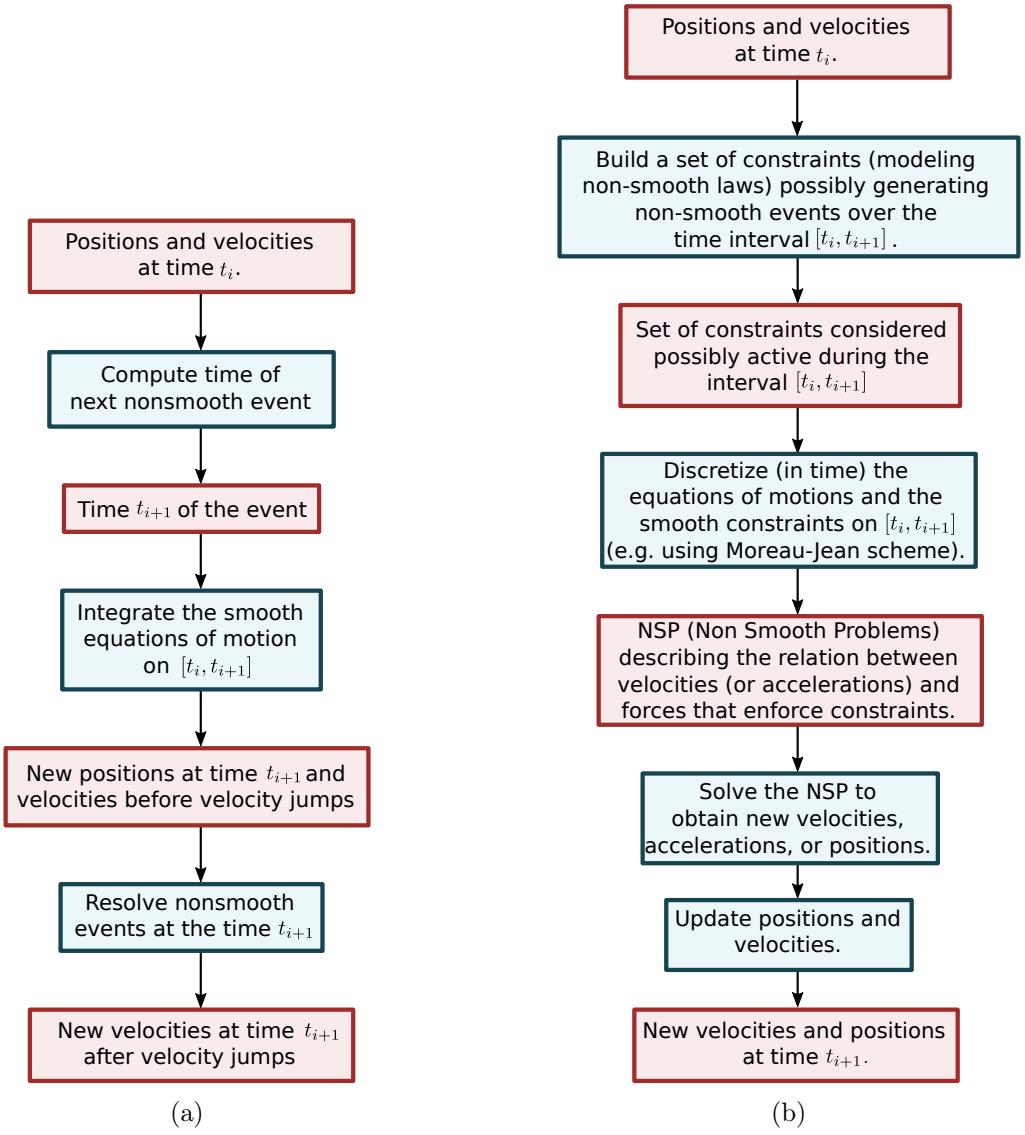


Figure 1.4: (a) One iteration of an event-driven integration scheme. (b) One iteration of a time-stepping integration scheme. Computations are shown in blue, and their outputs in red.

Further details regarding each integration scheme can be found in [4, 129].

The generation of contact information is the typical task of a CD engine. This can take different forms depending on the chosen integration scheme. On the one hand, event-driven integration schemes require the computation of the exact time t_i where any two B_i, B_k touch each other. On the other hand, time-stepping integration schemes require the definition of the constraints as smooth functions so that they can be discretized.

For the remainder of this manuscript, a time-stepping integration scheme is assumed. Thus, all the geometric computations described on the subsequent chapters are dedicated to computing all the informations necessary for the definition of the smooth contact constraints defined in Section 1.2.

1.2.3 Handling deformable curves

It has to be observed that the equations presented so far characterize the dynamics simulation of a mechanical system composed of rigid solids. In this manuscript, deformable bodies geometrically modeled as deformable Bézier curves are considered as well (refer to Section 1.3.1.2 for the definition of dilation). Following Acary et al. [4], deformable solids spatially discretized, e.g., using a finite element method, can be handled with the same equations as Equation (1.10), except that extra terms appear in the motion equations to account for forces due to deformations:

$$\begin{cases} M(\mathbf{q}(t))\frac{d\mathbf{v}}{dt}(t) = F_{int}(t,\mathbf{q}(t),\mathbf{v}(t)) - F_{ext}(t) - N(\mathbf{q}(t),\mathbf{v}(t)) + \sum_{i=0}^{m} \nabla g_i(\mathbf{q})\lambda_i, \\ 0 \leq g_i(\mathbf{q}) \perp \lambda_i \geq 0, i \in [1,m], \end{cases} \tag{1.10}$$

where F_{int} designates nonlinear internal forces, e.g., elasticity, and F_{ext} external forces, e.g., gravity. Overall, contact constraints for deformable curves can rely on the same geometric contact model as presented in Section 1.3 for rigid bodies.

1.2.4 Why the continuity of contact normals is desirable

The combination of contact constraints with the equations of motion given by Equations (1.6) and (1.10) involve the gradients ∇g_i of the gap functions g_i. If a g_i originates from the geometric definition of a punctual contacts between two solids $\mathcal{A}$ and $\mathcal{B}$, then its definition is given by a *geometric contact model* (refer to Section 1.3 for further details). Roughly speaking, g_i is expected to behave like some sort of signed distance function between $\mathcal{A}$ and $\mathcal{B}$ and must be continuous in order to have a meaningful physical interpretation. Moreover, its gradient should be continuous as required by time-stepping integration schemes (at least by Moreau-Jean's scheme presented in Figure 1.4b and its variants), even in the simplest case of perfect unilateral non-penetration constraints like the Signorini condition [122].

Now assume that the mechanical system MS is composed of two rigid bodies with strictly convex shapes $\mathcal{A}(\mathbf{q}), \mathcal{B}(\mathbf{q})$. And let g_i measure the smallest distance between $\mathcal{A}(\mathbf{q})$ and $\mathcal{B}(\mathbf{q})$ as shown in Figure 1.5.
Simple kinematic derivations (see [4]) yield:

$$\dot{g}_i(\mathbf{q}) = -\left\langle \mathbf{v}^i_{\mathcal{A}/\mathcal{B}}(\mathbf{q}), \mathbf{n}^i_{\mathcal{A}}(\mathbf{q}) \right\rangle, \tag{1.11}$$

where $\mathbf{v}^i_{\mathcal{A}/\mathcal{B}}$ refers to the velocity of the point $\mathbf{p}^i_{\mathcal{A}}$ seen instantaneously as belonging to $\mathcal{A}(\mathbf{q})$ moving with respect to $\mathcal{B}(\mathbf{q})$. The vector $\mathbf{n}^i_{\mathcal{A}}$ is the normal of $\mathcal{A}(\mathbf{q})$ at $\mathbf{p}^i_{\mathcal{A}}$. Note that because $\mathcal{A}$ and $\mathcal{B}$ are assumed rigid, we could equivalently write (noting $\mathbf{v}^i_{\mathcal{B}/\mathcal{A}}$ the velocity of $\mathbf{p}^i_{\mathcal{B}}$ seen instantaneously as belonging to $\mathcal{B}(\mathbf{q})$ moving with respect to $\mathcal{A}(\mathbf{q})$):

$$\dot{g}_i(\mathbf{q}) = -\left\langle \mathbf{v}^i_{\mathcal{B}/\mathcal{A}}(\mathbf{q}), \mathbf{n}^i_{\mathcal{B}}(\mathbf{q}) \right\rangle, \tag{1.12}$$

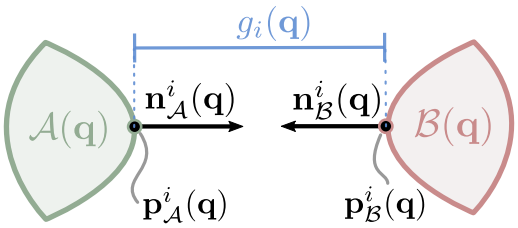


Figure 1.5: Two solids $\mathcal{A}$ and $\mathcal{B}$ with closest points $\mathbf{p}^i_{\mathcal{A}}$ and $\mathbf{p}^i_{\mathcal{B}}$ separated by a distance g_i. The vectors $\mathbf{n}^i_{\mathcal{A}}$ (resp. $\mathbf{n}^i_{\mathcal{B}}$ design the normals of $\mathcal{A}$ (resp. $\mathcal{B}$) at $\mathbf{p}^i_{\mathcal{A}}$ (resp. $\mathbf{p}^i_{\mathcal{B}}$)

by equiprojectivity of the point velocity field. Our following derivations refer to Equation (1.11) only. We assume that the equations of rigid body kinematics provide a Jacobian $\mathbf{J} \in \mathbb{R}^{3\times n}$ (with n the number of degree of freedoms of MS) that relate the spatial velocity $\mathbf{v}^i_{\mathcal{A}/\mathcal{B}}$ with the generalized velocities $\dot{\mathbf{q}}$:

$$\mathbf{v}^i_{\mathcal{A}/\mathcal{B}}(\mathbf{q}) = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}. \tag{1.13}$$

Here, $\mathbf{J}$ is generally smooth (C^∞). Combining Equation (1.11) with Equation (1.14) gives:

$$\nabla g_i(\mathbf{q}) = -\mathbf{J}(\mathbf{q})^T \mathbf{n}^i_{\mathcal{A}}(\mathbf{q}) \quad \text{(Recall that } \nabla g_i \text{ is a column vector of size } n\text{).} \tag{1.14}$$

As a consequence, $g_i(\mathbf{q})$ generally has the same regularity as $\mathbf{n}^i_{\mathcal{A}}$. This motivates the desire of having continuous contact normals with respect to the generalized coordinates if the constrained equations of motion are being integrated by a time-stepping scheme. Referring to the contact kinematic equations derived by Montana [93] and generalized by Visser et al. [139] to the case where $\mathcal{A}(\mathbf{q})$ and $\mathcal{B}(\mathbf{q})$ are not touching (i.e. $g_i(\mathbf{q}) \neq 0$) provides an in-depth derivation of the relation between g_i and $\mathbf{n}^i_{\mathcal{A}}$.

Furthermore, note that the simulation of physical phenomena like sliding friction or rolling resistance also require the $\mathbf{n}^i_{\mathcal{A}}$ and $\mathbf{n}^i_{\mathcal{B}}$ to be continuous with respect to the generalized coordinates. Indeed, friction models like the Signorini-Coulomb law rely of the computation of tangential sliding velocities. Assuming $(\mathbf{n}^i_{\mathcal{A}}, \mathbf{t}^{i,1}_{\mathcal{A}}, \mathbf{t}^{i,2}_{\mathcal{A}})$ forms an orthonormal basis where $\mathbf{t}^{i,1}_{\mathcal{A}}, \mathbf{t}^{i,2}_{\mathcal{A}}$ lie on the tangent plane of $\mathcal{A}$ at $\mathbf{p}^i_{\mathcal{A}}$, we have the tangential sliding velocity noted $\mathbf{v}^i_T \in \mathbb{R}^2$:

$$\mathbf{v}^i_T(\mathbf{q}) = \begin{bmatrix} \left\langle \mathbf{v}^i_{\mathcal{A}/\mathcal{B}}(\mathbf{q}), \mathbf{t}^{i,1}_{\mathcal{A}}(\mathbf{q}) \right\rangle \\ \left\langle \mathbf{v}^i_{\mathcal{A}/\mathcal{B}}(\mathbf{q}), \mathbf{t}^{i,2}_{\mathcal{A}}(\mathbf{q}) \right\rangle \end{bmatrix}, \tag{1.15}$$

$$= \begin{bmatrix} \mathbf{t}^{i,1}_{\mathcal{A}}(\mathbf{q})^T \\ \mathbf{t}^{i,2}_{\mathcal{A}}(\mathbf{q})^T \end{bmatrix} \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}. \tag{1.16}$$

Clearly, discontinuity of $\mathbf{n}^i_{\mathcal{A}}$ cause the discontinuity of $\mathbf{t}^{i,1}_{\mathcal{A}}$ and $\mathbf{t}^{i,2}_{\mathcal{A}}$. Therefore, if $\mathbf{n}^i_{\mathcal{A}}$ is discontinuous, $\mathbf{v}^i_T(\mathbf{q})$ is discontinuous as well. Such a discontinuity is likely to imped the convergence of non-linear, nonsmooth contact constraint methods (referring for example to the methods from Acary et al. [4]).

As a conclusion, it is clear that continuity of the contact normal at each contact point is a desired property for the robust integration of the equations of motion subjected to contact constraints. Note that, among the developments given in this section, the work presented in this manuscript is focused only on the design of a CD framework generating of contact informations with continuous contact normals. This requirement motivates our choice to combine distance-based geometric contact model presented in Section 1.3 with smooth representations of the geometrical shapes in contact presented in Section 1.4.3.

1.3 Choice of a geometric contact model

In practice, the exact nature of the g_i depends on the chosen *geometric contact model*, i.e., the geometric informations required by the dynamics solver in order to define non-penetration constraints, which will also be referred to as *contact constraints*. The two main models available display completely different features and drawbacks:

- *Distance-based* geometric contact models detailed in Section 1.3.1 rely on distance functions. While they have well-defined formulations, they fail to cover the case where g_i is negative because of the inherent positivity of distance functions;
- *Penetration-based* geometric contact models specifically require the objects to inter-penetrate in order to generate contact constraints. This condition enables algorithms to use some geometric representations with fast, but generally low accuracy, collision detection algorithms (see Section 1.4.1). However, these models often suffer from configurations where the gap function or its gradient becomes either discontinuous or is not well-defined. This approach and its limitations are detailed in Section 1.3.2

Note that we exclude here asymmetric contact models, e.g., point-versus-surface contact models [88, 13].

1.3.1 Distance-based geometric contact models

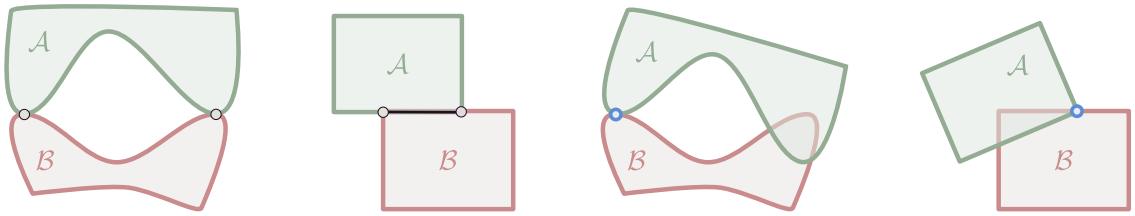
The main idea behind a distance-based geometric contact model is to predict future contacts, i.e., it outputs geometric informations to the dynamics solver so that contact constraints may be generated for actual contacts (where the objects actually touch) as well as for potential contacts (where the objects might touch in the near future). Moreover, potential contacts with contact points that are too distant, i.e., separated by a distance greater than some user-defined limit d_{max} (chosen such that no point of the B_k of MS is fast enough to travel this distance in one time-step), are ignored since they have no chance of becoming active during a time-step. In the remaining of this manuscript, potential contacts and actual contacts are both referred to as contacts.

A natural candidate solution to gap functions g is to use a notion of distance between $\mathcal{A}$ and $\mathcal{B}$. For example, a pair of points $\mathbf{p}_\mathcal{A} \in \mathcal{A}$ and $\mathbf{p}_\mathcal{B} \in \mathcal{B}$ such that their Euclidean distance $d = \|\mathbf{p}_\mathcal{A} - \mathbf{p}_\mathcal{B}\|$ is minimal, are the closest points between $\mathcal{A}$ and $\mathcal{B}$. Then, d_G is the *global minimal distance* and can be seen as a function of the generalized coordinates, i.e., $d_G(\mathbf{q}) : \mathbb{R}^n \to \mathbb{R}^+$. The gap function can then be defined as:

$$g(\mathbf{q}) = d_G(\mathbf{q}), \qquad \mathbf{q} \in \mathbb{R}^n. \tag{1.17}$$

While simple, this formulation has two major flaws:

1. It is hardly usable as part of a nonsmooth contact dynamics solver based on a time-stepping scheme. Indeed, whenever $\mathcal{A}$ and/or $\mathcal{B}$ is non-convex, a single such constraint fails to capture non-punctual contacts configurations like conformal contacts, i.e., when the contact is either a curve or a surface, and configurations with multiple contact points. Both scenarios are shown on Figure 1.6;
2. It cannot reach negative values. Therefore, it does not distinguish between contacts (Equation (1.8)) and penetrations (Equation (1.9)). Indeed, d_G is identically zero in $\mathbb{R}^n \setminus \mathcal{C}$ and in $\mathcal{C}$ since in both sets $\mathcal{A} \cap \mathcal{B} \neq \emptyset$ and the closest points can be chosen such that $\mathbf{p}_\mathcal{A} = \mathbf{p}_\mathcal{B} \in \mathcal{A} \cap \mathcal{B}$.



(a) (b) (c) (d)

Figure 1.6: (a) Configuration with multiple punctual contact points. (b) Configuration with a contact area that is not a single point. For both configurations, if only one pair of contact points is chosen at a time (blue), the dynamics solver will generate unrealistic rotations with respect to this point, generating the penetrations shown in (c,d)

The next sections present alternatives addressing those two issues. Section 1.3.1.1 describes the concept of Local Minimal Distances (LMD) targeted to handle multiple punctual contacts. Existing solutions for conformal contacts are described for the case where $\mathcal{A}$ and $\mathcal{B}$ are modeled as polyhedra. Section 1.3.1.2 modifies the definition of gap functions aforementioned with the introduction of an implicit dilation of the shape of B_k. Section 1.3.1.3 summarizes the methods that can be retained for the framework described in the subsequent chapters and provides some useful characterizations of LMDs.

1.3.1.1 Handling either multiple or conformal contacts

The first issue can be partially addressed with the generation of multiple gap functions for a single pair of B_k [90]. Instead of using only the global minimum of the Euclidean distance function, all the local minima called *local minimal distances* (LMD) are collected. For example in Figure 1.6a, the Euclidean distance function between $\mathcal{A}$ and $\mathcal{B}$ has two local minima. Thus, there exists two neighborhoods of $\mathbf{q}$ noted $\mathcal{D}_i \subset \mathbb{R}^n, i \in \{1,2\}$ on which two smooth functions $d_i : \mathcal{D}_i \to \mathbb{R}^+, i \in \{1,2\}$ that track those LMDs and thus, implicitly the two pairs of closest points (referred to as *LMD footpoints*), can be defined. Those two functions are used simultaneously as gap functions for this pair of rigid bodies. Therefore, instead of using one gap function associated with the global minimal distance between $\mathcal{A}$ and $\mathcal{B}$ as in Equation (1.17), multiple gap functions are defined:

$$g_i(\mathbf{q}) = d_i(\mathbf{q}), \qquad \mathbf{q} \in \mathcal{D}_i, \tag{1.18}$$

and combined using Equation (1.5) to delimit $\mathcal{C}$.

While using LMDs has the advantage of dealing with configurations where the set of contacts is composed of isolated points as in Figure 1.6a, this still does not handle conformal contacts like in Figure 1.6b. Several approaches with different levels of accuracy are possible, depending on the simulation. On the one hand, if $\mathcal{A}$ and $\mathcal{B}$ are deformable and are modeled in such a way that they may actually generate non-punctual contacts, then a finite set of contact points is no longer sufficient since a non null interaction force at any isolated point of the contact area results in deformations. Consequently, the isolated contact point becomes a contact surface between $\mathcal{A}$ and $\mathcal{B}$. Alternative contact models are thus necessary to render those constraints accurately. On the other hand, if the objects being simulated are strictly rigid, then it is reasonable to assume the contact area can be represented by a finite set of points. Two situations may occur:

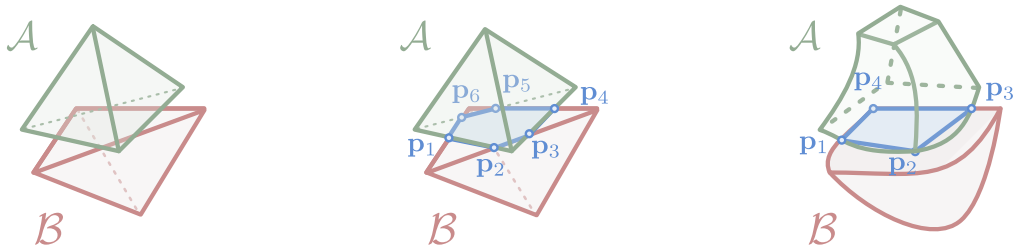
1. It may be possible to select a finite set of pairs of closest points in the contact area such that they bound $\mathcal{C}$ exactly, i.e., any additional constraint would be redundant as shown in Figure 1.7. This is the approach taken by Merlhiot [90] when solids are represented by

simplicial complexes [96], i.e., sets of points, segments, and triangles together with adjacency informations. The main idea is simple: the contact area being necessarily polyhedral, contact constraints are generated at each vertex of the contact surface (see Figure 1.8b).

This requires the definition of *quasi*-local minimal distances (quasi-LMD), which is presented at Section 1.3.1.4, in order to ensure that some LMD functions remain defined over the neighborhood of conformal contact configurations. Let us note that this approach is generally not sufficient when the geometric shapes are curved because the contact areas will be curved as well (see Figure 1.8c);

(a) (b)

Figure 1.7: Two scenarios with identical feasible spaces. (a) is composed of a capsule $\mathcal{A}$ and an infinite horizontal plane $\mathcal{B}$. (b) replaces the capsule by a pair of rigidly linked disks. In both scenarios, $\mathcal{A}$ has three degrees of freedom (two translations and one rotation) and $\mathcal{B}$ is static. In the two illustrated configurations, only a finite number of constraints (blue in (b)) is sufficient even if (a) has an infinite number of local minimizers of the distance function.



(a) (b) (c)

Figure 1.8: (a) A conformal contact between two tetrahedra. (b) The contact area is approximated by the six pairs of contact points $\mathbf{p}_i, i \in [1, 6]$ corresponding to isolated LMD between two edges or a vertex and an edge. (c) Configuration involving curved surfaces and curved edges. A finite set of contact points corresponding to the isolated LMDs is not sufficient as a small rotation, e.g., around the segment $\mathbf{p}_2\mathbf{p}_3$ would generate a penetration.

2. It may be impossible to determine a finite set of constraints that delimits $\mathcal{C}$ exactly or it may be too computationally intensive to do so. This is for example the case for a cylinder resting upright on a plane: the boundary of the conformal contact describes a circle that cannot be described by a finite number of contact points. In that case, the fact that manufactured rigid bodies never correspond exactly to the virtual models can be exploited (at least from a mathematical point of view). Indeed, manufacturing produces geometric errors that can lead to, e.g., misaligned axis of a hole with the axis of the shape being inserted, or lack of parallelism of some faces where conformal contacts should occur. In addition, manufactured surfaces are never perfectly smooth as shown is Figure 1.9. In both situations, the contacts between the real solids end up being actually punctual instead of conformal. Indeed, the actual locations of contact points are hardly predictable because their location depend on the real instance of a component. A naive approach is to sample the contact areas as proposed in the framework presented in this manuscript

(see Section 4.3.2) to partially address this issue when conformal contacts occur between two canonical surfaces or between an edge and a canonical surface. However, fast and accurate methods that apply to all configurations are yet to be found. More realistic simulations could be obtained with a more detailed study of the contact areas as performed by the tribology community [114]. Thus, alternative contact models, including statistical ones, between rough surfaces may be used [53]. Referring to Bhushan [17] gives access to a comparison of several rough contact models.

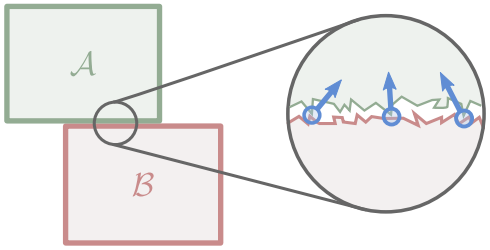


Figure 1.9: Zoom-in on what seems to be a conformal contact area between two rigid solids $\mathcal{A}$ and $\mathcal{B}$. Since the surfaces cannot be perfectly manufactured, asperities are present causing the actual contact to be localized at only a finite number of points (here, three) with distinct normals instead of one continuous contact surface.

Overall, it seems reasonable to state that whenever such a conformal contact case arises, the validity (from the point of view of mechanical modelling) of the approaches relying on the generation of distributions of punctual contacts should be questioned on a case-by-case basis.

1.3.1.2 Working around the non-negativity of distance functions

Distance functions as well as their global or local minima, are inherently positive. Therefore, no distinction between penetration and exact contact is possible since they both correspond to a distance equal to zero. This is problematic because a constraint solver typically needs to be able to explore numericaly configurations with negative gaps during its algorithmic work. Moreover, contact forces are generated when exact contacts occur in order to prevent penetrations at the next time-step, but some schemes or solvers may leave the system with residual penetrations after the current time step (constraints 'drifts'). In some applications, the presence of such penetrations requires additional *constraint stabilization terms*, i.e., terms that will push the objects out of the unrealistic configuration in which they currently are. Those constraint stabilization terms are often function of some measure of the penetration: the deeper the penetration, the stronger will be the stabilization terms. A common example of such stabilization is the Baumgarte Stabilization Method [14] and other examples can be found in [3, 21].

One could argue that the whole benefit of using distance-based methods is to avoid penetrations altogether. While theoretically true, the use of iterative methods for solving the equations of the dynamics system, the approximations performed by the differential equation integrators, and the limited accuracy of floating point operations, tend to generate deviations with respect to the theoretical solution that can be sufficiently large to output positions of the B_k with small inter-penetrations. Therefore, while the ability of handling large penetrations is not necessary, *small* ones (depending on the accuracy of the dynamics solver, the integration scheme, the time-step length, etc.) must be detected and stabilized.

A common approach for generating negative values using distance computations holds in the definition of the gap function g_i associated to the i-th LMD function d_i as:

$$g_i(\mathbf{q}) = d_i(\mathbf{q}) - 2\epsilon, \qquad \epsilon \in \mathbb{R}^{+*}, \tag{1.19}$$

where ϵ is a small positive value. A geometric interpretation of Equation (1.19) is to see each B_k shape as being *implicitly* dilated (see Figure 3.10) using a spherical envelope of radius ϵ. *Implicitly* means that the boundaries of B_k are not actually modified but 2ϵ is systematically subtracted from any distance computation result. This is equivalent to the definition of new boundaries of $\mathcal{A}$ and $\mathcal{B}$ such that they are obtained using envelopes of spheres whose centers lie on $\partial\mathcal{A}$ and $\partial\mathcal{B}$ as shown in Figure 3.10.

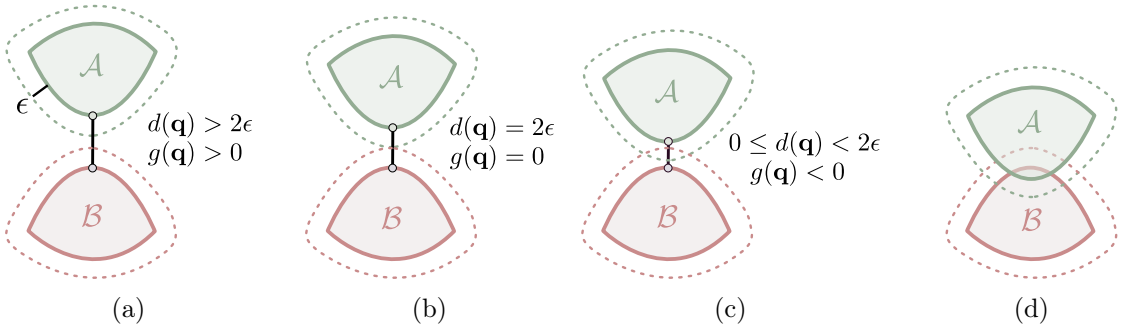


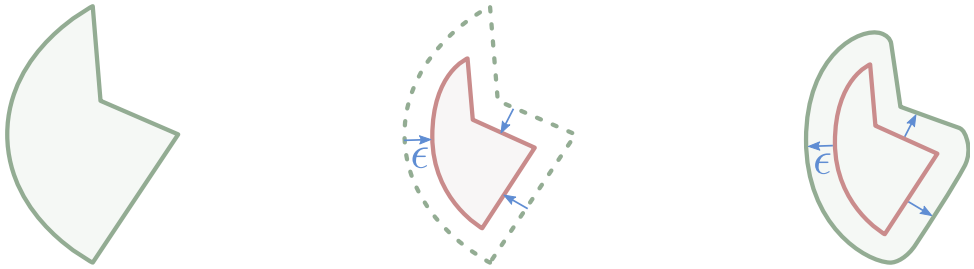
Figure 1.10: Solids $\mathcal{A}$ and $\mathcal{B}$ dilated by ϵ (dashed lines). (a) $d(\mathbf{q}) > 2\epsilon$: non-intersecting configuration. (b) $d(\mathbf{q}) = \epsilon$: touching configuration, i.e., the constraints solver will prevent them from getting closer because $g(\mathbf{q}) = 0$ even if the non-dilated shapes are not actually touching. (c) $d(\mathbf{q}) < 2\epsilon$: penetration configuration and $g(\mathbf{q}) < 0$. (d) The non-dilated shape are penetrating so the gap function is no longer defined.

This completes the definition of gap function that now tolerates negative values down to -2ϵ (see Figure 3.10c). The ϵ should be kept as small as possible because the constraints solver will not allow $\mathcal{A}$ and $\mathcal{B}$ to get closer than 2ϵ from each other (see Figure 3.10b). But it must be large enough to cope with non linear constraints discretization and computation roundoff. Its value is typically user-defined depending on the nature of the constraints solver and the time-stepping size, which usually affect the amplitude of the discretization errors generated.

The main drawback of this approach is that the simulated objects can appear to be 'floating', instead of actually touching each other, because they are always separated by a distance equal to 2ϵ, at least. Moreover, the simulation of insertions with mechanical clearances smaller than 2ϵ become impossible since the dilations would be larger than the gaps. Those issues can be addressed by a preliminary work of preparation of the models involved in the simulation. Such a preparation accounts for applying an erosion of ϵ to all BRep (such an operation is often readily available on CAD modelers). Therefore, the implicit dilation of ϵ will almost recover the original shape as shown in Figure 1.11. An unavoidable consequence is that convex sharp features (edges and vertices) are *rounded off* by the implicit dilation (see Figure 1.11c). However, this is usually an acceptable limitation in practice because sharp edges are generally avoided for mechanical parts where functional contacts occur since they can damage the parts or accelerate their wearing. In addition, different dilations could be applied to different solids. For example, if the simulation involves only two solids $\mathcal{A}$ and $\mathcal{B}$, an erosion and implicit dilation of 2ϵ could be applied to $\mathcal{A}$ while $\mathcal{B}$ is left unchanged. This can be useful if, e.g., this erosion/dilation changes the shape of $\mathcal{A}$ in a way that is less significant than modifying $\mathcal{B}$.

1.3.1.3 Characterization of Local Minimal Distances (LMD)

LMDs between two solids $\mathcal{A}$ and $\mathcal{B}$ can be characterized as in [90, 32]. Indeed the direction of the LMD, i.e., the line passing through both footpoints, appears 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 [15]. Let us recall some definitions from Bertsekas et al. [15] where $\mathbf{S}$ designates either $\mathcal{A}$ or $\mathcal{B}$. Given a point $\mathbf{p_S} \in \mathbf{S}$, which can be either in the interior or on the



(a) (b) (c)

Figure 1.11: (a) The original geometric shape of B_k. (b) The shape after an erosion of ϵ. (c) In green, the shape seen by the dynamics simulation because of the implicit dilation of radius ϵ. Note how most vertices have been rounded off by the dilation.

boundary of $\mathbf{S}$, its *tangents* are the vectors $\mathbf{y} \in \mathbb{R}^3$ such that there exists a sequence of points $\{\mathbf{p}^k\} \subset \mathbf{S}, \mathbf{p}^k \neq \mathbf{p_S}$, and:

$$\mathbf{p}^k \to \mathbf{p_S}, \qquad \frac{\mathbf{p}^k - \mathbf{p_S}}{\|\mathbf{p}^k - \mathbf{p_S}\|} \to \frac{\mathbf{y}}{\|\mathbf{y}\|}. \tag{1.20}$$

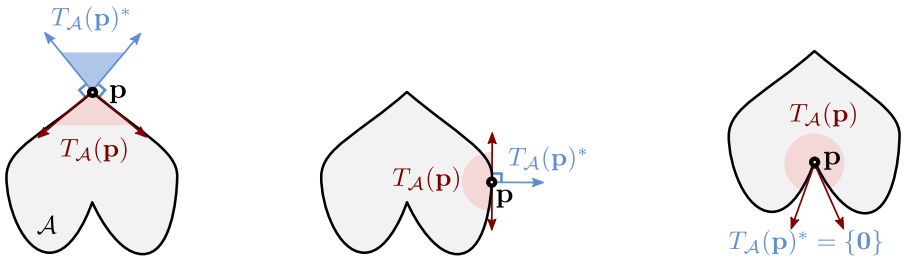
The set of all tangents at $\mathbf{p_S}$ forms a cone (in the sense of convex analysis) called *tangent cone* $T_\mathbf{S}(\mathbf{p_S})$. The *tangent cone polar*:

$$T_\mathbf{S}(\mathbf{p_S})^* = \left\{\mathbf{d} \in \mathbb{R}^3 \mid \forall \mathbf{v} \in T_\mathbf{S}(\mathbf{p_S}), \langle \mathbf{d}, \mathbf{v} \rangle \leq 0\right\}, \tag{1.21}$$

is the set of vectors opposite to all the tangents at $\mathbf{p_S}$. Some tangent cones and their polars are depicted in Figure 1.12 to illustrate common configurations. For sake of simplicity, a 2D domain is used rather than a 3D one. The tangent cone in Figure 1.12c is larger than a half-space, therefore its polar degenerates to the singleton $\{\mathbf{0}\}$.

Let us note that the tangent cone at a point of a C^1 surface (resp. C^1 curve) is a plane (resp. line). Consequently, vectors of their tangent cone polars are orthogonal to all vectors of their tangent cone:

$$T_\mathbf{S}(\mathbf{p_S})^* = \left\{\mathbf{d} \in \mathbb{R}^3 \mid \forall \mathbf{v} \in T_\mathbf{S}(\mathbf{p_S}), \langle \mathbf{d}, \mathbf{v} \rangle = 0\right\}. \tag{1.22}$$



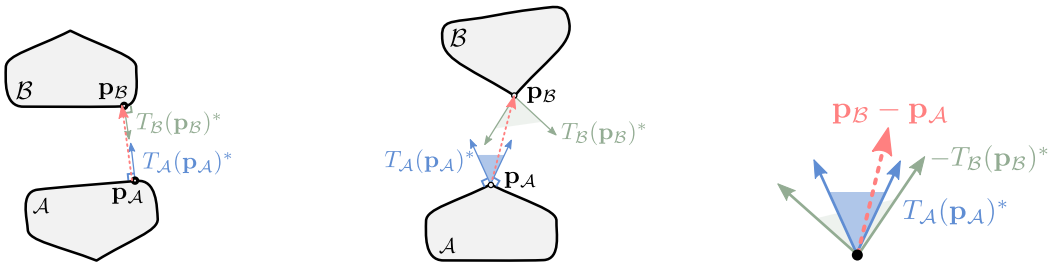
(a) (b) (c)

Figure 1.12: A tangent cone $T_\mathcal{A}(\mathbf{p})$ and its polar $T^*_\mathcal{A}(\mathbf{p})$ at a point $\mathbf{p}$ which is: (a) convex; (b) on an edge; (c) at a point such that $T^*_\mathcal{A}$ is degenerate.

The previous concepts produce the tools characterizing *critical points* of the squared distance function between $\mathcal{A}$ and $\mathcal{B}$. Indeed, two points $\mathbf{p}_\mathcal{A} \in \mathcal{A}$ and $\mathbf{p}_\mathcal{B} \in \mathcal{B}$ are critical points if and only if the following condition holds [32, 90]:

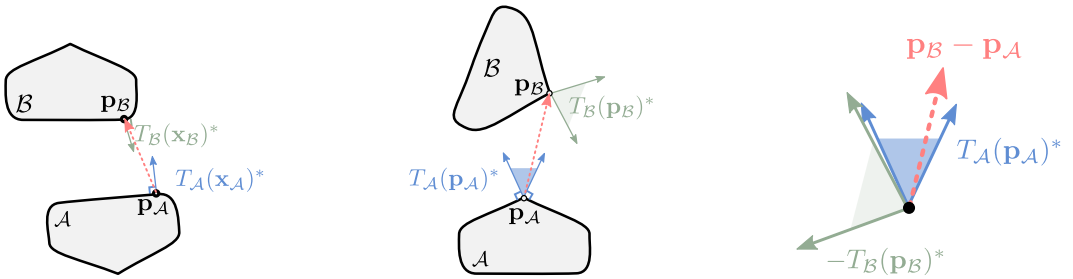
$$\mathbf{p}_\mathcal{B} - \mathbf{p}_\mathcal{A} \in T_\mathcal{A}(\mathbf{p}_\mathcal{A})^* \cap -T_\mathcal{B}(\mathbf{p}_\mathcal{B})^*. \tag{1.23}$$

Configurations where Equation (1.23) is satisfied are illustrated in Figure 1.13. Configurations where it is not satisfied are shown in Figure 1.14. Let us note that in this second case, the tangent cone polars $T_{\mathcal{A}}(\mathbf{p}_{\mathcal{A}})^*$ and $-T_{\mathcal{B}}(\mathbf{p}_{\mathcal{B}})^*$ do not even intersect (except at the null vector $\mathbf{0}$) so Equation (1.23) has no chance of being satisfied, no matter the direction of the line passing through $\mathbf{p}_{\mathcal{B}}$ and $\mathbf{p}_{\mathcal{A}}$.



(a) (b) (c)

Figure 1.13: Configurations where $\mathbf{p}_{\mathcal{A}}$ and $\mathbf{p}_{\mathcal{B}}$ are critical points. (a) The tangent cone polars are the normals of $\mathcal{A}$ and $\mathcal{B}$. They are collinear with the difference vector $(\mathbf{p}_{\mathcal{B}} - \mathbf{p}_{\mathcal{A}})$. (b) The tangent cone polars at vertices of $\mathcal{A}$ and $\mathcal{B}$ contain more than one direction. They are superimposed in (c) to highlight the fact that Equation (1.23) is satisfied.



(a) (b) (c)

Figure 1.14: Configurations where $\mathbf{p}_{\mathcal{A}}$ and $\mathbf{p}_{\mathcal{B}}$ are not critical points. (a) The tangent cone polars are the normals of the solids and they are not collinear with each other nor with $\mathbf{p}_{\mathcal{B}} - \mathbf{p}_{\mathcal{A}}$. (b) The tangent cone polars at vertices of the solid contain more than one direction. They are superimposed in (c) to highlight the fact that Equation (1.23) is not satisfied because $\mathbf{p}_{\mathcal{B}} - \mathbf{p}_{\mathcal{A}} \notin -T_{\mathcal{B}}(\mathbf{p}_{\mathcal{B}})^*$.

The goal of the proposed method being the computation of closest points, which are also critical points, the Equation (1.23) can be used to distinguish subsets of features that may or may not contain some closest points, as addressed in Chapter 3. More precisely, the Equation (1.23) is used only in Section 3.5 to design a tangent-based culling test on Bézier curves and in Section 4.4.2 for filtering potential LMDs that have been computed. The remaining culling tests based on tangent cones and tangent cone polars rely on a weaker condition, which is a direct consequence of Equation (1.23):

$$T_{\mathcal{A}}(\mathbf{p}_{\mathcal{A}})^* \cap -T_{\mathcal{B}}(\mathbf{p}_{\mathcal{B}})^* = \{\mathbf{0}\} \Rightarrow \mathbf{p}_{\mathcal{A}}, \mathbf{p}_{\mathcal{B}} \text{ are not critical points.} \tag{1.24}$$

Let us observe that this definition can cover non-isolated critical points, i.e., conformal contacts configurations, which are handled by the sampling method described in Section 4.3.2. Finally, Equation (1.23) has two immediate consequences regarding the location of the footpoints of a LMD (which must be critical points):

1. LMD footpoints cannot be located in $\mathrm{Int}(\mathcal{A})$ or $\mathrm{Int}(\mathcal{B})$ as they would have tangent cones polars that degenerate to $\{\mathbf{0}\}$. Thus, the problem reduces to finding contact points on $\partial\mathcal{A}$

and $\partial\mathcal{B}$ only, i.e., directly on their boundary representations without any consideration of their interiors because Equation (1.24) is guaranteed to be satisfied on interior points;

2. All edges exclusively formed of points with tangent cone polars containing exactly one non-zero direction could be ignored by the whole CD pipeline because their intersecting surfaces are sufficient to locate the same LMD footpoints. On a BRep model, those edges are referred to as G^1 *edges* in Section 1.4.3. Their systematic identification in order to ignore them for LMD computation is left to future works but some preliminary results are illustrated in Chapter 5.

1.3.1.4 Characterization of quasi-LMD

The goal of quasi-LMDs is to enlarge the domain of definition of the LMD functions d_i presented in Section 1.3.1.1. In other words, if $D_i \subset \mathbb{R}^n$ is the domain of definition of d_i, then a *quasi-LMD function* is a function $\tilde{d}_i$ with a domain of definition $\tilde{D}_i \supset D_i$ on which it is as smooth as d_i, and such that $\forall \mathbf{q} \in D_i, \tilde{d}_i(\mathbf{q}) = d_i(\mathbf{q})$. For reasons that will be clarified in the following developments, the process of extending d_i into $\tilde{d}_i$ is called the *angular regularization* of d_i. The quasi-LMD function $\tilde{d}_i$ is also called a *regularized LMD function*. Just like d_i, the quasi-LMD functions can be used to model gap functions either by taking $g_i(\mathbf{q}) = \tilde{d}_i(\mathbf{q})$ (similarly to Equation (1.18)), or by taking (similarly to Equation (1.19)):

$$g_i(\mathbf{q}) = \tilde{d}_i(\mathbf{q}) - 2\epsilon, \qquad \mathbf{q} \in \tilde{D}_i, \tag{1.25}$$

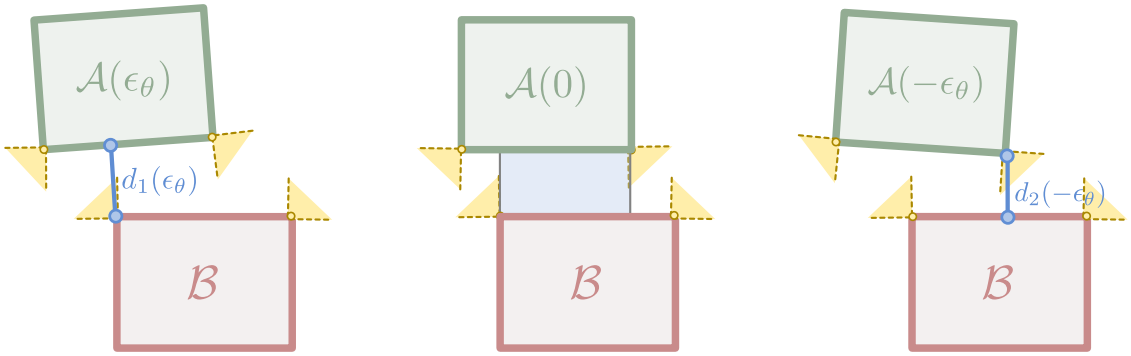
if the dilation introduced in Section 1.3.1.2 is to be taken in account.

In order to understand why D_i should be enlarged, Figure 1.15 shows a simple system composed of two bodies B_i, B_j, with respective geometric shapes $\mathcal{A}$ and $\mathcal{B}$. Here, B_i has one rotational degree of freedom, $\theta \in \mathbb{R}$, around its center of mass and B_j stands still. Figure 1.15b illustrates a conformal contact configuration with $\theta = 0$. Note that our use of *conformal contact* here refers to the area composed of non-strict minima of the distance function, i.e., even if there is no effective contact because $d_i(\mathbf{q}) \neq 0$. Figures 1.15a and 1.15c show the evolution of the LMD functions when $\mathcal{A}$ is subject to arbitrary small rotations ϵ_θ and $-\epsilon_\theta$. The tangent cone polars of some vertices of each rectangle is shown is yellow. Two major stability issues can be highlighted:

1. In the conformal configuration of Figure 1.15b, no LMD function is defined since there is no isolated minimum. Therefore, no constraint will be generated unless additional work is performed;

2. It could be considered that the configuration of Figure 1.15b never happens in practice because of the numerical errors introduced by the integration and the limited precision of floating point numbers. In that situation, the system ends up into either of the configurations shown on Figure 1.15a and Figure 1.15c. Let us take as reference configuration (a). If $\mathcal{A}$ and $\mathcal{B}$ come into contact in this configuration at a time-step t_i, then a small impulse applied to the points related to the LMD function d_1 will tip $\mathcal{A}$ in such a way that the configuration at the time-step t_{i+1} becomes configuration (b). For similar reasons, t_{i+2} brings the system back into the configuration (a). Repeated indefinitely, this results into an unstable behavior of $\mathcal{A}$ with respect to $\mathcal{B}$.

One way to avoid this is to refer to multiple distance functions defined simultaneously in all three configurations. The concept of quasi-LMD was introduced in [90] for the case of LMD computation between tessellated models. First of all, the polyhedral models are decomposed into a simplicial complex structure. Please, refer to Munkres [96] regarding details about this



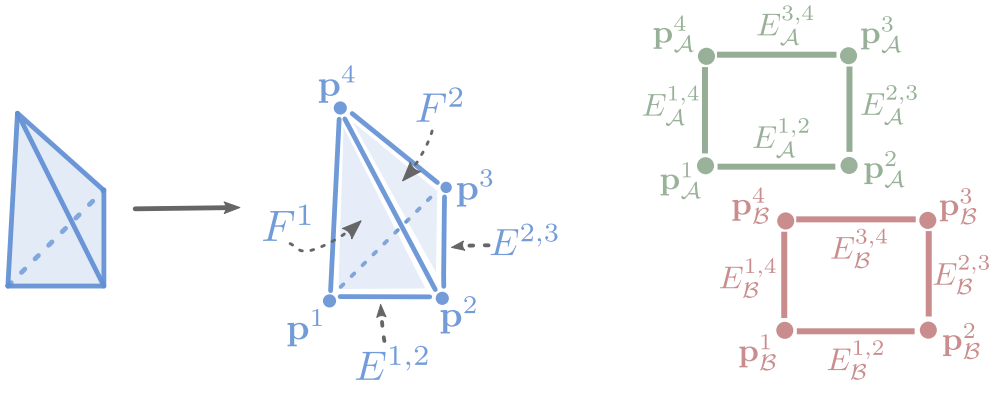
(a) (b) (c)

Figure 1.15: A static box $\mathcal{B}$ and a box $\mathcal{A}$ with one rotational degree of freedom. (a) $\mathcal{A}$ is sightly tilted by an angle ϵ_θ that can be chosen arbitrarily small. (b) Conformal contact configuration. (c) $\mathcal{A}$ is sightly tilted by an angle $-\epsilon_\theta$.

structure. Here, one of its main features only is of interest for the definition of a quasi-LMD. Let us describe this feature. A simplicial complex is composed of geometric linear elements of various dimensions subjected to convexity constraints. For example, a 3D simplicial complex is composed of vertices (0-dimensional, points), edges (1-dimensional, segments) and faces (2-dimensional, triangular areas) as shown in Figure 1.16. Those elements, are referred to as *features* in the remaining of this section and are related to each other by an inclusion relation where a n-dimensional entity contributes to the boundary of some entities of dimension m with $m > n$. In practice, this means that a polyhedral model can be described using a simplicial complex structure (see Figure 1.16a) by:

1. Creating one face per triangle;
2. Creating one edge per segment at the junction between each adjacent pair of triangles;
3. Creating one vertex at each common intersection point of several edges.

Note that this is an overly simplified presentation of the properties of a simplicial complex. A more complete and accurate description can be found in [90, 96]. Figure 1.16b shows the decomposition of the boundary of $\mathcal{A}$ and $\mathcal{B}$ (from Figure 1.15) into simplicial complex structures.



(a) (b)

Figure 1.16: (a) A tetrahedron and its boundary decomposition as a simplicial complex. Its vertices are identified as $\mathbf{p}^i$, its visible faces as F^i, and some of its edges as $E^{i,j}$ with i, j the indices of its adjascent faces. (b) The scenario of Figure 1.15b with the boxes shown as simplicial complexes.

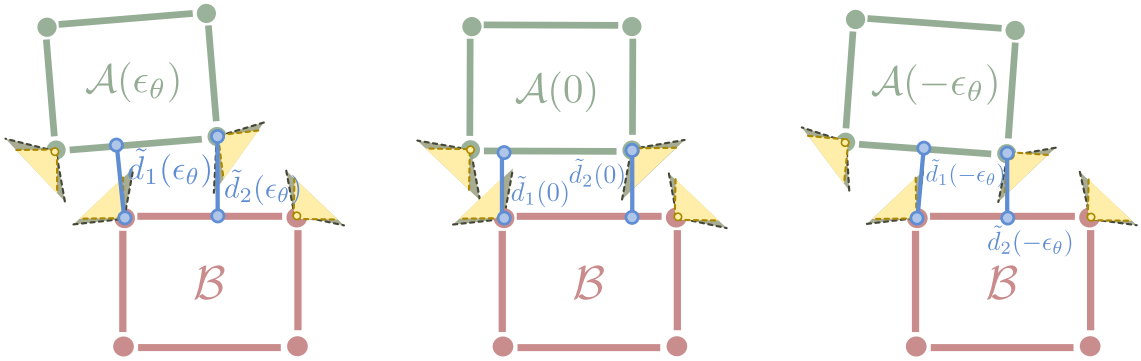
Once this decomposition is obtained, one distance function is assigned to each feature pair where one feature belongs to $\mathcal{A}$ and the other to $\mathcal{B}$. The restriction of those distance functions to the subsets of $\mathcal{C}$ such that Equation (1.23) is satisfied yields the LMD functions. Doing so has one immediate consequence for the scenario shown in Figure 1.15b. Indeed, if d_1 (resp. d_2) is the LMD functions between $E_{\mathcal{A}}^{1,2}$ and $\mathbf{p}_{\mathcal{B}}^4$ (resp. between $E_{\mathcal{B}}^{3,4}$ and $\mathbf{p}_{\mathcal{A}}^2$), then domains of definition of d_1 and d_2 both contain the element $\theta = 0$. Indeed, the Equation (1.23) holds for d_1 (resp. d_2), and $d_1(0)$ (resp. $d_2(0)$) properly represent isolated minima of the Euclidean distance between $\mathbf{p}_{\mathcal{B}}^4$ and $E_{\mathcal{A}}^{1,2}$ (resp. $\mathbf{p}_{\mathcal{A}}^2$ and $E_{\mathcal{B}}^{2,4}$). This modification however, does not address the fact that any arbitrary small rotation angle ϵ_θ applied to $\mathcal{A}$ will leave at least one of d_1 or d_2 undefined.

To avoid this issue, the vertices' tangent cone polars are dilated with a user-defined angle, as shown in Figure 1.17. The quasi-LMD functions $\tilde{d}_1, \tilde{d}_2$ are then defined as equal to d_1 and

(a) (b)

Figure 1.17: (a) A 2D tangent cone before (yellow) and after (green) dilation by an angle α. (b) Shape $\mathcal{A}$ from Figure 1.18 after decomposition into a simplicial complex, with all its vertices tangent cone polars (yellow) and their dilated versions (green).

d_2, except that they rely on those dilated tangent cone polars for the condition of criticality expressed by Equation (1.23). Overall, those two quasi-LMD functions remain defined on a neighborhood of the conformal configuration (see Figure 1.15b) as shown in Figure 1.18.



(a) (b) (c)

Figure 1.18: The regularized distance functions $\tilde{d}_1$ and $\tilde{d}_2$ remain defined over all three configurations. (a) The dilation of $T_{\mathcal{B}}(\mathbf{p}_{\mathcal{B}}^1)^*$ allowed $\tilde{d}_2$ to remain defined on ϵ_θ. (b) The decomposition into simplicial complex allowed $\tilde{d}_1$ and $\tilde{d}_2$ to be seen as LMD between isolated local minimizers of the distance function. (c) The dilation of $T_{\mathcal{A}}(\mathbf{p}_{\mathcal{A}}^2)^*$ allowed $\tilde{d}_1$ to remain defined on $-\epsilon_\theta$.

Overall, the quasi-LMD model has three requirements that constrain the geometric description of shapes as well as the conditions characterizing a critical point:

- The decomposition of the shapes boundaries into a simplicial complex structure;
- The dilation of all the tangent cone polars associated to each feature of the simplicial complex decomposition that have a dimension lower than two (or than one for 2D models),

as shown in Figure 1.17. The actual value of the angle is user-defined as it depends on the simulation. Typically, simulations involving larger angular velocities will require larger values;

- The definition of LMD functions for each feature pair. Their domain of definition is bounded by the positions for which the Equation (1.23) is satisfied now using the enlarged tangent cone polars instead of the original ones.

1.3.1.5 Summary and recommended choice of gap function

To summarize the analysis of the previous sections, the following combination of variants of the distance-based approach are recommended to achieve stable simulations:

- Use several LMDs (see Section 1.3.1.1) instead of one global minimal distance functions in order to properly handle multiple isolated contact points between non-convex shapes;
- Incorporate implicit dilations of the geometric models (see Section 1.3.1.2) in order to set up a definition of gap functions able to represent small penetrations to account for numerical roundoffs deriving from the various approximations performed by the dynamics solver;
- Apply angular regularization to the LMD functions (see Section 1.3.1.4) to ensure that all the gap functions are properly defined in a neighborhood of the conformal configurations occuring at the current time-step. As pointed out in Section 1.3.1.1, this is not sufficient to handle conformal contacts properly when the considered shapes are curved. However, this covers a range of configurations wide enough to be useful in practice, even with curved shapes. The issue of conformal contacts for curved shapes is further addressed in Section 4.3.2.

Overall, the gap functions considered by the mechanical simulation software targeted by our CD methods is given by Equation (1.25). Keeping this goal in mind is necessary as this specifies the required output of the proposed CD framework.

1.3.2 Penetration-based contact models

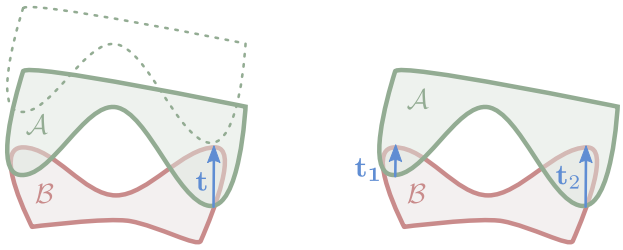
Penetration-based geometric contact models approach contact constraints from a point of view that is opposite to distance-based models. Indeed, instead of computing distances while the objects are disjoint, penetration-based approaches wait for the contacts to actually happen, i.e., it requires $\mathcal{A}$ and $\mathcal{B}$ to intersect in order to compute a measure of their penetration. Then, this measure is used as a gap function yielding negative values proportional to the amount of penetration between $\mathcal{A}$ and $\mathcal{B}$.

The most common penetration measure is the *penetration depth* (or *minimal translational distance*), which is the magnitude of the *minimal translational vector* [22] assigned to two intersecting objects. Assuming $\mathcal{A}$ and $\mathcal{B}$ are penetrating each other, the minimal translational vector is the vector representing the smallest translation needed to translate one of them, say, $\mathcal{A}$, such that: $\mathcal{A} \cap^* \mathcal{B} = \emptyset$[7]. Its direction is referred to as the *penetration direction*. Consequently, the exact computation of the penetration depth is a complex problem. For example, it has a time complexity of $O(m^3 n^3)$ between two polyhedral models where m and n are their respective number of triangles [35]. Efficient iterative methods [138] exist, especially for convex polyhedra.

To avoid the issue of multiple contact points mentioned in Section 1.3.1.1, a penetration depth can be computed locally [112, 72]. Those local measures rely on a partition of $\mathcal{A}$ and

[7] The operator $\cap^*$ stands for regularized Boolean intersection.

$\mathcal{B}$ into several connected regions where the penetration depth can be computed. This includes the convex decomposition of concave shapes with, for example, the method proposed by Lien et al. [83] or the use of the graphics hardware to render the 3D volumes prior to the search for the intersecting volumes [112]. Combining both was also attempted by Kim et al. [73]. The difference between the minimal translational distance and the local penetration depths is highlighted on Figure 1.19.



(a) (b)

Figure 1.19: (a) The minimal translational vector $\mathbf{t}$ between $\mathcal{A}$ and $\mathcal{B}$. As shown by dotted lines, translating $\mathcal{A}$ along $\mathbf{t}$ would position $\mathcal{A}$ and $\mathcal{B}$ just touching each other. (b) The local penetration vectors $\mathbf{t}_1$ and $\mathbf{t}_2$ deriving from the two overlapping regions of $\mathcal{A}$ and $\mathcal{B}$.

Penetration-based approaches have two major downsides:

1. The significant reliance on constraint stabilization. Indeed, penetration depths correspond to negative values of the gap functions and thus to measures of violation of the non-penetration constraints described in Section 1.2. Therefore, constraints stabilization like [14, 8, 56] has to be taken in account by the dynamics solver, as mentioned in Section 1.3.1.2. This causes several limitations.

   Firstly, if the amount of constraint violation is too large, then stabilization methods like [14] are likely to artificially introduce a sufficiently large kinetic energy into *MS* to disturb the behaviors of some B_i. Moreover, depending on the chosen stabilization method, the penetrating objects may be subjected to oscillations before converging to a stable position. In practice, those stability issues can be observed during simulations of component insertions, e.g., of $\mathcal{A}$ into $\mathcal{B}$, involving small mechanical clearances between them, which are the main applications targeted by the work presented in this manuscript.

   Secondly, some configurations similar to the ones shown on Figure 1.20 cannot be solved satisfyingly solely using the minimal translational distance or local penetration depths, due to the limited informations provided by the penetration depth. This is highlighted by Zhang et al. [143] and can be addressed by proposing an alternative penetration measure that includes rotational informations: the *generalized penetration depth* [143] shown in Figure 1.20c;

2. Whenever the medial axis of the penetrated objects intersect as shown at Figure 1.21, the contact kinematics equations are no longer well-defined as discussed by Pfeiffer and Glocker [105]. Indeed, crossing the medial axes may produce discontinuous changes of the penetration vector direction, causing the contact position and normals to be discontinuous with respect to the motion parameters because of the non uniqueness of the penetration vector direction (see Figure 1.21b) at the crossing point itself. Let us note that this issue occurs with distance-based approaches as well if one of the objects is concave as shown in Figure 1.21d. However, this is less of a concern since this does not cause uncertainties regarding the movement of $\mathcal{A}$, as long as the multiple potential contact points can be represented as footpoints of multiple LMDs.

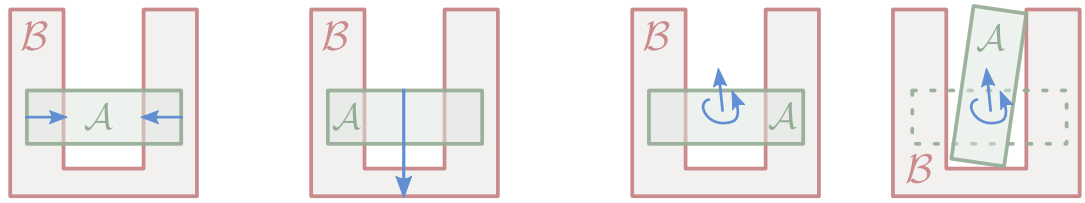


Figure 1.20: Configurations where non-penetration constraints are violated. (a) Local penetration depths fail to handle this case because the two constraints have opposite directions. (b) Minimal translational distances can be used to resolve the violation, but this would result in the unrealistic traversal of $\mathcal{A}$ through the bottom of $\mathcal{B}$. (c) Generalized penetration depth provides algorithms with a more realistic solution using both a rotation and a translation.

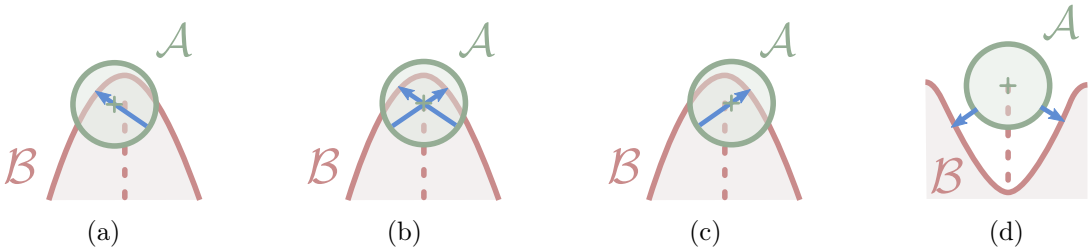


Figure 1.21: A disk $\mathcal{A}$ with its medial axis reduced to its center and a free-form shape $\mathcal{B}$ with its medial axis shown as a dotted line and penetration vectors as blue arrows. (a) The center of $\mathcal{A}$ lies right before the medial axis of $\mathcal{B}$. (b) The center of $\mathcal{A}$ lies on the medial axis of $\mathcal{B}$ and the penetration vector is non-unique. (c) A slight movement of $\mathcal{A}$ on the right caused the penetration vector direction to evolve discontinuously from (a) to (c). (d A similar singularity with distance computation with one concave shape (c).

> Alternatives to the penetration depth were proposed as an attempt to overcome the normal discontinuity issue. Some approaches define an alternative notion of penetration depth that is continuous with respect to the motion parameters [11, 28, 47, 144].

One of the most typical use of geometric contact models based on penetration-depth is for haptic rendering [89, 13]. Indeed, penalty-based methods, that rely only on penetration depths to compute contact forces, have been shown by Sagardia et al. [115] to be the most efficient approaches for haptic rendering in practice compared to impulse-based and constraint-based approaches (that may rely on distance-based geometric contact models instead).

1.4 Choice of geometric representation

The choice of a geometric representation for the B_k strongly impacts the performances and accuracy of the simulation as well. The two main families of representation are discrete volume representations and boundary representations. The former (see Section 1.4.1) approximates the shape interior with simple volumetric elements (typically spheres or cubes) while the latter only represents their boundary. This second family can be split into nonsmooth representations, i.e., triangle-based which are piecewise linear representations of the shape boundaries (see Section 1.4.2), or smooth representations, which represent CAD model boundaries with smooth curves and surfaces (see Section 1.4.3). It is noteworthy that the most accurate export formats used by CAD modelers are typically smooth boundary representations. Within the remaining of this manuscript, the term BRep will refer to smooth boundary representations only (that is, excluding polyhedral representations).

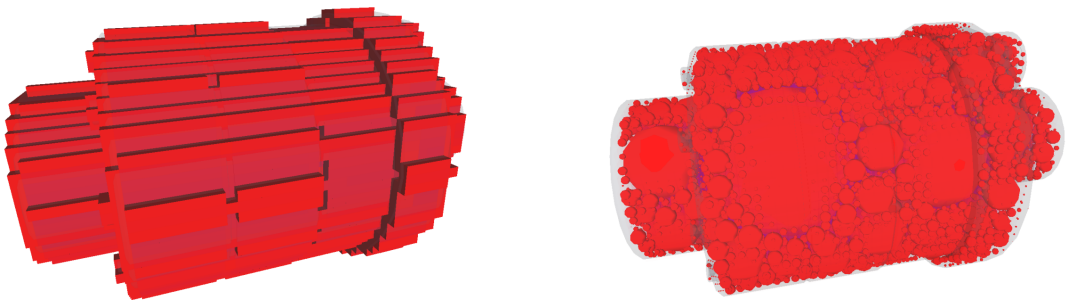
The following sections describe each category as well as their consequences on non-penetration constraints.

1.4.1 Discrete volume representations: fast with low accuracy

Volumetric representations are based on a discretization of the volume describing each B_k. The elementary volumes used by this discretization depend on the specific method chosen. Here, two popular approaches are described. One allows algorithms to be equipped with penetration depth computations Section 1.3.2 only while the other enables distance computations as well Section 1.3.1.

1. Voxels are cuboidal axis-aligned elements forming a grid (see Figure 1.22a) that can be used for penetration depth computations when they are combined with distance maps to represent an implicit surface (that may have continuous normals). They rely on two different geometric representations. When two shapes, say $\mathcal{A}$ and $\mathcal{B}$, are in a state of interpenetration, $\mathcal{A}$ has its volume approximated with voxels while $\mathcal{B}$ has its boundary covered by a finite number of points. The voxels, if any, of $\mathcal{A}$ in which points of $\mathcal{B}$ lie, are found and are used to estimate the penetration depth. Those approaches are very well suited to parallelization, including on high performance Graphics Processing Units (GPU) (See for example [38]);

2. Inner spheres trees [141] are another approach that fill the volume of each B_k with spheres of various diameters in a very compact way Figure 1.22b. Those spheres are organized into a tree structure in a way enabling the fast search of the closest spheres among those that approximate $\mathcal{A}$ and $\mathcal{B}$. Those closest spheres are used to obtain an approximate value of the distance. In case of interpenetration, the intersecting spheres can be found efficiently and used to produce an estimation of the penetration volume.



(a) (b)

Figure 1.22: A motor approximated with (a) voxels and (b) spheres of various diameters. Image credit [141].

Image-space based approaches (see for example [97, 9, 76, 59]) is a similar family of methods where the volume of each B_k is not explicitly approximated. Instead, the image of each B_k is rendered from different points of views using a traditional graphics rendering pipeline. The resulting 2D images are then exploited to observe the penetration depth or volume of $\mathcal{A}$ and $\mathcal{B}$. While being very fast because they exploit the power of GPU, the accuracy of these methods is strongly dependent upon the capabilities of the GPU rasterization pipeline itself (the resolution of its output in particular).

All the methods presented in this section suffer from similar issues which are inherent to volume-based approaches:

- Accuracy: the approximations performed induce significant deviations from the effective contact normal obtained from the shapes 'as manufactured' of $\mathcal{A}$ and $\mathcal{B}$;
- Lack of reliable information about tangents: a proper definition of tangents is often not readily available with those methods. The absence of accurate tangent directions prevents the simulation of friction.

This makes them unsuitable for applications where realism of the simulated phenomena is favored over performances. They are however, particularly adapted to applications requiring a high refresh rate where the computational time allocated to CD is scarce. Haptic simulations with force feedback is thus a typical example of use-case for volumetric representations.

1.4.2 Piecewise linear boundary representation: polyhedral approximations

Polyhedral approximations are the most popular approach throughout several communities because they offer a great balance between accuracy and efficiency of geometric queries. They are piecewise-linear representations of the boundary of B_k. Typically, a 3D modeler is used to create smooth representation of B_k. This smooth model is input into another software that generates a piecewise-linear approximation of the boundary of B_k. Most often, this generation process can be customized, e.g., to control the maximal chordal deviation between the original shape and its approximation, to refine the mesh in highly-curved areas, to homogenize the aspect of the triangles, i.e., to avoid elongated triangles, etc. All these parameters contribute to the accuracy of the geometric approximation of B_k and thus, to the accuracy of the overall simulation.

However, approximating the shape of B_k has one important drawback: it implies that the non-penetrations constraints are approximated as well, as shown in Figure 1.23. Let us note however that for more general cases, the approximation of $\partial\mathcal{C}$ would not be piecewise-linear if some B_k has at least one rotational degree of freedom. Indeed, rotations introduce additional non-linearities to the contact constraints.

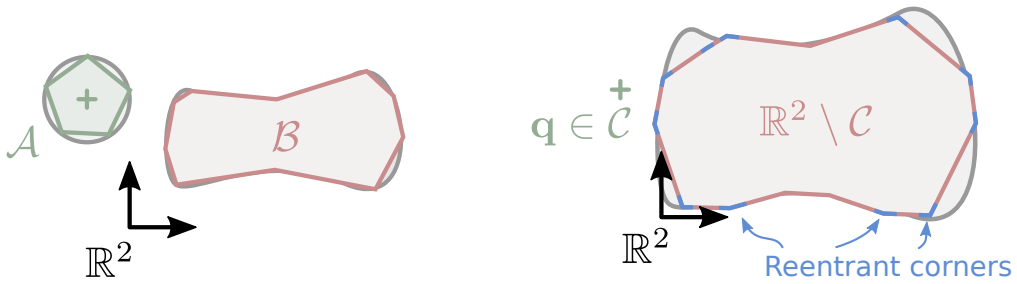


Figure 1.23: (a) The 2-degrees of freedom scenario from Figure 1.1 after tessellation. The original shapes are shown in grey behind their approximations in order to visualize their deviations. (b) The corresponding approximated feasible space $\mathcal{C}$ and approximated obstacle $\mathbb{R}^n \setminus \mathcal{C}$. The original obstacle is represented in gray.

In practice, since the normals at contact points no longer have continuous first order derivatives with respect to the generalized coordinates, the user may experience unrealistic interactions between $\mathcal{A}$ and $\mathcal{B}$ that may change their behavior in sudden and unexpected ways because of unrealistic impacts and jamming. Moreover, the necessary existence of *reentrant corners* (see Figure 1.23b) in the approximation of $\partial\mathcal{C}$ has been shown by Pfeiffer and Glocker [105, 50] to impede the stability of dynamics constraint solvers.

The discontinuity of contact normals is actually a well-understood problem for Finite Element Model simulations based on tetrahedral or triangular meshes. Indeed, multiple attempts to dodge this issue are made using *3D smoothing methods*, i.e., the piecewise-linear contact surfaces are

interpolated smoothly by surface patches [109, 54, 110, 99]. In the context of interactive and real-time dynamics simulations, the two main approaches attempt to address this issue are:

1. Reducing the chordal deviation. Indeed, using more triangles produces a better approximation of curved areas of each B_k. This reduces the approximation errors of the contact constraints and thus, the adverse effects of impacts on edges and vertices added at the cost of a higher number of contact points. Such an increase of the number of contacts raises important 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 desired;

2. Choosing a geometric contact model that yields smooth constraints even if the contacting shapes are not smooth. As mentioned in Section 1.1, penetration-based geometric contact models based on smooth measure of penetrations exist, i.e., the *growth distance* [47, 28] or the *continuous penetration depths* [144]. However, their ranges of application are limited: the former applies only to convex shapes while the latter produces contact constraints that cover only a smooth approximation of $\mathcal{C}$.

1.4.3 Smooth boundary representations: toward optimal accuracy

Smooth boundary representations are the most common *native* output of industrial CAD modelers. The BRep data structure describes the boundary ∂B_k of B_k as a 2-manifold CW-complex which is connected, closed, oriented, without self-intersections. The previous properties define a volume, which is also commonly called a 'solid' in the CAD community. The CW-complex structure (see Figure 1.24) includes three types of entities with a distinction between topological entities (vertices, edges, faces with restrictions over their parametric domains) and their corresponding geometric counterparts (points, curves, and surfaces) [128]:

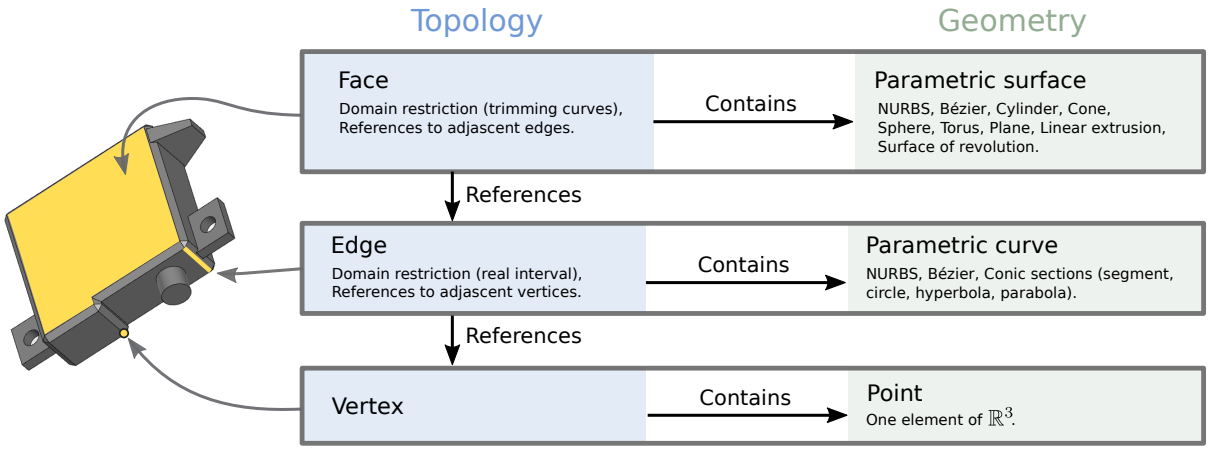


Figure 1.24: Constitutive entities of the BRep data structure.

1. *Faces* correspond to 2-manifold smooth entities. They combine a surface represented as a mapping $\sigma : \mathbb{R}^2 \to \mathbb{R}^3$ with trimming curves that reduces their parametric domain to a closed proper subset of the 2D plane $\mathbb{R}^2$. Each trimming curve corresponds to an edge of the BRep model;

2. *Edges* are 1-manifold entities located at the intersection of exactly two faces. The 2-manifold structure of the BRep guarantees that two faces cannot intersect without forming an edge, and three faces cannot meet at a single edge either. Edges are modeled by curves represented by the mapping $\alpha : \mathbb{R} \to \mathbb{R}^3$;

3. *Vertices* are 0-manifold entities located at the intersection of edges. Note that two intersecting edges do so at a vertex only. The geometry of a vertex is a single point of $\mathbb{R}^3$.

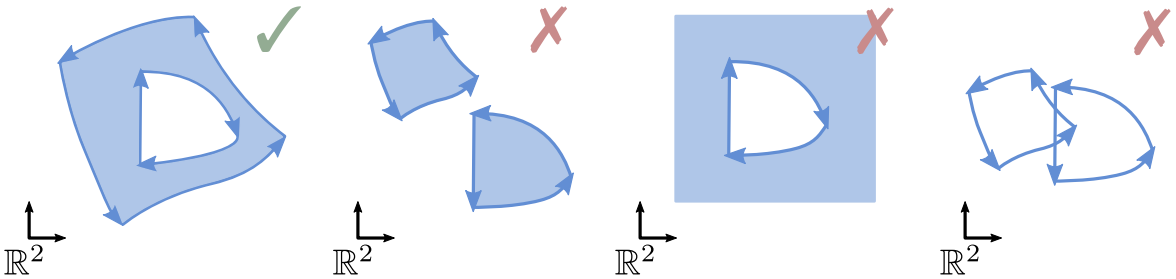
Each face of a BRep model associates a surface with a set of trimming loops composed of multiple smooth curves embedded in its 2D parametric plane $\mathbb{R}^2$. Those loops are necessarily satisfying the following properties:

- They do not self-intersect and do not intersect each-other;
- Exactly one of the loops named the *external loop* splits the 2D plane into two parts defining its interior and exterior such that all the other loops lie on its interior;
- A distinction can be made between the *solid domain* of a face and the part of $\mathbb{R}^2$ that is *trimmed out*. The solid domain is the subset of $\mathbb{R}^2$ corresponding to the material points of the solid represented by the BRep model. The trimmed out domain is the complement of the solid domain. To characterize this property, each loop (and thus each curve composing it) is oriented in such a way that given a point $\mathbf{p} \in \mathbb{R}^2$, the point $\mathbf{p}^*$, which is the closest to $\mathbf{p}$ and located on a trimming curve, and $\mathbf{t}^*$ the tangent of this curve at $\mathbf{p}^*$, then:

$$\mathbf{t}^* \times (\mathbf{p} - \mathbf{p}^*) > 0 \Rightarrow \mathbf{p} \in \text{ solid domain}, \tag{1.26}$$

  where the operator $\times$ is the cross product, i.e., $\mathbf{a} \times \mathbf{b} = \mathbf{a}_x\mathbf{b}_y - \mathbf{a}_y\mathbf{b}_x$. Informally speaking, the point $\mathbf{p}$ is inside the solid domain if and only if it is *located on the left hand side* of $\mathbf{t}^*$.

A valid set of trimming curves, as well as various invalid configurations are depicted in Figure 1.25.



(a) (b) (c) (d)

Figure 1.25: Various arrangements of trimming curves with their orientations (depicted by arrows). (a) has one external loop and one internal loop and is valid. Its solid domain is filled in blue. Other configurations are forbidden because (b) there are two external loops, (c) the curves are oriented in such a way that the solid domain is unbounded, and (d) the trimming curves intersect.

The orientation of faces is part of the orientation of a solid to enable the definition of its interior and exterior, which is also equivalent to the definition of the normal $\mathbf{n}$ at any G^2 point of ∂B_k such that $\mathbf{n}$ points either outward or inward with respect to the interior of B_k. Refer to [43] for definitions of the geometric continuities G^1 and G^2.

Within the remaining chapters, a *feature* designates either a vertex, an edge, or a face together with their associated geometric entity. Surfaces are categorized as being either canonical or non-canonical depending on their types. Canonical surfaces include spheres, cylinders, cones, tori, and planes. Non-canonical surfaces, also called *free-form surfaces*, are all the other types of surface that can participate to the definition of a BRep model which include Bézier, B-Spline

surfaces or, more generally, NURBS[8] surfaces. Let us observe that each NURBS surface can be subdivided into a finite number of rational Bézier surfaces [107] without loss of accuracy, i.e., the set of rational Bézier surfaces derived from a NURBS surface matches the latter exactly. Therefore, the design of algorithms that take into account rational Bézier surfaces rather than NURBS surfaces do not incorporate any deviation from ∂B_k. A similar distinction can be applied to curves. Canonical curves designate line segments and conics, i.e., circle, ellipse, hyperbola and parabola. Non-canonical curves are Bézier, B-Spline curves or, more generally, NURBS curves. Just like surfaces, NURBS curves can be decomposed into several rational Bézier curves without loss of accuracy. Here again, the design of algorithms that take into account rational Bézier curves rather than NURBS curves do not incorporate any deviation from ∂B_k.

Let us now assume that the orientation of ∂B_k is such that the faces normals $\mathbf{n}$ point outward. Then, the faces associated with each of these canonical surfaces can be subdivided into two disjoint categories:

1. *Concave* canonical faces have their normals pointing toward the reference curvature center of their associated surface, whatever the point considered, to conform to the concept of orientation index as defined by Li et al. [82];

2. *Convex* canonical surfaces have their normal pointing toward the direction opposite to the reference curvature center of their associated face.

All the types of convex and concave canonical surfaces are shown on Figure 1.26. Because the concept of concavity/convexity refers to the curvature of surfaces, the orientation index does not apply to a plane. The concept of concavity/convexity can be extended to a set of adjacent faces, including planar faces, based on the face normals, as defined over ∂B_k.

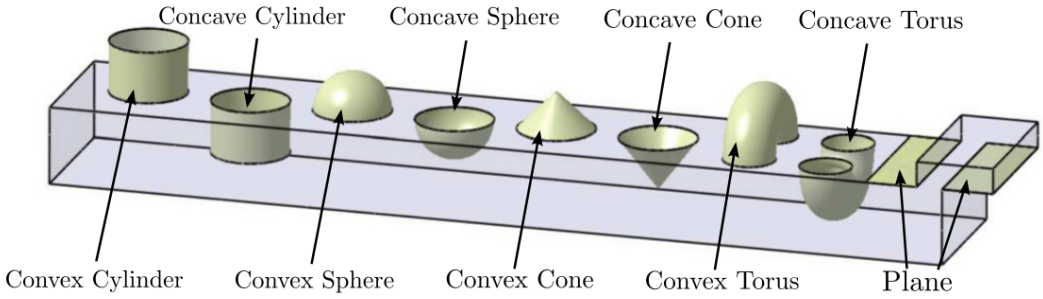


Figure 1.26: Distinction between *convex* and *concave* canonical surfaces.

The concepts of convex and concave edges however, are quite different from that of faces. It has first been introduced for edges lying at the intersection of two planar faces [74, 116, 117] and then mentioned by Sakerau et al. [117] for curved objects. The status of an edge depends on the status of each of its points. At a point along an edge, its status depends on the type, parameters, and relative position of the two surfaces that meet where it appears. A point edge $\mathbf{p}$ is said to be:

- G^1 if the two surfaces meeting at this point share the same tangent plane. In addition, the corresponding normals must point toward the same directions (their scalar product is positive). This definition distinguishes cases like Figure 1.27b from cases like Figure 1.27c which is not considered G^1 in this manuscript;

[8] Non Uniform Rational B-Splines.

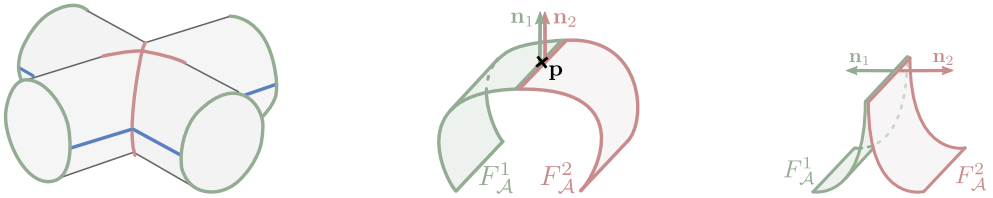
- *Convex* if the angle α between the outward normals of the two faces $F^1_{\mathcal{A}}, F^2_{\mathcal{A}}$, intersecting at the edge $E^{12}_{\mathcal{A}}$ containing this point is such that $0 < \alpha < \pi$. The value α is refered to as the *solid angle* at point $\mathbf{p}$ and is computed as:

$$\alpha = \langle \mathbf{n}_1 \times \mathbf{n}_2, \mathbf{t}_{12} \rangle \,, \tag{1.27}$$

  where $\mathbf{n}_1$ (resp. $\mathbf{n}_2$) is the normal of $F^1_{\mathcal{A}}$ (resp $F^2_{\mathcal{A}}$) at $\mathbf{p}$ and $\mathbf{t}_{12}$ the tangent of the edge at $\mathbf{p}$ with its orientation defined in accordance with the orientation of $E^{12}_{\mathcal{A}}$ in F_1 (the orientation of an edge is inherited from the orientation of its corresponding trimming curve);

- *Concave* if the solid angle α at $\mathbf{p}$ is such that $-\pi < \alpha < 0$.

An edge is assigned a status if all its points share the same status, e.g., an edge is convex if and only if all its points are convex. Prior work about edge status does not exhibit a complete taxonomy of edge statuses. The assignment of an edge status when some of its points are convex while others are concave is still an open problem. Likewise, a complete taxonomy of edge statuses requires complementary work.



(a) (b) (c)

Figure 1.27: (a) Examples of convex edges (green), concave edges (red), and G^1 edges (blue). (b) Example of G^1 edge. The normals $\mathbf{n}_1$ and $\mathbf{n}_1$ of the surface of each face $F^1_{\mathcal{A}}$ and $F^2_{\mathcal{A}}$ at a common point of the edge are shown to be collinear. (c) An edge that is not considered G^1 in this manuscript. While both surface of $F^1_{\mathcal{A}}$ and $F^1_{\mathcal{B}}$ share the same tangent plane along this edge, their normals $\mathbf{n}_1, \mathbf{n}_2$ at their common point $\mathbf{p}$ along the edge are antipodal, i.e., $\langle \mathbf{n}_1, \mathbf{n}_2 \rangle < 0$.

Those distinctions between canonical and non-canonical curves and surfaces is necessary to design fast, accurate and robust algorithms to compute tight bounding volumes (see Section 3.3), exploit known curvature information to filter potential contact pairs (see Section 2.2), and compute exact contact points using analytic methods (see Section 4.3.1).

The taxonomy of canonical and non-canonical faces is also at the basis of grouping criteria for faces, thus avoiding redundant computations (see Section 2.3). Actually, the real-time efficiency of the methods presented in this work comes from the observation that ∂B_k of industrial BRep models are mostly composed of canonical faces [118]. If free-form surfaces are encountered in ∂B_k, they often participate to blends between canonical areas or connection surfaces, which are small and sometimes not even represented in the CAD model (see Figure 1.28), e.g., chamfers. Moreover, ∂B_k contains canonical faces for simplicity of the corresponding modelling process as well as manufacturing purposes. Such observations lead to the existence of conformal contacts as well as smooth rolling and sliding motions.

Using smooth BRep models for CD has several clear advantages:

- Because the original shapes are exploited, no geometric approximation is performed over ∂B_k compared to approaches described in Sections 1.4.2 and 1.4.1. Therefore, maximal fidelity with respect to real physical behavior of *MS* is preserved;

- Because features of the BRep models are smooth, the distance function is smooth as well and so will be the gap functions and thus, the contact constraints. Therefore, unrealistic shocks due to contact constraints discretization are avoided. This consequence preserves also the fidelity with respect to the real physical behavior of *MS*;

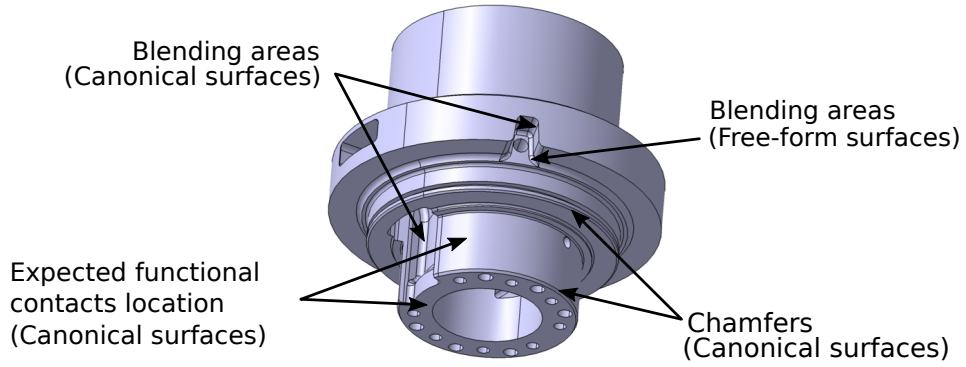


Figure 1.28: A mechanical part is composed mostly of canonical curves and surfaces. Free-form surfaces are mostly located in blending areas.

- A whole part of the boundary of $\mathcal{C}$ can be described with only one contact constraint corresponding to a LMD. This is possible due to the existence of exact contact kinematic function that provides the evolution of the location of closest points as a function of B_k's motion [93, 139]. Overall, less constraints are necessary to ensure non-penetration compared to methods based on polyhedral models that require several constraints to obtain a piecewise-linear approximation (see Section 1.4.2);

- No geometric approximation algorithm, i.e., tessellation, voxelization, etc., has to be executed on ∂B_k, and the user has less parameter to tune for model preparation before simulation. Indeed, there are cross influences between the distance function discretization, kinematic constraint discretization, and chordal deviation related to the geometry processing of ∂B_k. Those are significant advantages from the ergonomic standpoint as they reduce significantly the time spent for the model preparation and the monitoring of the quality of a simulation.

On the other hand, disadvantages are significant as well:

- While the gap functions g_i used for non-penetration constraints in the Equation (1.6) are non-linear because of the non-linearity of the parametrization of the positions (particularly the rotations), the use of smooth BRep adds even more nonlinear terms to g_i because of the non-linearity of the contact kinematics that depend explicitly on the shape of the models [93, 139]. This can reduce the accuracy of dynamics constraint solvers based on constraint linearizations performed at the beginning of the time-step. Heading toward optimal accuracy requires the use of a non-linear constraints solver [4] though it induces additional computation times. This leads to a clear issue related to the balance between model accuracy and computation time;

- The literature about real-time collision detections applied to smooth BRep models is scarce since most existing methods deal with only one individual surface or curve which would be inefficient if used as-is for complete BRep models. Some of them still rely on polyhedral approximations to yield the final outputs. The most representative methods are detailed in Section 1.5.

This last mentioned performance issue is the main topic of this manuscript. The study of non-linear constraints solvers mentioned in the first point is left to future works.

1.5 Distance computation on smooth boundary representations: existing methods

As mentioned in the introduction, this thesis aims at contributing to the CD component of dynamics simulation engines targeting applications where accuracy is more important than computation times as long as they remain real-time and with a reasonable update rate (above 30Hz and ideally 60Hz). Given the observations made regarding the choice of a geometric contact model in Section 1.3 and the choice of a geometric representation of the simulated objects in Section 1.4, a natural choice to favor accuracy is to combine a distance-based geometric contact model using LMD or quasi-LMD with implicitly dilated smooth BRep models to avoid most errors due to geometric approximations (the only remaining errors being at sharp edges that are rounded off by the erosion/dilation process discussed in Section 1.3.1.2).

However, existing approaches computing LMDs between pairs of smooth BRep models mostly deal with a pair of features (curves, surfaces, or points) only instead of entire geometric models. Thus, applying those methods to all possible feature pairs of two BRep solids would be inefficient since it would have a quadratic algorithmic complexity $O(mn)$ where m and n are the total number of features in each model. Nishita et al. and Zou et al. [100, 147] are two noticeable alternatives handling entire models using recursive feature tests of lower dimensions whenever a LMD footpoint is found to lie outside of the domain of a higher dimension feature.

The following sections analyze various existing methods addressing the three stages of typical LMD computation algorithms over a pair of features:

- Coarse localization of feature areas that may contain LMD footpoints. This operation identifies feature areas where LMD footpoints are likely to be located. Two families of methods can be distinguished: methods based on subdivisions of the feature domain (see Section 1.5.1) and methods based on a secondary (polyhedral or volumetric) representation (see Section 1.5.4). Let us note that methods based on secondary representations are generally applicable to full BRep models though there is no guarantee of generating all the possible LMD footpoints (with distances smaller than the user-defined threshold d_{max});

- Exact computation of the LMDs footpoints between the individual features (see Section 1.5.2);

- Removal of footpoints that lie outside of each face domain delimited by trimming curves (see Section 1.5.3). Footpoints lying outside of an edge domain have to be removed as well but this is straightforward since an edge domain reduces to a real interval.

For simplicity, until the end of this manuscript, the term *LMD* is generically used to identify either the footpoints of an LMD, their parametric coordinates, or the actual distance between those points. Either meaning should be clear from the context. For example, 'locating a LMD' implicitly means 'locating LMD footpoints and computing their parametric coordinates'.

1.5.1 Subdivision methods

Subdivision methods are algorithms localizing feature areas that may contain LMDs. Given a pair of features, all subdivision-based methods follow the same recursive procedure detailed in Algorithms 1 and 2. First of all, the whole domain of definition of both features is taken into account. Then, pairs of sub-domains that may contain a critical point of $\tilde{d}$ are recursively subdivided into smaller pieces until candidate solutions are isolated into independent sub-domains. Those small sub-domains are then stored for future processing in order to locate exactly the

LMD. The accurate locations of the critical points are computed by a numerical unconstrained optimization method initialized using a point chosen as accurately as possible in this small sub-domain. This last step is detailed in Section 1.5.3.

This simple scheme is summarized by Algorithm 1 for the LMD computation between the point $\mathbf{p}$ and the surface σ, and by Algorithm 2 for the LMD computation between two surfaces σ_1 and σ_2. The list R will be filled up with the parameters defining the LMD footpoints. Let us note that both algorithms are very similar besides the fact that both surfaces need to be subdivided in the second one.

Algorithm 1 Algorithmic scheme to compute LMDs between a point and a surface.

```
function ComputeLMDs(σ, p, R)
    if MayContainUniqueSolution(σ, p) then
        guess ← InitialGuess(σ, p)
        R ← R ∪ NumericalResolution(σ, p, guess)
    else
        σ' ← Subdivide(σ)
        for all σ'_i ∈ σ' do
            if MayContainSolution(σ'_i, p) then
                ComputeLMDs(σ'_i, p, R)
            end if
        end for
    end if
end function
```

Algorithm 2 Algorithmic scheme to compute LMDs between two surfaces.

```
function ComputeLMDs(σ1, σ2, R)
    if MayContainUniqueSolution(σ1, σ2) then
        guess ← InitialGuess(σ1, σ2)
        R ← R ∪ NumericalResolution(σ1, σ2, guess)
    else
        σ'_1 ← Subdivide(σ1)
        σ'_2 ← Subdivide(σ2)
        for all (σ'_{1,i}, σ'_{2,j}) ∈ (σ'_1, σ'_2) do
            if MayContainSolution(σ'_{1,i}, σ'_{2,j}) then
                ComputeLMDs(σ'_{1,i}, σ'_{2,j}, R)
            end if
        end for
    end if
end function
```

Methods for LMD computation between a point and a curve, between two curves, or between a curve and a surface would follow the same subdivision schemes. In addition, both methods could be applied even if the surfaces were deformable as long as all the computations, including subdivisions, occur at runtime. If the cache storing intermediate subdivision results proposed by Johnson et al. [70] is used then, deformable features cannot be handled without additional work for updating such a cache.

Both algorithms rely on several procedures, the details of which are subject to debates as expressed through the literature:

- MayContainSolution, also referred to as the *culling test*, tests whether a pair of features may contain a LMD (on a specific rectangular sub-domain for surfaces or on a specific real interval for curves) or not. If not, there is no need to subdivide either feature any further. This test must be conservative, i.e., it must not discard pairs that contain a solution, but not overly, i.e., it should do its best to discard as many pairs as it can that will not lead

to any solution. Thus, an ideal culling test would be fast to execute and would identify all the sub-domains that do not contain any LMD. Existing approaches are described in Section 1.5.1.1;

- MayContainUniqueSolution tests whether or not the given pairs of features may contain a single solution, at most. If so, an exact numerical computation of the LMD can be performed without risking to miss other local solutions if multiple solutions were existing over the features. Existing approaches are described in Section 1.5.1.3;

- Subdivide returns an array of feature areas σ' (or σ'_1 and σ'_2) by cutting the features' parametric domains into several parts. Also, it subdivides NURBS and Bézier curves and surfaces explicitly at their geometric level to meet the test conditions for MayContainSolution. Existing approaches are described in Section 1.5.1.2;

- InitialGuess returns the initial solution parameters to initialize the computation of accurate footpoint locations. The closer it is to the exact solution, the faster the iterative numerical resolution algorithm will converge. Existing approaches are detailed in Section 1.5.2.1;

- NumericalResolution is an iterative numerical optimizer that finds the exact closest pair of points. Existing approaches are detailed in Section 1.5.2.2.

1.5.1.1 Solution existence tests

Most authors focused mainly on the determination of an accurate test to identify areas of the search domain guaranteed not to contain any LMD. Most approaches rely on characteristics specific to one particular type of surface or curve to design efficient tests. Zhou et al. [146] and Elber et al. [39] exploit the fact that the distance function gradient involving B-spline curves and surfaces only, can be formulated as a B-spline form as well, i.e., it is a closed form. That way, the convex hull property of B-splines can be used to detect zones of the solution space that may contain the origin, i.e., points where the gradient vanishes correspond to critical points of the LMD function.

On the one hand, Zhou et al. [146] set up a test based on the graph of the distance function. It simultaneously checks that a solution exists on the given parameter ranges and, at the same time, reduces its size in such a way that the new sub-domain converges quadratically toward the solution. This leads to expensive computations involving either the explicit construction of $O(n^2)$ 2D convex hulls where n is the number of distance function parameters (the Projected-Polyhedron algorithm), or its reformulation as the resolution of n minimization and maximization problems with n constraints (the Linear Programming algorithm) to obtain a better refinement of the current solution search space and avoid explicit computation of convex hulls.

Beside root-finding algorithms based on the reformulation of the distance function gradient under a B-spline form, methods using bounding volumes of the curve/surface and its derivative(s) have been proposed. Snyder et al. [125] use interval arithmetic, which is equivalent to manipulating AABBs[9], on time-varying curves/surfaces to bound the distance function and its gradient. Then, if the origin is located inside of the bounds of the gradient, the corresponding curve or surface areas can be subdivided to refine the solution search.

Finally, regarding the point and/or NURBS (curve or surface) cases, some authors do not explicitly use any derivative (of the surfaces and of the distance function) to design a culling

[9] Axis Align Bounding Boxes are boxes with axes aligned along the coordinate axes of the reference frame of *MS*.

test [36, 87, 120]. Let us note that both Dyllong et al. [36] and Ma et al. [87] are known to erroneously exclude some curve/surface patches that may contain a solution: counterexamples are given in Selimovic and Chen [120, 25]. Given the interpolation property of NURBS, i.e., the fact that the control polygon (resp. polyhedron) endpoints coincide with the curve segment (resp. rectangular surface patch) extremities, Selimovic et al. [120] characterize whether the candidate point is to be projected onto the curve (resp. surface) interior or on one of its extremities. If the projection stands at an endpoint, there is no need to carry on the subdivision algorithm on this curve (resp. surface). This detection is performed in linear time with respect to the number of control points of the NURBS object. The test checks the existence of a plane containing one endpoint while having no intersection with the curve's (resp. surface's) control polygon (resp. polyhedron) and separating the candidate point from the curve (resp. surface) itself. If such a plane exists, then the corresponding curve or surface can be ignored.

1.5.1.2 Search space subdivision methods

The search space must be split into smaller areas in order to refine the localization of LMDs and discard the areas guaranteeing to contain any. Depending on the type of features involved in the distance computation problem, subdividing their parametric domains may impact their geometric representations. Notably, Bézier, NURBS curves (resp. surfaces) are usually explicitly subdivided into two other curves (resp. four other surfaces) of identical nature with new control points. This allows culling tests to exploit their geometric properties like endpoint interpolation, hodograph, and convex hull [107].

In any case, a choice must be made regarding how the parametric domain itself is subdivided. The simplest choice is to split the current parametric rectangle either along its largest dimension [39] in case of a Bézier surface or inside the domain at an interior knot when the curve/surface is a NURBS [103, 120].

A more adaptive approach proposed by Chang et al. [24] when the curves (resp. surfaces) are under Bézier form is to apply the GJK algorithm [48] to compute the closest point(s) between the convex hulls of their control points and retain their barycentric coordinates (obtained as a by-product of the GJK algorithm). Those barycentric coordinates are then exploited to locate a point along the curve (resp. surface) that is close to the result of this projection. Actually, this approach is identical to algorithms that find an initial guess for the numerical optimization algorithm presented shortly in Section 1.5.2.1 and detailed further in Section 4.3.1. However, this guess is used as a curve (resp. surface) subdivision point rather than an initialization of the exact LMD computation.

1.5.1.3 Ensuring the uniqueness of a solution

Very few approaches exist that rigorously ensure the uniqueness of a solution for a given pair of features. The usual method assumes that the flatter the involved feature areas are, the more likely unique a LMD will be over its sub-domain. Several authors propose *flatness conditions* that serve as a heuristic to assume the solution uniqueness. Johnson et al. [70] assume a feature area is flat if the half-angles of its tangent cones bounded with cones of revolution are small enough. In the special case of NURBS, Zhao et al. [145] claim that a feature area will be flat if all the weight (refer to [107] for detailed definitions) associated to each control point is bellow a given threshold. Finally, [36, 87, 120] propose to compute the distance from a Bézier patch control polyhedron to its planar approximation under the assumption that small distances imply flatness. Though in practice those small distances actually imply small thickness as shown by the Figure 1.29, the corresponding curve/surface can be erroneously reported as being flat.

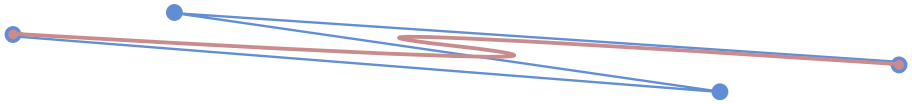


Figure 1.29: A Bézier curve (in red) erroneously considered flat by [36, 87, 120] because its control points (blue dots) are close to being collinear. Its control polygon is shown in blue.

Because they are only heuristics, none of these approaches actually ensure the uniqueness of a LMD for all configurations. A few robust methods exist [125, 39, 102] but might often require a very large number of subdivisions, in practice, before becoming conclusive over the subdivided sub-domains. Other methods rely on properties that are specific to a given type of geometric object, e.g., for the projection of a point on a NURBS curve, Chen et al. [26] exploit the variation diminishing property of the distance function also expressed as a NURBS curve. However, this is no longer applicable to surfaces because they do not benefit from this property (details about the properties of NURBS curves and surfaces can be found in [107]).

Within our framework, the tangent-cone based method presented by Johnson et al. [70] is used for splitting curves and surfaces into almost-flat areas at pre-computation time as described in Section 2.4.1. Moreover, special cases are made at runtime for LMD computation between some pairs of canonical surfaces to find all the closest points robustly using analytical solutions (see Section 4.3.1).

1.5.2 Numerical resolution of the optimization problem

The exact localization of one LMD between two points, surface areas, or curve areas can be obtained either analytically or using an iterative method, depending on the geometric properties of the two objects considered. On the one hand, choices of analytical methods to process canonical curves and surfaces are discussed in Section 4.3. On the other hand, iterative methods are two-steps processes. Firstly, a guess must be performed regarding the location of closest points (see Section 1.5.2.1) over the sub-domains identified by the recursive subdivision previously detailed in Section 1.5.1. Then, this guess is iteratively refined until it converges toward the exact LMD (see Section 1.5.2.2). As mentioned in Section 1.5.1.3, relying on methods generating only one LMD per feature area is sound as each pair of such sub-domain is assumed to contain only one LMD.

1.5.2.1 Finding an initial guess

Whenever the parametric domain of the input features are sufficiently narrowed by a subdivision method in such a way that they may only contain a unique solution, a solution guess must be chosen in order to initialize an iterative numerical solver. The naive method consists in taking the barycenter of the parametric sub-domain defining the feature area after subdivision [39, 145]. However, obtaining a better guess is desirable as it will affect directly the convergence rate of the numerical resolution method.

A significantly more accurate method relies on the approximation of the feature areas using polyhedrons (or polylines for curves), and on computing the closest points between those approximations. Then, parametrization functions that map any point on those polyhedra to a parametric coordinate are defined. Applying those functions to the closest points between the polyhedra provide the initial solution guesses. Several variants exist, depending on the choice of polyhedral approximation:

- Polyhedral approximations with vertices located onto the surface area (resp. curve segment), forming a mesh (reps. polyline), is used by [124] (See Figure 1.30a);

- As a special case, Bézier surfaces areas (resp. curves segments) are handled by Thompson et al. [134]. The polyhedral approximation chosen is their control polyhedrons (resp. polygons) (See Figure 1.30b);
- Similarly to [106], each surface area (resp. curve segment) can be approximated by a quadrilateral (resp. a line segment) (See Figure 1.30c).

This method, and the specific declination (similar to [124]) used by the framework presented in this thesis, is further detailed in Section 4.3.1.

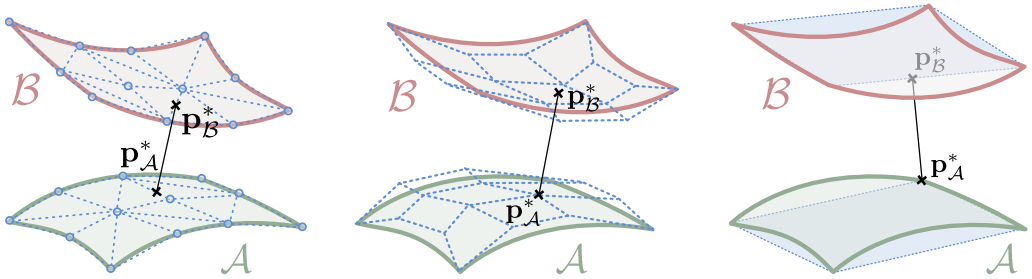


Figure 1.30: Various methods to set an initial solution between two surfaces $\mathcal{A}$ and $\mathcal{B}$ using polyhedral approximations. (a) Polyhedral approximation with all points located on the surface. (b) Using the control polyhedron of Bézier surfaces (points are not necessarily located on the surface). (c) Quadrilateral approximation. In all scenarios, $\mathbf{p}_{\mathcal{A}}$ and $\mathbf{p}_{\mathcal{B}}$ are the closest points between the polyhedral approximations.

1.5.2.2 Finding the solution

It is necessary to improve the initialization solution in order to find an accurate location of a local extrema of $\tilde{d}$. The problem to solve is usually a difficult minimization problem that has no closed-form solution, hence the use of iterative solvers. The most common method used is a bisection combined with a Newton-Raphson algorithm to cancel the gradient of $\tilde{d}$, or one of its variants like the Globalized Newton Method [108] for better robustness.

Authors have been more prolific regarding the point-projection problem, i.e., the localization of the point onto a curve or a surface closest to the given point. While Newton-Raphson based methods are still largely used, attempts were made to find alternatives using a local approximation of the curve/surface with a first or second order curve/surface that is tangent to the current iteration. The algorithmic scheme of those geometric iterative methods is the following one: the point is projected onto a local approximation of the curve/surface and then, it is inverted, i.e., the associated parameters on the original curves/surfaces are found approximately. This three-steps process (approximation, projection, and inversion) is repeated using those new parameters until convergence is achieved. Figure 1.31 illustrates one iteration of four approaches that follow this procedure. The simplest geometric iteration methods are based on a first-order approximation of the curve/surface [61, 58] (see Figure 1.31b). This approximation corresponds to the tangent plane (resp. line) of the surface (resp. curve) at the current iteration. A more accurate approach proposed by Hu et al. [62] replaces the line by a circle with a radius and center deduced from the normal curvature at the current iteration (see Figure 1.31c). Then, Liu et al. [86] improve further the convergence rate of this second order approximation for the 3D case using a torus patch instead of a circle (see Figure 1.31d). The torus radii are deduced from the surface principal curvatures at the current iteration. However, the inversion step is not easy because it requires the resolution of a bivariate over-constrained quadratic equation, which is solved approximately by a Newton-type iteration method. In the 2D case, Song et al. [126]

improves [62] further using a local approximation of the curve with a biarc (see Figure 1.31e). A biarc is a curve that is tangent to the normal curvature circles of two points located on the curve (the current iteration and another one a little further along the direction of the projection point).

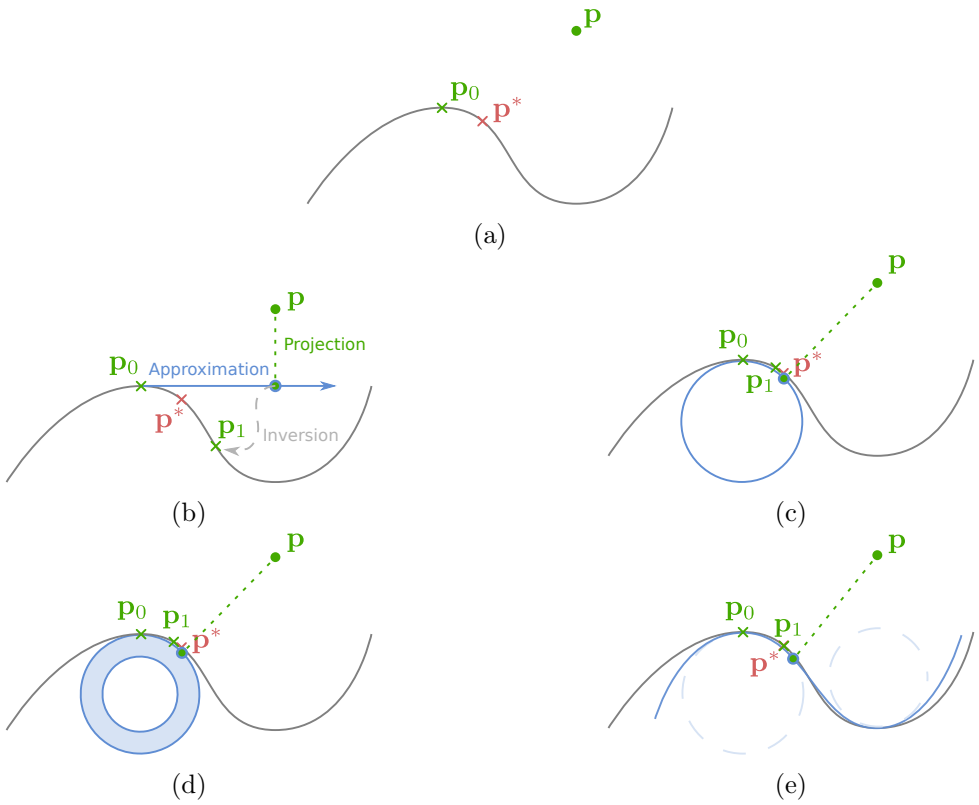


Figure 1.31: Section view of the four geometric iteration methods. (a) The initial configuration with the point $\mathbf{p}$ to project, the initialization solution $\mathbf{p}_0$, and the actual optimum $\mathbf{p}^*$ that has to be found. (b) one tangent-based [61, 58] iteration. (c) one circle-based [62] iteration. (d) one torus-based [86] iteration. (e) one biarc-based [126] iteration. The results of the three steps of each iteration are shown: approximation (blue), projection (green) and inversion (gray) to obtain the next iterate $\mathbf{p}_1$.

1.5.3 Handling trimmed surfaces

Trimming curves introduced in Section 1.4.3 must be taken in account by the collision detection framework in order to avoid generating LMD between non-solid areas of the objects that have no material existence because they are trimmed out. To detect whether the parameters of a LMD footpoint lie inside the solid domain of a face, [100] relies on the even-odd rule (illustrated in Figure 1.32), which is well-known for testing the containment of a point inside a polyhedron [5]. The method [119] used in our framework relies on a more efficient exploitation of the even-odd rule that does not require an explicit 2D ray cast and is presented in detail in Section 4.4.1.

In the context of contact tracking for haptic rendering, Nelson and Thompson [98, 133] make sure a point does not leave the solid domain bounded by trimming curves. This is achieved with a grid overlaid on the 2D plane and marking the grid cells intersecting a trimming curve as shown in Figure 1.33. Thus, the transition between the solid domain and the domain trimmed out is detected after testing if some of the marked cells are traversed after an update of the location of the LMD footpoint. The condition of Equation (1.26) is tested on the trimming curve segments intersecting those cells to determine whether the new LMD footpoint has to be trimmed out or

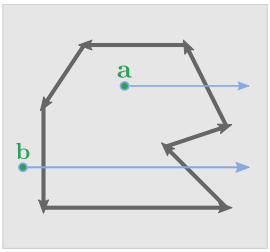
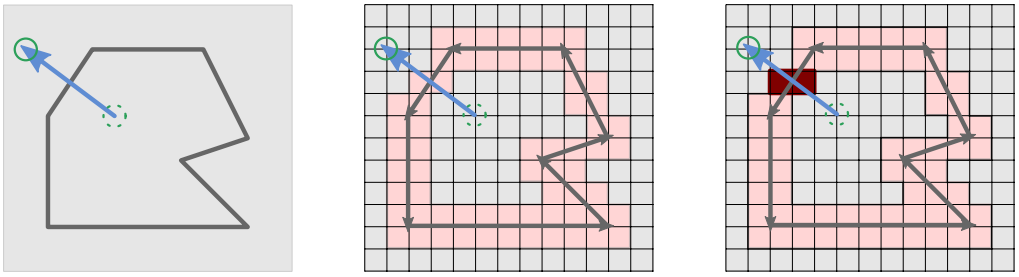


Figure 1.32: The even-odd rule for classifying a point with respect to trimming loops. A 2D ray is cast horizontally from the point. The ray associated to the point **a** intersects the trimming loop an odd number of times (one time), thus it is inside of the solid domain. The ray associated to the point **b** intersects the trimming loop twice, which is even, so it must be trimmed out.

not. This is straightforward since both approaches are restricted to trimming curves described as line segments.



(a) (b) (c)

Figure 1.33: Detecting a transition between the solid domain and the trimmed-out domain using a grid in the parametric plane. (a) A polygonal trimming loop (dark gray), the probe point (green), and the segment (blue) linking its new position (green circle) to its last one (dotted green circle) in the parametric plane. (b) The grid overlaid and the marked cells (pink). (c) Marked grid cells that intersect the segment linking the two positions of the probe point are identified in red.

1.5.4 Methods based on alternative representations

The other family of methods that attempt to solve the LMD computation problem is based on an alternative representation of either the geometric models or the distance function itself that exhibits useful properties to be exploited in combination with the original models. Any additional computation needed to produce such a representation is performed offline, i.e., before the dynamics simulation is started.

Firstly, if the surface is known by construction to be a convex NURBS patch, a first approach is to use the Newton method to compute the support-mapping function of the surface [137]. Recall that a support-mapping function $f_S : \mathbb{R}^3 \to S$ on a set S returns the point $\mathbf{p} \in S$ that maximizes the scalar product with the direction $\mathbf{v} \in \mathbb{R}^3$ given as input, i.e.:

$$f_S(\mathbf{v}) = \underset{\mathbf{p} \in S}{\operatorname{argmax}} \langle \mathbf{v}, \mathbf{p} \rangle \,. \tag{1.28}$$

This enables the use of existing support-map based algorithms like Gilbert-Foo et al. [49] to compute the exact smallest surface/surface or point/surface distance.

Secondly, many other methods [85, 106, 60, 29] are based on a polyhedral (resp. polygonal) approximation of the surface(s) (resp. curve(s)) involved in the distance computation. Given such an approximation, it is possible to use algorithms identifying common closest points between meshes to obtain an estimation of their original position on the exact model. Those approximate solutions can then be improved by any numerical method described in Section 1.5.2. Those mesh-based approaches mainly differ in the way the approximations are constructed and the guarantees they offer in relation with the original models. Lin et al. [85] approximate the surfaces of convex Bézier surfaces with ϵ-*polytopes* that guarantee a maximum distance error of $\epsilon > 0$ between the approximated surface and the original one. Piegl et al. [106] restrict its input to surfaces that do not bend more than 180 degrees and cover them with a quadrilateral mesh without cracks. Chou et al. [29] simultaneously uses two meshes to approximate a single oriented surface. While the first one is a typical mesh obtained by a tessellation, the second one, called *proxy mesh*, is obtained after offsetting the former at a distance δ toward the exterior of the object as defined by the surface normals (which is roughly equivalent to computing a tessellation of the surface dilated using a spherical envelop of radius δ). Thus, whenever the *proxy mesh* of one surface intersects the first mesh of the other, the two objects are separated by a distance smaller than δ. Then, an initialization of the solution location can be obtained by projecting the center of the triangle intersection loop back onto each surface. Typically, one would choose $\delta \geq d_{max}$ (with d_{max} the maximal LMD of interest as defined in Section 1.3.1). Note that none of these methods are conservative, i.e., none are able to guarantee that no LMD on the original model will be missed since polyhedral approximations do not reflect properly the curvature of the underlying smooth shapes.

Other very different approaches work directly on the formulation of the distance computation problem as a higher dimensional one [121, 81]. Such approaches are hard to use in practice because they often require complex pre-computations involving a symbolic computation system often leading to high memory usages.

On the one hand, Sohn et al. [81] use Line Geometry to transform the problem of LMD computation between two surfaces to an intersection problem between two 2-dimensional surfaces embedded in a four-dimensional manifold. The main idea is to exploit the fact that the two surface normals of the LMD must be collinear at the footpoints. At that position, each point onto each surface is replaced by an infinite line that passes through it and that is parallel to its normal. This representation is called the *Plücker image of the normal congruence* of the original surface. Finding the intersections of the Plücker image of two surfaces is then equivalent to finding the locations of critical points of the distance function. However, computing the Plücker images is not simple for arbitrary free-form surfaces. Sohn et al. [81] propose solutions for some canonical surfaces. In those cases, the LMD computation problem can be reformulated as a two-variables root finding problem.

One the other hand, Seong et al. [121] transform the distance minimization problem into a ray-cast against a higher-dimensional surface; the dimension of which depends on the degrees of freedom assigned to each body B_k plus the number of parameters of each feature. For example, the LMD computation between a 2D point and a 2D curve can be reformulated as a higher-dimensional problem parametrized by the position of a point relative to the curve. That is, if $\gamma(t) : \mathbb{R} \to \mathbb{R}^2$ is a 2D curve and $\mathbf{p} \in \mathbb{R}^2$ the point being projected, rather than searching for critical points from the problem extracting the roots of the 1-dimensional function:

$$f(t) = \left\langle \gamma(t) - \mathbf{p}, \gamma'(t) \right\rangle, \tag{1.29}$$

the problem is reformulated as the root extraction of the 3-dimensional function:

$$\tilde{f}(t, x, y) = \left\langle \gamma(t) - (x, y), \gamma'(t) \right\rangle, \tag{1.30}$$

where x and y are the (variable) coordinates of $\mathbf{p}$. Consequently, the implicit hypersurface $\tilde{f}(t, x, y) = 0$ contains all the possible LMD parameters for any position of $\mathbf{p}$ in the plane. To solve a LMD computation query, one has to cast on this hypersurface a ray that starts at $\mathbf{p}$ (and taking t equal to the parameter of the first endpoint of γ) and that propagates along a direction collinear with the coordinate axis representing parameter t in the high-dimensional space. However, the formulation of $\tilde{f}$ when the B_k have more degrees of freedom (typically 6 for 3D simulations) is hard and the resulting ray-casting problem on the corresponding hypersurface is expensive since it operates on a high-dimensional space.

1.6 Conclusion and presentation of the objectives

Contact constraints have been defined as a set of smooth constraints that delimit the valid positions of the various bodies B_k as parts of a complex mechanical system *MS*. Alternatively, they can be called non-penetration constraints since they prevent objects from being in a configuration where they overlap in space, which would be unrealistic provided that they are solid. The set of generalized coordinates without overlaps of the B_k has been named the feasible space $\mathcal{C}$. Contact constraints can be constructed from various geometric contact models among which:

1. The distance-based model is accurate and conveniently characterized in term of tangent cone polars (as defined in the field of convex analysis), but fails to represent small penetrations that are unavoidable due to numerical approximations generated by the integration process and the limited accuracy of floating point numbers. Moreover, multiple and conformal contacts are problematic if only one non-penetration constraint is defined for a pair of solids. Therefore, the existing concept of quasi-LMD has been described as an approach that generates multiple constraints using the local minima of the distance function between the solid boundaries represented as CW complexes. In addition, the dilation of tangent cone polars in order to enlarge the domain of definition of LMD functions has been described as an angular regularization of the distance functions that produces a better stability of the simulation near conformal configurations;

2. The penetration-based model is widely used for haptic and real-time simulations. The literature for the efficient computation of penetration information is massive and existing methods for volumetric models are extremely efficient. However, penetration-based approaches suffer from singularities when penetrations are too deep. Indeed, if the medial axis of the penetrating objects intersect, the contact kinematics are no longer well-defined and nonsmooth changes of the contact normals may occur. Moreover, penetration-based approaches require systematically the stabilization of contact constraints which may introduce potentially large ammounts of fictitious energy into the system. This is particularly problematic from a stability standpoint for simulations with small mechanical clearances.

Therefore, selecting a geometric contact model adapted to the targeted application is important since it strongly influences the accuracy and performance of the overall simulation.

In addition, a suitable representation of the geometric models of B_k has to be chosen as well. Discrete volume representations have been discussed to be less accurate though they lead to fast computations of penetration informations. Polyhedral representations are the current *de facto* choice for a good balance between accuracy and performance of the distance and penetration depth computation. However, contact constraints built from such representations approximate the feasible space, which causes some simulations to require prohibitively detailed approximations in order to achieve a satisfying level of accuracy at the cost of high computation times for

CD and a high number of contact constraints. Finally, smooth BRep representations provide the solver with an optimal accuracy of B_k, thus allowing the feasible space to be represented accurately with few constraints, only. Distances between two BRep models however, are hardly computed efficiently.

After discussing the types of applications targeted by the framework described in this manuscript, the choice of maximizing the accuracy of the contact constraints has been made. Indeed, our framework will be based on the computation of LMD or quasi-LMD between smooth BRep models. It is integrated as part of the complete real-time multibody dynamics simulation engine XDE [91]. Existing algorithms computing closest points between points, curves and surfaces have been presented but efficient methods for complete BRep models do not exist yet. Therefore, the objectives of the proposed approach enumerate:

- Design new data structures in order to accelerate the identification of feature areas of smooth (implicitly dilated) BRep models or deformable curves that may contain LMDs (or quasi-LMDs) smaller than a user-defined threshold d_{max}. As analyzed in Section 1.5, robust solution location mechanisms is a clear issue to the improvement of CD during insertion tasks with small clearances. To this end, our contributions are the definition of curvature-based compatibility attributes (see Section 2.2), supermaximal features (see Section 2.3), and feature splitting into quasi-flat parts at pre-computation time (see Section 2.4.1). Moreover, we define a new type of cone for bounding directions (like normals and tangents): the polyhedral cones (see Chapter 3). The dilation of polyhedral cones is also supported (see Section 3.4) in order to allow algorithms to compute quasi-LMDs;

- Select and improve existing methods for the exact computation of the LMD footpoints. Once again, Section 1.5 has shown that the computation of exact solutions is often slow and inadequate for real-time simulations. As a contribution, we present a region selection that depends on the types of features involved in the LMD computation (see Section 4.3) and a hybrid approach for deformable Bézier curves (see Section 4.3.1). The goal is to be as fast as possible while remaining sufficiently robust;

- Combine those new data structures and methods with existing approaches in order to build a CD framework able to handle complete BRep models as well as deformable curves, efficiently. Therefore, our new polyhedral tangent cones, supermaximal features, and compatibility attributes are added to a bounding volume hierarchy at pre-computation time (see Section 2.4.2). The traversal of the BVH and the implication of the added informations are presented in Section 4.2. The exploitation of temporal coherence (see Section 4.5) and the parallelization (see Section 4.6) of the whole CD pipeline are addressed as well in order to achieve real-time performances.

Finally, our framework is tested in the Chapter 5 on industrial scenarios for the real-time simulation of insertion tasks with small mechanical clearances.

Chapter 2

From the CAD model to a data structure for distance computation

Pre-computing data structures is necessary to transform the input geometric data into a representation that can be efficiently explored at runtime to compute the LMDs between BRep models after they have been arbitrarily moved in space. This chapter justifies the selection and describes the construction of a structure designed for representing the input BRep models: a bounding volume hierarchy filled with spatial bounding volumes (oriented bounding boxes), orientation-based bounding volumes (normal cones), and compatibility attributes. To that end, the input models are first analyzed in order to identify areas of the model that play no role with respect to collision detection, and pairs of features that have no chance of ever being in contact, so that they can be ignored at runtime. In addition, the new concept of supermaximal features is introduced to group BRep features with similar geometric shapes into a single one and avoid redundant computations at runtime. Finally, all remaining features are split into almost-flat areas and the tree structure hierarchizes them. Each such structure is then ready to be input to the runtime phase of the framework as detailed in the subsequent chapters.

2.1 Why pre-computing a data structure is necessary

Let $N_{\mathcal{A}}$ and $N_{\mathcal{B}}$ be the number features on the BRep modeling the shapes $\mathcal{A}$ and $\mathcal{B}$ of two bodies B_k and B_l. An example of a naive approach to find all LMD between them could be to:

1. Form each possible pair of features $(S^i_{\mathcal{A}}, S^j_{\mathcal{B}})$ where $S^i_{\mathcal{A}}$ is the i-th feature of $\mathcal{A}$ with $i \in \{1, \dots, N_{\mathcal{A}}\}$, and $S^j_{\mathcal{B}}$ the j-th feature of $\mathcal{B}$ with $j \in \{1, \dots, N_{\mathcal{B}}\}$;

2. For each such pair of features, compute all critical points between their underlying untrimmed geometry (curve, surface or point) using a method capable of finding all the critical points of the corresponding distance function. Examples of such methods, like subdivision-based methods, were presented in Section 1.5;

3. Filter the critical points computed between the untrimmed geometry, i.e., remove any point that is not a LMD, remove any LMD that is larger than the user-defined threshold d_{max}. Also, take into account the BRep structure to check that they do not lie in a hole and that the characterization with Equation (1.23) is verified. More details about this step are provided in Section 4.4.

While functional and implementable using methods already available in the literature, this approach would lead to very poor performances in practice. Indeed, the steps (2) and (3) have to be executed between all $(N_{\mathcal{A}} \times N_{\mathcal{B}})$ pairs of features, even if some of them are too far away from one another to be of interest from the dynamics simulation standpoint. Step (2) requires a minimization algorithm that may be very computationally intensive. For example, subdivision methods (see Section 1.5.1) rely on the recursive subdivision of curves and surfaces which are costly if performed at run-time. Finally, some critical points will be unnecessarily computed because they will be removed at step (3).

Pre-computed data structures aim at reducing those computation times based on an analysis of the input models and additional informations on top of the BRep structures to reduce the amount of work being performed at run-time. Typically, bounding volumes arranged into bounding volume hierarchies can be used to determine lower bounds on the distance between two features. In practice, these hierarchies reduce the quadratic complexity of the naive algorithm presented above since whole groups of features can be detected at once as being too far from one another. This is the purpose of the *culling test* presented in Section 2.4.3. Bounding volumes as well as the computation of the bounding volume hierarchy (BVH) for smooth BRep models are detailed in Section 2.4. Moreover, during the construction of the BVH, the features themselves are split into small areas that are almost flat to exploit local solution search algorithms instead of global methods. Also, this subdivision process helps to identify in advance subsets of the curves and surfaces domains that lie into a hole (See Section 2.4.1 for details).

In addition and because the pre-computation phase of a simulation is executed only once, it must incorporate as much analysis of the BRep model as possible in order to:

- Replace any NURBS curve/surface with its equivalent set of rational Bézier curves/surfaces. This contributes to lowering run-time computation costs since evaluating a NURBS is more expensive that evaluating a Bézier curve;

- Identify pairs of features that can never contain any LMD, independently from their spatial position and orientation. This can be achieved by defining curvature-based *compatibility attributes* as described in Section 2.2.

- Identify and group similar shapes so that, e.g., faces with the same geometric representations are considered as a single one. Those groups form *supermaximal features* as introduced in Section 2.3.

The Figure 2.1 summarizes all the operations performed at pre-computation time in order to produce a BVH. It has to be mentioned that the use of BVH is already very popular for CD between tessellated models [7, 41].

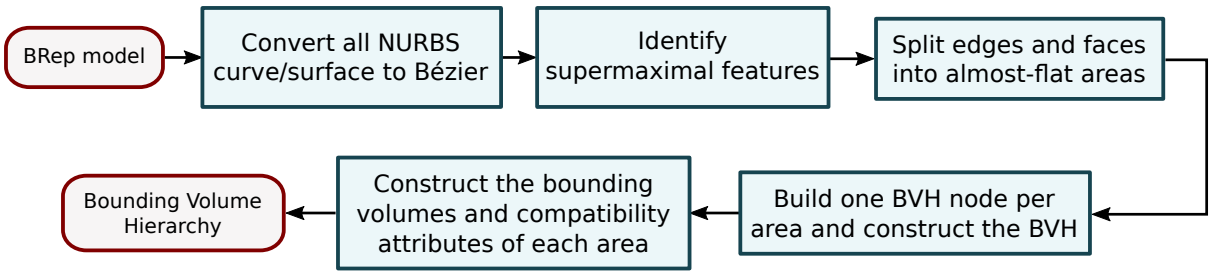


Figure 2.1: Successive steps to produce a Bounding Volume Hierarchy from a BRep model.

2.2 Curvature-based surface compatibility

It can be proven than no LMD can be found between some types of surfaces, independently from their relatives positions. Indeed, this sections aims at proving that, e.g., any critical point of the distance function between two concave cylinders will end up not being an LMD so finding them is not necessary in the first place.

Let $F_{\mathcal{A}}$ and $F_{\mathcal{B}}$ be two faces of BRep models describing $\mathcal{A}$ and $\mathcal{B}$, respectively. Let $\mathbf{p}^*_{\mathcal{A}}$ and $\mathbf{p}^*_{\mathcal{B}}$ be the critical points of the distance function satisfying the Equation (1.23), i.e.:

$$\mathbf{p}^*_{\mathcal{B}} - \mathbf{p}^*_{\mathcal{A}} \in T_{\mathcal{A}}(\mathbf{p}^*_{\mathcal{A}})^* \cap -T_{\mathcal{B}}(\mathbf{p}^*_{\mathcal{B}})^* \tag{2.1}$$

$$\iff \begin{cases} \langle \mathbf{p}^*_{\mathcal{B}} - \mathbf{p}^*_{\mathcal{A}}, \mathbf{n}_{\mathcal{A}} \rangle &= d, \\ \langle \mathbf{p}^*_{\mathcal{B}} - \mathbf{p}^*_{\mathcal{A}}, \mathbf{n}_{\mathcal{B}} \rangle &= -d, \end{cases} \tag{2.2}$$

where $d = \|\mathbf{p}^*_{\mathcal{B}} - \mathbf{p}^*_{\mathcal{A}}\| > 0$ and $\mathbf{n}_{\mathcal{A}}$ (resp. $\mathbf{n}_{\mathcal{B}}$) is the normal of $F_{\mathcal{A}}$ (resp. $F_{\mathcal{B}}$) at $\mathbf{p}^*_{\mathcal{A}}$ (resp. $\mathbf{p}^*_{\mathcal{B}}$).

Now, let us consider the plane $\mathbf{P}$ with a normal vector $\mathbf{n}_{\mathbf{P}}$ orthogonal to $\mathbf{n}_{\mathcal{A}}$ and $\mathbf{n}_{\mathcal{B}}$ and such that $\mathbf{p}^*_{\mathcal{A}} \in \mathbf{P}$ and $\mathbf{p}^*_{\mathcal{B}} \in \mathbf{P}$. Then, the intersection of $\mathbf{P}$ with $F_{\mathcal{A}}$ (resp. $F_{\mathcal{B}}$) yields a curve with parametrization $\gamma_{\mathcal{A}} : \mathbb{R} \to \mathbb{R}^3$ (resp. $\gamma_{\mathcal{B}}$) which can be assumed, without loss of generality, to be such that $\gamma_{\mathcal{A}}(0) = \mathbf{p}^*_{\mathcal{A}}$ (resp. $\gamma_{\mathcal{B}}(0) = \mathbf{p}^*_{\mathcal{B}}$), and to have an arc-length parametrization, i.e., their tangents always have a unit magnitude (including at 0). Thus:

$$\|\gamma'_{\mathcal{A}}(0)\| = \|\gamma'_{\mathcal{B}}(0)\| = 1. \tag{2.3}$$

Those elements are illustrated in Figure 2.2.

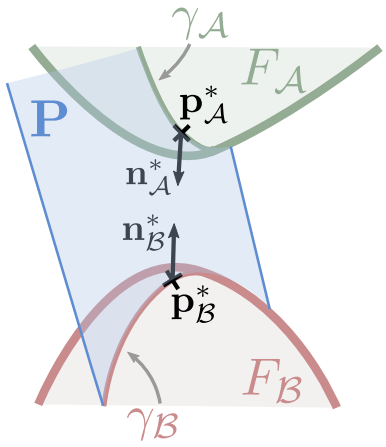


Figure 2.2: Example of configuration with two critical points $\mathbf{p}^*_{\mathcal{A}}, \mathbf{p}^*_{\mathcal{B}}$, the corresponding outward normals $\mathbf{n}_{\mathcal{A}}, \mathbf{n}_B$, and a choice of plane $\mathbf{P}$ containing the line passing through $\mathbf{p}^*_{\mathcal{A}}$ and $\mathbf{p}^*_{\mathcal{B}}$ and the curves $\gamma_{\mathcal{A}}$ and $\gamma_{\mathcal{B}}$

By definition of the normal curvature [23], the second derivatives of $\gamma_{\mathcal{A}}$ and $\gamma_{\mathcal{B}}$ are such that:

$$\gamma''_{\mathcal{A}}(0) = -\mathbf{n}_{\mathcal{A}}\kappa_{\mathcal{A}}, \tag{2.4}$$

$$\gamma''_{\mathcal{B}}(0) = -\mathbf{n}_{\mathcal{B}}\kappa_{\mathcal{B}}, \tag{2.5}$$

where $\kappa_{\mathcal{A}}$ (resp. κ_B) designate the normal curvature of $F_{\mathcal{A}}$ (resp. $F_{\mathcal{B}}$) at $\mathbf{p}^*_{\mathcal{A}}$ (resp. $\mathbf{p}^*_{\mathcal{B}}$) along the tangent direction $\gamma'_{\mathcal{A}}(0)$ (which is the same as the direction of $\gamma'_{\mathcal{B}}(0)$). Let us point out the minus sign in Equations (2.4) and (2.5) that come from the fact that the orientation convention chosen in Section 1.4.3 regarding the solid boundary is such that their normals point outward the solid.

Clearly, for $\mathbf{p}^*_{\mathcal{A}}$ and $\mathbf{p}^*_{\mathcal{B}}$ to be local minimizers of the distance function between $F_{\mathcal{A}}$ and $F_{\mathcal{B}}$, they must also be the local minimizers of the squared distance function $d_{\mathbf{P}}(s,t) = \|\gamma_{\mathcal{B}}(s) -$

$\gamma_{\mathcal{A}}(t)\|^2$ between the curves $\gamma_{\mathcal{A}}$ and $\gamma_{\mathcal{B}}$ obtained for any choice of $\mathbf{P}$, as defined above. Let us assume $\mathbf{P}$ fixed; $\mathbf{p}^*_{\mathcal{A}}$ and $\mathbf{p}^*_{\mathcal{B}}$ are local minimizers of $d_{\mathbf{P}}$ if and only if its Hessian $\mathcal{H}(d_{\mathbf{P}})$ at zero, i.e.:

$$\begin{aligned}\mathcal{H}(d_{\mathbf{P}})(0,0) &= 2\begin{bmatrix}\langle\gamma''_{\mathcal{A}}(0),\gamma_{\mathcal{A}}(0)-\gamma_{\mathcal{B}}(0)\rangle+\|\gamma'_{\mathcal{A}}(0)\|^2 & -\langle\gamma'_{\mathcal{A}}(0),\gamma'_{\mathcal{B}}(0)\rangle\\ -\langle\gamma'_{\mathcal{B}}(0),\gamma'_{\mathcal{A}}(0)\rangle & -\langle\gamma''_{\mathcal{B}}(0),\gamma_{\mathcal{A}}(0)-\gamma_{\mathcal{B}}(0)\rangle+\|\gamma'_{\mathcal{B}}(0)\|^2\end{bmatrix},\\ &= 2\begin{bmatrix}d\kappa_{\mathcal{A}}+1 & \pm 1\\ \pm 1 & d\kappa_{\mathcal{B}}+1\end{bmatrix},\end{aligned} \tag{2.6}$$

is symmetric definite positive (SDP). The actual sign of the off-diagonal elements depends on whether $\gamma'_{\mathcal{A}}(0)$ and $\alpha'_{\mathcal{B}}(0)$ point toward the same direction or not (though in any case, they must be collinear since they are both orthogonal to $\mathbf{n}_{\mathbf{P}}$ and to $(\mathbf{p}^*_{\mathcal{B}}-\mathbf{p}^*_{\mathcal{A}})$).

Let us recall that both eigenvalues of $\mathcal{H}(d_{\mathbf{P}})(0,0)$ must be positive for it to be SDP. Therefore, a necessary and sufficient condition for $\mathcal{H}(d_{\mathbf{P}})(0,0)$ to be SDP is to have a positive trace and determinant, i.e.:

$$d(\kappa_{\mathcal{A}}+\kappa_{\mathcal{B}}) > -2, \tag{2.7}$$

$$(d\kappa_{\mathcal{A}}+1)(d\kappa_{\mathcal{B}}+1)-1 > 0. \tag{2.8}$$

In particular, Equation (2.8) can be developed, under the assumption that $d \neq 0$, as:

$$\kappa_{\mathcal{A}}+\kappa_{\mathcal{B}} > -d\kappa_{\mathcal{A}}\kappa_{\mathcal{B}}. \tag{2.9}$$

The following properties depend on the signs of $\kappa_{\mathcal{A}}$ and $\kappa_{\mathcal{B}}$ only and can be deduced from Equations (2.7) and (2.9):

- If $\kappa_{\mathcal{A}} > 0$ and $\kappa_{\mathcal{B}} \geq 0$ (or $\kappa_{\mathcal{A}} \geq 0$ and $\kappa_{\mathcal{B}} > 0$), then $\mathcal{H}(d_{\mathbf{P}})(0,0)$ is SDP since both conditions are verified;

- If $\kappa_{\mathcal{A}} = 0$ and $\kappa_{\mathcal{B}} \leq 0$ (or $\kappa_{\mathcal{A}} \leq 0$ and $\kappa_{\mathcal{B}} = 0$), then $\mathcal{H}(d_{\mathbf{P}})(0,0)$ is not SDP because the Equation (2.9) is violated. Indeed, assuming $\kappa_{\mathcal{A}} = 0, \kappa_{\mathcal{B}} \leq 0$:

$$\kappa_{\mathcal{A}}+\kappa_{\mathcal{B}} > -d\kappa_{\mathcal{A}}\kappa_{\mathcal{B}}, \tag{2.10}$$

$$\Leftrightarrow \kappa_{\mathcal{B}} > 0, \quad \text{which is contradictory.} \tag{2.11}$$

- If $\kappa_{\mathcal{A}}$ and $\kappa_{\mathcal{B}}$ are both negative (and non-zero), then $\mathcal{H}(d_{\mathbf{P}})(0,0)$ is not SDP because Equations (2.7) and (2.9) cannot be satisfied simultaneously. Indeed we have:

$$\begin{cases} d(\kappa_{\mathcal{A}}+\kappa_{\mathcal{B}}) & > -2,\\ \kappa_{\mathcal{A}}+\kappa_{\mathcal{B}} & > -d\kappa_{\mathcal{A}}\kappa_{\mathcal{B}},\end{cases} \tag{2.12}$$

$$\Leftrightarrow \begin{cases} d(\kappa_{\mathcal{A}}+\kappa_{\mathcal{B}})\kappa_{\mathcal{A}}\kappa_{\mathcal{B}} & > -2\kappa_{\mathcal{A}}\kappa_{\mathcal{B}},\\ (\kappa_{\mathcal{A}}+\kappa_{\mathcal{B}})^2 & < -d\kappa_{\mathcal{A}}\kappa_{\mathcal{B}}(\kappa_{\mathcal{A}}+\kappa_{\mathcal{B}}),\end{cases} \tag{2.13}$$

$$\Leftrightarrow \begin{cases} d(\kappa_{\mathcal{A}}+\kappa_{\mathcal{B}})\kappa_{\mathcal{A}}\kappa_{\mathcal{B}} & > -2\kappa_{\mathcal{A}}\kappa_{\mathcal{B}},\\ d(\kappa_{\mathcal{A}}+\kappa_{\mathcal{B}})\kappa_{\mathcal{A}}\kappa_{\mathcal{B}} & < -(\kappa_{\mathcal{A}}+\kappa_{\mathcal{B}})^2,\end{cases} \tag{2.14}$$

$$\Rightarrow -2\kappa_{\mathcal{A}}\kappa_{\mathcal{B}} < -(\kappa_{\mathcal{A}}+\kappa_{\mathcal{B}})^2, \tag{2.15}$$

$$\Rightarrow -2\kappa_{\mathcal{A}}\kappa_{\mathcal{B}} < -2\kappa_{\mathcal{A}}\kappa_{\mathcal{B}}-\kappa^2_{\mathcal{A}}-\kappa^2_{\mathcal{B}}, \tag{2.16}$$

$$\Rightarrow 0 < -\kappa^2_{\mathcal{A}}-\kappa^2_{\mathcal{B}}, \quad \text{which is contradictory.} \tag{2.17}$$

Now, those developments can be used to prove that certain pairs of canonical surfaces never produce any LMD. To this end, the canonical surfaces are categorized in Table 2.1 with respect to the sign of the principal curvatures at each of their points. Note that the surface types have to be understood following the typology of canonical surfaces shown in Figure 1.26, i.e., this conforms to the concept of orientation index [82] mentioned in Section 1.4.3.

Surface type	The principal curvatures at each point are...
Plane	... both zero.
Convex sphere	... both positive.
Convex cone or cylinder	... one is zero while the other is positive.
Convex torus	... both positive or have opposite signs.
Concave torus	... both negative or have opposite signs.
Concave cone, cylinder or sphere	... both non-positive.

Table 2.1: Principal curvatures of the points of all the types of canonical surfaces.

Let us assume that $F_{\mathcal{A}}$ and $F_{\mathcal{B}}$ are represented by two non-intersecting canonical surfaces. If there exists a relative position of $F_{\mathcal{A}}$ and $F_{\mathcal{B}}$ such that a LMD with footpoints $\mathbf{p}^*_{\mathcal{A}}$ and $\mathbf{p}^*_{\mathcal{B}}$ can be found, then the conditions given by Equations (2.7) and (2.8) must hold for all choices of planes $\mathbf{P}$ as previously defined, i.e., with $\mathbf{P}$ containing the line passing through $\mathbf{p}^*_{\mathcal{A}}$ and $\mathbf{p}^*_{\mathcal{B}}$. Conversely, if for all possible relative positions of $F_{\mathcal{A}}$ with respect to $F_{\mathcal{B}}$, all pairs of critical points satisfying Equation (1.23) are such that there exists a choice of $\mathbf{P}$ such that the conditions given by Equations (2.7) and (2.8) are not validated, then none of them are LMD footpoints and no LMD can be found between $F_{\mathcal{A}}$ and $F_{\mathcal{B}}$, no matter their relative positions. In particular:

- No LMD can be found between two surfaces with all points having two non-positive principal curvatures. Indeed, the principal curvatures $\kappa_{min}, \kappa_{max}$ of a surface point correspond to the minimal and maximum normal curvature values along all possible tangent direction [23]. Therefore, if any two critical points $\mathbf{p}^*_{\mathcal{A}}$ and $\mathbf{p}^*_{\mathcal{B}}$ are found between $F_{\mathcal{A}}$ and $F_{\mathcal{B}}$, then any choice of $\mathbf{P}$ will yield two intersection curves $\gamma_{\mathcal{A}}$ and $\gamma_{\mathcal{B}}$ with normal curvatures $\kappa_{\mathcal{A}} \in [\kappa^{\mathcal{A}}_{min}, \kappa^{\mathcal{A}}_{max}]$ and $\kappa_{\mathcal{B}} \in [\kappa^{\mathcal{B}}_{min}, \kappa^{\mathcal{B}}_{max}]$ where $\kappa^{\mathcal{A}}_{min}, \kappa^{A}_{max}$ (resp. $\kappa^{\mathcal{B}}_{min}, \kappa^{B}_{max}$) are the principal curvatures of $F_{\mathcal{A}}$ (resp. $F_{\mathcal{B}}$) at the point $\mathbf{p}^*_{\mathcal{A}}$ (resp. $\mathbf{p}^*_{\mathcal{B}}$). Thus, since $\kappa^{\mathcal{A}}_{max} \leq 0$ and $\kappa^{\mathcal{B}}_{max} \leq 0$, we have $\kappa_{\mathcal{A}} \leq 0$ and $\kappa_{\mathcal{B}} \leq 0$. The developments from Equation (2.10) to Equation (2.11) or from Equation (2.12) to Equation (2.17) are then applicable to prove that the SDP conditions Equations (2.7) and (2.9) are violated;

- Following a similar reasoning, no LMD can be found between, e.g., a convex cylinder (or convex cone) with a plane since, following the same definitions as in the preceding developments, for each possible pair of critical points $\mathbf{p}^*_{\mathcal{A}}, \mathbf{p}^*_{\mathcal{B}}$, there exists a plane $\mathbf{P}$ such that $\kappa_{\mathcal{A}} = \kappa_{\mathcal{B}} = 0$ resulting in the violation of the condition of Equation (2.9). However, non-isolated minimizers corresponding to conformal contacts may exist.

Given the previous observations, each feature area of a shape is assigned *compatibility attributes*. Those attributes indicate whether or not a given pair of surfaces can contain any LMD at all. The Table 2.2 summarizes the incompatible pairs of surfaces. Convex spheres and convex tori are not included in this table since they are compatible with all the canonical surface types.

This concept of compatibility can be implemented extremely efficiently with the help of bit masks. An initialization phase assigns to each surface type a distinct *feature type ID* represented as integers. Convex canonical surface types do not have the same identifier than their concave counterpart, e.g., the identifier assigned to the 'convex sphere' surface type is not the same that of the 'concave sphere' surface type. Overall, we distinguish nine types of canonical surfaces are

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

Table 2.2: Compatibilities between different types of surfaces . 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.

illustrated by Figure 1.26, therefore the feature type IDs only range from 0 to 8. With those IDs at hand, an algorithm assigns to each feature area two bit masks:

1. A *type mask* with a 1 set to the position indicated by the feature type ID and zeros elsewhere;

2. A *compatibility mask* with 1 set to the positions indicated by all the feature type IDs it can collide with. For example, assuming the index 1 (resp. 5) corresponds to cylinders (resp. concave spheres), then the first bit of the cylinder's compatibility mask will be 1 because two convex cylinders may collide while its fifth entry will be 0 because a cylinder and a concave sphere cannot collide.

At runtime, those masks produce efficient compatibility tests between two nodes of the BVH. Indeed, algorithms only have to perform a bitwise *or* between the type mask of each node and the compatibility mask of the other. If any result is non-zero, then the nodes are compatible and the tree traversal can proceed under those nodes. If it is zero then, the nodes are incompatible and can be ignored.

Finally, the compatibility masks derived in this section solely take into account the type of canonical surface, that is, independently from their relative positions and dimensions. Filtering could be further improved with their incorporation the masks. For example, it seems clear that no LMD is possible between a sphere with a radius equal to $r_{\mathcal{A}}$ and a concave torus with minor radius equal to $r_{\mathcal{B}} < r_{\mathcal{A}}$. Indeed, at any critical point, a plane $\mathbf{P}$, as defined at the beginning of this section, could be chosen such that Equation (2.9) is violated for any choice of $d > 0$. All the criteria for each possible pair of canonical surfaces are yet to be determined to extend the use of the bit masks and the efficiency of curvature-based criterion described in this section.

Furthermore, the conditions expressed by Equations (2.7) and (2.9) depend on the distance d between the critical points. Therefore, bounds on the principal curvatures of all points of each face $F_{\mathcal{A}}$ and $F_{\mathcal{B}}$ combined with bounds of the distance separating them could be used to design a curvature-based culling test that takes into account the relative position of these faces. However, bounding the distance, e.g., using bounding volumes, proved either too inaccurate or too expensive in practice to be of any use to set up such an extended curvature-based test. Further works may lead to better results following these proposals.

2.3 Introducing supermaximal features to avoid redundant computations

This section introduces the new notion of *supermaximal features* that are geometric entities built on top of the initial BRep CAD structure imported into the CD framework. The supermaximal

features group features with similar geometric shapes. Their definition and identification methods are given in Section 2.3.1 and the corresponding data structures in Section 2.3.2. These data structures are exploited to avoid redundant computations at runtime in Section 4.2.3.

2.3.1 Definition and identification

Because of topological restrictions prescribed by their CW-complex structure, BRep models often contain disjoint areas of the same surface or curve that appear as several independent geometric entities (see faces $F_{\mathcal{A}}^{7}$ and $F_{\mathcal{A}}^{8}$ in Figure 2.3a). Such configurations derive from the geometric modelling process producing each body $B_k \in MS$. Indeed, the basic principle of a solid modelling process combines solid primitives that repeatedly trim surfaces of ∂B_k, hence the existence of adjacent and/or disjoint areas of the same surface in the BRep CAD model input for each B_k.

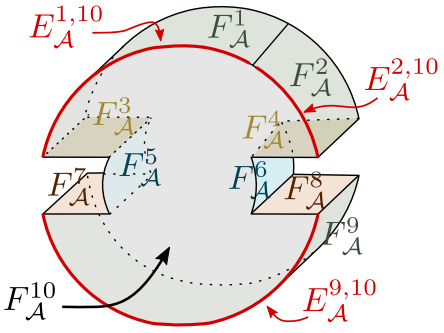
These areas coincide with faces or edges of the BRep data structure. Formally, given the BRep model $\mathcal{A}$ of a B_k, the i^{th} set ${}^{s}F_{\mathcal{A}}^{i}$ of possibly disjoint areas of the same canonical surface consisting of the BRep face set $\{F_{\mathcal{A}}^{l}, F_{\mathcal{A}}^{m}, \ldots, F_{\mathcal{A}}^{n}\}$, sharing the same orientation such that all entities of ${}^{s}F_{\mathcal{A}}^{i}$ share the same intrinsic geometric properties, forms a *supermaximal face*. In other words, these areas are embedded into the same untrimmed surface. Examples are given in Figure 2.3. One of the supermaximal faces of $\mathcal{A}$ is ${}^{s}F_{\mathcal{A}}^{1} = \{F_{\mathcal{A}}^{1}, F_{\mathcal{A}}^{2}, F_{\mathcal{A}}^{9}\}$ because these faces are areas of the same unbounded cylinder. The Figure 2.3c lists all the supermaximal faces of $\mathcal{A}$ (shown in Figure 2.3a).

Similarly, the j^{th} set ${}^{s}E_{\mathcal{A}}^{j}$ composed of the edges, with possible disjoint curves, $\{E_{\mathcal{A}}^{l}, E_{\mathcal{A}}^{m}, \ldots, E_{\mathcal{A}}^{n}\}$, at the intersection of two supermaximal faces ${}^{s}F_{\mathcal{A}}^{i}$ and ${}^{s}F_{\mathcal{A}}^{k}$ which are part of the same connected component of the intersection between the underlying untrimmed surfaces is a *supermaximal edge*. In other words, these curves are embedded into a single (continuous) intersection curve between two untrimmed surfaces. Examples are given in Figure 2.3. The supermaximal edges of $\mathcal{A}$ (shown in Figure 2.3a) that are not singletons are listed in Figure 2.3d (with $E_{\mathcal{A}}^{i,j}$ denoting the edges at the intersection of $F_{\mathcal{A}}^{i}$ and $F_{\mathcal{A}}^{j}$). One of those supermaximal edges is ${}^{s}F_{\mathcal{A}}^{5} = \{E_{\mathcal{A}}^{1,10}, E_{\mathcal{A}}^{2,10}, E_{\mathcal{A}}^{9,10}\}$ (a circle) because all its component edges lie at the intersection of the two supermaximal faces ${}^{s}F_{\mathcal{A}}^{1} = \{F_{\mathcal{A}}^{1}, F_{\mathcal{A}}^{2}, F_{\mathcal{A}}^{9}\}$ (green cylinder) and ${}^{s}F_{\mathcal{A}}^{5} = \{F_{\mathcal{A}}^{10}\}$ (gray plane).

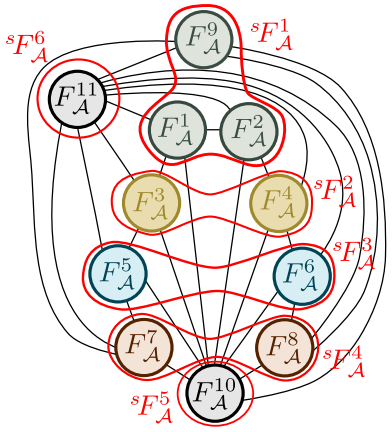
It has to be mentioned that *supermaximal faces* and *supermaximal edges* are somewhat similar to the *maximal edges* and *maximal surfaces* introduced by Li et al. [82] and Boussuge et al. [19] without requiring their constitutive elements to be adjacent into the BRep structure of $\mathcal{A}$. The term *supermaximal feature* designates either a supermaximal face or a supermaximal edge.

Finally, even if this definition of supermaximal edges is sufficiently general to include Bézier curves and surfaces, the current interest holds in the identification of supermaximal edges with canonical curves (Line, Circle, Ellipse, Hyperbola, Parabola) and canonical surfaces in order to apply the runtime optimization proposed in Section 4.2.3.

All the ${}^{s}F_{\mathcal{A}}^{i}$ and ${}^{s}E_{\mathcal{A}}^{j}$ can be generated at precomputation-time for each category of canonical surface without any knowledge about the construction tree of the solid $\mathcal{A}$ and in $O(n^2)$ time without any particular algorithm optimization. n stands for the number of edges or surfaces of a targeted category. Figure 2.4 shows the intrinsic properties that must be checked for each type of canonical face. It is important to point out that these supermaximal features, ${}^{s}F_{\mathcal{A}}^{i}$ or ${}^{s}E_{\mathcal{A}}^{j}$, are set up independently from the underlying parametrization of each constitutive area, $F_{\mathcal{A}}^{l}$ or $E_{\mathcal{A}}^{p}$. By definition of ${}^{s}F_{\mathcal{A}}^{i}$, all its constitutive faces are either concave or convex.



(a)



(b)

Supermaximal face	Surface type
${}^{s}F_{\mathcal{A}}^{1} = \{F_{\mathcal{A}}^{1}, F_{\mathcal{A}}^{2}, F_{\mathcal{A}}^{9}\}$	Cylinder
${}^{s}F_{\mathcal{A}}^{2} = \{F_{\mathcal{A}}^{3}, F_{\mathcal{A}}^{4}\}$	Plane
${}^{s}F_{\mathcal{A}}^{3} = \{F_{\mathcal{A}}^{5}, E_{\mathcal{A}}^{6}\}$	Cylinder
${}^{s}F_{\mathcal{A}}^{4} = \{F_{\mathcal{A}}^{7}, F_{\mathcal{A}}^{8}\}$	Plane
${}^{s}F_{\mathcal{A}}^{5} = \{F_{\mathcal{A}}^{10}\}$	Plane
${}^{s}F_{\mathcal{A}}^{6} = \{F_{\mathcal{A}}^{11}\}$	Plane

(c)

Supermaximal edge	Curve type
${}^{s}E_{\mathcal{A}}^{1} = \{E_{\mathcal{A}}^{5,10}, E_{\mathcal{A}}^{6,10}\}$	Circle
${}^{s}E_{\mathcal{A}}^{2} = \{E_{\mathcal{A}}^{3,10}, E_{\mathcal{A}}^{4,10}\}$	Line
${}^{s}E_{\mathcal{A}}^{3} = \{E_{\mathcal{A}}^{7,10}, E_{\mathcal{A}}^{8,10}\}$	Line
${}^{s}E_{\mathcal{A}}^{4} = \{E_{\mathcal{A}}^{1,10}, E_{\mathcal{A}}^{2,10}, E_{\mathcal{A}}^{9,10}\}$	Circle
${}^{s}E_{\mathcal{A}}^{5} = \{E_{\mathcal{A}}^{5,11}, E_{\mathcal{A}}^{6,11}\}$	Circle
${}^{s}E_{\mathcal{A}}^{6} = \{E_{\mathcal{A}}^{3,11}, E_{\mathcal{A}}^{4,11}\}$	Line
${}^{s}E_{\mathcal{A}}^{7} = \{E_{\mathcal{A}}^{7,11}, E_{\mathcal{A}}^{8,11}\}$	Line
${}^{s}E_{\mathcal{A}}^{8} = \{E_{\mathcal{A}}^{1,11}, E_{\mathcal{A}}^{2,11}, E_{\mathcal{A}}^{9,11}\}$	Circle

(d)

Figure 2.3: (a) A solid $\mathcal{A}$ and its faces. The hidden planar face parallel to $F_{\mathcal{A}}^{10}$ will be referred to as $F_{\mathcal{A}}^{11}$ (not shown here). The three edges $E_{\mathcal{A}}^{1,10}, E_{\mathcal{A}}^{2,10}, E_{\mathcal{A}}^{9,10}$ are represented in red. (b) BRep face-edge adjacency graph structure of $\mathcal{A}$. All the supermaximal faces are circled in red. (c) List of all supermaximal faces and their underlying (untrimmed) surface type. (d) List of all supermaximal edges that are not singletons and their underlying (untrimmed) curve type.

Surface type	Grouping conditions
Planes	Parallel normals Other axes are coplanar.
Cylinders	Collinear principal axes Same radius.
Cones	Collinear principal axes Same apices and half-angles.
Torus	Collinear axis Other axes are coplanar Same center Same radii.
Spheres	Same center Same radius.

Figure 2.4: Canonical faces grouping conditions.

2.3.2 Data structures of supermaximal features

As described in Section 1.4.3, BRep faces domains $\mathcal{D}_i \in F_{\mathcal{A}}^{i}$ 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 $\Phi_i : \mathcal{D}_i \longrightarrow \mathbb{R}^3$ are global homeomorphisms, even for periodic surfaces. One of those loops is identified as the *external 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 ${}^sF^j_{\mathcal{A}}$ can be defined as the set of triplets $\{(\Phi_{ref}, \mathcal{D}^i_{ref}, \Xi_i) \mid i \in I\}$, 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 $\forall \mathbf{u} \in \mathbb{R}^2, \mathbf{u} \in \mathcal{D}^i_{ref} \iff \Phi_i^{-1}(\Phi_{ref}(\mathbf{u})) \in \mathcal{D}_i$. Ξ_i are sets of modeler-dependent per-feature attributes and metadata. This definition can be improved by observing that all $\mathcal{D}^i_{ref}$ delimit subsets of $\mathcal{D}_{ref}$. Therefore, ${}^sF^j_{\mathcal{A}}$ can be completely characterized 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* and $\coprod_{i\in I} \Xi_i$ is the disjoint union of all attributes and metadata.

Because a supermaximal face may have multiple connected components, the trimming curves over its domain do not follow the same rules as trimming curves do over the domain of a single face. Indeed, each component $F^i_{\mathcal{A}} \in {}^sF^j_{\mathcal{A}}$ is delimited by loops that cannot intersect but can be nested into each other. For any two areas $(F^i_{\mathcal{A}}, F^k_{\mathcal{A}})$ of ${}^sF^j_{\mathcal{A}}$, three configurations may arise:

- Both are non-nested as in Figure 2.5a;
- Either $F^i_{\mathcal{A}}$ is inside a hole of $F^k_{\mathcal{A}}$ as in Figure 2.5b or the opposite. The alternation between solid and empty areas is automatically handled by [119] detailed in Section 4.4.1 without any algorithmic modification as it relies on the concept of even-odd-rule which makes no assumption regarding the existence of an outer loop for classification;
- $F^i_{\mathcal{A}}, F^k_{\mathcal{A}}$, $i \neq k$, actually form a maximal face, i.e., they share at least one edge as in Figure 2.5c. If two trimming curves that define a common edge end up superimposed in the parametric domain then, they can be safely removed as their contributions will annihilate during point classification by the even-odd-rule.

The generalization to an arbitrary number of face areas is immediate. Let us observe that because faces of a BRep never intersect apart along their bounding edges and because superimposed trimming curves are removed, the boundary domain of $\mathcal{A}$ covered by ${}^sF^j_{\mathcal{A}}$ contains an arbitrary number of faces that can be merged but never overlap.

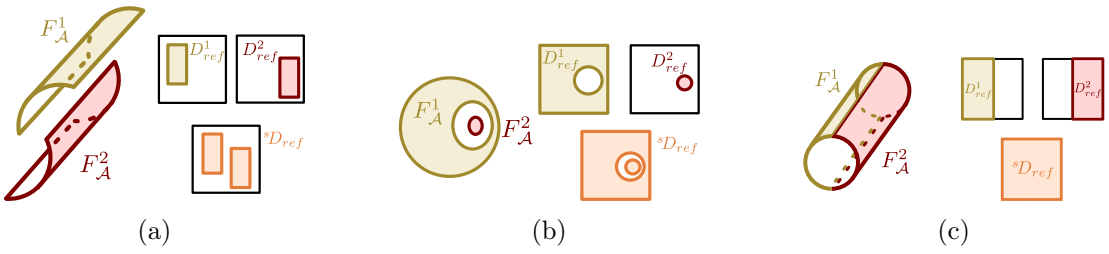
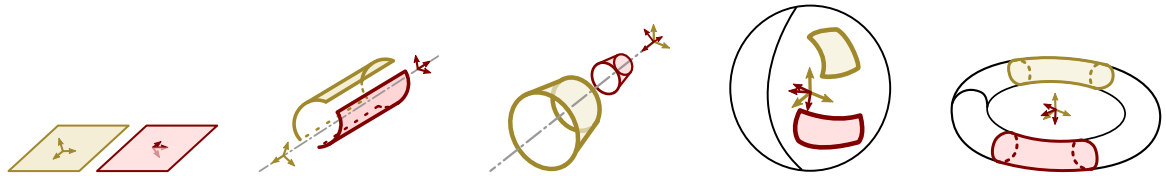


Figure 2.5: Three configurations for merging two parametric domains D^1_{ref} and D^2_{ref}. The resulting supermaximal domain ${}^sD_{ref}$ may have (a) two non-nested loops delimiting disjoint faces; (b) nested loops which alternately delimit boundary areas and holes; (c) only one loop: the two coinciding vertical trimming curves are no longer needed and have been removed.

Finally, given a domain of reference $\mathcal{D}_{ref}$, constructing the domain $\mathcal{D}^i_{ref}, i \in I$ requires the transformation of the trimming curves of $\mathcal{D}_i$ to ensure that both domains delimit the same boundary areas. Indeed, both faces may not have the same local coordinate systems because they may be generated by different operations during the construction process of $\mathcal{A}$ and each of these

operations may lead to two isometric surfaces that coincide. Even if the intrinsic properties of these surfaces do match, as shown in Figure 2.6, their reference frames may differ. The isometry depends on the reparametrization $\Phi_i^{-1} \circ \Phi_{ref}$ required to achieve the coincidence between these surfaces. The type of isometry required for each category of canonical face is summarized in Table 2.3.



(a) (b) (c) (d) (e)

Figure 2.6: Supermaximal faces composed of two areas which have different local coordinate systems (shown as arrows) even if their intrinsic geometric properties match.

Surface type	**Reparametrization type**
Planes	2D isometry and reflexion.
Cylinders	Translation of both parameters and reflection.
Cones	Translation of the angular parameter and reflection.
Torus	Translation of both parameters and reflection.
Spheres	2D isometry.

Table 2.3: Categories of re-parametrization for a change of local coordinate system of canonical surface areas belonging to the same supermaximal face.

The reasoning regarding supermaximal edges is similar to the case of supermaximal faces except that the types of re-parametrizations are necessarily shifts of a real interval.

To conclude, the data structure used for CD representing a supermaximal edge is given in Listing 2.1.

Listing 2.1: Data structure for one supermaximal edge.

```
struct SupermaximalEdge {
  Array<RealInterval> domain;
  Curve               reference_curve;
}
```

The structure `Array<RealInterval>` designates an array of real intervals, themselves represented by the structure on Listing 2.2. Each interval contains the domain of each curve forming the supermaximal edge. The reference curve corresponds to the curve with the reference parametrization Φ_{ref} identified in the previous discussion.

Listing 2.2: A real interval.

```
struct RealInterval {
  Real min;
  Real max;

  // Constructor.
  RealInterval(Real lowerBound, Real upperBound) {
    min = lowerBound;
    max = upperBound;
```

```
  }
}
```

The data structure for supermaximal faces is given by Listing 2.3. It is composed of a set of trimming curves that have been properly re-parametrized as per Table 2.3, and the reference surface with the reference parametrization Φ_{ref}.

Listing 2.3: Data structure for one supermaximal surface.

```
struct SupermaximalFace {
  Array<Curve2D> restriction_curves;
  Surface        reference_surface;
}
```

Let us note that the metadata Ξ mentioned in this section are of no use for CD and thus not inserted into those data structures.

2.4 Constructing the Bounding Volume Hierarchy (BVH)

The grouping of edges and faces into supermaximal entities defines a description of the boundary of each B_k with a minimal number of the B-Rep features: LMDs may be searched directly over these features rather than over every individual face or edge of the BRep model input. Further, to speed up CD queries between $\partial\mathcal{A}$ and $\partial\mathcal{B}$, they are embedded into a Bounding Volume Hierarchy (BVH) that takes into account not only the spatial extent of their vertices and supermaximal entities, but also their orientations.

2.4.1 Splitting features into quasi-flat areas and processing non simply connected domains

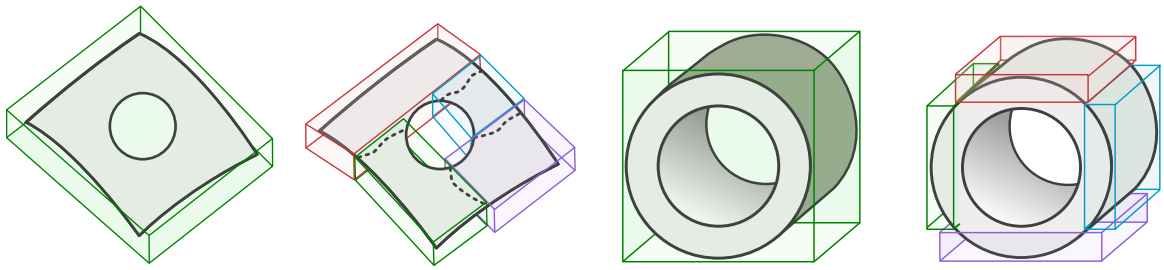
The objective of a BVH is to efficiently and accurately narrow down the location of the points of the BRep features defining LMDs. On the one hand, that is why polyhedron based approaches typically build a tree structure in such a way that each node contains at most one of its elementary geometric components, i.e., a triangle, and edge, or a point[1]. This is particularly efficient because those geometrical entities are generally quite small in term of spatial occupation. Moreover, they are all convex, implying that only one LMD can be found between any two of them, except when two segments are parallel or two triangles have parallel normals. Those last two cases are still easy to handle since they correspond to conformal configurations where the contact area a convex polyhedron and the corresponding contact constraints can be represented using a finite set of LMDs as discussed in Section 1.3.1.1.

On the other hand, the elementary geometric components of a BRep model are trimmed surfaces (for faces), trimmed curves (for edges), and points (for vertices). Often, they are both large in term of spatial occupation, and non-convex. Moreover, most of them contain holes. Using those as-is is thus extremely inadequate because they generate inefficient bounding volumes that are not tight enough and any two bounding volumes can potentially contain a large number of LMD footpoints. Therefore, solely using such nodes of the BVH to represent one feature of the BRep is not efficient. Instead, the domain of each feature can be split into smaller parts and each of them can be assigned to a node of the BVH. The subdivision should be such that:

[1]Most collision detection methods represent polyhedral models using a set of triangles without connection between them, i.e., a triangle soup. However, approaches like [90] also represent edges and vertices explicitly with their connections to obtain a simplicial complex structure.

- Each feature area assigned to one BVH node can be bounded tightly by one bounding volume. Such a bounding volume should be computed efficiently in order to avoid a large increase of pre-computation times;
- Any two such feature areas contain at most one pair of points forming a LMD because, as discussed in Section 4.2, some distance computation algorithms rely on numerical methods like Newton-Raphson, to obtain closest points between two features. The local nature of those methods leads us to generate leaves such that for any two of them, only one local solution exists, thus improving the robustness of the proposed approach. As much as possible, this generation process must rely on properties ensuring the desired uniqueness property.

Unfortunately, those two conditions are difficult to meet whenever non-convex BRep models are to be considered. Moreover, the second condition depends on the relative position of the two shapes, which cannot be predicted at pre-computation time. An experimentally satisfying method presented in the literature to head toward the satisfaction of these two assumptions is to ensure that each feature area bounded by a leaf is *almost flat* [70, 120]. This flatness may be obtained by subdividing the geometric feature into subsets that have a sufficiently small, up to a user-defined tolerance, tangent cone (for curves) or tangent cone polar (for surfaces). Figure 2.7 shows an example where a feature subdivision leads to much tighter bounding volumes. Here, they are chosen as Oriented Bounding Boxes (OBB). In practice, cones of revolution bounding the tangent cones (for curves) and tangent cone polars (for surfaces) are used to estimate the flatness as in [70]. If the apex-angle of the cone of revolution is smaller than a user-defined tolerance α_{max}, then the corresponding feature area is considered almost flat.



(a) (b) (c) (d)

Figure 2.7: (a) A surface with a hole, bounded by an OBB. (b) After subdivision, the parts are more tightly bounded by four OBBs and a part of the hole is not even inside any OBB. (c) A tube and its associated OBB touching its outermost cylinder. By definition, this OBB contains also all the other features located inside of the cylinder. (d) After being split into four parts, the outermost cylinder is bounded much more tightly with four OBB.

The subdivision process is a recursive procedure detailed by Algorithm 3 for the special case of a parametric surface S. The input of this algorithm is a parametric surface S with parameter bounds $(u_{min}, u_{max}, v_{min}, v_{max})$ characterizing the rectangular parametric domain of S. Its output `sub-domains` form a set of rectangular sub-domains R_i on which S is almost flat. Let us note that this is the most general approach that applies to Bézier surfaces as well. A dedicated process for each type of canonical surface can be set up since their curvature distribution can be fully characterized independently of their dimensions. For example, considering a cylinder defined on a rectangular parametric domain and such that, at each point the direction of principal curvature equal to 0 corresponds to the parametric direction u then, exactly $\frac{v_{max}-v_{min}}{\alpha_{max}}$ uniform subdivisions are needed. Indeed, the resulting areas will have v-coordinates that span a range of α_{max}, at most, and thus, a tangent cone polar inscribed into a cone of revolution of apex-angle equal α_{max}. Similar straightforward approaches apply to other canonical surfaces.

Approaches for curves are identical, except that the parametric domain to be split is one-dimensional only, and a bound of its tangent cone is used instead of a bound of its tangent cone polar.

Algorithm 3 Outputs to *sub-domains* the list of rectangular sub-domains resulting from the subdivision of the parametric domain of a given surface S such that each part is almost flat, i.e., has a tangent cone polar contained in a cone of revolution with apex-angle smaller than α_{max}.

function SplitSurface(S, $u_{min}, u_{max}, v_{min}, v_{max}$, sub-domains)
  $C \leftarrow$ cone of revolution bounding the tangent cone polar of S.
  if C has an apex angle greater than α_{max} **then**
    ▷ Cut the current domain into four parts.
    $u_{mid} \leftarrow \frac{u_{min}+u_{max}}{2}$
    $v_{mid} \leftarrow \frac{v_{min}+v_{max}}{2}$

    SplitSurface($S, u_{min}, u_{mid}, v_{min}, v_{mid}$)
    SplitSurface($S, u_{min}, u_{mid}, v_{mid}, v_{max}$)
    SplitSurface($S, u_{mid}, u_{max}, v_{min}, v_{mid}$)
    SplitSurface($S, u_{mid}, u_{max}, v_{mid}, v_{max}$)
  else
    Append $(u_{min}, u_{max}, v_{min}, v_{max})$ to *sub-domains*.
  end if
end function

The subdivision procedure described in Algorithm 3 does not take into account trimming curves, which are likely to make the domain of the surface non-rectangular. Two measures are taken in order to properly handle trimming curves:

1. The subdivision procedure is initialized with smallest and largest allowed parameters. This is equivalent to taking the Axis Aligned Bounding Box of the trimming curves;

2. Most of the remaining developments in this thesis (including the computation of bounding volumes) that apply to surfaces will work on rectangular domains only. Trimming curves will be ignored until the very end, i.e., they are taken into account only to determine if a LMD found between the untrimmed features is effectively valid, as described in Section 4.4.1.

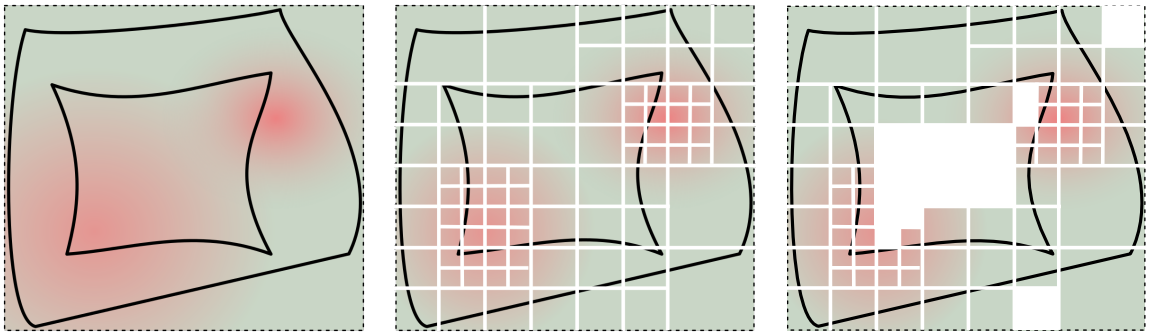
Figure 2.8 shows an example of domain subdivision of S with a curvature distribution shown with a color code that evolves from green to red. The closer to red a point on the parametric space is, the greater is the largest absolute curvature of the corresponding point on S. Recall that the absolute curvature κ_{abs} of a point $\mathbf{p}$ on S is defined as the sum of the absolute values of its principal curvatures κ_{min} and κ_{max} [23, 42]:

$$\kappa_{abs} = |\kappa_{min}| + |\kappa_{max}| \,. \tag{2.18}$$

It highlights how the subdivision algorithm is adaptive by generating more subdivisions on highly curved areas. Moreover, because trimming curves are not taken into account during the subdivision, some rectangular sub-domains output may lie completely outside the domain delimited by the trimming contours. Those sub-domains filled in white on Figure 2.8c are simply not included into the BVH because they do not define active contact areas of the CD process.

Checking whether a rectangular sub-domain, R, lies completely outside the domain delimited by a trimmed contour, T, is a two-steps process:

1. Check whether any of the edges of R intersects any curve of T. This is a one-variable root-finding problem that is straightforward to solve for simple curves (circles, lines, hyperbola, parabola). Iterative methods are needed for Bézier curves of degree higher than four.



(a) (b) (c)

Figure 2.8: (a) A set of trimming curves (black plain lines) and their AABB (dashed lines). Colors on the background indicate the curvature distribution of a surface defined over this domain. Red stands for high curvatures and green for low ones. (b) Example of domain subdivision resulting from Algorithm 3. (c) All the rectangular sub-domains filled in white are not included in the BVH because they fall outside the domain delimited by the trimmed contours.

Robustness is of foremost importance as the rejection of a R impacts the efficiency and accuracy of the whole framework. Indeed, rejecting erroneously R might prevent some contacts from being found, while retaining erroneously one induces unnecessary computations at runtime. Thus, a robust root isolation approach like the interval newton method [57] is preferred (and can be stopped as soon as one root can be proven to exist since its exact location is not needed);

2. If no such intersection exists, test whether any vertex of R lies inside T, or not. If none does then, the whole rectangular sub-domain is located outside and does not have to be included into the BVH.

Besides the fact that feature areas resulting from this subdivision are quasi-flat, it also makes OBB tighter in practice by preventing them from containing areas with holes, i.e., a sub-domain R containing a trimmed contour T, except when those holes are small enough to lie completely inside of a surface area that is almost flat. Those small holes will be ignored until the last stage of the CD procedure, i.e., when it is verified that LMD footpoints actually lie on the solid domain of some features (see Section 4.4.1).

Planes, however, are special cases where the Algorithm 3 always terminates without recurring at all. Therefore, even planar surfaces with large holes and a non-rectangular shape like Figure 2.9 will be entirely contained into a single node of the BVH. Consequently, a special case is mandatory to subdivide the solid domain $\mathcal{D} \subset \mathbb{R}^2$, delimited by the trimming curves of a planar face, into smaller regions R_i so that every R_i contains either no or only very small holes:

- As shown in Figure 2.9b, $\mathcal{D}$ is first tessellated such that the resulting mesh approximates its planar contours up to a user-defined maximal chordal error δ_{max}. The choice of δ_{max} dependends on how long and curved are contours of the holes on the considered planar face. However, this has no influence upon the accuracy of our framework, but this affects performances since the smaller the δ_{max} is, the less likely any resulting regions will contain holes;

- Then, each triangle is bounded by a 2D bounding box noted $\tilde{R}_i$ aligned with the axes of the reference frame where $\mathcal{D}$ lies. Note that because of the maximal error δ_{max} induced by the approximation, some areas of $\mathcal{D}$ may not be covered by any $\tilde{R}_i$;

- Each side of $\tilde{R}_i$ is enlarged by the maximal chordal deviation δ_{max} as shown in Figure 2.9c. The enlarged regions are noted R_i. That way, $\mathcal{D}$ is guaranteed to be covered completely by the union of all R_i, $\mathcal{D} \subset \bigcup_i R_i$. In the end, each leaf of the BVH containing an area of the plane defined on $\mathcal{D}$ will correspond to a region R_i.

Moreover, the tessellation algorithm should abide to a shape constraint, i.e., it must favor triangles with similar shapes and that are as close to being equilateral as possible. This reduces the amount of overlap between the computed AABB.

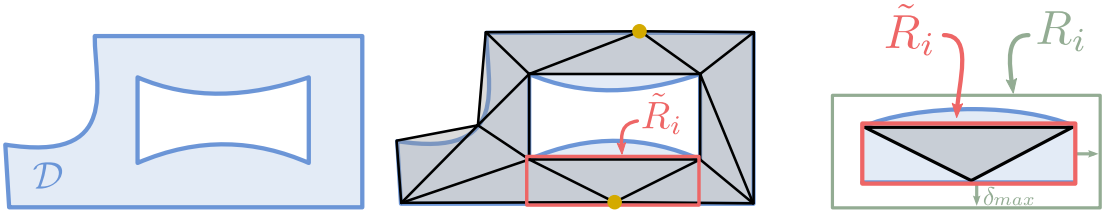


Figure 2.9: (a) The parametric domain of a non-rectangular plane face with a hole. (b) An example of tessellation and the bounding box associated with one of its triangles. Vertices highlighted in yellow have been generated in order to respect the shape constraint. (c) Zoom on the $\tilde{R}_i$ with the red bounding box. Every bounding box is enlarged by the maximal chordal error $\delta_{\max}$ to account for tessellation approximations and make sure areas not covered by any triangle is covered by one R_i, at least.

Finally, it is important to point out that this tessellation is solely used to compute the subdivision of the parametric space of planes. It is **not** used to compute a polyhedral approximation of a trimmed plane later used for CD.

2.4.2 BVH node structure and choice of bounding volumes

The choice of bounding volumes and the data structure for the BVH presented here is shown in Listing 2.4. Its content is given in a pseudo-C++ form with inheritance to highlight the similarities and differences between leaves of the BVH and other nodes which are called here *internal nodes*.

Listing 2.4: Data structures for the BVH.

```
// Abstract node with a reference to its parent and to bounding volumes.
class BVHNode {
  BVHNode parent;

  OBB               obb;
  RevolutionCone    normalCone;
  CompatiblityMask  mask;
}

// BVH Node that is not a leaf.
class BVHInternal: BVHNode {
  BVHNode rightChild;
  BVHNode leftChild;
}

// Leaf of the BVH.
class BVHLeaf: BVHNode {
  PolyhedralCone      polyhedralNormalCone;
```

```
  SupermaximalFeature supermax_feature;
  int                 feature_id;
  Real                u_min, u_max;
  Real                v_min, v_max;
}
```

This BVH is a binary tree with reference to parents, i.e., each node contains references to their children (except for leaves of course) and a reference to their parent. The parent reference is crucial to allow performance improvements introduced in Section 4.5.1. The root is assumed to have a parent set to `Null`.

The choice of bounding volumes is inspired from the Spacialized Normal Cone Hierarchy introduced by Johnson and Cohen [67] and incorporated by Merlhiot et al. [90] into the LMD computation between non-convex polyhedral shapes. This hierarchy combines two bounding volumes at each tree node:

1. A *spatial bounding volume* that bounds the space occupied by all the feature areas contained into the leaves of the sub-tree rooted by the considered node. While Johnson et al. [67] use bounding spheres, Oriented Bounding Boxes (OBB) are preferred here. Indeed, OBB are much tighter than bounding spheres, especially for surface and curve areas that are almost flat;

2. An *orientation-based bounding volume*, C_i, that bounds some orientation-dependent quantities like normals and tangents. Those bounds, like the cone of revolution, attached to each internal tree node, and polyhedral cones, attached to each leaf, are described in details in Chapter 3. Section 4.2.2 justifies the choice of using polyhedral cones at leaves, only.

References to BRep feature areas are only present at leaves of the BVH. To take into account both the concepts of supermaximal feature detailed in Section 2.3 and the subdivision into quasi-flat areas with R_i domains presented in Section 2.4.1, three elements are necessary:

1. The field `supermax_feature` references the subset of supermaximal feature this node represents (see Section 2.3);

2. Because supermaximal features may group several BRep features S_j of the input BRep CAD model, the index `feature_id` identifies which of the S_j is to be approximated by this node. In the remaining algorithm pseudo-code, indexing the BVH node supermaximal feature field, e.g., `node.supermax_feature[node.feature_id]`, returns the corresponding BRep feature S_j;

3. Finally, as described in Section 2.4.1, each leaf contains only one feature area with its associated parametric rectangular domain R_i. The limits of R_i are expressed by the fields (u_{min}, u_{max}) for the parameter u and (v_{min}, v_{max}) for the parameter v. If the feature area is a curve segment, then only (u_{min}, u_{max}) are used. Both are ignored if the feature is a vertex.

2.4.3 The culling tests

The *culling test* between two BVHNode decides whether they have no chance of containing a LMD or not. The culling test is said to *succeed* if it successfully detects that no LMD will be found by continuing the search on a given pair of BVHNode, and to *fail* otherwise. Such a test must be *conservative*, i.e., while the test must never succeed on pairs that actually support a LMD, it is acceptable for it to fail on two pairs that do not actually contain any LMD. The test combines all the bounding volumes set up in Section 2.4.2 into a specific order:

1. Test the curvature-based compatibility masks using the method described in Section 2.2;

2. Test whether normal cones of revolution contain antipodal directions, i.e., that they satisfy Equation (1.24). If they do not, then the given pair of BVHNode cannot lead to any LMD;

3. Test whether the two OBB, O_i and O_j, are separated by a distance greater than the user-defined maximal LMD of interest d_{max} or not. The naive method shown in Figures 2.10a and 2.10b computes the smallest distance d between O_i and O_j, and compares it with d_{max}. If $d > d_{max}$, then the culling test succeeds since d is an upper bound of the distance between O_i and O_j. However, computing the exact distance between O_i and O_j is more expensive than testing if they intersect. Indeed, while both methods can be performed in constant-time, the former has to locate exactly the closest points, e.g., using the GJK algorithm [49], while the latter only has to find a plane splitting $\mathbb{R}^3$ into two half-spaces, each containing either O_i or O_j (refer to the Separating Axis Theorem [52]).

   Thus, the distance-based test is transformed into an intersection test by enlarging each extent of one OBB, say O_i, by d_{max}, producing O_{Mi}. Then, O_{Mi} contains all the points of O_j that are closer than d_{max} from O_i so O_{Mi} must necessarily intersect O_j if the latter contains one of those points. Let us note that this test is slightly less discriminative than the distance-based test. Indeed, enlarging the extent of O_i actually includes also some points that are farther than d_{max} but the test stays conservative;

4. If both BVH nodes are leaves then, test whether their associated polyhedral normal cones C_i and C_j contain antipodal directions as described in Algorithm 4 or not. If no such direction exists, then the features contained into those leaves, S_i and S_j, have no chance of containing a LMD inside their R_i and R_j, respectively.

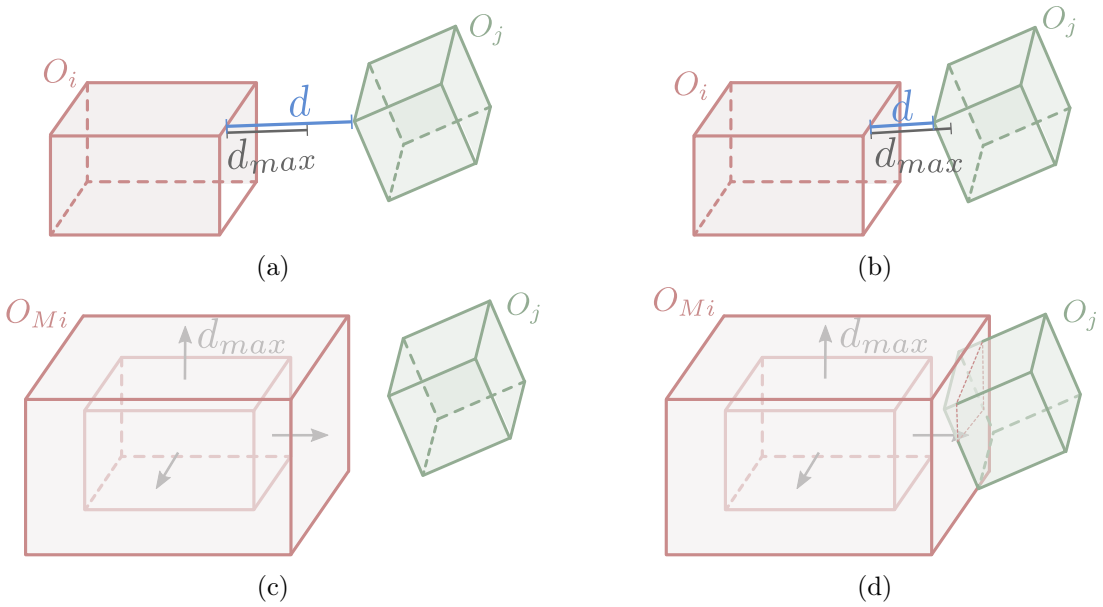


Figure 2.10: OBB culling test that succeeds in (a, c), and fails in (b, d). (a) and (b) compute the distance d between O_i and O_j, and compute it with the maximum LMD d_{max} needed. (c) and (d) transforms the distance computation into an intersection test with the enlargement of O_i by d_{max}.

The sequence of those tests is critical because they start with the cheapest one to end with the most expensive one. As soon as one test finds that no LMD exists using one of the test of the sequence, then the culling test succeeds and its following tests are no longer required.

2.4.4 Top-Down construction

In practice, the BVH is built using a top-down scheme. Firstly, all the vertices, faces, and edges are gathered together. Then, all faces and edges are subdivided into quasi-flat areas. Each such area and vertex has its spatial, O_i, and orientation-based, C_i, bounding volumes computed and stored together with a reference to the supermaximal feature (as defined in Section 2.3) it belongs to into a new BVH leaf. Finally, the binary tree structure is built by splitting recursively this set of leaves into two parts. For example, the following simple splitting rule is used, similarly to [52], that results in good results in practice:

- Given a set L of BVH leaves, compute the covariance matrix Σ obtained from all the centers $\mathbf{c}_i$ of the OBBs O_i associated with the leaf l_i;
- Compute the eigenvector $\mathbf{v}_{max}$ of Σ associated with the largest eigenvalue;
- Collect all the values of the scalar product $\langle \mathbf{v}_{max}, \mathbf{c}_i \rangle$ or all O_i in L, and compute their median m;
- Finally, L is split into two subsets L_1, L_2 such that $L_i = \{l_i \in L \mid \langle \mathbf{v}_{max}, \mathbf{c}_i \rangle \leq m\}$ and $L_j = \{l_i \in L \mid \langle \mathbf{v}_{max}, \mathbf{c}_i \rangle > m\}$.

The construction method of a BVH that produce an efficient traversal is an active topic of research and may rely on various cost functions that measure the quality of the possible slitting options [95, 51, 140]. Refer to the survey from Andersen et al. [7] and to Erleben et al. [41] for further details regarding various BVH construction approaches, including methods that do not follow a top-down scheme.

2.5 Conclusion

This chapter described the various analyses and steps performed to pre-compute a data structure necessary for the fast computation of LMDs. Firstly, two analyses are performed on the input BRep models:

1. Each canonical surface or curve is given *compatibility attributes* that indicates other surface and curve types with which no LMD is possible because of their respective curvatures;
2. Features are grouped into *supermaximal features* whenever their underlying geometries are identical if they were not trimmed.

Those three analyses allow algorithms to avoid unnecessary computations as explained in Chapter 4.

Besides those analyses, choices are made regarding the data structure used for LMD computation at runtime: the BVH. The efficiency of this data structure depends strongly on the bounding volumes capability of detecting efficiently when two areas of two BRep models have no chance to contain LMDs. Combining spatial bounding volumes, like bounding spheres that bound the space occupied by the objects, with orientation-based bounding volumes, like cones of revolution that bound the normals or tangents of the objects, is known to yield good results [70]. However, OBB are much tighter than bounding spheres when bounding surface and curve areas. Moreover, a tighter orientation-based bounding volume, the polyhedral cone, is described in Chapter 3.

Secondly, with all those observations in hand, the construction of a BVH is performed. It starts by splitting the domains of all the features into rectangular sub-domains over which they are *almost flat*. This has the benefit to:

- Ensure the tightness of the bounding volumes for features with high curvature and/or holes;
- Evolve toward the property where two such almost-flat areas will contain at most one LMD to be satisfied most of the time in practice. This is crucial for the use of iterative root-finding methods for the exact localization of a LMD and Section 4.3.1) enforces this property for some canonical faces;
- Ignore completely, for the rest of the LMD localization process, each rectangular sub-domain that lies completely outside of the domain delimited by restriction curves.

All computations described in the remainder of this thesis are performed on those almost-flat feature areas (or on their unions) defined on the rectangular subdomains R_i. Trimming curves are taken into account again only at the very end of the LMD computation process (see Section 4.4.1) to remove solutions that lie outside of the trimmed sub-domains.

Chapter 3

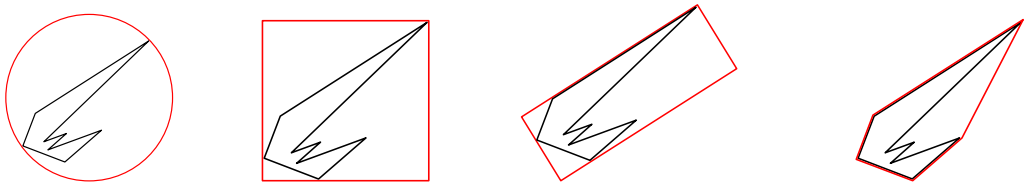
Tightening the bounds on solutions: take orientation into account with normal and tangent cones

The efficiency of a BVH is strongly correlated with the tightness of its bounding volumes and the computational cost of the corresponding culling tests. While plenty of spatial bounding volumes have been designed (bounding spheres, AABB, OBB, k-DOP, etc.), only one orientation-based bounding volume exists in the literature: the cone of revolution. To reduce this gap, this chapter defines a new orientation-based bounding volume that is much tighter at the cost of more computation-intensive culling tests: the polyhedral cone. After the definition of polyhedral cones, polyhedral normal cones, and polyhedral tangent cones, their generation methods for face areas parametrized on a rectangular domain, for edges areas, and for vertices, are described. Then, two culling tests are designed on pairs of polyhedral cones. The first one checks the existence of antipodal directions on both cones and is used for pairs of BRep features or a BRep feature and a deformable curve. The second test checks the existence of orthogonal directions and is designed for a pair of deformable curves. However, those tests are shown to have a high complexity, cubically proportional to the number of vectors used for the definition of each polyhedral cones. This performance issue will be mitigated in the next chapter.

3.1 How and why bounding the normals of a BRep feature

Bounding volumes, as those shown in Figure 3.1, are popular tools of early detection when two objects are too far to be of interest for CD. Indeed, they provide algorithms with an efficient way to compute a lower bound of the global minimal distance between the bounded objects. Though for this bound to be accurate, the bounding volumes themselves must be as close to the shape geometry as possible. However, the tighter a bounding volume is, the harder the actual distance bound computation becomes. For example, computing the distance between two bounding spheres is straightforward while computing the distance between two convex hulls requires complex iterative methods like the GJK algorithm [46]. Thus, a compromise must be reached between tightness and cost of the distance computation.

Because all those bounding volumes account only for the space occupied by the solids, they are called *spatial* bounding volumes. Section 1.3.1 has shown that critical points of the distance function are characterized by a condition of intersection between their tangent cone polars



(a) (b) (c) (d)

Figure 3.1: Common spatial bounding volumes sorted from the loosest to the tightest: (a) bounding sphere, (b) axis-aligned bounding box, (c) oriented bounding box, (d) convex hull. Other spatial bounding volumes not listed here exist, e.g., k-DOP and spherical shells [79]. This list is not exhaustive.

(see Equation (1.23)). Thus, similarly to spatial bounding volumes, it would be useful to define volumes that bound those cones. Then, if two such bounding volumes are disjoint, it is safe to conclude that the underlying tangent cone polars are disjoint as well. Because those volumes would depend on the shapes orientations, they are called *orientation-based* bounding volumes. The idea of bounding the tangent cone polars appeared in the context of CD with the introduction of *normal cones* (of revolution) on polyhedral models by Johnson et al. [67] but have not been formalized rigorously (as bounds of tangent cone polars) before their generalization to simplicial complexes by Merlhiot et al. [90]. Normal cones of revolution have been applied earlier to the distance computation problem between smooth surfaces as well [71, 132] but not for complete BRep models, which include edges and vertices at the boundaries of faces.

However, from a tightness point of view, one could say that cones of revolution are quite analog to bounding spheres except that they operate with angles and directions instead of distances and points. Equations (3.1) and (3.2) highlight their similarities: a bounding sphere B is defined with a center $\mathbf{c}$ and its radius r, while a cone of revolution C is defined with an axis $\mathbf{n}$ and its half apex-angle α:

$$B = \left\{\mathbf{p} \mid \|\mathbf{p}-\mathbf{c}\| \leq r, \mathbf{p} \in \mathbb{R}^3\right\}, \qquad \mathbf{c} \in \mathbb{R}^3, r \in \mathbb{R}, \tag{3.1}$$

$$C = \left\{\mathbf{v} \mid \text{angle}(\mathbf{v},\mathbf{n}) \leq \alpha, \mathbf{v} \in \mathbb{R}^3\right\}, \qquad \mathbf{n} \in \mathcal{S}^2, \alpha \in [0,\pi], \tag{3.2}$$

where $\text{angle}(\mathbf{v},\mathbf{n}) = \arccos\left(\left\langle \frac{\mathbf{v}}{\|\mathbf{v}\|}, \mathbf{n}\right\rangle\right)$ is the angle between $\mathbf{v}$ and $\mathbf{n}$, and $\mathcal{S}^2 \subset \mathbb{R}^3$ the unit sphere. Moreover, the intersection test between two bounding spheres $B_\mathcal{A}$ and $B_\mathcal{B}$ is also very similar to the intersection test between two cones of revolution $C_\mathcal{A}$ and $C_\mathcal{B}$:

$$B_\mathcal{A} \cap B_\mathcal{B} \neq \emptyset \iff \|\mathbf{p}_\mathcal{A} - \mathbf{p}_\mathcal{B}\| \leq r_\mathcal{A} + r_\mathcal{B}, \tag{3.3}$$

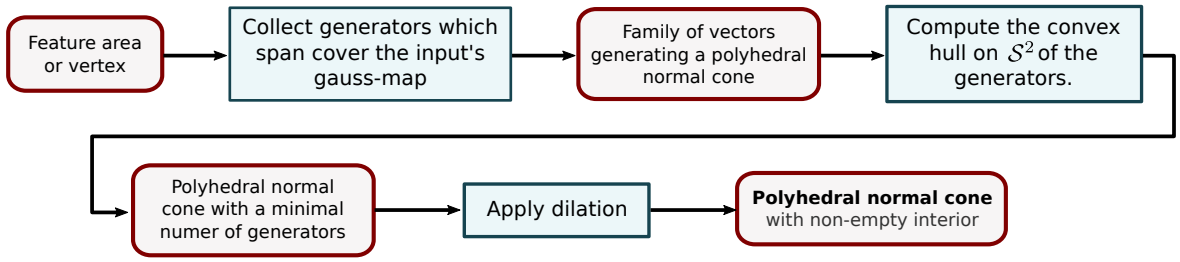
$$C_\mathcal{A} \cap C_\mathcal{B} \neq \emptyset \iff \text{angle}(\mathbf{n}_\mathcal{A}, \mathbf{n}_\mathcal{B}) \leq \alpha_\mathcal{A} + \alpha_\mathcal{B}. \tag{3.4}$$

Given those similarities, it is natural to wonder whether one could draw some inspiration from, say, OBB and convex hulls to design tighter orientation-based bounding volumes. The answer is positive when resorting to cones having a polygonal base, as illustrated in Figure 3.2.

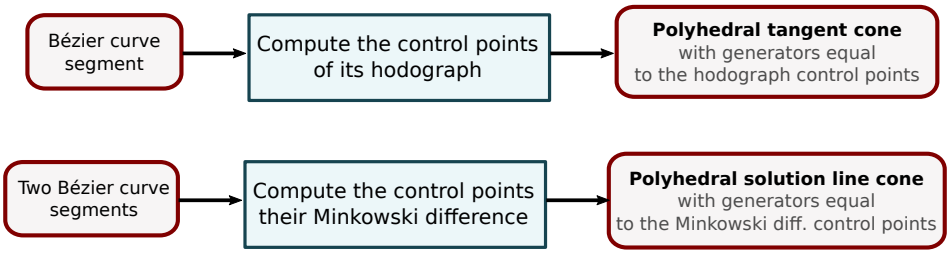
(a) (b)

Figure 3.2: (a) a cone of revolution and (b) a polyhedral cone. Their intersection with $\mathcal{S}^2$ is highlighted.

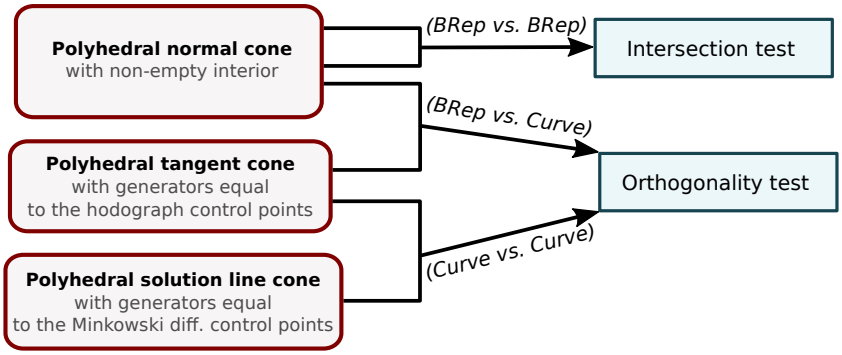
Section 3.2 introduces the definition of polyhedral cones and provides an algorithm for defining two of them. Section 3.3 details computation of polyhedral cones that bound the tangent cone polar of a feature area. The dilation of those cones is presented in Section 3.4 in order to address conformal contact configurations. Finally, Section 3.5 describes the usage of polyhedral cones to bound the difference of two features and tangents and how they can characterize the non-existence of LMDs between two curves. Note that, as specified in Section 2.5, all the presented algorithms that operate on surfaces do so on rectangular domains because processing their trimming curves is postponed to the runtime phase in Section 4.4.1. Finally, those contributions are summarized in Figure 3.3 (the technical terms not mentioned so far are explained in the subsequent chapters).



(a) Successive steps necessary to compute a dilated polyhedral normal cone that bounds tightly the tangent cone polars of a vertex or feature area.



(b) Successive steps necessary to compute a tangent cone that bounds tightly the tangent cone of a curve segment.



(c)

Figure 3.3: Summary of the proposed methods to compute (a) polyhedral normal cones for BRep feature areas, (b) tangent cones for curve segments, and (c) solution line cones for pairs of curve segments.

3.2 Obtaining tighter normal bounds with polyhedral cones

This section describes new tight bounds for the tangent cone polar of a vertex, a curve segment, or a surface patch of a BRep model. Following the literature [90, 71, 132, 67], we call those bounds *normal bounds* or *normal cones* even thouh they only bound the tangent cones polars but not necessarily the normal cone, as defined in convex analysis [16].

Let us first recall the concept of *Gauss Map*, $\mathcal{G}$, of a surface. $\mathcal{G}$ is a function $\mathcal{G} : \mathbb{R}^3 \to \mathcal{S}^2$ that maps each surface point to its normal $\mathbf{n}$. Each such $\mathbf{n}$ is a unit normal defining a point on the unit sphere $\mathcal{S}^2 := \{\mathbf{n} \in \mathbb{R}^3 \mid \|\mathbf{n}\| = 1\}$. This concept is extended to edges, and vertices, when considering that $\mathcal{G}$ maps a point $\mathbf{p}$ of any feature 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 3.2, $\mathcal{S}^2$ is embedded into an Euclidean space and given a local direct coordinate system centered at the origin. Its two poles are located at $+\mathbf{y}$ and $-\mathbf{y}$. As a reminder, the intersection curve of $\mathcal{S}^2$ with any plane passing through these poles (and the origin) is called a *meridian* (see Figure 3.4a) and planes orthogonal to the $\mathbf{y}$ axis (but not necessarily containing the origin) intersect $\mathcal{S}^2$ along *lines of latitude* (see Figure 3.4b).

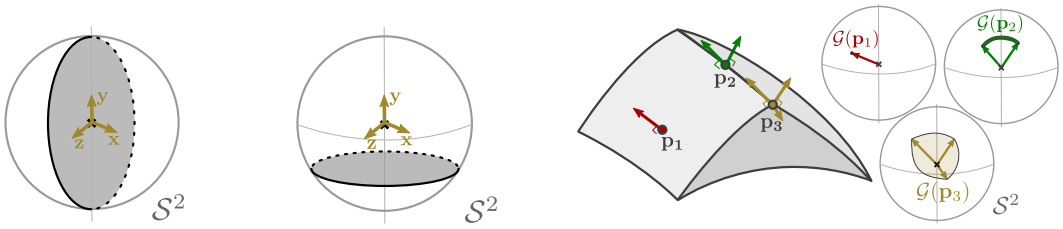


Figure 3.4: The unit sphere with its local coordinate system. An example of meridian (a), and line of latitude (b). (c) Points $\mathbf{p}_1, \mathbf{p}_2$, and $\mathbf{p}_3$ respectively on a face, an edge, and a vertex. The surfaces normals at those points are shown as arrows and their Gauss map images are drawn on $\mathcal{S}^2$.

While cones of revolution can be satisfyingly tight to bound normals of complex surfaces like Bézier surfaces, it is clear that they can be extremely loose for edges and canonical faces for which tangent cone polars often reduce to an arc on $\mathcal{S}^2$. Polyhedral cones are significant improvements for those cases.

3.2.1 Definitions from convex analysis

Following the usual definition of polyhedral cones from convex analysis [16], let $\mathbf{G} = \{\mathbf{g}_1, \ldots, \mathbf{g}_n\}$ be a finite set of vectors in $\mathbb{R}^3$. One calls:

$$C = \text{cone}(\mathbf{G}) = \text{cone}(\mathbf{g}_1, \ldots, \mathbf{g}_n) = \left\{ \mathbf{v} \mid \mathbf{v} = \sum_{j=1}^{n} \mu_j \mathbf{g}_j, \mu_j \geq 0, j = 1, \ldots, n \right\}, \tag{3.5}$$

a *finitely generated (polyhedral) cone* and the $\mathbf{g}_i$ are its *generators*. A finitely generated polyhedral cone is always convex and the origin is its apex except when it spans the whole space $C = \mathbb{R}^3$. In particular, the finite set $\mathbf{G} = \{\mathbf{g}_1, ..., \mathbf{g}_n\} \subset \mathcal{S}^2$ of unit vectors is a *minimal set of unit generators* if and only if none of its proper subsets generates the same cone, i.e.:

$$\forall \mathbf{G}' \subset \mathbf{G} \quad \mathbf{G}' \neq \mathbf{G} \Rightarrow \text{cone}(\mathbf{G}) \neq \text{cone}(\mathbf{G}'). \tag{3.6}$$

According to the Minkowski-Weil theorem [16], polyhedral cones can equivalently be defined as the intersection of a set of half-spaces whose boundary pass through the origin: $C =$

$\{\mathbf{v} \mid \langle \mathbf{n}_j, \mathbf{v} \rangle \leq 0\}$, $\mathbf{n}_j \in \mathbf{N}$ where the finite set of vectors $\mathbf{N} = \{\mathbf{n}_1, \ldots, \mathbf{n}_m\}$ are the (outward) normals of the planes of each half-space. Let us note that because each edge of a polyhedral cone is necessarily collinear with a generator of C, each face normal is necessarily related to two generators, i.e.:

$$\forall \mathbf{n} \in \mathbf{N}, \exists (\mathbf{g}_i, \mathbf{g}_j) \in \mathbf{G}^2 \text{ such that } \mathbf{n} = \mathbf{g}_i \times \mathbf{g}_j. \tag{3.7}$$

In particular, if $\mathbf{G}$ is a minimal set of unit generators, $\mathbf{G}$ is *sorted* if:

$$\forall i \in [0, n] \quad \mathbf{n}_i = \mathbf{g}_{i+1 \pmod n} \times \mathbf{g}_i. \tag{3.8}$$

Figure 3.5 shows a 3D polyhedral cone, its normals, and various possible sets of generators. Both the normal-based and the minimal generator set-based representations have practical uses. On the one hand, the normal-based representation is useful to characterize efficiently the position of a vector with regard to the polyhedral normal cone. In particular, whenever a closest point between a vertex and another feature is computed, one must check that the computed contact normal actually lies inside the vertex normal cone. This is easily performed when checking that the image of the contact normal on $\mathcal{S}^2$ lies on the same side of all the polyhedral cone half-spaces. On the other hand, the sorted minimal generator set-based representation is used by the intersection test described in the Section 3.2.2.

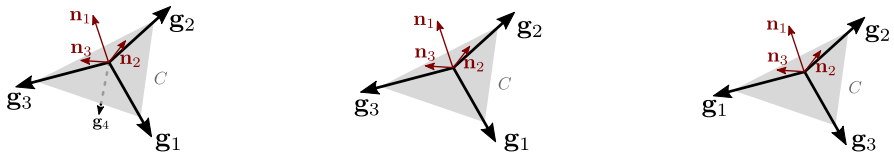


Figure 3.5: A polyhedral cone with its (outward) faces normals $\mathbf{N} = \{\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3\}$. Its generators set: (a) $\mathbf{G} = \{\mathbf{g}_1, \ldots, \mathbf{g}_4\}$ is not minimal because $\mathbf{g}_4$ is a linear combination of $\mathbf{g}_1$, $\mathbf{g}_2$ and $\mathbf{g}_3$; (b) $\mathbf{G} = \{\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3\}$ is minimal but not sorted because, e.g., $\mathbf{n}_1 \neq \mathbf{g}_2 \times \mathbf{g}_1$; (c) $\mathbf{G} = \{\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3\}$ is minimal and sorted.

In the following sections, $-C = \{-\mathbf{c} \mid \mathbf{c} \in C\}$ designates the cone opposite to C. This means, if $\mathbf{G}$ and $\mathbf{N}$ are respectively sets of generators (resp. face normals) of C, then $-\mathbf{G} = \{-\mathbf{g} \mid \mathbf{g} \in \mathbf{G}\}$ (resp. $-\mathbf{N} = \{-\mathbf{n} \mid \mathbf{n} \in \mathbf{N}\}$) are the generators (resp. face normals) of $-C$.

3.2.2 Checking that two normal cones contain antipodal directions

Let $C_\mathcal{A} = \text{cone}(\mathbf{g}_1^a, \mathbf{g}_2^a, \ldots, \mathbf{g}_m^a)$ and $C_\mathcal{B} = \text{cone}(\mathbf{g}_1^b, \mathbf{g}_2^b, \ldots, \mathbf{g}_n^b)$ be two polyhedral cones with nonempty interiors and m and n generators, respectively. The existence of LMDs given by Equation (1.23) and Figure 1.12e is satisfied if and only if their tangent cone polars contain antipodal directions, which we call *antipodal normals* here. Thus, given the two bounds $C_\mathcal{A}$ and $C_\mathcal{B}$ of those tangent cone polars, it is useful to identify feature pairs such that $C_\mathcal{A} \cap -C_\mathcal{B} = \emptyset$ because they are guaranteed not to contain any LMD. First, let us consider the following property:

Proposition 1. *If the interior* $\text{Int}(C_\mathcal{A})$ *and* $\text{Int}(C_\mathcal{B})$ *of two polyhedral cones* $C_\mathcal{A}$ *and* $C_\mathcal{B}$ *contain antipodal directions then, their Minkowski sum* $C_\mathcal{A} \oplus C_\mathcal{B}$ *is equal to the whole space* $\mathbb{R}^3$*, i.e:*

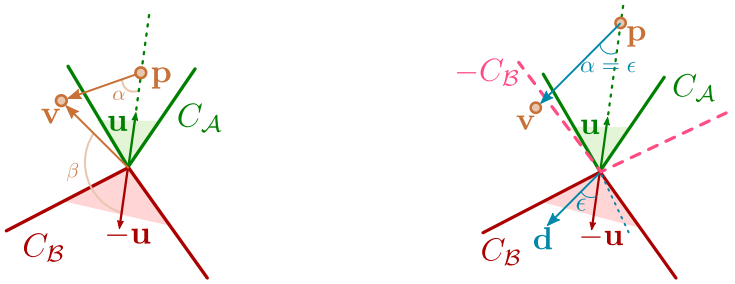
$$\text{Int}(C_\mathcal{A}) \cap \text{Int}(-C_\mathcal{B}) \neq \emptyset \Rightarrow C_\mathcal{A} \oplus C_\mathcal{B} := \text{cone}(\mathbf{g}_1^a, \ldots, \mathbf{g}_m^a, \mathbf{g}_1^b, \ldots, \mathbf{g}_n^b) = \mathbb{R}^3. \tag{3.9}$$

Proof. Let $\mathbf{v}$ be any vector of $\mathbb{R}^3$. If $\text{Int}(C_\mathcal{A}) \cap \text{Int}(-C_\mathcal{B}) \neq \emptyset$ then, there exists a unit vector $\mathbf{u} \in \text{Int}(C_\mathcal{A})$ such that $-\mathbf{u} \in \text{Int}(C_\mathcal{B})$. Let $\mathbf{p}$ be a vector defining a point p 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 follows:

$$\cos(\alpha) = \left\langle \frac{\mathbf{v} - \lambda\mathbf{u}}{\|\mathbf{v} - \lambda\mathbf{u}\|}, -\mathbf{u} \right\rangle, \tag{3.10}$$

$$\cos(\alpha) = \frac{\lambda - \langle \mathbf{v}, \mathbf{u} \rangle}{\|\mathbf{v} - \lambda\mathbf{u}\|}. \tag{3.11}$$

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 Figure 3.6a illustrates those elements.



(a) (b)

Figure 3.6: (a) 2D example of configuration of the elements of this proof. (b) A choice of ϵ and the corresponding location of $\mathbf{p} = \lambda\mathbf{u}$.

In addition, because $\text{Int}(C_\mathcal{B})$ is open, there exists an $\epsilon \in [0, \pi]$ such that:

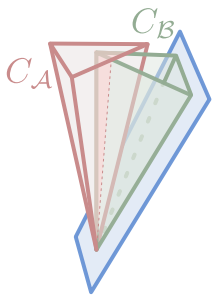
$$\forall \mathbf{d} \in \mathbb{R}^3, \arccos(\langle -\mathbf{u}, \mathbf{d} \rangle) \leq \epsilon \Rightarrow \mathbf{d} \in \text{Int}(C_\mathcal{B}). \tag{3.12}$$

When choosing λ such that $\alpha = \min(\epsilon, \beta)$ (as shown in Figure 3.6b), it comes that $(\mathbf{v} - \mathbf{p}) \in C_\mathcal{B} \Rightarrow \mathbf{v} \in \{\mathbf{c}_\mathcal{B} + \lambda\mathbf{u} \mid \mathbf{c}_\mathcal{B} \in C_\mathcal{B}\} \Rightarrow \mathbf{v} \in C_\mathcal{A} \oplus C_\mathcal{B}$ because $\forall \lambda \in \mathbb{R}^\star_+, \lambda\mathbf{u} \in C_\mathcal{B}$ by definition of a cone. Thus, $\mathbb{R}^3 \subset C_\mathcal{A} \oplus C_\mathcal{B}$. Because $C_\mathcal{A} \oplus C_\mathcal{B}$ is clearly a subset of $\mathbb{R}^3$, we conclude that $C_\mathcal{A} \oplus C_\mathcal{B} = \mathbb{R}^3$. □

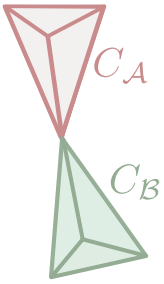
Polyhedral cones with empty interiors are not handled by this proposition. However, this limitation is avoided by dilating cones with empty interiors using the method described in Section 3.4.

The contrapositive of this property provides an efficient and conservative culling test, which we refer to as *intersection test*: any pair of BVH node for which the associated normal cones are such that $C_\mathcal{A} \oplus C_\mathcal{B} \neq \mathbb{R}^3$ can be rejected as their underlying feature areas do not have any antipodal normals. Moreover, because $C_\mathcal{A} \oplus C_\mathcal{B}$ is itself a finitely-generated polyhedral cone, it is convex and $C_\mathcal{A} \oplus C_\mathcal{B}$ 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 convex cone, it is sufficient to find a plane that contains one of its face and all its generators into a single half-space. Because of the relationship between the normals $\mathbf{n}_i \in \mathbf{N}$ and the cone generators $\mathbf{g}^a_j$ (Equation (3.7)), normals to faces of $C_\mathcal{A} \oplus C_\mathcal{B}$ can be obtained from the cross products of all combinations of generator pairs, i.e., $\mathbf{g}^a_i \times \mathbf{g}^a_j$, $\mathbf{g}^a_i \times \mathbf{g}^b_j$, and $\mathbf{g}^b_i \times \mathbf{g}^b_j$ for all $i \in [0\,..\,m], j \in [0\,..\,n]$.

The intersection test is summarized by Algorithm 4. For each potential face normal of $C_\mathcal{A} \oplus C_\mathcal{B}$ computed at line 5, the algorithm checks that all the generators of $C_\mathcal{A} \oplus C_\mathcal{B}$ lie on the same half-space. If they do so, a separating plane passing through the origin has been found and it is possible to conclude at line 7 that $Int(C_\mathcal{A}) \cap \text{Int}(-C_\mathcal{B}) = \emptyset$ because of Proposition 1. If this test fails for all face normals of $C_\mathcal{A} \oplus C_\mathcal{B}$, then the algorithm concludes at line 11 that



(a) Support plane of $C_{\mathcal{A}} \oplus C_{\mathcal{B}}$ passing through the origin found.



(b) No support plane of $C_{\mathcal{A}} \oplus C_{\mathcal{B}}$ passing through the origin found.

Figure 3.7: (a) Configuration where the two cones do not contain any antipodal directions, i.e., $C_{\mathcal{A}} \cap -C_{\mathcal{B}} = \emptyset$. (b) Configuration where the two cones contain antipodal directions.

$C_{\mathcal{A}}$ and $C_{\mathcal{B}}$ contain antipodal directions. The two possible results of the algorithm is shown on Figure 3.7

Let m and n be the number of generators of $C_{\mathcal{A}}$ and $C_{\mathcal{B}}$, the test at line 6 performs $O(m+n)$ scalar products and comparisons. This is repeated for all potential normals of $C_{\mathcal{A}} \oplus C_{\mathcal{B}}$, i.e., $O(m^2 + n^2 + mn)$ times. Therefore, the overall complexity of the intersection test is $O(m^3 + nm^2 + n^3 + mn^2 + nm^2 + mn^2)$, which can be assimilated to $O(n^3)$ assuming that the number of generators of both polyhedral cones does not differ by more than a small constant factor.

The polynomial complexity of this test makes polyhedral cones with a moderately large number of generators, inadequate for real-time applications. In order to avoid this issue, a practical implementation should rely on simplified polyhedral cones with limited numbers of generators as described in Section 4.2.2. Moreover, the exploitation of temporal coherence shown in Section 4.5.2 significantly reduces the cost of repeated intersection tests.

Algorithm 4 Polyhedral Normal Cone intersection test

1: **Inputs:** the generators $\mathbf{G}_{\mathcal{A}}, \mathbf{G}_{\mathcal{B}}$ of two polyhedral cones $C_{\mathcal{A}}$ and $C_{\mathcal{B}}$.
2: **Output:** whether or not $\mathrm{Int}(C_{\mathcal{A}}) \cap \mathrm{Int}(-C_{\mathcal{B}}) \neq \emptyset$.
3: **for all** $(\mathbf{g}_1, \mathbf{g}_2) \in (\mathbf{G}_{\mathcal{A}} \times \mathbf{G}_{\mathcal{A}}) \cup (\mathbf{G}_{\mathcal{B}} \times \mathbf{G}_{\mathcal{B}}) \cup (\mathbf{G}_{\mathcal{A}} \times \mathbf{G}_{\mathcal{B}})$ **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 $C_{\mathcal{A}} \oplus C_{\mathcal{B}}$.
6: **if** all the scalar products $\langle \mathbf{g}, \mathbf{n} \rangle$ have the same sign for all $\mathbf{g} \in \mathbf{G}_{\mathcal{A}} \cup \mathbf{G}_{\mathcal{B}}$ **then**
7: **return** FALSE ▷ All generators lie in the same half-space.
8: **end if**
9: **end if**
10: **end for**
11: **return** TRUE ▷ No separating plane found.

3.3 Generation of 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-minimal normal cones of all canonical surfaces, and to provide bounds that are not too loose for complex surfaces, edges, and vertices. Let us recall that the intersection test presented in Section 3.2.2 required polyhedral cones to have nonempty interiors. The following sections only compute

closed polyhedral normal bounds. Then, they are dilated in Section 3.4 to guarantee the non-empty interior property.

3.3.1 Meridian or 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 $\theta < \pi$. Otherwise, the resulting cone would contain all $\mathbb{R}^3$. If $\theta > \pi$, the corresponding surface must be subdivided into smaller pieces with 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. A minimal (closed) polyhedral bound of an arc of a meridian can thus be generated by the two vectors at its extremities (see Figure 3.8a). On the other hand, arcs of lines of latitude are the result of the intersection of $\mathcal{S}^2$ with a plane orthogonal to its revolution axis. Because this plane does not necessarily contain the origin, this intersection cannot be bounded as straightforwardly as meridians. Here, we limit our bounding method to the four half-spaces depicted through Figures 3.8b, 3.8c and 3.8d as a trade-off between the intersection test efficiency (see Section 3.2.2) and the thightness of the bounding volume. Figure 3.8e compares the resulting polyhedral cone to a cone of revolution that is much looser as it does not account for the lack of thickness of a single line of latitude.

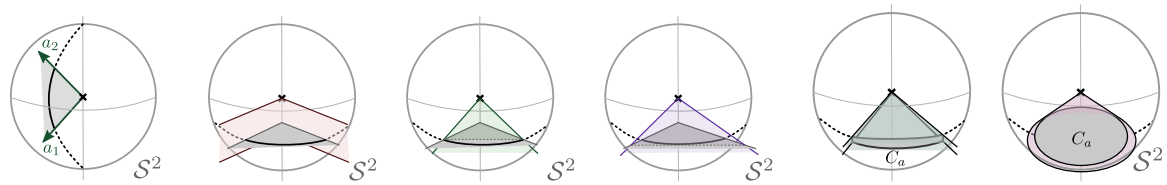
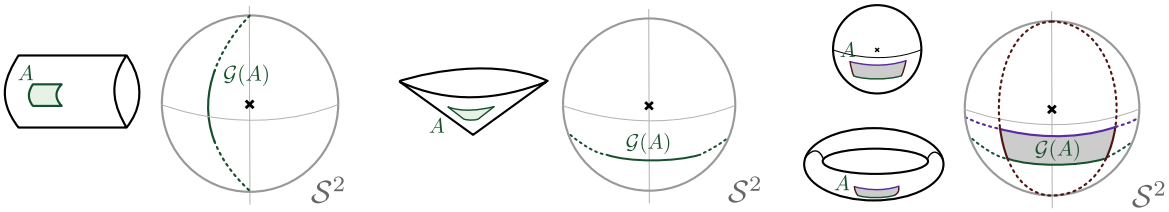


Figure 3.8: (a) Bounding an arc of meridian requires only the two generators $\{\mathbf{g}_1, \mathbf{g}_2\}$. Bounding arc of line of latitude is achieved using four planes: (b) two at its extremities, (c) one passing through the origin and both extremities, and (d) one passing though the origin and tangent to the curve (e.g. tangent to its middle point). (e) The resulting polyhedral cone (left) and cone of revolution (right) of the same line of latitude. The cone of revolution is much looser.

3.3.2 Canonical surfaces

The polyhedral normal cone of a plane is trivial as it 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. As shown in Figure 3.9, we can always rotate the shape such that:

- The Gauss Map image of a cylindrical area coincides with a meridian;
- The Gauss Map image of a conical area coincides with a line of latitude;
- The Gauss Map image of an area of a sphere, a torus, or an arbitrary revolution surface bounded by isoparametric curves, is exactly delimited by two meridians and two lines of latitude. Its bounds can be derived from that of each individual line and their combination with a half-space representation ends up with four half-spaces.



(a) (b) (c)

Figure 3.9: An area, named A, of a cylinder (a), a cone (b) a sphere and a torus (c), with their respective Gauss map images on $\mathcal{S}^2$ noted $\mathcal{G}(A)$ and bounded with a polyhedral cone.

3.3.3 Edges, vertices, and Bézier surfaces

Computing the Gauss map image 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. They are processed with a more general approach in three steps.

Firstly, the algorithm collects enough generators to form a polyhedral cone that bounds, not necessarily minimally, the polyhedral normal cone polar of each feature area. The exact method depends on the feature type:

- Bézier surfaces: an explicit computation of the scaled normal surface [71] is performed, which is also a Bézier surface. Its control points can be used as generators because they benefit from a convex hull property [107];
- 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;
- Vertices: the generators are given by the normals of the surfaces meeting at this point.

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 Merlhiot et al. [90] with a method based on the Gift-wrapping convex hull algorithm for the construction of normal cones of revolution, but no specific details were provided.

Thus, a new approach is proposed based on the 2D QuickHull algorithm [12] now operating on the unit sphere, i.e., it works similarly to extract the smallest polyhedral cone containing all the given generators on $\mathcal{S}^2$. This method is named *Spherical QuickHull* and the resulting polyhedral cone is the *spherical convex hull*.

Now, let us state some definitions. Given a direction $\mathbf{d} \in \mathbb{R}^3$, an *extremal point toward* $\mathbf{d}$, noted $\mathbf{p}^\star \in \mathcal{P} \subset \mathbb{R}^3$, is the one that maximizes the product $\mathbf{p}^\star = \operatorname{argmax}_{\mathbf{p}\in\mathcal{P}} \langle \mathbf{p}, \mathbf{d} \rangle$. Analogously, given a direction $\mathbf{d}$ and an axis $\mathbf{a} \in \mathbb{R}^3$ orthogonal to $\mathbf{d}$, we define the notion of *extremal generator toward* $\mathbf{d}$ *with respect to* $\mathbf{a}$ which is the generator $\mathbf{g}^\star$ that maximizes the angle $\mathbf{g}^\star = \operatorname{argmax}_{\mathbf{p}\in P} \arccos\left(\langle \mathbf{p}, \mathbf{a} \times \mathbf{d} \rangle\right)$ and $\langle \mathbf{p}, \mathbf{d} \rangle \geq 0$.

The proposed method is equivalent to a 2D QuickHull algorithm where the notion of extremal points is replaced by the notion of extremal generators. It is detailed in Algorithm 5. The method requires all $\mathbf{g} \in \mathbf{G}$ to lie strictly in a single half-space, i.e., $\operatorname{cone}(\mathbf{G}) \neq \mathbb{R}^3$. In particular, no two generators are allowed to be antipodal. Edges or faces with a set of normal cone generators that do not comply with those properties must be subdivided into areas with smaller normal cones. The algorithm is initialized with a finite set of unit generators $\mathbf{G} \subset \mathcal{S}^2$ collected with the procedure described above, and two generators $\mathbf{g}_1^\star$ and $\mathbf{g}_2^\star$ known to be extremal. For example, given two distinct generators (chosen arbitrarily) $\mathbf{g}_i, \mathbf{g}_j \in \mathbf{G}$, the algorithm can compute $\mathbf{g}_1^\star$ (resp. $\mathbf{g}_2^\star$) as the extremal generator toward $\mathbf{d} = \mathbf{g}_i \times \mathbf{g}_j$ (resp. $-\mathbf{d}$) with respect to the axis

$\mathbf{a} = \mathbf{g}_j - \mathbf{g}_i$ (resp. $-\mathbf{a}$). A first partial spherical convex hull represented by a set $\mathcal{F}$ of pairs of directions that delimit its faces is formed by $\text{cone}(\mathbf{g}_1^\star, \mathbf{g}_2^\star)$ (line 3). Then, the algorithm enlarges this partial spherical convex hull with new extremal generators pointing toward the normals of the faces computed so far (lines 5,6) and using them to form new faces (line 8). Once each computed face normal has been used to attempt to expand the partial spherical convex hull, the computation ends and $\mathcal{F}$ contains the minimal set of generators. Similarly to the original QuickHull algorithm, the line 9 is the main reason for its average $O(n\log(n))$ complexity instead of $O(n^2)$: whenever a new extremal generator is identified, generators on $\mathbf{G}$ that can be proven not to be extremal generators, are removed.

Algorithm 5 Spherical Polyhedral Cone generation

1: **Inputs:** the generators $\mathbf{G} \subset \mathcal{S}^2$ and two extremal generators $\mathbf{g}_1^\star$ and $\mathbf{g}_2^\star$.
2: **Outputs:** the convex hull generators.
3: $\mathcal{F} \leftarrow \{(\mathbf{g}_1^\star, \mathbf{g}_2^\star), (\mathbf{g}_2^\star, \mathbf{g}_1^\star)\}$
4: **repeat**
5: $(\mathbf{g}_1, \mathbf{g}_2) \leftarrow$ the next unvisited pair of $\mathcal{F}$
6: $\mathbf{m} \leftarrow \frac{\mathbf{g}_1+\mathbf{g}_2}{2}$
7: $\mathbf{g}^\star \leftarrow \underset{\mathbf{g}\in\mathbf{G}\vee\langle\mathbf{g},\mathbf{m}\rangle\geq 0}{\text{argmax}} \arccos(\langle\mathbf{g}, \mathbf{a}\rangle)$ ▷ Find an extremal generator toward $\mathbf{g}_1 \times \mathbf{g}_2$ wrt. the axis $\mathbf{g}_2 - \mathbf{g}_1$.
8: **if** $\mathbf{g}^\star \neq \mathbf{g}_1$ and $\mathbf{g}^\star \neq \mathbf{g}_2$ **then**
9: Replace $\{(\mathbf{g}_1, \mathbf{g}_2)\}$ by $\{(\mathbf{g}_1, \mathbf{g}^\star), (\mathbf{g}^\star, \mathbf{g}_2)\}$ in $\mathcal{F}$ ▷ $O(1)$ if $\mathbf{G}$ is a linked list.
10: $\mathbf{G} \leftarrow \{\mathbf{g} \in \mathbf{G} \mid \mathbf{g} \notin \text{cone}(\mathbf{g}_1, \mathbf{g}^\star, \mathbf{g}_2)\}$ ▷ Remove generators contained by the convex hull.
11: **end if**
12: **until** each pair on $\mathcal{F}$ has been visited once.
return $\mathcal{F}$

3.4 Dilating polyhedral normal cones to improve conformal contacts handling

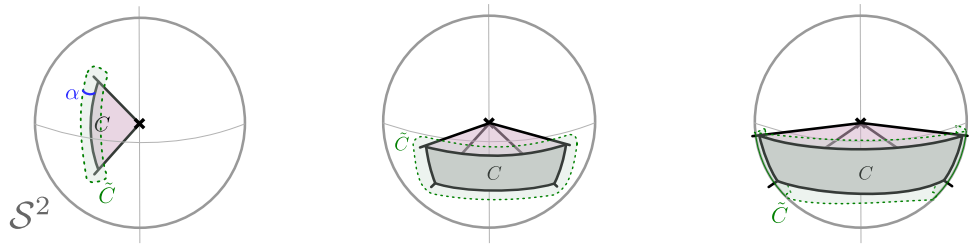
To ensure the intersection test shown in Section 3.2.2 is applicable, polyhedral cones computed in Section 3.3 must have non-empty interiors, even when bounding a part of a meridian. Moreover, enlarging bounds of the polyhedral normal cone is useful to support the computation of quasi-LMD which were detailed in Section 1.3.1.4. This is needed to ensure the stable simulation of some (near-) conformal contact configurations, e.g., when a box rests on a table and is subject to small rotations. This process of cone enlargement is called *dilation*. Given a *dilation angle* $\alpha \in \mathbb{R}^{+\star}$, dilating a polyhedral cone C amounts to computing a new polyhedral cone, called *dilated polyhedral cone*, that bounds the set:

$$\tilde{C} = \{\tilde{\mathbf{v}} \mid \exists \mathbf{v} \in C \text{ such that } \arccos(\langle\mathbf{v}, \tilde{\mathbf{v}}\rangle) \leq \alpha\}. \tag{3.13}$$

Figure 3.10 shows examples of such sets that must be bounded by a dilated polyhedral cone. Three cases may arise and are illustrated in Figure 3.10.

Let us now describe how the dilated polyhedral cone of $\tilde{C}$ with a dilation angle α is computed. At first, it is assumed that $Int(C) \neq \emptyset$ (see Figures 3.10b and 3.10c). The cone's minimal sorted set of n unit generators $G = \{\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_n\}$ is given.

When a polyhedral cone C has an empty interior, it is either a single point or an arc of $\mathcal{S}^2$. In the former case, any polyhedral cone bounding the cone of revolution centered at that



(a) (b) (c)

Figure 3.10: 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 $\mathbb{R}^3$.

Algorithm 6 Dilation of polyhedral normal cone with nonempty interior

Inputs: a minimal sorted set of unit generators $\mathbf{G}$ of a polyhedral cone C with nonempty interior.
Output: The polyhedral cone dilated by α.
$\tilde{\mathbf{N}} \leftarrow \emptyset$ ▷ The dilated polyhedral cone faces normals.
for all $(\mathbf{g}_i, \mathbf{g}_{i+1}) \in \mathbf{G}$ **do**
    $\mathbf{a} \leftarrow \mathbf{g}_{i+1} - \mathbf{g}_i$
    $\mathbf{n} \leftarrow \mathbf{g}_{i+1} \times \mathbf{g}_i$.
    $R \leftarrow$ rotation of angle α and axis $\mathbf{a}$.
    Append to $\tilde{\mathbf{N}}$ the normal $\mathbf{n}$ rotated by R.
end for
return the new polyhedral cone generators computed from the faces normals $\tilde{\mathbf{N}}$.

point and having a radius α can be used. In the latter case, the minimal set of generators of C contains exactly two elements, namely $\mathbf{g}_1$ and $\mathbf{g}_2$. The dilation using an angle α is illustrated in Figure 3.11. Figure 3.11a stands for the computation of two auxiliary generators $\mathbf{g}'_1$ and $\mathbf{g}'_2$ expressing the rotation of $\mathbf{g}_1$ and $\mathbf{g}_2$ with an angle α and $-\alpha$ around the axis $(\mathbf{g}_2 \times \mathbf{g}_1)$. Those auxiliary generators are then split into four in order to add a thickness to the cone (see Figure 3.11b). Those new generators $\{\tilde{\mathbf{g}_1}, \ldots, \tilde{\mathbf{g}_4}\}$ are obtained by rotating $\mathbf{g}'_1$ and $\mathbf{g}'_2$ with an angle α and $-\alpha$ with respect to $(\mathbf{g}_2 - \mathbf{g}_1)$.

(a) (b)

Figure 3.11: Dilation of a polyhedral cone with an empty interior.

3.5 More culling tests for Bézier curves using tangent cones and solution line cones

This section describes how tangent cones, as defined in Section 1.3.1, and the *solution line cone* [67] defined as the set of directions of all the difference vectors between all points of two (possibly infinite) sets of points, can be used to further check that two objects cannot contain a LMD. Given A and B, two sets of points, their solution line cone is defined as $S(A,B) = \{\lambda(\mathbf{a}-\mathbf{b}), \mathbf{a}\in A, \mathbf{b}\in B, \lambda\in\mathbb{R}\}$. Given a vector $\mathbf{v}\in S(A,B)$, the set $\{\lambda\mathbf{v}, \lambda\in\mathbb{R}\}$ is called a *solution line*. It is noteworthy that:

- $S(A,B) = S(B,A)$. Indeed:

$$\begin{aligned} S(B,A) &= \{\lambda(\mathbf{b}-\mathbf{a}), \mathbf{a}\in A, \mathbf{b}\in B, \lambda\in\mathbb{R}\}, \\ &= \{(-\lambda)(\mathbf{a}-\mathbf{b}), \mathbf{a}\in A, \mathbf{b}\in B, \lambda\in\mathbb{R}\}. \end{aligned}$$

  And, because $(-\lambda)\in\mathbb{R}$, it follows that: $-\lambda = \lambda'$, $\lambda'\in\mathbb{R}_+$:

$$\begin{aligned} &= \{\lambda'(\mathbf{a}-\mathbf{b}), \mathbf{a}\in A, \mathbf{b}\in B, \lambda'\in\mathbb{R}\}, \\ &= S(A,B); \end{aligned}$$

- As shown in Figure 3.12, the solution line cone can be seen as the union of all the lines passing through the points of the Minkowski sum $(A\oplus -B)$, i.e., $A\oplus -B = \{\mathbf{a}-\mathbf{b}, \mathbf{a}\in A, \mathbf{b}\in B\}$ with $-B = \{-\mathbf{b} \mid \mathbf{b}\in B\}$. We introduce the notation $A\ominus B$ which we refer to as the *Minkowski difference* of A and B given by $A\ominus B = A\oplus -B$;

- $S(A,B)$ is independent from the origin because translating both objects using the same vector $\mathbf{v}\in\mathbb{R}^3$ does not modify the solution line cone, i.e., $S(A+\mathbf{v}, B+\mathbf{v}) = S(A,B)$, where: $A+\mathbf{v} = \{\mathbf{a}+\mathbf{v} \mid \mathbf{a}\in A\}$ (resp. $B+\mathbf{v} = \{\mathbf{b}+\mathbf{v} \mid \mathbf{b}\in B\}$) designate the points of A (resp. B) translated by $\mathbf{v}\in\mathbb{R}^3$. This is enforced by the fact that:

$$\begin{aligned} (A+\mathbf{v})\ominus(B+\mathbf{v}) &= \{(\mathbf{a}+\mathbf{v})-(\mathbf{b}+\mathbf{v}), \mathbf{a}\in A, \mathbf{b}\in B\}, \\ &= \{\mathbf{a}-\mathbf{b}+\mathbf{v}-\mathbf{v}, \mathbf{a}\in A, \mathbf{b}\in B\}, \\ &= \{\mathbf{a}-\mathbf{b}, \mathbf{a}\in A, \mathbf{b}\in B\}, \\ &= A\ominus B. \end{aligned}$$

  Therefore, the reference frame of A and B on Figures 3.12a and 3.12c is not required.

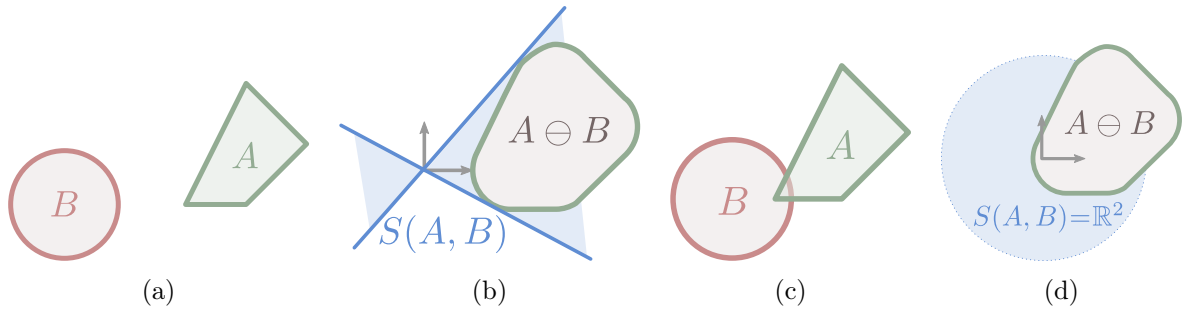


Figure 3.12: (a) Two sets $A, B\subset\mathbb{R}^2$. (b) Their Minkowski difference $A\ominus B$ and the corresponding solution line cone $S(A,B)$. (c, d) If A and B are convex and have intersecting interiors, their solution line cone is the whole space, i.e., $S(A,B) = \mathbb{R}^2$ for a 2D scenario.

Let us now describe how tangent cones and solution line cones can be used to design a conservative test of non-existence of LMD between two features. Initially, this has been introduced loosely by Johnson et al. [70] but only for LMD computation between a point and a curve or a surface. A similar idea appears in Johnson et al. [67] as well for the case of two polyhedral models but lacks proper formalism.

Let $\mathcal{A}$ and $\mathcal{B}$ be two solids (or curves). Equation (1.23) can be split into two conditions that must be simultaneously met for two points $\mathbf{p}_\mathcal{A}$ and $\mathbf{p}_\mathcal{B}$ to be critical points:

$$\begin{aligned} \mathbf{p}_\mathcal{A} - \mathbf{p}_\mathcal{B} &\in T_\mathcal{A}(\mathbf{p}_\mathcal{A})^*, \\ \mathbf{p}_\mathcal{A} - \mathbf{p}_\mathcal{B} &\in -T_\mathcal{B}(\mathbf{p}_\mathcal{B})^*. \end{aligned} \tag{3.14}$$

Considering curves and surfaces, at least of class C^1, those conditions are themselves equivalent to the following ones because of Equation (1.22):

$$\begin{aligned} \exists \mathbf{t}_\mathcal{A} \in T_\mathcal{A}(\mathbf{p}_\mathcal{A}), \langle \mathbf{t}_\mathcal{A}, \mathbf{p}_\mathcal{A} - \mathbf{p}_\mathcal{B} \rangle = 0, \\ \exists \mathbf{t}_\mathcal{B} \in T_\mathcal{B}(\mathbf{p}_\mathcal{B}), \langle \mathbf{t}_\mathcal{B}, \mathbf{p}_\mathcal{A} - \mathbf{p}_\mathcal{B} \rangle = 0. \end{aligned} \tag{3.15}$$

Thus, it comes that a pair of C^1 curves or surfaces $\mathcal{A}$ and $\mathcal{B}$ cannot contain local minimal distances if none of their solution lines is orthogonal to any of their tangents, i.e., if any of the following *orthogonality conditions* is met:

$$\begin{aligned} \forall \mathbf{s} \in S(\mathcal{A}, \mathcal{B}), \forall \mathbf{t}_\mathcal{A} \in T_\mathcal{A}(\mathcal{A}), \langle \mathbf{t}_\mathcal{A}, \mathbf{s} \rangle \neq 0, \\ \forall \mathbf{s} \in S(\mathcal{A}, \mathcal{B}), \forall \mathbf{t}_\mathcal{B} \in T_\mathcal{B}(\mathcal{B}), \langle \mathbf{t}_\mathcal{B}, \mathbf{s} \rangle \neq 0, \end{aligned} \tag{3.16}$$

where $T_\mathcal{A}(\mathcal{A}) = \bigcup_{\mathbf{p}_\mathcal{A} \in \mathcal{A}} T_\mathcal{A}(\mathbf{p}_\mathcal{A})$ (resp. $T_\mathcal{B}(\mathcal{B}) = \bigcup_{\mathbf{p}_\mathcal{B} \in \mathcal{B}} T_\mathcal{B}(\mathbf{p}_\mathcal{B})$) is the union of all the tangent cones of all points of $\mathcal{A}$ (resp. $\mathcal{B}$).

Now, those cones may have complex shapes and even non-convex ones. Thus, existing literature bound them with cones of revolution [67] and design a conservative test to check if any inequality of Equation (3.16) is met. Details are provided in Section 3.5.1. Those bounds are then improved using polyhedral cones in Section 3.5.2.

On the one hand, designing a culling test based on Equation (3.16) is essential for non-rectilinear curves that do not represent a BRep edge, e.g., deformable curves used for the simulation of beams, since the intersection test presented in Section 3.2.2 would be inefficient on them. Indeed, each point of such a curve has a Gauss map that spans an entire meridian of $\mathcal{S}^2$, which cannot be bounded tightly with a dilated polyhedral cone since it would have to contain $\mathbb{R}^3$ completely because of its convexity. Therefore, the following definitions and results are focused on Bézier curves even if most of them are applicable to any C^1 curve or surface. Figure 3.3 summarizes which types of cones are used for the culling tests depending on the types of feature areas involved in the LMD computation.

On the other hand, it likely that computing solution line cones for pairs of BRep features would not be worth it in practice since the intersection test based on polyhedral normal cone is already very discriminative.

3.5.1 The existing: tangent cones and solution line cones of revolution

The simplest and only method described in the literature so far, is the use of cones of revolution for bounding the tangent cones and solution lines cones. It was first presented in [70] for projecting a point on free-form curves and surfaces. A similar concept to solution line cones also exists under the name of *dual view cone* between two surfaces/curves [67].

3.5.1.1 Orthogonality test between cones of revolution

With a tangent cone and a solution line cone available, an efficient test can be proposed to check whether the orthogonality conditions of Equation (3.16) hold for some bounded subsets of curves. Such a test has been designed by Johnson et al. [70] when bounding the exact solution line cones and the tangent cones with cones of revolution. Let $C_{S(\mathcal{A},\mathcal{B})}$ be a cone of revolution bounding the solution line cone formed by $\mathcal{A}$ and $\mathcal{B}$. In addition, let $C_{T_\mathcal{A}(\mathcal{A})}$ (resp. $C_{T_\mathcal{B}(\mathcal{B})}$) be a cone of revolution bounding all the tangents of $\mathcal{A}$ (resp. $\mathcal{B}$). The conditions of Equation (3.16) can be rewritten as:

$$\forall \mathbf{s} \in C_{S(\mathcal{A},\mathcal{B})}, \forall \mathbf{t}_\mathcal{A} \in T_\mathcal{A}(\mathcal{A}), \langle \mathbf{t}_\mathcal{A}, \mathbf{s} \rangle \neq 0, \tag{3.17}$$

$$\forall \mathbf{s} \in C_{S(\mathcal{A},\mathcal{B})}, \forall \mathbf{t}_\mathcal{B} \in T_\mathcal{B}(\mathcal{B}), \langle \mathbf{t}_\mathcal{B}, \mathbf{s} \rangle \neq 0. \tag{3.18}$$

Let $\mathbf{v}_S, \mathbf{v}_\mathcal{A}$, and $\mathbf{v}_\mathcal{B}$ be the principal axis of $C_{S(\mathcal{A},\mathcal{B})}, C_{T_\mathcal{A}(\mathcal{A})}$, and $C_{T_\mathcal{B}(\mathcal{B})}$, respectively and $\theta_S, \theta_\mathcal{A}, \theta_\mathcal{B}$ be their apex half-angles. Then, Equation (3.17) is satisfied if and only if any of the two following conditions are verified:

$$\text{angle}(\mathbf{v}_S, \mathbf{v}_\mathcal{A}) - \theta_S - \theta_\mathcal{A} > \frac{\pi}{2}, \tag{3.19}$$

$$\text{angle}(\mathbf{v}_S, \mathbf{v}_\mathcal{A}) + \theta_S + \theta_\mathcal{A} < \frac{\pi}{2}. \tag{3.20}$$

Similarly, Equation (3.18) is equivalent to:

$$\text{angle}(\mathbf{v}_S, \mathbf{v}_\mathcal{B}) - \theta_S - \theta_\mathcal{B} > \frac{\pi}{2}, \tag{3.21}$$

$$\text{angle}(\mathbf{v}_S, \mathbf{v}_\mathcal{B}) + \theta_S + \theta_\mathcal{B} < \frac{\pi}{2}. \tag{3.22}$$

This can be seen by expanding Equations (3.17) and (3.18) with the definition of cones of revolution. For example, Equations (3.19) and (3.20) are obtained from Equation (3.17) with:

$$\forall \mathbf{s} \in C_{S(\mathcal{A},\mathcal{B})}, \forall \mathbf{t}_\mathcal{A} \in T_\mathcal{A}(\mathcal{A}), \langle \mathbf{t}_\mathcal{A}, \mathbf{s} \rangle \neq 0, \tag{3.23}$$

$$\Leftrightarrow \forall \mathbf{s} \in C_{S(\mathcal{A},\mathcal{B})}, \forall \mathbf{t}_\mathcal{A} \in T_\mathcal{A}(\mathcal{A}), \text{angle}(\mathbf{t}_\mathcal{A}, \mathbf{s}) \neq \frac{\pi}{2}, \tag{3.24}$$

$$\Leftrightarrow \begin{cases} \forall \mathbf{s} \in C_{S(\mathcal{A},\mathcal{B})}, \forall \mathbf{t}_\mathcal{A} \in T_\mathcal{A}(\mathcal{A}), \text{angle}(\mathbf{t}_\mathcal{A}, \mathbf{s}) > \frac{\pi}{2}, \\ \text{or} \quad \forall \mathbf{s} \in C_{S(\mathcal{A},\mathcal{B})}, \forall \mathbf{t}_\mathcal{A} \in T_\mathcal{A}(\mathcal{A}), \text{angle}(\mathbf{t}_\mathcal{A}, \mathbf{s}) < \frac{\pi}{2}, \end{cases} \tag{3.25}$$

$$\Leftrightarrow \begin{cases} \min\limits_{\mathbf{s} \in C_{S(\mathcal{A},\mathcal{B})}, \mathbf{t}_\mathcal{A} \in T_\mathcal{A}(\mathcal{A})} \text{angle}(\mathbf{t}_\mathcal{A}, \mathbf{s}) > \frac{\pi}{2}, \\ \text{or} \quad \max\limits_{\mathbf{s} \in C_{S(\mathcal{A},\mathcal{B})}, \mathbf{t}_\mathcal{A} \in T_\mathcal{A}(\mathcal{A})} \text{angle}(\mathbf{t}_\mathcal{A}, \mathbf{s}) < \frac{\pi}{2}, \end{cases} \tag{3.26}$$

$$\Leftrightarrow \begin{cases} \text{angle}(\mathbf{v}_S, \mathbf{v}_\mathcal{A}) - \theta_S - \theta_\mathcal{A} > \frac{\pi}{2}, \\ \text{or} \quad \text{angle}(\mathbf{v}_S, \mathbf{v}_\mathcal{A}) + \theta_S + \theta_\mathcal{A} < \frac{\pi}{2}. \end{cases} \tag{3.27}$$

The transition from Equation (3.26) to Equation (3.27) is obtained from the fact that no vector of $C_{S(\mathcal{A},\mathcal{B})}$ (resp. $T_\mathcal{A}(\mathcal{A})$) forms an angle greater than θ_S (resp. $\theta_\mathcal{A}$) with their axis $\mathbf{v}_S$ (resp. $\mathbf{v}_\mathcal{A}$).

3.5.1.2 Tangent cones of revolution for C^1 Bézier curves

The tangent cone (defined in Section 1.3.1.3) at any point $\mathbf{p}_\mathcal{A}$ of a C^1 curve segment $\gamma_\mathcal{A} : \mathcal{D}_\mathcal{A} \to \mathbb{R}^2$ is the line containing its tangent vector at that point (see Figure 3.13a), i.e., given a $t \in \mathbb{R}$:

$$T_{\gamma_\mathcal{A}}(\gamma_\mathcal{A}(t)) = \{\lambda \gamma'_\mathcal{A}(t) \mid \lambda \in \mathbb{R}\}. \tag{3.28}$$

Thus, our objective is to compute a bound of the tangent cones of all the points of $\gamma_{\mathcal{A}}$, i.e., a bound of:

$$T_{\gamma_{\mathcal{A}}}(\gamma_{\mathcal{A}}) = \bigcup_{t \in \mathcal{D}_{\mathcal{A}}} T_{\gamma_{\mathcal{A}}}(\gamma_{\mathcal{A}}(t)) \tag{3.29}$$

Computing $T_{\gamma_{\mathcal{A}}}(\gamma_{\mathcal{A}})$ amounts to bound the directions of its derivatives at all its points. Assuming $\gamma_{\mathcal{A}}$ is a Bézier curve of degree n with control points $\{\mathbf{a}_0, \ldots, \mathbf{a}_n\}$:

$$\gamma_{\mathcal{A}}(t) = \sum_{i=0}^{n} B_{i,n}(t)\mathbf{a}_i, \tag{3.30}$$

its derivative $\gamma'_{\mathcal{A}} : \mathcal{D}_{\mathcal{A}} \to \mathbb{R}^3$ can be expressed as a Bézier curve of degree $(n-1)$ called its *hodograph*:

$$\gamma'_{\mathcal{A}}(t) = n \sum_{i=0}^{n-1} B_{i,n-1}(t)(\mathbf{a}_{i+1} - \mathbf{a}_i). \tag{3.31}$$

Because the hodograph takes itself the form of a Bézier curve (see Figure 3.13b), it benefits the convex hull property [107]. Hence, it can be bounded by a cone of revolution $C^1_{\mathcal{A}}$ containing its control points as shown in Figure 3.13c. Doing so provides bounds of all the (oriented) tangents at all points of $\gamma_{\mathcal{A}}$. However, this provides only half of the lines spanned by the tangent cones defined by Equation (3.28). To bound the other halves, it is necessary to compute an additional cone of revolution $C^2_{\mathcal{A}} = -C^1_{\mathcal{A}}$ that contains the control points of $-\gamma'_{\mathcal{A}}(t)$ as shown in Figure 3.13d. That way, we have: $T_{\gamma_{\mathcal{A}}}(\gamma_{\mathcal{A}}) \subset (C^1_{\mathcal{A}} \cup C^2_{\mathcal{A}})$.

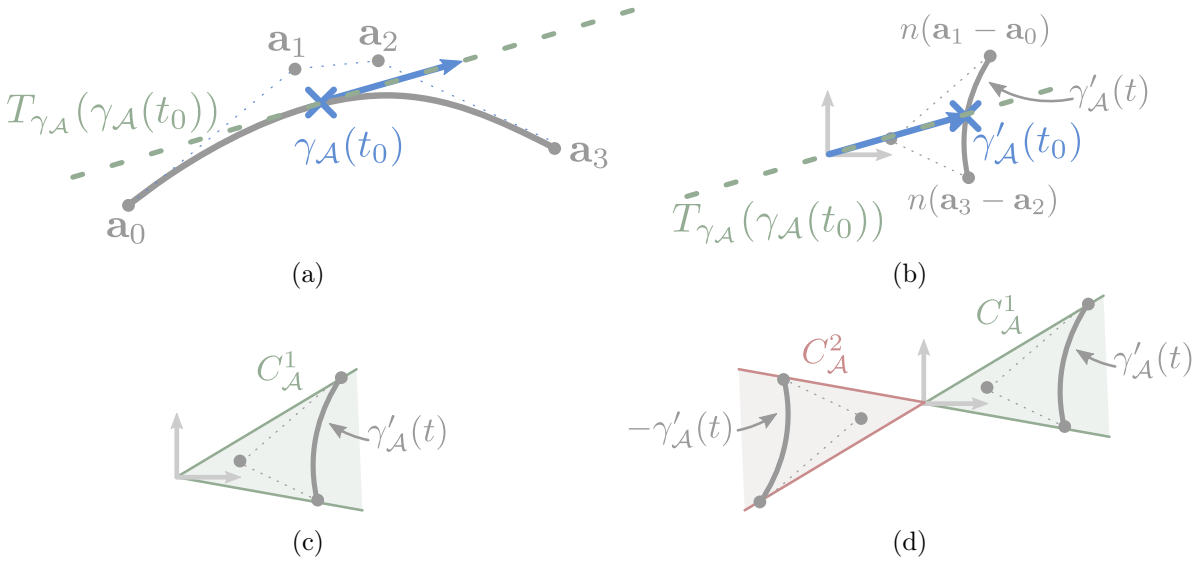


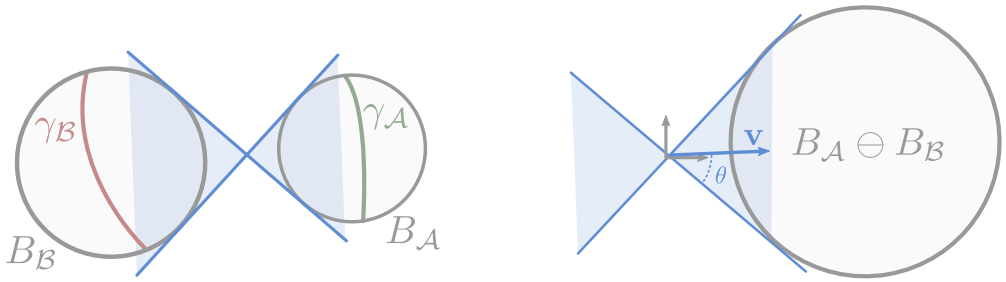
Figure 3.13: (a) A Bézier curve $\gamma_{\mathcal{A}}$ of degree $n = 3$, its control points $\mathbf{a}_i$, and its derivative at the point $\gamma_{\mathcal{A}}(t_0)$. The tangent cone at $\gamma_{\mathcal{A}}(t_0)$ is the dotted line $T_{\gamma_{\mathcal{A}}}(\gamma_{\mathcal{A}}(t_0))$. This first drawing has been scaled by a factor of 3. (b) The hodograph a' seen as a Bézier curve with control points $n(\mathbf{a}_{i+1} - \mathbf{a}_i)$. The derivative at t_0 corresponds to the vector $\gamma'_{\mathcal{A}}(t_0)$. (c) The cone of revolution that bounds the hodograph. (b) Two cones of revolution are necessary to bound $T_{\gamma_{\mathcal{A}}}(\gamma_{\mathcal{A}})$ completely.

3.5.1.3 Solution line cones of revolution for two C^1 Bézier curves

The solution line cone is obtained by finding a double revolution cone tangent to the bounding sphere of both curves simultaneously as shown in Figure 3.14a. A simpler approach for producing

an equally good result is to bound the Minkowski difference of both bounding spheres with a revolution cone (Figure 3.14a). Indeed, let $B_{\mathcal{A}}$ and $B_{\mathcal{B}}$ be the bounding spheres for the Bézier curves $\gamma_{\mathcal{A}}$ and $\gamma_{\mathcal{B}}$. Let us note $\mathbf{c}_{\mathcal{A}}$ and $r_{\mathcal{A}}$ (resp. $\mathbf{c}_{\mathcal{B}}$ and $r_{\mathcal{B}}$) their centers and radii, i.e., $B_{\mathcal{A}} = \{\mathbf{p} \in \mathbb{R}^3 \mid \|\mathbf{p} - \mathbf{c}_{\mathcal{A}}\| \leq r_{\mathcal{A}}\}$. Their Minkowski difference is given by: $B_{\mathcal{A}} \ominus B_{\mathcal{B}} = \{\mathbf{p} \in \mathbb{R}^3 \mid \|\mathbf{p} - (\mathbf{c}_{\mathcal{A}} - \mathbf{c}_{\mathcal{B}})\| \leq r_{\mathcal{A}} + r_{\mathcal{B}}\}$. Then, if the sphere $B_{\mathcal{A}} \ominus B_{\mathcal{B}}$ does not contain the origin, it is easy to determine that the solution line cone of revolution has its principal axis $\mathbf{v}$ and half apex angle θ given by:

$$\mathbf{v} = \frac{\mathbf{c}_{\mathcal{A}} - \mathbf{c}_{\mathcal{B}}}{\|\mathbf{c}_{\mathcal{A}} - \mathbf{c}_{\mathcal{B}}\|}, \qquad \theta = \arcsin\left(\frac{r_{\mathcal{A}} + r_{\mathcal{B}}}{\|\mathbf{c}_{\mathcal{A}} - \mathbf{c}_{\mathcal{B}}\|}\right).$$



(a) (b)

Figure 3.14: Computation of the solution line cone of revolution (a) from the bounding spheres of both curves; (b) from the Minkowski difference of the curves' bounding spheres.

3.5.2 Tighter bounds for C^1 Bézier curves with polyhedral tangent cones and polyhedral solution line cones

Tighter bounds can be obtained using polyhedral tangent cones and polyhedral solution line cones. To achieve this, an orthogonality test between two polyhedral cones and methods to compute them for C^1 Bézier curves are proposed.

3.5.2.1 Orthogonality test between two polyhedral cones

Given Equation (3.15), the difference $(\mathbf{p}_{\mathcal{A}} - \mathbf{p}_{\mathcal{B}})$ of two points $\mathbf{p}_{\mathcal{A}}$ and $\mathbf{p}_{\mathcal{B}}$, lying on the curves $\gamma_{\mathcal{A}}$ and $\gamma_{\mathcal{B}}$ and forming a LMD, is necessarily orthogonal to the tangents of both curves at those points. Thus, given a polyhedral tangent cone $C_{\mathcal{A}}$ and the solution line cone $C_{\mathcal{B}}$, the purpose of the test is to determine if any of the directions bounded by $C_{\mathcal{A}}$ can be orthogonal to a direction bounded by $C_{\mathcal{B}}$. If none can be found, then the corresponding curves cannot contain LMDs. To this end, Proposition 2 describes an *orthogonality test* between two polyhedral cones. Each cone is assumed not to contain any antipodal generator (further discussion regarding this restriction follows). The corresponding algorithm is described by Algorithm 7.

Proposition 2. *Let* $\mathbf{G}_{\mathcal{A}} = \{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n\} \subset \mathbb{R}^3$ *and* $\mathbf{G}_{\mathcal{B}} = \{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_m\} \subset \mathbb{R}^3$ *be the respective sets of* n *and* m *nonzero generators of two polyhedral cones* $C_{\mathcal{A}}$ *and* $C_{\mathcal{B}}$*. All vectors of* $C_{\mathcal{A}}$ *(resp.* $C_{\mathcal{B}}$*) are assumed not to contain any opposite directions, i.e:*

$$\forall \mathbf{a}_1, \mathbf{a}_2 \in C_{\mathcal{A}}, \text{angle}(\mathbf{a}_1, \mathbf{a}_2) < \pi,$$
$$\forall \mathbf{b}_1, \mathbf{b}_2 \in C_{\mathcal{B}}, \text{angle}(\mathbf{b}_1, \mathbf{b}_2) < \pi.$$

Then, $C_{\mathcal{A}}$ *does not contain any direction orthogonal to any direction of* $C_{\mathcal{B}} \Leftrightarrow$ *all scalar product* $\langle \mathbf{a}_i, \mathbf{b}_i \rangle, i \in [1..n], j \in [1..m]$ *are nonzero and have the same sign.*

Proof. Let $\mathbf{a}$ and $\mathbf{b}$ be two nonzero vectors of $C_{\mathcal{A}}$ and $C_{\mathcal{B}}$. They can be written as: $\mathbf{a} = \sum_i^n \alpha_i \mathbf{a}_i$ and $\mathbf{b} = \sum_j^m \beta_j \mathbf{b}_j$ with all $\alpha_i, \beta_j \in \mathbb{R}^+$ except for at least one α_i and one β_j which must be nonzero (setting all coefficient to 0 would produce a null vector which does not define any direction). Thus, their scalar product is:

$$\langle \mathbf{a}, \mathbf{b} \rangle = \sum_i^n \sum_j^m \alpha_i \beta_i \langle \mathbf{a}_i, \mathbf{b}_i \rangle . \tag{3.32}$$

Then, the purpose is to show that the condition:

$$\forall \mathbf{a}_i \in C_{\mathcal{A}}, \forall \mathbf{b}_j \in C_{\mathcal{B}}, \langle \mathbf{a}, \mathbf{b} \rangle \neq 0, \tag{3.33}$$

which is equivalent to (still assuming at least one α_i and at least one β_j is nonzero):

$$\forall \alpha_i, \beta_j \in \mathbb{R}^+, \forall i \in [1..n], j \in [1..m], \sum_i^n \sum_j^m \alpha_i \beta_j \langle \mathbf{a}_i, \mathbf{b}_j \rangle \neq 0, \tag{3.34}$$

is satisfied if and only if all $\langle \mathbf{a}_i, \mathbf{b}_j \rangle$ are nonzero and have the same sign.

Firstly, let us assume that Equation (3.34) holds. The objective is to prove that all $\langle \mathbf{a}_i, \mathbf{b}_j \rangle$ must be (1) nonzero and (2) of same sign:

1. If there exists indices (i_1, j_1) such that $\langle \mathbf{a}_{i_1}, \mathbf{b}_{j_1} \rangle = 0$, we can choose: $\mathbf{a} = \mathbf{a}_{i_1}$ and $\mathbf{b} = \mathbf{b}_{j_1}$ by setting all α_i and β_j to 0, except for α_1 and β_1 set to 1. Then, $\langle \mathbf{a}, \mathbf{b} \rangle = 0$, which contradicts Equation (3.33);

2. Assume there exists indices $i_1, i_2 \in [1..n]$ and $j_1 \in [1..m]$ such that $\langle \mathbf{a}_{i_1}, \mathbf{b}_{j_1} \rangle > 0$ and $\langle \mathbf{a}_{i_2}, \mathbf{b}_{j_1} \rangle < 0$. Let $\alpha_{i_2} = -\frac{\langle \mathbf{a}_{i_1}, \mathbf{b}_{j_1} \rangle}{\langle \mathbf{a}_{i_2}, \mathbf{b}_{j_1} \rangle} > 0$ and choose $\mathbf{a} = \mathbf{a}_{i_1} + \alpha_{i_2} \mathbf{a}_{i_2} \in C_{\mathcal{A}}$ and $\mathbf{b} = \mathbf{b}_{j_1} \in C_{\mathcal{B}}$. We have: $\langle \mathbf{a}, \mathbf{b} \rangle = 0$, which contradicts Equation (3.33). Thus, the scalar product of all generators of $C_{\mathcal{A}}$ with one chosen generator of $C_{\mathcal{B}}$ must have the same sign. The same reasoning applies by exchanging the roles of $\mathbf{a}$ and $\mathbf{b}$. Applying this observation to four arbitrarily-chosen generators $\mathbf{a}_{i_1}, \mathbf{a}_{i_2}, \mathbf{b}_{j_1}, \mathbf{b}_{j_2}$, it comes:

$$\begin{aligned}
\text{sign}\left(\langle \mathbf{a}_{i_1}, \mathbf{b}_{j_1} \rangle\right) &= \text{sign}\left(\langle \mathbf{a}_{i_2}, \mathbf{b}_{j_1} \rangle\right), \\
\text{sign}\left(\langle \mathbf{a}_{i_1}, \mathbf{b}_{j_2} \rangle\right) &= \text{sign}\left(\langle \mathbf{a}_{i_2}, \mathbf{b}_{j_2} \rangle\right), \\
\text{sign}\left(\langle \mathbf{a}_{i_1}, \mathbf{b}_{j_1} \rangle\right) &= \text{sign}\left(\langle \mathbf{a}_{i_1}, \mathbf{b}_{j_2} \rangle\right), \\
\text{sign}\left(\langle \mathbf{a}_{i_2}, \mathbf{b}_{j_1} \rangle\right) &= \text{sign}\left(\langle \mathbf{a}_{i_2}, \mathbf{b}_{j_2} \rangle\right).
\end{aligned}$$

By transitivity, all those signs are the same. This proves that all $\langle \mathbf{a}_i, \mathbf{b}_j \rangle$ must have the same sign for all i and j.

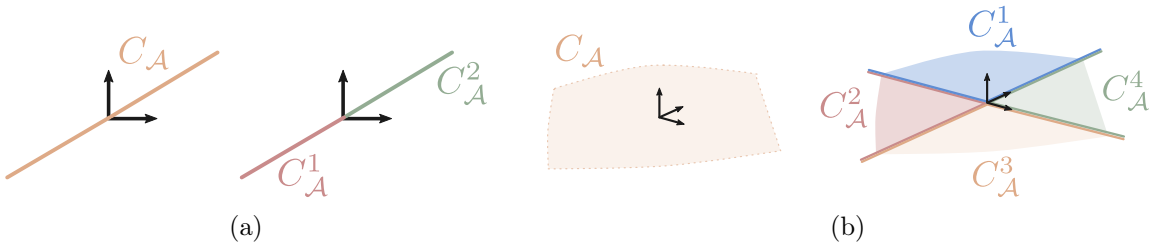
Conversely, it is clear that if all products are nonzero and have the same sign then, Equation (3.34) is necessarily satisfied because all α_i, β_j are positive and do not all vanish. □

Now, Proposition 2 is applicable only if $C_{\mathcal{A}}$ and $C_{\mathcal{B}}$ do not contain any antipodal directions. Because of their convexity, the only polyhedral cones that contain antipodal directions are those that span a subspace of $\mathbb{R}^3$ completely. As illustrated by Figure 3.15, a polyhedral cone equal to a line (resp. plane) can be split into two half-lines (resp. four quarters) to apply Proposition 2 on each of them. If one cone is equal to $\mathbb{R}^3$ then, the orthogonality test always succeeds.

Finally, let us observe that the result of Algorithm 7 is the same if $C_{\mathcal{A}}$ and/or $C_{\mathcal{B}}$ is reversed. Consequently, as long as the orthogonality of $C_{\mathcal{A}}$ and $C_{\mathcal{B}}$ is tested, there is no need to test orthogonality between, e.g., $-C_{\mathcal{A}}$ and $C_{\mathcal{B}}$ as well, even if solution line cones and tangent cones are bounded with two opposite polyhedral cones as discussed in the next paragraphs.

Algorithm 7 Polyhedral Cone Orthogonality test

Inputs: the generators $\mathbf{G}_\mathcal{A}, \mathbf{G}_\mathcal{B}$ of two polyhedral cones $C_\mathcal{A}$ and $C_\mathcal{B}$.
Output: TRUE if $\exists \mathbf{a} \in C_\mathcal{A}, \mathbf{b} \in C_\mathcal{B}$ such that $\langle \mathbf{a}, \mathbf{b} \rangle = 0$.
$d \leftarrow \langle \mathbf{a}_1, \mathbf{a}_2 \rangle$
for all $(\mathbf{a}_i, \mathbf{b}_j) \in (\mathbf{G}_\mathcal{A} \times \mathbf{G}_\mathcal{B})$ **do**
    if $d \langle \mathbf{a}_i, \mathbf{b}_j \rangle \leq 0$ **then**
        return TRUE
    end if
end for
return FALSE



(a) (b)

Figure 3.15: Polyhedral cones that contain antipodal directions. (a) A line $C_\mathcal{A}$ is split into two half line $C_\mathcal{A} = C^1_\mathcal{A} \cup C^2_\mathcal{A}$. (b) A plane $C_\mathcal{A}$ is split into four quarters $C_\mathcal{A} = C^1_\mathcal{A} \cup C^2_\mathcal{A} \cup C^3_\mathcal{A} \cup C^4_\mathcal{A}$.

3.5.2.2 Computation for Bézier curves

Computing a polyhedral tangent cone is straightforward when following a procedure similar to that of the tangent cone of revolution presented in Section 3.5.1.2: first compute the control points of the Bézier curve hodograph and use them directly as generators of the polyhedral tangent cone. Consequently, the obtained tangent bound is both tighter and cheaper to construct than the tangent cone of revolution, which necessitates further computations to find the principal axis and apex angle of the cone.

Furthermore, it is also possible to compute efficiently a tight polyhedral cone for bounding the solution line cone of two Bézier curves. First, let us observe that the Minkowksi difference of two Bézier curves $\gamma_\mathcal{A} : \mathcal{D}_\mathcal{A} \subset \mathbb{R} \to \mathbb{R}^2$ and $\gamma_\mathcal{B} : \mathcal{D}_\mathcal{B} \subset \mathbb{R} \to \mathbb{R}^2$ with control points $\{\mathbf{a}_0, \ldots, \mathbf{a}_m\}$ and $\{\mathbf{b}_0, \ldots, \mathbf{b}_m\}$, can be expressed as a Bézier surface:

$$\gamma_\mathcal{A}(u) = \sum_{i=0}^{m} B_{i,m}(u)\mathbf{a}_i,$$

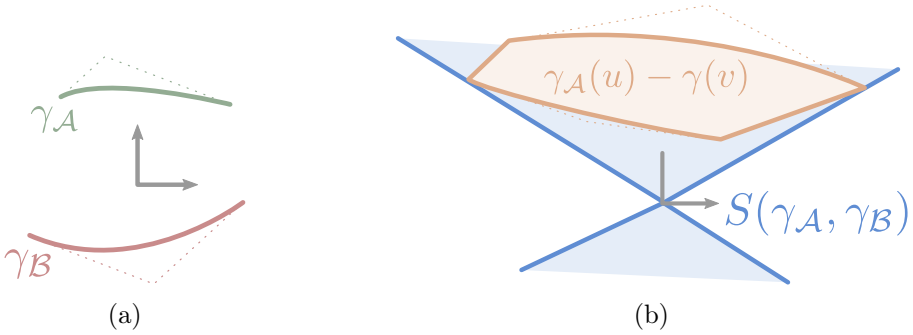
$$\gamma_\mathcal{B}(v) = \sum_{j=0}^{n} B_{j,n}(v)\mathbf{b}_j,$$

$$\sigma(u,v) = \gamma_\mathcal{A}(u) - \gamma_\mathcal{B}(v) = \sum_{i=0}^{m}\sum_{j=0}^{n} B_{i,m}(u)B_{j,n}(v)(\mathbf{a}_i - \mathbf{b}_j). \tag{3.35}$$

Then, the solution line cone is given by $S(\gamma_\mathcal{A}, \gamma_\mathcal{B}) = \{\lambda\mathcal{A} \ominus \mathcal{B}, \lambda \in \mathbb{R}\} = \{\lambda\sigma(u,v), \lambda \in \mathbb{R}, (u,v) \in \mathcal{D}_\mathcal{A} \times \mathcal{D}_\mathcal{B}\}$. Because of the convex hull property of Bézier surfaces [107], all points of $\mathcal{A} \ominus \mathcal{B}$ are bounded by the convex hull of their control points $\{(\mathbf{a}_i - \mathbf{b}_j), i \in [0,m], j \in [0,n]\}$. Thus, their solution line cone can be bounded by the polyhedral cone generated by the control points of their Minkowski difference:

$$S(\gamma_\mathcal{A}, \gamma_\mathcal{B}) \subset cone(\{\mathbf{a}_i - \mathbf{b}_j, i \in [0,m], j \in [0,n]\}). \tag{3.36}$$

Figure 3.16 shows an example of solution line cone obtained from two curves. Let us observe that in this case the bounding spheres of both curves intersect. Thus, a tight cone of revolution cannot be computed using the methods previously discussed in Section 3.5.1.3. Similarly, if the control polyhedron of two curves intersect, which is not the case in Figure 3.16a, a tight polyhedral solution line cone cannot be computed using the method presented in this section. In that case, this method outputs a cone containing $\mathbb{R}^3$ entirely.



(a) (b)

Figure 3.16: (a) Two Bézier curves $\gamma_{\mathcal{A}}$ and $\gamma_{\mathcal{B}}$. (b) Their solution line cone $S(\gamma_{\mathcal{A}}, \gamma_{\mathcal{B}})$ obtained from their difference surface $\sigma(u, v) = \gamma_{\mathcal{A}}(u) - \gamma_{\mathcal{B}}(v)$.

To summarize, tight polyhedral cone bounds for the solution line cone formed by two Bézier curves can be obtained when computing the control points of their Minkowski difference (seen as a surface) and using them as generators. This polyhedral cone bounding $S(\gamma_{\mathcal{A}}, \gamma_{\mathcal{B}})$ together with a polyhedral cone bounding all the tangents of either $\gamma_{\mathcal{A}}$ or $\gamma_{\mathcal{B}}$ can then be input to the Proposition 2 in order to perform an orthogonality test.

3.6 Conclusion

Starting with the observation that a massive amount of bounding volumes exist in the literature for bounding the space occupied by 3D objects (aka. spatial bounding volumes), this chapter highlights the fact that among the bounding volumes aimed at bounding directions like tangents and normals, only one exists in the literature: the cone of revolution. Therefore, a completely new, orientation-based bounding volume, related to concepts of convex hulls now applied to the unit sphere, has been introduced.

This new bounding volume, called a *polyhedral cone*, is much tighter than the existing cones of revolution. It is used to bound the tangent cone polars of BRep features. For the case of isolated deformable curves it is used to bound their tangent cones as well as the solution line cone formed by two curves. Moreover, a method for dilating any polyhedral cone has been described. Such a dilation allows the use of the concept of quasi-LMD recalled in Section 1.3.1.4.

Bounds of the tangent cone polars attached to areas of canonical surfaces cut along isoparametric curves are straightforward to obtain given methods for bounding meridians and lines of latitude of the unit sphere.

Bounds for edge segments can be obtained by merging polyhedral bounds of the tangent cone polar of surface areas intersecting at that edge.

Finally, tight bounds of tangent cone polar for Bézier surfaces, and bounds of solution line cones and tangent cones for Bézier curves can be obtained from the control points of their difference, normals, and tangents, formulated under Bézier form.

Those bounds help finding feature areas that cannot contain any LMD. Indeed, two tests are known to follow from the characterization of local minimal distances:

1. If no direction of one tangent cone polar is antipodal with no direction of the other tangent cone polar then, no LMD is possible for the bounded feature areas. This is the *intersection test*;

2. If no direction of one tangent cone polar is orthogonal with no direction of the solution line cone formed by the two feature areas, then no LMD is possible between them. This is the *orthogonality test*.

While those tests can be adapted to polyhedral cones, they have high polynomial complexities compared to those based on cones of revolution, which are constant-time. Therefore, practical implementations of polyhedral cones must rely on additional constructions described in the next chapter to obtain efficient timing performances while preserving much of their tightness.

Chapter 4

Achieving real-time simulations: a parallelized runtime CD pipeline with temporal coherence

This chapter details all the operations executed at runtime. Given the pre-computed elements presented in the previous chapters, and given the new positions of the objects after a time-step, the goal of the runtime phase is to compute all the LMD between each pair of bodies. The sequential simultaneous traversal of the bounding volume hierarchies that identifies all the feature areas that may contain a LMD is first presented. Then, choices regarding the exact closest point computation method between points, curves segments, and surfaces defined over a rectangular domain are detailed: analytical algorithms are always preferred whenever possible, depending on the nature of the interacting features. Some conformal contact configurations are also handled by a sample-based approach. Finally, once all the LMD footpoints are found, they have to be trimmed to select only those that do not lie in a hole. Furthermore, the computational efficiency of this pipeline can be significantly improved by introducing the assumption of temporal coherence, i.e., the fact that objects are assumed to follow a continuous motion such that successive positions do not vary too much between two time-steps. Finally, a complete parallel pipeline is presented to make this CD framework usable for real-time applications.

4.1 Description of the sequential CD pipeline

At run time, the computation of all LMDs between two BRep models $\mathcal{A}$ and $\mathcal{B}$ is performed through the three steps illustrated by Figure 4.1:

1. The two BVHs representing the BRep models are traversed simultaneously (see Figure 4.1(a)) to output all BVH leaf pairs containing feature areas that may contain a LMD. See Section 4.2 for details, i.e., pairs of leaves on which the culling tests presented in Section 2.4.3 failed;

2. A closest point algorithm is executed on the identified pairs' underlying untrimmed features (see Figure 4.1(b)), i.e., even points outside of the domains delimited by trimming curves may be output (see Section 4.3 for details);

3. Whenever one of the closest point pairs contains a point located on an edge or a vertex, further tests are applied with Equation (1.23) to make sure it is actually a LMD. Moreover,

the parametric coordinates of each closest point are tested for containment within the relevant curve/surface trimming loops (see Figure 4.1(c)) using a CPU version of Schollmeyer et al. [119]. Those validations of the computed closest points is described in Section 4.4.

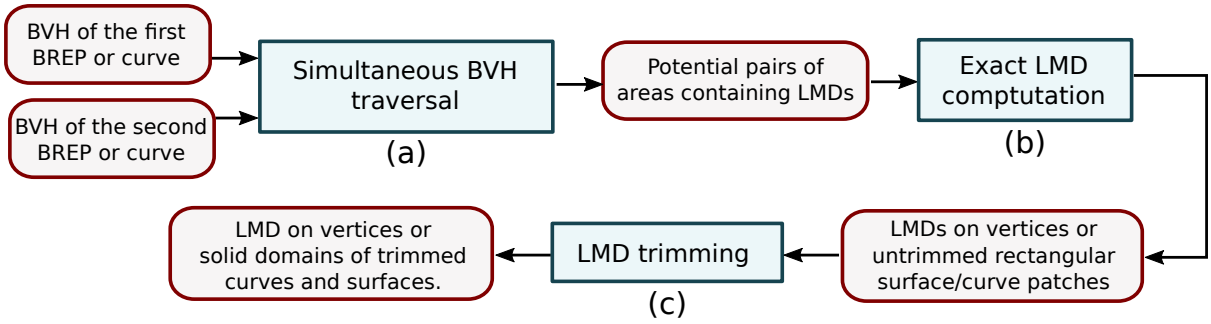


Figure 4.1: The proposed sequential collision detection pipeline that computes all LMDs between two BRep models of deformable curves.

The CD pipeline for the computation of LMD when one, or both, body models B_k is a deformable curve is a sightly modified version of Figure 4.1 where the simultaneous BVH traversal is:

- Ahead of the update of the bounding volumes stored into the pre-computed BVH for deformable curves in order to fit properly and tightly the new curve segments obtained after deformation;
- Followed, by the *dynamic* BVH traversal, i.e., for each pair of leaf reached by the original BVH traversal, the traversal is carried on with a dynamic subdivision of the deformable curves.

Because this process combines the traversal of pre-computed BVHs with the traversal of dynamically computed BVHs, this approach is identified as a hybrid approach and detailed in Section 4.3.3.

Finally, significant improvements of computations times of the whole CD pipeline are added when exploiting temporal coherence (see Section 4.5) and using parallelization (see Section 4.6).

4.2 The BVH traversal

Chapter 2 has detailed the offline computations used to prepare the data structure necessary to accelerate LMD computations at runtime. This data structure encompassing all the elements presented in Chapter 2 and Chapter 3 is the Bounding Volume Hierarchy (BVH). Each BRep model and deformable curve has one BVH that is filled with feature areas, bounding volumes, and several other attributes enumerated by Listing 2.4. This section describes how the BVH of two B-Rep solids $\mathcal{A}$ and $\mathcal{B}$ are exploited at runtime in order to locate, as efficiently as possible, areas that may contain LMD footpoints. Section 4.2.1 describes the traversal of the BVHs as it is typically performed in the literature for CD whenever BVHs are used, independently from the thosen geometric representation. Section 4.2.2 provides additional constructions to address performance concerns identified in Section 3.2.2 regarding polyhedral cones intersection tests during this traversal. Section 4.2.3 shows the benefits of supermaximal faces to avoid redundant traversals.

4.2.1 Simultaneous traversal and the Bounding Volume Test Tree

The simultaneous BVH traversal's goal is the identification of pairs of leaves corresponding to feature areas that potentially contain LMD footpoints through a simultaneous depth-first traversal. On the one hand, this identification makes use of conservative culling test (see Section 2.4.3), i.e., pairs that actually support a LMD must not be missed to avoid the generation of *false negatives*. On the other hand, it is acceptable to output pairs that do not actually contain any LMD, i.e., *false positives* are allowed.

A BVH has here a binary tree structure. Thus, traversing two of them simultaneously is the same as traversing a single one where each node describes a pair of BVH nodes (one belonging to the BVH of each BRep). This tree illustrated by Figure 4.2 is often referred to as the Bounding Volume Test Tree (BVTT). Indeed, the concept of BVTT is simply an abstract structure to help describe and understand the traversal: it is never persistent in computer memory explicitly, except for a few nodes forming a *front*. Such a front is used for temporal coherence and is detailed in Section 4.5.1. One node of a BVTT can be seen as a structure with three fields as listed by Listing 4.1.

Listing 4.1: Data structure for a node of the Bounding Volume Test Tree.

```
struct BVTTNode {
    BVHNode   bvhNode1;
    BVHNode   bvhNode2;
    BVTTStatus status;

    // Constructor from two BVTT nodes.
    BVTTNode(BVHNode node1, BVHNode node2) {
      bvhNode1 = node1;
      bvhNode2 = node2;
      status   = ToTraverse;
    }
}
```

The third field `status` and its default value `ToTraverse` are used for temporal coherence optimizations detailed in Section 4.5.1. The two children of one BVTT node are identified when visiting the children of one of its two BVH nodes, i.e., either of `bvhNode1` or of `bvhNode2` (See Algorithm 8). A simple heuristic to choose which one to traverse is to select the one with the *largest* OBB, where *largest* means the one with the greatest volume. The effect of actual choice modifies only the performances, i.e., the conservative aspect of the traversal is preserved.

The traversal using the concept of BVTT is detailed in Algorithm 9. The CullingTest procedure determines whether two BVHNode have any chance of leading to LMD footpoints. This was described in Section 2.4.3.

4.2.2 Simplified polyhedral cones and leaf-leaf tests

In practice, the cubic time complexity of the polyhedral cone intersection test described in Section 3.2.2 is too high to be affordable during the BVTT traversal. Therefore, two measures are taken to reduce its cost:

- As initially mentioned in Section 2.4.2, the intersection tests between polyhedral cones are preformed only at the leaves of the BVTT, i.e., it is the ultimate culling test before performing the exact contact point computation. Internal nodes rely on normal cones of revolution only;
- Simplify the polyhedral cones by placing an upper bound on the number of generators.

Algorithm 8 Computes a 2-element array containing the two children of a BVTT node, if any.

```
function BVTTNodeChildren(node)
    if node.bvhNode1 and node.bvhNode2 are leaves then
        ▷ This BVTTNode is a leaf, return nothing.
        return NULL
    else if node.bvhNode2 is a leaf or node.bvhNode1.obb has a greater volume than node.bvhNode2.obb then
        ▷ Traverse the first BVH node.
        child1 ← BVTTNode(node.bvhNode1.leftChild, node.bvhNode2)
        child2 ← BVTTNode(node.bvhNode1.rightChild, node.bvhNode2)
        return [child1, child2]
    else
        ▷ Traverse the second BVH node.
        child1 ← BVTTNode(node.bvhNode1, node.bvhNode2.leftChild)
        child2 ← BVTTNode(node.bvhNode1, node.bvhNode2.rightChild)
        return [child1, child2]
    end if
end function
```

Algorithm 9 Simultaneous BVH traversal, initialized by a BVTT node. The output array *newContacts* is filled with the LMD computed at the leaves of the BVTT.

```
function TraverseBVTT(node, newContacts)
    if CullingTest(node.bvhNode1, node.bvhNode2) then
        ▷ No LMD possible, do nothing.
    else
        children ← BVTTNodeChildren(node)
        if children = NULL then
            Add to newContacts the closest points on node.bvhNode1 and node.bvhNode2
        else
            ▷ Recursive calls on the children.
            TraverseBVTT(children[0], newContacts)
            TraverseBVTT(children[1], newContacts)
        end if
    end if
end function
```

A maximum of 4 generators is adopted by analogy with a 2D OBB which contains 4 vertices and is known to be a good compromise between tightness and complexity.

Limiting the number of sides of the polyhedral normal cones amounts to bounding the cone computed by the spherical convex hull described by Algorithm 5 with another cone having only four sides, i.e., its generator set has four elements. Computing efficiently the smallest four-sided cone is not straightforward and, due to time constraints, a combinatorial approach has been set up and the minimization of the algorithmic complexity has been left for future works.

The main idea of this construction method is to establish an analogy between the four-sided polyhedral cone, which is called a *Spherical Oriented Bounding Box* (SOBB) in this manuscript, and a 2D OBB. A simple brute-force method to compute the minimal 2D OBB of a convex polygon is illustrated in Figure 4.3 and detailed by Algorithm 10. It follows the property along which the minimal 2D OBB has necessarily a side that contains at least one side of the bounded polygon [45]. The method (see Algorithm 11) to compute an SOBB is similar to that of a 2D OBB when replacing the concept of *extremal point* by the concept of *extremal generator* presented in Section 3.3.3. While this brute-force procedure is $O(n^2)$, n being the number of generators, it could be much improved to work in $O(n)$ time when testing each side in a convenient order and exploiting an adaption of the *rotating calipers* method [135] on $\mathcal{S}^2$.

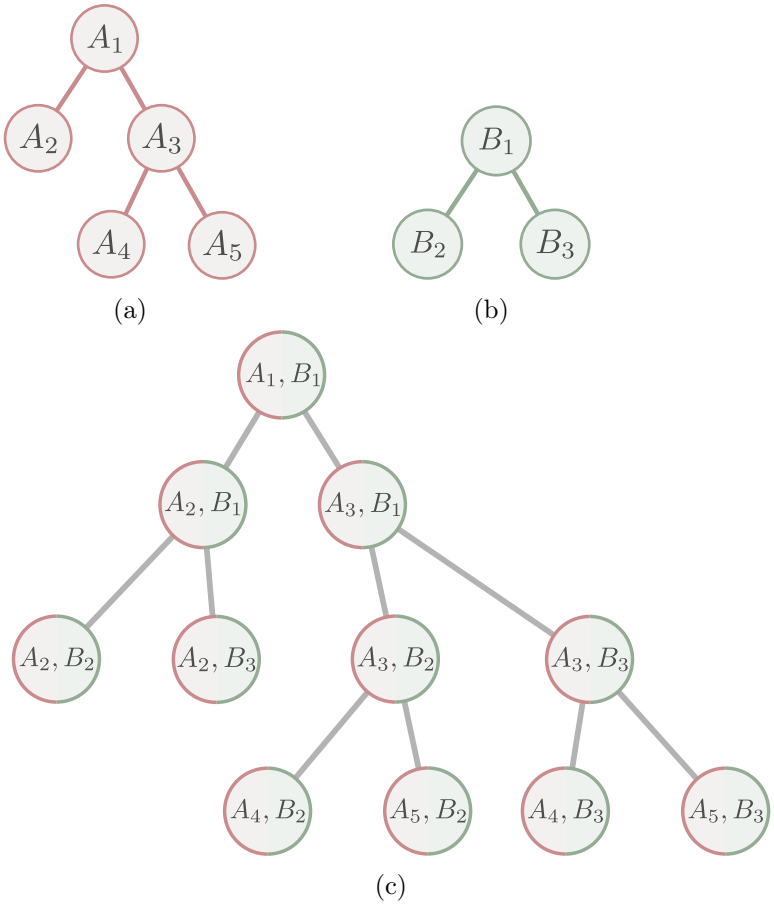


Figure 4.2: (a, b) Two small BVH with nodes identified by numbers and letters. (c) A BVTT with each node grouping one node from (a) and one from (b).

Algorithm 10 Brute-force minimal 2D OBB computation algorithm

Inputs: The sorted list of vertices V of a convex polygon.
Outputs: The minimal OBB containing V.
$Res \leftarrow$ the degenerate OBB containing $\mathbb{R}^2$.
for all pair $(\mathbf{v}_i, \mathbf{v}_{i+1}) \in V$ of **successive** vertices. **do**
    $\mathbf{n}_1 \leftarrow \mathbf{v}_{i+1} - \mathbf{v}_i$
    $\mathbf{n}_2 \leftarrow$ any nonzero vector orthogonal to $\mathbf{n}_1$.
    $(\mathbf{n}_3, \mathbf{n}_4) \leftarrow (-\mathbf{n}_1, -\mathbf{n}_2)$
    $\mathbf{v}_k^\star \leftarrow$ extremal point of V along $\mathbf{n}_k, k \in [1, 4]$.
    $OBB \leftarrow$ the OBB with four faces f_k with normals $\mathbf{n}_k$ and passing through $\mathbf{v}_k^\star$.
    $Res \leftarrow OBB$ if it has a smaller area than Res.
end for
return Res

One iteration of the Algorithm 11 is illustrated step-by-step in Figure 4.4 for a polyhedral cone with five generators, i.e., $G = \{\mathbf{g}_1, \ldots, \mathbf{g}_5\}$. Let us unfold the following steps:

(a) Let us consider any two successive generators $\mathbf{g}_1$ and $\mathbf{g}_2$ and compute the axis $\mathbf{a}_1 = \mathbf{g}_2 - \mathbf{g}_1$ (and $\mathbf{a}_2$ pointing toward the opposite direction);

(b) Set the extremal generators $\mathbf{g}_1^\star$ (resp. $\mathbf{g}_2^\star$) toward $\mathbf{g}_2 \times \mathbf{g}_1$ (resp. $\mathbf{g}_1 \times \mathbf{g}_2$) with respect to $\mathbf{a}_1$ (resp. with respect to $\mathbf{a}_2 := -\mathbf{a}_1$);

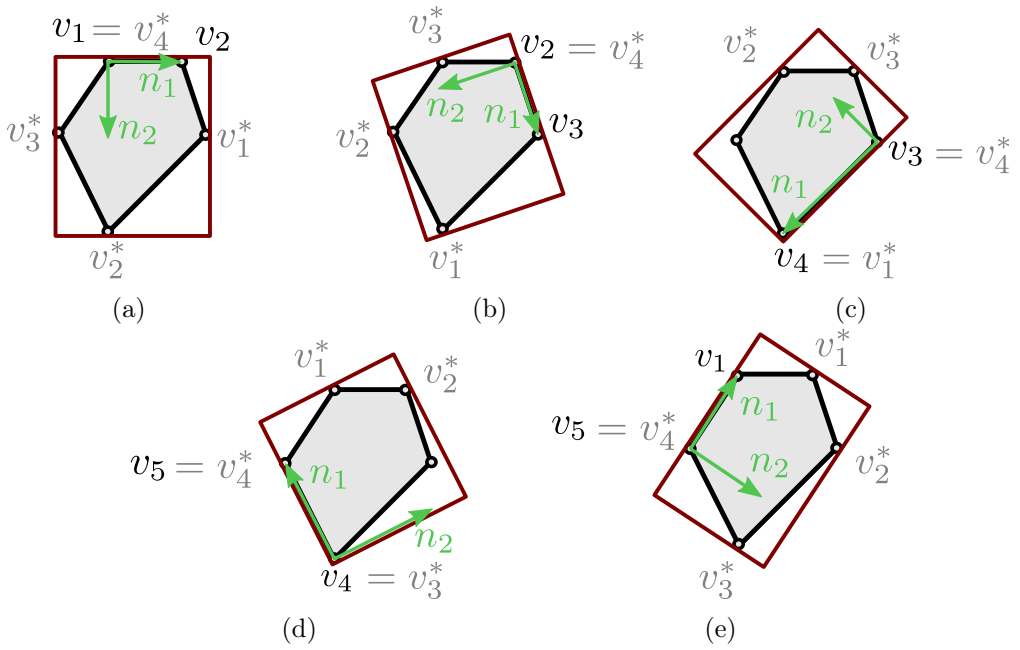


Figure 4.3: The five iterations of Algorithm 10 for the 2D OBB computation for a convex polygon. The goal is the selection of the smallest one among the five options, each with a side containing an edge of the polygon shown in green.

Algorithm 11 Brute-force SOBB computation algorithm

Inputs: The sorted minimal generator set G of a polyhedral cone C.
Outputs: An SOBB containing G.
$Res \leftarrow$ the degenerate polyhedral cone containing $\mathbb{R}^3$.
for all pair $(\mathbf{g}_i, \mathbf{g}_{i+1}) \in G$ of **successive** generators. **do**
    $\mathbf{n}_1 \leftarrow \mathbf{g}_{i+1} \times \mathbf{g}_i$
    $\mathbf{a}_1 \leftarrow \mathbf{g}_{i+1} - \mathbf{g}_i$
    $(\mathbf{n}_2, \mathbf{a}_2) \leftarrow (-\mathbf{n}_1, -\mathbf{a}_1)$
    $\mathbf{g}_k^\star \leftarrow$ extremal generator of C toward $\mathbf{n}_k$ wrt. $\mathbf{a}_k, k \in [1, 2]$.

    $\mathbf{n}_3 \leftarrow \mathbf{g}_1^\star \times \mathbf{g}_2^\star$
    $\mathbf{a}_3 \leftarrow \mathbf{g}_2^\star - \mathbf{g}_1^\star$
    $(\mathbf{n}_4, \mathbf{a}_4) \leftarrow (-\mathbf{n}_3, -\mathbf{a}_3)$
    $\mathbf{g}_k^\star \leftarrow$ extremal generator of C toward $\mathbf{n}_k$ wrt. $\mathbf{a}_k, k \in [3, 4]$.

    $SOBB \leftarrow$ the polyhedral cone with four faces f_k containing $\mathbf{a}_k$ and $\mathbf{g}_k^\star$.
    $Res \leftarrow SOBB$ if it has a smaller intersection area with $\mathcal{S}^2$.
end for
return Res

(c) Compute the axis $\mathbf{a}_3$ and finds the extremal generators $\mathbf{g}_3^\star$ and $\mathbf{g}_4^\star$;

(d) Finally, an SOBB is obtained from the planes passing through the computed axes and the related extremal points.

While this method computes a polyhedral cone with only four generators that bounds tightly the original one, it is not necessarily the minimal one. Several arbitrary choices are made in this algorithm. For example: $\mathbf{g}_2^\star$ could have been chosen equal to $\mathbf{g}_2$ (instead of $\mathbf{g}_1$), which would

have resulted in a completely different cone. The design of an algorithm producing the minimal polyhedral cone with at most four generators is left for future works.

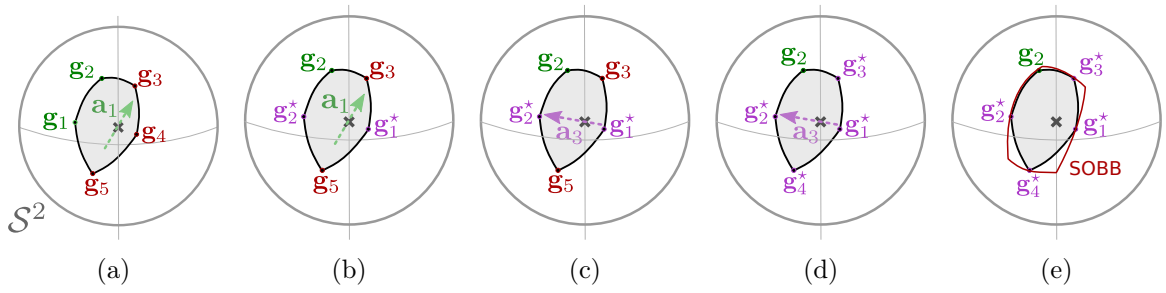


Figure 4.4: One iteration of Algorithm 11.

4.2.3 Avoiding redundant computations with supermaximal faces

Preserving a link between a feature area and the supermaximal feature (see the `supermax_feature` field of the `BVHLeaf` structure detailed in the Listing 2.4) plays an important role during BVTT traversal. Indeed, it would be a loss of computational efficiency to compute multiple times all contact points between $\mathcal{A}$ and $\mathcal{B}$ for which analytical solution exist. This may happen in a typical BVTT traversal as several nodes of the BVTT may contain pieces of the same pair of features.

For example, let $(l^1_\mathcal{A}, l^1_\mathcal{B})$ and $(l^2_\mathcal{A}, l^2_\mathcal{B})$ be two pairs of BVH leaves potentially containing LMD footpoints. Let us assume that $l^1_\mathcal{A}$ and $l^2_\mathcal{A}$ contain the areas $f^1_\mathcal{A}$ and $f^2_\mathcal{A}$ of the faces $F^1_\mathcal{A}$ and $F^2_\mathcal{A}$, themselves belonging to the same supermaximal face, i.e., $f^1_\mathcal{A} \subset F^1_\mathcal{A} \in {}^sF^1_\mathcal{A}$ and $f^2_\mathcal{A} \subset F^2_\mathcal{A} \in {}^sF^1_\mathcal{A}$. Similarly, let us assume that the leaves $l^1_\mathcal{B}$ and $l^2_\mathcal{B}$ contain the edges segments $e^1_\mathcal{B} \subset E^1_\mathcal{B} \in {}^sE^1_\mathcal{B}$ and $e^2_\mathcal{B} \subset E^2_\mathcal{B} \in {}^sE^1_\mathcal{B}$ as depicted in Figure 4.5.

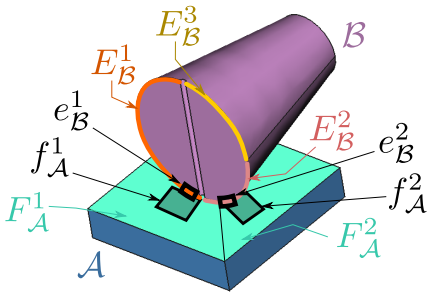


Figure 4.5: Segments $e^1_\mathcal{B}$ and $e^2_\mathcal{B}$ of a circular supermaximal edge ${}^sE^1_\mathcal{B} = \{E^1_\mathcal{B}, E^2_\mathcal{B}, E^3_\mathcal{B}\}$ in potential contact with the areas $f^1_\mathcal{B}$ and $f^2_\mathcal{B}$ of a planar supermaximal face ${}^sF^1_\mathcal{A} = \{F^1_\mathcal{A}, F^2_\mathcal{A}\}$.

Now, let us assume that there exists a method able to compute simultaneously and robustly all the LMDs between the underlying geometries of ${}^sF^1_\mathcal{A}$ and ${}^sE^1_\mathcal{B}$. This is typically the case for canonical features, as discussed in Section 4.3, when referring to *analytical methods*. Then, as soon as the step (a) of Figure 4.1 (the BVTT traversal) encounters the pair $(l^1_\mathcal{A}, l^1_\mathcal{B})$, it is saved for later processing by the step (b) of Figure 4.1 (the exact LMD footpoints computation stage). In addition, a reference to the pair of supermaximal features $({}^sF^1_\mathcal{A}, {}^sE^1_\mathcal{B})$ is saved into a set $\mathcal{E}$ (which typically takes the form of a hash-set on popular programming languages' standard libraries). Then, when the other candidate pair $(l^2_\mathcal{A}, l^2_\mathcal{B})$ is found by the step (a), it can safely be ignored, i.e., it does not have to be saved for later processing by the step (b). Indeed, the step (b) will already compute all the LMD footpoints between the underlying geometries of ${}^sF^1_\mathcal{A}$ and ${}^sE^1_\mathcal{B}$ when it processes the pair $(l^1_\mathcal{A}, l^1_\mathcal{B})$. Processing the pair $(l^2_\mathcal{A}, l^2_\mathcal{B})$ as well would have been

wasteful since the same LMD footpoints would have been found as they belong to the same supermaximal features.

4.3 Exact contact points computation

Once potential contact pairs have been identified, the actual contact points (if any) must be computed. This is achieved differently for non-deformable features (see Section 4.3.1) and for deformable curves (see Section 4.3.3). Indeed, the latter requires further analysis at runtime to isolate contact points on curves after their deformations.

As mentioned in Section 2.4.1, recall that the contact point computation is performed on surface areas with a rectangular sub-domain, i.e., trimming curves are ignored until after contact points are actually computed. Therefore, the search of LMD in Section 4.3.1 is limited to rectangular domains only. Processing the trimming curves is described in Section 4.4.1.

4.3.1 Algorithmic choices for non-deformable features

Analytic algorithms that compute simultaneously all closest points between canonical features often exist and are much cheaper than more general iterative methods. Therefore, whenever possible, a method dedicated to the types of features involved in the closest points computation is preferred. This choice of exact closest point algorithm depending on the feature geometry type is summarized in Table 4.1. Each intersection of a row and a column indicates the corresponding LMD computation approach, which is either:

- An analytic method using a specific algorithm like normal matching [27] or reducing the problem into a simpler straightforward one. For example, the Cylinder-Cylinder case is equivalent to computing closest points between two lines. The *Extr. Pt.* entry states that the algorithm has to compute the extremal point only (which definition was given in Section 3.3.3) of the torus or conic section toward the direction opposite to the plane's normal;
- The resolution of a quartic polynomial to compute its analytic solutions or use fast iterative methods [127] that can yield more robust results;
- An iterative root-finding method whenever the LMD computation amounts to finding roots of a polynomial of degree higher than 4 [27] for which no analytic solution exist.

Not represented in Table 4.1 are Bézier curves and surfaces, general surfaces of revolution, and surfaces of linear extrusion, for which an iterative method is systematically used.

	Cone	Cyl./Line	Plane	Torus/Circle	Sphere/Point	Conic section
Cone	Analytic [27]	Analytic [27]	N/A	Iterative	Point-Line	Iterative
Cyl./Line	sym	Line-Line	N/A	Quartic	Point-Line	Iterative
Plane	sym	sym	N/A	Extremal point	Point-Plane	Extremal point
Torus	sym	sym	sym	Iterative	Point-Circle	Iterative
Sphere/Point	sym	sym	sym	sym	Point-Point	Point-Conic sctn.
Conic section	sym	sym	sym	sym	sym	Iterative

Table 4.1: Computation methods for LMDs depending on the type of surfaces. *Pnt*, *Cyl.*, and *Conic sctn.* indicate respectively points, cylinders, and conic sections. sym indicates that this table is symmetric so the lower-triangular part has to be seen as identical as its upper-triangular part.

Whenever an iterative method is necessary, the Newton method can be used to find critical points of the squared distance function between the two features by canceling its gradient using

an iterative root-finding algorithm like the Gauss-Newton method. Given two leaves of distinct BVH trees containing two feature areas, an initial solution guess is necessary to initialize the iterative method. Because the BVH leaves conform to the properties described in Section 2.4.1, they contain feature areas defined on rectangular domains which are *almost flat*. Therefore, it is likely in practice that the closest points between polyhedral approximations of the surface on this rectangular domain are not too far from the actual closest points. The following method detailed by Algorithm 12 and illustrated by Figure 4.6a computes such an initial guess for two surface areas defined on the rectangular domains delimited by the BVH leaves. This procedure follows three steps which are easily adaptable to configurations, as shown in Figure 4.6b, where one or both feature areas are curves:

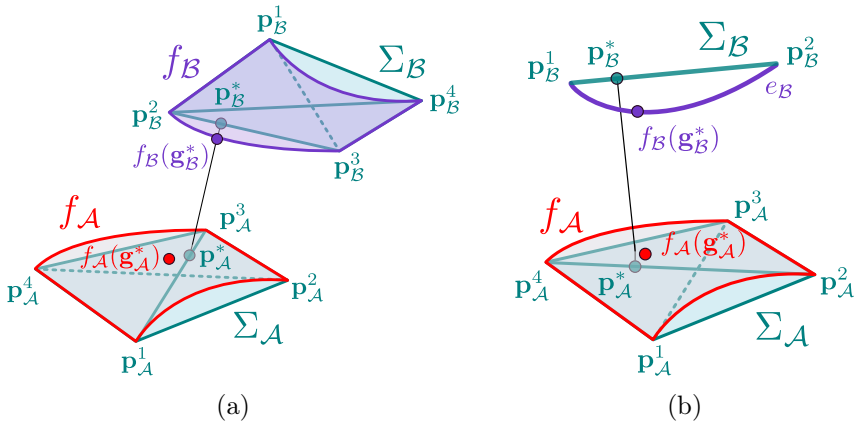


Figure 4.6: Use of polyhedral approximation to obtain a guess for the location of LMD footpoints. (a) Between two surfaces. (b) Between one curve and one surface.

1. Retrieve the parametric surfaces $f_{\mathcal{A}}, f_{\mathcal{B}}$ wrapped by both BVH nodes (lines 2 and 3). For both surfaces, retrieve the parametric coordinates (noted $\mathbf{u}^i_{\mathcal{A}}$ and $\mathbf{u}^i_{\mathcal{B}}$ for $i \in [0, 4]$) of all four corners of their rectangular sub-domain and compute the corresponding corner points $\mathbf{p}^i_{\mathcal{A}}, \mathbf{p}^i_{\mathcal{B}}$ (line 4 to 8). They are located on the surfaces. All four non-degenerate triangles constructed from those corner points form polyhedra $\Sigma_{\mathcal{A}}$ and Σ_B (lines 10 and 11) which can be seen as tetrahedral volumes;

2. Linear mappings $\mathbf{g}_{\mathcal{A}} : \partial\Sigma_{\mathcal{A}} \to \mathbb{R}^2$ and $\mathbf{g}_{\mathcal{B}} : \partial\Sigma_{\mathcal{B}} \to \mathbb{R}^2$ between points on the boundary of each tetrahedron and the parametric plane are defined such that $\mathbf{g}_{\mathcal{A}}(\mathbf{p}^i_{\mathcal{A}}) = \mathbf{u}^i_{\mathcal{A}}$ and $\mathbf{g}_{\mathcal{B}}(\mathbf{p}^i_{\mathcal{B}}) = \mathbf{u}^i_{\mathcal{B}}$. Then note that each point on the boundary of a tetrahedron can be obtained as a linear combination of its vertices:

$$\mathbf{p}^*_{\mathcal{A}} = \sum_{i=1}^{4} \alpha_i \mathbf{p}^i_{\mathcal{A}}, \qquad \mathbf{p}^*_{\mathcal{B}} = \sum_{i=1}^{4} \beta_i \mathbf{p}^i_{\mathcal{B}}. \tag{4.1}$$

   The coefficient α_i, β_i are called *barycentric coordinates* of $\mathbf{p}^*_{\mathcal{A}}, \mathbf{p}^*_{\mathcal{B}}$. They are positive and such that $\sum_{i=1}^{4} \alpha_i = 1$ and $\sum_{i=1}^{4} \beta_i = 1$, and at least one α_i and one β_i is zero. Thus, given a point $\mathbf{p}^*_{\mathcal{A}}$ (resp. $\mathbf{p}^*_{\mathcal{B}}$) on $\Sigma_{\mathcal{A}}$ (resp. $\Sigma_{\mathcal{B}}$), the mappings $\mathbf{g}_{\mathcal{A}}$ and $\mathbf{g}_{\mathcal{B}}$ can be written as:

$$\begin{aligned} \mathbf{g}_{\mathcal{A}}(\mathbf{p}^*_{\mathcal{A}}) &= \mathbf{g}_{\mathcal{A}}\left(\sum_{i=1}^{4} \alpha_i \mathbf{p}^i_{\mathcal{A}}\right) = \sum_{i=1}^{4} \alpha_i \mathbf{g}_{\mathcal{A}}\left(\mathbf{p}^i_{\mathcal{A}}\right) = \sum_{i=1}^{4} \alpha_i \mathbf{u}^i_{\mathcal{A}}, \\ \mathbf{g}_{\mathcal{B}}(\mathbf{p}^*_{\mathcal{B}}) &= \mathbf{g}_{\mathcal{B}}\left(\sum_{i=1}^{4} \alpha_i \mathbf{p}^i_{\mathcal{B}}\right) = \sum_{i=1}^{4} \alpha_i \mathbf{g}_{\mathcal{B}}\left(\mathbf{p}^i_{\mathcal{B}}\right) = \sum_{i=1}^{4} \alpha_i \mathbf{u}^i_{\mathcal{B}}; \end{aligned} \tag{4.2}$$

3. The closest points between the tetrahedra are computed using the GJK algorithm [49], which also produces their barycentric coordinates, are by-products (line 12). Those closest points are input to the mapping Equation (4.2) to obtain the parameters $\mathbf{g}^*_{\mathcal{A}}, \mathbf{g}^*_{\mathcal{B}}$ of the solution guesses on both surfaces (lines 13 and 14).

Algorithm 12 Estimates a good guess between two BVH leaves $l_{\mathcal{A}}, l_{\mathcal{B}}$ containing surface areas. Refer to Listing 2.4 for the exact meaning of each field of $l_{\mathcal{A}}$ and $l_{\mathcal{B}}$. The GJK algorithm is presented here as a function parametrized by two tetrahedra and returning the barycentric coordinates their closest points.

1: **function** FindGuess($l_{\mathcal{A}}$, $l_{\mathcal{B}}$)
2: $f_{\mathcal{A}} \leftarrow l_{\mathcal{A}}.supermax\_feature[l_{\mathcal{A}}.feature\_id]$
3: $f_{\mathcal{B}} \leftarrow l_{\mathcal{B}}.supermax\_feature[l_{\mathcal{B}}.feature\_id]$
4: $\mathbf{u}^1_{\mathcal{A}} \leftarrow (l_{\mathcal{A}}.u_{min}, l_{\mathcal{A}}.v_{min})$
5: $\mathbf{u}^2_{\mathcal{A}} \leftarrow (l_{\mathcal{A}}.u_{min}, l_{\mathcal{A}}.v_{max})$
6: $\mathbf{u}^3_{\mathcal{A}} \leftarrow (l_{\mathcal{A}}.u_{max}, l_{\mathcal{A}}.v_{min})$
7: $\mathbf{u}^4_{\mathcal{A}} \leftarrow (l_{\mathcal{A}}.u_{max}, l_{\mathcal{A}}.v_{max})$
8: $\mathbf{p}^i_{\mathcal{A}} \leftarrow f_{\mathcal{A}}(\mathbf{u}^i_{\mathcal{A}})$ for $i \in [1, 4]$
9: Compute $\mathbf{u}^i_{\mathcal{B}}, \mathbf{p}^i_{\mathcal{B}}$ similarly using $l_{\mathcal{B}}$.
10: $\Sigma_{\mathcal{A}} \leftarrow$ tetrahedron with vertices $\{\mathbf{p}^1_{\mathcal{A}}, \mathbf{p}^2_{\mathcal{A}}, \mathbf{p}^3_{\mathcal{A}}, \mathbf{p}^4_{\mathcal{A}}\}$
11: $\Sigma_{\mathcal{B}} \leftarrow$ tetrahedron with vertices $\{\mathbf{p}^1_{\mathcal{B}}, \mathbf{p}^2_{\mathcal{B}}, \mathbf{p}^3_{\mathcal{B}}, \mathbf{p}^4_{\mathcal{B}}\}$
12: $(\alpha_i, \beta_i) \leftarrow \text{GJK}(\Sigma_{\mathcal{A}}, \Sigma_{\mathcal{B}})$
13: $\mathbf{g}^*_{\mathcal{A}} \leftarrow \sum_{i=0}^{4} \alpha_i \mathbf{u}^i_{\mathcal{A}}$
14: $\mathbf{g}^*_{\mathcal{B}} \leftarrow \sum_{i=0}^{4} \beta_i \mathbf{u}^i_{\mathcal{B}}$
15: **return** $(\mathbf{g}^*_{\mathcal{A}}, \mathbf{g}^*_{\mathcal{B}})$
16: **end function**

The mapping given by Equation (4.2) is easily generalizable to polyhedra with an arbitrary number of points. Thus, better guesses might be achievable when considering more vertices located on each surface, and executing the GJK on their convex hulls. This would however, increase the computation time.

Now, if the polyhedral approximations of $\mathcal{A}$ and $\mathcal{B}$ intersect, this method cannot be used since the GJK algorithm will fail. As a brute-force alternative, the computation of the closest points takes place between all edge-edge and vertex-face pairs explicitly and the best pair is selected. Nevertheless, this second case barely occurs when $\mathcal{A}$ and $\mathcal{B}$ are kept separated with a small distance from each other, especially if it results from the implicit dilation presented in Section 1.3.1.2.

Even though they have a $O(1)$ time complexity, the GJK or its brute-force alternative involve a significant amount of arithmetic operations. Thus, Section 4.5.3 shows how the exploitation of the time coherence enables an update of the guess points at a much lower price during subsequent time steps.

4.3.2 Handling some conformal contact configurations with sampling

In Section 1.3.1, conformal contact configurations appear as non-punctual connected sets of non-strict local minima of the distance function between $\mathcal{A}$ and $\mathcal{B}$. Therefore, using the distance-based geometric contact model, *conformal contacts* do not necessarily correspond to effective contacts as shown in Figure 4.7. For example, a convex sphere concentric with (but not intersecting) a concave sphere are in a conformal contact configuration even if their radii are not equal (see Figure 4.7b).

As a classification of conformal contact configurations, two categories are proposed, based on the dimensionality of the contact area:

- 2-dimensional conformal contacts (see Section 4.3.2.1) are contact surfaces and may arise only between pairs of surfaces;

(a) (b)

Figure 4.7: Two configurations considered 'conformal' even if there is no effective contact between $\mathcal{A}$ and $\mathcal{B}$. The conformal contact areas, i.e., the footpoints of the non-punctual non-strict local minima of the squared ditance function, are highlighted in blue.

- 1-dimensional conformal contacts (see Section 4.3.2.2) are contact curves and may arise between two surfaces, a surface and a curve, or two curves.

Conformal configurations require special care because they cause LMD computation methods presented in Section 4.3.1 to fail since the local minimizers are no longer isolated points. The following sections propose some preliminary work that helps handling some simple cases occurring either between two canonical surfaces or a canonical surface and a curve of any type that is located at the intersection of two canonical surfaces. Note that it must be kept in mind that the modeling of the dynamics of conformal contacts as part of a time-stepping integration scheme is still an open problem since it triggers issues regarding both the geometrical and the mathematical well-posedness of the unilateral contact problem. Therefore, while our sampling-based approach can help reduce penetrations in conformal contact configurations by allowing the definition of a finite set of non-penetration constraints, this is by no mean a robust solution to overcome the limitations of existing geometric contact models exploited by time-stepping integration schemes.

Detection of conformal contact configurations other than the ones presented in the following sections are left to future works.

4.3.2.1 Conformal contacts of dimension two

As far as canonical surfaces are concerned, it is clear that a 2-dimensional conformal contact can occur between two surfaces if and only if they are of the same type with opposite normals. In that case, straightforward conditions on their intrinsic geometric parameters (center, axis, radius, etc.) can be found to detect a conformal contact configuration. Let $F^1_{\mathcal{A}}$ be a face modeled by a convex canonical surface and $F^1_{\mathcal{B}}$ modeled by a concave canonical surface (refer to Figure 1.26 for the typology of canonical surfaces). The following conditions must hold depending on the surfaces types:

- Planes must have collinear and opposite normals $\mathbf{n}_{\mathcal{A}}$ and $\mathbf{n}_{\mathcal{B}}$, and two points $\mathbf{p}_{\mathcal{A}} \in F^1_{\mathcal{A}}$ and $\mathbf{p}_{\mathcal{B}} \in F^2_{\mathcal{B}}$ with $\mathbf{p}_{\mathcal{A}} \neq \mathbf{p}_{\mathcal{B}}$ chosen arbitrarily on each planes, must be such that $\langle \mathbf{p}_{\mathcal{B}} - \mathbf{p}_{\mathcal{A}}, \mathbf{n}_{\mathcal{A}} \rangle \geq 0$. Indeed, $\langle \mathbf{p}_{\mathcal{B}} - \mathbf{p}_{\mathcal{A}}, \mathbf{n}_{\mathcal{A}} \rangle < 0$ could correspond to configurations where the Equation (1.23) characterizing critical points of the squared distance functions is violated for any pair of points on $F^1_{\mathcal{A}}$ and $F^2_{\mathcal{B}}$;

- Spheres must have the same center and the radius of $F^1_{\mathcal{A}}$ must be smaller than the radius of $F^1_{\mathcal{B}}$;

- Cylinders must be coaxial and the radius of $F^1_{\mathcal{A}}$ must be smaller than the radius of $F^1_{\mathcal{B}}$;

- Cones must have the same apex angles and be coaxial. Moreover, it is necessary that $\langle \mathbf{a}_\mathcal{A}, \mathbf{a}_\mathcal{B} \rangle > 0$, and $\langle \mathbf{p}_\mathcal{B} - \mathbf{p}_\mathcal{A}, \mathbf{a}_\mathcal{A} \rangle \leq 0$ where $\mathbf{a}_\mathcal{A}$, $\mathbf{a}_2$ are their axes, and $\mathbf{p}_\mathcal{A}$, $\mathbf{p}_\mathcal{B}$ their apexes;

- Tori must have the same axis, center, and average radius. The minor radius of $F^1_\mathcal{A}$ must be smaller than the minor radius of $F^1_\mathcal{B}$.

Detecting conformal contacts between two non-canonical surfaces is much more involved and not performed as part of the presented framework since they are very unlikely to happen, except at a discrete set of feasible configurations of the system *MS*. In practice, industrial models are generally designed such that conformal contacts are usually functional contacts and thus, often occur at canonical faces [118], which have well-known curvature distributions that can be more easily and accurately manufactured.

Once detected, the contact area on $F^1_\mathcal{A}$ is sampled with a finite set of points. Those points are then projected on $F^1_\mathcal{B}$ in order to obtain pairs of closest points that are used to define non-penetration constraints with the distance-based geometric model presented in Section 1.3.1.

4.3.2.2 Conformal contacts of dimension one

On the one hand, let us observe that conformal contacts between two curves are extremely rare and transitory. Thus, identifying them is generally of no interest for dynamics simulations. Conformal contacts between a curve and a canonical face, on the other hand, are more frequent and may arise in scenarios as common as a cylindrical bottle resting upright on a flat table. Only canonical surfaces are considered here. Other types of surfaces like developable surfaces could be of interest but left to future works.

Let $F^1_\mathcal{B}$ be a canonical face and $E^{1,2}_\mathcal{A}$ an edge. On the one hand, Figure 4.8 shows that it is sufficient to verify that $F^1_\mathcal{B}$ is in 2-dimensional conformal contact with either $F^1_\mathcal{A}$ or $F^2_\mathcal{A}$. If so, then any point of the (untrimmed) surface $F^1_\mathcal{B}$ can be paired with a point of the (untrimmed) surface $F^1_\mathcal{A}$ to be LMD footpoints between $\mathcal{A}$ and $\mathcal{B}$. Moreover, because $E^{1,2}_\mathcal{A} \subset F^1_\mathcal{A}$, all points of $E^{1,2}_\mathcal{A}$ can be paired with a point from $F^1_\mathcal{B}$ to form a local minimizer as well. Therefore, $E^{1,2}_\mathcal{A}$ and $F^1_\mathcal{B}$ are in 1-dimensional conformal contact. With a similar reasoning, it can be deduced that $E^{1,2}_\mathcal{B}$ and $F^1_\mathcal{A}$ are in 1-dimensional conformal contact too.

On the other hand, in the other, less common case, where $F^1_\mathcal{B}$ is not in conformal contact with neither $F^1_\mathcal{A}$ nor $F^2_\mathcal{A}$, then the derivation of criteria for the existence of a conformal contact between $E^{1,2}_\mathcal{A}$ and $F^1_\mathcal{B}$ with an arbitrary relative position is nontrivial and left for future works.

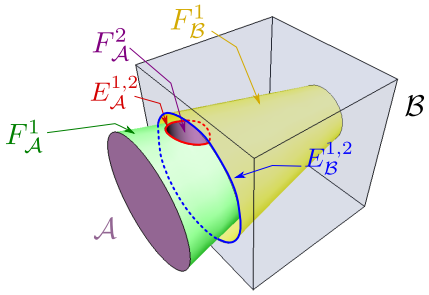


Figure 4.8: Example of configuration generating curve-surface conformal contacts. In this configuration, the cone $\mathcal{A}$ has a cylindrical hole bounded by $F^2_\mathcal{A}$ while the cube $\mathcal{B}$ has a conical hole bounded by $F^1_\mathcal{B}$.

Furthermore, it can be speculated that in order to keep the relative curvature constant along the whole contact curve, 1-dimensional conformal contacts between two non-intersecting canonical surfaces may only occur along their lines of curvature. The identification of those lines

of curvature, depending on the shapes orientation and types yield lines or circles that can be determined by studying each pair of surfaces individually.

Whenever a 1-dimensional conformal contact configuration is detected for the features associated to two leaves of the BVHs of $\mathcal{A}$ and $\mathcal{B}$, the contact curve is discretized using a user-defined number of uniformly distributed points. If the conformal contact involves two surfaces, the corresponding curves follow lines of curvature (which are either line segments or circles) from both surfaces. If it involves a face and a curve, namely $F^1_{\mathcal{B}}$ and $E^{1,2}_{\mathcal{A}}$, then samples are computed on $E^{1,2}_{\mathcal{A}}$ and projected on $F^1_{\mathcal{B}}$. Because $F^1_{\mathcal{B}}$ is assumed to be represented as a canonical surface, this projection is always straightforward and does not require an iterative root finder. Indeed, the samples themselves may be pre-computed offline and updated at run-time to take into account the relative displacement of $\mathcal{A}$ with respect to $\mathcal{B}$.

As a final note regarding conformal contact configurations, we stress the fact that the presented sampling-based approach can help dealing with penetration that may be induced by conformal contact configurations, but is not a robust solution. Indeed, some conformal contacts may induce configurations where the feasible space $\mathcal{C}$ is no longer manifold since the contact constraints may completely lock one (or multiple) degree of freedom of a solid. For example, a convex sphere travelling in a concave cylinder with the same radius loses two of its translational degrees of freedom as shown in Figure 4.9. The stable simulation of those configurations is still an open problem when using a time-stepping integration scheme and the mathematical well-posedness of the dynamics of smooth constrained multibody systems is at least unclear to us as it fails to satisfy the constraint qualification conditions described in Acary et al. [4].

(a) (b)

Figure 4.9: (a) A ball $\mathcal{A}$ with 6 degrees of freedom and a static tube $\mathcal{B}$. (b) After insertion of $\mathcal{A}$ into $\mathcal{B}$, the ball is only allowed to translate along the concave cylinder's axis. In other words, the number of degrees of freedom of the ball are reduced to 4 (three rotations and one translation).

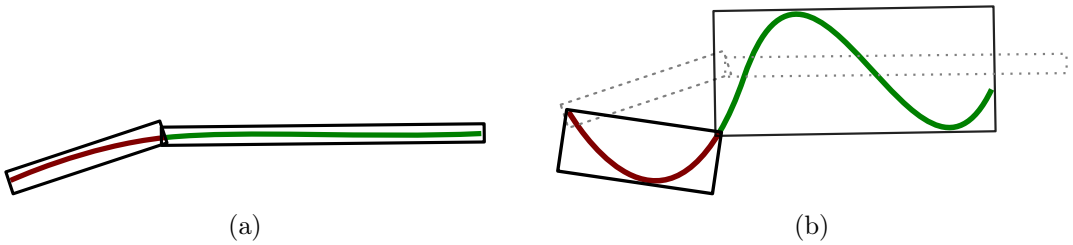
4.3.3 An hybrid approach for deformable Bézier curves

Deformable curves, e.g., used to model deformable beams to be simulated, are handled in a slightly different way than curves defined through BRep features. Because they are deformable, their distribution of curvature may change arbitrarily and significantly at each time step. Thus, following the same approach of BRep features, i.e., subdividing them offline into quasi-flat pieces, is problematic for several reasons:

1. After each deformation, the bounding volumes must be updated so that they continue to wrap properly the underlying curves, as shown in Figure 4.10. Those updates can be expensive on deep BVHs;

2. Increased curvature in some segments of the curve may break the flatness assumption presented in Section 2.4. Thus, it would not be accurate to execute a single Newton method during the step (b) of Figure 4.1 on a pair of such deformed curves since no assumption regarding the uniqueness of the LMD can be formulated;

3. Self-collision must be handled. This is especially complex when simulating knots and windings because they typically involve configurations prone to conformal contact generation.



(a) (b)

Figure 4.10: (a) The initial curve without deformation. The OBBs of its initial subdivision into quasi-flat pieces are shown. (b) After deformation, the initial OBBs (gray, dashed) no longer fit it so they have to be updated. Additionally, the bounded curves are no longer quasi-flat.

A typical approach to tackle those issues is the search for LMDs using dynamic subdivisions, as proposed by Johnson et al. [70]. The algorithm is then similar to those presented in Section 1.5.1 with a recursion that stops as soon as the subdivided pieces meet the flatness condition. One issue is the cost of computing the bounding volumes on-the-fly. One solution for rigid curves is to save the dynamically generated subdivision tree [70] to avoid future re-computations. However, this is not applicable for deformable curves since their bounding volume and geometry have to be updated at each time step.

Instead, the approach presented in this section combines pre-computed and runtime subdivisions as follows:

1. At pre-computation time, subdivide the deformable curves into a user-defined number of pieces at equidistant parameter values (see Figure 4.11a). Cluster the curve bounding volumes into a BVH in accordance with their adjacencies (see Figure 4.11b). This pre-computed tree structure will not be modified after deformation of the curve;

2. After a runtime deformation, update the bounding volumes in the tree structures (see Figure 4.11c). This is a bottom-up procedure: first, the bounding volumes of the leaves (containing one piece of curve each) are recomputed. Bounding volumes of parent nodes are computed such that they bound the bounding volumes of their children;

3. During the contact determination, all the steps of Figure 4.1 are performed though the first one is slightly modified. Those modifications are the subject of the remaining section.

The step (a), shown in Figure 4.1, is modified as follows for the computation of LMDs between two objects $\mathcal{A}$ and $\mathcal{B}$ where at least one, say, $\mathcal{A}$, is a deformable curve and the other one either a deformable curve or a rigid body, i.e., an invariant BRep model. The case where both are BRep was already addressed in Section 4.2. In a first place, the simultaneous traversal described in Section 4.2 is performed using the BVH computed offline but now, the culling tests executed during the descent depends on the types of the objects involved in the LMD computation:

1. If $\mathcal{B}$ is a rigid body, the culling test consists in an OBB-OBB intersection test followed by an orthogonality test between the BRep nodes' normal cones of revolution and the curve nodes' tangent cones;

2. If $\mathcal{B}$ is also a deformable curve then, only the OBB-OBB intersection test is performed.

Whenever a leaf of the BVTT corresponding to this traversal is reached, two cases may arise:

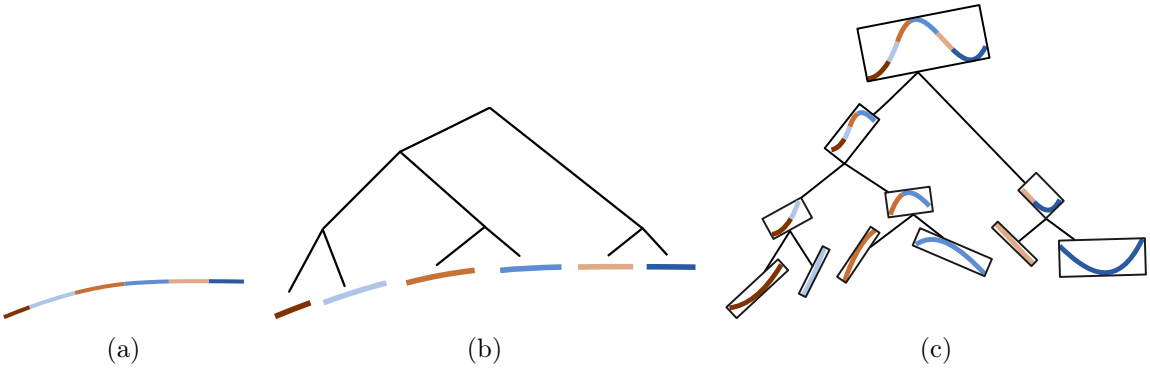


Figure 4.11: (a) the domain of the Bézier curve is split into several (here, 6) equal parts. Each part is shown with one color. (b) The tree structure groups adjacent parts together. (c) After a deformation, the tree structure has to be updated but its topology remains identical. The deformed curve is shown at the root of the tree.

1. If the curve segments contained into the BVTT leaves of the deformable curves validates the flatness condition discussed in Section 2.4.1, then the algorithm can proceed immediately to the step (b) of Figure 4.1 to perform the exact LMD computation;

2. If the curve segment contained into the BVTT leaves of the deformable curve does **not** validate the flatness condition discussed in Section 2.4.1 then, the localization of curve segments that may contain an LMD is not accurate enough to proceed to exact LMD computation. Therefore, the simultaneous BVH traversal carries on with an on-the-fly computation of new BVH nodes for the deformable curve(s) until the computed curve segments are almost flat. This process is described in details in Algorithm 13 for the case where $\mathcal{A}$ and $\mathcal{B}$ are both deformable curves, and in Algorithm 14 for the case where $\mathcal{A}$ is a deformable curve and $\mathcal{B}$ a rigid body.

Both dynamic traversal algorithms start with a bounding volume test in order to check if the curve segment obtained so far still has a chance of containing a LMD. There, the tests performed during the traversal with dynamics subdivisions are exclusively orientation-based. Indeed, between two deformable curve segments, only the orthogonality test presented in Section 3.5 between the solution line cone (see Section 3.5.1.3) and the tangent cones represented as polyhedral cones (see Section 3.5.2) is performed. Between one deformable curve segment and one BRep feature area, the same orthogonality test is performed but between the curve segment normal cone and the BRep feature area normal cone represented as polyhedral cones.

The choice of solely relying on orientation-based bounding volumes for the dynamic subdivisions come from practical experimentations (see Section 5.3) that lead to the following conclusion:

- The traversal executed between the pre-computed BVH already included OBB for their culling tests. Even if a deformation process reduces the tightness of these OBB, they remain useful enough to filter most pairs that are far from each other;

- Tight OBB are too expensive to be computed at runtime for each dynamic subdivision. In addition, tight polyhedral tangent cones, normal cones, and solution-line cones can be computed very efficiently from the control points of the curve derivatives, as shown in Section 3.5.2.2;

- Because the curve segments become flatter during the dynamic subdivision, the polyhedral tangent cones become smaller, making them very discriminatory even if $\mathcal{A}$ and $\mathcal{B}$ are extremely close to each other.

Algorithm 13 Simultaneous BVH traversal with recursive dynamic subdivisions. Case where both inputs are Bézier curves.

function DynamicTraversalCurveCurve($bezier_{\mathcal{A}}, bezier_{\mathcal{B}}$)
  $C_{\mathcal{A}}, C_{\mathcal{B}} \leftarrow$ polyhedral tangent cones of $bezier_{\mathcal{A}}$ and $bezier_{\mathcal{B}}$.
  $S \leftarrow$ solution line cone $S(A, B)$.
  if S is orthogonal to $C_{\mathcal{A}}$ and $C_{\mathcal{B}}$ **then**
    $(bezier^1_{\mathcal{A}}, bezier^2_{\mathcal{A}}) \leftarrow$ split $bezier_{\mathcal{A}}$ at its middle parameter.
    $(bezier^1_{\mathcal{B}}, bezier^2_{\mathcal{B}}) \leftarrow$ split $bezier_{\mathcal{B}}$ at its middle parameter.
    if $bezier_{\mathcal{A}}$ and $bezier_{\mathcal{B}}$ are both almost flat **then**
      Compute exact LMD between $bezier_{\mathcal{A}}$ and $node_{\mathcal{B}}$.
    else if $bezier_{\mathcal{A}}$ is almost flat **then**
      ▷ Recursive calls splitting $bezier_{\mathcal{B}}$ only.
      DynamicTraversalCurveCurve($bezier_{\mathcal{A}}, bezier^1_{\mathcal{B}}$)
      DynamicTraversalCurveCurve($bezier_{\mathcal{A}}, bezier^2_{\mathcal{B}}$)
    else if $bezier_{\mathcal{B}}$ is almost flat **then**
      ▷ Recursive calls splitting $bezier_{\mathcal{A}}$ only.
      DynamicTraversalCurveCurve($bezier^1_{\mathcal{A}}, bezier_{\mathcal{B}}$)
      DynamicTraversalCurveCurve($bezier^2_{\mathcal{A}}, bezier_{\mathcal{B}}$)
    else
      ▷ Recursive calls splitting both curves.
      DynamicTraversalCurveCurve($bezier^1_{\mathcal{A}}, bezier^1_{\mathcal{B}}$)
      DynamicTraversalCurveCurve($bezier^1_{\mathcal{A}}, bezier^2_{\mathcal{B}}$)
      DynamicTraversalCurveCurve($bezier^2_{\mathcal{A}}, bezier^1_{\mathcal{B}}$)
      DynamicTraversalCurveCurve($bezier^2_{\mathcal{A}}, bezier^2_{\mathcal{B}}$)
    end if
  end if
  ▷ Otherwise, no possible LMD between those curve segments.
end function

Algorithm 14 Simultaneous BVH traversal with recursive dynamic subdivisions. Case where both one input is a deformable Bézier curves and the other a BRep feature area identified by its BVH node $node_{\mathcal{B}}$.

function DynamicTraversalCurveBRep($bezier_{\mathcal{A}}, node_{\mathcal{B}}$)
  $C_{\mathcal{A}} \leftarrow$ polyhedral tangent cone of $bezier_{\mathcal{A}}$.
  if $C_{\mathcal{A}}$ is orthogonal to node2.*polyhedralNormalCone* **then**
    if $bezier_{\mathcal{A}}$ is almost flat **then**
      Compute exact LMD between $bezier_{\mathcal{A}}$ and $node_{\mathcal{B}}$.
    else
      $(bezier^1_{\mathcal{A}}, bezier^2_{\mathcal{A}}) \leftarrow$ split $bezier_{\mathcal{A}}$ at its middle parameter.
      ▷ Recursive calls.
      DynamicTraversalCurveBRep($bezier^1_{\mathcal{A}}, node_{\mathcal{B}}$)
      DynamicTraversalCurveBRep($bezier^2_{\mathcal{A}}, node_{\mathcal{B}}$)
    end if
  end if
  ▷ Otherwise, no possible LMD between this curve segment and the BRep feature area.
end function

4.4 Validation of closest points computed onto individual features

Whenever an analytical method exists, the procedures presented in the Section 4.3 computes all the LMDs between two untrimmed canonical surfaces, curves, or point, or, if an iterative minimization method must be used, they compute a LMD between two features trimmed to rectangular sub-domains that passed the culling test described in Section 2.4.3. In both cases, trimming curves potentially lying on a rectangular sub-domain are ignored. Therefore, it is necessary to check if the computed LMD footpoints do not actually lie in a hole.

Moreover, there is no guarantee that the LMD footpoints computed by a method from Section 4.3 between two points, curve segments, or surface areas will actually be LMD footpoints of the corresponding feature areas in $\mathcal{A}$ and $\mathcal{B}$. For example let $\gamma_{\mathcal{A}} : D_{\mathcal{A}} \to \mathbb{R}^3$ and $\gamma_{\mathcal{B}} : D_{\mathcal{B}} \to \mathbb{R}^3$ be two curves segments defined on domains $D_{\mathcal{A}}$ and $D_{\mathcal{B}}$ such that $\forall t \in D_{\mathcal{A}}, \gamma_{\mathcal{A}}(t) \in \mathcal{A}$ and $\forall t \in D_{\mathcal{B}}, \gamma_{\mathcal{B}}(t) \in \mathcal{B}$. To clarify the notations, we note here the sets:

$$\Gamma_{\mathcal{A}} = \{\gamma_{\mathcal{A}}(t) \mid t \in D_{\mathcal{A}}\} \subset \mathcal{A}, \tag{4.3}$$
$$\Gamma_{\mathcal{B}} = \{\gamma_{\mathcal{B}}(t) \mid t \in D_{\mathcal{B}}\} \subset \mathcal{B}. \tag{4.4}$$

The methods from Section 4.3 executed on $\gamma_{\mathcal{A}}$ and $\gamma_{\mathcal{B}}$ will output LMD footpoints $\mathbf{p}_{\mathcal{A}} \in \Gamma_{\mathcal{A}}$ and $\mathbf{p}_{\mathcal{B}} \in \Gamma_{\mathcal{B}}$ such that the characterization of LMDs from Equation (1.23) is satisfied. Note that $\mathbf{p}_{\mathcal{A}}$ and $\mathbf{p}_{\mathcal{B}}$ are seen here as point on $\Gamma_{\mathcal{A}}$ and $\Gamma_{\mathcal{B}}$, therefore:

$$\begin{cases} \mathbf{p}_{\mathcal{B}} - \mathbf{p}_{\mathcal{A}} & \in & T_{\Gamma_{\mathcal{A}}}(\mathbf{p}_{\mathcal{A}})^*, \\ \mathbf{p}_{\mathcal{B}} - \mathbf{p}_{\mathcal{A}} & \in & -T_{\Gamma_{\mathcal{B}}}(\mathbf{p}_{\mathcal{B}})^*. \end{cases} \tag{4.5}$$

Now, since $\mathbf{p}_{\mathcal{A}} \in \Gamma_{\mathcal{A}} \subset \mathcal{A}$ (resp. $\mathbf{p}_{\mathcal{B}} \in \Gamma_{\mathcal{B}} \subset \mathcal{B}$), we have:

$$T_{\mathcal{A}}(\mathbf{p}_{\mathcal{A}})^* \subseteq T_{\Gamma_{\mathcal{A}}}(\mathbf{p}_{\mathcal{A}})^*, \tag{4.6}$$
$$T_{\mathcal{B}}(\mathbf{p}_{\mathcal{B}})^* \subseteq T_{\Gamma_{\mathcal{B}}}(\mathbf{p}_{\mathcal{B}})^*. \tag{4.7}$$

Therefore, additional tests must be done in order to check that:

$$\begin{cases} \mathbf{p}_{\mathcal{B}} - \mathbf{p}_{\mathcal{A}} & \in & T_{\mathcal{A}}(\mathbf{p}_{\mathcal{A}})^*, \\ \mathbf{p}_{\mathcal{B}} - \mathbf{p}_{\mathcal{A}} & \in & -T_{\mathcal{B}}(\mathbf{p}_{\mathcal{B}})^*, \end{cases} \tag{4.8}$$

to ensure $\mathbf{p}_{\mathcal{A}}$ and $\mathbf{p}_{\mathcal{B}}$ are LMD footpoints when they are seen as points on $\mathcal{A}$ and $\mathcal{B}$.

To summarize, some of the LMD footpoints computed between two surfaces, curves, or points in Section 4.4 may:

- Actually lie into a hole delimited by trimming curves. Note that this case is not so frequent since the BVHs already filter out some closest points that would lie into holes. Indeed, the subdivision of each feature into quasi-flat areas presented in Section 2.4.1 took care of removing any rectangular sub-domain that lies entirely into a hole. The Section 4.4.1 details the necessary additional checks to be executed on LMD footpoints found on rectangular subdomains that intersect or contain some trimming curves.

- Not actually be LMD footpoints when seen as points of $\mathcal{A}$ and $\mathcal{B}$. The Section 4.4.2 provides the necessary additional checks. This case is not so frequent either since the Equation (1.23) that provides necessary conditions about the criticality of two points on $\mathcal{A}$ and $\mathcal{B}$ is already partially enforced by the orientation-based bounding volumes and the related culling tests during the BVH traversal, as described in Section 2.4.3. Indeed,

while the computation of normal cones of revolution and polyhedral normal cones do take into account the effective BRep model from an edge and vertex standpoint, they are only conservative bounds for all points on a feature area. Therefore they may contain more elements than the exact tangent cone polars.

4.4.1 Testing the potential LMD against trimming curves

To check whether a point lies into a hole of a face, or not, a CPU version of Schollmeyer et al. [119] is applied. The method is outlined in this section for completeness. First, a set of operations are performed at pre-computation time on the trimming curves of each face treated independently of the others:

- All the 2D trimming curves of the face are split into bi-monotonic subsets. The v coordinates of the splitting points (the red points in Figure 4.12b). Also, the locations of the extremities of each trimming curve (the blue points in Figure 4.12b) are collected and added into an unique array $V = \{v_1, v_2, ..., v_n\}$, assuming there is a total of n such points, altogether;

- Each curve is then split again at all points with v-coordinates equal to those collected on V (see the green points in Figure 4.12b);

- Assuming V is sorted in increasing order, trimming curve segments obtained after those splits are collected into $(n-1)$ sets $V_1, V_2, ..., V_{n-1}$. The set V_i contains the curve segments such that the v coordinate of their two extremities are equal to $\mathbf{v}_i, \mathbf{v}_{i+1}$, respectively;

- The bounding box of each curve segment in each V_i is computed. If any two such boxes intersect then, the corresponding curves are split again such that the resulting curves have non-intersecting bounding boxes;

- Within each set V_i the curves are sorted in *increasing order*, i.e., such that the bounding box of the k-th curve segment lies completely on the left of the $(k+1)$-th curve segment.

An example of this splitting and sorting procedure is illustrated in Figure 4.12b. Once this is performed, each V_i is partitioned into several u-intervals U_k^i that contain at most one bi-monotonic curve segment. Using the even-odd-rule [100] to check if a point lies inside a closed loop, it is possible to identify, at pre-computation time, some intervals that lie completely inside the solid domain, or completely outside. The only intervals where no decision can be made are those containing a trimming curve (see Figure 4.12a).

At run-time, given the parametric coordinates (u, v) of a point $\mathbf{p}$, it is possible to check very efficiently, with the following procedure, if it lies into a hole:

1. Find the V_i that contains the parametric coordinate v with a dichotomic search;

2. Find the interval U_k^i of V_i that contains the parametric coordinate u with a dichotomic search;

3. If U_k^i does not contain any trimming curve segment, then the decision regarding the trimming of $\mathbf{p}$ can be taken without further computations given the previous observations. Indeed, $\mathbf{p}$ is valid if U_k^i has been identified at pre-computation time as lying inside of the solid domain (green in Figure 4.12c), or invalid if U_k^i has been identified at pre-computation time as lying outside on the solid domain (red in Figure 4.12c).

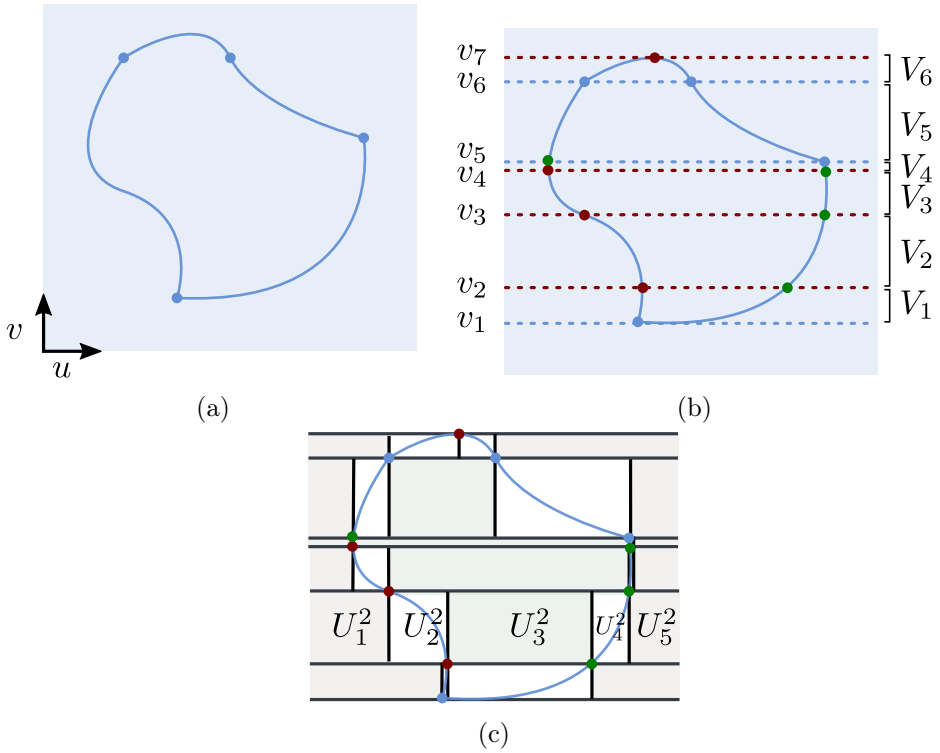


Figure 4.12: (a) Four trimming curves with their extremities highlighted with disks. (b) Each curve is split into bi-monotonic parts and sorted into intervals V_i. (c) The final partitioning of the parametric plane. The partitioning of V_2 into the u-intervals U_k^2 are explicitly named here. All points on the red (resp. green) parts are known to lie outside (resp. inside) of the delimited domain. Each white part contains a trimming curve segment.

4. If U_k^i contains one trimming curve segment ξ, the location of the (u, v) parameters of the point $\mathbf{p}$ with regard to ξ can be determined with a few iterations of a bisection method on the v parametric coordinate axis. Referring to the terminology and discussions in Section 1.4.3 regarding the orientation of trimming curves, the point $\mathbf{p}$ is considered part of the solid domain if it is proven to lies on the left hand side (as defined in Section 1.4.3) of the tangent of the point on ξ closest to (u, v). Otherwise is considered outside of the solid domain. Note that such a bisection is both efficient and accurate since all trimming curve segments are guaranteed to be bi-monotonic.

Further details and experiments can be found in Schollmeyer et al. [119].

4.4.2 Filtering the potential LMD using exact tangent cone polars

A pair of points $\mathbf{p}_\mathcal{A} \in \mathcal{A}$ and $\mathbf{p}_\mathcal{B} \in \mathcal{B}$ computed as described in Section 4.3 have to be evaluated using Equation (1.23) in order to check whether they really are footpoints of an LMD between $\mathcal{A}$ and $\mathcal{B}$, or not. To that matter, $T_\mathcal{A}(\mathbf{p}_\mathcal{A})^*$ and $T_\mathcal{B}(\mathbf{p}_\mathcal{B})^*$ have to be computed. This procedure depends on the feature each point belongs to. The following description addresses the point $\mathbf{p}_\mathcal{A}$ (the same applies for $\mathbf{p}_\mathcal{B}$):

- If $\mathbf{p}_\mathcal{A}$ is a point on a face then its tangent cone polar equals the normal of the corresponding surface at this point;
- If $\mathbf{p}_\mathcal{A}$ is a point on an edge then $T_\mathcal{A}(\mathbf{p}_\mathcal{A})^*$ equals the polyhedral cone generated by the normals to the two faces intersecting at $\mathbf{p}_\mathcal{A}$. An exception occurs when $\mathbf{p}_\mathcal{A}$ is a concave point, as defined in Section 1.4.3, in which case $T_\mathcal{A}(\mathbf{p}_\mathcal{A})^* = \{\mathbf{0}\}$;

- If $\mathbf{p}_\mathcal{A}$ matches a vertex, then computing $T_\mathcal{A}(\mathbf{p}_\mathcal{A})^*$ exactly is more involved. The case where $\mathbf{p}_\mathcal{A}$ lies at the intersection of planar faces has been studied by Merlhiot [90] to address configurations where $\mathcal{A}$ and $\mathcal{B}$ have polyhedral boundaries. In order to extend this work to cases where $\mathbf{p}_\mathcal{A}$ is at the intersection of smooth faces, the first-order approximation of $\partial\mathcal{A}$ at the point $\mathbf{p}_\mathcal{A}$ is taken. Indeed, $\partial\mathcal{A}$ can be assimilated to its piecewise-linear approximation on an arbitrarily small neighborhood of $\mathbf{p}_\mathcal{A}$. Each edge of this polyhedral approximation is collinear to the tangent at $\mathbf{p}_\mathcal{A}$ of one of the curve on $\partial\mathcal{A}$ meeting at $\mathbf{p}_\mathcal{A}$. Moreover, each triangle of this polyhedral approximation is orthogonal to the normal at $\mathbf{p}_\mathcal{A}$ of one of the surface on $\partial\mathcal{A}$ meeting at $\mathbf{p}_\mathcal{A}$.

If the targeted application of the physics simulation justifies the use of quasi-LMD, then the aforementioned cones assigned to edges and vertices have to be enlarged with a user-defined angle α. This enlargement is straightforward for the tangent cone polar of a point along an edge and was shown on Figure 1.17a since it is similar to the planar case. The enlarged the tangent cone polar of a vertex can be pre-computed offline using the dilation procedure described in Section 3.4.

4.5 Exploiting temporal coherence

The concept of *temporal coherence* is essential for re-using some intermediate results from the previous time step to speed up CD at the current one. It has been subjected to extensive researches, as highlighted by the literature [69, 136, 130]. The basic hypothesis is to assume that objects do not move *too much*, i.e., their positions evolve continuously and their velocities and time-step length are such that one position update generates small enough changes on the object positions to make whatever geometric results produced at the time step t_n very likely to be still valid, or at least easy to update, for the time step t_{n+1}. This basic assumption works well in practice and is beneficial to all stages of the CD pipeline:

- To avoid restarting the BVTT traversal from the root using the concept of *front tracking*, see Section 4.5.1;
- To, sometimes, avoid having to perform complex culling tests (see Section 4.5.2);
- To initialize more accurately and more efficiently iterative methods that locate LMD footpoints (see Section 4.5.3).

4.5.1 Front tracking

Front tracking is a common method for reducing the cost of the simultaneous BVH traversal using an explicit construction and storage of the BVH nodes where the traversal ended at the last time-step, t_n. Those stored nodes are then used as starting points for the traversal at the next time step , t_{n+1}. The array storing those nodes is called a *BVTT front*. Front tracking requires three steps:

1. Front nodes status assignment (see Figure 4.13b): one state is assigned to a BVTT node depending on the result of the culling test between the bounding volumes contained by its two underlying BVH nodes. *ToTraverse* is assigned if the culling test fails (new contacts are possible). Otherwise, *ToDelete* is assigned (no contact is possible);
2. Front traversal (see Figure 4.13c): the BVTT traversal presented in Section 4.2.1 is resumed starting with each node marked *ToTraverse*. Then, those nodes are replaced with the list of nodes the traversal stopped on. The procedure is detailed in Algorithm 17;

3. Partial front pruning (see Figure 4.13d): it removes some pairs of BVTT nodes marked as *ToDelete*. Such removal occurs only if the two following conditions are simultaneously satisfied:

   (a) Both BVTT nodes share the same BVTT parent node. This can be checked by the Algorithm 15, which tests if they have one BVH node is common and if their other BVH nodes share the same parent;

   (b) The culling test on their common BVTT parent node succeeds. Indeed, if the parent culling test fails it is extremely likely that the next time step will mark it as *ToTraverse*, meaning the next front may include its children again. So, it would be wasteful to perform such an unnecessary removal.

   If pruning occurs, one of the node being deleted is replaced by their common parent, while the other is marked *Deleted*. This is an efficient O(1) in-place modification adequate for the parallelization described in Section 4.6. Because the BVTT tree is never completely constructed, this BVTT parent node is not readily available. It can be recovered easily by looking at the parents of the BVH nodes contained by this BVTT nodes. Indeed, the BVTT parent node is the node containing the common parents (from the BVH trees) of its two underlying BVH nodes as described by Algorithm 16.

Let us observe that the third step is called *partial* front pruning because it makes the new front one level closer to the root of the BVTT, at most, than the previous one. A complete front pruning would have no such limit because it would basically apply this pruning process recursively and may even reduce the BVTT front to the root in extreme cases where all chances of LMD are lost. Such a variant is not adopted here because it interferes badly with the parallelization introduced in Section 4.6.

Algorithm 15 Test whether two BVTT nodes $n^1_{\mathcal{A}}$ and $n^2_{\mathcal{A}}$ share a common parent.

```
function HaveCommonBVTTParent(n¹_A, n²_A)
    if n¹_A.bvhNode1 = n²_A.bvhNode1 and n¹_A.bvhNode2.parent = n²_A.bvhNode2.parent then
        ▷ The second BVH node was traversed.
        return True
    else if n¹_A.bvhNode2 = n²_A.bvhNode2 and n¹_A.bvhNode1.parent = n²_A.bvhNode1.parent then
        ▷ The first BVH node was traversed.
        return True
    else
        return False
    end if
end function
```

The very first BVTT front is initialized with the root of the BVTT, which contains the roots of the two BVH being traversed. Pruning requires two neighboring BVTT nodes to be performed, so it cannot occur at its root.

The memory layout of the front is crucial for performance issues. Indeed, its nodes should be ordered in the same way as they would be encountered into a depth-first traversal of the BVTT, i.e., two nodes sharing the same parent must be adjacent in the BVTT front. This simplifies the *pruning* procedure described in Section 4.5.1 since it becomes straightforward to determine when two BVTT nodes have a common BVTT parent node. In this chapter, a BVTT front represented that way is said to be *sorted*. For example, the BVTT front shown on Figure 4.13a should be represented with the contiguous array:

$$[(A_2, B_2)(A_2, B_3)(A_3, B_2)(A_4, B_3)(A_5, B_3)].$$

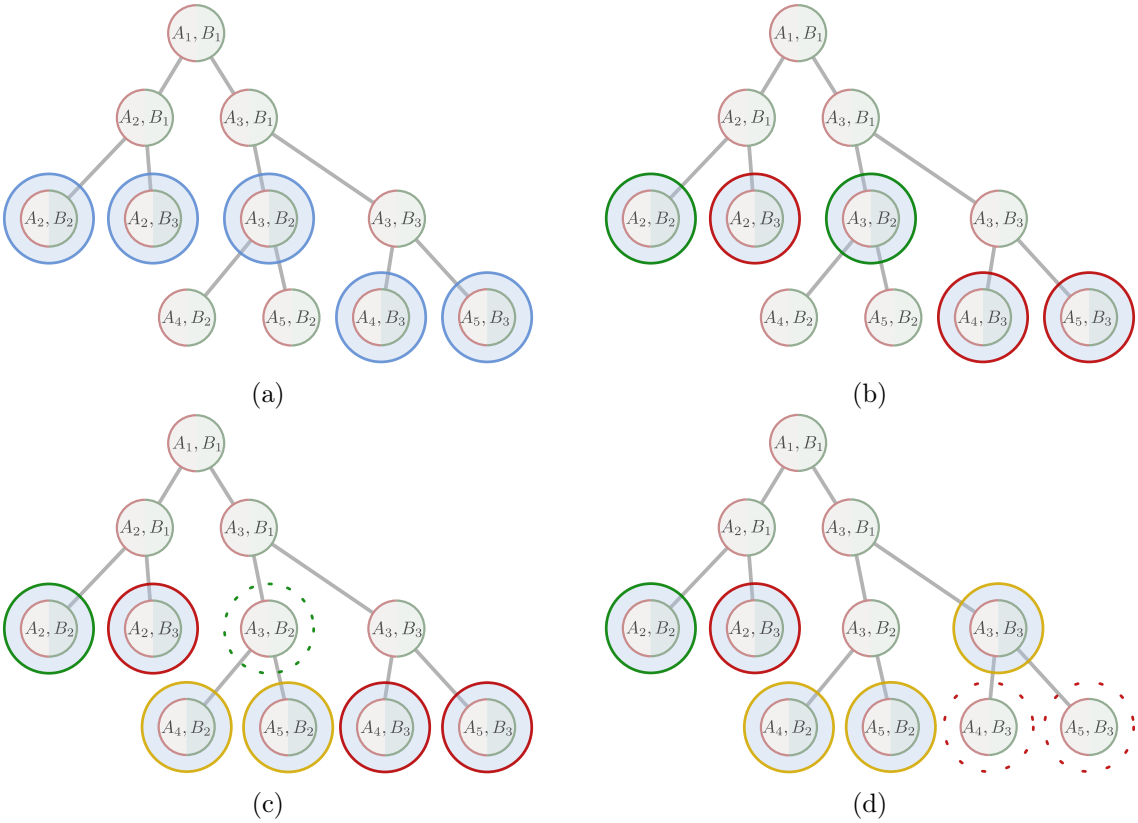


Figure 4.13: (a) A BVTT front. (b) Front nodes status assignment: BVTT front nodes are marked as *ToDelete* (red) or *ToTraverse* (green). (c) Front traversal: the node (A_2, B_2) is already a leaf while the node (A_3, B_2) is replaced by the result of the traversal (A_4, B_2) and (A_5, B_2). (d) Front pruning: the node (A_2, B_3) does not have any neighbor marked *ToDelete*. The nodes (A_4, B_3) and (A_5, B_3) are both marked *ToDelete* and share the same parent. They are thus replaced by (A_3, B_3).

After status assignment/traversal/pruning, the new front must retain this depth-first layout. For example, in Figure 4.13d, it should be represented with the contiguous array:

$$[(A_2, B_2)(A_2, B_3)(A_4, B_2)(A_5, B_2)(A_3, B_3)].$$

4.5.2 Temporal coherence for bounding volumes

Each bounding volume has its own intersection test that can benefit from temporal coherence:

OBB: As detailed in Section 2.4, the OBB culling test is an intersection test. When two OBB do not intersect, the Separating Axis Theorem [52] states that there exists one plane, at least, delimiting two half-spaces, each one containing one of the two OBBs. The normal of this plane is called a *separating axis*. Thus, if one such axis is found at one time-step, it is saved as it is likely to remain a separating axis during the next few time steps as long as the objects do not move too much (see Figure 4.14). At the next time-step, it is tested first, before trying to find another separating axis for the intersection test.

Normal cones of revolution: The intersection test between two cones of revolution computes the angle between both cone axes and compares them with the cones apex-angles. One way to benefit from coherence when the cones do not intersect would be compute the angle of the relative rotation movement between the objects $\mathcal{A}$ and $\mathcal{B}$ and track if it would exceed the minimal angle required to make the two normal cones of revolution contain antipodal directions.

Algorithm 16 Computes the common parent of two BVTT nodes $n^1_{\mathcal{A}}$ and $n^2_{\mathcal{A}}$. This assumes both nodes are already known to have a common parent as tested by the procedure HaveCommonBVTTParent. The BVTTNode procedure constructs a new BVTT nodes from two BVH nodes as show in Listing 4.1.

```
function BVTTNodeParent(n¹_A, n²_A)
    prerequisite HaveCommonBVTTParent(n¹_A, n²_A)
    if n¹_A.bvhNode1 = n²_A.bvhNode1 then
        ▷ The second BVH node was traversed.
        return BVTTNode(n¹_A.bvhNode1, n¹_A.bvhNode2.parent)
    else
        ▷ The first BVH node was traversed.
        return BVTTNode(n¹_A.bvhNode1.parent, n¹_A.bvhNode2)
    end if
end function
```

Algorithm 17 BVTT front node traversal. The TraverseBVTTFrontNode subroutine is similar to Algorithm 9, except that the front nodes are collected whenever the recursion stops. Difference are highlighted in red.

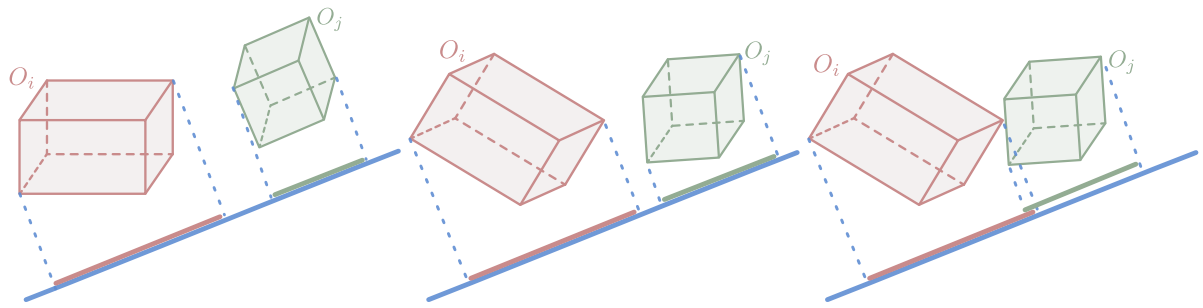
```
function TraverseBVTTFrontNode(node, newFront, newContacts)
    if node.status = ToTraverse then
        if CullingTest(node.bvhNode1, node.bvhNode2) then
            ▷ The + operator design array concatenation.
            Append node to newFront
        else
            children ← BVTTNodeChildren(node)
            if children = NULL then
                Append node to newFront
                Add to newContacts the closest points on node.bvhNode1 and node.bvhNode2
            else
                ▷ Recursive calls on the children.
                TraverseBVTTAndCollectFront(children[0], newFront, newContacts)
                TraverseBVTTAndCollectFront(children[1], newFront, newContacts)
            end if
        end if
    else
        ▷ The node is marked ToDelete, so no need to traverse it.
        Append node to newFront
    end if
end function


function TraverseBVTTFront(front, newFront, newContacts)
    newFront ← Empty array.
    newContacts ← Empty array.
    for all node ∈ front do
        TraverseBVTTFrontNode(node, newFront, newContacts)
    end for
end function
```

However, estimating this angle is costly. Thus, in practice, no coherence is implemented for normal cones of revolution, which already have a very cheap culling test.

Algorithm 18 Procedure to replace pairs of BVTT nodes $n^1_{\mathcal{A}}, n^2_{\mathcal{A}}$ marked *ToDelete* by their common parent.

```
function PruneFront(front)
    for all i ∈ 0.. front.size() - 1 do
        n¹_A ← front[i];
        n²_A ← front[i + 1];
        if n¹_A.status = n²_A.status = ToDelete then
            if n¹_A.bvhNode1 = n²_A.bvhNode1 then
                if n¹_A.bvhNode2.parent = n²_A.bvhNode2.parent then
                    ▷ Replace n¹_A by their common parent.
                    n¹_A.bvhNode2 = n¹_A.bvhNode2.parent
                    n²_A.status ← Deleted
                end if
            else if n¹_A.bvhNode2 = n²_A.bvhNode2 then
                ▷ Do the same but reverting the roles of n¹_A and n²_A.
                if n¹_A.bvhNode1.parent = n²_A.bvhNode1.parent then
                    ▷ Replace n¹_A by their common parent.
                    n¹_A.bvhNode2 = n¹_A.bvhNode2.parent
                    n²_A.status ← Deleted
                end if
            end if
        end if
    end for
end function
```



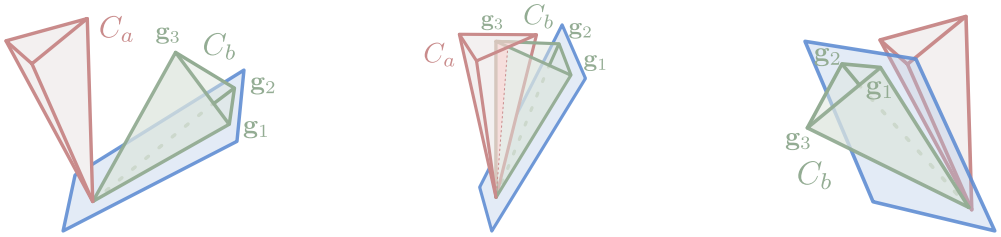
(a) Time step t_i (b) Time step t_{i+1} (c) Time step t_{i+2}

Figure 4.14: (a) Two OBBs O_i, O_j at the time step t_i and one of their separating axis in blue on which their projection do not overlap. (b) After small rotations and translations, the same axis is still a separating axis at the time step t_{i+1} so a complete intersection test will not be necessary. (c) After further movements before the time-step t_{i+2} the axis is no longer a separating axis: an intersection test is still needed.

Polyhedral normal cone: As shown by Proposition 1, if Algorithm 4 finds a plane containing two generators of the Minkowski difference of two polyhedral cones that is also one of its supporting plane then, they do not contain any antipodal direction. If the related feature areas do not rotate much, a plane containing the same (rotated) generators is extremely likely to still be a supporting plane of the Minkowski difference of the new polyhedral normal cones (See Figure 4.15). Reusing previously computed support plane dramatically improves the intersection test because, if it is still a support plane, the original $O(m^3)$ intersection test does not have to be executed.

4.5.3 Temporal coherence for LMD computation

Iterative LMD computation can benefit from temporal coherence when the relative positions of two rigid bodies $\mathcal{A}$ and $\mathcal{B}$ do not change too much then, the location of their closest points will do so. Thus, the closest points found during the time-step t_n could very well be re-used as the starting point of the iterative method at time-step t_{n+1}. This, in practice, provides good



(a) Time step t_i (b) Time step t_{i+1} (c) Time step t_{i+2}

Figure 4.15: (a) Two polyhedral cones at the time step t_i and a support plane of $C_a \oplus C_b$ shown in blue. Here, this plane passes through two of the generators $\mathbf{g}_1, \mathbf{g}_2$ of C_b. (b) Small rotations occurred before the time-step t_{i+1} but the plane passing through the same two generators $\mathbf{g}_1, \mathbf{g}_2$ is still a support plane of $C_a \oplus C_b$. No further polyhedral cone intersection test is necessary. (c) Further rotations occurred before the time-step t_{i+2}, making the plane passing through $\mathbf{g}_1$ and $\mathbf{g}_2$ no longer a support plane: the original $O(n^3)$ intersection test still needs to be executed.

convergence but can be further improved by exploiting contact kinematics [93, 139] to estimate how the closest points moved. Indeed, contact kinematics provides the relationship between the velocities $\dot{\mathbf{q}}_\mathcal{A}, \dot{\mathbf{q}}_\mathcal{B}$ of the closest points parameters and the relative velocities $\omega, \mathbf{v}$ of $\mathcal{A}$ and $\mathcal{B}$ expressed in a local coordinate system attached to those closest points. For example, the following equations were found by Visser et al. [139] for closest points lying on a pair of surfaces corresponding to a face of $\mathcal{A}$ and of $\mathcal{B}$:

$$\begin{aligned}
\dot{q}_\mathcal{A} &= M_\mathcal{A}^{-1} R (K_\mathcal{B}(dRK_\mathcal{A}R + I) + RK_\mathcal{A}R)^{-1} \left(\begin{bmatrix} -\omega_y \\ \omega_x \end{bmatrix} + K_\mathcal{B} \begin{bmatrix} \mathbf{v}_x \\ \mathbf{v}_y \end{bmatrix} \right), \\
\dot{q}_\mathcal{B} &= M_\mathcal{B}^{-1} ((dRK_\mathcal{A}R + I)K_\mathcal{B} + RK_\mathcal{A}R)^{-1} \left((dRK_\mathcal{A}R + I) \begin{bmatrix} -\omega_y \\ \omega_x \end{bmatrix} + RK_\mathcal{A}R \begin{bmatrix} \mathbf{v}_x \\ \mathbf{v}_y \end{bmatrix} \right), \\
\dot{d} &= \mathbf{v}_z.
\end{aligned} \tag{4.9}$$

The terms $M_{\mathcal{A}/\mathcal{B}}, K_{\mathcal{A}/\mathcal{B}}$, and R are obtained from the fundamental forms of the surfaces and are detailed in Visser et al. [139]. The quantity d designates the LMD and $\dot{d}$ its evolution with respect to time.

To summarize, exploiting temporal coherence to find the new position at time-step t_{n+1} of footpoints found at the time-step t_n, amounts to:

1. Integrate the contact kinematic differential equations (Equation (4.9)) taking as starting points the footpoints at t_n. The chosen integration scheme can remain quite basic because the real change in positions is assumed small. For example, an explicit Euler method is fast and accurate enough in practice;

2. Use the result of this integration as the solution guess for the iterative methods to find the actual new closest points;

3. If the iterative method converged to new footpoints, save their parameters so that they can be input to the contact kinematic equations at t_{n+2}.

Using this method, the GJK-based or brute-force solution guess estimation methods presented in Section 4.3.1 are needed only to find new closest points or before each execution of the iterative method that do not find any solution.

However, the temporal coherence assumption ends up being wrong in some cases, and closest points move significantly, or even disappear. In those cases, integrating contact kinematics with a

simple explicit integration scheme can end up generating new solution guesses that are extremely far from the solution at time t_{n+1}, preventing new closest points from being found on the original pair of feature areas because the convergence of the iterative method can no longer be obtained. To avoid this issue, whenever the integration of contact kinematic equations generates solution guesses that lie outside of the rectangular domains of the pair of feature area considered, the GJK-based guess estimation algorithm presented in Section 4.3.1 is used instead.

4.6 Parallelization of the CD pipeline

The CD pipeline presented in Section 4.2 can be slightly modified to take advantage of multi-core microprocessor architectures. The main idea is to split the pipeline into several components that can themselves be parallelized easily. Such an approach fits well with the front tracking strategy since each element of the front can be handled independently from the others. This approach is very similar to other front-tracking schemes based on parallel BVH traversal for triangle meshes like Tang et al. [130], except that here, it is necessary to handle a few inter-thread communications to avoid redundant computations over supermaximal features. The corresponding process is summarized in Figure 4.16.

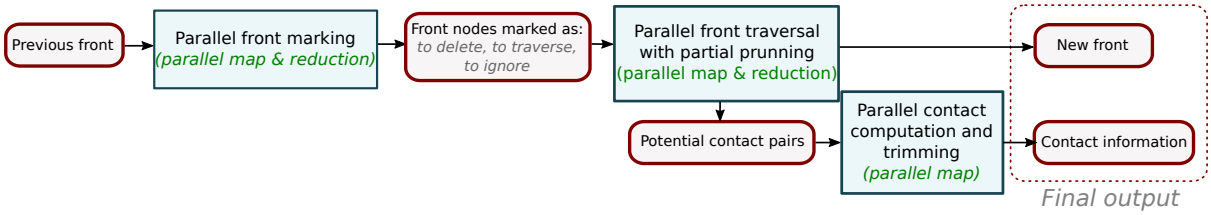


Figure 4.16: Parallel collision detection pipeline with front-tracking. Data are shown in red and algorithmic treatments in blue.

The following sections provide details about each stage. Parallelization has been implemented using the Intel Threading Building Blocks [1] (TBB) to benefit from efficient work balancing among threads. In particular, TBB is task-based, i.e., the user defines a set of *tasks* to be executed concurrently and TBB automatically distributes those tasks between several threads.

4.6.1 Parallel front nodes status assignment with partial pruning

Parallel front nodes status assignment is performed with a parallel mapping and reduction procedure. Recall that such a procedure combines two phases. Firstly, the mapping phase applies a treatment to each task in parallel. Secondly, the reduction phase merges successively several tasks together and applies another treatment on each pair of tasks being merged in parallel. The reduction proceeds until all tasks have been merged into a single one.

The front, represented as a *sorted* (see Section 4.5) array of BVTT nodes is split into equally-sized tasks to be handled in parallel. The parallel mapping operates on each such task, thus performing all the culling tests along the front. The decision of whether or not a node is to be traversed can be taken during this phase as well. However, deciding whether a node must be deleted or not depends on the status of its immediate neighbor (if any) that shares the same BVTT parent node. In the following developments, the term *neighbor* will be dedicated to nodes that share the same BVTT parent. Two scenarios may occur:

1. The node and its neighbor belong to the same task. In that case, the decision can be taken immediately;

2. The node and its neighbor are located into different tasks. That situation is handled during the parallel reduction phase when reading the culling test result for the corresponding nodes of each task.

Figure 4.17 illustrates this procedure, starting with the front shown in Figure 4.13a. At the first step, i.e., *Map*, each task assigns either a *ToDelete* (red) or *ToTraverse* (green) flag to each node. In this example, there is only one node per task so no pruning is performed sequentially at this point. During the first reduction, the nodes (A_2, B_2) and (A_2, B_3) are neighbors and being merged into the same task. It is the automatic scheduling provided by TBB that is responsible for selecting which adjacent tasks are being merged and in which order. However, they have different marks so they are left unchanged. During the second reduction, the nodes (A_4, B_3) and (A_5, B_3) are the only ones being neighbors and being merged into the same task. In addition, they are both tagged *ToDelete*. Therefore, the culling test is performed on their common parent. Here, the culling test passed, meaning pruning is applicable. This is done by replacing the node (A_4, B_3) by its parent (A_3, B_3), and setting the status of its neighbor (A_5, B_3) to *Deleted*. The third reduction does not do anything because the nodes (A_2, B_3) and (A_3, B_2) have different marks.

Let us point out that all these status assignment operations can be performed in-place without race-conditions since each task operates on different front nodes. Here also, the pruning only ends up setting the *Deleted* status to the removed nodes, i.e., the size of the array containing the front is not modified. Thus, all tasks share a pointer to the same array representing the front, i.e. the division into tasks is purely logical, and modify the nodes when assigning them a status and modifying it as required during pruning process.

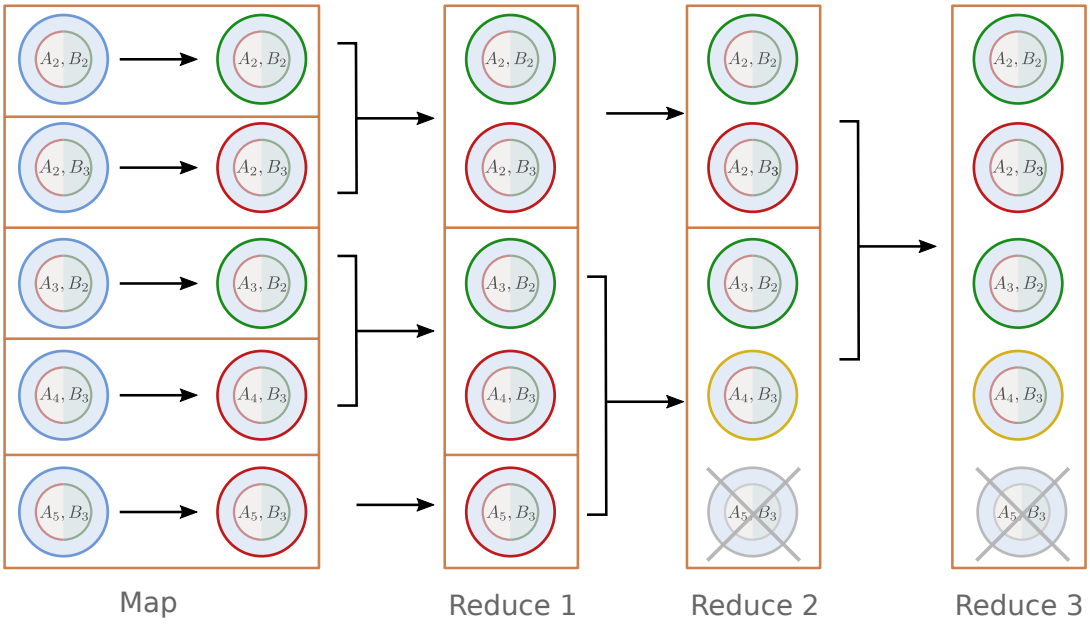


Figure 4.17: Parallel front nodes status assignment with partial pruning. One orange rectangle represents one task. Here, one partial pruning occurs during the second reduction.

4.6.2 Parallel front traversal and construction of the new BVTT front

Parallel front traversal is also a parallel mapping and reduction procedure. In the following, we assume the front is sorted (as defined in Section 4.5.1) and each of its BVTT nodes are referred to as N_k where the index is such that N_i appears before N_j in the front if and only if $i < j$:

- At first, one task T_k is created for each N_k of the BVTT front that has not been marked *Deleted*;
- Then, the mapping phase operates on each T_k in parallel. If a T_k contains a N_k marked *ToTraverse*, then the BVTT traversal presented in Section 4.2 is executed on the sub-tree of the BVTT rooted by N_k. The new BVTT nodes reached by this traversal form a partial front which is stored into an array A_k. If a T_k contains a N_k not marked *ToTraverse*, then A_k is simply initialized to $A_k = \{N_k\}$.
- Finally, the parallel reduction phase recursively combines pairs of tasks T_k and T_{k+1} by appending A_{k+1} to A_k. The recursive combination of all the A_k yield the new BVTT front. Moreover, all BVTT leaves on which the culling test failed are identified and collected for future processing by the parallel procedure presented in Section 4.6.3 for the exact LMD computation.

Special care must be taken regarding the optimization related to supermaximal features presented in Section 4.2.3. Indeed, this optimization relies on a set $\mathcal{E}$ that stores pairs of supermaximal features that do not need to be collected more than once for processing by the exact LMD computation algorithms. Thus, to avoid erroneous concurrent accesses to $\mathcal{E}$ during the parallel front traversal, the data structure containing $\mathcal{E}$ is replaced by a concurrent set (`tbb::concurrent_unordered_set`) that allows parallelized algorithms to perform concurrent modifications of $\mathcal{E}$ in a thread-safe way. Fortunately, the use of this concurrent data structure reduces the performances marginally only, because very few write operations occur compared to the number of read operations. This is due to the fact that only one write operation is performed per pair of supermaximal features for which an analytical LMD computation method exists, while one read operation is performed per pair of feature areas.

4.6.3 Parallel LMD computation and trimming

Parallel LMD computation is achieved through:

1. Preallocating a sufficiently large output buffer for the computation of LMDs between all pairs of BVH leaves collected during the parallel front traversal;
2. Executing in parallel each exact LMD computation algorithm over each selected pairs of BVH leaves. Each such algorithm execution is given a pointer to a sufficiently large number of free slots of the output buffer in order to store the computed LMD footpoints.

As mentioned in Section 4.3.1, the exact choice of LMD computation algorithms, and thus the size of their outputs, depends on the feature types referenced by each BVH leaf.

On the one hand, if the LMD computation algorithm being executed is based on the Newton method then, only one pair of LMD footpoints will be generated for the selected pair of BVH leaves. On the other hand, pairs of features where an analytical method can be executed may output more than one pair of LMD footpoints. This number of pairs of LMD footpoints generated is actually equal, at most, to four since no analytic root-finding algorithm can be set up for polynomials of degree greater than four, as stated by Abel's Impossibility Theorem [2]. To sum up, if N is the total number of BVH leaf pairs to be processed, preallocating a buffer able to store $4N$ pairs of LMD footpoints at most, is sufficient.

While very efficient, this approach uses more memory than necessary because four slots will be allocated to the output buffer even for, e.g., pairs relying on the Newton method where only one would have been sufficient. Therefore, if memory usage is of concern, several alternatives are possible to:

- Perform a sequential pass on each BVH leaf pair to determine the actual maximum number of solutions each of them can produce, i.e., n_i. This can be used to allocate a smaller array whose size equals $\sum_i n_i$. Simultaneously, an index corresponding to the location where the LMD computation algorithm should output its results is associated to each BVH leaf pair;
- Perform a parallel map and reduction where each LMD computation algorithm will output LMD footpoints into a local array during the *map* phase. The reduction then concatenates those arrays.

Within our application scope, the memory overhead of the method presented here is small enough to avoid relying on other approaches that induce runtime overheads.

4.6.4 Discussion regarding load-balancing

Work balancing is essential for good performances. Indeed, assigning work to each thread evenly helps keeping them busy and maximize the amount of computations performed simultaneously. While designing an efficient scheduler from the ground-up is a task too heavy to be undertaken during this thesis, simple observations can help selecting a good one among those available from a parallelization framework like TBB. Each phase of the parallel CD pipeline has its specificities:

1. Parallel status assignment with partial pruning: an equipartition of the number of front node among each thread provides a good load balancing in practice because the effective computations (culling test) are of the same nature and complexity for all the nodes of the BVTT front;
2. Parallel front traversal is a very unbalanced operation, and it is hard to predict accurately the amount of work induced by one traversal starting at one BVTT node. Moreover, the traversed sub-trees tend to have small depths so parallelization of the traversal of one sub-tree itself proved inefficient in practice. The work-stealing scheduling strategies, as provided by TBB, are good choices since it can adapt dynamically the work load when some tasks require more time that others;
3. Experiments show that the parallel LMD computation is fairly well balanced, even if the nature of the exact LMD computation algorithms differ. Indeed, Newton-based approaches generally converge quickly enough if they are started at the same convergence point (if any) found at the last time-step. Thus, an equipartition of the work among threads is an efficient choice.

These different workload characteristics justify the choice of splitting the CD pipeline into three steps. Indeed, it is possible to start the traversal as soon as a node is marked *ToTraverse*, and it is possible to perform the exact contact computation as soon as the traversal reaches a leaf. However, merging those three phases in any way would make the work load to be too unbalanced and reduce the efficiency of the parallelization. The Chapter 5 shows that the proposed parallel pipeline achieves real-time performances.

4.7 Conclusion

From a technical point of view, this chapter has described choices that can be made in order to compute efficiently LMDs between smooth BRep models as well as deformable curves.

A sequential approach based on simultaneous BVH traversals has been described to introduce choices made regarding exact LMD computations algorithms depending on the type of curves

or surfaces involved. Indeed, analytic algorithms can be used for most of them, which leads to better computational performances than the generic Newtown-based iterative root-finding algorithms. In addition, these analytical methods are very robust and helpful to enforce the existence of a unique solution over each feature. Also, analytical methods combined with the concept of supermaximal features avoid redundant computations, which also contributes to the performance improvements. A simplification of the polyhedral cone has been introduced to bound LMD directions and reduce the computational cost of the culling tests while keeping them sufficiently discriminative.

Deformable curves are special cases as they do not have tight normal cones like vertices, curves, and face areas of BRep models do. Therefore, polyhedral tangent cones and polyhedral solution line cones are much more suited to filter areas that may not contain LMD footpoints. In addition, their curvature distribution may vary between time steps. Consequently, a static subdivision in almost-flat parts at pre-computation time does not make sense. Thus, it is more efficient to rely on an hybrid approach where few regularly spaced static subdivisions, updated at each iteration, are combined with dynamic subdivisions to locate more accurately LMDs.

In order to achieve real-time simulations, improvements of bounding volumes, smart algorithmic choices, and avoidance of redundant LMD computations using supermaximal faces is not enough. Much is to be gained by considering two common approaches:

1. Taking advantage of temporal coherence: assuming the objects do not move much between two time steps, it is possible to avoid having to perform simultaneous BVH traversal from their roots. Instead, the traversal may be restarted from its stop after the last update. Moreover, this assumption improves dramatically intersection tests between polyhedral cones because the execution of the $O(n^3)$ intersection algorithm may not always be necessary;

2. The parallelization of the CD pipeline: this can be achieved in a three-step process in order to maximize load balancing. Performance gains are significant since the computation of exact LMDs is the most time-consuming part of the CD pipeline and can be efficiently parallelized without implementation difficulties.

The next chapter shows this CD framework, implemented as part of a complete multibody dynamics simulation engine with intermittent contacts, in action on industrial CAD models.

Chapter 5

Experimentations and benchmarks on industrial models

Our CD method is implemented within the interactive multibody physics simulation framework XDE [91] developed by the CEA[1]. Simulations are performed with no friction, no penetration, and are based on a time-stepping scheme inspired from the seminal work of Moreau [94] and Jean [65]. Contact constraints follow the Signorini contact law [122] and are solved using a Gauss-Seidel based iterative solver, similar to the algorithms described in Acary et al. [4].

In this chapter, the proposed approach is compared to a state-of-the art distance-based CD method that operates tessellated models only [90] and that is also integrated into the XDE simulation framework. The benchmarks presented in Sections 5.1 and 5.2 are performed on two scenarios simulating insertions with small mechanical clearances. The third benchmark, in Section 5.3, highlights performances improvements brought by polyhedral tangent cones and polyhedral solution line cones as part of our hybrid method for CD between deformable Bézier curves. However, no actual dynamics simulation involving those curves is demonstrated as it is not yet supported by the XDE framework.

5.1 First scenario: telerobotics insertion task

The first scenario presented is derived from use-cases belonging to the field of the teleoperation of robot arms performing insertion tasks inside a hostile (extremely hot or irradiated) environment. In particular, the models are inspired from industrial models[2] belonging to the French multinational company Areva, specializing into nuclear and renewable energy.

The simulation at hand addresses the slave arm side of a force-feedback teleoperation system. A robotic arm has to be manipulated by an operator (here, through a 3D mouse) in order to insert the conical mobile solid (see Figure 5.2b) into a receptacle with a static hole that has a similar shape (see Figure 5.2a). The mechanical clearance between the mobile solid and the hole after insertion is small, i.e., equal to 1% of the hole's largest radius. Various steps of the simulation are depicted in Figure 5.1. In order to complete the insertion task, the tooth pictured in Figure 5.2a on the receptacle must be properly aligned with the channel depicted in Figure 5.2b. The connection is of type bayonet locking system, which requires translational and rotational sliding motions.

[1]The French Alternative Energies and Atomic Energy Commission

[2]The actual models provided by the manufacturer are not demonstrated in this manuscript for confidentiality reasons. The replacement models proposed here have been assembled following slightly different shapes.

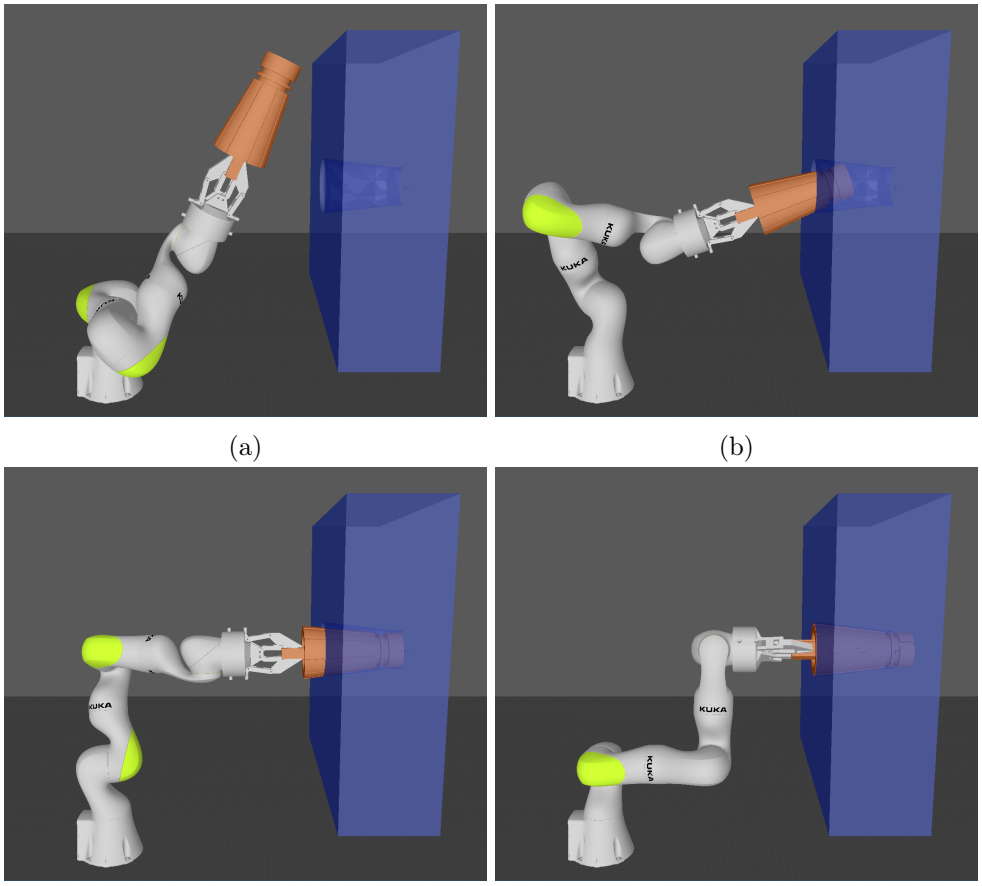


Figure 5.1: Interactive insertion of a conical solid (with a cylindrical end) into a hole with a similar shape.

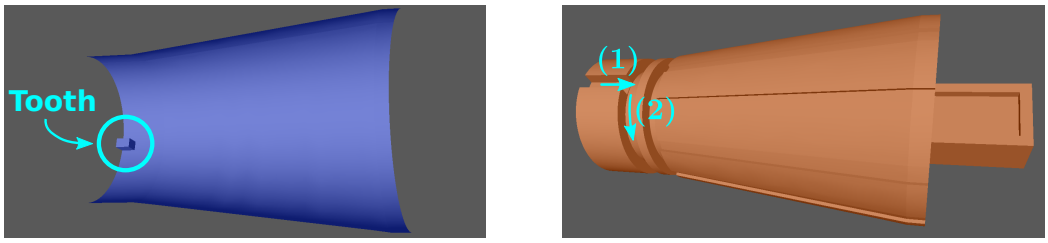


Figure 5.2: (a) Cut view of the static object hole. (b) Close view of the manipulated solid. It has to be translated then rotated such that the tooth circled in (a) follows the path marked in (b).

Section 5.2.1 describes the geometric models used for this simulation. Comparisons of computation times for CD between our framework and the polyhedron-based LMD computation framework LMD++ [90] are analyzed in Section 5.1.2.1. Scalability analysis of our parallel CD pipeline as well as a detailed view of the performance improvements brought by each of our contributions are presented as well in the Sections 5.1.2.2 and 5.1.2.3.

5.1.1 Models description

This section describes the geometry of the two input models. Both are depicted in Figures 5.3 and 5.4 with the following color code:

Colour	Signification
Gray	Planes
Dark blue	Cones
Light blue	Cylinders
Red (curve)	G^1 or concave curve

Also, both models contain few tori forming blending areas. Those figures display some elementary results of ongoing works regarding the identification of G^1 and concave edges. As mentioned in Section 1.3.1.3, these edges could be ignored by the CD framework since they cannot contain any LMD footpoint.

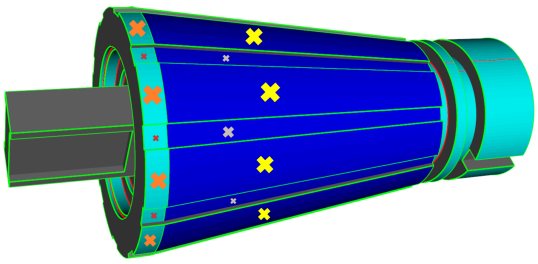


Figure 5.3: The manipulated solid. Coloured crosses indicate some faces parts of the same supermaximal face.

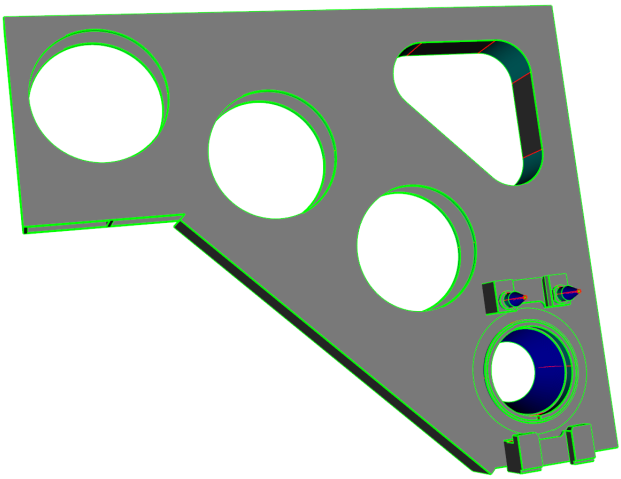


Figure 5.4: The receptacle with a conical hole (bottom-right).

Edges are mostly lines and circles, with a few Bézier curves. The Table 5.1 lists the exact number of faces and edges, as well as the number of supermaximal faces and edges. Clearly, the number of supermaximal edges and faces are more significant for the manipulated solid because the solid repetitive patterns highlighted in Figure 5.3 since the overall shape is almost symmetric with respect to a single axis. For example, the outermost conical (resp. cylindrical) surfaces marked with yellow (resp. orange) crosses are composed of several areas of the same untrimmed cone (resp. cylinder).

Because of the requirements described in Section 1.3.1.2 regarding the ability of a distance-based methods to output negative values of the gap function, the manipulated model is eroded then implicitly dilated by a margin equal to 0.5% of the largest radius of the manipulated solid. This dilation does not modify the mechanical clearance between the two solids and its only downside is to round some corners which does not influence the quality of the simulation here.

	Conical solid	Receptacle
Num. of faces	108	15
Num. of supermax faces	65	13
Num. of edges	194	25
Num. of supermax edges	136	24

Table 5.1: Number of faces and edges, and supermaximal entities for the first simulated scenario.

5.1.2 Running times comparisons

Three phases of the simulation can be distinguished through the benchmarks because they all show significantly different performance characteristics:

1. From the beginning to the time step 800: the *free flight* phase that corresponds to Figure 5.1a where the operator adjusts the position of the conical solid so that the insertion process can begin as in Figure 5.1b. Objects start being in contact at the time step 500;

2. Between the steps 801 to 1300: the *insertion and locking* phase where the operator slides (see Figure 5.1c) and rotates the solid into the receptacle hole in such a way that it becomes fully inserted. Let us note that some precise rotational manipulations are necessary in order to account for the tooth shown in Figure 5.2. Once fully inserted as in Figure 5.1d, small rotations are applied to the solid.

3. Finally, from time step 1301 to the end of the simulation, the *extraction* phase removes the solid from the hole. Objects stop being in contact at the time step 1400.

The following sections propose various benchmarks in order to validate the efficiency of our CD framework. Firstly, Section 5.1.2.1 compares the proposed framework with the distance based framework LMD++ [90], which was already integrated into the XDE dynamics simulation engine. LMD++ relies on polyhedral approximations of the two input models. The chosen polyhedral approximations are such that the maximal chordal error (on each model) is equal to the minimal mechanical clearance between the two solids. That way, the insertion using the polyhedral models is possible even though the resulting user experience is not optimal because of unrealistic interactions due to the numerical deviations caused by the geometric approximations. Using a finer tessellation would improve the simulation realism at worse performance price.

Secondly, the scalability of the parallel CD pipeline (see Section 4.6) is evaluated in Section 5.1.2.2. In addition, the computation times of each of its three components (parallel front marking, parallel front traversal, and parallel exact LMD computation) of the parallel CD pipeline are studied separately. Finally, Section 5.1.2.3 highlights the performance improvements brought by each contribution presented in this manuscript.

5.1.2.1 Comparison of the BRep-based framework with a Polyhedron-based framework

Figure 5.5 shows the computation times dedicated to CD using our framework compared to the time spent by LMD++. Let us observe that LMD++ does not exploit parallelism for the case of the LMD computation between two models only. Therefore, in addition to the computation times of our parallel CD pipeline described in Section 4.6, the following benchmarks also include computation times using our sequential pipeline presented in Section 4.1. As discussed in Section 5.1.2.2, the LMD++ framework is less likely to be able to benefit from parallelization as

much as our method does. Overall, one can see that the proposed method matches the performances of LMD++ when multithreading is disabled and even can perform the simulation in real-time when its parallelization (here, with 4 threads) is enabled.

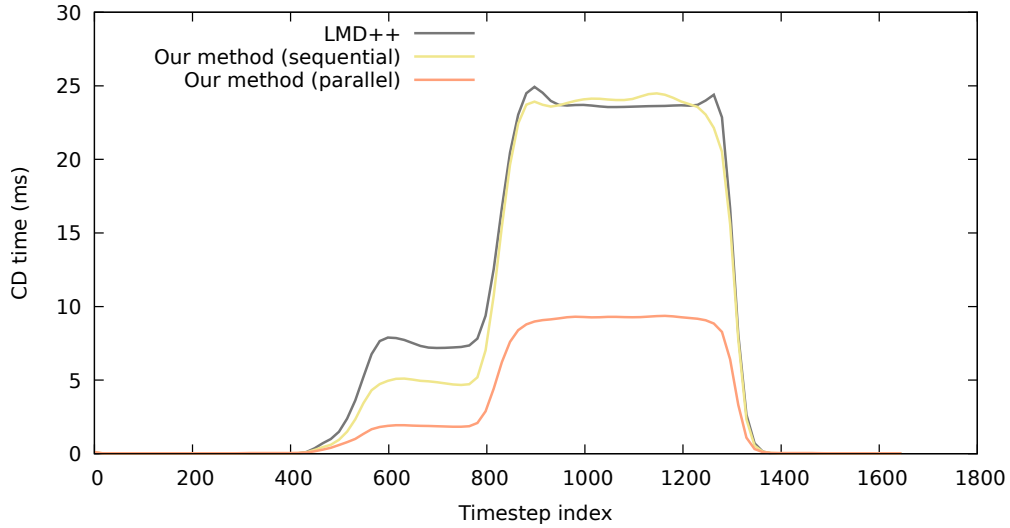


Figure 5.5: Times for the computation of LMDs during the simulation of the insertion depicted in Figure 5.1. Each curve corresponds to a different choice of CD method.

Besides computation times for CD, the number of generated LMD has a significant impact on the performance of the overall simulation. Indeed, each LMD represents at least one contact constraint that has to be handled by the dynamics solver (and even more constraints-per-contact can be necessary to simulate, e.g., friction) so the higher the number of contacts, the larger the time to solve the constraints. Figure 5.6 shows that the number of LMDs our method generates corresponds to less than 40% of the number of LMDs generated by the tessellation-based approach.

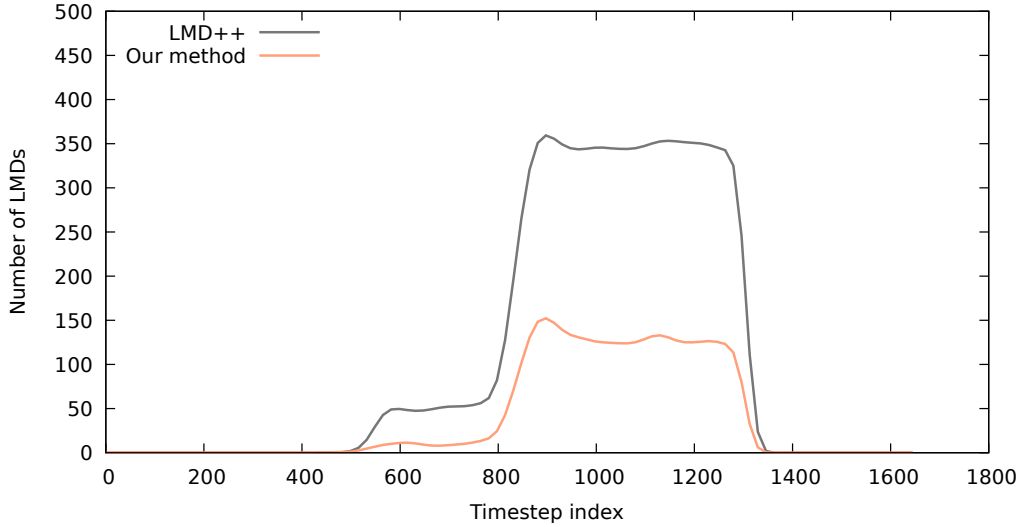


Figure 5.6: Number of LMDs generated by the LMD++ (grey) compared to our method (orange).

Finally, Figure 5.7 shows the computation times required by the whole physics engine to execute one time-step, i.e., including CD, resolution of the contact constraints, and position

updates. These curves highlight the fact that the lower number of LMDs generated further improves the overall performance of the proposed CD framework. Indeed, even the sequential version of our framework leads to better performances because of the lower number of contact constraints to be solved. However, it must be kept in mind that the XDE dynamics simulation engine relies on a linear constraints solver. As mentioned in Section 1.4.3, a non-linear constraints solver should be preferred when smooth BRep are used to represent the solid geometries. Such a non-linear constraints solver would induce additional computation times that are not reflected by the Figure 5.7.

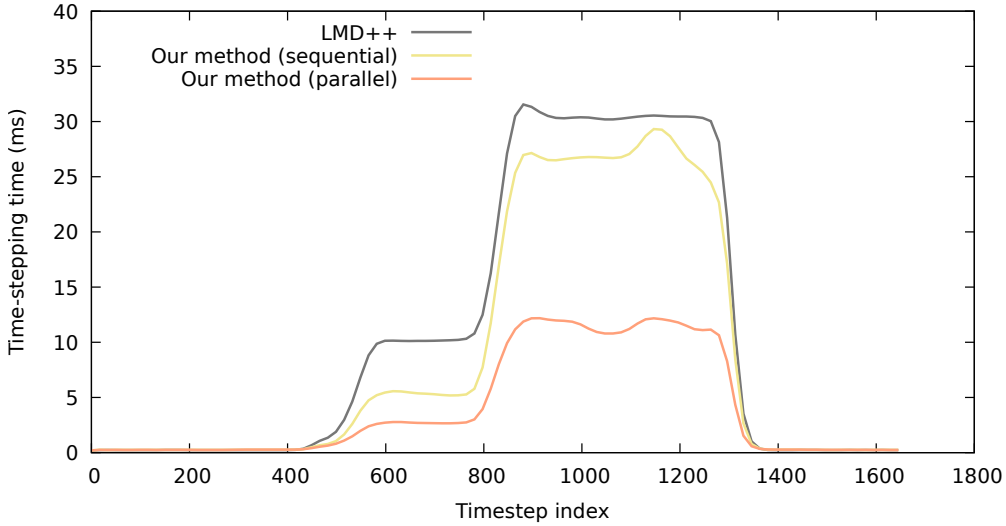


Figure 5.7: Times for the one step of the dynamics simulation engine (including CD, contact constraints resolution, and position updates) for the insertion depicted by Figure 5.1. Each curve correspond to one different choice of CD pipeline.

Overall, for this insertion task, the method presented in this manuscript is faster than the tessellation-based approach and generates much less contact points, making the overall simulation clearly more efficient and more accurate since it is not perturbed by numerical artifacts intrinsically generated by geometric approximations.

5.1.2.2 Evaluation of the tasks parallelism

Each curve of Figure 5.8 shows computation times for CD using the parallel CD pipeline described in Section 4.6 with various number of threads. The use of two threads brings an immediate performance improvement of 50%, allowing our framework to target real-time applications. Four threads, which equals the number of physical cores of the microprocessor used, brings an additional improvement of around 30%.

Figures 5.9 to 5.11 show similar benchmarks but for each elementary parallel stage (marking, traversal, and LMD computation) of the parallel pipeline as described in Figure 4.16. Each stage has a different amplitude in terms of CD times. Indeed:

- Parallel front marking takes at most 4.3ms in a single-threaded setting. Beyond 2 threads, the benefit of parallelism is reduced. This is mostly due to the uneven costs of the culling tests. Indeed, the cost of one culling test may vary significantly depending on what steps (refer to Section 2.4.3) is executed completely or only partially, due to temporal coherence (see Section 4.5.2);

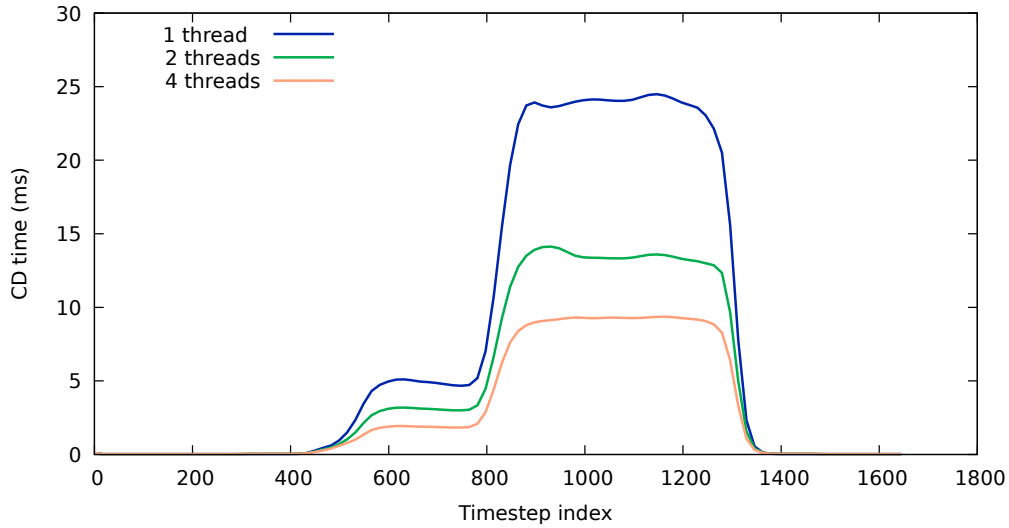


Figure 5.8: Computation times of the parallel CD pipeline during the insertion task. Each curve corresponds to a different number of threads.

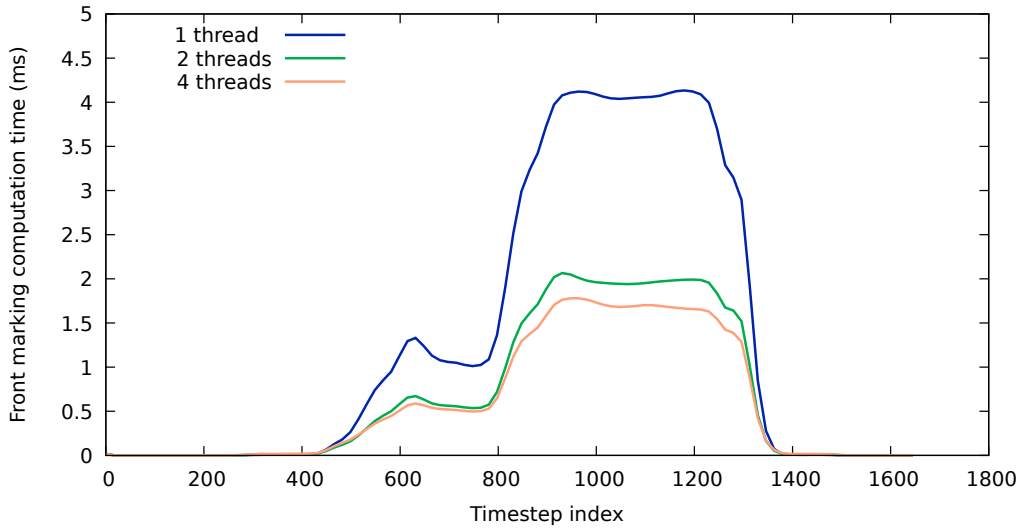


Figure 5.9: Parallel front marking computation times.

- Parallel front traversal takes at most 2ms in a single-threaded setting. Those low computation times come from the fact that the incremental updates of the BVTT front are cheap. At this point, most of this time is spent in the parallel reduction that assembles the new BVTT front, which intrinsically does not significantly benefit from parallelism;

- Parallel computation of the exact LMDs takes up to 19ms in a single-threaded setting. This is a quite well-balanced step that benefits greatly from parallelism because each LMD computation can be performed completely independently from the others. Moreover, temporal coherence often accelerates the convergence of iterative methods, making them as cheap as the analytical ones. Please, refer to Section 4.3 about the choices of LMD computation methods, and Section 4.5.3 about the related benefits of temporal coherence.

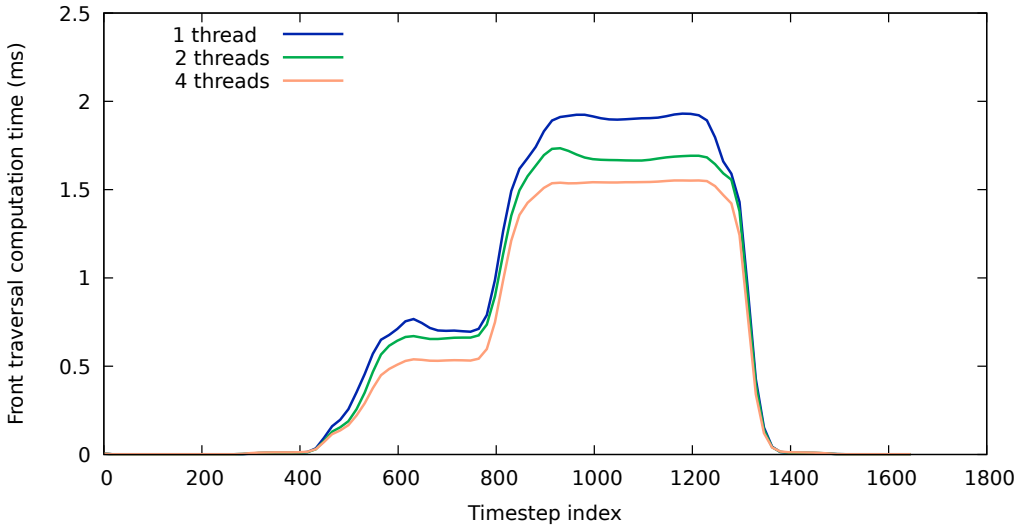


Figure 5.10: Parallel front traversal computation times.

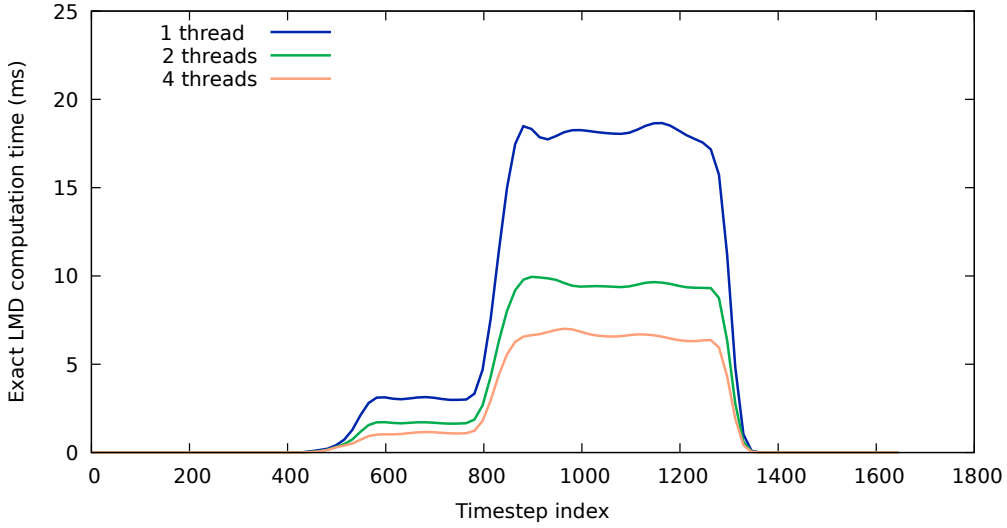


Figure 5.11: Parallel exact LMDs computation times (including checking that their footpoints do not not lie in a hole with the approach described in Section 4.4.1).

Clearly, the proposed CD pipeline is dominated by the exact LMD computation phase in terms of computation times. This can be explained by the cost of the analytical and iterative LMD computation methods identified in Section 4.3 compared to the low cost of the culling tests and front traversal when temporal coherence is exploited, as emphasized in Section 4.5.2. Those observations justify the choice (discussed in Section 4.6.4) of splitting the CD pipeline into three parallel phases since all of them have different peak computation times and characteristics regarding load-balancing.

Finally, the exact LMDs computation stage is the most time-consuming stage but also the one that benefits the most from parallelism. Therefore, the overall CD pipeline itself benefits significantly from parallelism. This is a clear distinction with polyhedron-based methods for which the most computationally-intensive phase is the BVTT traversal (since the exact LMD computation between segments and triangles is cheap), which is much less straightforward to parallelize efficiently.

5.1.2.3 Contribution-based evaluation of performance improvements

Figure 5.13 shows several computation times for CD. The topmost curve corresponds to the CD times spent by the implementation of the complete sequential CD pipeline illustrated in Figure 4.1. However, the culling test described in Section 2.4.3 only includes the curvature-based compatibility masks (see Section 2.2) and OBB intersection tests. Orientation-based bounding volumes are ignored. Moreover, identification of supermaximal features (see Section 2.3) has been disabled together with the corresponding optimization as per Section 4.2.3. Finally, exploitation of temporal coherence, as described in Section 4.5, has been deactivated as well, i.e., the whole BVTT traversal is restarted at its root at each time step. The other curves show how the successive additions of our contributions improve CD times. Figure 5.12 shows which contribution are taken into account for each performance curve and each curve can be identified using the same block color.

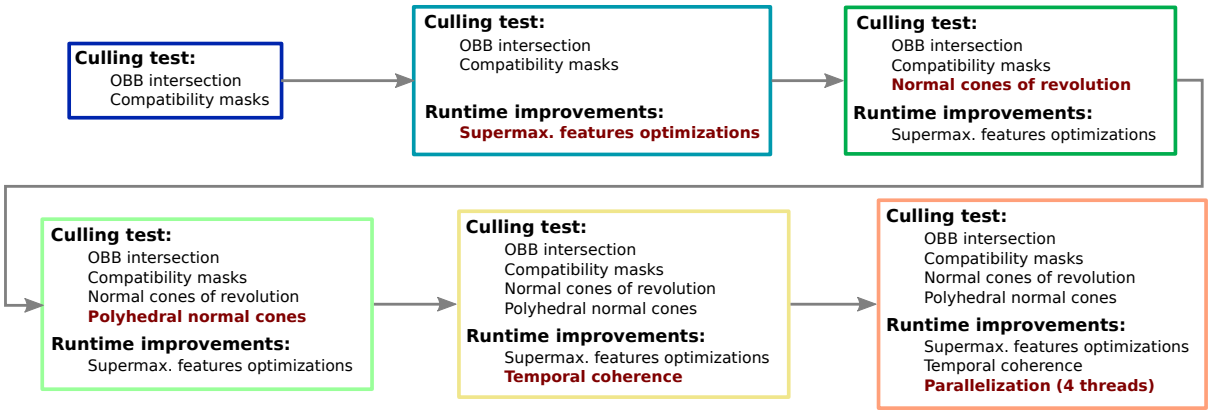


Figure 5.12: Features activated for the CD computation times shown in Figure 5.13. Each coloured box corresponds to the CD time curve with the same color. Each successive benchmark takes in account exactly one additional contribution highlighted in bold red.

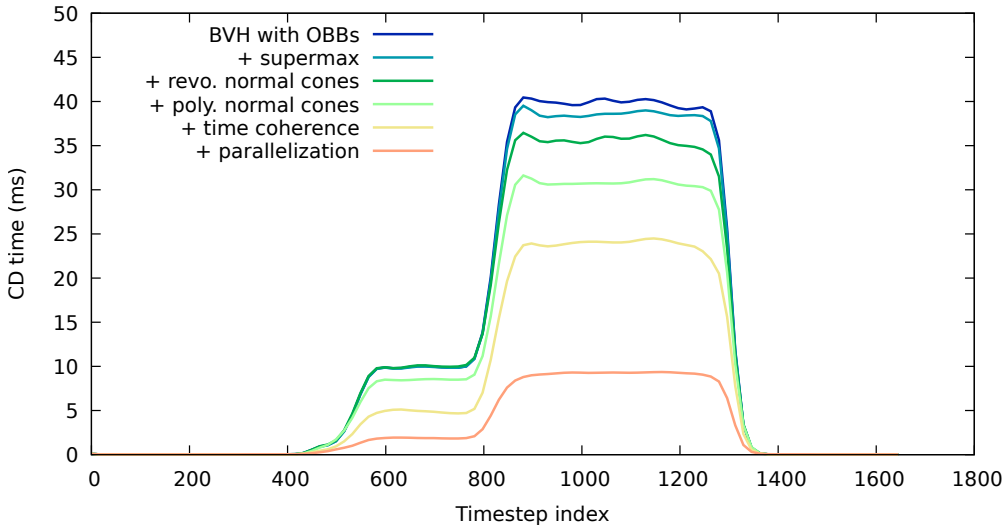


Figure 5.13: CD times at each time step during the whole insertion task. Each curves shows the effect of an added contribution on the computation times.

Several observations are noteworthy:

- The supermaximal feature-based optimization brings a slight performance improvement that becomes noticeable only once the full insertion is achieved. Indeed, that is the only configuration where the redundant computations discussed in Section 2.3 can occur, considering the localization of the supermaximal features identified (see Figure 5.3);

- The new polyhedral normal cones, presented in Chapter 3, brings a significant performance improvement over the normal cones of revolution from Johnson et al. [70], which has been adapted to entire BReps rather than isolated curves and surfaces, only. This validates the observation expressed in Section 3.1 regarding the fact that tighter orientation-based bounding volumes can lead to better performances;

- The parallelization of the CD pipeline presented in Section 4.6 is the most efficient contribution and contributes to the generation of an overall real-time simulation because it keeps CD times bellow 10ms, even after the complete insertion phase.

5.2 Second scenario: ROV teleoperation insertion task

The second scenario depicted in Figure 5.14 features the underwater insertion of a hotstab into a receptacle using a Remotely Operated Vehicle (ROV). A hotstab is a cylindrical object used to establish or prevent hydraulic connections. This scenario requires a great level of simulation accuracy since the smallest mechanical clearance is equal to 0.1mm where the largest hotstab diameter is of 40mm. Similarly to the previous scenario, the task is divided into different steps. Firstly, the hotstab is initialized outside of the receptacle and is inserted. Then, once fully inserted, the user applies small rotations to the hotstab with respect to its symmetry axis. Finally, it is extracted from the receptacle to recover its initial position.

Section 5.2.1 describes the geometric models used for this simulation. Comparison of computation times for CD between our framework and the polyhedron-based LMD computation framework LMD++ [90] are available in Section 5.2.2. The scalability analysis of the proposed parallel CD pipeline and the performance improvements brought by each of the contributions, are not presented for this second scenario since the results lead to the same conclusions than the first one.

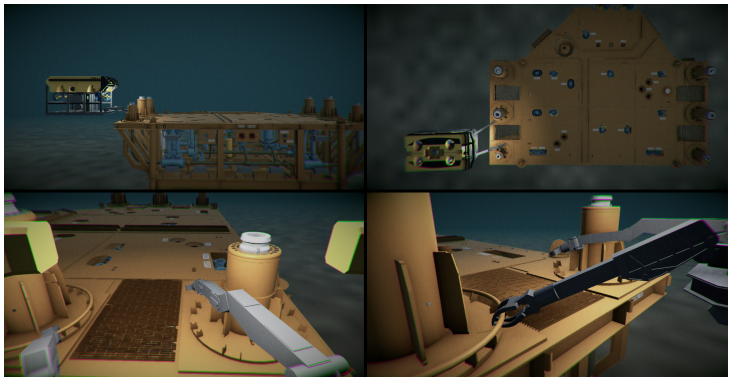
5.2.1 Models description

This simulation scenario involves two solids. The receptacle is static and shown in Figure 5.15. Figure 5.15b gives a transparent view of the receptacle in order to visualize the hole. Note that the hole itself has a complex shape composed of successions of several conical and cylindrical surfaces.

The hotstab is mobile and user-driven with a 3D mouse, and shown in Figure 5.16. It comes into two versions for comparison:

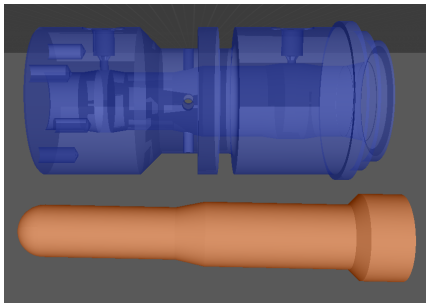
- Figure 5.16a shows the original industrial hotstab model as given by our industrial partner. Let us note that it contains several details such as engraved writings and a handle that is not expected to interact with the receptacle;

- Figure 5.16b shows a simplified hotstab model, also made available by our industrial partner, where functional features are represented, only. Several geometric details have been removed since they are not influencing the quality of the simulation. In practice, they are needed by the real system to ensure hydraulic connections. Moreover, the tip of the hotstab has been replaced by a spherical area whereas the original solid contained NURBS surfaces and conical areas. While this slightly modifies the shape of the tip, this does not

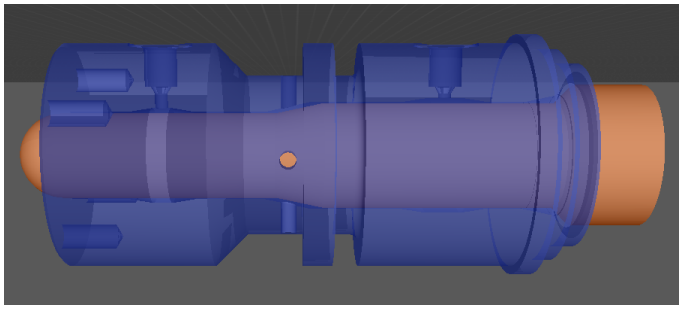


(a)

(b)



(c)

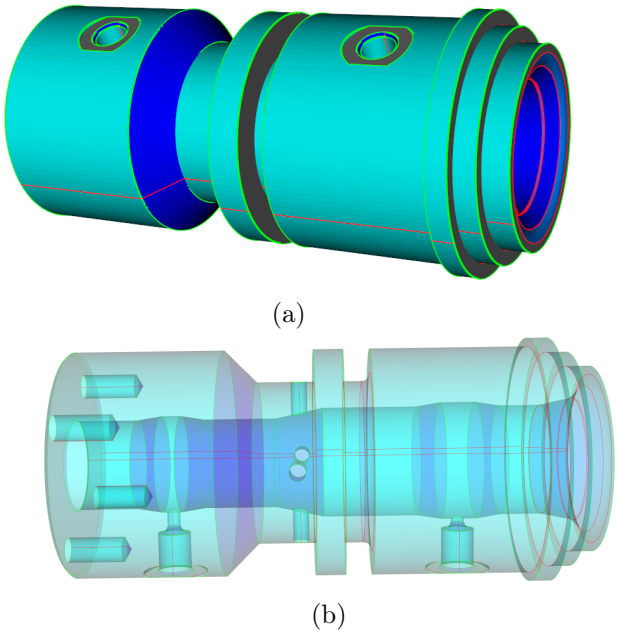


(d)

Figure 5.14: An underwater (yellow and gray) ROV about to operate on a (orange) manifold. (a) Seen from several angles. (b) Closeup view. (c) The hotstab (orange) before insertion into the receptacle (blue). (d) The hotstab after insertion into the receptacle.

significantly influence the simulation from the point of view of our industrial partner because contacts with the tip are not functional contacts and tend generally to be avoided. Besides the tip, dimensions of the functional parts and thus, the mechanical clearance expected after insertion have not been altered.

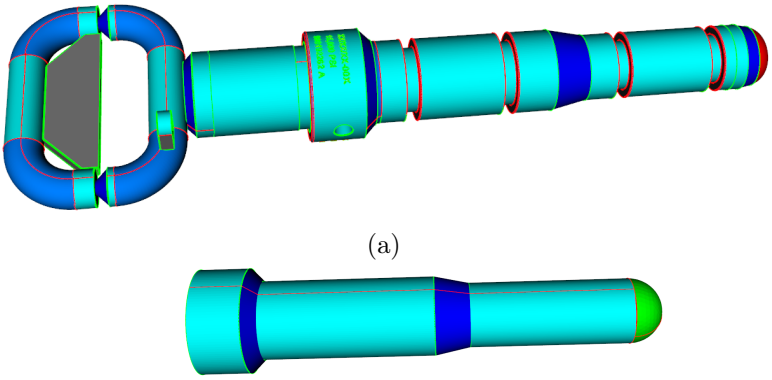
No simplification has been made on the original receptacle model. All the models presented in this section follow the colour code hereunder:



(a)

(b)

Figure 5.15: (a) Industrial model of a receptacle. (b) Transparent view.



(a)

(b)

Figure 5.16: (a) Original industrial model of a hotstab. (b) Defeatured and simplified model.

Colour	**Signification**
Light blue	Cylinders
Dark blue	Tori (handle) and Cones
Green (surface)	Spheres
Red (surface)	Bézier surfaces or NURBS
Green (curve)	Non-G^1 and non-concave curve
Red (curve)	G^1 or concave curve

Also, Figures 5.16 and 5.16a display some elementary results about ongoing works regarding the identification of G^1 and concave edges (some of them are not properly identified yet). As mentioned in Section 1.3.1.3, such edges could be ignored by the CD framework since they cannot contain any LMD footpoint. Just like the first scenario, edges are mostly lines and circles. The Table 5.2 lists the exact number of faces and edges, as well as the number of supermaximal faces and edges.

	Original hotstab	Simplified hotstab	Receptacle
Num. of faces	993	15	64
Num. of supermax faces	759	7	58
Num. of edges	1608	10	78
Num. of supermax edges	1451	5	72

Table 5.2: Number of entities and supermaximal entities for the hotstab scenario.

5.2.2 Running times comparisons

The performance comparison between LMD++ and the proposed framework regarding the insertion of the hotstab follows a similar procedure than the one described in Sections 5.1.2 and 5.1.2.1. In particular, when the LMD++ framework is used, the input models are tessellated with a maximal chordal error of 0.1mm, which corresponds to the smallest mechanical clearance allowed between the hotstab and the receptacle. This results in a much higher number of triangles than in the first scenario, thus causing even larger performance differences between our framework and LMD++. The three graphs presented here (see Figures 5.17 to 5.19) are analogous to the Figures 5.5 to 5.7 presented for the first scenario, i.e., they respectively plot the time for CD, the number of LMD generated, and the total time required by XDE to perform one time step. The hotstab model used for those benchmarks is the simplified model (see Figure 5.16b). However, Figure 5.17 also includes, in green, the CD times spent by the parallel CD pipeline using the original hotstab model. The hotstab is in a fully inserted position and is being applied small rotations in-place during the time steps 650 to 850.

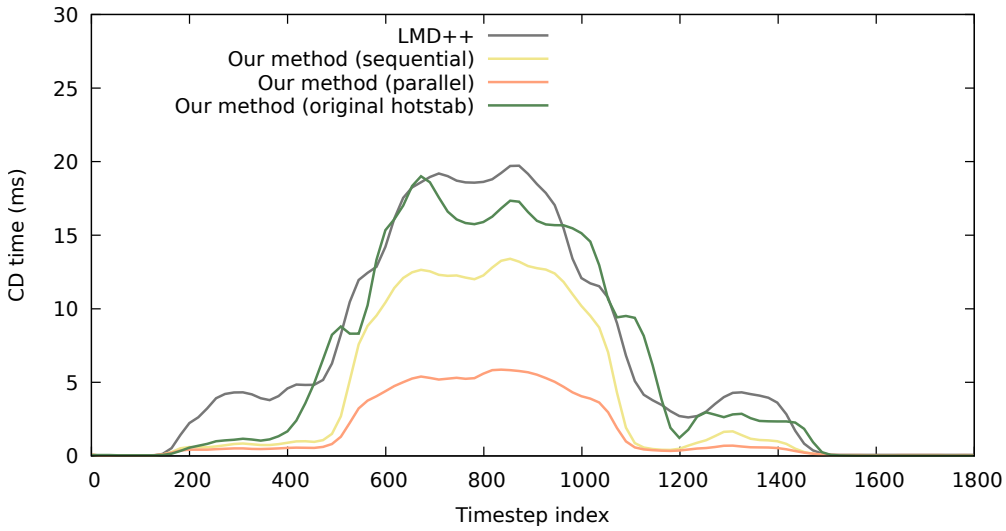


Figure 5.17: Times for the computation of LMDs during the simulation of the insertion depicted by Figure 5.14.

The conclusions are the same as for the first scenario in Section 5.1.2.1; especially since the benefits of the proposed framework appear even more clearly. Indeed, the number of LMDs generated after insertion is reduced to less than 50 (instead of more than 700 when using LMD++). Thus, the overall computation time for one time step is almost divided by 7 when using the simplified model, which does not induce any geometric approximation that has a consequence regarding functional contacts, i.e., the mechanical clearances remain the same as in the original hotstab model.

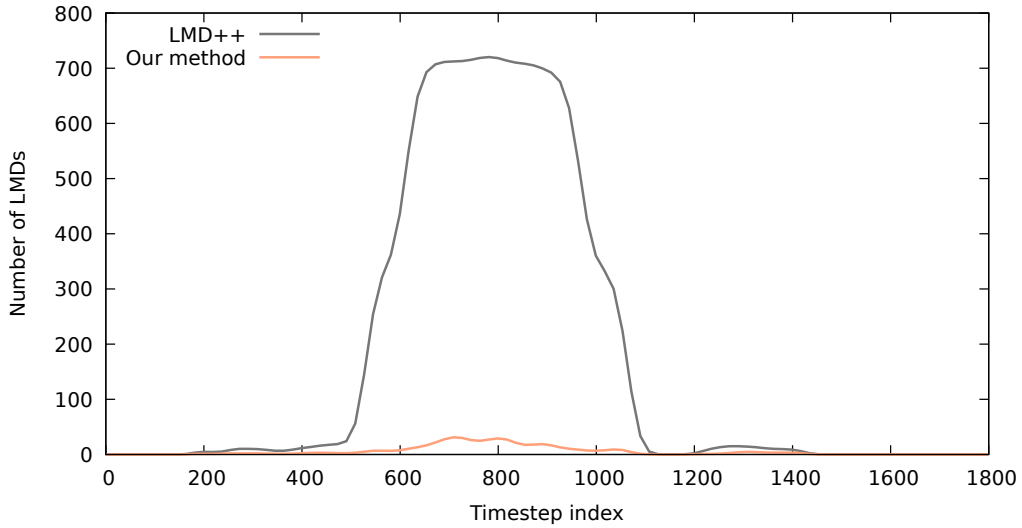


Figure 5.18: Number of LMDs generated by the LMD++ (grey) compared to our method (orange).

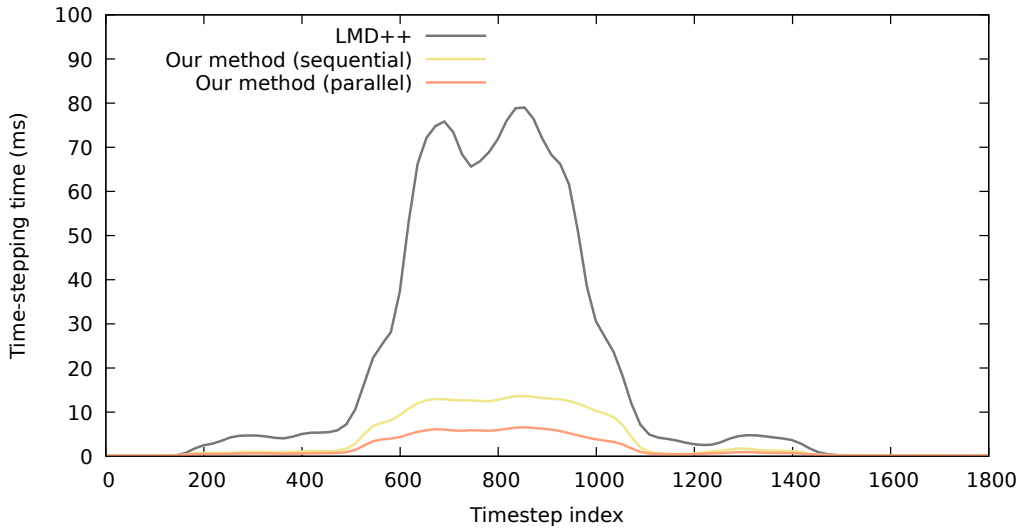


Figure 5.19: Times for one step of the dynamics simulation engine (including CD, contact constraints resolution, and position updates) for the insertion depicted by Figure 5.14. Each curve correspond to one different choice of CD pipeline.

5.3 Third scenario: curve deformation

Due to time constraints and the limited support of the dynamics of deformable beams in the XDE framework, the experimentation on deformable beams is limited to prescribed deformations, without any dynamic contact interaction with other solids. Therefore, the purpose of this benchmark is to show the efficiency of our main contribution regarding LMD computation between isolated curves: our new polyhedral tangent cones and polyhedral solution line cones presented in Section 3.5.2. To this end, the simulation scenario incorporates the difficult case of the self-collision detection on a spring subjected to a cycle of compression and released configurations, as shown in Figure 5.20. From a CD standpoint, this type of self-collision detection is difficult to handle because, when the spring gets compressed, it reaches a configuration close to a conformal contact with itself (the contact being a spatial curve).

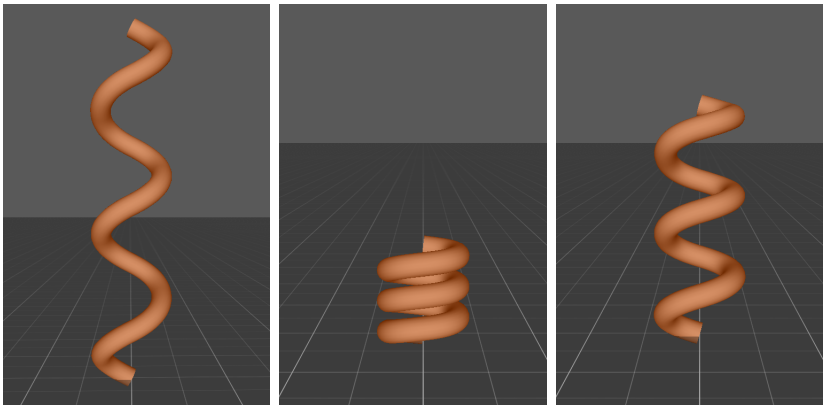


Figure 5.20: A spring represented as a set of dilated Bézier curves. (a) Before compression. (b) During compression. (c) After compression.

In Figure 5.20, the spring is described with a set of six Bézier curves of degree three. Each such curve is implicitly dilated by a margin equal to the spring's cross-section radius. This choice of curves is motivated by the future objective to simulate beams with circular cross-sections using a Finite Element Method with spatial discretization. Each such finite element would be represented as a cubic Hermite spline, which is equivalent to a Bézier curve of degree three. Moreover, note that similar almost-conformal self-collision may occur as well when the curves represent beams wound around another component, e.g., a pulley.

Figure 5.21 highlights the efficiency of the polyhedral cones when it comes to the culling test. It shows, at each time-step, how many potential contact pairs have been identified by the dynamic subdivision method given by Algorithm 13. In particular, the culling test on the line 4 of Algorithm 13 is modified to be either:

- The orthogonality test between the curves' tangent cones and solution line cones bounded by cones of revolution, as described in Section 3.5. This is the method from Johnson et al. [70] (gray curve);
- The orthogonality test from Johnson et al. [70] combined with intersection tests of OBBs in order to filter curve segments that are too far apart from each other (black curve);
- The orthogonality test between the curves' tangent cones and solution line cones bounded by the polyhedral cones described in Section 3.5.2 (orange curve);
- The orthogonality test between the polyhedral tangent and solution line cones combined with intersection tests of OBBs in order to filter curve segments that are too far apart (yellow curve).

In all cases, the recursive subdivision stops whenever the culling tests succeed or when the curves become almost flat because of the line 7 of Algorithm 13.

As discussed in Section 2.4.1, the flatness condition leads to test whether the tangent cone of each curve can be bounded by a cone of revolution smaller than a user-defined threshold (here 0.5rad), or not. This flatness criterion remains the same independently from the choice of orientation-based bounding volume for the line 4 of Algorithm 13 in order to keep the comparison fair.

Let us observe that among all those culling tests, the method based on cones of revolution alone (without OBBs) is the less discriminative. Actually, polyhedral cones are sufficiently tight to reduce the effect of the OBBs to a negligible proportion, here.

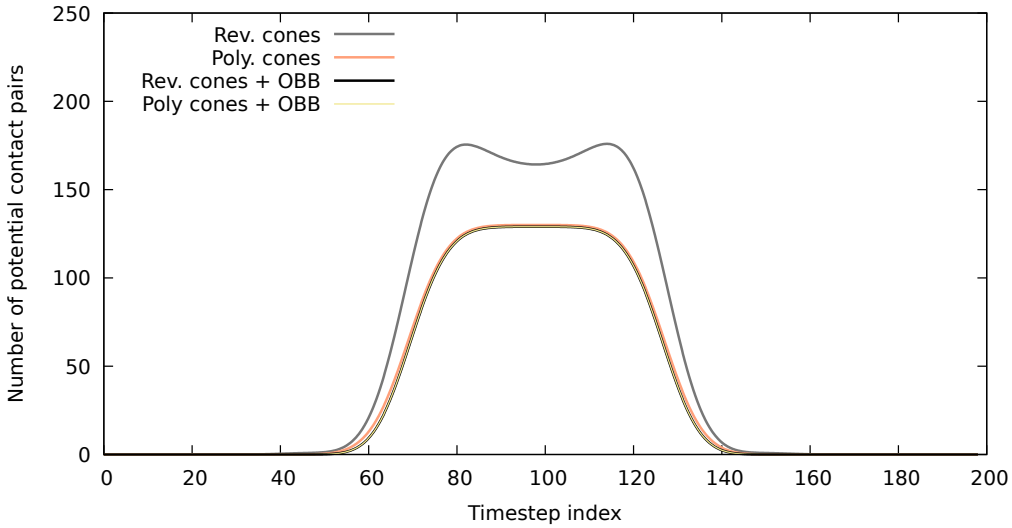


Figure 5.21: Number of curve segments pairs potentially containing LMD footpoints. Each curve corresponds to a different combination of bounding volumes for the culling test.

Finally, Figure 5.22 shows the overall CD times for each combination of bounding volumes. It is noteworthy that in this scenario the OBB culling test (including the computation of the OBBs themselves) is too costly to be beneficial. Ignoring it in favor of the polyhedral-cone based orthogonality test alone is thus preferable since it is sufficiently tight to filter as many potential pairs as when it is combined with OBBs.

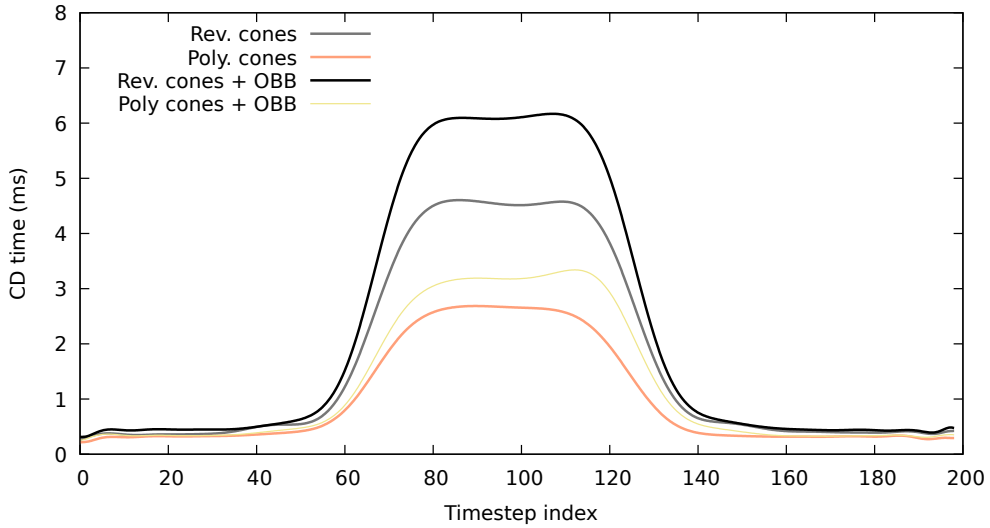


Figure 5.22: Self-collision detection times during the compression and extension of the spring. Each curve corresponds to a different combination of bounding volumes for the culling test.

Overall, it has been shown that the proposed polyhedral cones are significantly tighter than cones of revolution when it comes to identify potential pairs containing LMD footpoints along isolated curves. Moreover, the better reduction of potential pairs offered by polyhedral cones leads to lower computation times for the whole CD process because less executions of the iterative root-finding method, here a Newton-based approach as discussed in Section 4.3.1, will be necessary to find the exact location of LMDs.

5.4 Conclusion

The CD framework presented in this manuscript has been implemented as part of a real-time multibody dynamics simulation framework and tested on several scenarios motivated by industrial needs. The two first ones focused on insertion tasks with small mechanical clearances performed by teleoperated robotic components. The typical goal of such simulations is the training of human operators using a virtual scene before operating on the real system. This reduces the cost and complexity of setting up a mock-up as training environment, and avoids damaging any real equipment because of human errors naturally expected during their learning process. Additionally, a high level realism of the simulated interactions, while keeping the simulation real-time, is necessary to learn skills directly applicable on the real equipment.

Benchmark results show how each of the contributions helps reaching the real-time performance observed on the industrial scenarios. Moreover, these benchmarks show that performing CD on smooth BRep can provide better performances and accuracy than a tessellation-based, distance-based approach [90], especially when carrying out simulations of insertion tasks with small mechanical clearances. Indeed, the parallel framework performs more than twice as fast as a distance-based tessellation-based approach while producing a much smoother simulation because of the lack of any linear approximation of the geometric shapes. Moreover, the number of LMDs is reduced by an order of magnitude, which significantly lowers the computation times required for the resolution of the contact dynamics since this results in much less constraints to be solved. Overall, real-time performances are achieved while not suffering from any numerical artifact due to the rigid body geometry.

A third scenario has been presented to compare the proposed framework with pre-existing works [67] to demonstrate the efficiency of the proposed polyhedral tangent cones and polyhedral solution line cones presented in Chapter 3 compared to cones of revolution for self-collision detection of deformable Bézier curves. In the future, this will allow the dynamics engine to handle the simulation of smooth Bézier curves with finite elements relying on spatial discretization and Hermite shape functions. Indeed, Hermite curve being special cases of Bézier curves, this will allow the CD framework to operate on the same geometric representation than the finite element method.

Conclusion and perspectives

The work presented in this manuscript has been strongly motivated by the simulation of multi-body dynamics in the case of insertion tasks under small mechanical clearances. Such simulations involve sliding motions requiring the preservation of the smooth representations of the CAD geometric models. These representations are mandatory in order to generate accurate motions, i.e., without discontinuities of the contact normals. Indeed, relying on alternative representations like polyhedra introduces numerical errors and normal discontinuities that may prevent the simulation to be feasible at all without resorting to tessellations that are so thin that the CD and the resolution of the contact constraints becomes too computationally intensive to be performed in real-time. Some works toward the improvements of LMD computation on self-colliding curves were also proposed but are yet to be tested and exploited on industrial scenarios.

A first, complete, real-time framework for the computation of LMD or quasi-LMD between two smooth BRep models has been presented. This framework combines the adaptation of data structures widely used for CD between polyhedral models with a new orientation-based bounding volume and several analyses of the input models, in order to benefit from the specificities of smooth BRep representations. The key elements of the proposed framework are:

- The definition of **supermaximal features**, i.e., groups of edges and groups of surfaces that share the same underlying untrimmed geometry. This grouping prevents the CD framework from executing too many analytical LMD computation methods whenever several pairs of trimmed canonical features would involve the same underlying (untrimmed) geometrical shapes;
- The definition of **compatibility attributes** that avoids attempting to compute LMD between pairs of surfaces that never can contain any LMD because of their respective curvature distributions. In particular, it is proven than one shape exclusively composed of points with negative principal curvatures never can contain an LMD with another surface composed exclusively of points with at least one non-positive principal curvature. As part of future works, this analysis could be refined to derive culling tests based on curvature bounds of two surface areas. Such tests will allow the design of curvature-based bounding volumes that would elegantly complete the family of bounding volumes currently composed of spatial bounding volumes and orientation-based bounding volumes;
- The active use of **bounding volume hierarchies** for LMD and quasi-LMD computation between complete smooth BRep models derived from CAD models. This hierarchy relies on a specific combination of bounding volumes: an OBB and a normal cone for BRep feature areas, and an OBB and a tangent cone for deformable curves. These bounding volumes are tight, especially because all feature areas are split into almost-flat areas, i.e., such that any two such areas can be assumed to contain at most one LMD;
- The definition of a new bounding volume, the **polyhedral cone** (polyhedral normal cone, polyhedral tangent cone, and polyhedral solution line cone) that can be used to bound

LMD directions. This orientation-based bounding volume is much tighter than the only existing alternative, i.e., the cone of revolution [67, 70]. Our experiments show that using this bounding volume helps culling out more pairs of feature areas that cannot contain any LMD. A method for taking **temporal coherence** into account to accelerate the culling tests involving two polyhedral cones is also provided;

- The proposition of an **hybrid method** for the computation of LMD between **two deformable curves**. This combines curve subdivisions performed at runtime with a small pre-computed BVH updated whenever the curve is deformed. The use of polyhedral tangent cones and polyhedral solution line cones improves significantly the efficiency of the culling test compared to approaches based on cones of revolution [67, 70];

- The proposed CD pipeline has been completely **parallelized**. In particular, the load balance between the different stages of the algorithm motivates the separation of three distinct phases of the CD pipeline, which are parallelized individually. Most of the gains come from the parallelization of the last stage, i.e., the execution of iterative (or analytical) methods to locate exactly the LMD footpoints.

Overall the simulation of insertions with small mechanical clearances using smooth representations is made possible in **real time** by the framework described in this manuscript.

We presented the design of our sequential CD pipeline including the identification of supermaximal features and the exploitation of temporal coherence in Crozet et al. [32]. Moreover, the curvature-based compatibility attributes and our new orientation-based bounding volume, the polyhedral normal cone, have been presented in Crozet et al. [33]. Our parallel pipeline, the polyhedral tangent cones and polyhedral solution line cones, and our hybrid method for handling deformable curves have not been published outside of this manuscript yet. Our CD framework has been integrated completely into the industrial-grade multibody dynamics engine XDE [91], making it the first real-time oriented physics engine with such a feature reaching this level computational of performance.

Several improvements can be made in order to further improve the performance and accuracy of the multibody dynamics simulation of complex industrial scenarios:

- The Section 2.2 derived curvature-based compatibility conditions that can be used to determine if two surfaces can result in any LMDs. However, the Equations (2.7) and (2.9) could be further exploited in order to design what could be called *curvature-based culling tests*. For example, given bounds on the distance between two surface areas and tight bounds on their principal curvatures, it may be possible exploit Equations (2.7) and (2.9) to determine if they cannot contain any LMD. However, some preliminary works showed that bounding the distance, e.g., using bounding volumes, proved either too inaccurate or too expensive in practice to be of any use to set up such an extended curvature-based test. Further works may lead to better results following these proposals.

- Our framework strongly rely on the subdivision into almost flat feature areas for computing LMD between free-form curves and surfaces. As discussed in Section 2.4.1, this flatness condition originates from the need to ensure that at most one LMD exists between two pairs of feature. While this works well in practice, this is not a robust guarantee. Therefore, increased robustness could be achieved by designing tests of uniqueness of LMD between two feature areas that are both robust and cheap.

- A method for limiting the number of sides of a polyhedral normal cone was presented in Section 4.2.2. However, the proposed approach was a brute-force algorithms that computes some of the possible results and selects the best one. The efficient computation of the smallest four-sided polyhedral cone bounding a n-sided polyhedral cone is not straightforward but could further improve the tightness of those polyhedral normal cones. This would make the culling test even more discriminative.

- Some preliminary works have been presented in Section 4.3.2 for the identification and handling of conformal contacts. However, only the special case of conformal contacts between a surface and a curve located at the intersection of two canonical surfaces has been studied. Configurations like 1-dimensional conformal contacts between two surfaces are yet to be detected robustly. In addition, besides the geometric aspect of the detection of conformal contact configurations, we stress that their accurate modeling as part of a dynamics engine based on a time-stepping scheme is still an open problem given the current limitation of existing geometric contact models. Indeed, conformal contacts are prone to generate singularities on the feasible space $\mathcal{C}$ that may no longer be manifold;

- On the one hand, computing LMDs on polyhedral approximations allow algorithms to generate contact constraints that correspond to an approximation of the feasible space boundary $\partial\mathcal{C}$. On the other hand, computing LMDs on smooth BRep models generates much less constraints to represent $\partial\mathcal{C}$ exactly while those constraints contain non-linearities due to the curvature of the surfaces and curves they involve. Such non-linearities allow configurations where the LMD footpoints may move quickly even if the simulated bodies move only slightly. This causes stability problems to constraint solvers that systematically linearise the contact constraints. Unfortunately, most existing real-time dynamics engines do so. Thus, in practice, the non-linear non-penetration constraints defined from the smooth LMD functions cannot be enforced properly, leading to penetrations. One way of reducing those penetrations is to lower the time step. However, it is necessary to switch to a non-linear constraint solver as in the Non-Linear Non-Smooth Contact Dynamics (non-linear NSCD) method described by Acary et al. [4] if the prevention of penetrations is essential. The actual implementation of such a method requires further work from the CD standpoint because the LMD footpoints must be tracked while the solver iteratively applies virtual displacements to the solids in order to solve the non-linear constraints. Such tracking, while theoretically resolved by contact kinematics equations [93, 139] for individual curves and surfaces, can be hard in practice when the LMD footpoints may move from one feature to another adjacent one;

- Several industrial scenarios involve manipulations that combine functional contacts that need to be simulated with high accuracy on the one hand, and non-functional contacts which are transitory but necessary to switch from one task to another. For example, the insertion task presented in Section 5.1 is actually a subset of a much more complex scene and scenario where the teleoperated robot arm has to perform several insertions on various holes of the receptacle. Accidental contact during the transition between one insertion and another are not functional contacts and thus do not necessitate a smooth representation of the geometric models to be simulated with a satisfying level of accuracy. Therefore, the need to mix several geometric representations could be anticipated by, e.g., allowing fast CD on polyhedra for non-functional contacts, while the proposed method is applied using a smooth BRep model in configurations of functional contacts. Such a hybrid approach could offer the user the best of those two representations both in terms of performances and accuracy.

- Finally, given the current level of performance of our CD framework, we can expect to be able to run haptic simulations on high-end machines. The use of smooth BReps would significantly improve the user-experience as the continuity of the contact normals would generate a realistic feeling of sliding motion when the user manipulates a curved object.

While the various improvements presented above have yet to be developed, the previous chapters have shown that some industrial use-cases requiring a high level of accuracy can already be processed in real-time using our CD framework.

Bibliography

[1] Intel threading building blocks. https://www.threadingbuildingblocks.org/.

[2] Abel N.H. Beweis der unmöglichkeit, algebraische gleichungen von höheren graden als dem vierten allgemein aufzulösen. Journal für die reine und angewandte Mathematik, 1:65–84, 1826.

[3] Acary Vincent. Projected event-capturing time-stepping schemes for nonsmooth mechanical systems with unilateral contact and Coulomb's friction. Computer Methods in Applied Mechanics and Engineering, 256:224–250, apr 2013.

[4] Acary Vincent and Brogliato Bernard. Numerical Methods for Nonsmooth Dynamical Systems. Springer, 2008.

[5] Alciatore David G. and Miranda Rick. A winding number and point-in-polygon algorithm. Department of Mechanical Engineering Colorado State University, Fort Collins, CO, 1995.

[6] Algoryx . AGX Dynamics.

[7] Andersen K A. A survey of algorithms for construction of optimal Heterogeneous Bounding Volume Hierarchies. Technical report, University of Copenhagen, 2006.

[8] Anitescu Mihai and Hart Gary D. A constraint-stabilized time-stepping approach for rigid multibody dynamics with joints, contact and friction. International Journal for Numerical Methods in Engineering, 60(14):2335–2371, 2004.

[9] Baciu G., Wong Wingo Sai-Keung Wong Wingo Sai-Keung, and Sun Hanqiu Sun Hanqiu. RECODE: an image-based collision detection algorithm. Proceedings Pacific Graphics '98. Sixth Pacific Conference on Computer Graphics and Applications (Cat. No.98EX208), 1998.

[10] Ballard P. The dynamics of discrete mechanical systems with perfect unilateral constraints. Archive for Rational Mechanics and Analysis, 154(3):199–274, 2000.

[11] Baraff David. Dynamic Simulation of Non-Penetrating Rigid Bodies. PhD thesis, Cornell University, 1992.

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

[13] Barbic J. and James D.L. Six-DoF Haptic Rendering of Contact Between Geometrically Complex Reduced Deformable Models. IEEE Transactions on Haptics, 1(1):39–52, 2008.

[14] Baumgarte J. Stabilization of constraints and integrals of motion in dynamical systems. Computer Methods in Applied Mechanics and Engineering, 1:1–16, 1972.

[15] Bertsekas Dimitri P. Convex Analysis and Optimization. Athena Scientific, 2003.

[16] Bertsekas DP, Nedić A, and Ozdaglar AE. Convex analysis and optimization. 2003.

[17] Bhushan Bharat. Contact mechanics of rough surfaces in tribology: multiple asperity contact. Tribology Letters, 4:1–35, 1998.

[18] Boeing Adrian and Bräunl Thomas. Evaluation of real-time physics simulation systems. In Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia - GRAPHITE '07, volume 1, page 281, New York, New York, USA, 2007. ACM Press.

[19] Boussuge Flavien, Léon Jean-Claude, Hahmann Stéfanie, and Fine Lionel. Extraction of generative processes from B-Rep shapes and application to idealization transformations. Computer-Aided Design, 46:79–89, 2014.

[20] Brogliato Bernard and Thibault Lionel. Well-posedness results for non-autonomous dissipative complementarity systems. (June), 2006.

[21] Brüls Olivier, Acary Vincent, and Cardona Alberto. Simultaneous enforcement of constraints at position and velocity levels in the nonsmooth generalized-alpha scheme. Computer Methods in Applied Mechanics and Engineering, 281(1):131–161, nov 2014.

[22] Cameron S A and Culley R K. Determining the Minimum Translational Distance between Two Convex Polyhedra. Proceedings of the IEEE International Conference on Robotics and Automation(ICRA), 3:591–596, 1986.

[23] Carmo MP Do. Differential geometry of curves and surfaces. 1976.

[24] Chang Jung-Woo, Choi Yi-King, Kim Myung-Soo, and Wang Wenping. Computation of the minimum distance between two Bézier curves/surfaces. Computers & Graphics, 35(3):677–684, jun 2011.

[25] Chen Xiao-Diao, Su Hua, Yong Jun-Hai, Paul Jean-Claude, and Sun Jia-Guang. A counterexample on point inversion and projection for NURBS curve. Computer Aided Geometric Design, 24(5):302, jul 2007.

[26] Chen Xiao-Diao, Yong Jun-Hai, Wang Guozhao, Paul Jean-Claude, and Xu Gang. Computing the minimum distance between a point and a NURBS curve. Computer-Aided Design, 40(10-11):1051–1054, oct 2008.

[27] Chen Xiao-Diao, Yong Jun-Hai, Zheng Guo-Qin, Paul Jean-Claude, and Sun Jia-Guang. Computing minimum distance between two implicit algebraic surfaces. Computer-Aided Design, 38(10):1053–1061, oct 2006.

[28] Chong Jin Ong and Gilbert E.G. Growth distances: new measures for object separation and penetration. IEEE Transactions on Robotics and Automation, 12(6):888–903, 1996.

[29] Chou Wusheng and Xiao Jing. Real-time and Accurate Multiple Contact Detection between General Curved Objects. In International Conference on Intelligent Robots and Systems, pages 556–561. IEEE, oct 2006.

[30] CMLabs . Vortex Studio Simulation Platform.

[31] Coumans Erwin. Bullet physics simulation. In ACM SIGGRAPH 2015 Courses on - SIGGRAPH '15, page 1, New York, New York, USA, 2015. ACM Press.

[32] Crozet Sébastien, Léon Jean-Claude, and Merlhiot Xavier. Fast Computation of Contact Points for Robotic Simulations Based on CAD Models Without Tessellation. In 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 937–944, Daejeon, Korea, 2016. IEEE.

[33] Crozet Sébastien, Léon Jean-Claude, and Merlhiot Xavier. Fast and Accurate Local Minimal Distance Computation between Industrial CAD Models for Interactive Dynamics Simulations. In CAD'17, pages 191–195, Okayama, aug 2017. CAD Solutions LLC.

[34] Diktas Engin and Sahiner Ali Vahit. Distance computation using OBB-trees. In 2009 24th International Symposium on Computer and Information Sciences, pages 426–431. IEEE, sep 2009.

[35] Dobkin David, Hershberger John, Kirkpatrick David, and Suri Subhash. Computing the intersection-depth of polyhedra. Algorithmica, 9(6):518–533, Jun 1993.

[36] Dyllong Eva and Luther Wolfram. Distance calculation between a point and a NURBS surface. 2000.

[37] Dzonou Raoul and Monteiro Marques Manuel D.P. A sweeping process approach to inelastic contact problems with general inertia operators. European Journal of Mechanics, A/Solids, 26(3):474–490, 2007.

[38] ElBadrawy Asma A., Hemayed Elsayed E., and Fayek Magda B. Rapid collision detection for deformable objects using inclusion-fields applied to cloth simulation. Journal of Advanced Research, 3(3):245–252, 2012.

[39] Elber Gershon and Kim Myung-Soo. Geometric constraint solver using multivariate rational spline functions. In Proceedings of the sixth ACM symposium on Solid modeling and applications - SMA '01, pages 1–10, New York, New York, USA, 2001. ACM Press.

[40] Ericson Christer. Real-Time Collision Detection. CRC Press, Inc., Boca Raton, FL, USA, 2004.

[41] Erleben Kenny, Sporring Jon, Henriksen Knud, and Dohlman Kenrik. Physics-based Animation (Graphics Series). Charles River Media, Inc., Rockland, MA, USA, 2005.

[42] Farin Gerald. Curves and Surfaces for Computer-Aided Geometric Design (Third Edition). Academic Press, Boston, third edition edition, 1993.

[43] Farin Gerald. The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling. Elsevier, 2002.

[44] Featherstone Roy. Rigid Body Dynamics Algorithms. 2008.

[45] Freeman H and Shapira R. Determining the minimum-area encasing rectangle for an arbitrary closed curve. Communications of the ACM, 18(7):409–413, jul 1975.

[46] Gilbert E., Johnson D., and Keerthi S. A fast procedure for computing the distance between complex objects in three space. In Proceedings. 1987 IEEE International Conference on Robotics and Automation, volume 4, pages 1883–1889. Institute of Electrical and Electronics Engineers, 1987.

[47] Gilbert E.G. and Chong Jin Ong . New distances for the separation and penetration of objects. In Proceedings of the 1994 IEEE International Conference on Robotics and Automation, volume 1, pages 579–586. IEEE Comput. Soc. Press, 1994.

[48] Gilbert E.G. and Foo C.P. Computing the distance between smooth objects in three dimensional space. In Proceedings, 1989 International Conference on Robotics and Automation, pages 158–163. IEEE Comput. Soc. Press, 1989.

[49] Gilbert EG and Foo CP. Computing the distance between general convex objects in three-dimensional space. IEEE Transactions on Robotics and Automation, 6(1):53–61, 1990.

[50] Glocker Christoph. Impacts with global dissipation index at reentrant corners. Contact Mechanics International Symposium, (June 2001):45–52, 2002.

[51] Goldsmith Jeffrey and Salmon John. Automatic Creation of Object Hierarchies for Ray Tracing. IEEE Computer Graphics and Applications, 7(5):14–20, may 1987.

[52] Gottschalk S, Lin M. C., and Manocha D. OBBTree: A Hierarchical Structure for Rapid Interference Detection. In Proceedings of the 23rd annual conference on Computer graphics and interactive techniques - SIGGRAPH '96, number 8920219, pages 171–180, New York, New York, USA, 1996. ACM Press.

[53] Greenwood J. A. and Williamson J. B. P. Contact of Nominally Flat Surfaces. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 295(1442):300–319, 1966.

[54] Hachani Maha and Fourment Lionel. 3D Contact Smoothing Method Based on Quasi-C1 Interpolation. In Trends in Computational Contact Mechanics, pages 23–40. 2011.

[55] Hairer Ernst and Wanner Gerhard. Solving Ordinary Differential Equations I, volume 8 of Springer Series in Computational Mathematics. Springer Berlin Heidelberg, Berlin, Heidelberg, 1993.

[56] Hairer Ernst and Wanner Gerhard. Solving Ordinary Differential Equations II, volume 14 of Springer Series in Computational Mathematics. Springer Berlin Heidelberg, Berlin, Heidelberg, feb 1996.

[57] Hansen E.R. and Greenberg R.I. An interval Newton method. Applied Mathematics and Computation, 12(2-3):89–98, may 1983.

[58] Hartmann Erich. On the curvature of curves and surfaces defined by normalforms. Computer Aided Geometric Design, 16(5):355–376, jun 1999.

[59] Heidelberger Bruno, Teschner M., and Gross M. Detection of collisions and self-collisions using image-space techniques. Journal of WSCG, 12(3):145–152, 2004.

[60] Henshaw W. D. An Algorithm for Projecting Points onto a Patched CAD Model. Engineering with Computers, 18(3):265–273, oct 2002.

[61] Hoschek Josef and Lasser Dieter. Fundamentals of computer-aided geometric design. Peters, Wellesley, Mass, 1993.

[62] Hu Shi-Min and Wallner Johannes. A second order algorithm for orthogonal projection onto curves and surfaces. Computer Aided Geometric Design, 22(3):251–260, mar 2005.

[63] Hubbard Philip M. Interactive collision detection. In Proceedings of IEEE Symposium on Research Frontiers in Virtual Reality, pages 24–31, 1993.

[64] Hubbard Philip M. Approximating polyhedra with spheres for time-critical collision detection. ACM Transactions on Graphics, 15(3):179–210, jul 1996.

[65] Jean M. The non-smooth contact dynamics method. Computer Methods in Applied Mechanics and Engineering, 177(3-4):235–257, jul 1999.

[66] Jiménez P., Thomas F., and Torras C. 3D collision detection: A survey. Computers and Graphics, 25(2):269–285, 2001.

[67] Johnson David E. and Cohen Elaine. Spatialized normal cone hierarchies. In Proceedings of the 2001 symposium on Interactive 3D graphics - SI3D '01, pages 129–134, New York, New York, USA, 2001. ACM Press.

[68] Johnson DE. Minimum distance queries for haptic rendering. Zhurnal Eksperimental'noi i Teoreticheskoi Fiziki, (May 2005), 2005.

[69] Johnson D.E. and Cohen E. Bound coherence for minimum distance computations. In Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C), volume 3, pages 1843–1848. IEEE, 1999.

[70] Johnson DE and Cohen E. Distance extrema for spline models using tangent cones. Graphics Interface, 2005.

[71] Kim Deok-Soo, Papalambros Panos Y., and Woo Tony C. Tangent, normal, and visibility cones on Bézier surfaces. Computer Aided Geometric Design, 12(3):305–320, may 1995.

[72] Kim Y.J., Lin M.C., and Manocha D. DEEP: dual-space expansion for estimating penetration depth between convex polytopes. In Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292), volume 1, pages 921–926. IEEE.

[73] Kim Young J., Lin Ming C., and Manocha Dinesh. Fast Penetration Depth Estimation Using Rasterization Hardware and Hierarchical Refinement. In Springer Tracts in Advanced Robotics, volume 7 STAR, pages 505–521. 2004.

[74] Kim Y.S. Recognition of form features using convex decomposition. Computer-Aided Design, 24(9):461–476, sep 1992.

[75] Klosowski J.T., Held M., Mitchell J.S.B., Sowizral H., and Zikan K. Efficient collision detection using bounding volume hierarchies of k-DOPs. IEEE Transactions on Visualization and Computer Graphics, 4(1):21–36, 1998.

[76] Knott Dave and Pai D.K. CInDeR: Collision and interference detection in real-time using graphics hardware. Computer Graphics Forum, 2003.

[77] Kockara S, Halic T, Iqbal K, Bayrak C., and Rowe Richard. Collision detection: A survey. In International Conference on Systems, Man and Cybernetics, pages 4046–4051. IEEE, oct 2007.

[78] Krishnamurthy Adarsh, McMains Sara, and Haller Kirk. GPU-accelerated minimum distance and clearance queries. IEEE transactions on visualization and computer graphics, 17(6):729–42, jun 2011.

[79] Krishnan S., Gopi M., Lin M., Manocha D., and Pattekar A. Rapid and Accurate Contact Determination between Spline Models using ShellTrees. Computer Graphics Forum, 17(3):315–326, aug 1998.

[80] Kry Paul G. and Pai Dinesh K. Continuous contact simulation for smooth surfaces. ACM Transactions on Graphics, 22(1):106–129, jan 2003.

[81] Kyung-Ah Sohn , Juttler B., Myung-Soo Kim , and Wenping Wang . Computing distances between surfaces using line geometry. In 10th Pacific Conference on Computer Graphics and Applications, 2002. Proceedings., pages 236–245. IEEE Comput. Soc, 2002.

[82] Li K., Foucault G., Léon J.-C., and Trlin M. Fast global and partial reflective symmetry analyses using boundary surfaces of mechanical components. Computer-Aided Design, 53:70–89, aug 2014.

[83] Lien Jyh-Ming and Amato Nancy M. Approximate convex decomposition of polyhedra and its applications. Computer Aided Geometric Design, 25(7):503–522, oct 2008.

[84] Lin M and Gottschalk Stefan. Collision detection between geometric models: A survey. In Proc. of IMA conference on mathematics of surfaces, pages 37–56, 1998.

[85] Lin Ming C. and Manocha Dinesh. Fast interference detection between geometric models. The Visual Computer, 11(10):542–561, 1995.

[86] Liu Xiao-Ming, Yang Lei, Yong Jun-Hai, Gu He-Jin, and Sun Jia-Guang. A torus patch approximation approach for point projection on surfaces. Computer Aided Geometric Design, 26(5):593–598, jun 2009.

[87] Ma Ying Liang and Hewitt W.T. Point inversion and projection for NURBS curve and surface: Control polygon approach. Computer Aided Geometric Design, 20(2):79–99, may 2003.

[88] McNeely William A, Puterbaugh Kevin D, and Troy James J. Six degree-of-freedom haptic rendering using voxel sampling. In ACM SIGGRAPH 2005 Courses on - SIGGRAPH '05, page 42, New York, New York, USA, 2005. ACM Press.

[89] McNeely William A, Puterbaugh Kevin D, and Troy James J. Six degree-of-freedom haptic rendering using voxel sampling. In ACM SIGGRAPH 2005 Courses on - SIGGRAPH '05, page 42, New York, New York, USA, 2005. ACM Press.

[90] Merlhiot Xavier. A robust, efficient and time-stepping compatible collision detection method for non-smooth contact between rigid bodies of arbitrary shape. In Multibody Dynamics, Eccomas Thematic Conference, 2007.

[91] Merlhiot Xavier, Garrec Jérémie Le, Saupin Guillaume, and Andriot Claude. The XDE mechanical kernel: Efficient and robust simulation of multibody dynamics with intermittent nonsmooth contacts. In International Conference on Multibody System Dynamics, pages 5–6, 2012.

[92] Mirtich BV. Impulse-based dynamic simulation of rigid body systems. PhD thesis, University of California, Berkeley, 1996.

[93] Montana D. J. The Kinematics of Contact and Grasp. The International Journal of Robotics Research, 7(3):17–32, jun 1988.

[94] Moreau J. J. Unilateral Contact and Dry Friction in Finite Freedom Dynamics. In Nonsmooth Mechanics and Applications, pages 1–82. Springer Vienna, Vienna, 1988.

[95] Muller G., Schafer S., and Fellner D.W. Automatic creation of object hierarchies for radiosity clustering. In Proceedings. Seventh Pacific Conference on Computer Graphics and Applications (Cat. No.PR00293), volume 7, pages 21–29,. IEEE Comput. Soc, may 1987.

[96] Munkres James R. Elements of Algebraic Topology. Addison-Wesley, 1993.

[97] Myszkowski Karol, Okunev Oleg G, and Kunii Tosiyasu L. Fast collision detection between complex solids using rasterizing graphics hardware empu er. pages 497–511, 1995.

[98] Nelson Donald D., Johnson David E., and Cohen Elaine. Haptic rendering of surface-to-surface sculpted model interaction. In ACM SIGGRAPH 2005 Courses on - SIGGRAPH '05, page 97, New York, New York, USA, 2005. ACM Press.

[99] Neto D.M., Oliveira M.C., Menezes L.F., and Alves J.L. A contact smoothing method for arbitrary surface meshes using Nagata patches. Computer Methods in Applied Mechanics and Engineering, 299:283–315, feb 2016.

[100] Nishita Tomoyuki, Sederberg Thomas W., and Kakimoto Masanori. Ray tracing trimmed rational surface patches. ACM SIGGRAPH Computer Graphics, 24(4):337–345, sep 1990.

[101] NVIDIA . NVIDIA physX SDK.

[102] Oh Young Taek, Kim Yong Joon, Lee Jieun, Kim Myung Soo, and Elber Gershon. Efficient point-projection to freeform curves and surfaces. Computer Aided Geometric Design, 29(5):242–254, jun 2012.

[103] Page Francis and Guibault Frangois. Collision detection algorithm for NURBS surfaces in interactive applications. In Canadian Conference on Electrical and Computer Engineering. Toward a Caring and Humane Technology, volume 2, pages 1417–1420. IEEE, 2003.

[104] Palmer I. J. and Grimsdale R. L. Collision Detection for Animation using Sphere-Trees. Computer Graphics Forum, 14(2):105–116, may 1995.

[105] Pfeiffer Friedrich and Glocker Christoph. Multibody Dynamics with Unilateral Contacts. Wiley-VCH Verlag GmbH, Weinheim, Germany, aug 1996.

[106] Piegl LA and Tiller W. Parametrization for surface fitting in reverse engineering. Computer-Aided Design, 33(8):593–603, jul 2001.

[107] Piegl Les and Tiller Wayne. The NURBS Book. Monographs in Visual Communication. Springer Berlin Heidelberg, Berlin, Heidelberg, 1997.

[108] Polak Elijah. Optimization Algorithms and Consistent Approximations, volume 124 of Applied Mathematical Sciences. Springer New York, New York, NY, 1997.

[109] Puso Michael Anthony and Laursen Tod A. A 3D contact smoothing method using Gregory patches. International Journal for Numerical Methods in Engineering, 54(8):1161–1194, jul 2002.

[110] Qian Xiaoxiang, Yuan Huina, Zhou Mozhen, and Zhang Bingyin. A general 3D contact smoothing method based on radial point interpolation. Journal of Computational and Applied Mathematics, 257:1–13, feb 2014.

[111] Quinlan S. Efficient distance computation between non-convex objects. In Proceedings of the 1994 IEEE International Conference on Robotics and Automation, pages 3324–3329. IEEE Comput. Soc. Press.

[112] Redon Stephane and Lin Ming C. A Fast Method for Local Penetration Depth Computation. Journal of Graphics, GPU, and Game Tools, 11(2):37–50, jan 2006.

[113] Reggiani M., Mazzoli M., and Caselli S. An experimental evaluation of collision detection packages for robot motion planning. In International Conference on Intelligent Robots and System, volume 3, pages 2329–2334. IEEE, 2002.

[114] Rohatgi Pradeep K., Tabandeh-Khorshid Meysam, Omrani Emad, Lovell Michael R., and Menezes Pradeep L. Tribology for Scientists and Engineers. 2013.

[115] Sagardia Mikel and Hulin Thomas. Evaluation of a penalty and a constraint-based haptic rendering algorithm with different haptic interfaces and stiffness values. Proceedings - IEEE Virtual Reality, pages 64–73, 2017.

[116] Sakurai Hiroshi and Chin Chia-Wei. Definition and Recognition of Volume Features for Process Planning. pages 65–80. 1994.

[117] Sakurai Hiroshi and Dave Parag. Volume decomposition and feature recognition, part II: curved objects. Computer-Aided Design, 28(6-7):519–537, jun 1996.

[118] Samuel N. M., Requicha Aristides A. G., and Elkind S. A. Methodology and Results of an Industrial Part Survey. Technical report, University of Rochester, 1979.

[119] Schollmeyer Andre and Fröhlich Bernd. Direct trimming of NURBS surfaces on the GPU. ACM Transactions on Graphics, 28(3):1, 2009.

[120] Selimovic Ilijas. Improved algorithms for the projection of points on NURBS curves and surfaces. Computer Aided Geometric Design, 23(5):439–445, jul 2006.

[121] Seong Joon-Kyung, Johnson David E, and Cohen Elaine. A higher dimensional formulation for robust and interactive distance queries. In Proceedings of the 2006 ACM symposium on Solid and physical modeling - SPM '06, page 197, New York, New York, USA, 2006. ACM Press.

[122] Signorini Antonio. Sopra alcune questioni di elastostatica. Atti della Societa Italiana per il Progresso delle Scienze, 1933.

[123] Smith Russell. The Open Dynamics Engine.

[124] Snyder John M. An interactive tool for placing curved surfaces without interpenetration. In Proceedings of the 22nd annual conference on Computer graphics and interactive techniques - SIGGRAPH '95, pages 209–218, New York, New York, USA, 1995. ACM Press.

[125] Snyder John M., Woodbury Adam R., Fleischer Kurt, Currin Bena, and Barr Alan H. Interval methods for multi-point collisions between time-dependent curved surfaces. In Proceedings of the 20th annual conference on Computer graphics and interactive techniques - SIGGRuAPH '93, pages 321–334, New York, New York, USA, 1993. ACM Press.

[126] Song Hai-Chuan, Xu Xin, Shi Kan-Le, and Yong Jun-Hai. Projecting points onto planar parametric curves by local biarc approximation. Computers & Graphics, 38:183–190, feb 2014.

[127] Strobach Peter. The fast quartic solver. Journal of Computational and Applied Mathematics, 234(10):3007–3024, 2010.

[128] Stroud Ian. Boundary Representation Modelling Techniques. Springer London, London, 2006.

[129] Studer Christian. Numerics of Unilateral Contacts and Friction, volume 47 of Lecture Notes in Applied and Computational Mechanics. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[130] Tang Min, Manocha Dinesh, and Tong Ruofeng. MCCD: Multi-core collision detection between deformable models using front-based decomposition. Graphical Models, 72(2):7–23, mar 2010.

[131] Teschner M and Kimmerle S. Collision Detection for Deformable Objects. Computer Graphics Forum, 24(x):61 – 81, 2005.

[132] Thomas F, Turnbull Colin, Ros L., and Cameron S. Computing signed distances between free-form objects. In International Conference on Robotics and Automation, volume 4, pages 3713–3718. IEEE, 2000.

[133] Thompson Thomas V. and Cohen Elaine. Direct haptic rendering of complex trimmed NURBS models. In ACM SIGGRAPH 2005 Courses on - SIGGRAPH '05, page 89, New York, New York, USA, 2005. ACM Press.

[134] Thompson Thomas V., Johnson David E., and Cohen Elaine. Direct haptic rendering of sculptured models. In Proceedings of the 1997 symposium on Interactive 3D graphics - SI3D '97, number Figure 1, pages 167–176, New York, New York, USA, 1997. ACM Press.

[135] Toussaint Godfried. Solving Geometric Problems with the Rotating Calipers. In IEEE Melecon83, number May, pages 1–8, 1983.

[136] Tropp Oren, Tal Ayellet, Shimshoni Ilan, and Dobkin David P. Temporal Coherence in Bounding Volume Hierarchies for Collision Detection. International Journal of Shape Modeling, 12(02):159–178, dec 2006.

[137] Turnbull C. and Cameron S. Computing distances between NURBS-defined convex objects. In Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146), volume 4, pages 3685–3690. IEEE, 1998.

[138] van den Bergen Gino. Collision Detection in Interactive 3D Environments. Elsevier, 2003.

[139] Visser M., Stramigioli S., and Heemskerk C. Screw bondgraph contact dynamics. In IEEE/RSJ International Conference on Intelligent Robots and System, volume 3, pages 2239–2244. IEEE, 2002.

[140] Wald Ingo. On fast Construction of SAH-based Bounding Volume Hierarchies. In 2007 IEEE Symposium on Interactive Ray Tracing, volume 1, pages 33–40. IEEE, sep 2007.

[141] Weller Rene and Zachmann Gabriel. A unified approach for physically-based simulations and haptic rendering. Proceedings of the 2009 ACM SIGGRAPH Symposium on Video Games, Sandbox '09, c:151–160, 2009.

[142] Yao Jen-Chih. Variational Inequalities with Generalized Monotone Operators. Mathematics of Operations Research, 19(3):691–705, aug 1994.

[143] Zhang Liangjun, Kim Young J., Varadhan Gokul, and Manocha Dinesh. Generalized penetration depth computation. Computer-Aided Design, 39(8):625–638, aug 2007.

[144] Zhang Xinyu, Kim Young J., and Manocha Dinesh. Continuous penetration depth. Computer Aided Design, 46(1):3–13, 2014.

[145] Zhao Wei and Lan Ying. A Fast Collision Detection Algorithm Based on Distance Calculations between NURBS Surfaces. In 2012 International Conference on Computer Science and Electronics Engineering, pages 534–537. IEEE, mar 2012.

[146] Zhou Jingfang, Sherbrooke Evan C., and Patrikalakis Nicholas M. Computation of stationary points of distance functions. Engineering with Computers, 9(4):231–246, dec 1993.

[147] Zou Zhihua and Xiao Jing. Tracking minimum distances between curved objects with parametric surfaces in real time. In Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453), volume 3, pages 2692–2698. IEEE, 2003.