Constructions géométriques à précision fixée

Philippe Guigue

UNIVERSITÉ DE NICE-SOPHIA ANTIPOLIS
UFR Sciences

École doctorale STIC

THÈSE

pour obtenir le titre de

Docteur en Sciences

de l'UNIVERSITÉ de Nice-Sophia Antipolis

Spécialité

Informatique

présentée et soutenue par

Philippe GUIGUE

TITRE

Constructions géométriques à précision fixée

Thèse dirigée par Olivier DEVILLERS

soutenue le 5 Décembre 2003

Jury

Président	André GALLIGO	Professeur, UNSA
Rapporteurs	Jean-Michel MOREAU	Professeur, LIRIS
	Jean-Michel MULLER	Directeur de recherches, CNRS - LIP
Examinateurs	Olivier DEVILLERS	Directeur de recherches, INRIA
	André LIEUTIER	Consultant scientifique, DASSAULT
	Sylvain PION	Chargé de recherches, INRIA

Remerciements

Je voudrais tout d'abord remercier Olivier pour m'avoir accueilli au sein de l'équipe PRISME, pour m'avoir encadré et orienté pendant ces trois années ainsi que pour la confiance qu'il m'a témoignée.

Je remercie également André Galligo, Jean-Michel Moreau, Jean-Michel Muller, André Lieutier et Sylvain Pion pour avoir accepter d'être membre de mon jury et pour avoir pris le temps de se pencher sur mes travaux.

Merci également à l'ensemble des personnes de l'équipe PRISME/GEOMETRICA qui m'ont aidé de près ou de loin dans ce travail. Je tiens à remercier tout particulièrement Sylvain Pion pour ses commentaires et critiques toujours éclairés ainsi que Pierre Alliez pour avoir sacrifié de son temps à la relecture et à l'amélioration du manuscrit.

Un merci tout spécial à Agnès pour son aide et son soutien inestimable durant ces années passées ici.

Enfin, je tiens à remercier tous mes amis de longue date ainsi que toutes les personnes que j'ai eu le plaisir de rencontrer pendant cette thèse, dans le désordre : Monique Teillaud, Andreas Fabri, Katrin Dobrindt, Pierre-Marie Gandoin, Raphaëlle Chaine, David Cohen-Steiner, Thomas Lewiner, Jean-Marie Morvan, Frank Da, Dominique Amar, Fatma, Aurélie et Stéphane, Louaï, François et Darka, Anders et Mania, Christophe, Steve, Marie, Marc, Luca, ...

Je dédie cette thèse à mes grands-parents, mes parents, mon frère, et à *meinen Schatz* Julia.

Résumé

Les problèmes de robustesse liés à la substitution du calcul exact sur les réels par le calcul flottant approché sont souvent un obstacle à l'implantation pratique des algorithmes géométriques. Si l'adoption du paradigme exact apporte une solution satisfaisante à ce type de problèmes pour les algorithmes ayant un résultat purement combinatoire, cette solution ne permet cependant pas de résoudre en pratique le cas des algorithmes qui réutilisent voire cascadent la construction de nouveaux objets géométriques.

Cette thèse aborde le problème de l'arrondi sur la grille entière du résultat d'opérations booléennes sur des régions polygonales et propose plusieurs notions d'arrondi permettant de garantir certaines propriétés métriques et topologiques intéressantes entre le résultat exact et sa version arrondie telles que la garantie de relations d'inclusion et la préservation de la convexité du résultat. Nos méthodes sont basées sur l'utilisation de constructeurs élémentaires arrondis pour lesquels nous présentons également plusieurs algorithmes efficaces. Nous proposons enfin des tests rapides permettant la détection robuste d'intersection entre plusieurs types d'objets convexes dans le plan et dans l'espace. L'ensemble de ces solutions trouvent une application directe en CAO et en graphisme.

Abstract

Robustness problems resulting from the substitution of floating-point arithmetic for exact arithmetic on real numbers are often an obstacle to the practical implementation of geometric algorithms. For algorithms that are purely combinatorial, the use of the exact computation paradigm gives a satisfactory solution to this problem. However, this paradigm does not allow to practically solve the case of algorithms that reuse or cascade the construction of new geometric objects.

This thesis treats the problem of rounding the result of set operations on polygonal regions onto the integer grid. We propose several rounding modes that allow to guarantee some interesting metric and topological properties between the exact result and its rounded counterpart such as inclusion properties and convexity preservation. Our methods are based on the rounding of elementary geometric constructions, e.g. rounding a vertex of a convex polygon to its nearest interior grid point, for which we propose efficient algorithms. We finally present some fast overlap tests that allow to detect in a robust manner the intersection of several kinds of convex objects in the plane and in the three dimensional space. All methods have a direct application in domains such as CAD and computer graphics.

Table des matières

Introduction

Depuis sa création dans les années 1970, la géométrie algorithmique a contribué à la description et à l'analyse de nombreux algorithmes permettant de calculer ou manipuler des objets géométriques. Le passage de la description théorique de ces algorithmes à une implantation effective soulève toutefois des problèmes délicats qui constituent un obstacle majeur à l'exploitation pratique des résultats de la discipline. La source principale de ces difficultés peut être directement liée à l'utilisation d'hypothèses simplificatrices qui facilitent l'élaboration des algorithmes et la preuve de leur validité mais qui ne sont pas vérifiées en pratique par l'ordinateur. Parmi celles-ci, l'hypothèse selon laquelle les opérations arithmétiques sur les nombres réels peuvent être traitées de manière exacte par la machine est particulièrement importante. Un algorithme géométrique se fonde en effet sur l'évaluation de primitives numériques qui conditionnent son comportement et la qualité de son résultat. L'évaluation approchée (et donc non exacte) du résultat de ces calculs par l'arithmétique à précision fixée de l'ordinateur introduit potentiellement des incohérences dans le déroulement du programme et a par conséquent un impact crucial sur la fiabilité des implantations réelles des algorithmes théoriques.

On distingue classiquement deux types de primitives numériques dans un algorithme géométrique : les *prédicats* répondent à des questions géométriques élémentaires et retournent un résultat parmi un nombre fini de réponses (par exemple un point est *au dessus*, *sur* ou *en dessous* d'un segment). Ils sont associés aux décisions prises par l'algorithme. Les *constructeurs* effectuent une construction de base sur des objets géométriques élémentaires et retournent des quantités numériques (par exemple les coordonnées d'un point d'intersection). Ils permettent ainsi le calcul du plongement géométrique de nouveaux objets, et sont nécessaires dans certains cas à la production et à la représentation du résultat de l'algorithme.

Certains problèmes géométriques sont purement combinatoires (par exemple le calcul d'une triangulation), c'est-à-dire calculent une information de nature discrète. Leur résolution pratique ne nécessite par conséquent que l'évaluation de prédicats. Comme nous venons de le voir, les prédicats jouent un rôle critique car ils conditionnent le déroulement d'un programme et il est par

conséquent essentiel de les évaluer exactement. Ces dernières années de nombreux travaux ont contribué à la conception de méthodes logicielles de calcul exact ainsi qu'à la mise au point de techniques d'accélération qui permettent d'évaluer exactement ces prédicats sans trop pénaliser les temps de calcul. Cependant, ce type d'approche amène à utiliser des représentations bien définies des objets géométriques : typiquement les coordonnées des objets en entrée sont supposées être des entiers de taille fixée.

Le problème s'avère toutefois beaucoup plus délicat pour les algorithmes qui construisent de nouveaux objets géométriques, c'est-à-dire calculent le plongement géométrique de nouveaux objets en faisant appel à des constructeurs. Un constructeur ne retourne en effet pas un résultat discret comme le prédicat mais des quantités numériques. Dans le cas d'un constructeur retournant un résultat rationnel, la complexité de ces quantités, c'est-à-dire leur taille en nombre de bits, est par exemple directement proportionnelle à celle de ses arguments.

Lorsque l'on construit un nouvel objet on se trouve classiquement devant l'alternative suivante. La première possibilité consiste à utiliser l'arithmétique arrondie de l'ordinateur. On conserve ainsi une représentation usuelle de la construction mais ceci au risque de perdre certaines propriétés caractéristiques de la structure exacte. Par exemple, un polygone peut être simple et convexe par nature et perdre ces propriétés dans un arrondi sans précautions, alors que l'on veut utiliser ce polygone comme entrée d'un autre algorithme qui va supposer ces propriétés vérifiées.

La deuxième possibilité consiste à adopter une arithmétique exacte et ainsi à garantir la cohérence et les propriétés géométriques de l'objet construit au prix d'une représentation très coûteuse de l'objet. La représentation exacte du résultat du constructeur implique, en effet, soit une représentation implicite de l'objet construit, souvent lourde et difficile à exploiter et qui complique considérablement les algorithmes qui manipulent ces objets, soit l'utilisation d'un type de nombres exact, qui provoque une croissance significative du nombre de bits nécessaires à la représentation et qui a pour effet de rendre très rapidement impraticables les algorithmes cascadant ces constructions, c'est-à-dire réutilisant en entrée le résultat d'une construction retournée par un autre algorithme. Ce type de problème est particulièrement important en CAO et l'obtention de méthodes d'arrondi de structures géométriques limitant les exigences arithmétiques de l'implantation robuste d'algorithmes cascadant ces constructions constitue un enjeu crucial pour ce domaine.

Dans ce cadre, l'objectif principal de cette thèse est de définir une représentation à précision fixée de certaines constructions géométriques préservant les propriétés caractéristiques de l'objet original. Ces méthodes d'arrondi nous permettront ainsi de contrôler la complexité numérique des objets géométriques construits tout en assurant la cohérence de ces objets. Plus précisément, nous

nous intéresserons au plongement dans un ensemble de valeurs entières du résultat d'opérations booléennes sur des régions polygonales dans le plan. Nous définirons ainsi plusieurs notions d'arrondi au niveau de régions polygonales comme l'arrondi par défaut et par excès garantissant certaines propriétés intéressantes telles que la planarité de la région arrondie, des relations d'inclusion et de proximité entre le résultat exact de l'opération et sa version arrondie, voire la préservation de la convexité du résultat.

Les objectifs exposés ci-dessus nécessitent bien sûr avant d'arrondir un objet complexe tel qu'un polygone, de savoir arrondir ses éléments de base, ses sommets. Nous aborderons donc séparément le volet arithmétique : comment obtenir les coordonnées d'une construction géométrique simple, c'est-à-dire comment calculer l'arrondi d'un objet géométrique simple (comme un sommet d'un polygone ou un point d'intersection entre deux segments) avec des garanties sur la qualité du résultat, et le volet plus algorithmique et combinatoire : comment arrondir des objets complexes (polygones et régions polygonales) en préservant certaines propriétés de la structure et concevoir des algorithmes utilisant des constructions arrondies plutôt que des constructions exactes.

Parmi les constructions géométriques simples abordées dans la partie arithmétique, nous présenterons quelques algorithmes d'arrondi certifié de constructions élémentaires au plus proche point entier. Le calcul du plus proche point entier de constructions géométriques simples est assez facile à l'aide d'outils arithmétiques complexes et relativement lents ; nous mènerons par conséquent l'étude de techniques d'accélération, similaires à celles conçues pour les prédicats, pour ces calculs. Nous proposerons ainsi quelques méthodes basées sur des techniques de filtrage numérique permettant d'éviter dans la plupart des cas l'utilisation d'une arithmétique exacte et ainsi de réduire l'espace et le temps nécessaires aux calculs. Nous décrirons, d'autre part, un nouvel algorithme permettant l'arrondi d'un sommet d'un polygone convexe au plus proche point entier appartenant à ce polygone et qui sera utilisé par les méthodes d'arrondi de régions polygonales.

Enfin, nous nous intéresserons à un type de prédicat particulier, les tests d'intersection de polygones convexes dans l'espace bi- et tridimensionnel, et plus spécifiquement au test d'intersection de triangles tridimensionnels qui est un prédicat très utilisé par les algorithmes de détection de collision. L'utilisation de méthodes constructives dans le domaine de l'informatique graphique est souvent un moyen d'obtenir des algorithmes naturels et très efficaces pour ce type de primitive. Nous montrerons comment ces constructions, qui ne sont pas inhérentes au problème, peuvent être supprimées et remplacées par des tests d'orientation au bénéfice d'une amélioration des performances globales de l'algorithme, c'est-à-dire à la fois en termes de fiabilité et de temps de calcul.

L'ensemble des méthodes présentées dans cette thèse a été élaboré avec un souci d'effectivité. En particulier, l'accent a été mis sur l'obtention d'algorithmes n'utilisant que des primitives numériques de faible degré algébrique ce qui permet de limiter les exigences arithmétique et algorithmique de leur implantation robuste.

Organisation du manuscrit

Cette thèse est organisée en quatre chapitres. Le premier chapitre a pour objet de mettre en évidence à travers un exemple particulier, l'algorithme de balayage de Bentley-Ottmann, l'interaction subtile entre l'aspect numérique et l'aspect combinatoire typique aux algorithmes géométriques et qui est à la source des problèmes de robustesse. Cet algorithme particulier nous permettra en outre de présenter l'évolution de la géométrie algorithmique face aux problèmes issus de l'implémentation effective des algorithmes géométriques, vers ce que l'on appelle le *calcul géométrique*.

Le deuxième chapitre propose une nouvelle méthode permettant de tester l'intersection de triangles et de polygones bi- et tridimensionnels. Une analyse comparative avec les différentes méthodes existantes nous permettra de mesurer les performances de l'implémentation obtenue aussi bien en termes de robustesse qu'en termes de rapidité de calcul. Les résultats exposés dans ce chapitre ont fait l'objet d'une publication dans *the Journal of Graphics Tools* [38] et ont été intégrés dans le bibliothèque d'algorithmes géométriques CGAL (*Computational Geometry Algorithms Library*) [8].

Le troisième chapitre décrit un ensemble de primitives permettant l'arrondi certifié au plus proche point entier de constructions géométriques élémentaires telles que le point d'intersection de segments dans le plan, le centre du cercle circonscrit à un triangle bidimensionnel ou le centre de la sphère circonscrite à un tétraèdre dans l'espace. Ce chapitre décrit, de plus, un algorithme existant permettant l'arrondi du sommet d'un secteur convexe au plus proche point entier appartenant à ce secteur et présente une nouvelle méthode pour le calcul de l'arrondi d'un sommet d'un polygone convexe au plus proche point entier de ce polygone.

Finalement, le quatrième chapitre aborde le problème de l'arrondi de structures géométriques complexes sur une grille uniforme et présente plusieurs techniques de régularisation permettant le plongement du résultat d'opérations booléennes sur des régions polygonales dans un ensemble de valeurs entières. Pour chaque opération, nous proposons un mode d'arrondi par défaut (certifiant l'inclusion de la région arrondie dans la région exacte) et un mode d'arrondi par excès (certifiant l'inclusion de la région exacte dans la version arrondie) et prouvons quelques propriétés caractéristiques garanties par ces modes d'arrondi telles que la planarité du résultat, la détermination

d'une borne sur l'erreur géométrique introduite et d'une borne sur le nombre de sommets de la version arrondie. Nous décrivons de plus des algorithmes pratiques permettant d'obtenir ces arrondis et analysons leur complexité.

Chapitre 1

Calcul géométrique

La géométrie algorithmique a produit de nombreux résultats combinatoires et algorithmiques. Le succès et la productivité de cette discipline sont en grande partie attribuables à l'adoption d'hypothèses qui facilitent considérablement l'élaboration des algorithmes et offrent un cadre simple pour leur analyse. Outre l'hypothèse désormais classique de *position générale* des données qui suppose l'absence de cas particuliers, la preuve de la validité et de la complexité asymptotique des algorithmes de la géométrie algorithmique sont en effet basées sur un modèle de calculateur qui permet de traiter de manière exacte et en temps constant les opérations sur les réels.

Cependant, le concept de nombre réel et l'hypothèse d'un calcul exact sur ce type de nombres est une pure abstraction mathématique et ne saurait être représenté dans un ordinateur s'appuyant sur le paradigme du calcul à précision fixée. Dans sa forme la plus simple, ce paradigme suppose qu'une certaine précision (une mémoire de taille finie dépendante de la machine) est disponible pour représenter et manipuler les nombres. Cette précision bornée n'est évidemment pas suffisante pour se substituer au calcul sur les réels et rend inévitables les erreurs d'arrondi.

Les algorithmes géométriques sont malheureusement particulièrement sensibles aux problèmes de précision numérique, spécifiquement parce qu'ils combinent d'une part des aspects combinatoires (liés par exemple à la manipulation de graphes), et d'autre part des aspects numériques (liés à la représentation numérique des objets géométriques manipulés, par exemple, leurs coordonnées cartésiennes). L'utilisation naïve d'une arithmétique flottante conduit ainsi fréquemment à des problèmes de robustesse qui rendent très difficile l'obtention d'implantations fiables d'algorithmes théoriques : les erreurs d'arrondi effectuées dans l'évaluation des tests géométriques introduisent des incohérences topologiques qui rendent parfois le résultat de l'algorithme géométriquement incorrect voire provoquent l'arrêt inopiné du programme. Si ces problèmes sont bien connus, ils restent toutefois difficiles à maîtriser et constituent un des principaux obstacles

à l'utilisation des résultats de la géométrie algorithmique. Ainsi, au-delà de la conception et de l'analyse des algorithmes, une part importante de la recherche récente en géométrie algorithmique, que l'on appelle *calcul géométrique*, a pour but de résoudre les questions délicates posées par leur programmation afin de garantir un calcul fiable.

Le but de ce chapitre est de présenter à travers un problème particulier les grandes tendances de l'évolution de la géométrie algorithmique face aux difficultés de l'implémentation effective des algorithmes géométriques.

Nous décrivons à la section 1.1 un algorithme fondamental de la discipline : l'algorithme de balayage de Bentley-Ottmann. Cet algorithme particulier nous permettra tout au long de ce chapitre d'illustrer les problèmes soulevés et les solutions apportées par le calcul géométrique. Les sections 1.2 et 1.3 reviennent plus en détail sur les hypothèses utiles à la conception d'algorithmes en géométrie, à savoir l'adoption du modèle Real-RAM et l'hypothèse d'absence de cas dégénérés. La section 1.4 donne une description de l'arithmétique flottante IEEE 754 qui constitue le standard de l'arithmétique à précision fixée sur ordinateur. Les problèmes de robustesse dûs à l'implémentation des algorithmes de la géométrie algorithmique à l'aide d'une arithmétique approchée sont évoqués à la section 1.5. Une des approches pour résoudre ces problèmes de manière générale suit le paradigme du calcul géométrique exact [72] dont un exemple d'application est donné à la section 1.6. Ce type de solution donne des méthodes robustes qui une fois couplées à des méthodes de perturbations symboliques permettent de résoudre le problème des cas dégénérés au niveau de l'arithmétique. Ces méthodes ne sont cependant directement envisageables en pratique que dans quelques cas restreints qui sont décrits à la section 1.7 du fait de son coût en temps.

L'adoption du paradigme exact en géométrie algorithmique a toutefois ouvert un nouveau champ de recherche dont le but est de garantir un calcul fiable et efficace des algorithmes géométriques. La démarche qui sous-tend ces études consiste à distinguer deux types de primitives numériques intervenant dans ces algorithmes : les *prédicats* (cf. section 1.8) et les *constructeurs* (cf. section 1.10). Ces dernières années de nombreux travaux ont permis le développement d'une arithmétique des prédicats exacte tant par la conception de méthodes de calcul exact que par la mise au point et l'étude de *filtres arithmétiques* (cf. section 1.9) permettant d'obtenir des temps de calculs raisonnables. Cette approche garantit la robustesse et donne des résultats remarquables pour les algorithmes dont la sortie est purement combinatoire.

Les problèmes de robustesse s'avèrent cependant beaucoup plus délicats pour les algorithmes construisant de nouveaux objets géométriques, c'est-à-dire utilisant des constructeurs, du fait de la croissance des coefficients numériques intervenant dans l'algorithme. La section 1.11 expose

finalement le développement des méthodes d'arrondi de constructions, aussi appelée méthodes de régularisation, nécessaires au contrôle et à la limitation de cette croissance.

1.1 Un exemple d'algorithme géométrique : Bentley-Ottmann

Plutôt que d'inventorier les grands problèmes traités par la géométrie algorithmique nous nous intéressons à une classe de problèmes géométriques précis : le problème du balayage d'un ensemble de segments dans le plan.

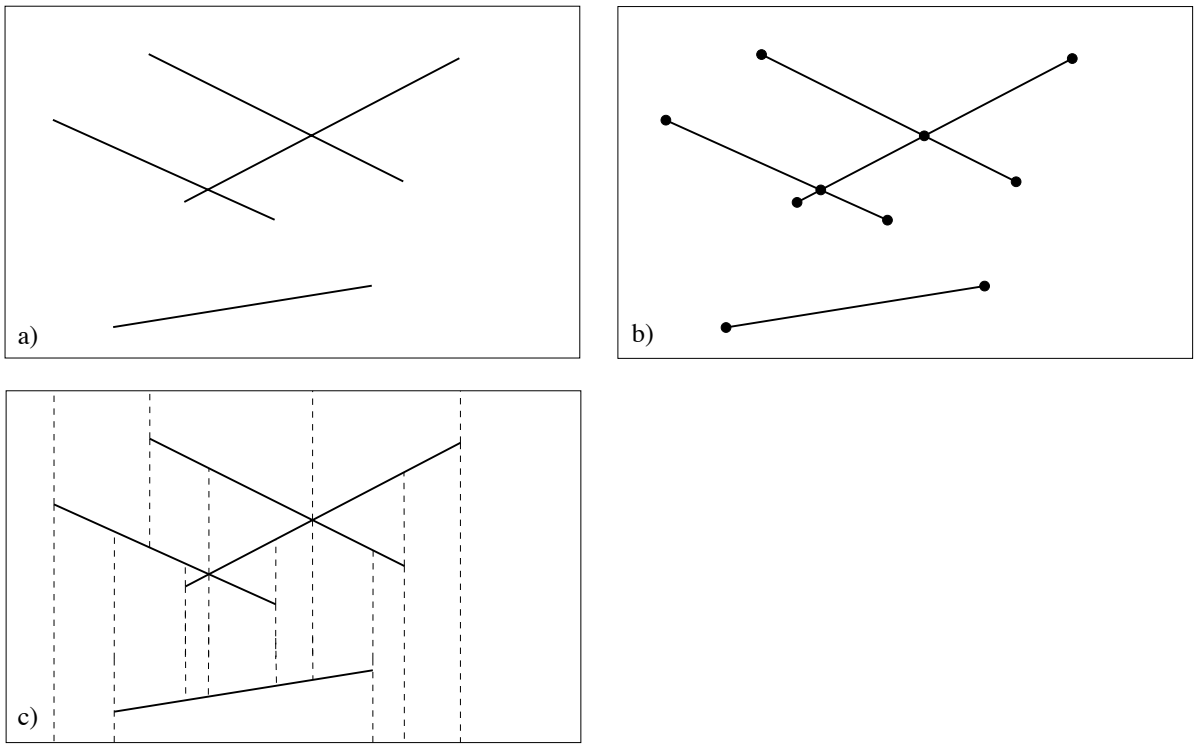

FIG. 1.1 – a) Un ensemble de segments S, b) l'arrangement correspondant et c) le cloisonnement vertical de S.

Étant donné un ensemble fini S de segments dans le plan représentés par les coordonnées de leurs extrémités, on distingue les problèmes suivants :

- déterminer les paires de segments de S qui se coupent (cf. figure 1.1.a),
- calculer l'arrangement A induit par l'ensemble des segments de S, c'est-à-dire les relations d'incidence du graphe obtenu en interprétant l'union des segments comme un graphe planaire. L'arrangement A est obtenu à partir de S en coupant chacun des segments initiaux

en leur point d'intersection avec les autres segments et en déterminant l'ordre cyclique des arêtes obtenues en chaque sommet (extrémités ou points d'intersections) (cf. figure 1.1.b).

– calculer le cloisonnement vertical V de S (cf. figure 1.1.c). Le cloisonnement vertical d'un ensemble de segments est obtenu en traçant, à partir de chaque extrémité et de chaque intersection de segments, deux cloisons verticales (vers le haut et vers le bas) que l'on étend jusqu'à ce qu'elles rencontrent un segment de S ou un segment de la boîte englobant l'ensemble des segments de S. Le cloisonnement vertical d'un ensemble de segments S correspond à la subdivision de chacune des régions définies comme les parties connexes de $\mathbb{R}^2 \setminus S$ en régions élémentaires trapézoïdales. Cette décomposition du plan peut être considérée comme le prototype de toute une classe de décomposition géométriques analogues, appelées aussi cloisonnements.

L'algorithme de Bentley-Ottmann [4] de balayage du plan constitue l'une des premières solutions classiques de la géométrie algorithmique. Cet algorithme particulier est l'un des plus fameux de la discipline et a véritablement donné naissance à une technique algorithmique spécifique à la géométrie algorithmique. Outre les problèmes déjà évoqués liés à l'arrangement de segments, la technique de balayage s'applique en effet à de nombreux autres problèmes géométriques comme par exemple le calcul du diagramme de Voronoï [24].

Le principe général d'un algorithme par balayage consiste à simuler le balayage du plan par une droite verticale allant de gauche à droite. La structure recherchée (la liste des paires de segments qui s'intersectent, l'arrangement voire le cloisonnement de S) est découverte progressivement, au fur et à mesure du déplacement de cette droite.

Afin de simplifier la description de l'algorithme, nous supposerons les segments en entrée en *position générale*. L'hypothèse simplificatrice de position générale est commune dans les descriptions théoriques des algorithmes en géométrie algorithmique et est abordée plus en détail à la section 1.3. Pour notre problème particulier cette hypothèse consiste à supposer que les sommets de l'arrangement (c'est-à-dire les extrémités de segment et les points d'intersection entre segments) ont des abscisses distinctes. Cette condition interdit de fait les segments verticaux, les sommets initiaux confondus ainsi que les points d'intersection communs à plus de deux segments.

L'algorithme est basé sur les deux observations suivantes :

– la liste ordonnée des segments coupés par la droite de balayage, appelée *front de balayage*, ne change que lorsque la droite passe sur certaines positions critiques (extrémités des segments initiaux, points d'intersection entre les segments) que l'on appelle *événements*. Un événement *extrémité* provoque par exemple l'insertion ou la suppression du segment associé, un événement correspondant à un point d'intersection entre deux segments permute

l'ordre de ces segments.

– seules les paires de segments consécutives dans le front de balayage peuvent s'intersecter (excepté pour les cas dégénérés).
– pour s'intersecter en un point i deux segments doivent être consécutifs dans la liste des segments coupés par la droite de balayage à gauche de i. Ainsi, l'ensemble des points d'intersection entre les segments initiaux peut être déterminé en testant uniquement l'intersection entre segments adjacents dans le front de balayage.

Les différentes situations où une nouvelle adjacence entre deux segments est créée dans la liste ordonnée des segments coupés par la droite de balayage sont décrites par la figure 1.2.

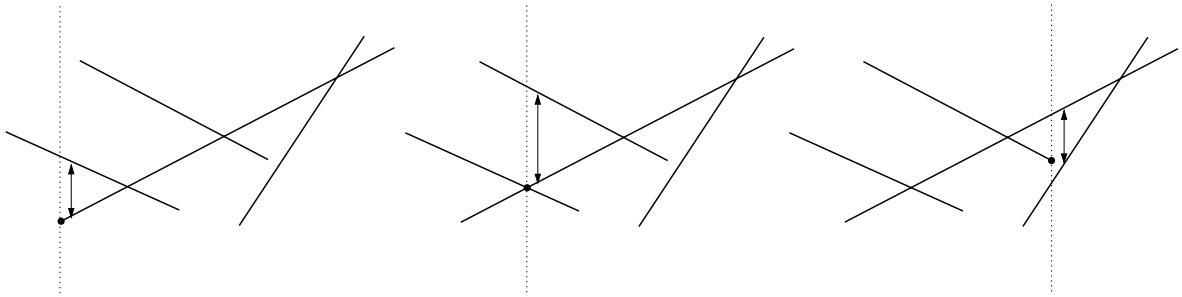

FIG. 1.2 – Les trois événements durant lesquels les paires de segments qui s'intersectent deviennent pour la première fois adjacents dans le front de balayage. La droite de balayage est représentée en pointillés.

L'algorithme manipule plusieurs structures de données. La première est une structure pour gérer les événements à venir, appelée la *liste des événements*. Cette structure est initialisée avec la liste triée en x des sommets initiaux (correspondant aux extrémités des segments). Ensuite, l'algorithme utilise une structure pour représenter le front de balayage. Ce front de balayage est constitué par l'ensemble des segments coupés par la droite de balayage, c'est-à-dire l'ensemble des segments dont le domaine de définition en x contient l'événement courant, triés selon l'ordonnée de leur intersection avec la droite par ordre croissant. Enfin, l'algorithme emploie une structure pour représenter le résultat dans le cas ou une structure combinatoire est calculée (arrangement ou cloisonnement partiel).

Initialement, le front de balayage est vide. Tant que l'ensemble des événements n'a pas été traité l'algorithme maintient :

– la liste X des événements triés en x en insérant les événements intersection découverts lors du balayage à leur place et en supprimant les événements de la liste une fois ceux-ci traités.

– le front de balayage, en tenant à jour à chaque événement la liste Y des segments coupés par la droite de balayage triés selon l'ordonnée de leur intersection avec la droite. D'après les observations précédentes, cette tâche se réduit à insérer ou à supprimer du front de balayage les segments initiaux lors du traitement des événements extrémités, et à permuter deux segments de la liste lors du traitement d'événements intersection.

La liste des événements et le front de balayage sont classiquement représentés par des arbres binaires équilibrés (bien que dans sa version originale l'algorithme de Bentley-Ottmann utilise une queue de priorité pour représenter la liste des événements) qui permettent d'effectuer chaque opération de recherche, d'insertion et de suppression (d'un événement ou d'un segment) en $O(\log n)$ opérations de comparaisons.

1.2 Le modèle de calculateur Real-RAM

Au delà de la conception d'algorithmes théoriques, l'objectif poursuivi en géométrie algorithmique est d'en analyser les performances théoriques et pratiques afin de pouvoir mesurer et comparer leur efficacité réelle. Pour cela on a besoin d'une modélisation simple des ordinateurs qui parvienne à mettre à jour les ressources et les opérations critiques d'un algorithme et traduise ainsi correctement la complexité asymptotique de celui-ci.

On différencie classiquement deux modèles de calculateur en informatique théorique, la machine de Turing et le modèle Real-RAM. La machine de Turing utilise une représentation binaire des nombres et mesure la complexité des algorithmes en termes de nombre d'opérations sur ces bits. Le modèle Real-RAM (*modèle de calculateur réel à accès aléatoire*), quant à lui, suppose que les opérations sur les réels sont exactes et se font en temps constant. Plus précisément, ce dernier modèle suppose que l'on dispose d'un calculateur pour lequel chaque nombre réel peut être représenté exactement dans un espace mémoire de taille constante et pour lequel l'accès à ces espaces mémoire, les opérations arithmétiques de base (addition, soustraction, multiplication, division, racines, logarithmes ...) ainsi que de comparaisons sur ces nombres peuvent être effectuées de manière exacte et en temps constant.

Par sa simplicité, le modèle Real-RAM s'est imposé comme un standard pour l'étude des complexités que ce soit en temps ou en espace mémoire des algorithmes et constitue le modèle d'arithmétique en vigueur en informatique. En plus d'offrir un cadre pour l'étude de la complexité des algorithmes, le modèle Real-RAM et plus particulièrement l'hypothèse d'un calcul exact sur les réels permet de se placer dans le cadre de la géométrie classique (et non discrète) et ainsi de prouver la validité des algorithmes théoriques. On peut en effet utiliser les théorèmes usuels

de la géométrie, typiquement les propriétés géométriques du plan euclidien, et ainsi obtenir des algorithmes élaborés s'appuyant sur les propriétés combinatoires et topologiques du problème à résoudre.

En adoptant ce modèle, l'algorithme de balayage de la section précédente s'analyse de la manière suivante. Les opérations critiques en termes de temps de calcul de l'algorithme proviennent de la mise à jour à chaque événement de la liste des événements et du front de balayage. Il existe exactement $2n + a$ événements où n est le nombre de segments en entrée et a est le nombre de points d'intersection entre ces segments. Or, si X et Y sont représentés par des arbres équilibrés, leur accès et leur mise à jour nécessite $O(\log n)$ comparaisons par événement. On obtient donc une solution en $O((n + a) \log n)$ nettement plus efficace que l'algorithme naïf (en $O(n^2)$) qui teste l'intersection de toutes les paires de segments. On note cependant que la borne inférieure de ce problème est $\Theta(n \log n + a)$ et qu'il existe depuis des algorithmes optimaux pour déterminer les paires de segments qui s'intersectent [3] et pour calculer l'arrangement des segments [11].

Comme nous le verrons dans la suite de ce chapitre, l'hypothèse faite par le modèle Real-RAM (et par conséquent dans cette analyse) supposant que les opérations sur les réels peuvent être évaluées de manière exacte et en temps constant ne correspond pas à la réalité d'un ordinateur et peut poser des problèmes rapidement observables dans le cas des algorithmes géométriques.

1.3 Dégénérescences

Après la possibilité de manipuler les nombres réels, l'absence de dégénérescence est une autre hypothèse simplificatrice de la description théorique des algorithmes géométriques. La notion de cas dégénéré est directement liée à l'utilisation de tests géométriques dans les algorithmes. Ces tests correspondent à des questions géométriques élémentaires et admettent un nombre fini de réponses possibles. Dans les cas typiques, ces réponses sont par exemple données par le signe d'une expression numérique et sont donc au nombre de trois : positif, négatif ou nul. On peut interpréter ces trois réponses comme deux réponses génériques (positif ou négatif) et une réponse intermédiaire, que l'on qualifiera de *dégénérée*, et qui correspond à un cas limite entre ces deux cas génériques. En excluant ces cas dégénérés, l'hypothèse de position générale permet de se passer des traitements relatifs aux cas particuliers et ainsi d'obtenir des algorithmes plus simples.

On peut distinguer deux types de dégénérescences, celles intrinsèquement liées au problème, et celles résultant du choix de l'algorithme utilisé. Pour le problème de détection des points d'intersection d'un ensemble de segments, trois segments concourants correspondent à une dégénérescence intrinsèque au problème : le résultat est modifié si on déplace un peu les segments

(on passe d'un point d'intersection à trois). Par contre, l'algorithme de balayage compare des abscisses de points ; deux points à la même abscisse sera donc considéré comme un cas dégénéré mais ne modifie en rien le résultat.

Bien que moins contraignante que l'hypothèse du calcul exact sur les réels, l'hypothèse de position générale est cependant souvent remise en question en pratique par certains jeux de données. Le non respect de cette hypothèse conduit ainsi fréquemment à un manque de fiabilité de l'implémentation d'algorithmes géométriques prouvés théoriquement.

Afin de s'abstraire en pratique de cette hypothèse deux solutions sont possibles. Le traitement explicite des cas particuliers (en adaptant l'algorithme prévu pour des données en position générale) est souvent un premier type de solution envisageable. Cependant, ce type de démarche peut se révéler très fastidieux et complique notablement les algorithmes, ce qui se traduit assez souvent par une dégradation des performances globales de l'algorithme en termes de temps de calcul et d'occupation mémoire (les structures de données ont en effet souvent besoin d'être modifiées).

L'alternative à la gestion explicite des cas particuliers est donnée par la simulation de position générale à l'aide de techniques de perturbation symbolique [71, 19, 20, 67, 1]. L'avantage de ces méthodes est de résoudre le problème du traitement des cas dégénérés sans modifier l'algorithme, c'est-à-dire au niveau des prédicats. Cette solution nécessite toutefois l'utilisation d'une arithmétique exacte. D'autre part, l'inconvénient principal est que l'on résout en fait un problème limite du problème initial. Dans le cas où l'on souhaite calculer l'arrangement d'un ensemble de trois segments concourants, l'utilisation d'une technique de perturbation symbolique conduira par exemple à la construction de trois points d'intersection distincts et non pas à la construction d'un point d'intersection commun.

1.4 L'arithmétique flottante IEEE 754

La norme ANSI/IEEE 754 (*Standard for binary Floating-Point Arithmetic* [44]) est aujourd'hui la représentation la plus commune en termes de calculs en virgule flottante en informatique et est maintenant implantée dans la quasi totalité des architectures et plates-formes actuelles. Celle-ci a été définie dans le but d'améliorer la qualité et la portabilité du calcul flottant. Le standard IEEE 754 spécifie précisément le codage des nombres à virgule flottante ainsi que le comportement des opérations qui les manipulent.

Nous donnons dans cette section quelques propriétés de cette arithmétique et renvoyons le lecteur intéressé par une description plus détaillée à la norme elle-même [44] ou à l'article de synthèse [30].

Représentation La norme IEEE 754 spécifie quatre formats possibles ; cependant nous n'utiliserons dans la suite que les nombres à virgule flottante simple et double précision correspondant respectivement aux types `float` et `double` du langage C. Un nombre flottant double précision est codé sur deux mots mémoires de 32 bits et utilise un bit pour le signe s, 52 bits pour la mantisse m et 11 bits pour l'exposant e (cf. figure 1.3). La valeur d'un nombre flottant double précision est égale à $(-1)^s \times m_0, m \times 2^{e-1023}$. En notation normalisée (c'est-à-dire lorsque $m_0 \neq 0$), le bit de poids le plus fort de la mantisse est forcément égal à 1 et il n'est donc pas nécessaire de représenter ce chiffre, ce qui revient à disposer d'une mantisse de 53 chiffres binaires significatifs. Les nombres représentables en machine en double précision sont donc régulièrement espacés avec des intervalles de 2^{p-52} entre 2^p et 2^{p+1}.

Les valeurs extrêmes de l'exposant permettent de coder les exceptions et les nombres dénormalisés (le nombre zéro est, par exemple, représenté par une mantisse égale à zéro associée au plus petit exposant possible). L'utilisation d'une représentation dénormalisée permet de pallier à l'absence de nombre représentable dans le voisinage de zéro (entre -2^{-1023} et 2^{-1023}). En notation dénormalisée (c'est-à-dire lorsque $e = 0$), $m_0 = 0$ et la valeur du nombre représenté est égale à $(-1)^s \times 0, m \times 2^{-1023}$, ce qui permet de représenter les nombres de l'intervalle $[-2^{-1023}$ et $2^{-1023}]$ avec des écarts réguliers de 2^{-1076}.

Un nombre flottant simple précision est quant à lui codé sur un seul mot mémoire de 32 bits et utilise un bit pour le signe, 8 bits pour l'exposant et $23 + 1$ bits pour la mantisse, sa valeur est alors égale à $(-1)^s \times m_0, m \times 2^{e-127}$.

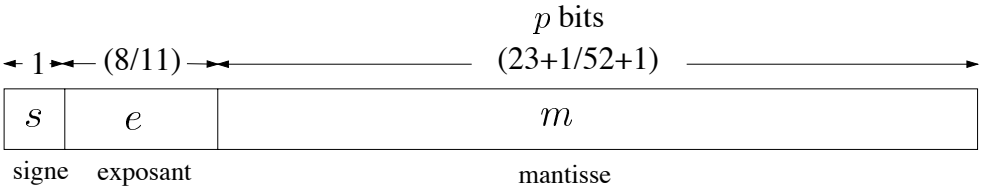

FIG. 1.3 – Codage des nombres flottants simple et double précision selon la norme IEEE754.

Opérations et modes d'arrondis La norme spécifie très précisément le résultat des opérations arithmétiques de base ($+$,$-$,$\times$,/ ,$\sqrt{}$). Dans le standard IEEE, un arrondi se produit dès lors que le résultat d'une opération de base n'est pas représentable. Le système se comporte dans ce cas comme si le calcul était fait avec une précision *infinie* puis arrondi (on parle d'arrondi *certifié* ou *correct*). Les principales propriétés de l'arrondi correct sont la prévisibilité et la reproductibilité du comportement du système et la possibilité d'élaborer des algorithmes et des preuves

(arithmétique d'intervalles, arithmétique stochastique, etc...).

Par défaut, le mode d'arrondi utilisé est le mode d'arrondi au plus proche (c'est le plus proche nombre représentable du résultat exact qui est retourné). Le standard impose toutefois que trois autres modes d'arrondi soient disponibles, à savoir, le mode d'arrondi vers zéro, le mode d'arrondi vers $+\infty$ et le mode d'arrondi vers $-\infty$ (cf. figure 1.4).

Nous utiliserons dans la suite uniquement le mode d'arrondi au plus proche. Dans les cas où l'arrondi au plus proche est ambiguë, c'est-à-dire lorsque le résultat exact de l'opération est exactement entre deux nombres flottants représentables, le résultat retourné est le nombre flottant dont la mantisse est paire (IEEE *round-to-even digit rule*).

On note donc dans la suite $\oplus$, $\ominus$, $\otimes$, et $\oslash$ les opérations d'addition, de soustraction, de multiplication et de division entre deux flottants double précision selon le mode d'arrondi au plus proche.

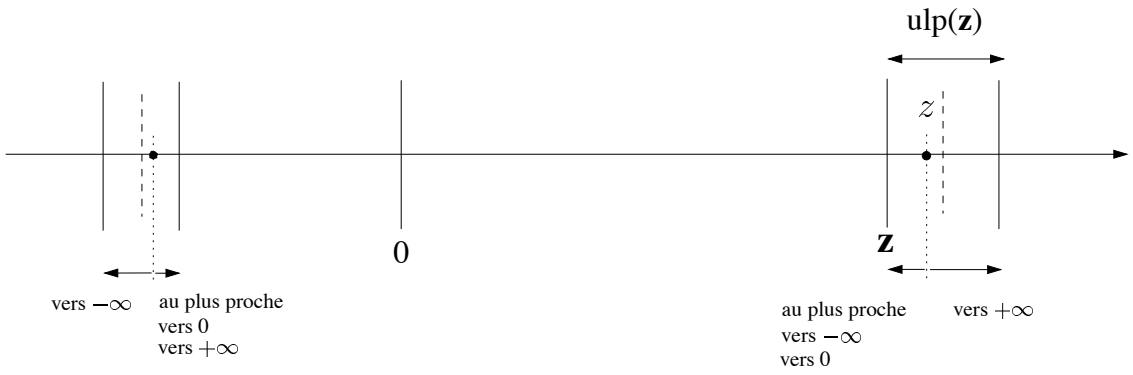

FIG. 1.4 – Les modes d'arrondis IEEE 754.

Propriétés Les propriétés de l'arithmétique IEEE ne sont évidemment pas celles des réels, la propriété absente la plus cruciale étant l'associativité. On a toutefois la commutativité pour l'addition et la multiplication ($x \oplus y = y \oplus x$ et $x \otimes y = y \otimes x$), la soustraction vérifie $x \ominus y = -(y \ominus x)$. De plus, l'arrondi préserve l'ordre respectif de deux nombres, bien que les inégalités strictes deviennent larges ($a \times b < c \times d \Rightarrow a \otimes b \leq c \otimes d$). Grâce à l'utilisation de la notation dénormalisée, la soustraction et la comparaison sont bien compatibles ($a = b \Leftrightarrow a \ominus b = 0$). Le caractère associatif est toutefois perdu ($(a \oplus b) \ominus c \neq a \oplus (b \ominus c)$) puisque les bits les moins significatifs du résultat d'une opération disparaissent.

On peut cependant *utiliser* cette non associativité. L'addition puis la soustraction de la quantité $\mathbf{C} = 3 \times 2^{51}$ calculent, par exemple, le plus proche entier d'un nombre flottant double précision

(avec arrondi vers la valeur paire si le nombre de départ a comme partie fractionnaire $\frac{1}{2}$), pourvu que ce nombre soit plus petit que 2^{51} en valeur absolue. L'addition a pour effet la disparition des bits de poids faible du nombre initial, la soustraction est par contre exacte. Ainsi, la plus proche valeur entière **e** de $\mathbf{z} \in]-2^{51}, +2^{51}[$ est obtenue par $\mathbf{e} = (\mathbf{z} \oplus \mathbf{C}) \ominus \mathbf{C}$, on peut ensuite récupérer la partie fractionnelle **f** de **z** par $\mathbf{f} = \mathbf{z} \ominus \mathbf{e}$, cette soustraction étant calculée exactement.

Arithmétique flottante et calcul d'erreur On note $\mathrm{ulp}(\mathbf{z})$ (*unit in last place*) la valeur du plus petit bit dans le développement de la mantisse d'un nombre flottant **z**, c'est-à-dire la différence entre **z** et le nombre flottant qui lui est immédiatement supérieur en valeur absolue (cf. figure 1.4). Les bornes d'erreur de chacune des opérations de base (c'est-à-dire la différence entre la valeur réelle z du résultat et la valeur flottante **z** calculée) se calculent en utilisant les propriétés de la norme qui garantissent une erreur absolue maximale égale à $\frac{1}{2}\mathrm{ulp}(\mathbf{z})$ dans le cas où le mode d'arrondi est au plus proche et $\mathrm{ulp}(\mathbf{z})$ pour les autres modes.

Chaque valeur réelle z peut être représentée par une valeur approchée **z** et une erreur $\mathbf{e}_z$ toutes deux sous la forme de nombres flottants IEEE, tels que l'on soit certain que $|z - \mathbf{z}| \leq \mathbf{e}_z$.

Initialement, si la valeur z n'est pas représentable par un nombre flottant double précision, on a

$$|z - \mathbf{z}| \leq |(\mathbf{z} + \frac{1}{2}\mathrm{ulp}(\mathbf{z})) - \mathbf{z}|$$

$$\mathbf{e}_z = \frac{1}{2}\mathrm{ulp}(\mathbf{z}).$$

On obtient les bornes suivantes pour les opérations d'addition, multiplication et division avec le mode d'arrondi au plus proche.

Addition et soustraction

$$\begin{array}{rcccccl} |z - \mathbf{z}| & = & |(z - (x+y)) & + & ((x+y) - (\mathbf{x}+\mathbf{y})) & + & ((\mathbf{x}+\mathbf{y}) - \mathbf{z})| \\ & \leq & 0 & + & \mathbf{e}_x + \mathbf{e}_y & + & \frac{1}{2}\mathrm{ulp}(\mathbf{z}) \end{array}$$

d'où

$$\mathbf{e}_{x+y} = \mathbf{e}_x + \mathbf{e}_y + \frac{1}{2}\mathrm{ulp}(\mathbf{x} \oplus \mathbf{y}).$$

Multiplication

$$\begin{array}{rcccccl} |z - \mathbf{z}| & = & |(z - (xy)) & + & ((xy) - (\mathbf{xy})) & + & ((\mathbf{xy}) - \mathbf{z})| \\ & \leq & 0 & + & (\mathbf{x} - x)(\mathbf{y} - y) - (\mathbf{x} - x)\mathbf{y} - (\mathbf{y} - y)\mathbf{x} & + & \frac{1}{2}\mathrm{ulp}(\mathbf{z}) \end{array}$$

d'où

$$\mathbf{e}_{x \times y} = \mathbf{e}_x \times \mathbf{e}_y + \mathbf{x} \times \mathbf{e}_y + \mathbf{y} \times \mathbf{e}_x + \frac{1}{2}\mathrm{ulp}(\mathbf{x} \otimes \mathbf{y}).$$

Division

$$\begin{array}{rcccccc} |z-\mathbf{z}| & = & |(z-(x/y)) & + & ((x/y)-(\mathbf{x}/\mathbf{y})) & + & ((\mathbf{x}/\mathbf{y})-\mathbf{z})| \\ & \leq & 0 & + & \frac{(x-\mathbf{x})\mathbf{y}-(y-\mathbf{y})\mathbf{x}}{y\mathbf{y}} & + & \frac{1}{2}\mathrm{ulp}(\mathbf{z}) \end{array}$$

d'où

$$\mathbf{e}_{x/y} = \frac{\mathbf{x}\times\mathbf{e}_y + \mathbf{y}\times\mathbf{e}_x}{|\mathbf{y}|\times(|\mathbf{y}|-\mathbf{e}_y)} + \frac{1}{2}\mathrm{ulp}(\mathbf{x}\oslash\mathbf{y}).$$

On note qu'il est nécessaire dans ce cas d'avoir une borne inférieure sur $(|\mathbf{y}|-\mathbf{e}_y)$.

1.5 Problèmes de robustesse

Les méthodes les plus sophistiquées de la géométrie algorithmique sont théoriquement très efficaces car elles s'appuient sur des propriétés géométriques et mathématiques. L'algorithme de Bentley-Ottmann utilise par exemple intensivement la propriété de transitivité de l'ordre lexicographique : si $a <_{xy} b$, et $b <_{xy} c$, alors on peut déduire $a <_{xy} c$ sans avoir à comparer explicitement a et c. Ces théorèmes et ces propriétés assurent la cohérence logique de l'algorithme : l'évaluation exacte des primitives utilisées par l'algorithme garantit que les situations de contradiction interne sont évitées. Cette cohérence logique assure à son tour que le programme produira toujours un résultat et garantit la validité du résultat calculé (par exemple la planarité de l'arrangement calculé).

Les problèmes de robustesse des algorithmes géométriques sont directement liés à l'incompatibilité entre d'une part la description théorique des algorithmes s'appuyant sur ces théorèmes vérifiés dans le cadre de la géométrie classique, c'est-à-dire dans le monde continu, et d'autre part la nature finie de la représentation des nombres en machine. Les propriétés géométriques sur lesquelles sont basés les algorithmes se traduisent, en effet, par l'évaluation de primitives numériques durant l'exécution de l'algorithme. Les erreurs d'arrondi de l'arithmétique flottante utilisée comme substitut à la représentation et à la manipulation des nombres réels peut conduire sur certains jeux de données à des décisions ou à des résultats intermédiaires erronés invalidant ces propriétés. Ces erreurs se propagent dans l'algorithme et remettent en cause la cohérence interne du programme. L'algorithme rentre alors dans une situation théoriquement impossible qui sèment le trouble dans les structures de données et conduit dans la presque totalité des cas à un comportement anormal.

Les algorithmes géométriques sont particulièrement sensibles aux problèmes d'instabilité numérique du fait de leur nature essentiellement discontinue liée à leur caractère combinatoire. Les méthodes et structures de données en géométrie se basent souvent sur des tests de signe de dé-

terminants ou de polynômes. Une erreur "négligeable" numériquement dans le programme peut provoquer une inversion de signe. Cette simple erreur de signe peut alors avoir des effets catastrophiques sur la suite du déroulement de l'algorithme. Du fait qu'ils constituent un obstacle majeur à l'utilisation pratique des résultats de la géométrie algorithmique, les problèmes de robustesse ont engendré un débat vigoureux dans la communauté de la géométrie algorithmique et il existe de nombreux articles [22, 23, 43, 53, 26] rapportant les conséquences pratiques de ce type de problèmes dans l'histoire de la discipline.

Le reste de cette section illustre par quelques exemples représentatifs les conséquences possibles des imprécisions numériques de l'arithmétique flottante sur l'algorithme de balayage de Bentley-Ottmann présenté à la section 1.1.

Exemples d'incohérences

On peut distinguer deux types de conséquences relatives aux erreurs d'arrondi de l'arithmétique flottante. Le premier bouleverse la cohérence interne des algorithmes géométriques et mène à un arrêt inopiné ou à une non terminaison du programme, le second affecte uniquement la cohérence géométrique du résultat calculé : l'algorithme termine normalement mais retourne un résultat géométriquement incohérent.

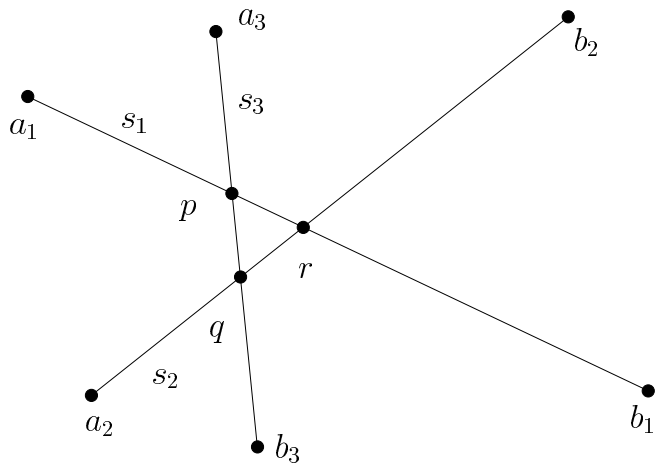

FIG. 1.5 – Incohérences potentielles de l'algorithme de balayage.

La figure 1.5 illustre un premier exemple d'inconhérence potentielle. Les segments s_1, s_2, et s_3 forment un triangle ayant pour sommets p, q, et r. L'ordre horizontal exact des points d'intersection est donné par $p <_{xy} q <_{xy} r$ et l'ordre vertical des segments à gauche de p par $s_2 < s_1 < s_3$. Pour des situations presque dégénérées, c'est-à-dire lorsque les coordonnées de ses sommets ne

diffèrent que par leur bit de poids le plus faible, les erreurs d'arrondi intervenant dans le calcul des coordonnées des points d'intersection p, q et r, peuvent mener à un traitement des événements selon un ordre qui n'est plus consistant et conduit dans la plupart des cas à une terminaison prématurée de l'algorithme. On peut en effet facilement montrer que parmi l'ensemble des cinq ordres différents de l'ordre exact, seul l'ordre $r <_{xy} q <_{xy} p$ ne mène pas à une terminaison inopinée de l'algorithme. L'ordre $p <_{xy} r <_{xy} q$ implique, par exemple, lors du traitement de l'événement r la transposition de deux segments (à savoir s_1 et s_2) qui ne sont pas adjacents dans le front de balayage et mène à une terminaison prématurée du programme.

Un autre cas de problème de robustesse est donné par l'exemple suivant. Supposons que l'abscisse calculée du point d'intersection p de la figure 1.5, découvert lors de l'insertion du segment s_3 dans le front de balayage (en a_3), est surestimée et telle que $b_3 <_{xy} p <_{xy} r$ (c'est-à-dire le point d'intersection est inséré au mauvais endroit dans la liste des événements). Les segments s_2 et s_3 n'étant à aucun moment adjacents dans le front de balayage, l'intersection entre ces deux segments n'est pas détectée. Le traitement de l'événement suivant b_3 cause la suppression du segment s_3 du front de balayage. Lors du traitement de l'événement p l'algorithme avorte en tentant d'inverser l'ordre des arêtes s_1 et s_3 dans la structure de donnée représentant le front des arêtes actives (le front de balayage ne contient plus ce segment).

Il existe un autre type de problèmes lié aux imprécisions numériques : le programme n'échoue pas mais donne un résultat faux ou incohérent. Ce type de situation est illustré par les figures 1.6.a et 1.6.b. La figure 1.6.a montre un cas d'oubli d'un point d'intersection lors du balayage. L'approximation de l'abscisse du point d'intersection entre les segments s_1 et s_2 calculée à l'aide de l'arithmétique flottante se révèle être plus petite que l'abscisse de l'extrémité gauche de s_3. Les segments s_3 et s_4 sont, quant à eux, correctement insérés respectivement dessous s_2 et dessus s_1 dans le front de balayage et l'intersection entre les segments s_3 et s_4 n'est pas détectée.

Dans la situation décrite par la figure 1.6.b, l'algorithme ne rencontre aucune incohérence durant son exécution, la construction approchée des points d'intersection mène cependant à la perte de la cohérence topologique de l'arrangement calculé (le plongement du graphe calculé n'est pas planaire). Dans les deux situations décrites, un autre algorithme utilisant le graphe calculé comme entrée et qui suppose la cohérence des données a alors de grandes chances d'être lui aussi confronté à des problèmes de fiabilité.

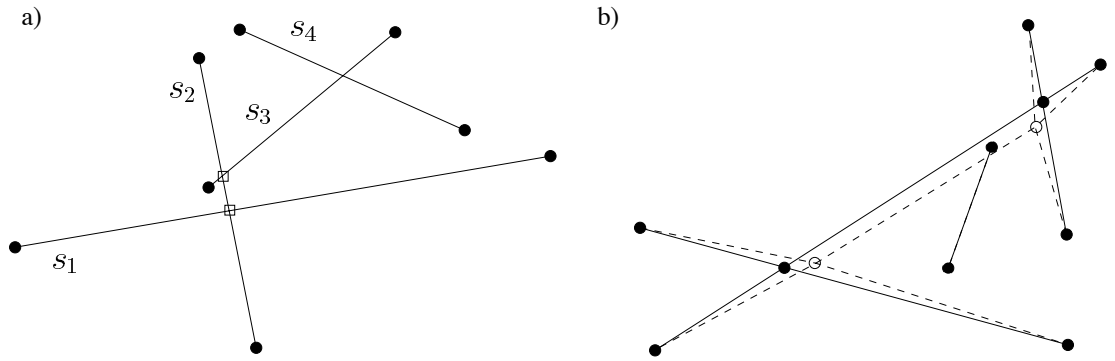

FIG. 1.6 – a) Exemple d'oubli d'un point d'intersection. b) Perte de la planarité de l'arrangement liée à la construction approchée des sommets d'intersection. L'arrangement exact est représenté en traits pleins, l'arrangement calculé est représenté en pointillés.

1.6 Arithmétique exacte rationnelle

L'utilisation d'une arithmétique exacte, en général d'une arithmétique rationnelle, est historiquement une des premières solutions utilisées pour résoudre les problèmes de précision numérique [52]. L'utilisation d'une telle arithmétique garantit que toutes les valeurs et toutes les expressions numériques intervenant dans l'algorithme sont évaluées et représentées exactement (et non plus en faisant confiance au calcul arrondi de l'arithmétique flottante). Chaque décision prise par l'algorithme est donc correcte et chaque construction est calculée exactement, ce qui permet de garantir que les théorèmes usuels seront vérifiés en machine. Ce type d'approche est véritablement l'une des premières solutions à s'inscrire dans le cadre plus général du *paradigme du calcul géométrique exact* [72] défini plus tard par Dubé et Yap.

Afin qu'une telle solution soit applicable, le choix de l'arithmétique à utiliser s'avère crucial et doit être adapté à la nature des expressions numériques à évaluer. L'évaluation exacte des primitives numériques d'un algorithme n'implique en effet pas forcément de savoir calculer exactement sur les réels. Dans la plupart des cas, une arithmétique exacte restreinte à un sous ensemble des réels est suffisante. Pour de nombreuses applications il est par exemple possible de substituer le calcul exact sur les nombres réels par une arithmétique entière ou rationnelle.

Une illustration est donnée par le problème du balayage d'un ensemble de segments dans le plan. Dans ce cas, l'adoption d'une représentation rationnelle des sommets en entrée et l'utilisation d'une arithmétique rationnelle dans le programme suffit à garantir la robustesse de l'algorithme de balayage. Les primitives numériques utilisées par l'algorithme de Bentley-Ottmann sont en effet les suivantes :

- $(p <_{xy} q)$? (c'est-à-dire $(p <_x q$ et $p <_y q)$?) pour trier lexicographiquement les points en entrée.
- $(p <_y qr)$? pour maintenir la liste ordonnée des segments coupés par la droite de balayage, une formule exprimant ce test est le test d'orientation (cf. section 2.1.2) que nous retrouverons tout au long de cette thèse, et qui est très utilisé en géométrie algorithmique.
- $pq \cap rs \neq \emptyset$ pour tester l'intersection de deux segments. Dans le cas où les deux segments se coupent en un point unique, le point d'intersection $i = pq \cap rs$ peut être calculé à l'aide de la formule suivante
$$i = p + (q - p)\frac{pr \times ps}{pq \times sr}$$
où $\times$ désigne le produit vectoriel.
- $(p <_{xy} (qr \cap st))$?, $((pq \cap rs) <_{xy} (pq \cap tu))$? et $((pq \cap rs) <_{xy} (tu \cap vw))$? pour maintenir la liste triée des événements.

En plus des segments originaux, l'algorithme manipule donc essentiellement les coordonnées de points d'intersection qui s'expriment sous la forme de fonctions rationnelles des données en entrée. En adoptant une représentation rationnelle des coordonnées des extrémités des segments en entrée, l'évaluation et la comparaison exacte de nombres rationnels permet donc une mise à jour correcte du front de balayage ainsi qu'un traitement des événements dans un ordre cohérent (puisque exact).

En pratique, un nombre rationnel $\frac{p}{q}$ est représenté par un couple d'entier (p, q). Chacune des opérations est alors programmée selon les règles classiques de l'arithmétique rationnelle (par exemple, $\frac{p}{q} + \frac{r}{s} = \frac{ps+rq}{qs}$, $\frac{p}{q} \times \frac{r}{s} = \frac{pr}{qs}$, $\frac{p}{q} \div \frac{r}{s} = \frac{ps}{qr}$, ...). Afin de contourner le problème de la représentation des entiers sur un nombre fini de bits (typiquement les entiers de la machine sont codés sur 32 ou 64 bits), une bibliothèque logicielle proposant des entiers de longueur variable est toutefois nécessaire. L'idée est d'allouer de manière dynamique la mémoire nécessaire à la représentation d'un entier (typiquement, un tableau de taille k bits), on n'a alors pas d'autre limite que la taille mémoire de la machine, on parle alors d'entier grande précision ou d'entier multiprécision. Les entiers multiprécisions proposés par les bibliothèques LEDA (*Library of Efficient Data types and Algorithms*) [51] et GMP (*Gnu Multiple Precision arithmetic library*) [34], par exemple, permettent ainsi d'avoir une représentation exacte du corps des nombres rationnels.

Les opérations de l'arithmétique en grande précision sont bien évidemment plus lentes que les opérations à précision fixée de la machine. Le temps d'exécution de chaque opération dépend désormais de la taille des opérandes : l'addition et la soustraction de deux entiers se font par exemple en temps linéaire dans le nombre de bits nécessaires à la représentation binaire des

opérandes, pour la multiplication les principaux algorithmes sont l'algorithme de Karatsuba qui opère en temps $O(n^{\log_2 3})$, celui de Toom-Cook en $O(n^{\log_3 5})$, enfin l'algorithme de Schönhage en $O(n \log n \log\log n)$. De plus, les opérations sur les nombres grande précision sont codées dans des bibliothèques logicielles et non plus cablées dans l'unité arithmétique du processeur.

Une alternative communément admise (tout au moins dans certains contextes limités) au calcul géométrique exact est l'*heuristique des epsilons* [37] qui consiste à considérer qu'une valeur est nulle si elle est plus petite en valeur absolue qu'une certaine quantité fixée. Cependant, la gestion de ces quantités s'avère très rapidement problématique pour certains algorithmes.

1.7 Limitations du calcul exact

Le paradigme du calcul géométrique exact est donc une solution générale qui permet apparemment de résoudre tous les problèmes de manque de fiabilité des programmes géométriques. Le point le plus délicat de cette approche est cependant de la rendre efficace.

Dans le cas des rationnels grande précision utilisés pour notre exemple, le principal problème se pose au niveau de la taille des nombres utilisés. Une telle implantation fait, en effet, systématiquement croître le dénominateur et par conséquent le nombre de chiffres manipulés au cours du calcul. Une solution à ce problème consiste à contenir cette croissance du nombre de chiffres au strict nécessaire en exprimant systématiquement le résultat d'une opération sous forme de fraction irréductible. Cette transformation se traduit cependant par un temps de calcul plus important à chaque opération.

Cette croissance de la taille des nombres manipulés est d'autant plus problématique que le résultat (exact) de certains algorithmes est souvent cascadé, c'est-à-dire réutilisé comme entrée d'un autre programme (cf. section 1.11). Les modeleurs géométriques éditent, par exemple, des formes qui sont réutilisées et modifiées un grand nombre de fois ; la complexité numérique des valeurs manipulées est dans ce cas directement proportionnelle au nombre de modifications effectuées qui n'est souvent pas borné a priori. L'utilisation d'une telle arithmétique ne permet plus par conséquent de vérifier l'hypothèse du modèle Real-RAM selon laquelle chaque opération arithmétique peut être effectuée en temps constant, et remet ainsi en cause les garanties de complexité théorique des algorithmes.

De plus, bien que les nombres rationnels offrent un sous ensemble des réels suffisant pour de nombreux calculs géométriques, ils ne sont toutefois pas adaptés à certains problèmes pratiques. Ce type d'arithmétique ne permet pas par exemple d'opérer de rotations ou de calcul de distance. Une rotation d'un angle $r\pi$ avec $r \in \mathbb{Q}$ d'un segment du plan introduit potentiellement des

fonctions trigonométriques dans l'expression de ces sommets, le calcul d'une distance des racines carrées, qui nécessitent de savoir représenter et manipuler les nombres algébriques.

L'adoption du paradigme du calcul exact en géométrie algorithmique a ouvert tout un champ de recherche en géométrie algorithmique. L'approche générale de ces travaux consiste à analyser plus en détail les primitives numériques intervenant dans les algorithmes géométriques afin d'évaluer plus précisément leur influence sur la robustesse de l'implémentation et ainsi rendre le calcul exact le plus efficace possible. La démarche qui sous-tend ces études consiste classiquement à distinguer deux types de primitives numériques intervenant dans un algorithme géométrique : les *prédicats* et les *constructeurs*.

1.8 Qu'est-ce qu'un prédicat ?

Les algorithmes géométriques peuvent être découpés en deux parties logiques distinctes : une partie combinatoire et une partie numérique. La partie combinatoire manipule des structures discrètes (par exemple le graphe d'incidence de l'arrangement) alors que la partie numérique manipule des nombres (typiquement les coordonnées des objets géométriques). Classiquement, l'entrée d'un algorithme est purement numérique (un ensemble de segments non structuré dans le plan) et le but de l'algorithme est alors de calculer une structure combinatoire ou une construction numérique à partir de ces données de départ.

Le rôle d'un prédicat est d'assurer l'interface entre objets numériques et structure combinatoire, il correspond à une question géométrique élémentaire que se pose l'algorithme combinatoire pour prendre ces décisions et calculer le résultat. Les prédicats sont associés aux branchements de l'algorithme et conditionnent en ce sens complètement le déroulement et la validité de celui-ci. L'évaluation exacte des prédicats est donc cruciale pour la robustesse des algorithmes géométriques.

En pratique, un prédicat est une fonction qui prend en argument des objets géométriques simples (par exemple des points) et retourne une valeur discrète de type booléen ou énuméré. Un exemple typique de prédicat utilisé par l'algorithme de Bentley-Ottmann est donné par le prédicat de comparaison $<_{xy}$ qui compare les coordonnées des événements à traiter ou le test d'intersection qui détermine si deux segments donnés se coupent. Le concept de prédicat s'oppose à un autre type de primitive numérique, que l'on appelle *constructeur*, et qui sera introduit à la section 1.10.

La modélisation d'algorithmes géométriques à l'aide de prédicats permet de bénéficier des avantages suivants. Du point de vue de la conception d'algorithmes, tout d'abord, l'utilisation

de prédicats permet de se placer dans le cadre de la géométrie classique (*i.e.* continue) et offre ainsi une formulation naturelle des tests à effectuer dans un algorithme. D'un point de vue plus pratique, les prédicats isolent les calculs numériques et les problèmes de robustesse associés dans des primitives bien définies. Les problèmes d'instabilité numérique sont de cette manière encapsulés dans ces primitives et peuvent être résolus localement à l'aide d'outils arithmétiques adéquats.

Évaluer un prédicat consiste généralement à déterminer le signe $(-, 0, +)$ d'une expression voire d'une combinaison booléenne d'expressions polynomiales (un prédicat de comparaison $a < b$ revient en fait à tester $a - b < 0$). Une manière de mesurer la *complexité* d'un prédicat consiste à considérer celui-ci comme une combinaison de polynômes multivariés dont les arguments font partie d'un sous-ensemble des variables en entrée de l'algorithme (typiquement des coordonnées de points). Le *degré* d'un prédicat correspond alors au maximum des degrés algébriques des polynômes intervenant dans son expression.

Il existe un lien direct entre le degré d'un prédicat et la complexité de son calcul exact. Le degré d'une expression permet en effet d'évaluer approximativement quelle est la précision minimale nécessaire à un calcul exact avec des entiers multiprécisions : le calcul d'un prédicat de degré 2 nécessitera par exemple approximativement deux fois le nombre de bits nécessaires à la représentation exacte de ces données en entrée.

Comme nous le verrons par la suite, l'évaluation exacte des prédicats intervenant dans un algorithme ne nécessite pas forcément le recours à un type d'arithmétique exacte multiprécision (c'est-à-dire de savoir représenter et calculer exactement les valeurs intervenant dans ces primitives). On peut, par exemple, montrer que certains prédicats, comme le signe de déterminants, peuvent s'évaluer en simple précision de manière fiable et efficace [2]. Pour des prédicats un peu plus compliqués, une solution consiste à ne faire dans ce cas les calculs exacts que si nécessaire : avant de recourir au calcul exact on utilise un filtre arithmétique qui est bien plus rapide que le calcul exact et très souvent suffisant.

1.9 Prédicats et filtres arithmétiques

Comme nous venons de le voir, un prédicat consiste à évaluer le signe d'une expression P en un point x où x est un ensemble de valeurs décrivant une instance du problème, par exemple des coordonnées de points. Pour garantir la robustesse d'un algorithme utilisant ce prédicat, on veut que cette évaluation soit exacte.

Puisque le calcul arrondi ne nous offre aucune garantie sur l'exactitude du résultat, une so-

lution naturelle consiste alors à utiliser une arithmétique exacte adaptée à la nature de P et x pour calculer $P(x)$ puis prendre son signe. Par exemple, si P est un polynôme et x un vecteur d'entiers on utilisera une arithmétique exacte entière. Malheureusement, de telles arithmétiques sont lentes et la solution du tout exact est très coûteuse et mène à des temps de calcul souvent jugés prohibitifs.

Le prix du calcul exact semble d'autant plus exagéré que le calcul approché fournit très souvent la bonne réponse et ne se trompe que dans quelques cas rares. Une approche raisonnable consiste à utiliser un calcul approché afin de filtrer ces cas faciles et de n'utiliser le calcul exact que lorsque c'est nécessaire. Le principe d'un filtre arithmétique consiste à associer au calcul approché un test de certification qui permet soit de garantir l'exactitude du résultat calculé, soit de détecter un cas limite qui nécessite l'utilisation d'une arithmétique appropriée.

Un filtre arithmétique est donc composé d'un évaluateur approché qui calcule une approximation $\widetilde{P}(x)$ de $P(x)$ et d'un certificateur. Le certificateur consiste classiquement à accompagner l'évaluation approchée d'une majoration $\epsilon_{P(x)}$ sur l'erreur d'approximation commise. À partir de cette erreur, un intervalle autour de $\widetilde{P}(x)$ contenant la valeur exacte de $P(x)$ peut être déduit. Si cet intervalle ne contient pas zéro, le signe de la valeur exacte est le même que celui de la valeur approchée calculée. Dans le cas contraire, l'erreur calculée ne permet pas de conclure de manière sûre, on fait alors appel à un autre filtre et à un calcul plus coûteux.

Dans la grande majorité des cas, le calcul flottant suffit pour conclure (le signe de la valeur approchée est bien celui de la valeur exacte) et l'arithmétique exacte n'est que rarement invoquée. Globalement, la perte de performance est minime. Tout comme le choix de l'arithmétique nécessaire à l'évaluation exacte du signe de $P(x)$, les différents filtres dépendent de la formulation de P et également du type des données x. La manière dont la borne d'erreur est calculée donne lieu à plusieurs types de filtres [59].

Filtres statiques Le principe des filtres dits statiques est d'obtenir une majoration de l'erreur d'approximation a priori. Dans le cas le plus courant, la valeur approchée $\widetilde{P}(x)$ de $P(x)$ est simplement obtenue avec l'arithmétique IEEE en mode d'arrondi au plus près. Une borne supérieure sur l'erreur d'approximation est déduite à partir d'hypothèses spécifiques sur le domaine de définition des données x, par exemple, les données en entrée sont des entiers provenant d'un intervalle borné.

À chaque prédicat est donc associé une borne supérieure M sur les valeurs absolues des composantes de x. À partir de cette borne et de la précision de l'arithmétique utilisée pour calculer l'approximation, un majorant $\epsilon_P(M)$ sur l'erreur du calcul flottant indépendant de x est alors

calculé en appliquant les formules de propagation d'erreurs de la section 1.4 sur des majorations de tous les résultats intermédiaires du calcul de $P(x)$. Le code associé au calcul filtré de $P(x)$ compare alors les valeurs de $\widetilde{P}(x)$ et de $\epsilon_P(M)$: si $|\widetilde{P}(x)| < \epsilon_P(M)$, le signe de $\widetilde{P}(x)$ est le même que celui de $P(x)$ sinon un calcul plus précis est demandé.

L'avantage des filtres statiques provient du fait que le calcul d'un majorant sur l'erreur est effectué une fois pour toute à la compilation (par le programmeur ou le précompilateur) ce qui garantit une certification rapide. L'inconvénient majeur est la nécessité de connaître une majoration sur la valeur absolue des données. Cette majoration doit être valable pour toutes les instances en entrée possibles. Le plus souvent le majorant n'est donc pas optimal et le taux d'échec (c'est-à-dire le nombre de fois où le certificateur ne permet pas de conclure) est élevé. De plus, les filtres statiques ne permettent pas de traiter les expressions contenant des divisions (cf. section 1.4).

Filtres dynamiques Une alternative intéressante aux filtres statiques consiste à utiliser des filtres dits dynamiques. L'idée des filtres dynamiques est de s'abstraire de la contrainte de majoration des composantes de x. Leur principe est de mener parallèlement à chaque opération arithmétique, le calcul d'un résultat approché et le calcul d'une borne sur l'erreur (à l'aide d'une arithmétique d'intervalle ou d'un calcul de propagation d'erreurs par exemple). La borne d'erreur est dans ce cas calculée à partir des valeurs effectives des variables en entrée, c'est-à-dire en utilisant les informations disponibles à l'exécution du prédicat.

Les filtres dynamiques sont très faciles à utiliser et sont très puissants, c'est-à-dire qu'ils ont un excellent taux de succès. Cependant, ils restent relativement chers (jusqu'à un facteur quatre sur le temps de calcul d'une expression) puisqu'ils nécessitent de contrôler les erreurs d'arrondis à l'exécution pour chaque opération arithmétique tout au long du calcul.

Entre le cas du filtre statique où une borne sur l'erreur est calculée à l'avance en fonction d'hypothèses sur les données et le cas du filtre dynamique ou l'erreur est majorée à chaque opération arithmétique menant au calcul de $\widetilde{P}(x)$, il existe toute une variété de solutions intermédiaires.

Filtres adaptatifs L'idée des filtres adaptatifs est de calculer une borne sur l'erreur qui dépende à nouveau de x, mais cette fois à l'aide d'une formule globale et non plus pas à pas et au fur et à mesure du calcul de $P(x)$. On obtient ainsi une expression du type $|\widetilde{P}(x) - P(x)| \leq E_P(x)$. Généralement les expressions de P et de E_P sont assez similaires et certaines parties du calcul de ces deux expressions seront communes limitant ainsi le surcoût lié au calcul de $E_P(x)$.

Un moyen d'obtenir ce résultat est de considérer le polynôme $|P|$ obtenu en remplaçant dans P toutes les soustractions par des additions et toutes les variables par leur valeur absolue (par

exemple si $P(x,y) = 3x^2 - 2xy$ alors $|P|(x,y) = 3x^2 + 2|xy|$). On cherche alors à exprimer $E_P(x)$ sous la forme $c|P|$.

Filtres semi-statiques Une variante des filtres statiques consiste à s'abstraire de la contrainte de majoration des composantes de x. Ainsi au lieu de calculer M une seule fois à la compilation, on peut mettre à jour cette borne durant l'exécution à chaque appel du prédicat. La borne est ajustée automatiquement en fonction de x. On parle alors de filtres semi-statiques.

Filtres statiques adaptatifs Le prédicat ne fait que vérifier que les données sont plus petites que les bornes courantes. Cette contrainte est vérifiée à chaque appel du prédicat. Si x n'est pas conforme alors la borne M est réévaluée à la composante de x ayant la plus grande valeur absolue et l'erreur d'arrondi est recalculée. Sinon, les bornes d'erreurs restent valides bien que légèrement surestimées.

Efficacité des filtres Un calcul filtré opère donc en deux phases, celui-ci effectue dans un premier temps un calcul approché et un calcul d'erreur, puis, en cas d'échec du filtre, un calcul exact. L'efficacité d'un calcul filtré se mesure donc à l'aide de trois paramètres, le temps de calcul du filtre T_{filtre}, le temps de calcul par la méthode exacte T_{exact} et le taux d'échec du filtre $P_{\mathit{échec}}$. La complexité d'un calcul filtré est calculée à l'aide de la formule suivante :

$$T_{\mathit{filtre}} + P_{\mathit{échec}} \times T_{\mathit{exact}}.$$

Les deux premiers paramètres dépendent essentiellement de l'arithmétique et de la technique de calcul d'erreur utilisées ainsi que de la formule P que l'on doit évaluer. Le temps de calcul de $P(x)$ avec une arithmétique approchée sera uniquement sensible au nombre d'opérations arithmétiques utilisées, alors que le temps de calcul avec une arithmétique exacte dépendra aussi fortement de la taille des opérandes intervenant dans le calcul et sera donc également sensible au degré du polynôme P. Le taux d'échec du filtre dépend quant à lui étroitement de la configuration précise x que l'on étudie, et est donc beaucoup plus délicat à évaluer [13, 17].

Filtres cascadés Le principe des filtres cascadés est d'utiliser plusieurs filtres arithmétiques à la suite. L'idée est d'utiliser en premier lieu le filtre le plus rapide mais le moins précis, puis ensuite les filtres moins efficaces mais ayant un meilleur taux de succès. Cette approche de filtres cascadés semble donner les meilleures performances moyennes sur de nombreux jeux de données [16].

Les nombreux types de filtres apportent donc une solution robuste et efficace (en termes de vitesse et de taux d'échec) à l'évaluation exacte du signe des prédicats. Il existe de plus de nombreuses implantations avec des outils de génération automatique [59, 16] dans des bibliothèques géométriques comme CGAL [8]. Dans les cas où le calcul exact est inévitable (échec du filtre) ces prédicats filtrés ont souvent recours à des bibliothèques multiprécision telles que LEDA ou GMP qui couvrent une grande partie des besoins du calcul géométrique.

1.10 Qu'est-ce qu'un constructeur ?

Certains algorithmes ne calculent pas uniquement une structure combinatoire à partir de données numériques en entrée, mais effectuent également des constructions géométriques, c'est-à-dire produisent des quantités numériques. On ne s'intéresse ainsi plus uniquement à l'aspect combinatoire du résultat (par exemple les relations d'incidences entre les segments dont on désire calculer l'arrangement) mais aussi au plongement géométrique de celui-ci (par exemple les coordonnées effectives de tous les sommets de l'arrangement).

Un constructeur est une primitive numérique qui effectue une construction de base sur des objets géométriques élémentaires. Par exemple, étant donnés deux segments se coupant dans le plan on désire calculer leur intersection. Les constructeurs sont nécessaires à la production du résultat des algorithmes géométriques qui construisent de nouveaux objets puisqu'ils permettent le plongement géométrique explicite de leur résultat. En pratique, un constructeur est une fonction qui prend en argument des objets géométriques simples (segments, points,. . .) et retourne une ou plusieurs quantités numériques représentant le nouvel objet construit (par exemple, l'ensemble de ses coordonnées).

1.11 Constructions et régularisation

Ainsi, contrairement au cas des prédicats, le résultat d'un constructeur n'est pas un choix (*i.e.* une valeur discrète) mais un résultat potentiellement réel et nécessite par conséquent l'utilisation d'un type de nombre adapté afin d'être représenté exactement. L'exactitude d'une telle représentation est le plus souvent cruciale, notamment lorsque l'on réutilise un objet construit en entrée d'un autre algorithme. Une construction approchée peut en effet provoquer la perte de certaines propriétés vérifiées par la construction exacte et ainsi introduire des incohérences dans le déroulement d'un algorithme qui se base sur ces propriétés. La représentation exacte de ces constructions pose toutefois elle aussi certains problèmes dans le cas d'algorithmes cascadés, c'est dans ce cas

l'espace mémoire et le temps de calcul nécessaires aux algorithmes qui constituent un obstacle.

Pour illustrer ce problème, considérons par exemple le calcul d'opérations booléennes de régions polygonales dans le plan qui constitue un problème fondamental dans le domaine de la conception assistée par ordinateur. Étant donnés deux régions polygonales A et B dans le plan, c'est-à-dire deux régions du plan composées de polygones simples ayant un intérieur et un extérieur bien définis, l'intersection de leurs intérieurs se déduit facilement du calcul de l'arrangement de leurs arêtes. Précisément, la région d'intersection correspond aux faces de l'arrangement qui appartiennent aux deux régions initiales. Supposons maintenant que les sommets des deux régions en entrée sont issus de la grille entière et donnés sous leur forme cartésienne par deux entiers codés sur p bits. Les sommets construits par l'algorithme (*i.e.* les sommets de l'arrangement qui correspondent à l'intersection d'une arête de A avec un arête de B) nécessitent alors un type de nombre et une arithmétique rationnelle multiprécision pour être calculés et représentés exactement. Plus précisément, les coordonnées (x, y) d'un tel sommet sont données par le rapport de deux déterminants qui nécessite respectivement $3p + 3$ et $2p + 2$ bits pour être représentés. Supposons maintenant, que l'on utilise le résultat C de l'opération d'intersection de A et B, et une autre région D (elle aussi issue d'une opération booléenne entre deux régions polygonales dont les coordonnées sont codées sur p bits) comme opérandes d'une nouvelle opération d'intersection $C \cap D$. Les sommets de l'arrangement associé nécessitent cette fois jusqu'à $9p + 9$ bits pour être représentés exactement. On peut ainsi montrer que le nombre de bits nécessaires à la représentation rationnelle (exacte) des sommets d'une région polygonale croît exponentiellement avec la hauteur de son arbre de construction, c'est-à-dire avec le nombre d'opérations effectuées dans l'histoire de la construction de cette région [31, 55]. Par conséquent, le temps de calcul et l'espace mémoire nécessaire à l'exécution de l'algorithme augmentent considérablement et ce quelque soit la représentation exacte (rationnelle ou implicite [28, 45]) des sommets utilisée.

Ainsi, dans la presque totalité des cas un système géométrique doit utiliser des méthodes d'arrondi après chaque étape de construction afin de limiter la complexité numérique et les besoins en espace mémoire. Les méthodes d'arrondi (ou de régularisation) réduisent le degré des constructions à un degré un en calculant une valeur approchée de la construction exacte. Cette approximation rend inévitable la perte de certaines propriétés inhérentes à la construction (par exemple le point d'intersection de deux segments n'appartient plus forcément aux segments) mais est choisie de manière à conserver d'autres propriétés topologiques souhaitables, comme par exemple, la planarité de l'arrangement de segments calculé. Les méthodes d'arrondi nécessitent par conséquent de prouver la préservation de ces propriétés. Ces preuves ne sont pas triviales et ne peuvent pas toujours être généralisées. Ainsi, chaque application doit être considérée séparément.

Plusieurs méthodes d'arrondi ont été mises au point afin de plonger des constructions exactes dans un ensemble de valeurs entières tout en préservant certaines propriétés de la structure. Ces algorithmes portent sur l'arrondi de l'arrangement de segments dans le plan [35, 32, 36, 42, 58], le diagramme de Voronoï dans le plan [14] et l'arrangement de triangles dans l'espace [27]. Le cadre général de ces approches suppose de savoir calculer le plus proche point entier de constructions géométriques élémentaires. Typiquement, le paradigme du *Snap Rounding* qui permet d'arrondir un arrangement de segments du plan, résout les problèmes de robustesse en perturbant l'ensemble des sommets de l'arrangement au plus proche point entier. La validité de la méthode dépend ainsi des constructeurs élémentaires utilisés qui doivent satisfaire à des spécifications d'arrondi précises (cf. Chapitre 3).

Chapitre 2

Tests d'intersection rapides et robustes

Les problèmes de robustesse des algorithmes géométriques ont été largement étudiés ces dernières années [70]. Il ressort de ces travaux qu'une bonne solution consiste en un modèle qui sépare le plus possible les tests géométriques (*prédicats*) de la partie combinatoire de l'algorithme. De cette manière, les problèmes de robustesse dûs aux erreurs numériques sont isolés dans les prédicats et peuvent être résolus à l'aide d'outils arithmétiques appropriés.

De ce point de vue, un prédicat est un sous-programme qui répond à une question géométrique élémentaire. Celui-ci a typiquement deux aspects indépendants. Le premier aspect est une formulation algébrique du problème, le plus souvent représentée par une combinaison booléenne d'expressions algébriques dont le signe donne la réponse au test géométrique. Le deuxième aspect est le type d'arithmétique utilisé afin d'évaluer la valeur de cette expression : il peut s'agir d'une arithmétique approchée du type IEEE, qui est rapide mais inexacte, ou d'un type d'arithmétique plus élaboré qui permet de certifier l'exactitude du résultat.

Pour exprimer un même prédicat géométrique, plusieurs formules algébriques peuvent parfois être utilisées, et on aimerait déterminer la plus efficace. Dans ce sens, plusieurs critères ou quantités peuvent être utilisés afin d'évaluer la qualité d'un prédicat. Dans un modèle Real-RAM, la mesure serait le nombre d'opérations arithmétiques, dans l'optique d'une évaluation exacte du prédicat, une mesure supplémentaire à prendre en compte est le degré de l'expression algébrique utilisée. Le degré arithmétique d'un prédicat [49] est défini comme étant le plus haut degré des polynômes apparaissant dans sa formulation algébrique. Cette mesure permet de connaître approximativement quelle est la précision minimale nécessaire au calcul exact du prédicat. Un prédicat de bas degré a ainsi plusieurs avantages : utilisé avec une arithmétique approchée, il permet une meilleure précision, utilisé avec une arithmétique exacte, il permet de limiter les exigences arithmétiques étant donné que moins de bits sont alors nécessaires pour représenter les nombres.

Ce chapitre présente des algorithmes rapides répondant au prédicat : les objets convexes A et B s'intersectent-ils ? où l'on désigne par objet un polygone planaire plongé dans l'espace bi- ou tridimensionnel.

Les différents prédicats proposés se basent exclusivement sur l'évaluation de tests d'orientation sur les objets en entrée, voire la comparaison des coordonnées (supposées exactes) des objets en entrée. Le test d'orientation est abondamment utilisé dans les algorithmes géométriques et il existe de nombreuses techniques [2, 69] permettant d'évaluer celui-ci exactement pour un coût raisonnable (cf. section 1.9).

Le suite du chapitre est organisée de la manière suivante. La première section introduit quelques notations et rappelle la définition des tests d'orientation bi- et tridimensionnel. La deuxième et la troisième sections présentent une nouvelle formulation du test d'intersection de deux triangles bi- et tridimensionnels ne nécessitant qu'un petit nombre d'appels au test d'orientation[1]. Enfin les deux dernières sections proposent une généralisation au cas de deux polygones convexes.

2.1 Notations et définitions

2.1.1 Notations

Dans la suite, A et B désignent deux polygones planaires ayant respectivement pour sommets $\mathbf{a}_1, \ldots, \mathbf{a}_n$ et $\mathbf{b}_1, \ldots, \mathbf{b}_m$, chaque sommet étant représenté par ses coordonnées cartésiennes. On suppose que A et B sont strictement convexes, par conséquent trois sommets d'un polygone donné ne peuvent être colinéaires. Une arête est représentée par ses deux points extrémités. On note $\mathbf{ab}$ le segment orienté de $\mathbf{a}$ vers $\mathbf{b}$ et $(\mathbf{ab})$ la droite support à $\mathbf{ab}$ (c'est-à-dire la droite contenant $\mathbf{ab}$) orienté de $\mathbf{a}$ vers $\mathbf{b}$. Enfin, π_A et π_B désignent respectivement les plans support aux polygones A et B, et L la droite d'intersection, si elle existe, des plans π_A et π_B. On note finalement respectivement $\mathbf{u} \cdot \mathbf{v}$ et $\mathbf{u} \times \mathbf{v}$ le produit scalaire et le produit vectoriel des vecteurs $\mathbf{u}$ et $\mathbf{v}$.

2.1.2 Orientation de trois points dans le plan

Étant donnés trois points du plan $\mathbf{a} = (a_x, a_y)$, $\mathbf{b} = (b_x, b_y)$ et $\mathbf{c} = (c_x, c_y)$, le prédicat d'orientation bidimensionnel détermine de quel côté de la droite orientée $(\mathbf{ab})$ le point $\mathbf{c} = (c_x, c_y)$ se

[1] Ce dernier prédicat a fait l'objet de l'article [38].

situe. Une formulation possible de ce prédicat est la suivante :

$$[\mathbf{a},\mathbf{b},\mathbf{c}] := \begin{vmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \\ 1 & 1 & 1 \end{vmatrix} = \begin{vmatrix} a_x - c_x & b_x - c_x \\ a_y - c_y & b_y - c_y \end{vmatrix} = \mathbf{ab} \times \mathbf{ac} \tag{2.1}$$

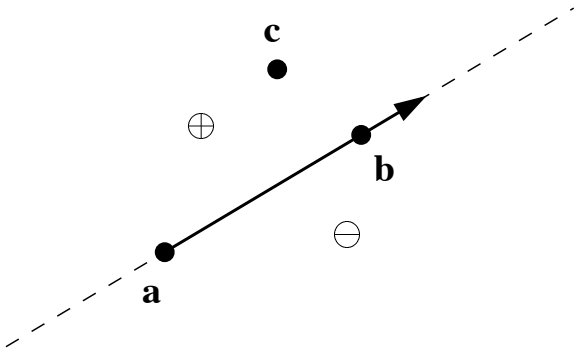

FIG. 2.1 – $[\mathbf{a},\mathbf{b},\mathbf{c}]$ est positif si le triplet de point $\mathbf{a},\mathbf{b},\mathbf{c}$ est dans le sens direct.

Le signe de $[\mathbf{a},\mathbf{b},\mathbf{c}]$ est positif si le triplet de point $\mathbf{a},\mathbf{b},\mathbf{c}$ est dans le sens direct (c'est-à-dire le point $\mathbf{c}$ est à gauche de la droite orienté $(\mathbf{ab})$), négatif si le le triplet de point $\mathbf{a},\mathbf{b},\mathbf{c}$ est dans le sens indirect (le point $\mathbf{c}$ est à droite de $(\mathbf{ab})$) ou zéro si les trois points sont colinéaires (cf. figure 2.1).

2.1.3 Orientation de quatre points dans l'espace

Étant donnés quatre points tridimensionnels $\mathbf{a} = (a_x, a_y, a_z),\ \mathbf{b} = (b_x, b_y, b_z),\ \mathbf{c} = (c_x, c_y, c_z)$, et $\mathbf{d} = (d_x, d_y, d_z)$, on définit

$$[\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}] := -\begin{vmatrix} a_x & b_x & c_x & d_x \\ a_y & b_y & c_y & d_y \\ a_z & b_z & c_z & d_z \\ 1 & 1 & 1 & 1 \end{vmatrix} = (\mathbf{d}-\mathbf{a}) \cdot ((\mathbf{b}-\mathbf{a}) \times (\mathbf{c}-\mathbf{a})) \tag{2.2}$$

Le signe de $[\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}]$ a deux interprétations géométriques, chacune correspondant à une règle de la main droite. Il détermine si le point $\mathbf{d}$ est dessus, dessous ou sur le plan défini par les points $\mathbf{a}$, $\mathbf{b}$, et $\mathbf{c}$, où dessus désigne la direction pointée par $((\mathbf{b}-\mathbf{a}) \times (\mathbf{c}-\mathbf{a}))$. De manière équivalente, le signe de $[\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}]$ indique si une vis orientée de $\mathbf{a}$ vers $\mathbf{b}$ se visse dans la direction de $\overrightarrow{\mathbf{cd}}$ ou

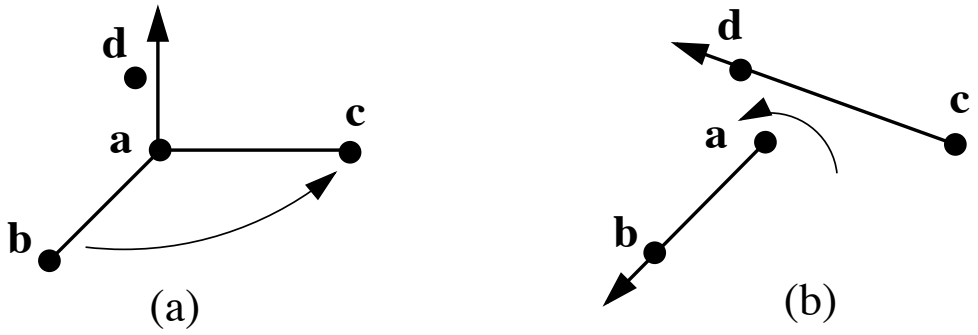

FIG. 2.2 – $[\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}]$ est positif selon la règle de la main droite pour les plans et les droites.

dans le sens contraire (cf. figure 2.2). Pour chacune des interprétations, le résultat est zéro si et seulement si les quatre points sont coplanaires.

Dans la suite, nous distinguons pour chaque test d'orientation le cas où la valeur du déterminant est négative, positive ou nulle ; nous entendons donc par signe l'une de trois valeurs discrètes -1, $+1$, ou 0 et considérons que deux signes sont différents si et seulement si ils ne correspondent pas tous les deux à une seule de ces valeurs discrètes.

La prochaine section aborde le problème de la détection d'intersection entre deux triangles tridimensionnels. Ce test d'intersection particulier est fondamental pour de nombreux algorithmes géométriques comme par exemple les algorithmes de détection de collision.

2.2 Test d'intersection Triangle/Triangle 3D

Soit $A = \Delta\ \mathbf{a}_1\mathbf{a}_2\mathbf{a}_3$ et $B = \Delta\ \mathbf{b}_1\mathbf{b}_2\mathbf{b}_3$ deux triangles tridimensionnels (cf. figure 2.3). Cette section est organisée de la manière suivante : la sous section 2.2.1 présente dans un premier temps les principaux algorithmes développés pour déterminer si A et B s'intersectent, la sous section 2.2.2 décrit ensuite une nouvelle formulation de l'algorithme, enfin la dernière sous section analyse comparativement l'ensemble des méthodes en termes d'efficacité et de robustesse.

2.2.1 Méthodes existantes

Nous étudions dans la suite les deux principales méthodes existantes. Les solutions apportées par chacune d'elles sont fondamentalement différentes. La première solution exposée est appelée

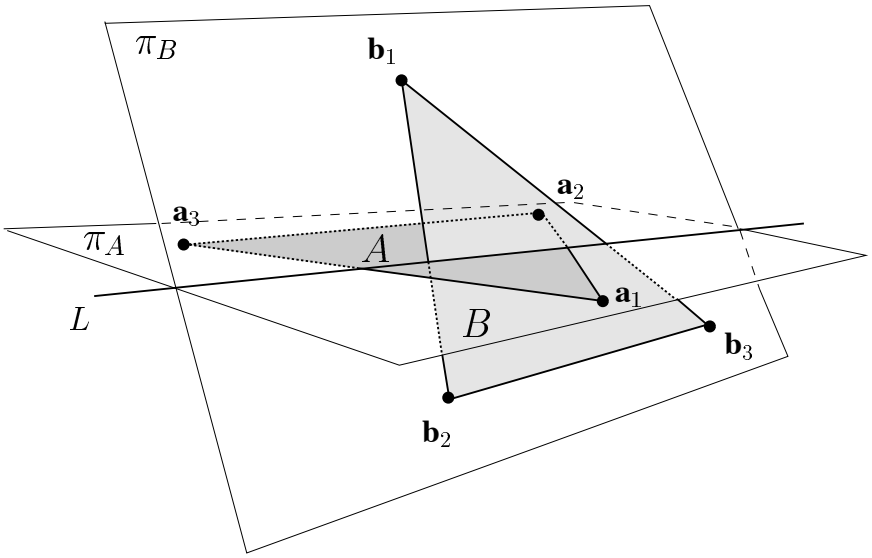

FIG. 2.3 – Les triangles et leur plan support.

interval overlap method et a été introduite par Möller [57]. La deuxième méthode provient du package ERIT [41]. Ces deux méthodes sont comparables en termes d'efficacité et s'avèrent être les plus rapides pour cette tâche.

Interval Overlap Method

La méthode proposée par Möller calcule dans un premier temps, l'équation du plan

$$\pi_B : \mathbf{n}_B \cdot \mathbf{x} + d_B = 0$$

supportant le triangle B, où $\mathbf{n}_B = (\mathbf{b}_2 - \mathbf{b}_1) \times (\mathbf{b}_3 - \mathbf{b}_1)$ et $d_B = -n_B \cdot \mathbf{b}_1$.

Les distances (signées) $d_{\mathbf{a}_i}$, $i = 1, 2, 3$, des sommets de A au plan π_B (au facteur $||\mathbf{n}_B||^2$ près) sont alors calculées en substituant les sommets dans l'équation du plan. À ce stade, si aucun des sommets n'appartient au plan π_B (c'est-à-dire aucune des distances n'est égale à zéro) et si les trois distances ont même signe alors A est entièrement contenu dans un des demi-espaces ouverts induits par π_B et les triangles ne peuvent s'intersecter.

Le même test est alors effectué avec B et π_A. Si l'ensemble des distances calculées est égal à zéro, les deux triangles sont coplanaires et ce cas est traité séparément comme cas spécial. Dans le cas contraire, l'intersection de π_A et π_B est la droite $L(t) = \mathbf{o} + t\mathbf{d}$ où $\mathbf{d} = \mathbf{n}_A \times \mathbf{n}_B$ est la direction de la droite et $\mathbf{o}$ un point appartenant à la droite. Grâce aux tests précédents, l'intersection mutuelle de chacun des triangles avec la droite L est garantie. Ces intersections

forment des intervalles sur L et les triangles s'intersectent si et seulement si ces intervalles se chevauchent.

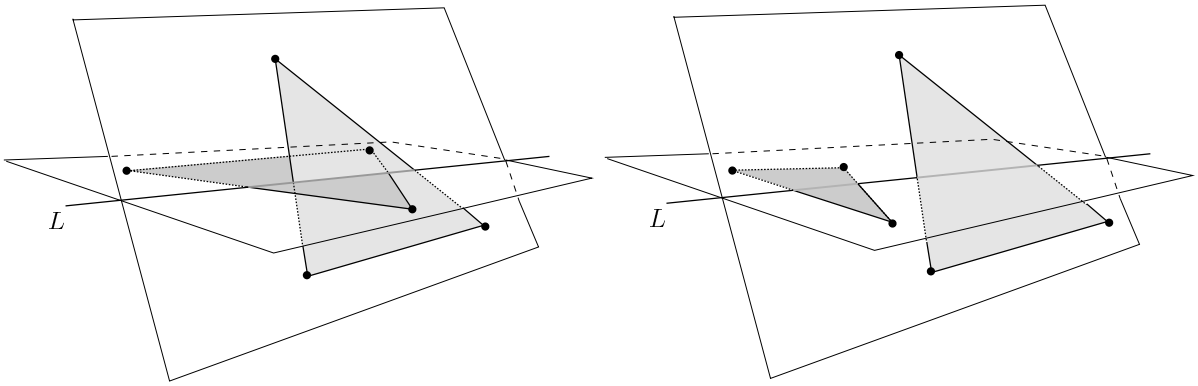

FIG. 2.4 – L'algorithme proposé par Möller calcule les bornes des intervalles formés par les triangles sur L et teste si ces intervalles s'intersectent.

L'algorithme calcule alors, pour chacun des triangles, l'intervalle scalaire (sur L) correspondant à l'intersection du triangle avec L.

La situation est illustrée par la figure 2.5. La valeur du paramètre t_1 correspondant au point d'intersection $\mathbf{i}$ entre la droite $L(t)$ et l'arête $\mathbf{a}_1\mathbf{a}_2$ est obtenu en remarquant que les triangles $\Delta\mathbf{a}_1\mathbf{i}\mathbf{a}_1{}'$ et $\Delta\mathbf{a}_2\mathbf{i}\mathbf{a}_2{}'$ (où $\mathbf{a}_1{}'$ et $\mathbf{a}_2{}'$ sont les projections de $\mathbf{a}_1$ et $\mathbf{a}_2$ sur π_B) sont similaires. Ainsi,

$$L(t_1) = \mathbf{a}_1{}'' + (\mathbf{a}_2{}'' - \mathbf{a}_1{}'')\frac{d_{\mathbf{a}_1}}{d_{\mathbf{a}_1} - d_{\mathbf{a}_2}} \tag{2.3}$$

où $\mathbf{a}_i{}''$ désigne la projection du sommet $\mathbf{a}_i$ sur la droite $L(t)$.

Les valeurs de chacune des bornes des deux intervalles s'obtiennent de manière similaire et le test d'intersection initial est réduit à un test d'intersection entre deux intervalles monodimensionnels.

Le cas spécial où les deux triangles sont coplanaires est traité en projetant ceux-ci sur le plan $z = 0$, $y = 0$, ou $x = 0$ qui maximise l'aire des triangles (*i.e.* en considérant les sommets bidimensionnels obtenus en supprimant la coordonnée correspondant à la composante maximale du vecteur normal aux triangles). Un test bidimensionnel est alors appliqué : chacune des arêtes de A est tout d'abord testée afin de déterminer si elle intersecte une arête de B. Si au moins une intersection est détectée alors les triangles s'intersectent, sinon un test d'inclusion est effectué en testant l'inclusion d'un sommet de A dans B et vice versa.

Les différentes étapes de l'algorithme sont résumées ci-dessous :

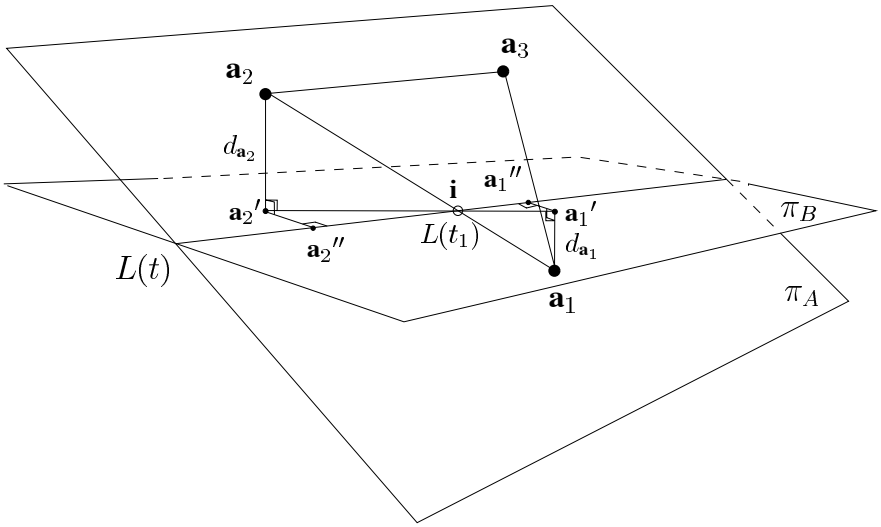

FIG. 2.5 – Les triangles $\Delta\mathbf{a}_1\mathbf{i}\mathbf{a}_1{}'$ et $\Delta\mathbf{a}_2\mathbf{i}\mathbf{a}_2{}'$ sont similaires.

1. Calcul de l'équation du plan supportant le triangle B.
2. Retourner *faux* si tous les sommets de A se situent du même côté du plan
3. Calcul de l'équation du plan supportant le triangle A.
4. Retourner *faux* si tous les sommets de B se situent du même côté du plan
5. Calcul de la droite d'intersection des deux plans,
6. Calcul des intervalles formés sur L par chaque triangle
7. Retourner *vrai* si les intervalles s'intersectent, *faux* sinon.

La méthode ERIT

La solution proposée par Held [41] se découpe de la manière suivante :

1. Calcul de l'équation du plan $\pi_B : \mathbf{n}_B \cdot \mathbf{x} + d_B$ supportant le triangle B.
2. Retourner *faux* si tous les sommets de A se situent du même côté du plan
3. Calcul du segment d'intersection $s = \pi_B \cap A$
4. Retourner *vrai* si s intersecte ou est contenu dans B, *faux* sinon.

Les deux premières étapes sont identiques à la méthode précédente. La troisième étape nécessite le calcul de deux points **u** et **v** de π_B représentant les extrémités du segment d'intersection s. En réutilisant les distances signées $d_{\mathbf{a}_1}$, $d_{\mathbf{a}_2}$, et $d_{\mathbf{a}_3}$ calculées à l'étape 2, le point **u** résultant

de l'intersection de, par exemple l'arête $\mathbf{a}_1\mathbf{a}_2$ avec le plan π_B (cf. figure 2.6), est calculé grâce à l'équation suivante

$$\mathbf{u} = \mathbf{a}_1 + \frac{d_{\mathbf{a}_1}}{d_{\mathbf{a}_1} - d_{\mathbf{a}_2}}(\mathbf{a}_2 - \mathbf{a}_1). \tag{2.4}$$

Le point **v** est calculé de manière similaire. L'étape 4 est accomplie en projetant B, **u**, et **v** sur le plan $x = 0$, $y = 0$, ou $z = 0$ maximisant l'aire de B. Le test bidimensionnel teste alors l'appartenance de chacune des extrémités du segment aux demi-plans induits par les arêtes du triangle. Si une des droites supportant les arêtes du triangle sépare le triangle du segment, il ne peut y avoir d'intersection, inversement, si une des extrémités appartient au triangle, il y a intersection. Dans les autres cas, chacune des arêtes du triangle est testée ; dès qu'une intersection avec le segment est détectée, il existe une intersection sinon les triangles sont disjoints.

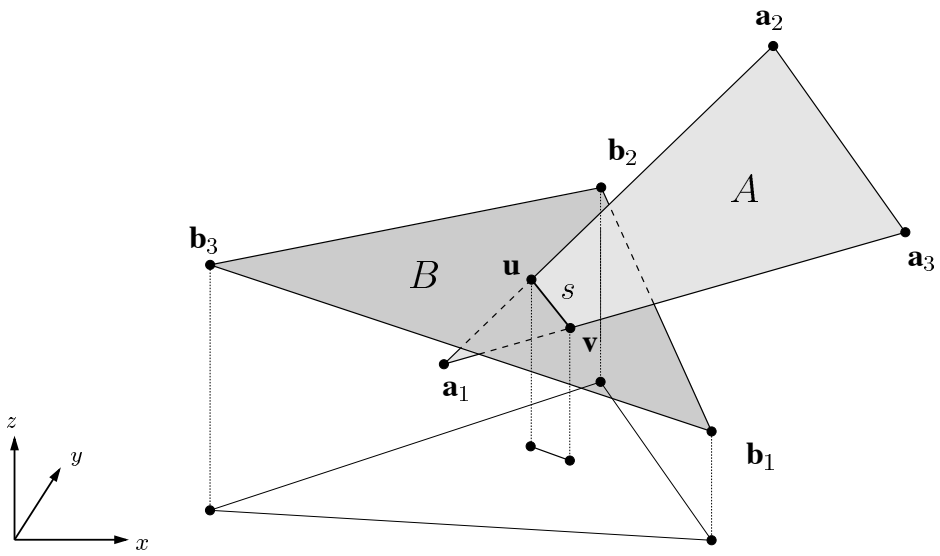

FIG. 2.6 – L'algorithme proposé par Held construit le segment $s = A \cap \pi_B$, et teste ensuite l'intersection de s et B après projection sur un plan adéquat.

Malgré leur efficacité, les deux algorithmes décrits ci-dessus ont un inconvénient lié à la présence de constructions intermédiaires dans le processus menant à la solution. En effet, le fait d'utiliser (ou comparer) certaines valeurs issues de constructions intermédiaires élève le degré du prédicat et par conséquent les exigences arithmétiques d'une implémentation exacte de celui-ci. D'autre part, ces constructions ne sont pas intrinsèques au problème qui ne nécessite que de localiser des points relativement à des plans.

La prochaine section présente une nouvelle formulation du test d'intersection de deux triangles tridimensionnels. Celle-ci est basée exclusivement sur le calcul du signe de tests d'orientation

tridimensionnel sur les entrées (*i.e.* ne nécessite aucune construction intermédiaire) et n'effectue qu'un petit nombre d'appels à ce test (le prédicat effectue 8 tests d'orientation tridimensionnels dans le pire des cas).

2.2.2 Description de l'algorithme

Le principe général de l'algorithme est le suivant : chacun des sommets est dans un premier temps classé par rapport au plan support de l'autre triangle, les sommets de chaque triangle sont ensuite permutés dans une forme canonique, l'algorithme conclut enfin en évaluant le signe de deux prédicats d'orientation.

L'algorithme débute de manière similaire à celui de Möller en testant l'intersection mutuelle de chacun des triangles avec le plan support de l'autre. Pour cela, l'algorithme vérifie dans un premier temps l'intersection de A avec le plan π_B en calculant le signe des déterminants $d_{\mathbf{a}_1} = [\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3, \mathbf{a}_1]$, $d_{\mathbf{a}_2} = [\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3, \mathbf{a}_2]$, et $d_{\mathbf{a}_3} = [\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3, \mathbf{a}_3]$. Trois situations sont possibles. Si aucun des signes n'est égal à zéro et si les trois signes sont identiques, les sommets de A se situent dans le même demi-espace induit par π_B et il ne peut y avoir d'intersection entre les deux triangles. Si les trois signes sont égaux à zéro, les deux triangles sont coplanaires et le problème est ramené à un test d'intersection bidimensionnel (cf. section 2.3.2) après avoir projeté les triangles sur le plan $z = 0$, $x = 0$ ou $y = 0$ qui maximise l'aire des triangles projetés. Enfin, si les trois déterminants ont des signes différents, les sommets de A se situent de part et d'autre du plan π_B et A intersecte certainement π_B. Dans ce dernier cas, l'algorithme vérifie de manière similaire l'intersection de B avec π_A.

L'algorithme applique ensuite, en fonction du signe $(-1, 0, +1)$ des déterminants calculés, une permutation sur les sommets de chacun des triangles de la manière suivante. Une première permutation circulaire est tout d'abord effectuée sur les sommets de A (resp. de B) afin de se ramener à une configuration où le signe de $d_{\mathbf{a}_1}$ (resp. $d_{\mathbf{b}_1}$) est soit strictement supérieur, soit strictement inférieur aux signes de $d_{\mathbf{a}_2}$ et $d_{\mathbf{a}_3}$ (respectivement aux signes de $d_{\mathbf{b}_2}$ et $d_{\mathbf{b}_3}$). Si nécessaire, une opération de transposition supplémentaire est alors effectuée sur les sommets $\mathbf{b}_2$ et $\mathbf{b}_3$ (respectivement $\mathbf{a}_2$ et $\mathbf{a}_3$) afin d'inverser les signes de ces déterminants et ainsi garantir que le signe de $d_{\mathbf{a}_1}$ (resp. $d_{\mathbf{b}_1}$) est bien supérieur aux signes de $d_{\mathbf{a}_2}$ et $d_{\mathbf{a}_3}$ (resp. $d_{\mathbf{b}_2}$ et $d_{\mathbf{b}_3}$).

Grâce aux permutations précédentes chacune des arêtes incidentes à $\mathbf{a}_1$ et à $\mathbf{b}_1$ intersecte la droite $L = \pi_A \cap \pi_B$ en un point unique. Soient donc respectivement **i**, **j**, **k**, et **l** les points d'intersection de L avec les arêtes $\mathbf{a}_1\mathbf{a}_3$, $\mathbf{a}_1\mathbf{a}_2$, $\mathbf{b}_1\mathbf{b}_2$, et $\mathbf{b}_1\mathbf{b}_3$. Ces points forment deux intervalles fermés

I_1 et I_2 sur L correspondant à l'intersection des deux triangles A et B avec L. Or, à ce stade, l'algorithme dispose d'assez d'informations pour ordonner de manière cohérente les bornes de chaque intervalle, précisément, on a $I_1 = [\mathbf{i}, \mathbf{j}]$ et $I_2 = [\mathbf{k}, \mathbf{l}]$ si l'on oriente L dans la direction de $\mathbf{n} = \mathbf{n}_A \times \mathbf{n}_B$ où $\mathbf{n}_A = (\mathbf{a}_2 - \mathbf{a}_1) \times (\mathbf{a}_3 - \mathbf{a}_1)$ et $\mathbf{n}_B = (\mathbf{b}_2 - \mathbf{b}_1) \times (\mathbf{b}_3 - \mathbf{b}_1)$ (cf. figure 2.7).

Ainsi, le test d'intersection des deux intervalles I_1 et I_2 revient à vérifier que la borne inférieure de chaque intervalle est bien inférieure à la borne supérieure de l'autre, en l'occurrence, $\mathbf{k} \leq \mathbf{j}$ et $\mathbf{i} \leq \mathbf{l}$. Étant donné que chacune des arêtes incidentes à $\mathbf{a}_1$ ou $\mathbf{b}_1$ intersecte sûrement L, cette dernière condition est vérifiée si et seulement si

$$[\mathbf{a}_1, \mathbf{a}_2, \mathbf{b}_1, \mathbf{b}_2] \leq 0 \ \wedge \ [\mathbf{a}_1, \mathbf{a}_3, \mathbf{b}_3, \mathbf{b}_1] \leq 0.$$

Nous renvoyons le lecteur à la figure 2.7 et à la seconde interprétation du prédicat d'orientation tridimensionnel donné à la section 2.1.2 afin de s'en convaincre.

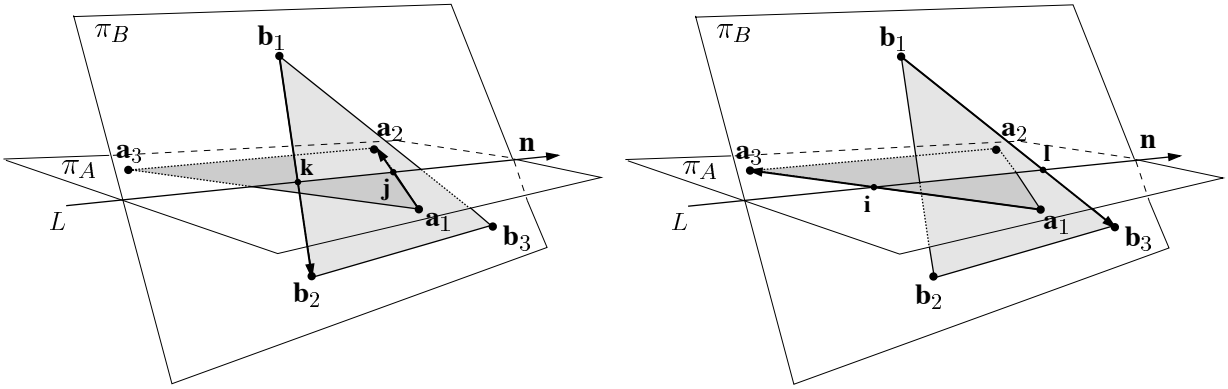

FIG. 2.7 – Une fois les sommets de chaque triangle permutés dans une forme canonique, les triangles s'intersectent si et seulement si $[\mathbf{a}_1, \mathbf{a}_2, \mathbf{b}_1, \mathbf{b}_2] \leq 0$ et $[\mathbf{a}_1, \mathbf{a}_3, \mathbf{b}_3, \mathbf{b}_1] \leq 0$.

Par conséquent, dans cette nouvelle formulation, la construction explicite d'objets intermédiaires nécessaires dans l'algorithme de Möller est évitée, le calcul et le tri des intervalles scalaires (étapes 6 et 7 page 45) étant remplacés par l'évaluation du signe d'au plus deux prédicats d'orientation sur les sommets originaux.

Les différentes étapes de l'algorithme sont résumées ci-dessous :

1. Tester l'intersection de A avec le plan π_B.
2. Retourner *faux* si $A \cap \pi_B = \emptyset$

3. Tester l'intersection de B avec le plan π_A.
4. Retourner *faux* si $B \cap \pi_A = \emptyset$
5. Permutation de A et B sous une forme canonique.
6. Retourner la valeur de la conjonction $[\mathbf{a}_1, \mathbf{a}_2, \mathbf{b}_1, \mathbf{b}_2] \leq 0 \ \wedge \ [\mathbf{a}_1, \mathbf{a}_3, \mathbf{b}_3, \mathbf{b}_1] \leq 0$.

Description de l'intersection Une description du segment d'intersection peut être obtenue soit en remplaçant les deux derniers prédicats d'orientation par trois nouveaux, soit en évaluant deux prédicats d'orientation supplémentaires à la fin de l'algorithme (cf. figure 2.8). Ces tests déterminent l'ordre des bornes **i**, **j**, **k**, et **l** sur L et permettent par conséquent de déduire les deux bornes de l'intervalle $I_1 \cap I_2$.

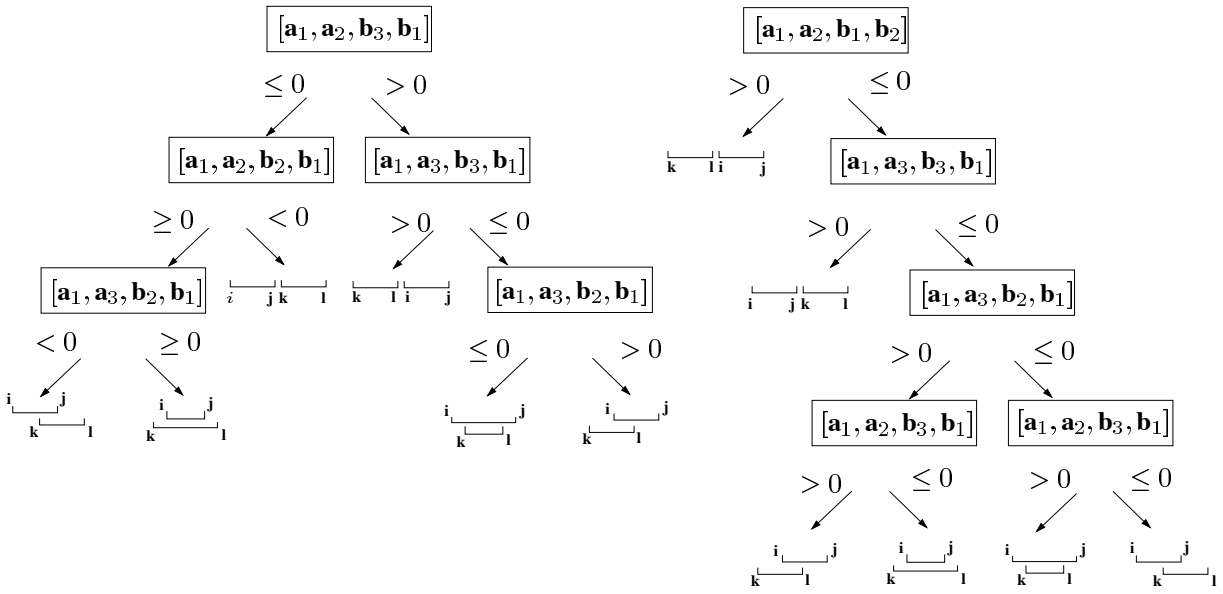

FIG. 2.8 – Description symbolique de l'intersection des deux triangles.

À partir de cette description symbolique de l'intersection, chaque point extrémité du segment d'intersection peut être construit, chaque borne de l'intervalle d'intersection correspondant à une intersection entre une arête d'un des triangles avec le plan de l'autre. Pour construire ces points d'intersection, l'algorithme doit calculer la valeur d'un polynôme rationnel ayant un numérateur de degré quatre et un dénominateur de degré trois. Plus précisément, le point d'intersection $\mathbf{p}_i$ entre une arête **pq** et le plan défini par les sommets **a**, **b**, et **c** est obtenu à l'aide de l'expression suivante :

$$\mathbf{p}_i = \frac{(\mathbf{p}-\mathbf{a}).\mathbf{n}}{(\mathbf{p}-\mathbf{q}).\mathbf{n}}(\mathbf{q}-\mathbf{p}) \qquad \text{où} \qquad \mathbf{n} = (\mathbf{b}-\mathbf{a}) \times (\mathbf{c}-\mathbf{a}).$$

2.2.3 Analyse comparative

Dans cette section, nous donnons le degré, le nombre d'opérations arithmétiques ainsi que les temps d'exécution des différents algorithmes exposés. Notre algorithme a été implémenté en C [39] et a été intégré dans la bibliothèque géométrique CGAL [8]. Le code (appelé dans la suite tri_tri_overlap_test) ne traite pas les triangles dégénérés, c'est-à-dire les triangles dont les trois sommets sont colinéaires.

Degrés arithmétiques

Dans notre algorithme, toutes les décisions de branchement correspondent à l'évaluation du signe de prédicats d'orientation tridimensionnels. Or, chacun de ces prédicats d'orientation s'exprime comme un polynôme de degré trois sur les données en entrée c'est-à-dire sur les coordonnées des sommets définissant les triangles (cf. Équation 2.2), par conséquent, l'algorithme est de degré trois.

L'algorithme proposé par Möller, quant à lui, utilise une paramétrisation de la droite d'intersection des deux plans supports des triangles $L(t) = \mathbf{o} + t\mathbf{d}$ où $\mathbf{d}$ est le produit vectoriel des deux normales aux triangles et $\mathbf{o}$ un point appartenant à la droite. Sa méthode calcule ensuite pour chaque triangle les valeurs de t représentant les deux bornes de l'intervalle scalaire correspondant à l'intersection du triangle et de L. L'algorithme compare finalement les valeurs obtenues.

Deux implémentations différentes sont disponibles à partir du site web [56]. La première version (tri_tri_intersect) utilise une projection simplifiée des sommets sur L et compare des polynômes rationnels ayant un numérateur de degré quatre et un dénominateur de degré trois. Cependant, cette version utilise des divisions et correspond, par conséquent, à l'évaluation d'un polynôme réel de degré sept. La deuxième version (NoDivTriTriIsect) remplace ces divisions par quelques multiplications supplémentaires. Ce dernier code est plus rapide mais doit quant à lui évaluer des polynômes de degré treize.

La méthode ERIT calcule le segment d'intersection du triangle B avec le plan π_A dont chaque extrémité s'exprime sous la forme d'un polynôme rationnel ayant un numérateur de degré quatre et un dénominateur de degré trois. L'algorithme effectue ensuite un test d'intersection triangle/ segment bidimensionnel qui nécessite l'évaluation de polynômes de degré deux sur les données en entrée. L'algorithme ERIT a par conséquent un degré égal à huit.

Ainsi, les méthodes basées exclusivement sur l'évaluation du signe de prédicats d'orientation permettent de limiter les exigences arithmétiques d'une implémentation robuste. Si les opérations arithmétiques sont implantées à l'aide de l'arithmétique flottante IEEE754, moins de bits

sont erronés à cause de l'arrondi et l'on peut prouver que les décisions de branchement sont prises correctement avec une forte probabilité si les sommets sont distribués uniformément et indépendamment dans un cube [13]. Si les opérations sont implantées exactement à l'aide d'une arithmétique multiprécision par exemple, ces méthodes nécessitent moins de calcul ou permettent de traiter des données appartenant à un plus grand domaine.

Nombre d'opérations arithmétiques

La Table 2.1 indique le nombre d'opérations arithmétiques pour chacune des implémentations. Pour cela, chaque code a été découpé en parties logiques selon la structure de ses étapes de rejet. Pour chacun des codes, on distingue cinq types d'opérations : addition/soustraction, multiplication, comparaison, affectation, division et valeur absolue. Implantées avec une arithmétique approchée, la quasi-totalité de ces opérations (excepté la division) nécessitent un nombre identique de cycles afin d'être exécutées. L'opération de division peut s'avérer être de quatre à huit fois plus coûteuse que les autres opérations.

Code	ADD	MUL	CMP	ASS	DIV	ABS
tri_tri_overlap_test()						
A intersecte π_B ?	24	17	2	21	0	0
B intersecte π_A ?	24	17	2	21	0	0
$\min(I_2) > \max(I_1)$?	14	9	7/13	12/32	0	0
$\min(I_1) > \max(I_2)$?	14	9	1	12	0	0
Möller - NoDivTriTriIsect()						
A intersecte π_B ?	21	20	2	15	0	0
B intersecte π_A ?	21	20	2	15	0	0
I_1 intersecte I_2 ?	15	23	7/18	32/42	0	3
Möller - tri_tri_intersect()						
A intersecte π_B ?	21	20	2	15	0	0
B intersecte π_A ?	21	20	2	15	0	0
I_1 intersecte I_2 ?	15	10	8/24	25	4	3
ERIT - TriTri3D()						
B intersecte π_A ?	24	15	6/11	24	0	0
s intersecte B ?	20/42	8/16	14/26	26/29	1/2	0

TAB. 2.1 – Nombre minimum et maximum d'opérations arithmétiques. Cas général.

Performances

La Table 2.2 présente une évaluation des performances des différentes méthodes en termes de temps de calcul et de robustesse pour différents types de données en position générale et en position dégénérée. Les paires de triangles *aléatoires* ont été obtenues en tirant pseudo-aléatoirement la valeur des coordonnées de chaque sommet à l'aide de la fonction `drand48()` issue de la bibliothèque standard C. Les données *dégénérées* correspondent à des paires de triangles obtenues en générant deux arêtes coplanaires qui s'intersectent ainsi que deux points de part et d'autre du plan et en arrondissant les coordonnées exactes au nombre flottant double précision le plus proche.

	(1)		(2)		(3)		(4)	
Code	μ sec	%	μ sec	%	μ sec	%	μ sec	%
tri_tri_overlap_test()	0.34	0	0.18	0	0.33	10	0.32	41
Möller - NoDivTriTriIsect()	0.46	0	0.27	0	0.46	33	0.46	53
Möller - tri_tri_intersect()	0.53	0	0.28	0	0.53	31	0.54	54
Held - TriTri3D()	0.48	0	0.30	0	0.48	43	0.47	29
tri_tri_intersection_test()	0.55	0	0.19	0	0.53	9	0.43	40
Möller - tri_tri_intersect_with_isectline()	0.62	0	0.30	0	0.61	31	0.60	54

TAB. 2.2 – Temps de calcul (en micro secondes) et pourcentage de mauvaises réponses des algorithmes implantés avec une arithmétique IEEE double précision. Les différents types de données sont des paires de triangles : (1) aléatoires avec intersection, (2) aléatoires sans intersection, (3) presque dégénérées avec intersection, et (4) presque dégénérées sans intersection.

En plus des performances des différents prédicats d'intersection, le tableau présente les résultats obtenus pour le code tri_tri_intersect_with_isectline() proposé par Möller ainsi que pour notre code tri_tri_intersection_test() qui sont des versions construisant le segment d'intersection.

Les temps d'exécution ont été obtenus en effectuant une moyenne du temps nécessaire au calcul de plusieurs millions de tests d'intersection. Les différentes implémentations ont été obtenues à l'aide du compilateur GNU 2.95 et l'option de compilation -O2. Les temps de calcul ont été mesurés sur un pentium III 1GHz à l'aide de la fonction de la bibliothèque standard `clock`. Dans sa version IEEE double précision, notre test d'intersection s'avère être à la fois plus fiable et entre 10 et 20 % plus rapide que la plus efficace des autres méthodes pour les différentes données testées.

Si l'exactitude du résultat est une priorité, notre algorithme peut bénéficier de techniques d'ac-

célération [69, 6, 2] afin de minimiser le coût de l'évaluation exacte des différents prédicats d'orientation intervenant dans le test d'intersection.

2.3 Test d'intersection Triangle/Triangle 2D

Cette section présente un algorithme pour tester efficacement l'intersection de deux triangles bidimensionnels. Ce test est utilisé entre autre comme sous routine du test tridimensionnel de la section 2.2.2 afin de traiter le cas spécial où les deux triangles en entrée sont coplanaires.

2.3.1 Méthodes existantes

On distingue deux méthodes communément utilisées, toutes deux sont basées sur le test d'orientation bidimensionnel.

La première est une application directe de la propriété suivante : les triangles $A = \Delta\mathbf{a}_1\mathbf{a}_2\mathbf{a}_3$ et $B = \Delta\mathbf{b}_1\mathbf{b}_2\mathbf{b}_3$ sont disjoints si et seulement si au moins l'une des droites supports aux arêtes de A et de B sépare les deux triangles. Si l'on suppose A et B orientés dans le sens direct (c'est-à-dire dans le sens inverse des aiguilles d'une montre), la droite support à l'arête $\mathbf{a}_1\mathbf{a}_2$ sépare A et B si et seulement si $[\mathbf{a}_1, \mathbf{a}_2, \mathbf{b}_1] < 0$, $[\mathbf{a}_1, \mathbf{a}_2, \mathbf{b}_2] < 0$, et $[\mathbf{a}_1, \mathbf{a}_2, \mathbf{b}_3] < 0$. L'algorithme teste ainsi successivement la droite support à chacune des 6 arêtes, si au moins l'une d'elles sépare A et B, les triangles sont disjoints, sinon les triangles s'intersectent.

Une deuxième solution, assez similaire, consiste à tester l'intersection de chacune des arêtes de A avec les arêtes de B. Si au moins une intersection est détectée, les deux triangles s'intersectent. Sinon, un test est effectué afin de déterminer si A est inclus dans B ou vice versa. Cette dernière étape étant accomplie en testant l'inclusion d'un sommet de A dans B et vice versa.

Chacune de ces méthodes nécessite dans le pire des cas (*i.e.* lorsque les deux triangles sont disjoints) l'évaluation d'au moins 18 tests d'orientation bidimensionnels (la première solution nécessite deux tests supplémentaires pour déterminer l'orientation des triangles en entrée). Nous proposons dans la prochaine section une nouvelle formulation de ce test d'intersection ne nécessitant que 10 tests d'orientation dans le pire des cas.

2.3.2 Description de l'algorithme

L'algorithme se déroule de la manière suivante. La première étape détermine l'orientation de chacun des deux triangles et procède si nécessaire à une opération de transposition de deux de leurs sommets afin d'orienter les triangles $\mathbf{a}_1\mathbf{a}_2\mathbf{a}_3$ et $\mathbf{b}_1\mathbf{b}_2\mathbf{b}_3$ dans le sens direct. La seconde étape

localise le sommet $\mathbf{a}_1$ dans l'arrangement des droites supports aux arêtes de B et applique si nécessaire une permutation circulaire sur les sommets de B afin de ramener le sommet $\mathbf{a}_1$ dans une région de l'arrangement correspondant à un des deux cas génériques. Pour chacun de ces cas, la dernière étape fournit finalement la solution en invoquant au plus cinq tests d'orientation successifs. Dans le pire des cas, l'algorithme dans sa totalité effectue deux opérations de transposition, une permutation circulaire et dix tests d'orientation bidimensionnels (huit si l'orientation initiale des triangles est connue). Nous détaillons dans la suite les deux principales étapes de l'algorithme à savoir le principe de localisation du sommet $\mathbf{a}_1$ ainsi que les arbres de décision correspondant aux deux cas génériques.

Étant donnés le sommet $\mathbf{a}_1$ et le triangle B orienté dans le sens direct, la localisation de $\mathbf{a}_1$ dans l'arrangement des droites supports aux arêtes de B s'effectue en évaluant successivement le signe des tests d'orientation $[\mathbf{b}_1, \mathbf{b}_2, \mathbf{a}_1]$, $[\mathbf{b}_2, \mathbf{b}_3, \mathbf{a}_1]$, et $[\mathbf{b}_3, \mathbf{b}_1, \mathbf{a}_1]$ (voir figure 2.9).

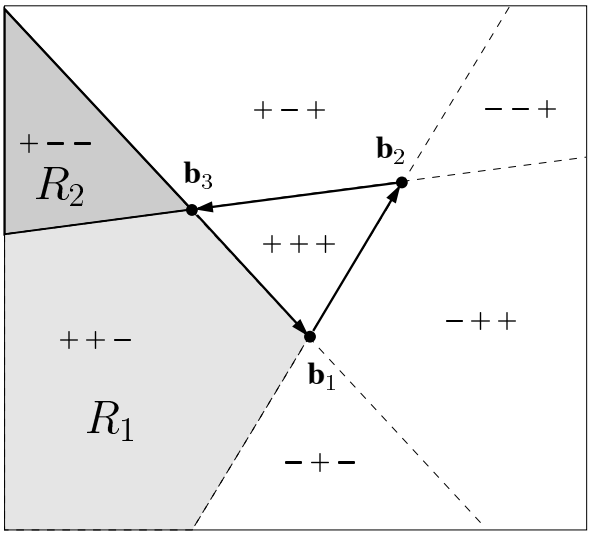

FIG. 2.9 – Sous l'hypothèse que B est orienté dans le sens direct, les différentes combinaisons de signe indiquant la position relative de $\mathbf{a}_1$ sont représentées. Les droites délimitant chaque région correspondent à une valeur nulle.

Les cas où, un seul signe est égal à zéro et les deux autres sont positifs, deux des signes sont égaux à zéro, ou les trois signes sont positifs, correspondent respectivement aux cas où $\mathbf{a}_1$ est sur une arête, sur un sommet, ou est à l'intérieur de B et révèlent une intersection entre les deux triangles A et B (le cas où les trois signes sont égaux à zéro est impossible étant donné que les triangles de départ sont supposés être non dégénérés). Dans tous les autres cas, une permutation circulaire peut être appliquée sur les sommets de B de manière à ce que $\mathbf{a}_1$ appartienne soit à

l'intérieur de la région R_1, soit à l'intérieur ou sur la frontière de la région R_2 définies par la figure 2.9.

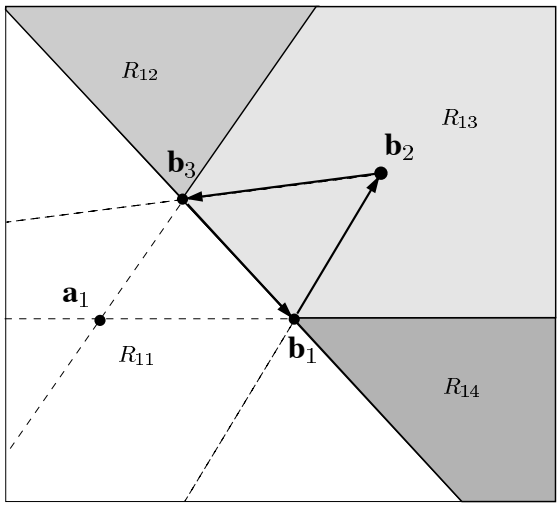

FIG. 2.10 – $\mathbf{a}_1$ appartient à la région R_1, le plan est décomposé en quatre régions.

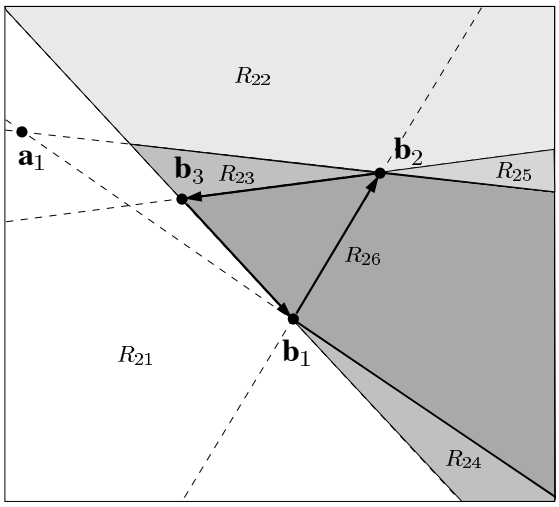

FIG. 2.11 – $\mathbf{a}_1$ appartient à la région R_2, le plan est décomposé en six régions.

Supposons tout d'abord qu'après permutation des sommets du triangle B, $\mathbf{a}_1$ se retrouve à l'intérieur de la région R_1. L'algorithme procède alors à la localisation de $\mathbf{a}_2$ en déterminant, d'une manière similaire à la localisation de $\mathbf{a}_1$ précédente, à laquelle des régions représentées sur la figure 2.10 le sommet $\mathbf{a}_2$ appartient. Les différents tests effectués sont détaillés par la figure 2.12.

On distingue les quatre situations suivantes :

- Le sommet $\mathbf{a}_2$ appartient à la région R_{11} (Test I). Dans ce cas, l'intersection entre les triangles dépend de la position du sommet $\mathbf{a}_3$. Si celui-ci appartient aussi à la région R_{11} (Test II.b) la droite supportant l'arête $\mathbf{b}_3\mathbf{b}_1$ sépare les deux triangles et il ne peut y avoir d'intersection, sinon, l'algorithme conclut en évaluant au plus deux nouveaux tests d'orientation (Test III.b et IV.b), les triangles étant disjoints si et seulement si l'une des droites supports aux arêtes $\mathbf{a}_2\mathbf{a}_3$ et $\mathbf{a}_3\mathbf{a}_1$ sépare les deux triangles.
- Le sommet $\mathbf{a}_2$ appartient à la région R_{12} (Test II.a). Les triangles sont disjoints, B étant orienté dans le sens direct.
- Le sommet $\mathbf{a}_2$ appartient à la région R_{13} (Test III.a). Les deux triangles s'intersectent, une intersection entre les arêtes $\mathbf{a}_1\mathbf{a}_2$ et $\mathbf{b}_3\mathbf{b}_1$ ayant été détectée.
- Enfin, si le sommet appartient à la région R_{14} (Test III.a), l'algorithme conclut en évaluant

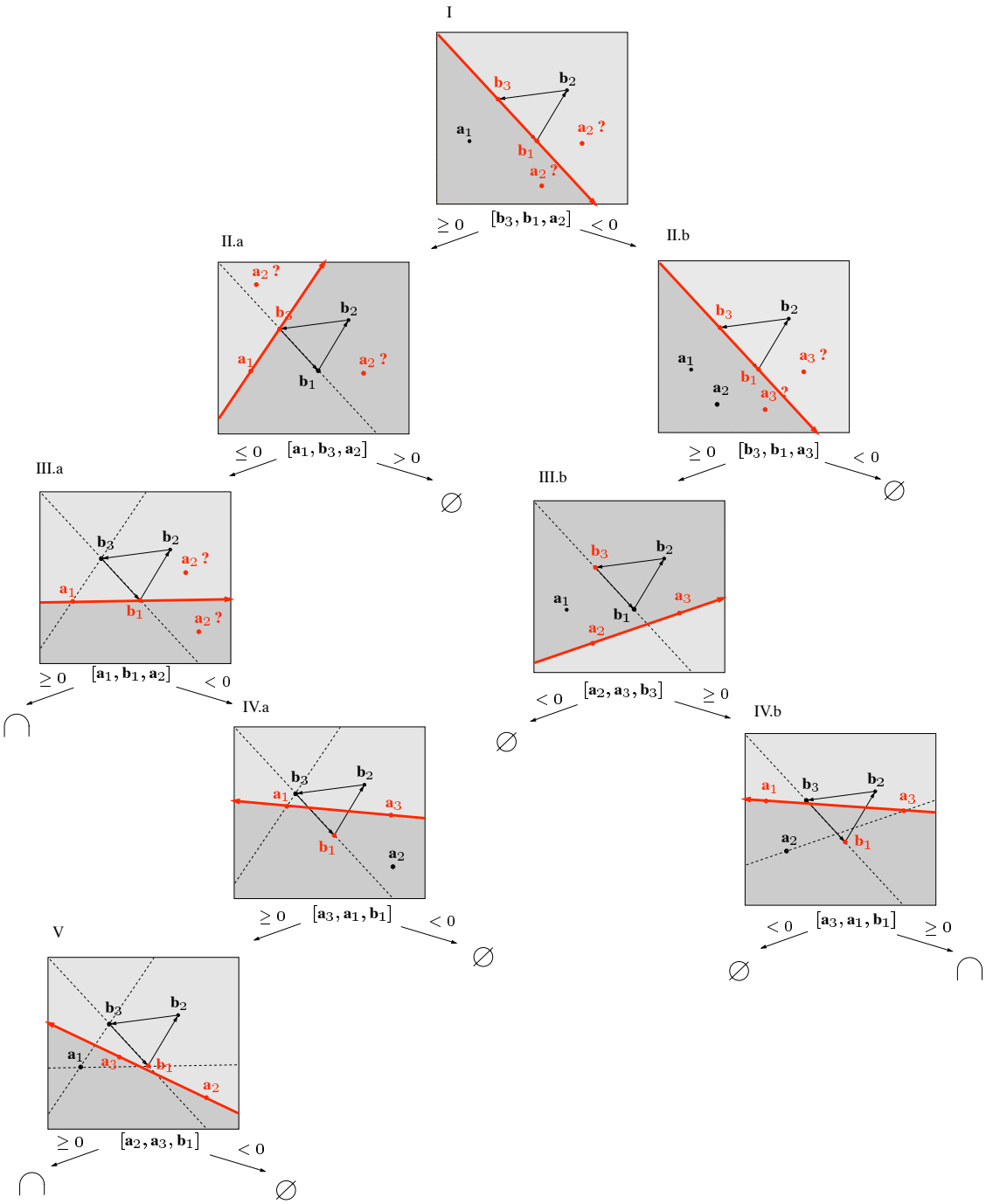

FIG. 2.12 – Arbre de décision correspondant au cas où $\mathbf{a}_1$ appartient à la région $\mathbf{R}_1$.

au plus deux nouveaux tests d'orientation, les triangles ne pouvant s'intersecter que lorsque $\mathbf{b}_1$ est inclus dans le triangle A (Test IV.a et V).

Dans le cas où, après permutation des sommets de B, $\mathbf{a}_1$ appartient à la région R_2, l'algorithme

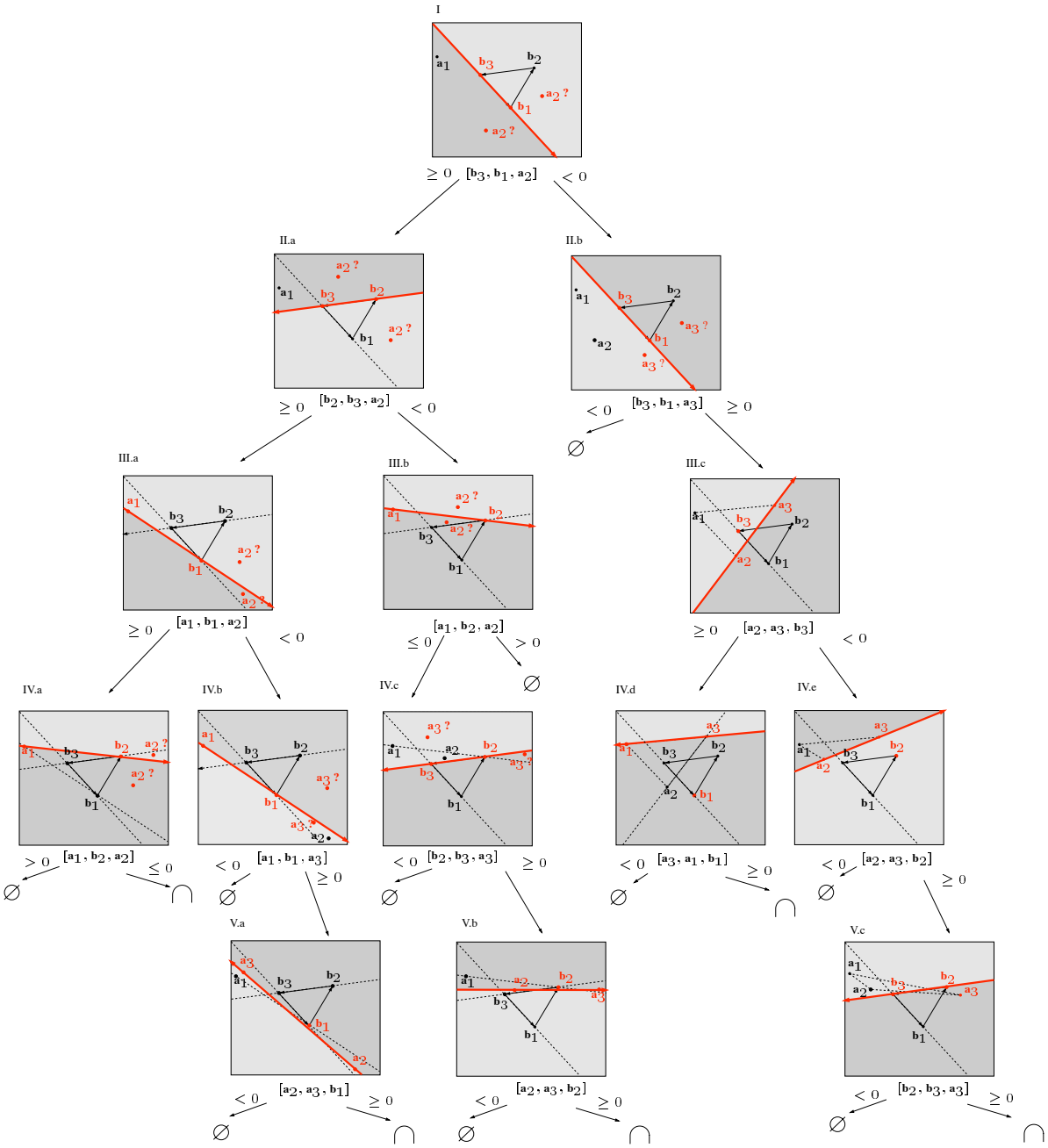

FIG. 2.13 – Arbre de décision correspondant au cas où $\mathbf{a}_1$ appartient à la région R_2.

procède à l'aide de l'arbre de décision de la figure 2.13. On distingue cette fois six situations distinctes correspondant aux six régions représentées à la figure 2.11.

– Le sommet $\mathbf{a}_2$ appartient à la région R_{21} (Test I). Dans ce cas, l'intersection entre les triangles dépend de la position du sommet $\mathbf{a}_3$. Si celui-ci appartient aussi à R_{21} (Test II.b), la droite support à l'arête $\mathbf{b}_3\mathbf{b}_1$ sépare A de B et les triangles sont disjoints, sinon, l'al-

gorithme teste successivement si les triangles peuvent être séparés par au moins l'une des droites supports aux arêtes $\mathbf{a}_2\mathbf{a}_3$ (Tests III.c et IV.e), $\mathbf{a}_3\mathbf{a}_1$ (Test IV.d) et $\mathbf{b}_2\mathbf{b}_3$ (Tests V.c). Si au moins l'une de ces conditions est vérifiée, les triangles sont disjoints, sinon les triangles s'intersectent.

- Le sommet $\mathbf{a}_2$ appartient à la région R_{22} (Tests II.a et III.b). Les deux triangles sont disjoints, A étant orienté dans le sens direct.
- Le sommet $\mathbf{a}_2$ appartient à la région R_{23} (Test III.b). L'algorithme teste alors successivement si les triangles sont séparables par l'une des droites supports aux arêtes $\mathbf{b}_2\mathbf{b}_3$ (Test IV.c), $\mathbf{a}_2\mathbf{a}_3$ (Test V.b). Si au moins l'une de ces conditions est vérifiée, il ne peut y avoir d'intersection, sinon les triangles s'intersectent.
- Le sommet $\mathbf{a}_2$ appartient à la région R_{24} (Test III.a). L'algorithme teste l'inclusion du sommet $\mathbf{b}_1$ dans le triangle A (Tests IV.b et V.a) qui est une condition nécessaire à l'intersection des triangles.
- Le sommet $\mathbf{a}_2$ appartient à la région R_{25} (Test IV.a). Les triangles sont disjoints, A étant orienté dans le sens direct.
- Le sommet $\mathbf{a}_2$ appartient à la région R_{26} (Test IV.a). Les deux triangles s'intersectent puisque, dans ce cas, l'arête $\mathbf{a}_1\mathbf{a}_2$ intersecte le triangle B.

2.4 Test d'intersection de deux polygones convexes 3D

Nous proposons dans cette section une généralisation du test d'intersection de deux triangles tridimensionnels de la section 2.2.2 au cas de deux polygones convexes. Soit donc A et B deux polygones convexes tridimensionnels, c'est-à-dire, deux polygones convexes planaires plongés dans l'espace tridimensionnel et π_A et π_B leur plan support respectif (cf. figure 2.14).

Chaque polygone en entrée est inclus dans son plan support, ainsi A et B ne peuvent s'intersecter que le long de L la droite d'intersection de leur plan. Or, l'intersection de chacun des polygones avec le plan de l'autre forme un intervalle sur L, il suffit donc pour déterminer si les polygones en entrée s'intersectent de tester si ces intervalles se chevauchent.

Par conséquent, l'algorithme détermine dans un premier temps deux arêtes de A (respectivement deux arêtes de B) perçant le plan de B (respectivement le plan de A) dans des directions opposées. Étant donnée une arête $\mathbf{a}_i\mathbf{a}_j$ de A, on dit que $\mathbf{a}_i\mathbf{a}_j$ perce π_B du côté positif vers le côté négatif si $[\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2, \mathbf{a}_i]$ et $[\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2, \mathbf{a}_j]$ n'ont pas tous les deux le même signe et $[\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2, \mathbf{a}_i] \geq 0$ et $[\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2, \mathbf{a}_j] \leq 0$. Similairement, $\mathbf{a}_i\mathbf{a}_j$ perce π_B du côté négatif vers le côté positif si $[\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2, \mathbf{a}_i]$ et $[\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2, \mathbf{a}_j]$ n'ont pas tous les deux le même signe et

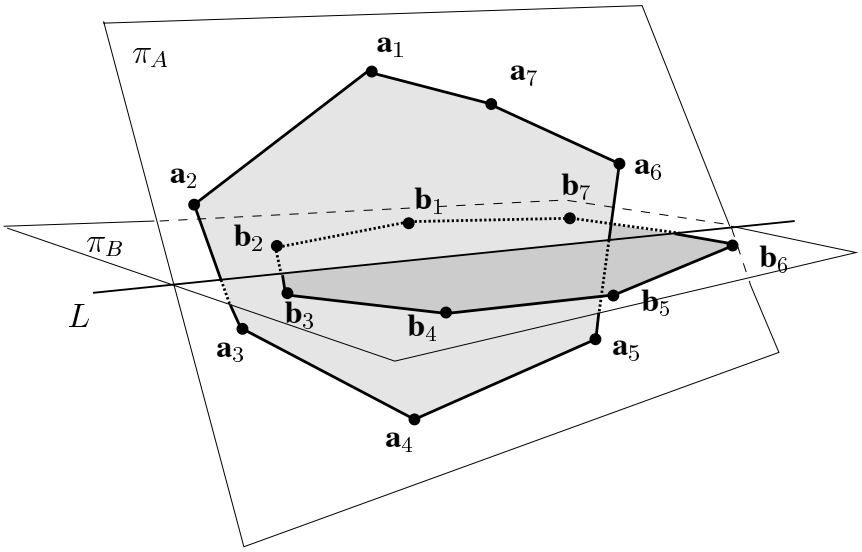

FIG. 2.14 – Les polygones et leur plan support.

$[\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2, \mathbf{a}_i] \leq 0$ et $[\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2, \mathbf{a}_j] \geq 0$.

Si de telles arêtes n'existent pas pour au moins un des polygones, soit l'intersection mutuelle de chacun des polygones avec le plan support de l'autre est vide, soit les polygones sont coplanaires. Le premier cas est détecté dès qu'un sommet de l'un des polygones n'appartient pas au plan support de l'autre. Le second cas, quant à lui, se produit lorsque au moins trois des sommets d'un polygone sont coplanaires avec le plan de l'autre et est traité en effectuant un test bidimensionnel (voir section 2.5) après avoir projeté les sommets de chacun des polygones sur un plan adéquat.

Dans tous les autres cas, soit $\overrightarrow{\mathbf{a}_i\mathbf{a}_j}$ et $\overrightarrow{\mathbf{a}_k\mathbf{a}_l}$ deux arêtes orientées de A perçant respectivement π_B du côté positif vers le côté négatif et du côté négatif vers le côté positif et $\overrightarrow{\mathbf{b}_i\mathbf{b}_j}$ et $\overrightarrow{\mathbf{b}_k\mathbf{b}_l}$ deux arêtes orientées de B perçant respectivement π_A du côté positif vers le côté négatif et du côté négatif vers le côté positif (cf figure 2.15). La réponse au test d'intersection initial est alors obtenue en évaluant la conjonction suivante

$$[\mathbf{a}_i, \mathbf{a}_j, \mathbf{b}_i, \mathbf{b}_j] \leq 0 \ \wedge \ [\mathbf{a}_k, \mathbf{a}_l, \mathbf{b}_l, \mathbf{b}_k] \leq 0. \tag{2.5}$$

Afin de déterminer les arêtes $\mathbf{a}_i\mathbf{a}_j$, $\mathbf{a}_k\mathbf{a}_l$, $\mathbf{b}_i\mathbf{b}_j$ et $\mathbf{b}_k\mathbf{b}_l$ nécessaires à l'évaluation de 2.5, l'algorithme, dans sa version la plus simple, calcule successivement la position des sommets de chaque polygone en entrée relativement au plan support de l'autre. La complexité totale de l'algorithme en termes de nombre de tests d'orientation est donc égale à $m + n + 2$ où m et n désignent respectivement le nombre de sommets de A et de B.

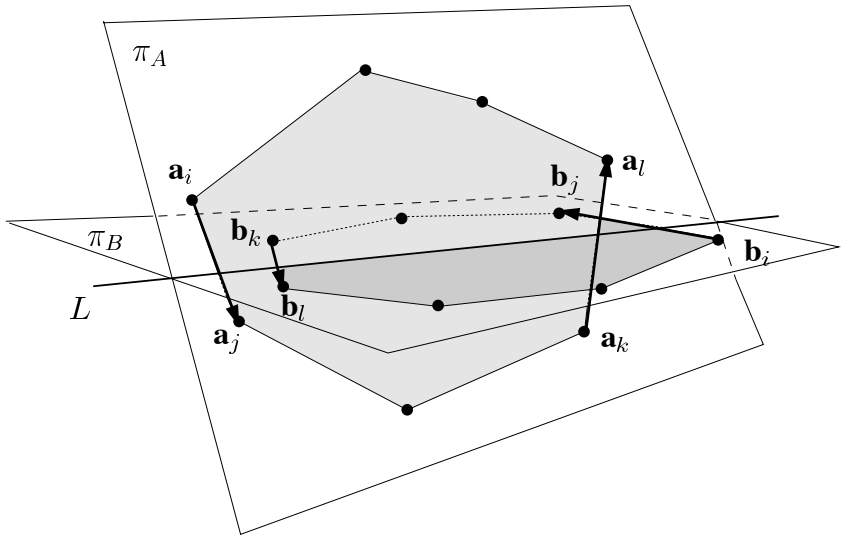

FIG. 2.15 – Une fois les arêtes $\mathbf{a}_i\mathbf{a}_j$, $\mathbf{a}_k\mathbf{a}_l$, $\mathbf{b}_i\mathbf{b}_j$ et $\mathbf{b}_k\mathbf{b}_l$ déterminées, le test d'intersection retourne la valeur du test $[\mathbf{a}_i, \mathbf{a}_j, \mathbf{b}_i, \mathbf{b}_j] \leq 0 \ \wedge \ [\mathbf{a}_k, \mathbf{a}_l, \mathbf{b}_l, \mathbf{b}_k] \leq 0$.

Lorsque $(m+n)$ est grand, on peut toutefois déterminer les arêtes $\mathbf{a}_i\mathbf{a}_j$, $\mathbf{a}_k\mathbf{a}_l$, $\mathbf{b}_i\mathbf{b}_j$ et $\mathbf{b}_k\mathbf{b}_l$ en temps $O(\log(mn))$ grâce à une recherche dichotomique sur les sommets de A et de B. Étant donnée $\mathbf{a}_1, \ldots, \mathbf{a}_m$ la liste ordonnée des sommets de A, le principe de la recherche des arêtes $\mathbf{a}_i\mathbf{a}_j$ et $\mathbf{a}_k\mathbf{a}_l$ est le suivant.

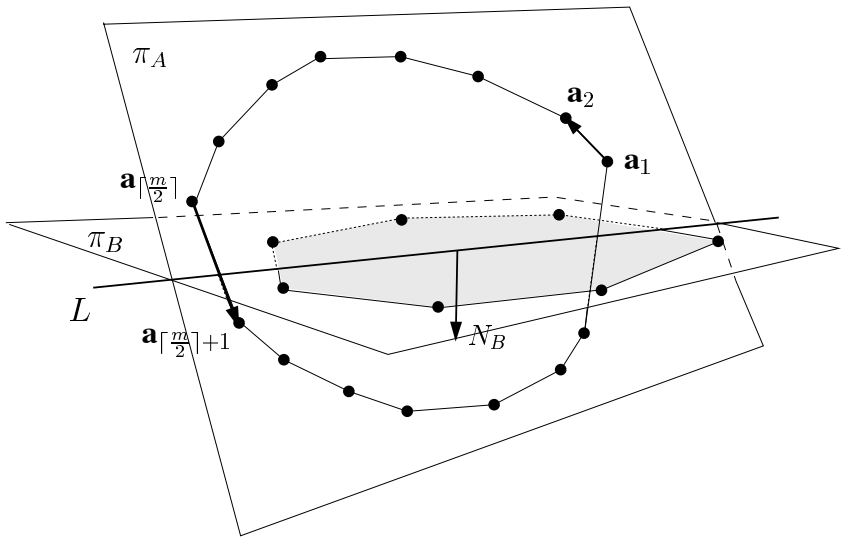

FIG. 2.16 – Principe de la recherche dichotomique. Cas où **v** < 0 et **v'** > 0, la sous-chaîne $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_{\lceil \frac{m}{2} \rceil}$ peut être supprimée.

Si $[\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2, \mathbf{a}_1].[\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2, \mathbf{a}_{\lceil \frac{m}{2} \rceil}] < 0$, les deux sous-chaines $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_{\lceil \frac{m}{2} \rceil}$ et $\mathbf{a}_{\lceil \frac{m}{2} \rceil}, \mathbf{a}_{\lceil \frac{m}{2} \rceil + 1}, \ldots, \mathbf{a}_m$ intersectent le plan π_B, et les arêtes $\mathbf{a}_i\mathbf{a}_j$ et $\mathbf{a}_k\mathbf{a}_l$ peuvent être déterminées selon une recherche dichotomique classique (suivant le côté de π_B auquel appartient le médian de la sous-chaîne), en évaluant le signe de $2 \times \log(\lceil \frac{m}{2} \rceil)$ tests d'orientation.

Dans le cas contraire, c'est-à-dire si $\mathbf{a}_1$ et $\mathbf{a}_{\lceil \frac{m}{2} \rceil}$ se situent du même côté de π_B, la dichotomie se fait à partir du signe de $\mathbf{v} = (\overrightarrow{N_B} \cdot \overrightarrow{\mathbf{a}_1\mathbf{a}_2})$ et de $\mathbf{v'} = (\overrightarrow{N_B} \cdot \overrightarrow{\mathbf{a}_{\lceil \frac{m}{2} \rceil}\mathbf{a}_{\lceil \frac{m}{2} \rceil + 1}})$, où $\overrightarrow{N_B}$ désigne la normale à π_B orientée du côté opposé à $\mathbf{a}_1$ et $\mathbf{a}_{\lceil \frac{m}{2} \rceil}$ (cf. figure 2.16), de la manière suivante.

Si $\text{signe}(\mathbf{v}) > \text{signe}(\mathbf{v'})$ la chaîne formée par les sommets $\mathbf{a}_{\lceil \frac{m}{2} \rceil}, \mathbf{a}_{\lceil \frac{m}{2} \rceil + 1}, \ldots, \mathbf{a}_m, \mathbf{a}_1$ ne peut percer π_B et peut par conséquent être ignorée, si par contre, $\text{signe}(\mathbf{v}) < \text{signe}(\mathbf{v'})$, la chaîne $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_{\lceil \frac{m}{2} \rceil}$ peut être ignorée.

Enfin, si $\text{signe}(\mathbf{v}) = \text{signe}(\mathbf{v'})$, l'algorithme calcule le signe de $\mathbf{v''} = (\overrightarrow{N_B} \cdot \overrightarrow{\mathbf{a}_1, \mathbf{a}_{\lceil \frac{m}{2} \rceil}})$, et compare le signe de **v''** avec celui de **v**. Si $\text{signe}(\mathbf{v}) = \text{signe}(\mathbf{v''})$, la sous-chaine $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_{\lceil \frac{m}{2} \rceil}$ peut être ignorée, dans le cas contraire, la sous-chaîne $\mathbf{a}_{\lceil \frac{m}{2} \rceil}, \mathbf{a}_{\lceil \frac{m}{2} \rceil + 1}, \ldots, \mathbf{a}_m, \mathbf{a}_1$ peut être ignorée.

2.5 Test d'intersection de deux polygones convexes 2D

Cette section présente une méthode pour tester l'intersection de deux polygones strictement convexes du plan. La méthode proposée est basée sur le concept de somme de Minkowski qui permet de transformer efficacement le test initial en un test d'inclusion d'un point dans un polygone convexe.

2.5.1 Méthodes existantes

De nombreuses méthodes ont été développées afin de détecter l'intersection voire calculer la distance entre deux objets convexes dans l'espace bidimensionnel. Les principaux résultats proviennent de la géométrie algorithmique et cherchent, par conséquent, à atteindre une complexité asymptotique optimale. Après l'algorithme de Schwartz [66] qui détecte l'intersection de deux polygones convexes ayant respectivement m et n sommets en temps $O(\log^2(m+n))$, plusieurs solutions en temps $O(\log(m+n))$ ont été développées, on note parmi celles-ci les algorithmes de Dobkin et Kirkpatrick [18], et Chazelle et Dobkin [9, 10].

Cependant, bien que ces dernières méthodes garantissent des performances asymptotiques optimales, les constantes cachées impliquées dans leur complexité ne sont souvent pas négligeables lorsque $m+n$ est relativement petit. L'algorithme proposé par Chazelle et Dobkin [10] nécessite par exemple la recherche des extrema de fonctions bimodales et utilise des expressions

arithmétiques comme fonctions d'adressage. D'autre part, ces algorithmes nécessitent souvent l'utilisation de constructions telles que le calcul de points d'intersection entre les arêtes des polygones voire entre des droites support à ces polygones (cf. [18, 10]). Par conséquent, il n'est pas évident que de telles solutions soient efficaces pour certains problèmes pratiques où les polygones en entrée sont de petite taille. En ce sens, nous présentons dans la suite un algorithme sous-linéaire basé sur le concept de la somme de Minkowski qui s'avère particulièrement efficace pour ce type d'entrée. Son principe est identique à celui de l'algorithme proposé par Schwartz, cependant, contrairement à ce dernier, son implémentation est très efficace lorsque les polygones ont de l'ordre d'une dizaine de sommets, de plus, la robustesse de son implantation dépend uniquement de l'évaluation exacte de prédicats de type test d'orientation pour lesquels il existe de nombreuses méthodes de calcul rapide [5, 13, 17, 59, 60].

2.5.2 La somme de Minkowski

Le concept de somme de Minkowski est largement utilisé dans des domaines comme la planification de trajectoire [7, 29], l'analyse d'image [68] ou le placement de formes [48].

Étant donnés deux polygones A et B, la somme de Minkowski de A et B est définie par :

$$A \oplus B = \{a + b \mid a \in A,\ b \in B\} \tag{2.6}$$

où $a + b$ désigne la somme vectorielle des vecteurs a et b (cf. figure 2.17).

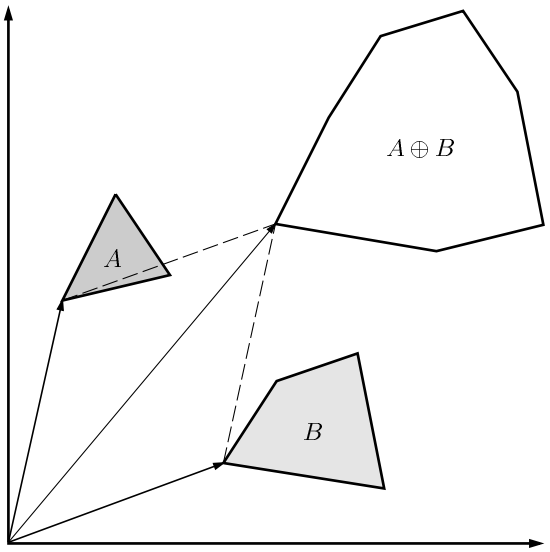

FIG. 2.17 – La somme de Minkowski de deux polygones.

Observation 2.5.1 [12] *Étant donnés deux objets A et B et $S = A \oplus B$, un point extrême de S dans la direction $\vec{d}$ correspond à la somme des points extrêmes de A et de B dans la direction $\vec{d}$.*

Théorème 2.5.2 [12] *Étant donnés A et B deux polygones convexes dans le plan ayant respectivement m et n arêtes, la somme de Minkowski $A \oplus B$ est un polygone convexe ayant au plus $m + n$ arêtes.*

Avec $(-B) = \{-b \mid b \in B\}$, c'est-à-dire en prenant le symétrique de B par rapport à l'origine, on remarque que les polygones A et B s'intersectent si et seulement si $A \oplus (-B)$ contient l'origine [12] (cf figure 2.18). Cette dernière propriété est à la base du test d'intersection proposé par Schwartz [66].

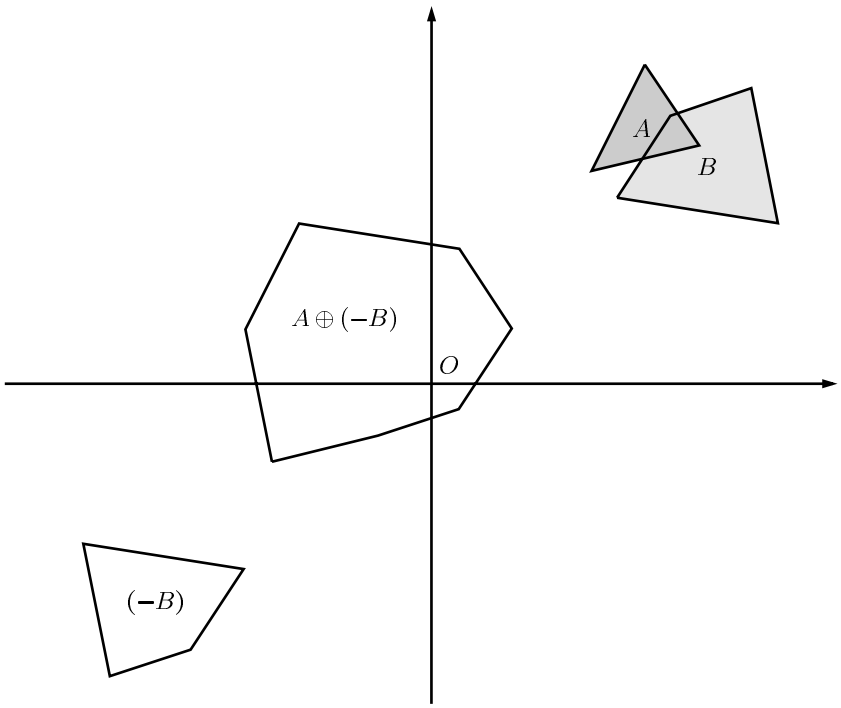

FIG. 2.18 – Détection d'intersection avec la somme de Minkowski.

2.5.3 Description de l'algorithme

Soit A et B deux polygones convexes dans le plan définis par la liste circulaire de leurs sommets ordonnés dans le sens direct, et $\overrightarrow{x^+}$ et $\overrightarrow{x^-}$ deux directions dans le plan correspondant respectivement aux vecteurs $(1, 0)$ et $(-1, 0)$.

L'algorithme détermine, dans un premier temps, les points extrêmes de chacun des polygones dans les directions $\overrightarrow{x^+}$ et $\overrightarrow{x^-}$ (*i.e.* les sommets ayant respectivement la plus grande et la plus petite abscisse). Soit donc, $\mathbf{l}_A$ et $\mathbf{r}_A$ (respectivement $\mathbf{l}_B$ et $\mathbf{r}_B$) les sommets extrêmes de A (respectivement de B) dans la direction $\overrightarrow{x^-}$ et $\overrightarrow{x^+}$.

D'après l'observation 2.5.1, on sait que les points extrêmes $\mathbf{l}_S$ et $\mathbf{r}_S$ de $S = A \oplus (-B)$ dans les directions $\overrightarrow{x^-}$ et $\overrightarrow{x^+}$ correspondent respectivement aux points $\mathbf{l}_A - \mathbf{r}_B$ et $\mathbf{r}_A - \mathbf{l}_B$. Ainsi, si l'abscisse de $\mathbf{l}_S$ est strictement supérieure à zéro ou si l'abscisse de $\mathbf{r}_S$ est strictement inférieure à zéro, S ne peut contenir l'origine. Sinon, l'algorithme teste l'orientation du point origine relativement au segment $\mathbf{l}_S\mathbf{r}_S$ (cf. figure 2.19). Si le point origine et le segment $\mathbf{l}_S\mathbf{r}_S$ sont colinéaires, S contient l'origine. Si le point origine se trouve à droite de $(\mathbf{l}_S\mathbf{r}_S)$, l'algorithme détermine à partir de $\mathbf{l}_S$ la séquence des arêtes de S, en fusionnant selon leur pente les arêtes de A entre $\mathbf{l}_A$ et $\mathbf{r}_A$ et les arêtes de B entre $\mathbf{r}_B$ et $\mathbf{l}_B$, jusqu'au premier sommet $\mathbf{s}_i$ de S ayant une abscisse positive ou nulle. Enfin, si le point origine se trouve à gauche de $(\mathbf{l}_S\mathbf{r}_S)$, l'algorithme détermine cette fois à partir de $\mathbf{r}_S$ la séquence des arêtes de S, en fusionnant selon leur pente les arêtes de A entre $\mathbf{r}_A$ et $\mathbf{l}_A$ et les arêtes de B entre $\mathbf{l}_B$ et $\mathbf{r}_B$, jusqu'au premier sommet $\mathbf{s}_i$ de S ayant une abscisse négative ou nulle. Une fois l'arête $\mathbf{s}_{i-1}\mathbf{s}_i$ déterminée, la solution au test initial est obtenue en évaluant le signe du test d'orientation $[\mathbf{s}_{i-1}, \mathbf{s}_i, \mathbf{o}]$. Si celui-ci est positif ou nul, les deux polygones en entrée s'intersectent, sinon les polygones sont disjoints.

Les différentes étapes de l'algorithme sont résumées ci-dessous, on suppose A et B orientés dans le sens direct :

1. Recherche des sommets d'abscisse minimale, $\mathbf{l}_A$ et $\mathbf{l}_B$, et maximale, $\mathbf{r}_A$ et $\mathbf{r}_B$, de chacun des polygones en entrée.
2. Soit $\mathbf{l}_S = \mathbf{l}_A - \mathbf{r}_B$ et $\mathbf{r}_S = \mathbf{r}_A - \mathbf{l}_B$. Retourner *faux* si $\mathbf{l}_S.x > 0$ ou $\mathbf{r}_S.x < 0$.
3. Calcul de la séquence d'arêtes de S connectant soit $\mathbf{l}_S$ à $\mathbf{s}_i$ soit $\mathbf{r}_S$ à $\mathbf{s}_i$ selon le signe de $[\mathbf{l}_A - \mathbf{r}_B, \mathbf{r}_A - \mathbf{l}_B, \mathbf{o}]$, retourner *vrai* si $[\mathbf{l}_A - \mathbf{r}_B, \mathbf{r}_A - \mathbf{l}_B, \mathbf{o}] = 0$.
4. Retourner *faux* si $[\mathbf{s}_{i-1}, \mathbf{s}_i, \mathbf{o}] \geq 0$, *vrai* sinon.

2.5.4 Complexité de l'algorithme

L'étape 1 est réalisée en effectuant un parcours linéaire de la liste des sommets de A et de B et nécessite $m + n - 2$ comparaisons. L'étape 2 revient à comparer les valeurs de $\mathbf{l}_A$ et $\mathbf{r}_B$ et les valeurs de $\mathbf{r}_A$ et $\mathbf{l}_B$ qui sont les coordonnées des sommets en entrée. Une fois le signe de $[\mathbf{l}_A - \mathbf{r}_B, \mathbf{r}_A - \mathbf{l}_B, \mathbf{o}]$ déterminé, l'étape 3 revient à fusionner deux sous-listes d'arêtes triées selon leur pente et nécessite dans le pire des cas $m + n - 3$ tests d'orientation (les sous-listes d'arêtes

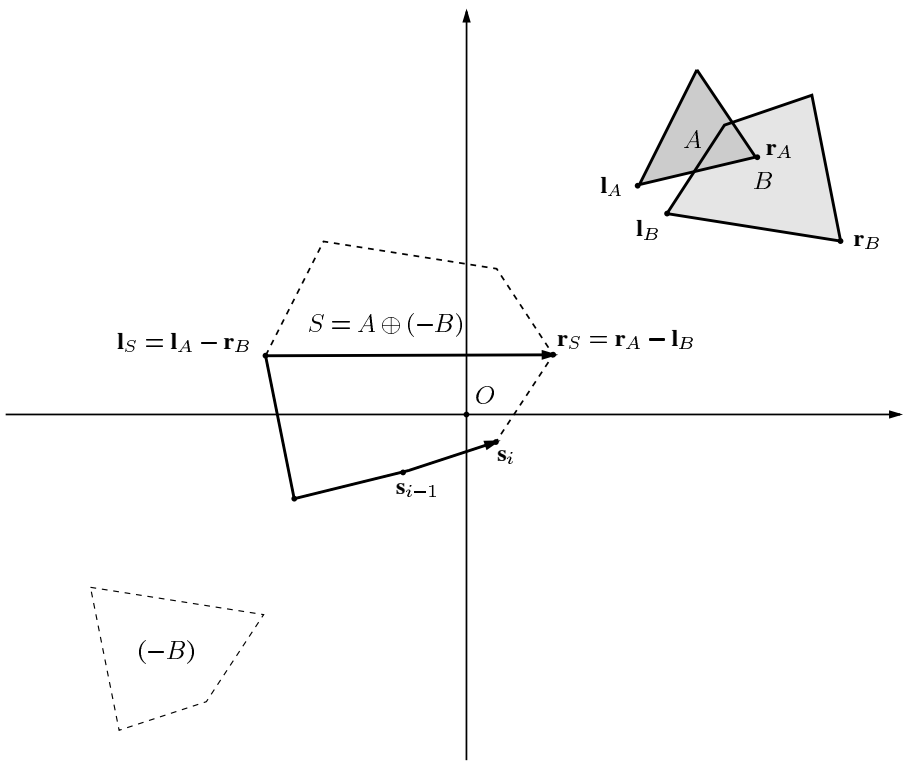

FIG. 2.19 – Si $[\mathbf{l}_S, \mathbf{r}_S, \mathbf{o}] < 0$, l'intersection des deux polygones en entrée dépend du signe de $[\mathbf{s}_{i-1}, \mathbf{s}_i, \mathbf{o}]$, où $\mathbf{s}_i$ correspond au premier sommet, rencontré en parcourant la frontière de S dans le sens direct à partir de $\mathbf{l}_S$, ayant une abscisse positive.

connectant $\mathbf{l}_A$ à $\mathbf{r}_A$ et $\mathbf{l}_B$ à $\mathbf{r}_B$ ayant respectivement une taille maximale égale à $m-1$ et $n-1$) et $m+n-2$ comparaisons. L'étape 4 effectue enfin un test d'orientation supplémentaire.

L'algorithme dans sa totalité effectue donc dans le pire des cas $m+n$ comparaisons de coordonnées et $m+n+1$ tests d'orientation bidimensionnels ($m+n-1$ si l'orientation de chaque polygone en entrée est connue).

On note que les entrées des tests d'orientation intervenant aux étapes 3 et 4 ne correspondent pas aux sommets des objets en entrée mais s'expriment sous la forme de différences (potentiellement inexactes) entre les sommets des objets de départ. Afin de remédier à ce problème, on calcule en pratique ces tests à l'aide de l'opérateur suivant :

$$[\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}]' = - \begin{vmatrix} a_x - b_x & c_x - d_x \\ a_y - b_y & c_y - d_y \end{vmatrix}$$

où $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, et $\mathbf{d}$ désignent quatre points bidimensionnels. On a ainsi, par exemple, $[\mathbf{l}_A - \mathbf{r}_B, \mathbf{r}_A - \mathbf{l}_B, \mathbf{o}] = [\mathbf{l}_A, \mathbf{r}_B, \mathbf{r}_A, \mathbf{l}_B]'$. Nous renvoyons le lecteur aux références citées à la fin de la section

2.5.1 pour des exemples de méthodes efficaces et robustes permettant l'évaluation du signe de ce type de déterminant.

2.6 Tests d'intersection CGAL

La bibliothèque CGAL[2], issue d'une collaboration entre huit équipes de recherche financées par l'Union Européenne, vise à promouvoir l'usage des algorithmes géométriques au sein du monde industriel et scientifique. Cette bibliothèque, écrite dans le langage C++, est organisée en trois parties principales : le noyau (*kernel library*), la bibliothèque d'algorithmes fondamentaux (*basic library*), et la bibliothèque support (*support library*).

Le noyau regroupe les objets géométriques de base de taille finie (points, droites, cercles, segments, ...), ainsi que les opérations qui les manipulent (prédicats, constructions, intersections, ...). La bibliothèque d'algorithmes fondamentaux regroupe, quant à elle, des structures de données et des algorithmes permettant de créer et de manipuler des objets tels que des enveloppes convexes, des triangulations, des cartes planaires, des polyèdres, etc... Tous ces algorithmes s'interfacent avec le noyau et utilisent de manière intensive le mécanisme des classes de *traits* afin d'offrir une plus grande flexibilité au programmeur. Enfin, la bibliothèque support regroupe les interfaces avec les outils de visualisation externes, les outils arithmétiques, etc...

La bibliothèque CGAL offre un cadre idéal pour étudier les différentes solutions aux problèmes de robustesse et permet d'obtenir des solutions génériques et efficaces aux problèmes posés par les algorithmes fondamentaux du fait de l'interchangeabilité des composants. La bibliothèque offre en effet plusieurs possibilités pour traiter les problèmes de robustesse. D'une part, tous les algorithmes sont paramétrés par le type de nombres qui est utilisé pour représenter les coordonnées des objets géométriques. De cette façon, il est très facile d'utiliser une bibliothèque de types de nombres exacts. De plus, les prédicats dans CGAL sont sous la forme de fonctions génériques (*template*). Grâce au mécanisme de spécialisation, l'utilisation de prédicats filtrés et exacts à la place de la version générique se fait ainsi automatiquement en fonction du type de nombres utilisé.

Le package *Intersection* La package *Intersection* fournit deux familles d'algorithmes à la bibliothèque, d'une part, des tests permettant la détection d'intersection de deux d'objets (bi- ou

[2] `http ://www.cgal.org`

tridimensionnels) et, d'autre part, des routines permettant de calculer l'objet correspondant à l'intersection effective de ces objets.

Pour cela, le package *Intersection* surcharge en fonction du type des objets en argument les fonctions génériques *CGAL : :do_intersect()* et *CGAL : :intersection()* respectivement dédiées à la détection et au calcul d'intersections. La bibliothèque fournit une version de ces deux fonctions pour la plupart des objets bidimensionnels du noyau.

Notre contribution a consisté en l'intégration dans la bibliothèque [8] des prédicats d'intersection entre un triangle et la plupart des objets tridimensionnel du noyau.

Plus précisément, la fonction

bool *CGAL : :do_intersect (Type1<R> obj1, Type2<R> obj2)*

a été surchargée afin de permettre son utilisation pour les types d'objets suivants :

Type1	*Type2*	# tests orientation
Triangle_2<R>	*Triangle_2<R>*	10
Triangle_3<R>	*Point_3<R>*	4
Triangle_3<R>	*Line_3<R>*	3
Triangle_3<R>	*Ray_3<R>*	4
Triangle_3<R>	*Segment_3<R>*	4
Triangle_3<R>	*Triangle_3<R>*	8

Le tableau donne pour chacun des prédicats, le nombre maximal (i.e dans le pire des cas) de tests d'orientation (bi- ou tridimensionels selon le cas) nécessaires à son évaluation.

Chapitre 3

Constructions géométriques élémentaires certifiées arrondies

Comme nous l'avons vu à la section 1.11, le cas des constructeurs géométriques et plus particulièrement le cas des constructions cascadées met à mal le paradigme du calcul exact. Chaque construction augmente l'espace mémoire nécessaire à la représentation exacte des objets, et le fait de cascader ces constructions conduit à des algorithmes dont le degré n'est plus borné a priori.

Ce chapitre tente de répondre à la question : comment calculer l'arrondi d'un objet géométrique simple (tel qu'un point d'intersection entre deux segments du plan) avec des garanties sur la qualité de l'arrondi ? De la même manière que l'arithmétique IEEE garantit un arrondi certifié pour chaque opération arithmétique élémentaire permettant par le calcul de propagation d'erreurs d'estimer l'erreur de calculs plus complexes, on souhaite en effet garantir un arrondi certifié de constructions géométriques élémentaires afin de permettre la conception et l'analyse d'algorithmes robustes construisant des structures plus complexes. Dans une certaine mesure, ce point de vue est une alternative à l'application naïve du paradigme du calcul exact : le calcul exact garantit la robustesse des algorithmes mais accroît l'espace mémoire nécessaire à la représentation des objets manipulés ; notre approche préconise l'emploi d'un calcul exact *localement* afin d'arrondir le résultat de constructions selon une spécification précise et déterministe et de contrôler ainsi la taille mémoire des objets construits.

Ce chapitre est organisé de la manière suivante. Nous présentons, tout d'abord, à la section 3.1 le modèle de géométrie et l'arithmétique adoptés. La section 3.2 décrit la méthode de filtrage arithmétique utilisée dans la suite. La section 3.3 aborde le problème de l'arrondi de points d'intersection de segments dans le plan qui constitue le constructeur de base pour les algorithmes d'arrondi d'arrangements de segments (cf. section 4.1). Les sections 3.4 et 3.5 généralisent la

technique de filtrage utilisée au cas du calcul du centre du cercle circonscrit à trois points du plan et du calcul du centre de la sphère circonscrite à quatre points dans l'espace utilisés dans les algorithmes de reconstruction de surface[1]. La section 3.6 reprend finalement la méthode développée par H.S. Lee et R.S. Chang [47] afin d'arrondir un sommet convexe au plus proche point entier sous la contrainte que le point arrondi se trouve dans le secteur convexe défini par les deux arêtes incidentes au sommet. Enfin, la section 3.7 présente une nouvelle méthode permettant l'arrondi d'un sommet d'un polygone convexe au plus proche point entier appartenant à ce polygone.

3.1 Modèle de géométrie à précision fixée

3.1.1 Arithmétique flottante et nombres entiers

Comme nous l'avons vu à la section 1.4, le résultat d'un calcul flottant est toujours arrondi à un nombre représentable. Par conséquent, si on peut garantir que le résultat exact d'une opération de base est représentable, alors les nombres flottants font en fait du calcul exact. Les flottants double précision font ainsi d'excellents entiers sur 53 bits avec les avantages suivants : gestion plus facile des débordements, possibilité de calculer avec des multiples de 2^k directement, sans faire explicitement des multiplications par 2^{-k} pour repasser aux entiers. De plus, sur la plupart des architectures, le calcul sur les flottants double précision est plus rapide que les entiers 64 bits voire même plus rapide que les entiers 32 bits.

La virgule flottante permet de représenter les nombres avec une précision relative à sa valeur. Un nombre proche de zéro aura plus de chiffres après la virgule qu'un nombre plus grand. Pour un certain nombre d'applications géométriques où les données sont des points, l'origine du repère ne joue pas un rôle particulier et il n'est pas pertinent de calculer avec plus de précision les coordonnées des points proches de l'origine. L'utilisation de nombres en virgule fixe, où le nombre de chiffres après la virgule est toujours le même peut donc se justifier.

Si les données initiales sont des nombres simple précision en virgule fixe, c'est-à-dire des entiers de l'intervalle $[-2^{24}, 2^{24}]$ (à un facteur multiplicatif près), l'utilisation de l'arithmétique double précision sur 53 bits de mantisse permet d'avoir un certain nombre de calculs rapides et exacts. L'arithmétique double précision garantit par exemple le calcul exact du déterminant d'une matrice 2×2 lorsque les coefficients de la matrice sont des entiers bornés en valeur absolue par 2^{26}.

Étant donnés **a** et **b** deux nombres entiers représentables sur respectivement $p_{\mathbf{a}}$ et $p_{\mathbf{b}}$ bits (un

[1] Les sections 3.3 à 3.5 ont fait l'objet du rapport de recherche [15].

nombre entier **a** est représentable sur $p_{\mathbf{a}}$ bits si $|\mathbf{a}| < 2^{p_{\mathbf{a}}}$), le nombre de bits nécessaires à la représentation exacte du résultat des opérations d'addition, soustraction et multiplication entre **a** et **b** est donné par la Table 3.1.

E	p_E
Entier **a**	$\log_2(\mathbf{a}) + 1$
$\mathbf{a} \oplus \mathbf{b},\ \mathbf{a} \ominus \mathbf{b}$	$\max(p_{\mathbf{a}}, p_{\mathbf{b}}) + 1$
$\mathbf{a} \otimes \mathbf{b}$	$p_{\mathbf{a}} + p_{\mathbf{b}}$

TAB. 3.1 – Nombre de bits maximum nécessaires à la représentation du résultat de l'addition, de la soustraction et de la multiplication de deux nombres entiers **a** et **b** représentables sur $p_{\mathbf{a}}$ et $p_{\mathbf{b}}$ bits.

3.1.2 Grille uniforme entière

Afin de se conformer au mode d'arrondi IEEE au plus proche utilisant la règle de l'arrondi au bit pair (*round-to-even digit rule*) en cas d'ambiguïté, nous considérons les modèles géométriques suivants.

Étant donné $\mathbf{C}_{pair} = [-\frac{1}{2}, \frac{1}{2}]$ et $\mathbf{C}_{impair} =]-\frac{1}{2}, \frac{1}{2}[$, la fonction d'arrondi au plus proche entier associe à chaque nombre réel x l'unique entier i tel que $x \in i + \mathbf{C}_i$, où $+$ désigne la somme de Minkowski et $\mathbf{C}_i$ est égal à $\mathbf{C}_{pair}$ ou $\mathbf{C}_{impair}$ selon la parité de i.

De manière similaire, on pose $\mathbf{C}_{i,j} = \mathbf{C}_i \times \mathbf{C}_j$ et on définit la *zone d'influence* du point entier bidimensionnel $p = (i, j)$ comme étant l'ensemble des points réels appartenant à la région $p + \mathbf{C}_{i,j}$.

Enfin, étant donné un point tridimensionnel $p = (x, y, z)$ à coordonnées réelles, on définit le plus proche point entier de p, l'unique point entier $g = (i, j, k)$ tel que $p \in g + \mathbf{C}_{i,j,k}$ où $\mathbf{C}_{i,j,k} = \mathbf{C}_i \times \mathbf{C}_j \times \mathbf{C}_k$.

Nous supposons dans la suite que les seuls points représentables sont les points (bi- ou tridimensionnels) dont les coordonnées sont entières et bornées en valeur absolue par 2^{24}, *i.e.* appartenant à l'intervalle $\{-2^{24}, \ldots, 2^{24}\} \subset \mathbb{Z}$.

3.2 Technique de filtrage

Le principe des constructeurs utilisés pour l'arrondi de constructions élémentaires au plus proche point entier est le suivant : dans un premier temps, une approximation des coordonnées de la construction exacte est calculée à l'aide de l'arithmétique flottante double précision. À partir de cette approximation et du calcul d'une borne sur l'erreur d'approximation commise, un intervalle contenant la valeur exacte de la construction est déduit (cf. figure 3.1.a).

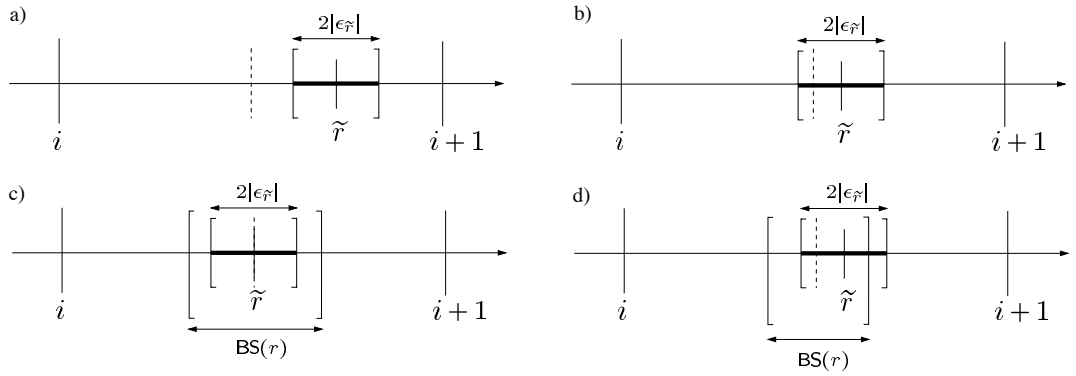

FIG. 3.1 – Technique de filtrage utilisée pour le calcul du plus proche point entier de constructions élémentaires. On note respectivement $\widetilde{r}$, $\epsilon_{\widetilde{r}}$, et $\mathsf{BS}(r)$ l'approximation, la borne d'erreur sur l'approximation et l'intervalle séparant les deux valeurs possibles de l'expression encadrant le milieu de l'intervalle entier.

Un premier filtre teste alors si le milieu de l'intervalle formé par les deux plus proches nombres entiers de l'approximation calculée appartient à l'intervalle d'erreur construit. Si ce n'est pas le cas (cf. figure 3.1.a), l'approximation calculée est suffisante pour garantir la validité du résultat, et le plus proche point entier de l'approximation est retourné. Sinon, au moins deux résultats sont possibles [2] et l'approximation ne permet pas d'en déduire la solution correcte (cf. figure 3.1.b). Un second filtre basé sur la *borne de séparation* [3] de l'expression à évaluer est alors utilisé afin de détecter les cas où le résultat se situe exactement au milieu de l'intervalle formé par les deux entiers encadrant l'approximation calculée. Ce dernier détermine la distance entre les deux valeurs possibles de l'expression encadrant le milieu de l'intervalle entier et teste l'inclusion de l'intervalle d'erreur dans cet intervalle de séparation. Si l'intervalle d'erreur est totalement

[2] Pour l'ensemble des constructeurs filtrés abordés dans ce chapitre, l'erreur d'approximation sera inférieure à 1. Par conséquent, il existera, en cas d'échec du filtre, exactement deux résultats possibles.

[3] Étant donnée une expression E ayant pour valeur ξ, une borne de séparation $sep(E)$ est un nombre réel positif tel que $\xi \neq 0$ implique $|\xi| \geq sep(E)$.

inclus dans cet intervalle (cf. figure 3.1.c), le résultat exact de la construction se situe exactement au milieu de l'intervalle entier et le plus proche point entier de ce nombre (calculé à l'aide de la règle d'arrondi au bit pair) est retourné. Dans le cas contraire (cf. figure 3.1.d), le résultat du constructeur reste ambiguë et un calcul plus précis (utilisant par exemple la technique des expansions flottantes) est invoqué.

3.3 Point d'intersection de deux segments du plan

Constructeur 1 : *Étant donnés quatre points de la grille entière* $p_1 = (x_1, y_1)$, $p_2 = (x_2, y_2)$, $p_3 = (x_3, y_3)$, *et* $p_4 = (x_4, y_4)$, *déterminer si les segments* $s_1 = p_1p_2$ *et* $s_2 = p_3p_4$ *s'intersectent en un point unique et calculer dans ce cas les coordonnées du plus proche point entier du point d'intersection exact (cf. figure 3.2).*

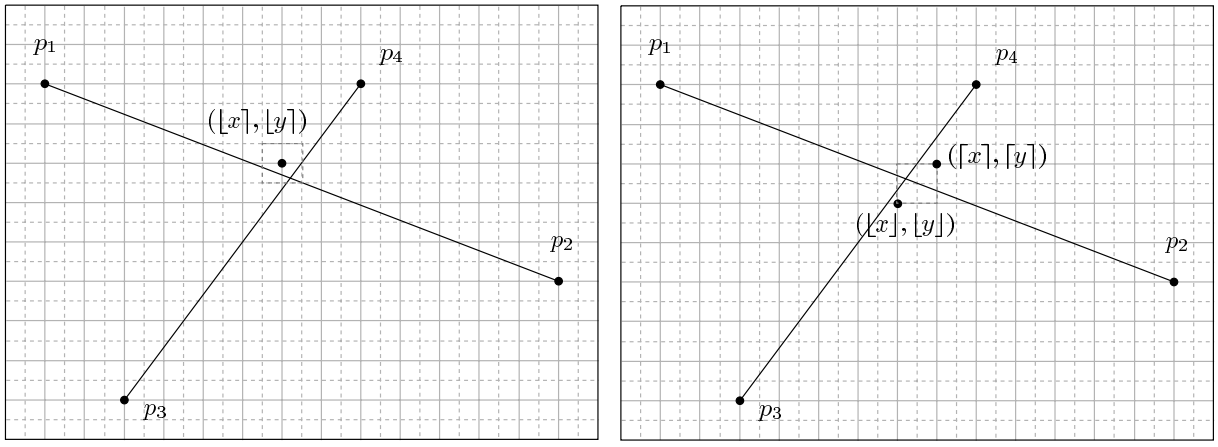

FIG. 3.2 – Le constructeur 1 (à gauche) calcule le plus proche point entier du point d'intersection exact. Le constructeur 2 (à droite) calcule les deux points entiers $(\lfloor x \rfloor, \lfloor y \rfloor)$ et $(\lceil x \rceil, \lceil y \rceil)$ encadrant le point d'intersection exact.

Formulation Étant donnés $u = [p_1, p_4, p_3] = p_1p_4 \times p_1p_3 = \begin{vmatrix} x_{13} & x_{43} \\ y_{13} & y_{43} \end{vmatrix}$ et $v = [p_2, p_4, p_3] = p_2p_4 \times p_2p_3 = \begin{vmatrix} x_{23} & x_{43} \\ y_{23} & y_{43} \end{vmatrix}$ où x_{ij} désigne la différence $x_{p_i} - x_{p_j}$, on distingue les situations suivantes :

- si $uv > 0$, l'intersection des deux segments est vide,
- si $u = 0$ et $v = 0$, les deux segments sont colinéaires et le prédicat d'intersection se réduit à ordonner les extrémités des segments sur la droite support aux segments,

- si $u = 0$ et $v \neq 0$, le segment s_1 intersecte la droite (p_3p_4) en p_1, et $s_1 \cap s_2 = p_1$ si et seulement si $[p_1, p_2, p_3].[p_1, p_2, p_4] \leq 0$,
- si $u \neq 0$ et $v = 0$, le segment s_1 intersecte la droite (p_3p_4) en p_2, et $s_1 \cap s_2 = p_2$ si et seulement si $[p_1, p_2, p_3].[p_1, p_2, p_4] \leq 0$,
- enfin, si $uv < 0$, le segment s_1 intersecte la droite (p_3p_4) entre p_1 et p_2, et les segments s_1 et s_2 s'intersectent en un point unique si et seulement si $[p_1, p_2, p_3].[p_1, p_2, p_4] \leq 0$.

La première étape de l'algorithme détermine en fonction des valeurs de u et v à quel cas générique mentionné ci-dessus correspondent les segments en entrée. Le constructeur n'est alors invoqué que lorsque $uv < 0$ et $[p_1, p_2, p_3].[p_1, p_2, p_4] \leq 0$ (dans tous les autres cas, l'intersection est soit vide, soit définie par les sommets en entrée).

Étant donnés deux segments $s_1 = p_1p_2$ et $s_2 = p_3p_4$ s'intersectant en un point unique, le point d'intersection $p = s_1 \cap s_2$ peut être calculé à l'aide de la formule suivante :

$$p = p_1 + (p_2 - p_1)\frac{\mathrm{N}}{\mathrm{D}} \quad \text{où} \quad \mathrm{N} = u \quad \text{et} \quad \mathrm{D} = u - v \tag{3.1}$$

Calcul d'une borne sur l'erreur d'approximation On rappelle que l'on suppose les coordonnées des extrémités des segments représentées par des entiers simple précision (*i.e.* des entiers appartenant à l'intervalle $[-2^{24}, 2^{24}]$), et toutes les opérations calculées à l'aide de l'arithmétique flottante double précision.

Soit $\widetilde{x}$ le nombre flottant double précision obtenu par le calcul approché de $x_1 + x_{21}\frac{\mathrm{N}}{\mathrm{D}}$, c'est-à-dire le nombre donné par l'évaluation de $x_1 \oplus ((x_{21} \otimes \mathrm{N}) \oslash \mathrm{D})$. Le terme x_{21} correspond à une différence entre des coordonnées et est donc représentable sur $24 + 1$ bits. Les termes N et D correspondent respectivement à un déterminant 2×2 et à une somme de déterminants 2×2 dont les coefficients sont aussi des différences et sont par conséquent calculés exactement à l'aide de l'arithmétique flottante double précision disposant de $53 > 2 \times 24 + 3 = 51$ bits.

Puisque $\mathbf{u} = 2^{-53}$ est une borne sur l'erreur relative de chacune des opérations arithmétiques double précision, $\widetilde{x_i} = (x_{21} \otimes \mathrm{N}) \oslash \mathrm{D}$ vérifie alors les inégalités suivantes :

$$|\widetilde{x_i}|(1 - 2\mathbf{u}) \lesssim |\widetilde{x_i}|(1 - \mathbf{u})^2 \leq \left|\frac{x_{21}\mathrm{N}}{\mathrm{D}}\right| \leq |\widetilde{x_i}|(1 + \mathbf{u})^2 \approx |\widetilde{x_i}|(1 + 2\mathbf{u})$$

d'où

$$\left|\widetilde{x_i} - \frac{x_{21}\mathrm{N}}{\mathrm{D}}\right| \lesssim 2\mathbf{u}\,|\widetilde{x_i}| << 1.$$

Puisque x_1 et $\lfloor\widetilde{x_i}\rceil$ sont des entiers inférieurs ou égaux à 2^{25} en valeur absolue, il n'y a pas d'erreur commise dans l'addition finale. Le plus proche entier $\lfloor\widetilde{x}\rceil$ de la valeur approchée $\widetilde{x}$ est donc différent du plus proche entier de la valeur exacte x seulement lorsque $\lfloor\widetilde{x_i}(1 - 2\mathbf{u})\rceil \neq$

$\lfloor \widetilde{x_i}(1+2\mathbf{u}) \rceil$, c'est-à-dire

$$\frac{1}{2} - \left| \widetilde{x_i} - \lfloor \widetilde{x_i} \rceil \right| \leq 2\mathbf{u}|\widetilde{x_i}|. \tag{3.2}$$

On rappelle que la plus proche valeur entière d'un nombre flottant double précision X peut être obtenue en additionnant et en soustrayant la valeur 3×2^{51} à X lorsque la valeur absolue de X est inférieure à 2^{51} et le mode d'arrondi utilisé est le mode d'arrondi au plus proche. Le résultat de chacune des opérations de soustraction apparaissant dans l'expression (3.2) est représentable par un nombre flottant double précision et est donc calculé exactement. Enfin, on note que $2\mathbf{u} = 2^{-52}$ est également représentable et que le produit $2\mathbf{u} \otimes |\widetilde{x_i}|$ est calculé exactement puisque celui-ci nécessite seulement une mise à jour de l'exposant de $|\widetilde{x_i}|$.

Une version filtrée du constructeur 1 peut donc être obtenue en calculant dans un premier temps la valeur $\widetilde{x_i}$ obtenue avec l'arithmétique double précision, puis en testant le prédicat :

$$\frac{1}{2} \ominus \left| \widetilde{x_i} \ominus \lfloor \widetilde{x_i} \rceil \right| > 2\mathbf{u}|\widetilde{x_i}|.$$

Si le prédicat est vérifié, l'approximation calculée à l'aide de l'arithmétique flottante est suffisamment précise pour permettre de certifier le résultat et l'algorithme retourne le plus proche entier de l'abscisse du point d'intersection exact ($x_1 \oplus \lfloor \widetilde{x_i} \rceil$). Sinon, l'algorithme cascade un deuxième filtre arithmétique basé sur les bornes de séparation de l'expression.

Borne de séparation Dans les cas d'échec du premier filtre, on sait que l'intervalle d'erreur construit autour de l'approximation $\widetilde{x_i}$ contient le milieu m de l'intervalle formé par les deux entiers encadrant l'approximation. Le second filtre utilisé détermine alors les deux plus proches valeurs possibles de la construction exacte encadrant m et teste leur appartenance à l'intervalle d'erreur.

Étant donnés deux rationnels de même dénominateur $r = \frac{n}{d}$ et $r' = \frac{n'}{d}$, il est facile de montrer que si r et r' sont distincts (c'est-à-dire $n \neq n'$) alors $|r - r'| \geq \frac{1}{|d|}$ (la valeur absolue de la différence entre les numérateurs de r et r' ne pouvant être dans ce cas inférieure à 1). Ainsi, en remarquant qu'il est toujours possible d'exprimer m, le milieu de l'intervalle entier encadrant l'approximation $\widetilde{x_i}$, sous la forme d'un rationnel ayant un dénominateur pair, on a forcément $\left| \frac{x_{21\mathrm{N}}}{\mathrm{D}} - m \right| \geq \frac{1}{2|\mathrm{D}|}$ si x_i et m sont distincts.

Par conséquent, comme l'intervalle d'erreur autour de $\widetilde{x_i}$ contenant x_i est de longueur $4\mathbf{u}|\widetilde{x_i}|$, m est la seule valeur possible de la construction si et seulement si

$$\frac{1}{2|\mathrm{D}|} > 4\mathbf{u}|\widetilde{x_i}|$$

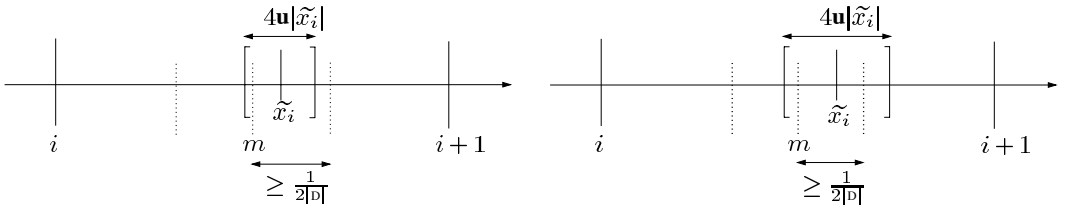

FIG. 3.3 – À gauche, $\frac{1}{2|\mathrm{D}|} > 4\mathbf{u}|\widetilde{x}_i|$ et m est la seule valeur possible de la construction, à droite $\frac{1}{2|\mathrm{D}|} < 4\mathbf{u}|\widetilde{x}_i|$ et le résultat de la construction reste ambiguë.

c'est-à-dire

$$|\widetilde{x}_i\mathrm{D}| < (8\mathbf{u})^{-1} = 2^{50} \tag{3.3}$$

(cf. figure 3.3).

Comme 2^{51} est exactement représentable et puisque le résultat de l'opération flottante $\widetilde{x}_i \otimes \mathrm{D}$ est arrondie au plus proche nombre double précision, une condition suffisante pour vérifier l'inégalité 3.3 est

$$|\widetilde{x}_i \otimes \mathrm{D}| < 2^{51}.$$

Si ce dernier prédicat est vérifié, m est le seul résultat possible et l'algorithme retourne la valeur $\lfloor x_1 \oplus m \rceil$. Dans le cas contraire, le résultat reste ambiguë et l'algorithme conclut en déterminant le signe exact de $(2x_i - \lfloor 2\widetilde{x}_i \rceil)$ à l'aide de techniques d'expansions [62, 69] (cf. Algorithme 1).

Remarque : Un résultat plus fort a été proposé par Priest [63] pour le cas des segments dont les coordonnées des extrémités sont des nombres flottants simple précision (et non plus uniquement des entiers). Son algorithme utilise une technique d'expansion flottante basée sur l'arithmétique double précision et retourne le plus proche nombre flottant simple précision représentable du résultat exact. Cependant, contrairement aux affirmations de l'auteur, le fait de calculer le quotient des approximations du numérateur et du dénominateur arrondis au plus proche nombre flottant double précision ne suffit pas à garantir l'arrondi du résultat au plus proche nombre flottant simple précision. En prenant,

$$p_1 = \begin{pmatrix} -4\alpha + 12 \\ 3\alpha - 7 \end{pmatrix},\ p_2 = \begin{pmatrix} 4\alpha \\ 3\alpha + 1 \end{pmatrix},\ p_3 = \begin{pmatrix} 3\alpha - 3 \\ -4\alpha + 7 \end{pmatrix},\ \text{et } p_4 = \begin{pmatrix} 3\alpha + 1 \\ 4\alpha - 1 \end{pmatrix}$$

par exemple, et si α est choisi de manière à ce que les segments p_1p_2 et p_3p_4 s'intersectent, l'abscisse x du point d'intersection des deux segments est égale à $3\alpha + \frac{1}{2} - \frac{1}{2\alpha^2 - 5\alpha + 2}$. Avec

```
double C = 3 ⊗ 2^51 ;
double round_to_int( double X ) { retourner ((C ⊕ X) ⊖ C) ;}

double sign( double X ) { retourner (( X > 0 ) ? 1 : (( X < 0 ) ? -1 : 0 )) ;}

double C' = 3 ⊗ 2^76 ;
#define split( A, A_h, A_l ) A_h = (C' ⊕ A) ⊖ C'; A_l = A ⊖ A_h;
/* Étant donné un entier A représentable sur p ≤ 53 bits, la fonction split calcule
une expansion non recouvrante de A telle que A_l < 2^25 et A = A_h + A_l.*/

segment2d_segment2d_intersection( P1, P2, P3, P4 ) {
/* Précondition : p1p2 et p3p4 s'intersectent en un point unique */

N = (X3 ⊖ X1) ⊗ (Y4 ⊖ Y1) ⊕ (X1 ⊖ X4) ⊗ (Y3 ⊖ Y1) ;
D = (X2 ⊖ X1) ⊗ (Y3 ⊖ Y4) ⊕ (X4 ⊖ X3) ⊗ (Y2 ⊖ Y1) ;
X = ((X2 ⊖ X1) ⊗ N) ⊘ D ;
X̃ = round_to_int(X) ; /* 2 additions */

si ( 0.5 ⊖ |X ⊖ X̃| > 2u ⊗ |X| ) /* 2 valeurs absolues, 2 additions, 1 multiplication, 1 comparaison */
alors retourner ( X1 ⊕ X̃ ) ; /* 1 addition */

si ( |X ⊗ D| < 2^50 ) /* 1 valeur absolue, 1 multiplication, 1 comparaison */
alors retourner round_to_int(X1 ⊕ 0.5 ⊗ round_to_int(2 ⊗ X)) ;
/* 5 additions, 2 multiplications */

T1 = 2 ⊗ (X2 ⊖ X1);
T2 = round_to_int(2 ⊗ X);
split(N, Nh, Nl) ;
split(D, Dh, Dl) ;
T = (T1 ⊗ Nh ⊖ T2 ⊗ Dh) ⊕ (T1 ⊗ Nl ⊖ T2 ⊗ Dl);
retourner round_to_int(X1 ⊕ 0.5 ⊗ (T2 ⊕ sign(D) ⊗ sign(T))) ;
/* 8 multiplications, 16 additions, 2 signes */ }
```

Algorithme 1: Constructeur 1 (calcul de l'abscisse uniquement).

$\alpha = 2^{22}-1$, le calcul approché du quotient des valeurs du numérateur et du dénominateur arrondis au plus proche nombre double précision donne $\widetilde{x}$ =12582909,5. Le fait de convertir ce nombre, en utilisant en cas d'ambiguïté la règle de l'arrondi au nombre pair, donne alors 12582910 qui est différent du plus proche nombre flottant simple précision de x valant 12582909.

Constructeur 2 : *Étant donnés quatre points de la grille entière* $p_1 = (x_1, y_1)$, $p_2 = (x_2, y_2)$, $p_3 = (x_3, y_3)$, *et* $p_4 = (x_4, y_4)$, *déterminer si les segments* $s_1 = p_1p_2$ *et* $s_2 = p_3p_4$ *s'intersectent en un point unique et calculer dans ce cas les coordonnées des points entiers* $(\lfloor x \rfloor, \lfloor y \rfloor)$ *et* $(\lceil x \rceil, \lceil y \rceil)$ *où* (x, y) *sont les coordonnées du point d'intersection (cf. figure 3.2).*

Le code filtré du constructeur 2 se déduit directement des bornes calculées précédemment.

La première partie de l'algorithme (jusqu'à l'instruction $\widetilde{\mathrm{X}}$ = `round_to_int`(X)) est strictement identique à celle du constructeur 1, le pseudo-code de la seconde partie est donné par l'algorithme 2.

```
...
si ( |X ⊖ X̃| > 2u ⊗ |X| ) alors retourner ( X1 ⊕ floor(X) , X1 ⊕ ceil(X) );
si ( |X ⊗ D| < 2^50 ) alors retourner ( X1 ⊕ X̃ , X1 ⊕ X̃ );
T1 = (X2 ⊖ X1);
split(N, Nh, Nl);
split(D, Dh, Dl);
T = (T1 ⊗ Nh ⊖ X̃ ⊗ Dh) ⊕ (T1 ⊗ Nl ⊖ X̃ ⊗ Dl);
S = X̃ ⊕ 0.5 ⊗ ( sign(D) ⊗ sign(T) );
retourner ( X1 ⊕ floor(S) , X1 ⊕ ceil(S) );
```

Algorithme 2: Constructeur 2 (calcul de l'abscisse uniquement).

3.4 Centre du cercle circonscrit à un triangle dans le plan

Constructeur 3 : *Étant donnés trois points de la grille entière* $p_1 = (x_1, y_1)$, $p_2 = (x_2, y_2)$, *et* $p_3 = (x_3, y_3)$, *calculer le plus proche point entier du centre du cercle circonscrit à ces trois points (cf. figure 3.4).*

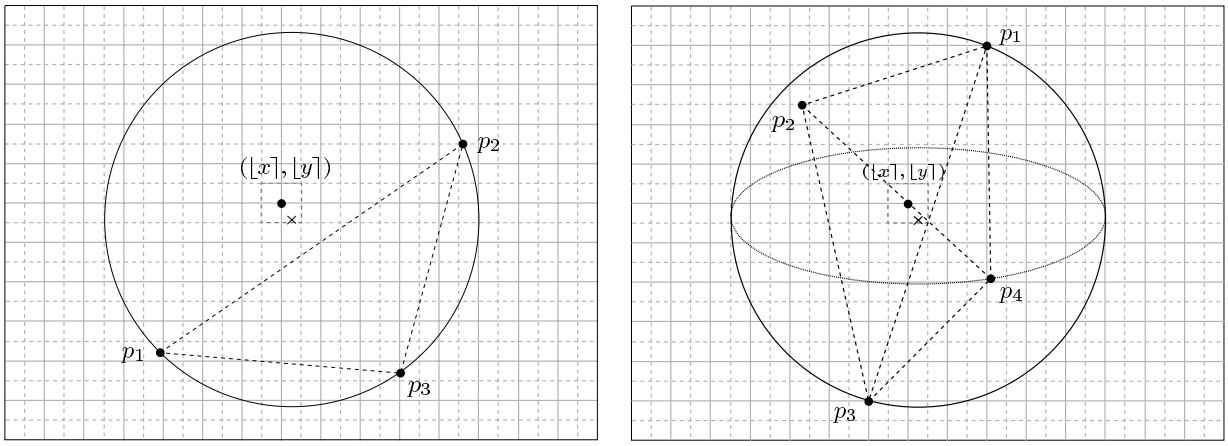

FIG. 3.4 – Le constructeur 3 (à gauche) construit le plus proche point entier du centre du cercle circonscrit au triangle $p_1p_2p_3$. Le constructeur 4 (à droite) construit le plus proche point entier du centre de la sphère circonscrite au tétraèdre $p_1p_2p_3p_4$.

Formulation Étant donnés trois points bidimensionnels p_1, p_2, et p_3 non alignés, les coordonnées du centre du cercle circonscrit au triangle $p_1p_2p_3$ peuvent s'écrire de la manière suivante :

$$\left(x_1 - \frac{\begin{vmatrix} y_{21} & ||p_{21}||^2 \\ y_{31} & ||p_{31}||^2 \end{vmatrix}}{2\begin{vmatrix} x_{21} & y_{21} \\ x_{31} & y_{31} \end{vmatrix}}, y_1 + \frac{\begin{vmatrix} x_{21} & ||p_{21}||^2 \\ x_{31} & ||p_{31}||^2 \end{vmatrix}}{2\begin{vmatrix} x_{21} & y_{21} \\ x_{31} & y_{31} \end{vmatrix}} \right) \tag{3.4}$$

où x_{ij} (respectivement y_{ij}) désigne la différence $x_i - x_j$ (respectivement $y_i - y_j$) et où $||p_{ij}||$ désigne la norme euclidienne de $p_i - p_j$. On note que la condition de non alignement est vérifiée dès lors que le dénominateur D de l'expression (3.4) est différent de zéro.

Calcul d'une borne sur l'erreur d'approximation Le dénominateur D apparaissant dans l'expression des coordonnées du centre du cercle est le résultat du calcul d'un déterminant 2×2 dont les coefficients sont des entiers représentables sur $24 + 1$ bits et ne nécessite par conséquent que 51 bits pour être calculé exactement. Les numérateurs de chacune des coordonnées du centre sont quant à eux des déterminants 2×2 dont les coefficients de la première et de la deuxième colonne sont respectivement des entiers nécessitant au plus 25 et 51 bits pour être calculés et représentés exactement.

En posant $\mathrm{N} = y_{21}||p_{31}||^2 - y_{31}||p_{21}||^2$, $\widetilde{\mathrm{N}} = y_{21} \otimes ||p_{31}||^2 \ominus y_{31} \otimes ||p_{21}||^2$ et $\overline{\mathrm{N}} = |y_{21}| \otimes ||p_{31}||^2 \oplus$

$|y_{31}| \otimes ||p_{21}||^2$, on obtient ainsi

$$|\widetilde{\mathrm{N}} - \mathrm{N}| \leq (2\mathbf{u} + \mathbf{u}^2)\overline{\mathrm{N}}.$$

Une borne supérieure sur l'erreur d'approximation commise sur le calcul de l'abscisse du centre du cercle circonscrit est alors obtenue de la manière suivante :

$$\begin{aligned}\left|\frac{\mathrm{N}}{\mathrm{D}} - (\widetilde{\mathrm{N}} \oslash \mathrm{D})\right| &\leq (2\mathbf{u} + \mathbf{u}^2)\frac{\overline{\mathrm{N}}}{|\mathrm{D}|} + \mathbf{u}|\widetilde{\mathrm{N}} \oslash \mathrm{D}| \\ &\leq (1+\mathbf{u})(2\mathbf{u} + \mathbf{u}^2)|\overline{\mathrm{N}} \oslash \mathrm{D}| + \mathbf{u}|\widetilde{\mathrm{N}} \oslash \mathrm{D}| \\ &\leq (3\mathbf{u} + 3\mathbf{u}^2 + \mathbf{u}^3)|\overline{\mathrm{N}} \oslash \mathrm{D}|\end{aligned}$$

Par conséquent, le produit $4\mathbf{u} \otimes |\overline{\mathrm{N}} \oslash \mathrm{D}|$ est une borne stricte sur l'erreur d'approximation du quotient $\frac{\mathrm{N}}{\mathrm{D}}$. Le plus proche nombre entier du quotient $\widetilde{x} = x_1 \ominus (\widetilde{\mathrm{N}} \oslash \mathrm{D})$ est donc potentiellement différent du plus proche nombre entier du quotient exact $x = x_1 - \frac{\mathrm{N}}{\mathrm{D}}$ lorsque $\lceil \widetilde{x} - 4\mathbf{u} \otimes |\overline{\mathrm{N}} \oslash \mathrm{D}| \rfloor \neq \lceil \widetilde{x} + 4\mathbf{u} \otimes |\overline{\mathrm{N}} \oslash \widetilde{\mathrm{D}}| \rfloor$.

Une solution possible au calcul du constructeur 3 consiste alors à calculer l'expression $\widetilde{x}$ à l'aide de l'arithmétique flottante double précision et à vérifier ensuite le prédicat

$$\frac{1}{2} \ominus \left|\widetilde{x} \ominus \lfloor x \rceil\right| > 4\mathbf{u} \otimes |\overline{\mathrm{N}} \oslash \mathrm{D}|.$$

Comme pour la primitive précédente, si le prédicat est vérifié, l'arithmétique double précision est suffisante pour garantir la validité du résultat et l'algorithme retourne l'entier $x_1 \ominus \lfloor \widetilde{\mathrm{N}} \oslash \widetilde{\mathrm{D}} \rceil$. Dans le cas contraire, l'algorithme procède à un second filtrage basé sur les bornes de séparation de l'expression. On obtient dans ce cas l'inégalité

$$\frac{1}{2|\mathrm{D}|} > 8\mathbf{u} \otimes |\overline{\mathrm{N}} \oslash \mathrm{D}|$$

que l'on peut exprimer sous la forme

$$\overline{\mathrm{N}} < (16\mathbf{u})^{-1} = 2^{49}.$$

Si ce second filtre échoue également, une arithmétique plus précise est finalement utilisée afin de certifier le résultat (cf. Algorithme 3).

3.5 Centre de la sphère circonscrite à un tétraèdre

Constructeur 4 : *Étant donnés quatre points entiers dans l'espace tridimensionnel* $p_1 = (x_1, y_1, z_1)$, $p_2 = (x_2, y_2, z_2)$, $p_3 = (x_3, y_3, z_3)$, *et* $p_4 = (x_4, y_4, z_4)$, *calculer le plus proche point du centre de la sphère circonscrite à ces quatre points (cf. figure 3.4).*

```
circumcenter2d( P1, P2, P3 ) {

X21 = X2 ⊖ X1 ; Y21 = Y2 ⊖ Y1 ;
X31 = X3 ⊖ X1 ; Y31 = Y3 ⊖ Y1 ;
D   = X21 ⊗ Y31 ⊖ X31 ⊗ Y21 ;

si ( D == 0 ) alors /* Les trois sommets sont colinéaires */
sinon  N1 = Y21 ⊗ (X31 ⊗ X31 ⊕ Y31 ⊗ Y31) ;
       N2 = Y31 ⊗ (X21 ⊗ X21 ⊕ Y21 ⊗ Y21) ;
       C = (N1 ⊖ N2) ⊘ (2 ⊗ D) ;
       N̄ = |N1| ⊕ |N2| ;  /* 1 addition, 2 valeurs absolues */
       C̃ = round_to_int(C) ;  /* 2 additions */

       si ( 0.5 ⊖ |C ⊖ C̃| > 4u ⊗ |N̄ ⊘ D| )
       /* 1 division, 2 valeurs absolues, 2 additions, 1 multiplication, 1 comparaison */
       alors retourner (X1 ⊖ C̃) ;        /* 1 addition */

       si ( N̄ < 2^49 ) alors retourner round_to_int(X1 ⊖ 0.5 ⊗ round_to_int(2 ⊗ C)) ;
       /* 5 additions, 2 multiplications, 1 comparaison */

       /* Calcul du signe exact de E = ((N1 − N2)/D − round_to_int(2C)) */
       retourner round_to_int(X1 ⊖ 0.5 ⊗ (round_to_int(2 ⊗ C) ⊕ sign(E))) ; }
```

Algorithme 3: Constructeur 3 (calcul de l'abscisse uniquement).

Formulation Étant donnés quatre points tridimensionnels p_1, p_2, p_3, et p_4 non coplanaires et en utilisant les mêmes notations que la section précédente, le centre de la sphère circonscrite au tétraèdre $p_1p_2p_3p_4$ peut s'écrire de la manière suivante :

$$p_1 + \frac{||p_{41}||^2(p_{21} \times p_{31}) + ||p_{31}||^2(p_{41} \times p_{21}) + ||p_{21}||^2(p_{31} \times p_{41})}{2 \begin{vmatrix} x_{21} & y_{21} & z_{21} \\ x_{31} & y_{31} & z_{31} \\ x_{41} & y_{41} & z_{41} \end{vmatrix}} \tag{3.5}$$

Remarque On note que les formules utilisées pour exprimer les constructeurs 1 à 4 dépendent essentiellement de différences entre les valeurs des coordonnées des sommets en entrée. Les erreurs d'approximation intervenant dans le calcul des coordonnées de ces constructions sont

par conséquent uniquement proportionnelles à ces différences et totalement indépendantes de la valeur absolue des coordonnées jusqu'à l'addition finale. Cette propriété est souvent souhaitable en pratique où les sommets sont généralement plus proches entre eux que de l'origine [21].

Erreur d'approximation Pour cette primitive particulière le calcul d'une borne supérieure précise sur l'erreur d'approximation commise par l'arithmétique flottante double précision s'avère plus délicat que pour les primitives précédentes. L'erreur d'approximation commise lors du calcul du dénominateur D de l'expression n'est désormais plus nécessairement proportionnelle à la valeur absolue de l'approximation $\widetilde{\mathrm{D}}$ calculée, et le calcul d'une borne inférieure sur $\left|\widetilde{\mathrm{D}} - \left|\mathrm{D} - \widetilde{\mathrm{D}}\right|\right|$ nécessaire au calcul de la propagation d'erreur de l'opération de division n'est plus possible.

Par conséquent, la solution proposée consiste à calculer le plus proche nombre flottant double précision de D à l'aide d'expansions flottantes (le calcul nécessitant 24 opérations élémentaires au lieu de 6). Ce calcul est nécessaire à la stabilité du constructeur puisque celui-ci assure que la condition de non coplanarité est vérifiée ($\mathrm{D} \neq 0$) si et seulement si $\widetilde{\mathrm{D}} \neq 0$.

Le numérateur de l'expression est une somme de trois termes de degré quatre. Une borne d'erreur sur l'erreur commise lors de son évaluation approchée à l'aide de l'arithmétique flottante double précision est obtenue en sommant les erreurs d'arrondi maximales commises lors des trois multiplications et des deux additions. D'où, avec N le dénominateur exact et $\widetilde{\mathrm{N}}$ son approximation à l'aide de l'arithmétique double précision

$$|\widetilde{\mathrm{N}} - \mathrm{N}| \leq (3\mathbf{u} + 2\mathbf{u}^2)\left(\left(\left|||p_{41}||^2 \otimes q_{23}\right| \oplus \left|||p_{31}||^2 \otimes q_{42}\right|\right) \oplus \left|||p_{21}||^2 \otimes q_{34}\right|\right).$$

Avec $\overline{\mathrm{N}} = (||p_{41}||^2 \otimes |q_{23}| \oplus ||p_{31}||^2 \otimes |q_{42}|) \oplus (||p_{21}||^2 \otimes |q_{34}|)$ et $\epsilon_{\mathrm{N}} = (3\mathbf{u}+2\mathbf{u}^2)\overline{\mathrm{N}}$, une borne supérieure sur l'erreur d'approximation commise lors de l'évaluation approchée $\widetilde{x}$ de l'abscisse x du centre de la sphère circonscrite au tétraèdre en entrée est alors donnée par le calcul suivant :

$$\begin{aligned}
|\widetilde{x} - x| &\leq \frac{\epsilon_{\mathrm{N}}|\widetilde{\mathrm{D}}| + \mathbf{u}\widetilde{\mathrm{D}}|\widetilde{\mathrm{N}}|}{|\widetilde{\mathrm{D}}|(|\widetilde{\mathrm{D}}| - \mathbf{u}\widetilde{\mathrm{D}})} + \mathbf{u}|\widetilde{\mathrm{N}} \oslash \widetilde{\mathrm{D}}| \\
&\leq \frac{(3\mathbf{u} + 2\mathbf{u}^2)\overline{\mathrm{N}}}{(1-\mathbf{u})|\widetilde{\mathrm{D}}|} + \frac{\mathbf{u}|\widetilde{\mathrm{N}}|}{(1-\mathbf{u})|\widetilde{\mathrm{D}}|} + \mathbf{u}|\widetilde{\mathrm{N}} \oslash \widetilde{\mathrm{D}}| \\
&\leq \frac{\mathbf{u}}{1-\mathbf{u}}(5 + 5\mathbf{u} + 2\mathbf{u}^2)|\overline{\mathrm{N}} \oslash \widetilde{\mathrm{D}}| \\
&< (5\mathbf{u} + 10\mathbf{u}^2 + 7\mathbf{u}^3 + 2\mathbf{u}^4)|\overline{\mathrm{N}} \oslash \widetilde{\mathrm{D}}| < 8\mathbf{u}|\overline{\mathrm{N}} \oslash \widetilde{\mathrm{D}}|
\end{aligned}$$

Le constructeur filtré calcule donc dans un premier temps $\widetilde{x} = \widetilde{\mathrm{N}} \oslash \widetilde{\mathrm{D}}$ et teste ensuite le prédicat

$$\frac{1}{2} \ominus \left| \widetilde{x} \ominus \lfloor x \rceil \right| > 8\mathbf{u} \otimes |\overline{\mathrm{N}} \oslash \widetilde{\mathrm{D}}|.$$

Là encore, si le prédicat est vérifié l'entier $x_1 + \lfloor \widetilde{x} \rceil$ est l'entier recherché, sinon un calcul plus exact est mené afin de décider lequel des deux entiers $(x_1 + \lfloor \widetilde{x} \rfloor)$ et $(x_1 + \lceil \widetilde{x} \rceil)$ correspond au plus proche nombre entier de l'abscisse exacte du centre de la sphère (cf. Algorithme 4).

Remarque : Contrairement au cas du calcul du point d'intersection de deux segments (constructeurs 1 et 2), les constructeurs 3 et 4 peuvent produire un résultat dont les coordonnées sont en valeurs absolues supérieures à 2^{53} et qui ne peuvent être représentées exactement par un nombre double précision. Bien que les bornes d'erreurs obtenues pour ces constructeurs restent valides pour de telles instances en entrée, la représentation exacte du résultat nécessite dans ce cas l'utilisation d'outils arithmétiques adaptés.

Performances : L'approche proposée utilisant un calcul approché et une certification du résultat s'avère en moyenne de 1,5 à 2 fois plus lente qu'un calcul flottant double précision sans précaution et plus de 2 fois plus rapide qu'un calcul entier simple et double précision (entiers sur 32 et 64 bits) pour l'ensemble des constructeurs présentés (cf. [15]). Le gain en temps comparé à une implémentation utilisant une arithmétique rationnelle exacte du type *leda_rational* est de l'ordre d'un facteur 100.

Concernant la robustesse des différentes implémentations, les constructeurs basés sur une arithmétique entière garantissent un résultat conforme aux spécifications d'arrondi mais nécessitent une restriction significative du domaine de définition des données afin d'éviter les problèmes de débordements numériques. L'implémentation flottante double précision n'est pas confrontée à ce type de problème mais n'offre en contrepartie aucune garantie sur le résultat calculé. Lors du calcul d'un diagramme de Voronoï tridimensionnel de 10000 sommets entiers (uniformément distribués dans un cube de côté 2^{24} centré à l'origine) par exemple au moins un centre de sphère est en moyenne incorrect[4] et peut du fait de l'instabilité du calcul se trouver arbitrairement loin du centre exact. Les constructeurs filtrés arrondis proposés constituent une solution à ces problèmes.

[4] Le nombre de constructions erronées peut s'avérer être beaucoup plus important pour des données réelles.

```
circumcenter3d( P1, P2, P3, P4 ){

XCROSS23 = Y21 ⊗ Z31 ⊖ Z21 ⊗ Y31;
XCROSS42 = Y41 ⊗ Z21 ⊖ Z41 ⊗ Y21;
XCROSS34 = Y31 ⊗ Z41 ⊖ Z31 ⊗ Y41;

split(XCROSS23, H1, L1);
split(XCROSS34, H2, L2);
split(XCROSS42, H3, L3);
Dh = X21 ⊗ H2 ⊕ X31 ⊗ H3 ⊕ X41 ⊗ H1;
Dl = X21 ⊗ L2 ⊕ X31 ⊗ L3 ⊕ X41 ⊗ L1;
D  = 2 ⊗ (Dl ⊕ Dh);

si ( D == 0 ) alors      /* Les quatre sommets sont coplanaires. */
sinon N1 = (X41 ⊗ X41 ⊕ Y41 ⊗ Y41 ⊕ Z41 ⊗ Z41) ⊗ XCROSS23 ;
      N2 = (X31 ⊗ X31 ⊕ Y31 ⊗ Y31 ⊕ Z31 ⊗ Z31) ⊗ XCROSS42 ;
      N3 = (X21 ⊗ X21 ⊕ Y21 ⊗ Y21 ⊕ Z21 ⊗ Z21) ⊗ XCROSS34 ;
      N = N1 ⊕ N2 ⊕ N3 ;
      C = N ⊘ D ;
      C̃ = rount_to_int(C);
      N̄ = |N1| ⊕ |N2| ⊕ |N3|.

      si ( 0.5 ⊖ |C ⊖ C̃| > 8u ⊗ |N̄ ⊘ D| )
      /* 1 division, 2 valeurs absolues, 2 additions, 1 multiplication */
      alors retourner ( X1 ⊕ C̃)   /* 1 addition */

      /* Calcul de la valeur exacte de C */
      retourner round_to_int(X1 ⊕ C)
}
```

Algorithme 4: Constructeur 4 (calcul de l'abscisse uniquement).

3.6 Plus proche point entier du sommet d'un secteur convexe

Constructeur 5 : *Étant donnés deux segments du plan s'intersectant en un point unique, calculer le plus proche point entier du point d'intersection sous la contrainte que celui-ci se trouve dans un secteur donné parmi ceux formés par les droites support aux deux segments (cf. figure 3.7).*

Le secteur solution peut contenir zéro, une ou deux des quatre directions x^+, x^-, y^+ et y^- définies par les axes de coordonnées. La méthode d'arrondi proposée reprend la solution développée par Lee et Chang [47] concernant l'approximation entière du sommet d'un secteur ouvert et complète celle-ci en traitant le cas (omis dans l'article original) où exactement deux directions sont contenues dans le secteur. On distingue donc trois situations distinctes, le cas le plus simple se produit lorsque seulement une de ces quatre directions se dirige entre les directions des deux arêtes définissant le secteur.

Sans perdre en généralité, supposons qu'il s'agisse de la direction y^+ (cf. figure 3.5.a). Dans ce cas, une solution peut facilement être calculée. Soit $v = (v_x, v_y)$ le sommet convexe à arrondir. Le point de la grille le plus proche de v à l'intérieur du secteur défini par les arêtes incidentes à v se trouve soit sur la droite $L_1 : x = \lfloor v_x \rfloor$ soit sur la droite $L_2 : x = \lceil v_x \rceil$. De plus, ce point a une ordonnée minimale parmi les points de la grille à l'intérieur de ce secteur. On pose $(\lfloor v_x \rfloor, y_1)$ (respectivement $(\lceil v_x \rceil, y_2)$) l'intersection entre L_1 (respectivement L_2) et les arêtes incidentes au sommet v, $p_1 = (\lfloor v_x \rfloor, \lceil y_1 \rceil)$ et $p_2 = (\lceil v_x \rceil, \lceil y_2 \rceil)$. Étant donné que le secteur atteint son point d'ordonnée minimale en v, le point de la grille le plus proche de v dans le secteur est soit p_1 soit p_2[5].

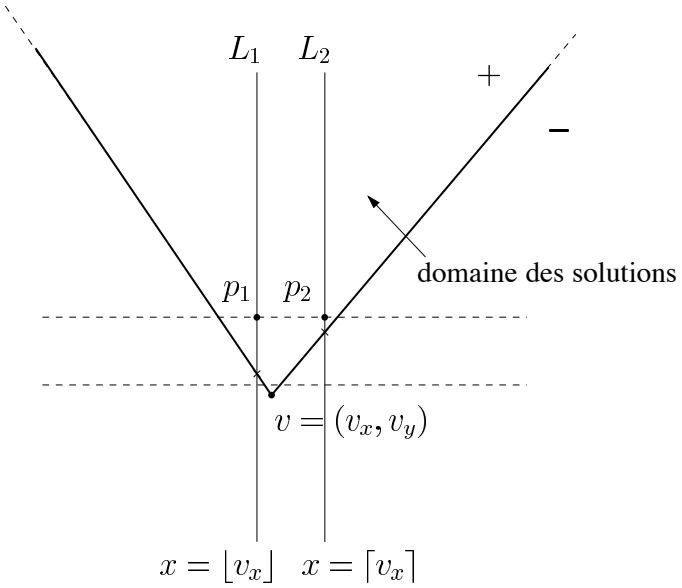

FIG. 3.5 – Arrondi local d'un sommet convexe, cas simple. Les droites verticales de la grille se dirigent à l'intérieur du secteur.

[5]Le lecteur est renvoyé à l'article original [47] pour les preuves complètes.

L'heuristique utilisée est alors très simple : si p_1 et p_2 ont des ordonnées différentes alors le sommet original est arrondi au point d'ordonnée minimale. Dans la cas contraire, c'est-à-dire si les deux points ont des ordonnées identiques, le sommet original est arrondi au point ayant l'abscisse la plus proche du sommet v (p_1 dans l'exemple de la figure 3.5).

Le deuxième cas se produit lorsque deux directions (et non plus une seule) se dirigent dans le domaine des solutions. Sans perdre en généralité, supposons que ces directions soient y^+ et x^+ (les autres cas pouvant se ramener facilement à celui-ci par une opération de symétrie par rapport à l'origine ou aux axes) et prenons $(\lfloor v_x \rfloor, y_1)$ et $(\lceil v_x \rceil, y_2)$ les deux points correspondants respectivement à l'intersection des droites L_1 et L_2 avec les arêtes incidentes au sommet. Comme précédemment, le point entier recherché appartient soit à L_1 soit à L_2. Cependant, celui-ci n'est plus nécessairement le point d'ordonnée minimale parmi les points de la grille appartenant à l'intérieur du secteur (cf. le point $(\lceil v_x \rceil, \lceil y_2 \rceil)$ de la figure 3.6.a).

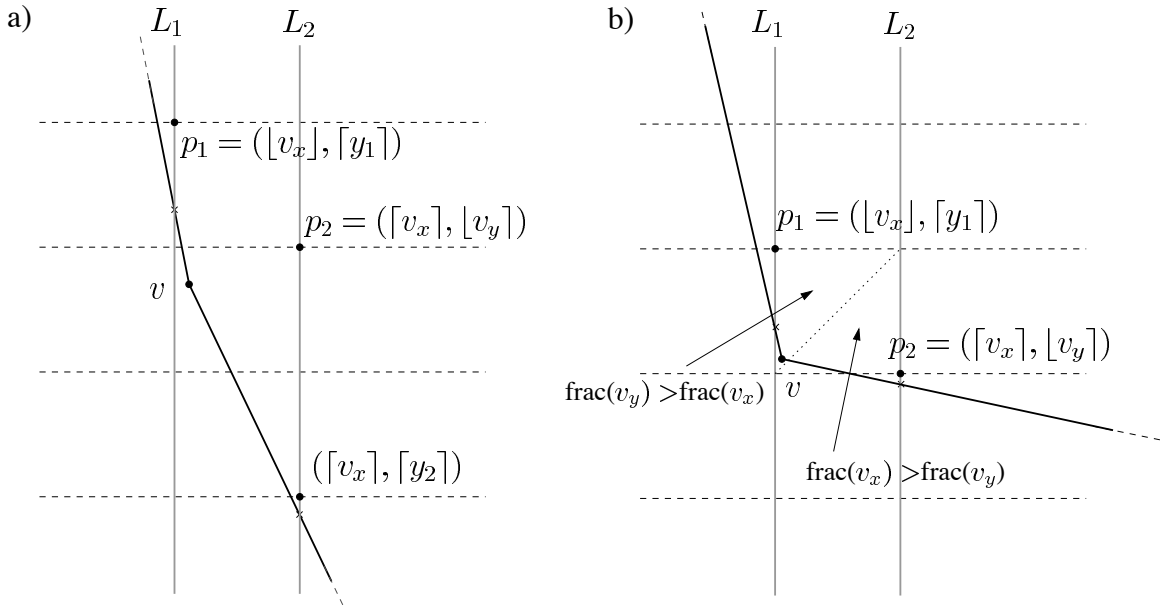

FIG. 3.6 – Arrondi local d'un sommet convexe, cas où le secteur contient deux axes. À gauche, les ordonnées de p_1 et p_2 sont supérieures à $\lceil v_y \rceil$, la solution est le point d'ordonnée minimale. À droite, le cas où $p_{1y} = \lceil v_y \rceil$ et $p_{2y} < \lceil v_y \rceil$, la solution dépend ici de la partie fractionnaire des coordonnées de v et est égale à p_1 si frac(v_y) >frac(v_x) et à p_2 si frac(v_x) >frac(v_y).

La solution recherchée peut toutefois être obtenue simplement de la manière suivante. Soient $p_1 = (\lfloor v_x \rfloor, \max(\lceil y_1 \rceil, \lfloor v_y \rfloor))$ et $p_2 = (\lceil v_x \rceil, \max(\lceil y_2 \rceil, \lfloor v_y \rfloor))$, si $p_{1y} \neq \lceil v_y \rceil$ ou $p_{2y} \geq \lceil v_y \rceil$ (cf. figure 3.6.a), la solution recherchée correspond au point d'ordonnée minimale (ou au point

d'abscisse la plus proche du sommet à arrondir si les points p_1 et p_2 ont même ordonnée). Dans le cas contraire, c'est-à-dire si $p_{1y} = \lceil v_y \rceil$ et $p_{2y} < \lceil v_y \rceil$ (cf. figure 3.6.b), la solution recherchée dépend de la partie fractionnaire des coordonnées de v. Si $\text{frac}(v_y) > \text{frac}(v_x)$ le sommet v est plus proche de p_1 que de p_2. Si $\text{frac}(v_x) > \text{frac}(v_y)$ le sommet v est plus proche de p_2 que de p_1. Enfin si $\text{frac}(v_x) = \text{frac}(v_y)$ le sommet v est équidistant de p_1 et de p_2.

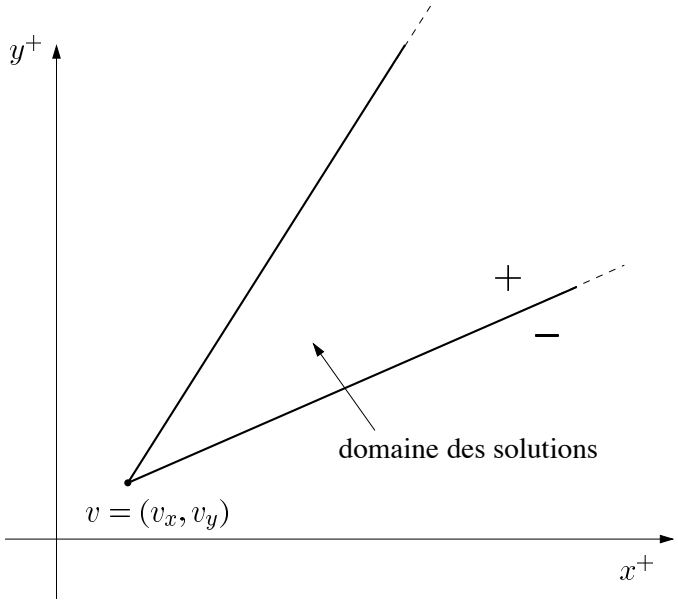

FIG. 3.7 – Le cas difficile : aucune des quatre directions ne se dirige dans le secteur.

Le cas plus difficile pour l'arrondi local d'un sommet convexe se produit lorsqu'aucune des directions x^+, x^-, y^+, y^- ne se dirige dans le secteur défini par les arêtes incidentes à v (cf. figure 3.7). Sans perdre en généralité, supposons maintenant que ce secteur se situe dans le premier quadrant (les autres cas pouvant se ramener facilement à celui-ci par une opération de symétrie par rapport à l'origine ou aux axes). L'idée est d'introduire une transformation linéaire projetant les points de la grille sur des points à coordonnées entières afin de ramener le secteur au cas simple, c'est-à-dire au cas où y^+ se dirige dans le secteur.

Pour cela, nous introduisons le principe de fractions continues issu de la théorie des nombres et utilisé pour calculer des approximations rationnelles d'un nombre réel.

Étant donné un nombre réel positif r, il existe une unique représentation de r, appelée déve-

loppement en fraction continue, de la forme :

$$r = q_0 + \cfrac{1}{q_1 + \cfrac{1}{q_2 + \cfrac{1}{q_3 + \dots}}}$$

où $q_0 = \lfloor r \rfloor$ et $q_1, q_2, q_3, \dots$ sont des entiers strictement positifs.

Les nombres $q_1, q_2, q_3, \dots$ sont définis par la relation de récurrence

$$x_0 = r, \quad q_0 = \lfloor r \rfloor, \quad x_{i+1} = \frac{1}{x_i - q_i}, \quad q_{i+1} = \lfloor x_{i+1} \rfloor.$$

On note que si r est un rationnel, c'est-à-dire si $r = \frac{a}{b}$ où a et b sont des entiers, le développement en fraction continue de r est fini. De plus, dans ce cas, les termes q_i sont un sous-produit du calcul du plus grand diviseur commun de a et de b et peuvent ainsi être calculés en temps $O(\log(\min(a, b)))$.

On appelle enfin développement partiel ou *réduite de rang* i ($i = 0, \dots, n$) la fraction :

$$\frac{a_i}{b_i} = q_0 + \cfrac{1}{q_1 + \cfrac{1}{\dots + \cfrac{1}{q_i}}}.$$

Parmi les nombreuses propriétés vérifiées par le développement en fraction continue [50], on note que les nombres a_i et b_i satisfont la récurrence

$$a_{i+1} = q_{i+1} a_i + a_{i-1}, \quad b_{i+1} = q_{i+1} b_i + b_{i-1}$$

et vérifient l'identité

$$a_{i+1} b_i - a_i b_{i+1} = (-1)^i \tag{3.6}$$

Si l'on représente, de plus, les réduites $\frac{a_i}{b_i}$ de r, pour $i = 0, \dots, n$, sous forme vectorielle $v_i = (b_i, a_i)$, on obtient la récurrence

$$v_i = q_i v_{i-1} + v_{i-2} \text{ où } v_{-2} = (1, 0) \text{ et } v_{-1} = (0, 1).$$

Nous appellerons $v_0, \dots, v_n$ les réduites (géométriques) du point $v = (b, a)$ correspondant à r. Par l'équation (3.6), deux réduites géométriques consécutives v_i et v_{i+1} constituent une base

de la grille entière $\mathbb{Z}^2$; autrement dit, l'ensemble $\{c_1 v_i + c_2 v_{i+1} | c_1, c_2 \in \mathbb{Z}\}$ est égal à la grille entière $\mathbb{Z}^2$.

Si $r > 0$ est irrationnel, les réduites impaires $v_{-1}, v_1, \ldots$ sont les sommets de l'enveloppe convexe discrète de la région s'étendant au dessus de la droite l dans le quadrant positif du plan, où l est la droite d'équation $y = rx$ (cf. figure 3.8). De la même manière, les réduites paires $v_{-2}, v_0, \ldots$ sont les sommets de l'enveloppe convexe discrète de la région s'étendant au dessous de l dans le quadrant positif du plan. Ces deux chaînes polygonales se situent dans les demi-plans définis par l, et intersectent l seulement si r est rationnel.

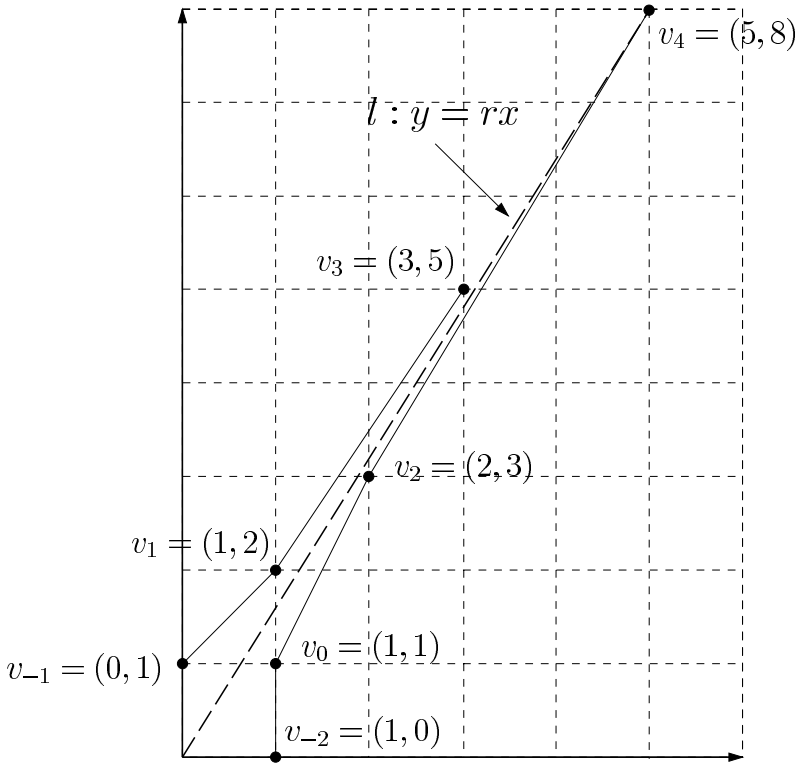

FIG. 3.8 – Chaînes de réduites paires et impaires générées pour $r = \frac{8}{5}$.

La connexion entre fraction continue et enveloppe discrète d'un demi-plan a été découverte en 1896 par Klein [46]. Beaucoup plus récemment, les développements en fractions continues ont été utilisés pour le calcul de chaînes polygonales constituant l'enveloppe discrète de formes géométriques spécifiques [35, 47, 40].

Revenons maintenant à notre problème initial : soit α un nombre réel compris entre les pentes des deux arêtes définissant le secteur et soit $v_i = (a_i\ ,\ b_i)$ les réduites du nombre réel α. Puisque

l'intervalle borné par une paire de réduites consécutives est contenu dans l'intervalle borné par une paire de réduites consécutives possédant de plus petits indices, il existe un entier i pour lequel v_i se dirige dans le secteur mais pas v_{i-2} ni v_{i-1}.

Dans le cas où aucune des quatre directions x^+, x^-, y^+, y^- ne se dirige dans le secteur défini par les arêtes incidentes à v, l'algorithme d'arrondi local du sommet convexe est alors le suivant :

- Application (si nécessaire) d'une translation entière et d'une opération de symétrie afin de ramener le secteur dans le premier quadrant.
- Choix d'un α tel que le vecteur $(1\ ,\ \alpha)$ se dirige dans le secteur.
- Calcul du premier entier i tel que v_i se dirige dans le secteur.
- Calcul du point le plus à gauche parmi les points entiers d'ordonnée minimale à l'intérieur du secteur dans le nouveau système de coordonnées utilisant v_{i-1} comme vecteur unitaire de l'axe des x et v_i comme vecteur unitaire de l'axe des y.

Si les pentes des arêtes incidentes à v sont des rationnels dont le dénominateurs sont inférieurs à d, alors le temps nécessaire au calcul de v_i est borné par $O(\log d)$. Si la grille entière utilisée est de taille $N \times N$, alors le temps de calcul de v_i est en $O(\log N)$ étant donné qu'il existe au plus $O(\log N)$ réduites géométriques ayant un dénominateur inférieur à N. De plus, il est prouvé [47] que le choix de α n'influe pas sur le temps de calcul de i à condition que $(1\ ,\ \alpha)$ se dirige dans le secteur.

3.7 Plus proche point entier visible depuis un sommet d'un polygone convexe

Étant donné un polygone convexe P tel que chaque arête de P admet un segment à coordonnées entières comme segment support. Nous proposons dans cette section une nouvelle méthode permettant de calculer le plus proche point entier d'un sommet v de P sous la contrainte que ce point se trouve à l'intérieur ou sur la frontière de P. L'algorithme opère en temps $O(n \log n \log N)$ où n désigne le nombre de sommets de P et $N \times N$ est la taille de la grille entière contenant l'ensemble des segments à coordonnées entières supports aux arêtes de P.

La section est organisée de la manière suivante : nous introduisons dans un premier temps le concept d'enveloppe discrète d'un segment ainsi que quelques lemmes nécessaires à la preuve de l'algorithme, nous donnons ensuite une description détaillée de l'algorithme et analysons sa complexité et son degré algébrique. Enfin, nous comparons ses performances avec l'algorithme développé par Lee et Chang [47].

On suppose dans la suite le polygone P orienté dans le sens direct. On note respectivement ab et cd les segments (à coordonnées entières) supports aux arêtes entrante et sortante en v et S le secteur convexe obtenu en prolongeant les arêtes ab et cd de P incidentes à v. On note de plus y_1 (resp. y_2) l'ordonnée du point d'intersection de la droite verticale $L_1 : x = \lfloor v_x \rfloor$ (resp. de la droite $L_2 : x = \lceil v_x \rceil$) avec l'arête ab (resp. l'arête cd) et p_1 (resp. p_2) le point entier de coordonnées $(\lfloor v_x \rfloor, \lceil y_1 \rceil)$ (resp. de coordonnées $(\lceil v_x \rceil, \lceil y_2 \rceil)$) (cf. figure 3.5).

Lemme 3.7.1 *Si l'axe y^+ (et uniquement cet axe) se dirige dans le secteur convexe S et si au moins l'un des deux points p_1 et p_2 appartient à P, alors le plus proche point entier de v appartenant à P est soit p_1 soit p_2.*

Démonstration: Si l'axe y^+ est le seul axe se dirigeant dans le secteur S, le plus proche point entier de v dans S correspond au point entier d'ordonnée minimale parmi p_1 et p_2. Supposons, sans perdre en généralité, que p_1 soit ce point. Si p_1 appartient à P, p_1 est clairement le plus proche point entier de v appartenant à P, dans le cas contraire, par convexité de P, il ne peut exister de point entier appartenant à P d'ordonnée plus petite que celle de p_2. De plus, si p_2 appartient à P ce point est le point entier de P ayant l'abscisse la plus proche de celle de v, par conséquent, p_2 est forcément le plus proche point entier de v dans P. □

Le lemme suivant permet de traiter le cas où p_1 et p_2 n'appartiennent pas à P. L'idée consiste à construire un sous-polygone convexe Q à partir de P tel que pour un sommet w donné de Q le plus proche point entier de w appartenant à Q est aussi le plus proche point entier de v appartenant à P. Avant d'énoncer le lemme proprement dit, nous introduisons la technique de perturbation d'un segment entier introduite par Greene et Yao [35].

Étant donné un segment à coordonnées entières ab et un point entier p tel que p est l'extrémité d'un segment isothétique unitaire intersectant en son intérieur le segment ab, l'enveloppe discrète de ab relativement à p, notée $\mathcal{D}(ab, p)$, est définie comme la plus courte chaîne polygonale à sommets entiers connectant a à p et p à b de telle sorte qu'il n'existe aucun point entier entre la chaîne $\mathcal{D}(ab, p)$ et le segment ab (cf. figure 3.9).

Étant donnés a, b, et p, l'enveloppe discrète $\mathcal{D}(ab, p)$ peut être calculée en temps $O(\log |ab|)$, où $|ab|$ désigne la longueur du segment ab, à l'aide de l'algorithme proposé par Greene et Yao [35] qui est basé sur les suites de Farey et le concept de développement en fractions continues. Nous renvoyons le lecteur à l'article original pour plus de détails et la description complète de l'algorithme. Une propriété intéressante de $\mathcal{D}(ab, p)$ que nous utiliserons dans la suite est que $\mathcal{D}(ab, p)$ possède la même monotonie en x et en y que le segment original ab.

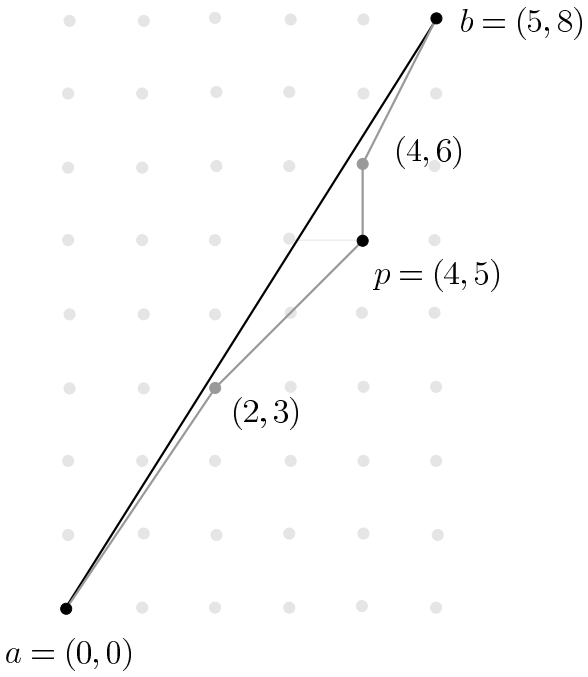

FIG. 3.9 – L'enveloppe discrète $\mathcal{D}(ab, p)$ de ab relativement à p.

On note dans la suite $\mathcal{D}(ab, a \to p)$ et $\mathcal{D}(ab, p \to b)$ les sous-chaînes convexes de $\mathcal{D}(ab, p)$ connectant respectivement a à p et p à b.

Lemme 3.7.2 *Si l'axe y^+ (et uniquement cet axe) se dirige dans le secteur S et si aucun des points p_1 et p_2 n'appartient à P, alors soit P ne contient aucun point entier, soit on peut déduire un sous-polygone convexe Q de P et un sommet w de Q tel que le plus proche point entier de v appartenant à P est aussi le plus proche point entier de w appartenant à Q.*

Démonstration: Soit h le sommet d'ordonnée maximale (et d'abscisse minimale en cas d'ambiguïté) de P. Si h se situe entre les droites L_1 et L_2, par convexité de P et puisque ni p_1 ni p_2 n'appartient à P, P ne peut contenir de point entier. Dans le cas contraire, supposons que h se situe à droite de L_2 (cf. figure 3.10) et posons $\sigma = \mathcal{D}(cd, p_2 \to d)$ (le cas où h se trouve à gauche de L_1 se traite de manière similaire par symétrie avec l'axe des y en prenant $\sigma' = \mathcal{D}(ab, a \to p_1)$). Puisque le polygone convexe P atteint son point d'ordonnée maximale à droite de L_2 et ne contient ni p_1 ni p_2, la partie de P à gauche de L_1 est forcément vide de point entier et l'intersection de σ avec P est soit vide soit une chaîne convexe. Si $P \cap \sigma = \emptyset$, P est compris entre σ et cd et ne peut contenir de point entier. Sinon, soit w l'extrémité de $P \cap \sigma$ la plus proche de p_2, S l'ensemble des points p du plan tels que pour chaque arête $s_i s_{i+1}$ de σ (orientée de p_2 vers d) $[s_i, s_{i+1}, p] \geq 0$

et Q le polygone convexe obtenu en intersectant P avec S. Puisque h se situe à droite de L_2 et comme $p_2 \notin P$, l'arête de P intersectée en w est orientée de gauche à droite et de haut en bas. De même, l'arête de σ intersectée en w a même monotonie en x et en y que le segment cd et est donc orientée de droite à gauche et est horizontale ou croissante en y. Par conséquent w correspond au plus petit sommet de Q selon l'ordre lexicographique, et le plus proche point entier de w dans Q correspond (s'il existe) au point entier d'ordonnée minimale le plus à gauche de Q. Puisque P et Q contiennent exactement les mêmes points entiers, et puisque Q se situe à gauche de L_2 ce point est aussi le plus proche point entier de v appartenant à P. □

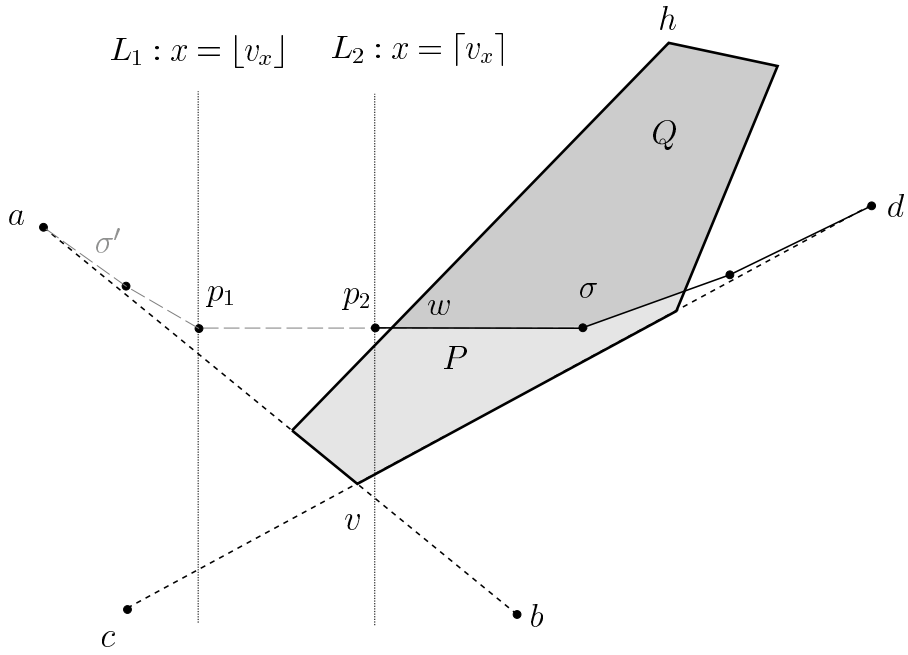

FIG. 3.10 – Le plus proche point entier de w appartenant à Q (en gris foncé) est le plus proche point entier de v appartenant à P.

Lemme 3.7.3 *Si le secteur S se situe dans le premier quadrant et le plus proche point entier de v dans S n'appartient pas à P, alors soit P ne contient aucun point entier, soit on peut déduire un sous-polygone convexe Q de P et un sommet w de Q tel que le plus proche point entier de w appartenant à Q est aussi le plus proche point entier de v appartenant à P.*

Démonstration: Par un argument similaire à celui de la preuve précédente, le plus grand sommet h de P selon l'ordre xy-lexicographique se trouve soit à droite soit à gauche du vecteur $\overrightarrow{vp}$ et au

moins l'une des deux chaînes polygonales $\sigma_1 = \mathcal{D}(ab, a \to p)$ et $\sigma_2 = \mathcal{D}(cd, p \to d)$ n'intersecte pas P (cf. figure 3.11). Si ni σ_1 ni σ_2 n'intersectent P, P ne contient aucun point entier. Sinon supposons que h se trouve à gauche de $\overrightarrow{vp}$ (le cas où h se trouve à droite de $\overrightarrow{vp}$ se traite de manière similaire par symétrie par rapport à la droite support à $\overrightarrow{vp}$) et posons Q le polygone convexe obtenu en remplaçant dans P l'arête supportée par ab par $\sigma_1 \cap P$ et w le point de $\sigma_1 \cap P$ le plus proche de p. L'arête de σ_1 intersectée en w par P a même monotonie que l'arête ab et est donc décroissante en x et en y. De plus, l'arête de P intersectée en w est comprise entre les sommets v et h (on rappelle que l'on suppose P orienté dans le sens direct) et est par conséquent orientée de gauche à droite et de bas en haut. Par conséquent, w correspond au plus petit sommet de Q selon l'ordre lexicographique et le plus proche point entier de w appartenant à Q correspond (s'il existe) au point entier d'ordonnée minimale le plus à gauche de Q. Puisque P et Q contiennent exactement les mêmes points entiers et comme v se situe à gauche de $L : x = p_x$, ce point est aussi le plus proche point entier de v dans P. □

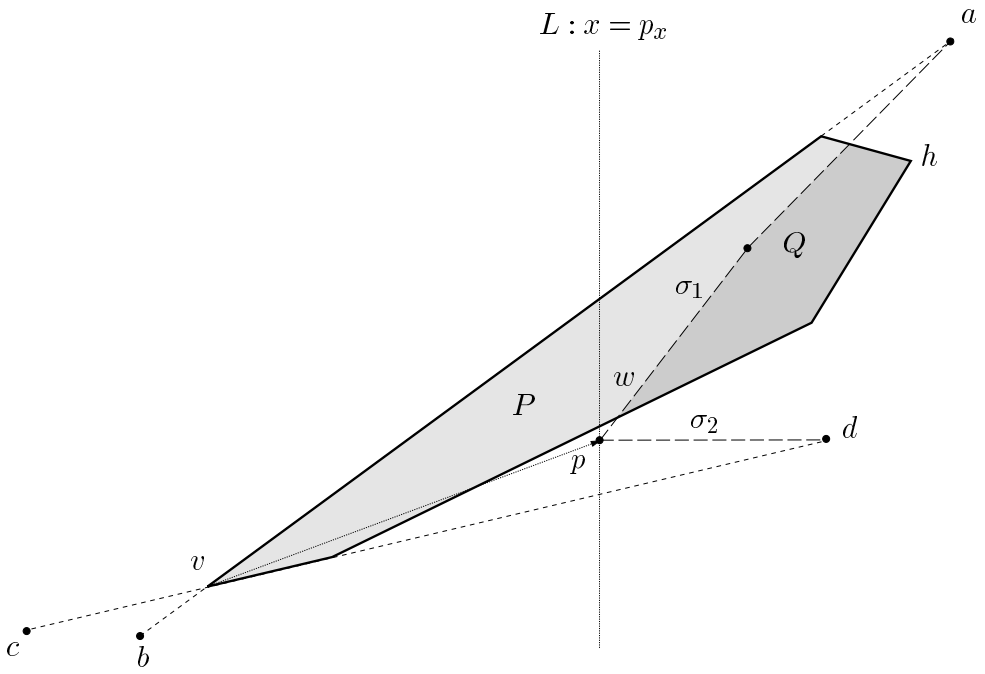

FIG. 3.11 – Cas où S se situe dans le premier quadrant et σ_1 intersecte P.

Le lemme suivant permet de traiter le dernier cas possible à savoir le cas où les axes y^+ et x^+ se dirigent tous les deux à l'intérieur du secteur S.

Avec $p_1 = (\lfloor v_x \rfloor, \max(\lceil y_1 \rceil, \lfloor v_y \rfloor))$ et $p_2 = (\lceil v_x \rceil, \max(\lceil y_2 \rceil, \lfloor v_y \rfloor))$, où y_1 (resp. y_2) désigne

l'ordonnée du point d'intersection entre le segment ab (resp. le segment cd) et la droite L_1 (resp. la droite L_2), on a ainsi :

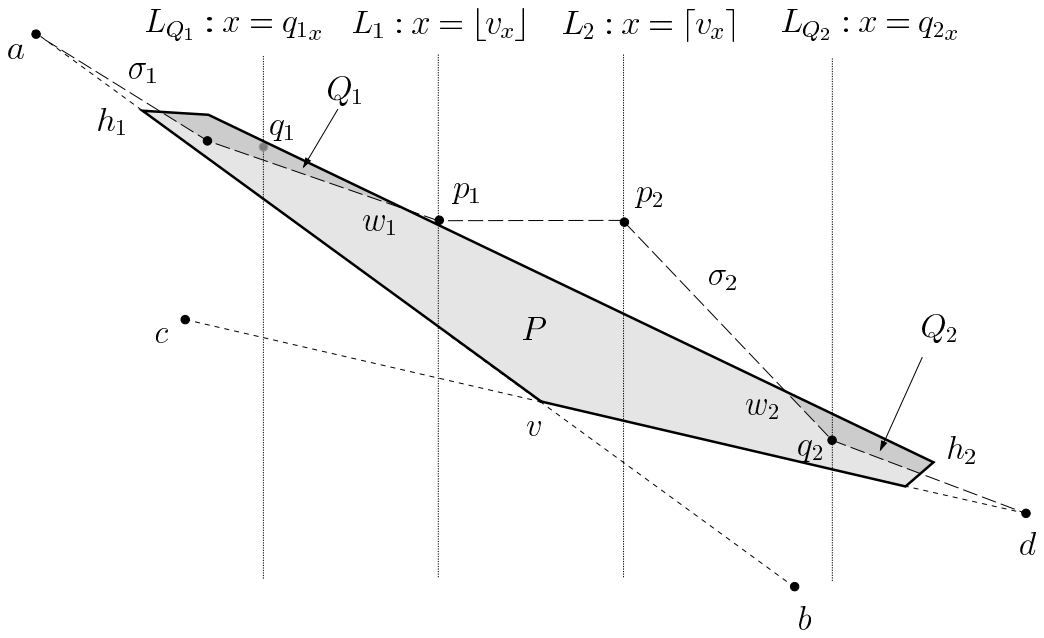

FIG. 3.12 – Cas où le secteur S contient les axes y^+ et x^+ et les deux chaînes σ_1 et σ_2 intersectent P.

Lemme 3.7.4 *Si le secteur S contient les axes y^+ et x^+ et si aucun des points p_1 et p_2 n'appartient à P, alors soit P ne contient aucun point entier, soit on peut déduire au plus deux sous-polygones convexes Q_1 et Q_2 et deux sommets w_1 et w_2 tels que si q_1 (resp. q_2) est le plus proche point entier de w_1 (resp. de w_2) appartenant à Q_1 (resp. à Q_2) alors le plus proche point entier de v appartenant à P se situe soit sur la droite d'équation $L_{Q_1} : x = q_{1x}$ soit sur la droite d'équation $L_{Q_2} : x = q_{2x}$.*

Démonstration: Soit $\sigma_1 = \mathcal{D}(ab, a \to p_1)$ et $\sigma_2 = \mathcal{D}(cd, p_2 \to d)$, si ni σ_1 ni σ_2 n'intersectent P, P ne peut contenir aucun point entier, dans le cas contraire la région polygonale Q, obtenue en remplaçant respectivement dans P les arêtes supportées par ab et cd par les chaînes $\sigma_1 \cap P$ et $\sigma_2 \cap P$, correspond soit à un polygone convexe, soit à deux polygones convexes Q_1 et Q_2 (cf. figure 3.12). Supposons que σ_2 intersecte P et posons w_2 le sommet de Q_2 correspondant à l'extrémité de $\sigma_2 \cap P$ la plus proche de p_2. Par convexité de P, l'arête de P intersectée par σ_2 en w_2 est orientée de droite à gauche et a une pente négative. De même, l'arête de σ_2 intersectée en w_2 a la même monotonie en x et en y que le segment orienté cd, et est donc orientée vers la

droite avec une pente négative. Par conséquent, w_2 est un sommet d'abscisse minimale parmi les sommets de Q_2 et le plus proche point entier de w_2 appartenant à Q_2, que l'on note q_2, correspond (si il existe) à un point d'abscisse minimale parmi les points entiers de Q_2. Comme v et Q_2 se trouve de part et d'autre de la droite L_2 et comme les points entiers appartenant à P et se trouvant à droite de L_2 sont exactement identiques aux points entiers appartenant à Q_2, le plus proche point entier de v à droite de L_2 et appartenant à P correspond à un point d'abscisse minimale parmi les points entiers de Q_2 et a donc même abscisse que q_2. Le cas où σ_1 intersecte P se traite de manière similaire par symétrie par rapport à la droite $y = x$, le point q_1 étant dans ce cas le point d'abscisse maximale parmi les points entiers de P gauche de L_1. □

Nous donnons dans la suite une description détaillée de l'algorithme permettant de déterminer, étant donné un polygone convexe P dont chaque arête admet un segment à coordonnées entières comme segment support et un sommet v de ce polygone, le plus proche point entier de v appartenant à P.

Dans un premier temps, une opération de symétrie par rapport à l'origine ou aux axes et appliquée aux arêtes de P incidentes à v de manière à ramener le secteur S dans l'une des trois situations génériques suivantes : l'axe y^+ (et seulement cet axe) se dirige dans S, le secteur S se situe dans le premier quadrant, les axes y^+ et x^+ se dirigent tous les deux dans le secteur S.

Dans le premier cas, on distingue deux situations suivant l'existence de points entiers appartenant à P sur les droites verticales de la grille L_1 et L_2 encadrant v. Si de tels points existent, la solution au problème peut être facilement obtenue puisque le plus proche point entier de v appartenant à P correspond soit à p_1, soit à p_2. Dans le cas contraire, le sommet h de P correspondant au sommet d'ordonnée maximale est déterminé. Si h se situe entre L_1 et L_2, l'algorithme termine puisque P ne peut contenir aucun point entier, sinon, l'algorithme calcule l'une des deux chaînes polygonales $\sigma_1 = \mathcal{D}(ab, a \to p_1)$ et $\sigma_2 = \mathcal{D}(cd, p_2 \to d)$ selon que h se situe à gauche de L_1 ou à droite de L_2. À partir de cette chaîne, un sous-polygone convexe Q de P et un nouveau sommet w de ce polygone sont calculés de telle sorte que la solution recherchée corresponde au plus proche point entier de w dans ce polygone (cf. lemme 3.7.2). Le polygone Q étant de taille inférieure à P (en termes de nombre de sommets cf. page 93), le problème de départ est donc réduit à un sous problème, et le point recherché est obtenu par récursion en appelant l'algorithme avec le polygone Q et le sommet w comme entrée. La récursion s'arrête lorsqu'un point entier est trouvé ou lorsque le sous-polygone construit lors d'un appel récursif est vide, l'existence d'un

point entier dans P étant dans ce cas impossible.

Dans le cas où le secteur S se situe après transformation dans le premier quadrant, l'algorithme opère similairement à partir du plus proche point entier p appartenant au secteur S. Si p appartient à P, ce point est clairement la solution recherchée, dans le cas contraire, d'après le lemme 3.7.3 et à partir du calcul de l'une des deux chaînes polygonales σ_1 et σ_2 (selon la position du sommet d'abscisse maximale de P relativement à $\overrightarrow{vp}$), un nouveau polygone Q et un nouveau sommet w de ce polygone peuvent être obtenus de sorte que la solution recherchée corresponde au plus proche point entier de w appartenant à Q. Là aussi, la solution recherchée est déterminée en appelant récursivement l'algorithme avec Q et w comme entrée jusqu'à ce qu'un point entier soit trouvé ou que le sous-polygone calculé soit vide.

Enfin, si le secteur S de départ contient après transformation les directions y^+ et x^+, l'algorithme distingue deux situations selon qu'il existe ou pas des points entiers qui appartiennent à P sur les droites verticales L_1 et L_2 de la grille encadrant le sommet v. Si au moins un tel point existe, la solution au problème posé est le point entier de L_1 ou de L_2 qui appartient à P et qui est le plus proche de v. Dans le cas contraire, d'après le lemme 3.7.4, et à partir des deux chaînes σ_1 et σ_2 au plus deux nouveaux polygones Q_1 et Q_2 et deux nouveaux sommets w_1 et w_2 appartenant à ces polygones peuvent être obtenus tels que, étant donné q le plus proche point entier de w_1 appartenant à Q_1 et r le plus proche point entier de w_2 appartenant à Q_2, la solution recherchée est le point entier appartenant à la droite $L_q : x = q_x$ ou à la droite $L_r : x = r_x$ qui est le plus près du sommet v. On note qu'au plus un axe se dirige dans chacun des secteurs convexes S_1 et S_2 définis respectivement par les arêtes incidentes à w_1 dans Q_1 et à w_2 dans Q_2. Par conséquent, afin de déterminer la solution au problème de départ, l'algorithme peut diviser le polygone d'entrée en deux nouveaux polygones lors du tout premier appel de l'algorithme mais jamais lors des appels récursifs.

Nous analysons dans la dernière partie de cette section les performances de l'algorithme en termes de complexité asymptotique et de degré algébrique et comparons celles-ci aux performances de l'algorithme proposé par Lee et Chang [47] qui constitue, à notre connaissance, la seule solution au problème publiée à ce jour et qui résout le problème en temps $O(n + \log l)$ où l dénote le diamètre[6] du polygone P et n le nombre de ses sommets,

Complexité de l'algorithme Comme nous l'avons vu à la section précédente, les points entiers p_1, p_2 et p peuvent être calculés en temps $O(\log N)$ si la grille entière contenant les arêtes supports

[6]Le diamètre d'un ensemble de points est la plus grande distance entre deux points de cet ensemble (cf. [61]).

aux arêtes de P est de taille $N \times N$. De même, les chaînes polygonales σ peuvent être construites en temps $O(\log N)$ à l'aide de l'algorithme proposé par Greene et Yao [35]. La partie cruciale de l'analyse de l'algorithme réside dans la détermination du nombre d'appels récursifs et du temps nécessaire à la construction des sous-polygones dans le pire des cas.

Si n désigne le nombre de sommets du polygone P initial, on montre que si le point recherché ne peut être déterminé (ou si sa non-existence ne peut être prouvée) à partir de p (ou à partir de p_1 et p_2 selon le cas), le sous-polygone utilisé lors de l'appel récursif suivant a au plus $n-1$ sommets. Ainsi au plus n appels récursifs sont nécessaires pour calculer la solution au problème initial. On note dans la suite s l'arête de σ qui intersecte P en w et $H(s)$ le demi-plan induit par l'arête s qui ne contient pas v.

On distingue deux cas distincts selon que s possède ou non l'une de ses extrémités dans P. Supposons, tout d'abord, qu'aucune des extrémités de s n'appartienne à P. Dans ce cas, le polygone Q correspond à l'intersection de P avec $H(s)$. Le polygone Q construit contient alors clairement au plus $n-1$ arêtes, puisque, par construction de σ, aucune des deux arêtes de P incidentes à v ne peut apparaître dans Q, et une seule nouvelle arête (due à l'intersection avec le demi-plan) est introduite.

Le cas où l'une des extrémités de s appartient à P est plus délicat puisqu'une partie des arêtes de P incidentes à v peut alors apparaître dans Q. On montre que l'on peut cependant remplacer le polygone Q introduit précédemment par un polygone R possédant au plus $n-1$ sommets sans modifier le résultat final de l'algorithme. Pour cela, soit $H_n(P)$ l'ensemble des n demi-plans définissant P et $H_{n-2}(P)$ l'ensemble obtenu en supprimant les deux demi-plans induits par ab et cd de $H_n(P)$. On montre dans la suite que la région convexe R définie comme l'intersection de $H(s)$ avec $H_{n-2}(P)$ et qui contient au plus $n-1$ sommets est telle que le plus proche point entier de w dans R est identique au plus proche point entier de w dans Q (cf. figure 3.13). Pour cela, supposons qu'il existe un sommet entier p' appartenant à R tel que la distance de p' à w soit inférieure à la distance de w au plus proche point entier de w appartenant à Q. Par construction de R et puisque $p' \notin Q$, p' appartient nécessairement à $S_w \setminus Q$ où S_w désigne le secteur convexe défini par les arêtes de Q incidentes à w (cf. figure 3.13). Or ceci est impossible puisque dans ce cas le point entier p'' correspondant à l'extrémité de s appartenant à P est forcément plus proche de w que p'. Ainsi, l'algorithme effectue donc au plus n appels récursifs si n est le nombre de sommets du polygone de départ.

À chaque itération, le test d'intersection entre la chaîne σ et la frontière de P ainsi que la détermination du sommet w peuvent être faits en temps $O(\log N \log n)$ en effectuant pour chacune des $O(\log d) < O(\log N)$ arêtes de la chaîne orientée σ une recherche dichotomique sur les

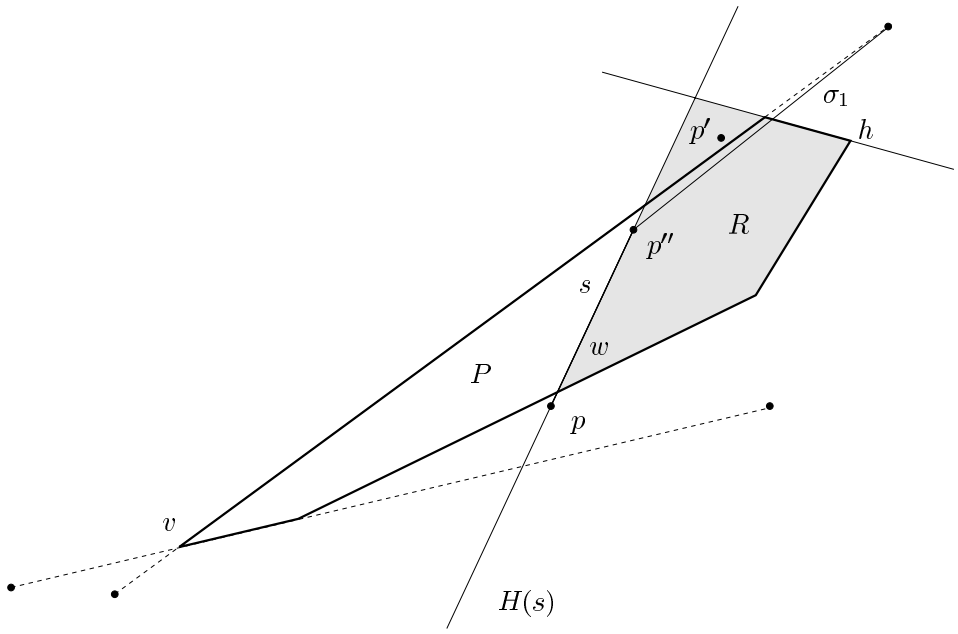

FIG. 3.13 – Le polygone R (grisé sur la figure) a au plus $n-1$ sommets si P possède n sommets.

sommets de P. Le même temps est nécessaire pour la construction de l'intersection de P avec au plus deux demi-plans. Une itération complète coûte donc au plus $O(\log n \log N)$. La complexité asymptotique totale de l'algorithme est donc $O(n \log n \log N)$.

Degré algébrique de l'algorithme Les expressions numériques de plus haut degré algébrique évaluées par l'algorithme correspondent clairement au calcul à chaque itération du nouveau sommet w à arrondir et à la détermination du plus proche point entier de v dans le cas où l'algorithme retourne deux points candidats (cas où y^+ et x^+ se dirigent dans le secteur associé aux arêtes incidentes au sommet initial v).

Étant donnée une chaîne σ et un polygone convexe P, la construction du nouveau sommet w consiste à calculer (si il existe) le point d'intersection de σ avec une arête de P le plus proche d'une des extrémités de σ. Si chacune des arêtes de P admet un segment à coordonnées entières comme segment support, la détermination des arêtes $s \in \sigma$ et $uv \in P$ s'intersectant en w nécessite de savoir tester l'intersection de deux segments à coordonnées entières et de savoir ordonner sur le segment support ab à l'arête s_i les points d'intersection correspondant aux extrémités de s et à l'intersection de s_i avec uv. Si l'on note respectivement a_ib_i et $a_{i'}b_{i'}$ les arêtes supports à deux arêtes successives de P et uv une arête de σ intersectant a_ib_i, le segment orienté a_ib_i est

intersecté par uv avant $a_{i'}b_{i'}$ si et seulement si

$$|[a_i, u, v]([a_i, b_i, a_{i'}] - [a_i, b_i, a_{i'}])| < |[a_i, a_{i'}, b_{i'}]([a_i, b_i, u] - [a_i, b_i, v])|$$

et la détermination des arêtes définissant w nécessite donc l'évaluation d'expressions de degré quatre (la construction proprement dite de w est de degré trois puisque les deux arêtes définissant w sont toutes les deux à coordonnées entières).

Dans le cas où l'algorithme retourne deux candidats g_1 et g_2 à l'arrondi d'un sommet v de P, la détermination de la solution recherchée revient à évaluer le signe de $S = ||g_2 v|| - ||g_1 v||$, le sommet v étant plus proche de g_1 que de g_2 si $S > 0$, plus proche de g_2 que de g_1 si $S < 0$, et équidistant de g_1 et g_2 si $S = 0$.

À l'aide du calcul suivant, on montre que ce prédicat est aussi de degré quatre (on rappelle que g_1 et g_2 ont des coordonnées entières et que v_x et v_y sont des nombres rationnels ayant un dénominateur identique de degré 2 et un numérateur de degré 3) :

$$\begin{aligned}
\text{signe}(S) &= \text{signe}(\|g_2 v\|^2 - \|g_1 v\|^2) \\
&= \text{signe}(((g_{2x} - v_x)^2 - (g_{1x} - v_x)^2) + ((g_{2y} - v_y)^2 - (g_{1y} - v_y)^2))) \\
&= \text{signe}(((g_{2x}^2 - g_{1x}^2) - 2v_x(g_{2x} - g_{1x}) + (g_{2y}^2 - g_{1y}^2) - 2v_y(g_{2y} - g_{1y})))
\end{aligned}$$

Ainsi l'algorithme nécessite l'évaluation de rationnels de degré au plus quatre.

En comparaison, l'algorithme proposé par Lee et Chang nécessite l'évaluation de nombres algébriques faisant intervenir des fonctions trigonométriques, des racines carrées et des nombres rationnels de haut degré. Leur algorithme nécessite entre autres de savoir calculer le plus proche point entier du centroïde d'un triangle ayant pour sommets des sommets de P, ce qui correspond à une construction de degré 29 si les sommets de P sont donnés sous la forme de points d'intersection de segments à coordonnées entières.

D'autre part, malgré une complexité asymptotique moins bonne que celle de l'algorithme développé par Lee et Chang, notre solution présente l'avantage de traiter tout sommet convexe inférieur ou égal à π (la méthode de Lee et Chang ne traite que les sommets convexes inférieur à $\frac{\pi}{2}$) et s'avère très efficace en pratique (le nombre d'itérations nécessaires à l'algorithme s'avère être de l'ordre d'une petite constante).

Chapitre 4

Constructions géométriques complexes à précision fixée

Ce chapitre aborde le problème de l'arrondi d'opérations booléennes sur des régions polygonales du plan. Le calcul de telles opérations arrondies est crucial dans de nombreux domaines applicatifs comme la conception assistée par ordinateur ou l'informatique graphique. Les modeleurs géométriques éditent, en effet, des formes qui sont réutilisées et modifiées un grand nombre de fois (par des opérations de translation, de rotation, d'intersection ...). Si ces opérations sont effectuées de manière exacte, la complexité numérique des valeurs manipulées (typiquement les coordonnées des sommets) est dans ce cas directement proportionnelle au nombre de modifications effectuées. Plus précisément, l'espace nécessaire à la représentation exacte d'un sommet croit exponentiellement avec la hauteur de son arbre de construction [55], le temps de calcul nécessaire aux opérations sur de telles constructions augmente par conséquent dramatiquement [73]. Afin de contrôler la complexité numérique des sommets issus d'opérations cascadées et de limiter les exigences arithmétiques d'une implantation robuste d'un algorithme utilisant ces constructions, il faut donc être capable de définir une représentation à précision fixée de ces objets géométriques.

De la même manière que le résultat d'une opération flottante est arrondi à un nombre flottant, on cherche ainsi à arrondir les structures géométriques. Cet arrondi ne peut être effectué de manière brutale, par exemple en arrondissant les coordonnées de points calculés car certaines propriétés caractéristiques de la structure pouvant être réutilisées par d'autres algorithmes seraient perdues. Par exemple un polygone peut être simple et convexe par nature et perdre ces propriétés dans un arrondi sans précaution. De plus, cette perte peut affecter directement le déroulement d'algorithmes supposant des entrées dotées de telles propriétés.

L'arrondi de structures géométriques est cependant un sujet peu exploré et très peu de travaux ont été réalisés dans cette direction à l'exception de méthodes pour l'arrondi de certaines subdivisions polygonales et polyédriques. Des solutions satisfaisantes garantissant la robustesse des algorithmes en préservant certaines propriétés topologiques de la structure sont ainsi seulement connues pour l'arrondi d'arrangement de segments [35, 32, 36, 42, 55, 58], le diagramme de Voronoï [14] et l'arrangement de triangles dans l'espace [27].

Nous traitons ici le problème de l'arrondi du résultat d'opérations booléennes (intersection, réunion, complémentaire, ...) sur des régions polygonales du plan et présentons une méthode de régularisation qui permet de garantir certaines propriétés topologiques et géométriques intéressantes entre le résultat exact et sa version arrondie. Plus précisément, étant données deux régions polygonales planaires A et B ayant leur sommets sur la grille entière, nous introduisons dans la suite deux notions différentes d'arrondi, le mode d'arrondi *par défaut* et le mode d'arrondi *par excès*, construisant deux régions polygonales $A \underline{\cap} B$ et $A \overline{\cap} B$ respectant les coordonnées entières et telles que $A \underline{\cap} B \subseteq A \cap B \subseteq A \overline{\cap} B$. Nous prouvons enfin certains résultats intéressants sur la distance de Hausdorff, la taille et la convexité vérifiées par ces régions arrondies.

La suite de ce chapitre est organisée de la manière suivante. La section 4.1 examine les problèmes liés à l'arrondi d'arrangements de segments de droite dans le plan et inventorie les différentes solutions proposées dans la littérature. La section 4.2 introduit quelques notations et définitions et présente les notions d'arrondi d'intersections de régions polygonales *par défaut* et *par excès* ainsi que les méthodes permettant de les calculer. Cette section présente enfin les propriétés de ces modes d'arrondi pour les opérations d'intersection et de réunion ainsi que leur généralisation au cas de régions polygonales quelconques.

4.1 Arrondi d'arrangement des segments de droite dans le plan

Une technique d'arrondi géométrique doit avoir au moins les deux propriétés suivantes : 1) garantir un objet arrondi proche de l'objet original, et 2) préserver le plus possible les propriétés de la structure originale (par exemple, les relations d'incidences dans le cas de l'arrangement). La première condition nous suggère d'arrondir chaque sommet de l'arrangement exact vers son plus proche point appartenant à la grille. Cependant, il est clair que cet arrondi sans précaution ne prévient pas des inconsistances topologiques entre l'arrangement original et sa version arrondie.

Problèmes avec l'arrondi d'intersections

Greene et Yao [35] inventorient les problèmes pouvant survenir lorsqu'un arrangement de segments est plongé dans un ensemble de valeurs entières, c'est-à-dire lorsque chaque point d'intersection est arrondi au point à coordonnées entières le plus proche.

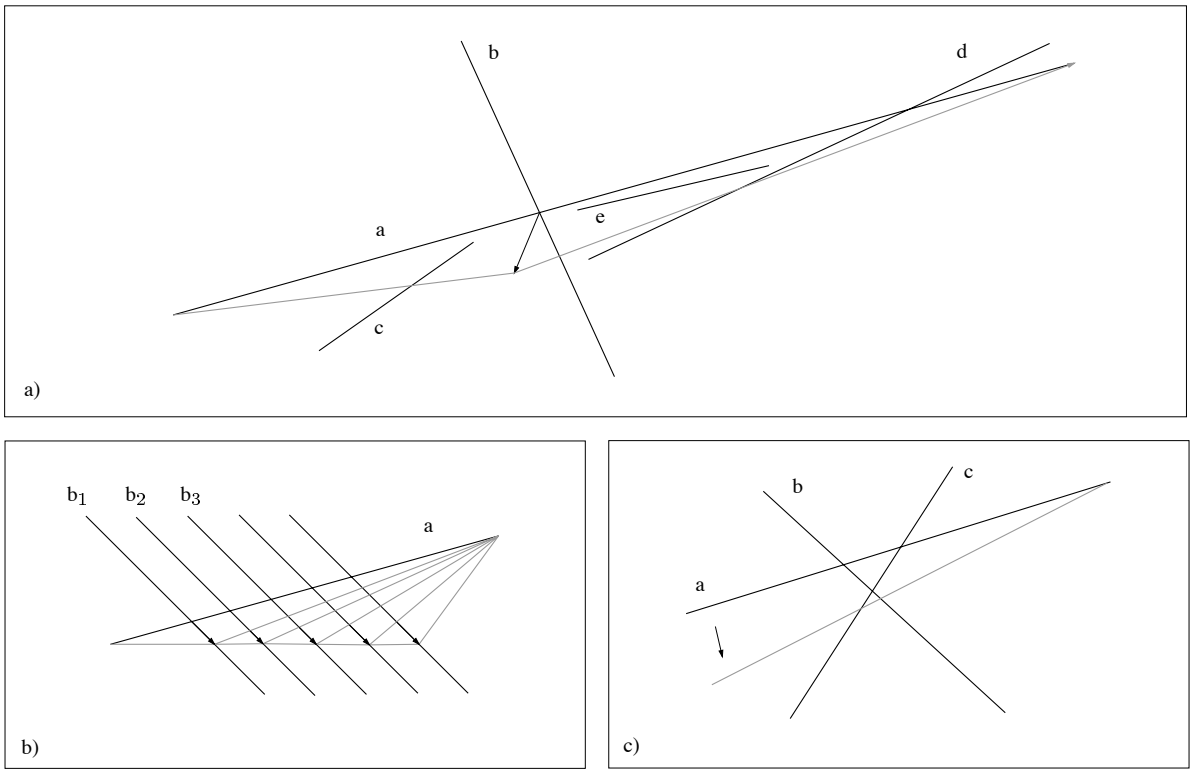

FIG. 4.1 – a) Problèmes avec les arrondis d'intersections, b) Dérive de segment, c) Inversion.

Les auteurs distinguent les problèmes suivants :

- *Fausses intersections* : l'intersection arrondie du segment a et du segment b de la figure 4.1.a crée une intersection entre le segment a et le segment c qui n'était pas présente dans l'arrangement original.
- *Itérations multiples :* la propagation de *fausses* intersections nécessite de réitérer l'algorithme d'intersection jusqu'à ce qu'aucune intersection ne soit détectée.
- *Déplacement des points d'intersections* : dans la figure 4.1.a, le point d'intersection des segments a et d a été déplacé de sa position originale d'une distance considérable.
- *Dérive de segments* : la figure 4.1.b montre qu'une série d'intersections du segment a avec les segments $b_1, b_2, \ldots, b_n$ peut provoquer une *dérive* du segment original d'une distance proportionnelle à n.

– *Inversion topologique :* l'ordre des segments a et e a été inversé dans la figure 4.1.a ; la *dérive* d'un segment a inversé l'orientation d'un triangle dans la figure 4.1.c.

La section suivante passe en revue les différentes approches de la littérature permettant d'arrondir à une précision inférieure un arrangement de segments tout en étant exempt des différents problèmes mentionnés ci-dessus. L'ensemble de ces solutions présente l'avantage de s'appliquer à de nombreux algorithmes et d'avoir une erreur d'arrondi bornée. Elles changent, cependant, la structure combinatoire du résultat si nécessaire. De par la nature discrète de la grille, aucune méthode d'arrondi ne peut en effet espérer préserver la topologie originale exactement : des segments ou des points disjoints doivent être fusionnés. De même, aucune technique connue ne préserve parfaitement la rectitude de tout les segments ; Milenkovic et Nackman [54] ont par ailleurs démontré qu'il peut être impossible de préserver cette dernière propriété[1].

4.1.1 La technique de perturbation de Greene-Yao

Greene et Yao [35] ont été les premiers à développer une technique d'approximation à précision finie permettant de plonger un arrangement exact de segments de droite dans le plan dans un ensemble de valeurs entières tout en conservant un certain nombre de propriétés topologiques.

Leur approche consiste à considérer chaque segment comme un ruban flexible :
– Chaque sommet de l'arrangement est perturbé au point de la grille le plus proche (notons ces points $r(a)$ et $r(b)$ pour le segment ab).
– Pour tout autre sommet v, si le segment $vr(v)$ intersecte ab, déformer le ruban $r(a)r(b)$ jusqu'à $r(v)$.
– Chaque point de la grille est considéré comme un obstacle et on interdit aux rubans de passer au dessus de ces obstacles.

Certains obstacles ajoutent des replis (*kinks*) à la représentation arrondie d'un segment, et chaque segment est par conséquent transformé en une chaîne polygonale contenant plusieurs segments (cf. figure 4.2.b). Cette méthode de fragmentation présente plusieurs avantages. D'un part, elle permet de garantir que l'on n'introduit pas de nouvelles intersections (de nouvelles incidences, par contre, peuvent apparaître), ni d'inconsistance topologique. D'autre part, elle assure qu'il n'existe pas de point de la grille entre le segment original et sa version arrondie. En contrepartie, le nombre de sommets additionnels requis peut être très grand. Cette méthode

[1] Déterminer l'existence d'une approximation à une précision inférieure d'une structure géométrique préservant la structure combinatoire originale est un problème NP-complet.

introduit précisément $\Omega(k \log |ab|)$ sommets en trop sur la version arrondie du segment ab, où k est le nombre de points d'intersection portés par le segment ab. On note enfin que la version arrondie de ab ne dépend que du segment ab et des segments qui le coupent et non de ceux qui lui sont proches.

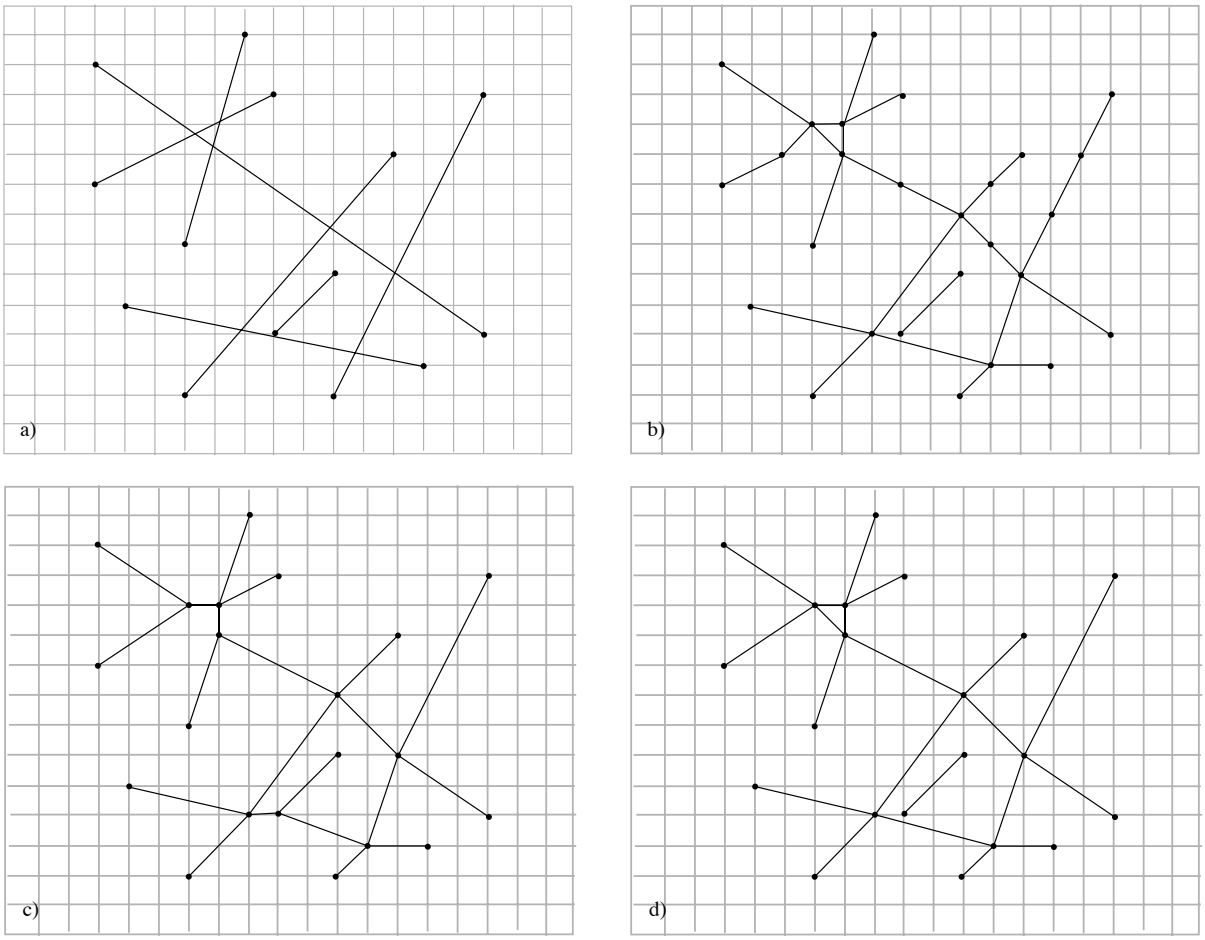

FIG. 4.2 – a) Un arrangement de segments, b) sa version perturbée par la méthode de Greene-Yao, c) sa forme arrondie par la méthode du Snap Rounding et d) sa forme arrondie par la méthode du Shortest Path Rounding.

4.1.2 Le paradigme du Snap Rounding

La méthode du Snap Rounding [35, 32, 36, 42, 58] résout le problème de cette fragmentation excessive des segments originaux en réduisant le nombre de sommets nécessaires à la représentation arrondie de l'arrangement.

Le principe du Snap Rounding est le suivant :

- Chaque pixel (carré de la grille entière) contenant une extrémité ou un point d'intersection, c'est-à-dire un sommet de l'arrangement, est considéré comme *chaud*.
- Chaque segment intersectant un pixel chaud est fragmenté afin de passer par son centre.

Le Snap Rounding constitue une approche intéressante de par sa simplicité et ses *bonnes* propriétés mathématiques. Il est, par exemple, très facile de démontrer que la version arrondi s' d'un segment original s est à au plus une distance $\frac{\sqrt{2}}{2}$ de s, c'est-à-dire est comprise dans la somme de Minkowski de s avec un pixel centré à l'origine [36]. Cette technique est, de plus, combinatoirement efficace puisqu'elle génère une fragmentation raisonnable des segments originaux pour garantir la consistance topologique (cf. figure 4.2.c). La complexité de l'arrangement arrondi, en termes de nombre de sommets, est en effet plus petite ou égale à celle de l'arrangement exact (plusieurs sommets distincts peuvent en effet s'arrondir à un même pixel), le nombre d'arêtes distinctes de l'arrangement arrondi est par contre en $\Theta(n^2)$ si n est le nombre de segments en entrée de l'arrangement (la version arrondie d'un segment dépend de l'ensemble des segments qui lui sont proches).

4.1.3 Le paradigme du Shortest Path Rounding

La technique du Shortest Path Rounding a été introduite par Milenkovic [55]. Le principe de cette technique est d'arrondir chaque sommet non représentable sur la grille au point représentable le plus proche et de remplacer chaque segment par une chaîne polygonale qui passe par ses sommets arrondis et maintient tout les autres sommets arrondis du bon côté selon une condition de plus court chemin (cf. figure 4.2.d).

Cette méthode présente l'avantage d'introduire une erreur géométrique et un changement combinatoire minimal. De plus, contrairement aux autres techniques d'arrondi géométrique, cette méthode ne s'applique pas seulement à l'arrondi sur la grille entière mais à n'importe quel treillis ayant des cellules connectés.

4.2 Arrondi d'opérations booléennes de régions polygonales

4.2.1 Notations et définitions

Dans le plan muni du repère cartésien usuel, nous considérons l'ensemble des *points entiers*, *i.e.* l'ensemble des points appartenant à la grille entière $\mathbb{Z}^2$.

Nous utilisons dans la suite les notations suivantes. Nous désignons par *polygone à coordonnées entières* un polygone simple, c'est-à-dire un polygone fermé ayant un intérieur et un extérieur bien définis (les incidences sont permises : un sommet ou une arête d'un polygone peut apparaître plusieurs fois dans la liste des sommets ou des arêtes de ce polygone), dont chaque sommet est un point entier, et par *région polygonale à coordonnées entières* une région du plan composée de polygones simples à coordonnées entières et pouvant inclure des trous à un niveau de profondeur arbitraire (cf. figure 4.3).

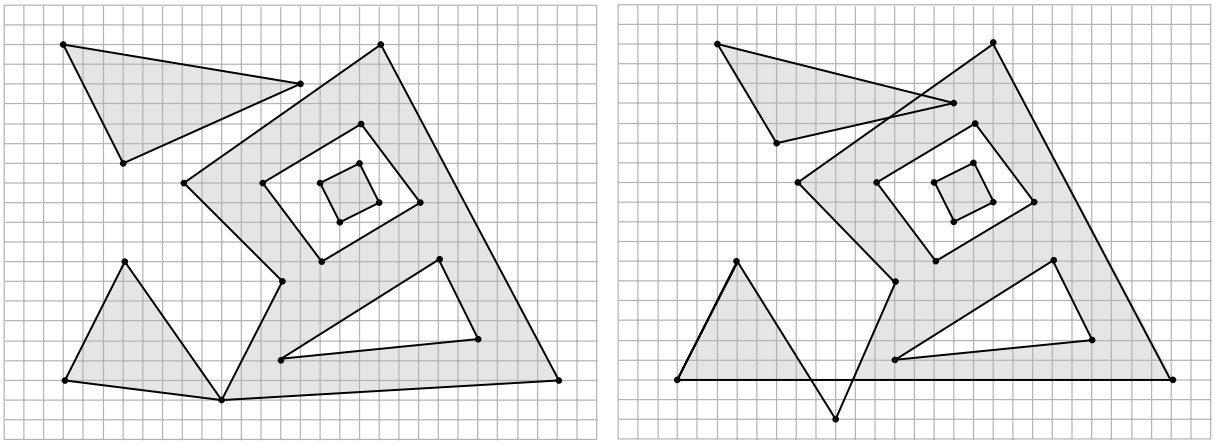

FIG. 4.3 – À gauche, une région polygonale valide avec des trous. À droite, une région polygonale non valide (la frontière s'auto-intersecte).

Nous utiliserons de plus les définitions suivantes. Soit p et q deux points de A, on dit que q est *visible* depuis p dans A si chaque point du segment de droite pq appartient à A, c'est-à-dire $pq \cap \mathsf{A} = pq$. La *région de visibilité*, $V_{\mathsf{A}}(p)$ d'un point $p \in \mathsf{A}$ est définie comme l'ensemble des points $q \in \mathsf{A}$ visibles depuis p dans A. Le *plus proche point représentable* de p, noté $v_{\mathsf{A}}(p)$, est défini comme le plus proche point de p appartenant à $V_{\mathsf{A}}(p)$ ayant des coordonnées entières. Étant donné un sommet $v \in \mathsf{A}$ et une arête $e \in \mathsf{A}$, v est dit *verticalement visible* depuis e s'il existe un segment de droite vertical connectant v à e et entièrement contenu dans A.

Enfin, on dit qu'une région polygonale A est *incluse* dans une région polygonale B, et on note $\mathsf{A} \subseteq \mathsf{B}$, lorsque tout point appartenant à A est aussi un point de B. Le complémentaire de A,

c'est-à-dire l'ensemble des points du plan qui *ne sont pas* dans A, est noté A^C. La frontière de A est notée $\partial\mathsf{A}$. Finalement, on note $|\mathsf{A}|$ la *complexité* de la région A, c'est-à-dire le nombre de sommets distincts de A.

Définition 4.2.1 ***(Distance de Hausdorff entre deux ensembles)***
Étant donnés deux ensembles A *et* B*, la distance de Hausdorff entre* A *et* B *est définie par la fonction*

$$d_H(\mathsf{A},\mathsf{B}) = \max(d_h(\mathsf{A},\mathsf{B}), d_h(\mathsf{B},\mathsf{A}))$$

où

$$d_h(\mathsf{A},\mathsf{B}) = \max_{a\in\mathsf{A}} \min_{b\in\mathsf{B}} d(a,b)$$

est la mesure de Hausdorff directionnelle entre A *et* B *et* $d(a,b)$ *désigne dans notre cas la distance euclidienne entre les points* a *et* b*.*

La distance de Hausdorff mesure la proximité mutuelle de deux ensembles de points. Elle indique la distance maximale d'un ensemble au point le plus proche de l'autre ensemble. Dans le cas où les ensembles A et B sont des régions polygonales, $d_h(\mathsf{A},\mathsf{B})$ s'applique à tout les points définissant ces régions (*i.e.* aux points appartenant à la frontière et à l'intérieur de la région).

4.2.2 Opérations et modes d'arrondi

On définit plusieurs modes d'arrondi au niveau des régions polygonales. Ces modes d'arrondi affectent le résultat des opérations booléennes (*union* $\cup$*, intersection* $\cap$ et *différence* $\setminus$) effectuées sur des régions polygonales à coordonnées entières.

On distingue les deux modes suivants :

- *par défaut*. Les sommets du résultat de l'opération booléenne sont arrondis aux points de la grille de manière à obtenir l'inclusion de la région arrondie dans le résultat exact (cf. figure 4.4).
- *par excès*. Les sommets du résultat de l'opération booléenne sont arrondis aux points de la grille de manière à obtenir l'inclusion du résultat exact dans la région arrondie (cf. figure 4.4).

On notera par la suite, $\underline{\cap}$ l'opération d'intersection de régions polygonales utilisant le mode d'arrondi par défaut et $\overline{\cap}$ celle utilisant le mode d'arrondi par excès. On rappelle que par application des lois de Morgan, toutes les opérations booléennes (à savoir l'intersection $\cap$, la réunion

$\cup$, le complémentaire ^{C}, la différence $\setminus$) se réduisent à l'intersection et au complémentaire. On définit donc de manière similaire $\underline{\cup}$, $\overline{\cup}$, $\underline{\setminus}$ et $\overline{\setminus}$.

Ces différents modes d'arrondi préservent certaines propriétés et critères qu'il nous faut définir.

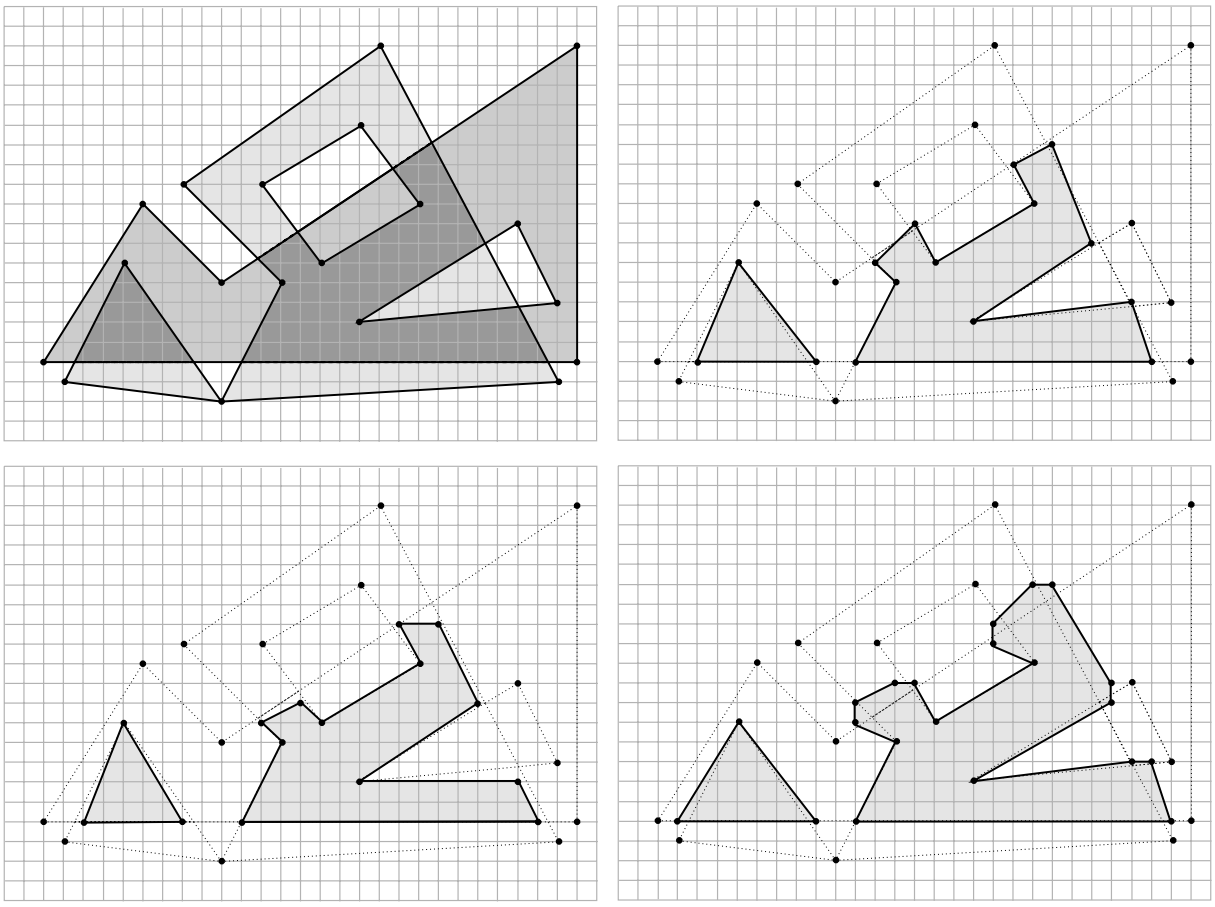

FIG. 4.4 – De gauche à droite et de haut en bas, deux régions polygonales à coordonnées entières, la version arrondie de leur intersection selon le paradigme du Shortest Path Rounding, la version arrondie par défaut, la version arrondie par excès.

Le mode d'arrondi par défaut Étant donné R le résultat de l'intersection de deux régions polygonales à coordonnées entières A et B, le plongement $\underline{\mathsf{R}}$ de R dans un ensemble de valeurs entières selon le mode d'arrondi par défaut vérifie les propriétés suivantes :

1. la région polygonale $\underline{\mathsf{R}}$ est une région polygonale valide à coordonnées entières,
2. $\underline{\mathsf{R}}$ est inclus dans R,

3. $d_H(\mathsf{R}^C, \underline{\mathsf{R}}^C) < \sqrt{2}$,
4. $|\underline{\mathsf{R}}| \leq |\mathsf{R}|$,
5. Un sommet concave de $\underline{\mathsf{R}}$ a toujours un correspondant concave dans R.

Le mode d'arrondi par excès Étant donné R le résultat de l'intersection de deux régions polygonales à coordonnées entières A et B, le plongement $\overline{\mathsf{R}}$ de R dans un ensemble de valeurs entières selon le mode d'arrondi par excès vérifie les propriétés suivantes :

1. la région polygonale $\overline{\mathsf{R}}$ est une région polygonale valide à coordonnées entières,
2. $\overline{\mathsf{R}}$ contient R,
3. $d_H(\,\overline{\mathsf{R}}\,,\,\mathsf{R}\,) < \sqrt{2}$,
4. $|\overline{\mathsf{R}}| < 2|\mathsf{R}| + 3k$ où k est le nombre de sommets de R résultant de l'intersection d'une arête de A et d'une arête de B.

Propriétés des opérations booléennes arrondies Les modes d'arrondi par excès et par défaut vérifient les propriétés suivantes :

$$\mathsf{A} \mathbin{\underline{\cup}} \mathsf{B} = \left((\mathsf{A}^C) \mathbin{\overline{\cap}} (\mathsf{B}^C)\right)^C, \;\; \mathsf{A} \mathbin{\overline{\cup}} \mathsf{B} = \left((\mathsf{A}^C) \mathbin{\underline{\cap}} (\mathsf{B}^C)\right)^C$$

et

$$\mathsf{A} \mathbin{\underline{\setminus}} \mathsf{B} = \mathsf{A} \mathbin{\underline{\cap}} (\mathsf{B}^C), \;\; \mathsf{A} \mathbin{\overline{\setminus}} \mathsf{B} = \mathsf{A} \mathbin{\overline{\cap}} (\mathsf{B}^C).$$

On note que le passage au complémentaire d'une région polygonale A est trivial et consiste généralement à inverser l'orientation de chacun des polygones définissant A.

Les opérations arrondies d'intersection et de réunion sont commutatives, c'est-à-dire $\mathsf{A} \circ \mathsf{B} = \mathsf{B} \circ \mathsf{A}$ pour toutes régions polygonales à coordonnées entières A et B et pour $\circ \in \{\underline{\cap}, \overline{\cap}, \underline{\cup}, \overline{\cup}\}$. Le passage au complémentaire est idempotent : $(\mathsf{A}^C)^C = \mathsf{A}$ pour tout A.

La suite de cette section décrit les algorithmes de calcul de telles approximations. Étant données A et B, le calcul de leur intersection se déduit facilement du calcul de l'arrangement de leurs arêtes. Le résultat, s'il existe, est une région polygonale définie par les faces et les arêtes de l'arrangement appartenant à la fois à A et à B. Les seuls sommets de R non représentables sur la grille entière (et par conséquent à arrondir) sont des points d'intersection entre une arête de A et une arête de B. Ces sommets sont des sommets convexes de R, ils forment un angle intérieur inférieur à π.

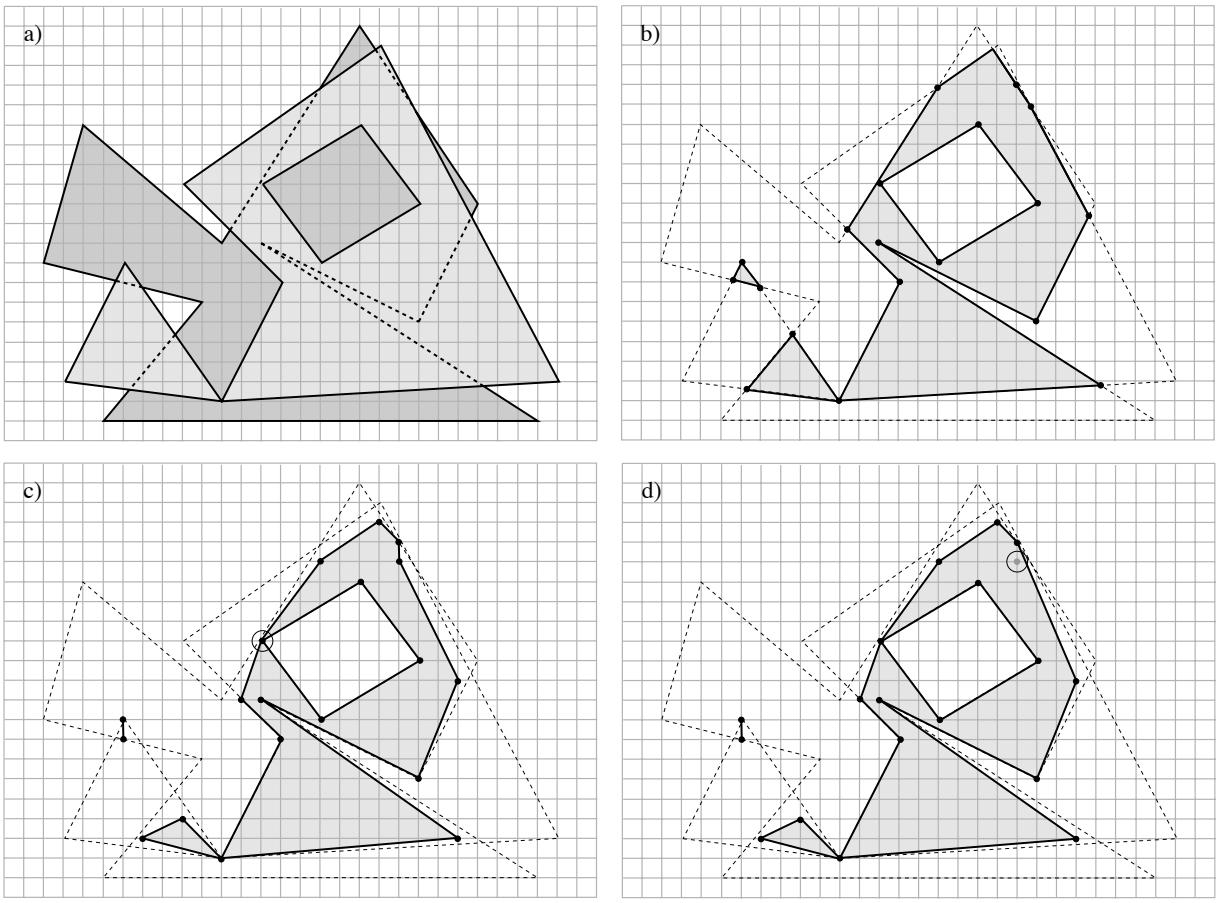

FIG. 4.5 – a) Les deux régions polygonales en entrée. b) La frontière de la région exacte d'intersection est vue comme un ruban élastique fixé en chacun des sommets par un clou. c) Chaque sommet non représentable est perturbé au plus proche point entier de son polygone de visibilité et on interdit au ruban élastique de traverser ces obstacles (cf. le sommet cerclé). d) Élimination des sommets ayant perdu leur convexité (cf. le sommet cerclé).

4.2.3 Intersection arrondie par défaut

Principe de l'arrondi

Étant données deux régions polygonales valides (cf. figure 4.5.a), le processus d'arrondi selon le mode par défaut peut se décrire simplement à l'aide de l'analogie utilisée dans [35]. L'idée consiste à voir la frontière de la région exacte d'intersection comme un ruban élastique cloué en chaque sommet (cf. figure 4.5.b). Chacun des clous est alors considéré comme un obstacle et on interdit au ruban élastique de traverser ces obstacles pendant la phase d'arrondi de chacun des sommets au plus proche point entier de son polygone de visibilité (cf. figure 4.5.c). La dernière

étape consiste finalement à supprimer de la région obtenue l'ensemble des sommets formant un angle concave et ne correspondant pas à un sommet concave de la région d'intersection exacte (cf. figure 4.5.d).

L'algorithme est basé sur la carte de visibilité verticale des sommets concaves de R. Le but de la construction d'une telle carte est double : 1) celle-ci permet, tout d'abord, d'obtenir une décomposition de la région d'intersection originale en polygones convexes permettant de se passer de calculs de visibilité complexes, 2) elle détermine pour chaque arête de la région d'intersection un sous-ensemble des sommets originaux introduisant potentiellement des nouvelles intersections.

Décomposition verticale des sommets concaves

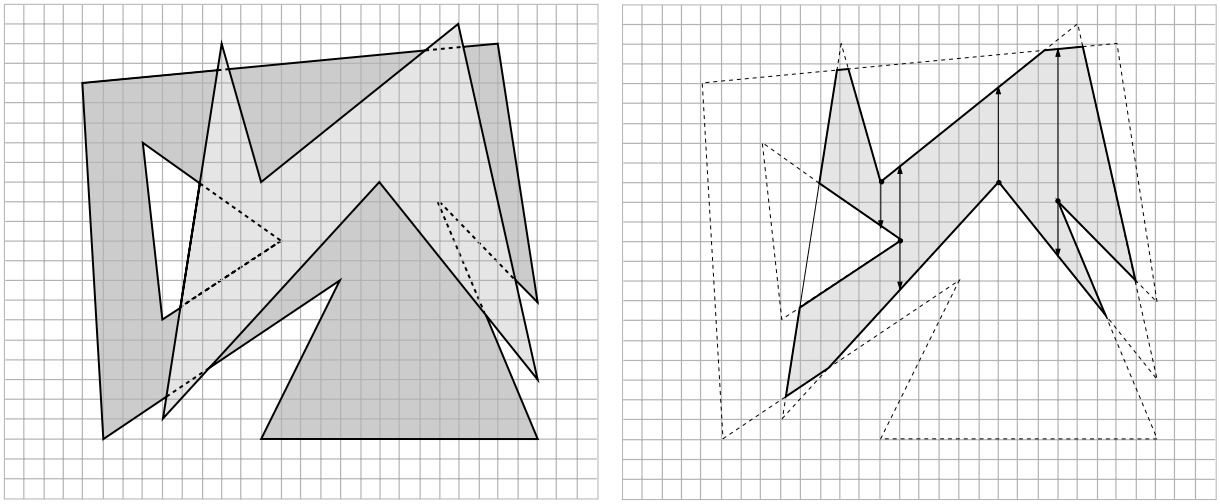

FIG. 4.6 – À droite, deux régions polygonales. À gauche, la carte de visibilité verticale des sommets concaves de leur intersection.

Étant donné R l'intersection de deux régions polygonales à coordonnées entières A et B, les seuls sommets de R n'appartenant pas à la grille entière sont les sommets de R résultant de l'intersection d'une arête de A avec une arête de B. Par définition de l'opération d'intersection, ces sommets forment seulement des sommets convexes R. Les sommets concaves de R correspondent par conséquent à des sommets originaux appartenant à l'une des deux régions en entrée et sont donc des sommets entiers.

La carte de visibilité verticale des sommets concaves de R, notée $DV(\mathsf{R})$, peut être définie comme une décomposition de R obtenue en traçant, à partir de chaque sommet concave de R, deux segments verticaux, appelés cloisons, vers le haut et vers le bas que l'on étend dans R jus-

qu'à leur rencontre avec l'extérieur de R (cf. figure 4.6). Le résultat de la décomposition verticale des sommets concaves de R est une partition de R en polygones convexes.

Les lemmes 4.2.2 et 4.2.3 énoncent les propriétés de la carte de visibilité verticale des sommets concaves de l'intersection de deux régions polygonales à coordonnées entières du plan.

Lemme 4.2.2 *Étant donné* R *le résultat exact de l'intersection de régions polygonales à coordonnées entières et* p *un sommet de* R*, si* C *est une cellule convexe de* $DV(\mathsf{R})$ *ayant* p *comme sommet, alors* $v_{\mathsf{R}}(p) = v_C(p)$.

Démonstration: La démonstration est basée sur un raisonnement par l'absurde. Supposons que $v_{\mathsf{R}}(p) \neq v_C(p)$. Les points entiers $v_{\mathsf{R}}(p)$ et $v_C(p)$ étant distincts, $v_{\mathsf{R}}(p)$ ne peut appartenir à C. Par conséquent, le segment de droite connectant p à $v_{\mathsf{R}}(p)$ doit intersecter la frontière de C. Puisque $v_{\mathsf{R}}(p)$ est visible depuis p, l'arête de C intersectée correspond à une cloison de la décomposition verticale de R. Or ceci est impossible car par construction de la carte $DV(\mathsf{R})$, il existe alors deux

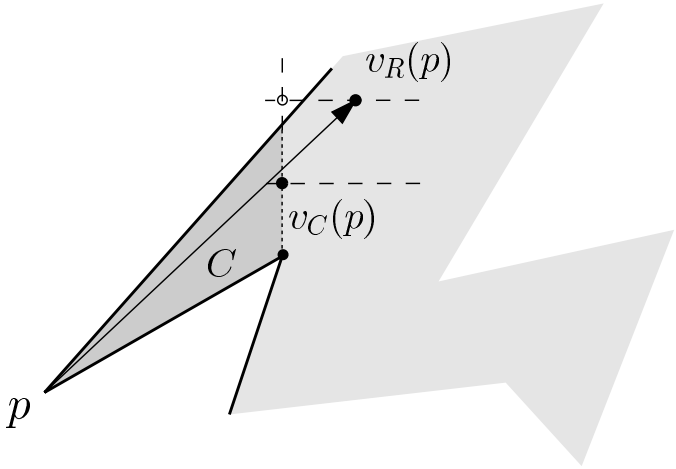

FIG. 4.7 – Si p est un sommet du polygone convexe C alors $v_{\mathsf{R}}(p) = v_C(p)$.

points entiers immédiatement au dessus et au dessous du point d'intersection plus proches de p que $v_{\mathsf{R}}(p)$ (cf. figure 4.7). Au moins l'un de ces deux points appartient à la cloison intersectée par le segment connectant p à $v_{\mathsf{R}}(p)$ et est donc, par convexité de C, visible depuis p. Ceci contredit le fait que $v_{\mathsf{R}}(p) \notin C$ et par conséquent l'hypothèse faite au début de la démonstration. □

Lemme 4.2.3 *Soient* R *le résultat exact de l'intersection de régions polygonales à coordonnées entières,* pq *une arête de* R*,* $\underline{\mathsf{R}}$ *la version arrondie par défaut de* R*, et* $\sigma(pq)$ *la chaîne polygonale*

connectant $v_{\mathsf{R}}(p)$ à $v_{\mathsf{R}}(q)$ et correspondant à la version arrondie de pq dans $\underline{\mathsf{R}}$*. L'ensemble des sommets de $\sigma(pq)$ entre $v_{\mathsf{R}}(p)$ et $v_{\mathsf{R}}(q)$ sont des sommets concaves de* R *verticalement visibles depuis pq dans* R*.*

Démonstration: Par construction de la chaîne arrondie $\sigma(pq)$, les sommets de $\sigma(pq)$ entre $v_{\mathsf{R}}(p)$ et $v_{\mathsf{R}}(q)$ ne peuvent correspondre qu'à des sommets concaves de R. Il nous suffit de montrer que l'ensemble de ces sommets sont verticalement visibles depuis pq dans R. Ici encore, la démonstration utilise un raisonnement par l'absurde.

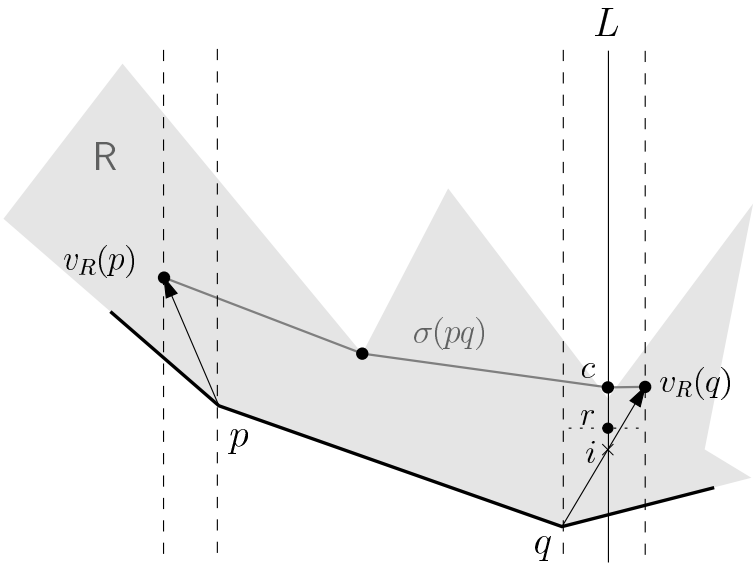

FIG. 4.8 – Si c est un sommet de $\sigma(pq)$ entre $v_{\mathsf{R}}(p)$ et $v_{\mathsf{R}}(q)$ alors c est verticalement visible depuis pq dans R.

Supposons qu'il existe un sommet c de $\sigma(pq)$ entre $v_{\mathsf{R}}(p)$ et $v_{\mathsf{R}}(q)$ tel que c n'est pas verticalement visible depuis pq dans R. Puisque c n'est pas verticalement visible depuis pq, c doit forcément se situer dans l'un des deux intervalles en x induits par les segments $pv_{\mathsf{R}}(p)$ et $qv_{\mathsf{R}}(q)$ (cf. figure 4.8). Supposons, sans perdre en généralité, que c appartienne à l'intervalle en x induit par les segment $pv_{\mathsf{R}}(p)$ et posons i le point d'intersection entre le segment $pv_{\mathsf{R}}(p)$ et la droite verticale L passant par le sommet c.

Puisque c est un sommet concave de R, c est un point à coordonnées entières et L est une droite verticale de la grille. Or ceci mène à une contradiction, puisque dans ce cas il existe un point entier r sur L entre i et c visible depuis p et plus proche de p que $v_{\mathsf{R}}(p)$, ce qui contredit l'hypothèse selon laquelle $v_{\mathsf{R}}(p)$ est le plus proche point entier visible depuis p dans R. □

Description de l'algorithme

Soit A et B deux régions polygonales à coordonnées entières, l'algorithme se déroule en trois étapes. La première calcule le résultat exact R de l'intersection de A et B ainsi que la carte de visibilité $DV(\mathsf{R})$ à l'aide d'un algorithme classique de balayage du plan de type Bentley-Ottmann. À partir des régions A et B, la construction de la carte de visibilité verticale des sommets convexes de R s'obtient en modifiant légèrement l'algorithme de Bentley-Ottmann afin que ce dernier détecte durant le balayage les sommets concaves de R et calcule leur visibilité verticale à l'aide de la structure utilisée pour représenter le front de balayage. À partir de la carte de visibilité verticale de R, la deuxième étape arrondit chacun des sommets de R non représentable sur la grille entière au plus proche point entier de sa cellule convexe (cf. chapitre précédent section 3.7). D'après le lemme 4.2.2, ce point est le plus proche point entier visible à l'intérieur de R. Chaque arête s de R est d'autre part remplacée par une chaîne polygonale $\sigma(s)$ visitant l'ensemble des sommets concaves visibles verticalement depuis s dans l'ordre de leur projection verticale sur s. La dernière étape effectue finalement une variante de l'algorithme de Graham [33] pour le calcul d'enveloppes convexes sur l'ensemble des polygones définissant la région obtenue. Cette procédure a pour effet de supprimer l'ensemble des sommets concaves ne correspondant pas à des sommets concaves originaux, c'est-à-dire, l'ensemble des sommets concaves correspondant soit à un point d'intersection arrondi soit à la visite d'un sommet verticalement visible. Une analyse simple montre que cette dernière étape est linéaire dans le nombre de sommets de chaque polygone.

Théorème 4.2.4 *Étant données deux régions polygonales à coordonnées entières, le calcul de leur intersection selon le mode d'arrondi par défaut peut être effectué en temps* $O((n+k)\log n + km\log m\log N)$*, où* n *désigne le nombre total de sommets des régions en entrée,* $N \times N$ *est la taille de la grille entière contenant ces régions et* k *et* m *désignent respectivement le nombre de paires d'arêtes qui s'intersectent parmi les arêtes des régions en entrée et le nombre de sommets total de l'intersection exacte.*

Démonstration: Étant données A et B les deux régions polygonales en entrée de l'algorithme, la carte de visibilité verticale des sommets concaves de $\mathsf{A} \cap \mathsf{B}$ est un sous ensemble du cloisonnement vertical des arêtes de A et B et peut par conséquent être calculée en temps $O((n+k)\log n)$ où n désigne le nombre total de sommets des deux régions en entrée et k le nombre de paires d'arêtes qui s'intersectent. La deuxième étape calcule au plus k plus proche points entiers visibles dans des cellules convexes de taille au plus égale à m en temps $O(k(m\log m\log N))$ (cf.

section 3.7). Elle produit dans le pire des cas un ensemble de polygones ayant un total de $m+2r$ sommets où r est le nombre de sommets concaves de la région d'intersection exacte (chaque sommet concave étant verticalement visible par au plus deux arêtes). La dernière étape applique finalement un algorithme linéaire sur chacun de ces polygones. En utilisant le fait que $r < m$ et $k \leq m$, on obtient ainsi un temps de calcul en $O((n+k)\log n + km \log m \log N)$ dans le pire des cas pour l'algorithme complet. On remarque enfin que le second terme de la complexité asymptotique de l'algorithme $k(m \log m \log N)$ peut s'exprimer plus exactement sous la forme $\sum_{i=0}^{k}(m_i \log m_i)\log N$ où m_i désignent le nombre de sommets de la cellule convexe contenant le i^{eme} point d'intersection. □

Observation 4.2.5 *L'arrondi par défaut de l'intersection de deux régions polygonales à coordonnées entières peut être calculé en utilisant des primitives numériques avec un degré algébrique au plus égal à quatre.*
Le calcul de l'arrangement des arêtes des deux régions polygonales en entrée nécessite de tester uniquement l'intersection de deux arêtes (degré 2) et la possibilité de trier deux points d'intersection le long d'une arête (degré 4). La construction de la carte de visibilité verticale des sommets concaves de la région d'intersection nécessite de plus la possibilité d'évaluer les primitives $(s \cap l) <_{xy} (s \cap l')$ et $(s \cap s') <_{xy} (s \cap l)$ où s et s' sont des arêtes issues des régions polygonales en entrée et l et l' correspondent à des cloisons de la carte de visibilité. Du fait que les cloisons sont supportées par des droites verticales d'abscisse entière, ces prédicats ont respectivement un degré égal à deux et à quatre. De même, l'arrondi de chaque sommet non représentable de la région d'intersection peut être calculé uniquement à l'aide de primitives de degré inférieur ou égal à quatre (cf. section 3.7).

La suite de la section introduit les lemmes utiles à la preuve des propriétés métriques et topologiques de la région polygonale calculée.

Lemme 4.2.6 *L'ensemble des sommets de la région polygonale calculée sont des points entiers appartenant à la région exacte d'intersection.*

Démonstration: Il existe trois types de sommets dans l'approximation finale : les points d'intersection arrondis, les sommets originaux et les sommets correspondant aux sommets concaves verticalement visibles. Puisque chacun des points d'intersection est arrondi au plus proche point entier visible, le premier type de sommet correspond clairement à des points entiers de l'intersection exacte. Les deux autres types de sommets correspondent à des sommets entiers de l'une des opérandes de l'intersection. □

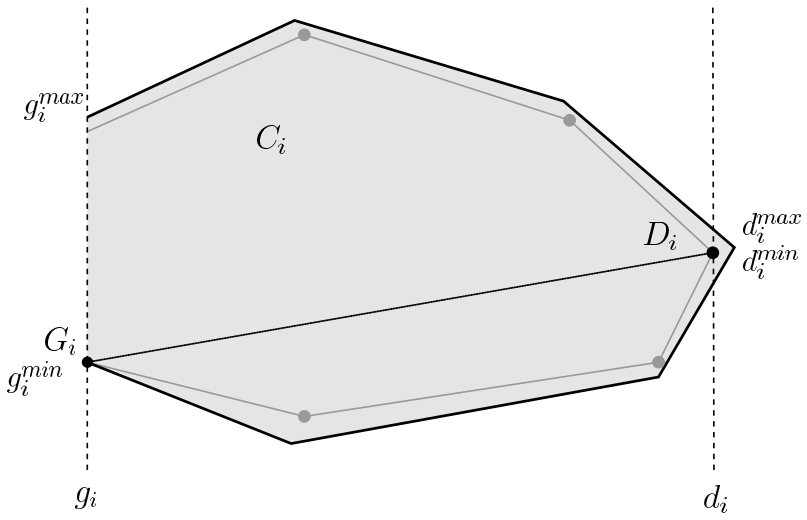

FIG. 4.9 – Les versions arrondies des chaînes polygonales connectant g_i^{min} à d_i^{min} et g_i^{max} à d_i^{max} ont des convexités opposées et ne peuvent s'inverser dans C_i.

Observation 4.2.7 *Étant donnée pq une arête de* R*, la chaîne polygonale $\sigma(pq)$ ayant pour extrémités $v_{\mathsf{R}}(p)$ et $v_{\mathsf{R}}(q)$ et correspondant à la version arrondie de l'arête pq dans* $\underline{\mathsf{R}}$ *est par construction entièrement contenue dans* R.

Lemme 4.2.8 *La région polygonale calculée est une région polygonale valide.*

Démonstration: On démontre qu'aucune intersection n'est introduite dans l'approximation finale (bien que de nouvelles incidences soient permises). Soit $C_i = 0, .., m$ l'ensemble des cellules convexes de la décomposition verticale du résultat exact de l'intersection. Pour chaque C_i, soit g_i et d_i les deux droites verticales passant respectivement par le point entier le plus à gauche et le plus à droite appartenant à C_i (voir. figure 4.9). Dans le cas où l'intersection de C_i avec g_i (resp. d_i) est une cloison de C_i, on note g_i^{min} et g_i^{max} (resp. d_i^{min} et d_i^{max}) les points d'intersection d'ordonnée minimale et d'ordonnée maximale entre g_i et C_i (resp. entre l_i et C_i), et on note G_i (resp. D_i) le sommet concave sur g_i (resp. sur d_i) à partir duquel a été créée la cloison. Sinon, on note $g_i^{min} = g_i^{max}$ (resp. $d_i^{min} = d_i^{max}$) le sommet de C_i le plus à gauche (resp. le plus à droite) de C_i et on note $G_i = v_{\mathsf{R}}(g_i^{min})$ (resp. $D_i = v_{\mathsf{R}}(d_i^{min})$). La version arrondie de la chaîne polygonale connectant g_i^{min} à d_i^{min} (resp. d_i^{max} à g_i^{max}) étant convexe par convexité de la chaîne originale, celle-ci se situe forcément dessous (resp. dessus) l'arête connectant L_i à G_i. Par conséquent, l'ordre vertical des deux chaînes arrondies ne peut être inversé à l'intérieur de C_i. □

Lemme 4.2.9 *Le nombre de sommets distincts de la région polygonale correspondant à l'arrondi*

par défaut $\underline{\mathsf{R}}$ *de la région d'intersection* $\mathsf{R} = \mathsf{A} \cap \mathsf{B}$ *est inférieur ou égal au nombre de sommets de la région d'intersection exacte.*

Démonstration: La démonstration est une conséquence directe du fait que chaque sommet de R non représentable sur la grille entière s'arrondit en au plus un point entier et que chacun des sommets additionnels apparaissant dans $\underline{\mathsf{R}}$ correspond à un sommet concave de l'intersection exacte. □

Les lemmes 4.2.10 et 4.2.11 permettent d'obtenir une borne sur la distance de Hausdorff entre la région $\underline{\mathsf{R}}^C$ et R^C.

Lemme 4.2.10 *Soit* p *un sommet de* R *et* $\mathsf{L}(\mathsf{R})$ *l'union de tous les points de la grille, de tous les segments unitaires de la grille et de tous les carrés unitaires de la grille appartenant à la frontière ou à l'intérieur de* R*. Le segment connectant* p *à* $v_{\mathsf{R}}(p)$ *ne peut intersecter l'intérieur de* $\mathsf{L}(\mathsf{R})$*.*

Démonstration: Afin d'intersecter l'intérieur de $\mathsf{L}(\mathsf{R})$, le segment $pv_{\mathsf{R}}(p)$ doit nécessairement intersecter l'intérieur d'un segment unitaire à coordonnées entières s appartenant à $\partial\mathsf{L}(\mathsf{R})$ (cf. Figure 4.10). Les deux extrémités de s sont plus proches de p que $v_{\mathsf{R}}(p)$ et correspondent, par définition de $\mathsf{L}(\mathsf{R})$, à des points à coordonnées entières appartenant à R, ils ne peuvent par conséquent être visibles depuis p. Ainsi, les segments connectant p aux extrémités de s doivent nécessairement intersecter en leur intérieur la frontière de R. Mais ceci est impossible puisque, par définition, les segments $pv_{\mathsf{R}}(p)$ et s ne peuvent intersecter en leur intérieur $\partial\mathsf{R}$ et il ne peut exister de sommet concave de R visible depuis p dans le triangle ayant p et les deux extrémités de s comme sommets puisque l'ensemble des points de ce triangle sont plus proches de p que $v_{\mathsf{R}}(p)$ et un sommet concave est forcément à coordonnées entières. □

Soit P une région polygonale du plan, on note dans la suite P° l'intérieur de P c'est-à-dire le plus grand ouvert du plan contenu dans P.

Lemme 4.2.11 *Étant données deux régions polygonales à coordonnées entières du plan* A *et* B*,* $d_H((\mathsf{A}\underline{\cap}\mathsf{B})^C, (\mathsf{A}^\circ \cap \mathsf{B}^\circ)^C) < \sqrt{2}$*.*

Démonstration: Par l'observation 4.2.7, on sait que la région $\underline{\mathsf{R}} = \mathsf{A}\underline{\cap}\mathsf{B}$ est incluse dans $\mathsf{R} = \mathsf{A}\cap\mathsf{B}$ et la distance de Hausdorff directionnelle $d_h((\mathsf{R}^\circ)^C, \underline{\mathsf{R}}^C)$ est nulle. On prouve dans la suite que tout les points de $\mathsf{R} \setminus \underline{\mathsf{R}}$ sont à une distance inférieure à $\sqrt{2}$ de $\partial\mathsf{R}$. Soit pq une arête de R et soit E_{pq} le polygone ayant les sommets de la chaîne arrondie $\sigma(pq)$, q et p comme sommets. On

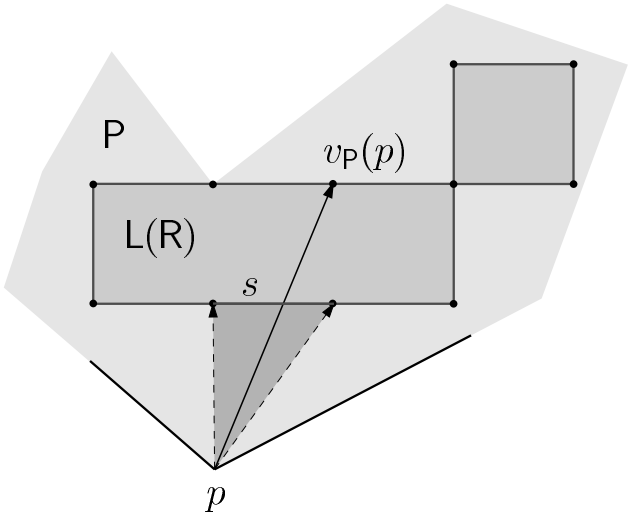

FIG. 4.10 – Le segment $pv_{\mathsf{R}}(p)$ ne peut intersecter l'intérieur de $\mathsf{L}(\mathsf{R})$.

remarque que $(\bigcup_{pq\in\mathsf{R}} E_{pq})$ partitionne $\mathsf{R}\setminus\underline{\mathsf{R}}$ excepté pour les polygones R_i de R tels que $\underline{\mathsf{R}_i}=\emptyset$. La preuve de la borne est triviale pour ces polygones puisque dans ce cas $\mathsf{R}_i\cap\mathbb{Z}^2=\emptyset$. Il est par conséquent suffisant de montrer que $d_h(E_{pq},\partial\mathsf{R})<\sqrt{2}$ pour toute arête pq de R.

Soit $\mathsf{L}(\mathsf{R})$ l'union de tous les points de la grille, de tous les segments unitaires de la grille et de tous les carrés unitaires de la grille appartenant à la frontière ou à l'intérieur de R. Par définition de $\mathsf{L}(\mathsf{R})$, l'ensemble des points de $\mathsf{R}\setminus\mathsf{L}(\mathsf{R})^\circ$ est à une distance inférieure à $\sqrt{2}$ de R, on suppose par conséquent dans la suite que E_{pq} n'est pas entièrement inclus dans $\mathsf{R}\setminus\mathsf{L}(\mathsf{R})^\circ$.

Par le lemme 4.2.10, on sait que les segments $pv_{\mathsf{R}}(p)$ et $qv_{\mathsf{R}}(q)$ ne peuvent intersecter l'intérieur de $\mathsf{L}(\mathsf{R})$, ainsi, afin d'intersecter $\mathsf{L}(\mathsf{R})^\circ$, $\sigma(pq)$ doit intersecter $\mathsf{L}(\mathsf{R})^\circ$. De plus, par convexité de la chaîne $\sigma(pq)$, il doit exister dans ce cas au moins un point de la grille entière différent de $v_{\mathsf{R}}(p)$ et de $v_{\mathsf{R}}(q)$ appartenant à l'intérieur ou à la frontière de E_{pq}. Supposons sans perdre en généralité que l'arête pq est orientée de gauche à droite avec une pente positive ou nulle et que le polygone E_{pq} se trouve au dessus de pq. Enfin, soit v_l le sommet entier minimal selon l'ordre xy lexicographique qui appartient à l'ensemble S de tous les points de la grille, différents de $v_{\mathsf{R}}(p)$ et de $v_{\mathsf{R}}(q)$, qui se trouvent à l'intérieur ou sur la frontière de E_{pq} (cf. figure 4.11).

En utilisant les mêmes arguments que ceux utilisés pour la preuve du lemme 4.2.3 et puisque v_l est le point minimal de S selon l'ordre xy-lexicographique, on montre que v_l est verticalement visible depuis pq et que le segment unitaire vertical de la grille ayant v_l comme sommet d'ordonnée maximale intersecte pq en un point i_p. De manière similaire, puisque l'arête pq est orientée de gauche à droite avec une pente positive ou nulle et puisque par le lemme 4.2.10 les segments

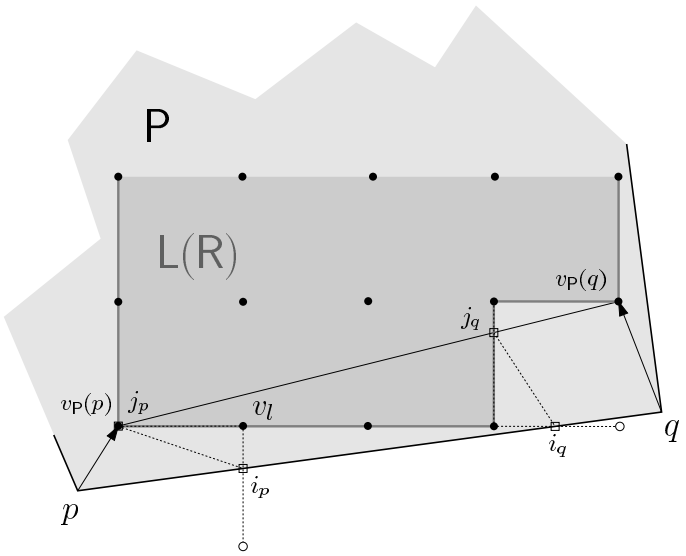

FIG. 4.11 – Les polygones $pi_pj_pv_{\mathsf{R}}(p)$ et $j_qi_qqv_{\mathsf{R}}(q)$ sont contenus dans $\mathsf{R} \setminus \mathsf{L}(\mathsf{R})^\circ$ et le polygone $i_pi_qj_qj_p$ est à une distance inférieure à $\sqrt{2}$ de pq.

$pv_{\mathsf{R}}(p)$ et $qv_{\mathsf{R}}(q)$ ne peuvent intersecter l'intérieur de $\mathsf{L}(\mathsf{R})$, le segment unitaire horizontal de la grille ayant v_l comme extrémité d'abscisse maximale intersecte $\sigma(pq)$ en un point j_p. On note que, par construction, les points i_p et j_p appartiennent à la frontière d'un même carré unitaire de la grille et sont par conséquent tels que $||i_pj_p|| < \sqrt{2}$.

En remplaçant p par q et en appliquant une opération de symétrie sur E_{pq} afin que l'arête qp soit orientée de gauche à droite avec une pente positive ou nulle et telle que E_{pq} soit au dessus de qp, on définit de manière similaire deux points i_q et j_q sur pq et sur $\sigma(pq)$ tels que $||i_qj_q|| < \sqrt{2}$. On conclut que $d_h(E_{pq}, \partial\mathsf{R}) < \sqrt{2}$ puisque les polygones $pi_pj_pv_{\mathsf{R}}(p)$ et $j_qi_qqv_{\mathsf{R}}(q)$ sont contenus dans $\mathsf{R} \setminus \mathsf{L}(\mathsf{R})^\circ$ (par définition de v_l) et le polygone $i_pi_qj_qj_p$ est contenu dans la somme de Minkowski de i_pi_q avec l'intérieur d'un disque de rayon $\sqrt{2}$ centré à l'origine (cf. figure 4.11), de plus, par convexité de $\sigma(pq)$, la portion de $\sigma(pq)$ entre j_p et j_q est incluse dans $i_pi_qj_qj_p$. □

4.2.4 Intersection arrondie par excès

Principe de l'arrondi et description de l'algorithme

Étant données deux régions polygonales valides à coordonnées entières (cf. figure 4.12.a), le processus menant au calcul de la région d'intersection de ces régions selon le mode d'arrondi par excès peut être découpé en trois étapes. Son principe consiste à ramener le problème initial

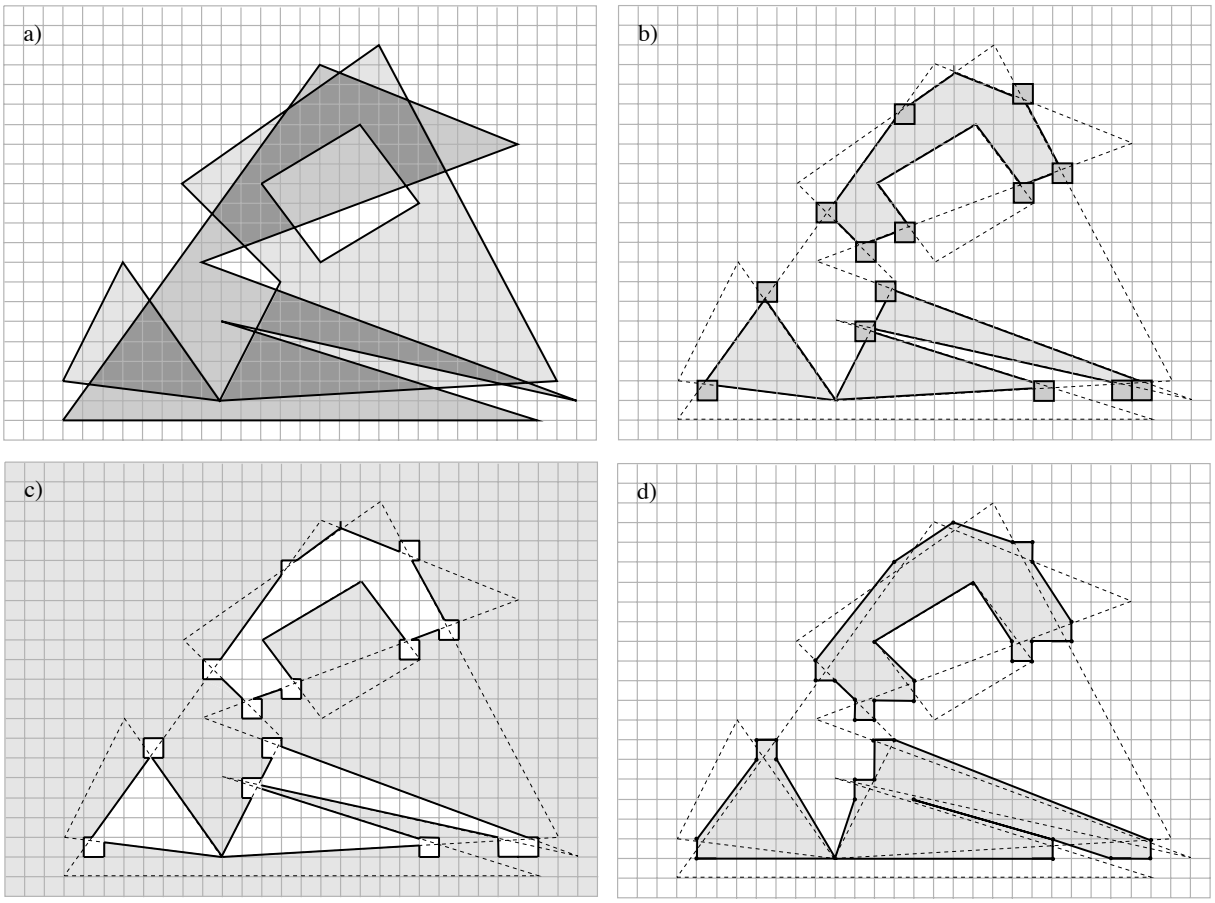

FIG. 4.12 – Intersection arrondie par excès. a) Les deux régions polygonales en entrée. b) Calcul de la région d'intersection R et de l'ensemble I des quadrilatères contenant chaque point d'intersection non représentable sur la grille, c) La région $(\mathsf{R}^C \cap I^C)$, d) La version arrondie par excès $\overline{\mathsf{R}}$ résultant du calcul de $(\mathsf{R}^C \underline{\cap} I^C)^C$.

à un calcul d'intersection selon le mode d'arrondi par défaut défini à la section précédente. Pour cela, l'intersection exacte R des deux régions en entrée est tout d'abord calculée. À chacun des sommets $v = (v_x, v_y)$ de R non représentable sur la grille entière est ensuite associé le quadrilatère, aligné sur les axes, ayant respectivement $(\lfloor v_x \rfloor, \lfloor v_y \rfloor)$ et $(\lceil v_x \rceil, \lceil v_y \rceil)$ comme coin inférieur gauche et coin supérieur droit (cf. figure 4.12.b). La version arrondie $\overline{\mathsf{R}}$ selon le mode par excès de la région R est finalement obtenue à partir de R et de l'ensemble I de ces quadrilatères contenant les points non représentables (cf. figure 4.12.c) en effectuant l'opération $\left((\mathsf{R}^C) \underline{\cap} (I^C)\right)^C$ (cf. figure 4.12.d).

Théorème 4.2.12 *Étant données deux régions polygonales* A *et* B *à coordonnées entières, le*

calcul de leur intersection selon le mode d'arrondi par excès peut être effectué en temps $O((n+h)\log n + kp\log p\log N)$*, où* n *désigne le nombre total de sommets des régions en entrée,* h *désigne le nombre de paires d'arêtes qui s'intersectent parmi les arêtes des régions en entrée,* $N \times N$ *est la taille de la grille entière contenant ces régions,* k *désigne le nombre d'intersection entre les arêtes de* $\mathsf{R} = \mathsf{A} \cap \mathsf{B}$ *et les arêtes de* I*, et* $p = \max(|\mathsf{R}|, |\mathsf{R}^C \cap I^C|)$.

Démonstration: Le calcul de la région d'intersection R se fait en temps $O((n+h)\log n)$ où n désigne le nombre total de sommets des régions A et B en entrée et h le nombre d'intersections entre les arêtes de A et les arêtes B. Le calcul de la carte de visibilité des sommets concaves de $(\mathsf{R}^C \cap I^C)$ se fait en temps $O((|\mathsf{R}|+h)\log(|\mathsf{R}|+h)+k)$ où k est le nombre d'intersection entre les arêtes de R et les arêtes I. Enfin, l'algorithme calcule au plus k plus proche points entiers visibles dans des cellules convexes de taille au plus égale à $m = |\mathsf{R}^C \cap I^C|$ en temps $O(km\log m\log N)$ (cf. section 3.7) et produit un ensemble de polygones ayant un total de $O(m)$ sommets. La dernière étape applique finalement un algorithme linéaire sur chacun de ces polygones. En utilisant le fait que $h \leq |\mathsf{R}|$ et en posant $p = \max(|\mathsf{R}|, |\mathsf{R}^C \cap I^C|)$, on obtient ainsi un temps de calcul en $O((n+h)\log n + kp\log p\log N)$ dans le pire des cas pour l'algorithme complet. □

Observation 4.2.13 *L'arrondi par excès de l'intersection de deux régions polygonales à coordonnées entières peut être calculé en utilisant des primitives numériques ayant un degré algébrique au plus égal à quatre.*

On utilise pour vérifier cette affirmation des arguments similaires à l'observation 4.2.5.

Le lemme suivant permet de borner la distance de Hausdorff entre la région d'intersection exacte et sa version arrondie. On note que la borne sur l'erreur géométrique énoncée page 106 ne peut être déduite directement à partir du lemme 4.2.11 puisqu'il est nécessaire pour cet arrondi de borner la distance entre les points de $\overline{\mathsf{R}}$ et les points de la région l'intersection exacte et non uniquement la distance entre la région $\overline{\mathsf{R}}$ et la région $(\mathsf{P}^C \cap I^C)^C$. En d'autres termes, nous devons prouver qu'il ne peut exister de point de $\overline{\mathsf{R}}$ proche d'un quadrilatère de I et à une distance supérieure à $\sqrt{2}$ de la région R.

Lemme 4.2.14 *Étant données deux régions polygonales à coordonnées entières du plan* A *et* B*,* $d_H((\mathsf{A}\overline{\cap}\mathsf{B}), (\mathsf{A} \cap \mathsf{B})) < \sqrt{2}$.

Démonstration: Puisque la région $\mathsf{R} = \mathsf{A} \cap \mathsf{B}$ est incluse dans $\overline{\mathsf{R}} = \mathsf{A}\overline{\cap}\mathsf{B}$, la distance de Hausdorff directionnelle de R à $\overline{\mathsf{R}}$ est nulle. Par conséquent, il est suffisant de montrer que tout point

de $\overline{\mathsf{R}} \setminus \mathsf{R}$ est à une distance inférieure à $\sqrt{2}$ de R. On remarque que puisque les quadrilatères de I ne contiennent aucun point entier en leur intérieur, l'union de tous les points de la grille, de tous les segments unitaires de la grille et de tous les carrés unitaires de la grille appartenant à la frontière ou à l'intérieur de $(\mathsf{R}^C \cap \mathsf{I}^C)$ correspond à l'union de tous les points de la grille, de tous les segments unitaires de la grille et de tous les carrés unitaires de la grille appartenant à la frontière ou à l'intérieur de R^C. Ainsi, si le polygone E_{pq} (comme défini dans la preuve du lemme 4.2.11) est inclus dans $(\mathsf{R}^C \cap \mathsf{I}^C) \setminus \mathsf{L}(\mathsf{R}^C \cap \mathsf{I}^C)^\circ$, E_{pq} est aussi inclus dans $\mathsf{R}^C \setminus \mathsf{L}(\mathsf{R}^C)^\circ$, et l'ensemble des points de E_{pq} est à une distance inférieure à $\sqrt{2}$ de la frontière de R. Sinon, en utilisant les mêmes arguments que pour la preuve du lemme 4.2.11, on montre que la partie de E_{pq} qui intersecte l'intérieur de $\mathsf{L}(\mathsf{R}^C)^\circ$ se trouve à une distance inférieure à $\sqrt{2}$ de pq. Par conséquent, si l'arête pq est issue d'une arête de la région R alors la preuve de la borne est directe. Dans le cas opposé, c'est-à-dire si l'arête pq est issue d'un pixel Q de I, et en utilisant les mêmes arguments que pour la preuve du lemme 4.2.11, on montre qu'il doit exister une droite de la grille passant par v_l (comme défini dans la preuve) qui intersecte pq. Cette droite ne peut intersecter l'arête pq en son intérieur puisque pq est incluse dans un segment unitaire de la grille. De plus, si la droite intersecte pq en une de ses extrémités alors le point d'intersection est forcément un point de la grille et par conséquent la version arrondi $\sigma(pq)$ de pq est incluse dans pq (cf. Figure 4.13). On conclut dans ce cas que $d_h(E_{pq}, \partial\mathsf{R}) < \sqrt{2}$ puisqu'il existe un sommet de R dans le quadrilatère Q (à savoir le sommet de R qui a causé la présence de Q dans I). □

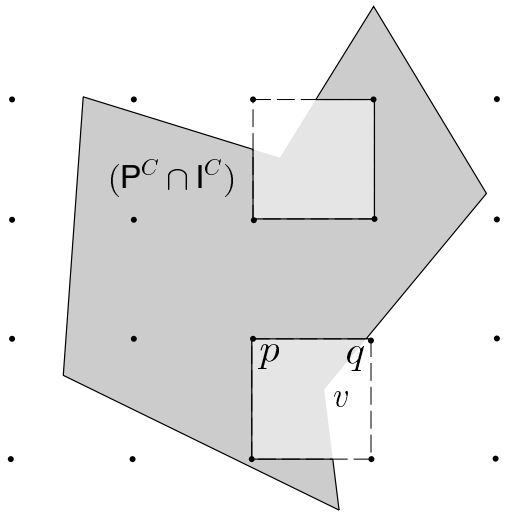

FIG. 4.13 – Si l'arête pq est issue d'une arête de I et au moins une de ses extrémités est un point de la grille alors $\sigma(pq) \subseteq pq$ et $\sigma(pq)$ est à une distance inférieure à $\sqrt{2}$ du sommet v de R.

Lemme 4.2.15 *Le nombre de sommets distincts de la région polygonale correspondant à l'arrondi par excès* $\overline{\mathsf{R}}$ *de la région d'intersection* $\mathsf{R} = \mathsf{A} \cap \mathsf{B}$ *est inférieur à* $|\mathsf{R}| + 3k + h$ *où* k *est le nombre de sommets de* R *non représentables sur la grille entière et* h *est le nombre de points d'intersection entre les arêtes de* R *et les arêtes de* I *non représentables sur la grille entière.*

Démonstration: Puisque la région $\overline{\mathsf{R}}$ est égale au complémentaire de l'arrondi par défaut de la région $\mathsf{R}_I = (\mathsf{R}^C \cap I^C)$ et $|\underline{\mathsf{R}_I}| \leq |\mathsf{R}_I|$ (cf. théorème 4.2.16), le nombre de sommets de $\overline{\mathsf{R}}$ est borné par $|\mathsf{R}_I|$. Or si R possède n sommets dont k sont non représentables sur la grille entière, $|I| \leq 4k$ et la région R_I possède au plus $(n-k)+4k$ sommets entiers et h sommets non entiers où h désigne le nombre de points d'intersection non représentables sur la grille entière entre les arêtes de R et les arêtes de I, d'où $|\mathsf{R}| + 3k + h$. □

Remarque : Bien que la complexité de la région R_I utilisée comme borne de la complexité de $\overline{\mathsf{R}}$ soit dans le pire des cas $O(nk)$, il s'avère que la complexité de la version arrondie par excès de R peut être contrôlée afin de garantir un nombre de sommets linéaire en le nombre de sommets de la région d'intersection exacte.

La garantie d'une telle borne se base sur les observations suivantes. Soient s_i et s_j deux arêtes de R connectant chacune deux quadrilatères Q et Q' de I et soit P le polygone de $\mathsf{R}^C \cap I^C$ compris entre les arêtes s_i et s_j et inclus dans l'enveloppe convexe des sommets de Q et Q', la version arrondie de P apparaissant dans $\overline{\mathsf{R}}$ (correspondant à l'arrondi par défaut de P) possède au plus deux sommets distincts et a donc une épaisseur nulle (cf. figure 4.14).

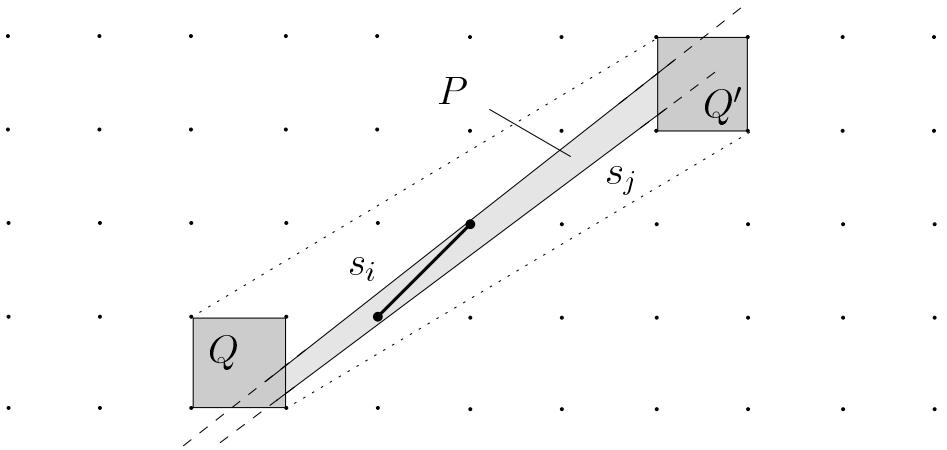

FIG. 4.14 – L'arrondi par défaut du polygone P a au plus deux sommets distincts et une épaisseur nulle.

Par conséquent, l'arrondi de P peut être retiré de $\overline{\mathsf{R}}$ sans modifier les garanties métriques (et d'inclusion) de $\overline{\mathsf{R}}$ relativement à R. Le nombre de sommets de la région arrondie obtenue en supprimant de $\overline{\mathsf{R}}$ l'ensemble des composantes ayant moins de trois sommets distincts et correspondant à des polygones de $\mathsf{R}^C \cap I^C$ peut alors être borné par $(n-k)+4k+(n-k)+3k$. Il existe en effet au plus $(n-k)$ arêtes (orientées) dans R_I possédant une origine à coordonnées entières et incidentes à un quadrilatère de I et au plus $3k$ arêtes dans R_I qui connectent des paires de quadrilatères de I distinctes (par la relation d'Euler, on sait en effet qu'un graphe planaire ayant k sommets possède au plus $3k-6$ arêtes distinctes si $k > 2$). En remarquant, de plus, qu'au moins deux sommets de chaque quadrilatère Q de I sont comptés soit en trop (si le sommet de Q est inclus dans R) soit deux fois (si l'intersection d'une arête de R avec un arête de Q s'arrondit en un sommet de Q ou si plus d'un sommet non entier de R est inclus dans Q), on obtient finalement, $|\overline{\mathsf{R}}| < (n-k)+4k+(n-k)+3k-2k = 2n+3k$.

Contrairement au mode d'arrondi par défaut, le mode d'arrondi par excès n'offre aucune garantie sur la préservation de la convexité (ou de la concavité) des sommets de la région d'intersection exacte. En effet, certains sommets concaves de R peuvent disparaître dans $\overline{\mathsf{R}}$, de même de nouveaux sommets concaves (correspondant à des sommets de I et donc n'apparaissant pas dans R) peuvent être ajoutés dans $\overline{\mathsf{R}}$. Alors que la perte de la concavité des sommets et de la topologie originale du résultat (par fusion de composantes connexes distinctes dans R) est intrinsèquement liée au problème et est inévitable dans certains cas, l'apparition de sommets concaves superflus dans $\overline{\mathsf{R}}$ est strictement liée à la manière dont l'arrondi $\overline{\mathsf{R}}$ est construit et donc imputable à l'algorithme utilisé.

On propose cependant deux améliorations envisageables de l'algorithme afin de réduire le nombre de sommets concaves additionnels apparaissant dans $\overline{\mathsf{R}}$.

Une première option consiste à remplacer les quadrilatères de I par des triangles (cf. figure 4.15.b), c'est-à-dire à ne plus utiliser pour chaque sommet (v_x, v_y) non entier de R le quadrilatère Q de sommets $(\lfloor v_x \rfloor, \lfloor v_y \rfloor)$, $(\lceil v_x \rceil, \lfloor v_y \rfloor)$, $(\lceil v_x \rceil, \lceil v_y \rceil)$ et $(\lfloor v_x \rfloor, \lceil v_y \rceil)$, mais l'un des convexes formé par au plus trois sommets de Q contenant le point (v_x, v_y) (par exemple le triangle $T = ((\lfloor v_x \rfloor, \lfloor v_y \rfloor), (\lceil v_x \rceil, \lfloor v_y \rfloor), (\lceil v_x \rceil, \lceil v_y \rceil))$ si la partie fractionnaire de v_x est supérieure ou égale à celle de v_y). Cette solution présente un avantage évident par rapport à l'utilisation de carrés unitaires (ou de rectangles d'épaisseur nulle si v_x ou v_y est un entier) puisqu'elle réduit d'un facteur k le nombre de sommets entiers potentiellement ajoutés lors du passage de R à R_I. Elle garantit ainsi une réduction des sommets concaves superflus apparaissant dans $\overline{\mathsf{R}}$ dans le pire des cas, tout en préservant les mêmes bornes métriques que celles obtenues avec l'utilisation de

quadrilatères. Cette solution présente cependant un inconvénient vis à vis de l'algorithme proposé puisqu'on ne garantit plus que $\overline{\mathsf{R}} = \overline{\mathsf{R}^t}^t$ si R^t désigne le résultat d'une opération de symétrie sur R par rapport à une droite ou à un point assurant une bijection entre les points de la grille.

Une deuxième amélioration envisageable consiste à relâcher les sommets concaves de $\overline{\mathsf{R}}$ n'ayant aucun correspondant dans R si leur relâchement ne provoque pas de changement de topologie par rapport à $\overline{\mathsf{R}}$ (*i.e.* la région polygonale obtenue après relâchement du sommet ne s'auto-intersecte pas). Là encore cette solution permet une réduction du nombre de sommets superflus de $\overline{\mathsf{R}}$ d'un facteur $O(k)$ dans le meilleur des cas, toutefois quelques précautions sont à prendre afin de conserver une distance maximale entre les points de $\overline{\mathsf{R}}$ et de R inférieure à $\sqrt{2}$. Plus précisément, pour conserver cette borne seuls les sommets concaves de $\overline{\mathsf{R}}$ apparaissant entre deux sommets v et v' tels que v et v' se situent tout les deux à une distance inférieure à $\sqrt{2}$ d'une arête s de R pourront être supprimer. D'autre part, comme pour la modification précédente, une certaine forme de canonicité est perdue puisque pour une région $\overline{\mathsf{R}}$ donnée, si X désigne l'ensemble des sommets concaves superflus pouvant être supprimés de $\overline{\mathsf{R}}$ sans modifier la borne métrique, la suppression d'un sommet donné peut dépendre de l'ordre de traitement des éléments de l'ensemble X.

Afin d'éviter ce problème, on peut envisager de relâcher tout les sommets concaves superflus de $\overline{\mathsf{R}}$ à condition que leur relâchement ne crée pas d'erreur géométrique supérieure à $\sqrt{2}$ (ceci même si la condition de planarité est violée), et de remplacer les arêtes vv' qui créant une auto-intersection dans la représentation obtenue par la plus courte chaîne polygonale entièrement incluse dans $(\mathsf{R}^C \cap I^C)$ connectant v à v' (cf. figure 4.15.c et figure 4.15.d). On note qu'une telle chaîne peut être construite de manière efficace à partir de la carte de visibilité des sommets concaves de $(\mathsf{R}^C \cap I^C)$ utilisée pour construire $\overline{\mathsf{R}}$.

4.2.5 Propriétés des opérations booléennes arrondies

Cette dernière section reprend l'ensemble des propriétés vérifiées par l'arrondi d'intersections et de réunions de régions polygonales à coordonnées entières du plan pour chacun des modes.

Les théorèmes 4.2.16 et 4.2.17 énumèrent l'ensemble des propriétés garanties par l'arrondi par défaut de l'intersection et l'arrondi par excès de la réunion de deux régions polygonales à coordonnées entières.

Théorème 4.2.16 *Soit* $\underline{\mathsf{R}}$ *la version arrondie par défaut de l'intersection* R *de régions polygonales à coordonnées entières,* $\underline{\mathsf{R}}$ *vérifie les propriétés suivantes :*

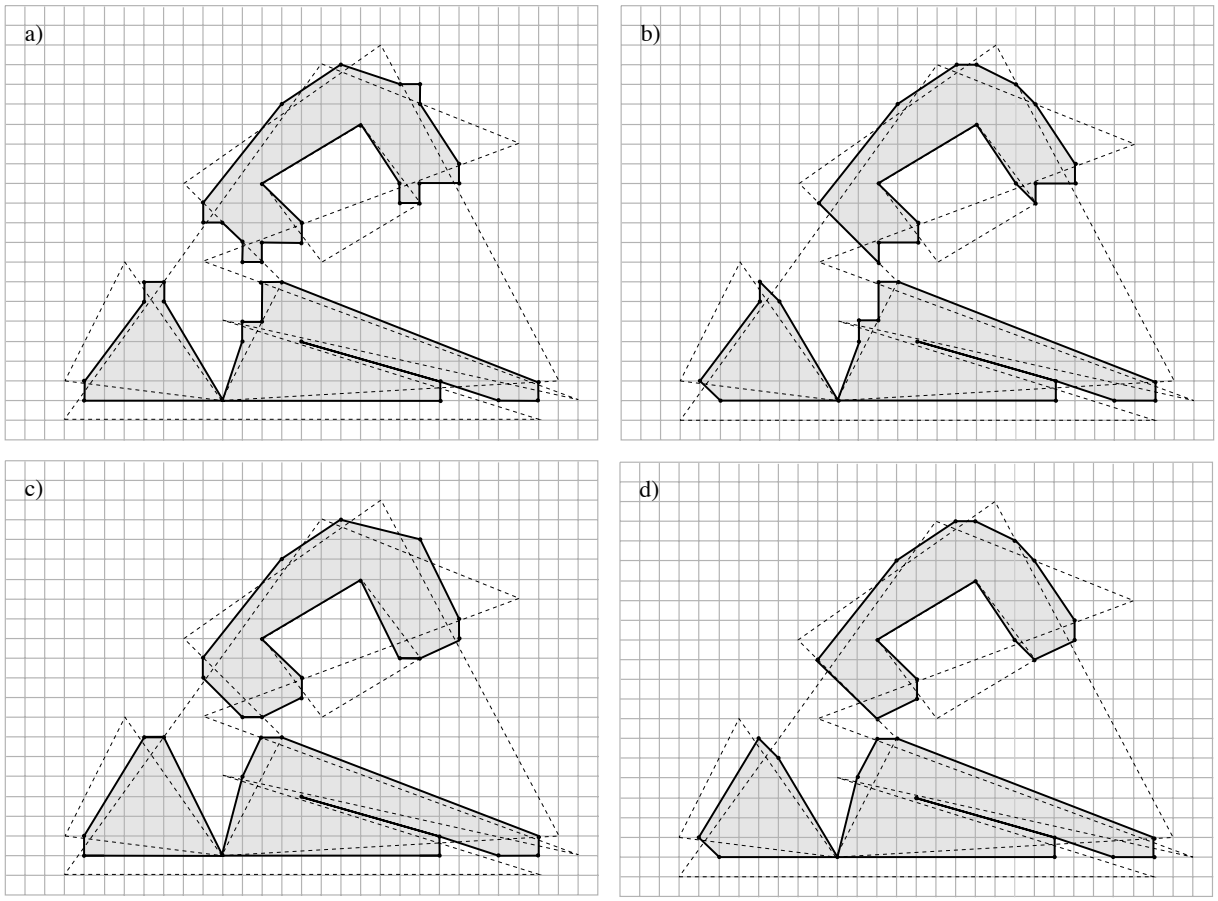

FIG. 4.15 – a) Arrondi avec des quadrilatères. b) Arrondi avec des triangles. c) Arrondi avec des quadrilatères après relâchement des sommets concaves superflus. d) Arrondi avec des triangles après relâchement des sommets concaves superflus.

1. $\underline{\mathsf{R}}$ *est une région polygonale valide à coordonnées entières,*
2. $\underline{\mathsf{R}}$ *est inclus dans* R*,*
3. $d_H(\underline{\mathsf{R}}^C, \mathsf{R}^{\circ\, C}) < \sqrt{2}$,
4. $|\underline{\mathsf{R}}| \leq |\mathsf{R}|$,
5. *Un sommet concave de* $\underline{\mathsf{R}}$ *a toujours un correspondant concave dans* R*.*

La démonstration de la propriété 1 est une conséquence directe des lemmes 4.2.6 et 4.2.8. À partir de cette propriété et par construction de $\underline{\mathsf{R}}$ la démonstration de la propriété 2 est triviale. La propriété 3 est démontrée par le lemme 4.2.11. La propriété 4 est démontrée par le lemme 4.2.9. La démonstration de la propriété 5 est une conséquence directe de la dernière étape de

l'algorithme.

À partir du lemme 4.2.11 et par passage au complémentaire, on obtient le théorème suivant :

Théorème 4.2.17 *Soit* $\overline{\mathsf{U}}$ *la version arrondie par excès de la réunion* U *de régions polygonales à coordonnées entières,* $\overline{\mathsf{U}}$ *vérifie les propriétés suivantes :*

1. $\overline{\mathsf{U}}$ *est une région polygonale valide à coordonnées entières,*
2. $\overline{\mathsf{U}}$ *contient* U*,*
3. $d_H(\,\overline{\mathsf{U}}\,,\,\mathsf{U}) < \sqrt{2}$,
4. $|\overline{\mathsf{U}}| \leq |\mathsf{U}|$,
5. *Un sommet convexe de* $\overline{\mathsf{U}}$ *a toujours un correspondant convexe dans* U.

Les deux derniers lemmes énoncent enfin l'ensemble des propriétés garanties par l'arrondi par excès de l'intersection et l'arrondi par défaut de la réunion de deux régions polygonales à coordonnées entières.

Théorème 4.2.18 *Soit* $\overline{\mathsf{R}}$ *la version arrondie par excès de l'intersection* R *de régions polygonales à coordonnées entières,* $\overline{\mathsf{R}}$ *vérifie les propriétés suivantes :*

1. $\overline{\mathsf{R}}$ *est une région polygonale valide à coordonnées entières,*
2. $\overline{\mathsf{R}}$ *contient* R*,*
3. $d_H(\,\overline{\mathsf{R}}\,,\,\mathsf{R}\,) < \sqrt{2}$,
4. $|\overline{\mathsf{R}}| < |\mathsf{R}| + k + h$, *où* k *est le nombre de sommets de* R *non représentables sur la grille entière et* h *désigne le nombre d'intersections entre les arêtes de* R *et les arêtes de* I.

Les démonstrations des propriétés 1 et 2 sont une conséquence directe du théorème 4.2.16 puisque $\overline{\mathsf{R}} = (\underline{(\mathsf{R}_I)^C})^C$ et les seuls sommets non entiers de $(\mathsf{R}_I)^C$ correspondent à des sommets convexes de $(\mathsf{R}_I)^C$. La propriété 3 est démontrée par le lemme 4.2.14. La démonstration de la propriété 4 est démontrée par le lemme 4.2.15.

Par passage au complémentaire on obtient enfin :

Théorème 4.2.19 *Soit* $\underline{\mathsf{U}}$ *la version arrondie par défaut de la réunion* U *de régions polygonales à coordonnées entières,* $\underline{\mathsf{U}}$ *vérifie les propriétés suivantes :*

1. $\underline{\mathsf{U}}$ *est une région polygonale valide à coordonnées entières,*
2. $\underline{\mathsf{U}}$ *est inclus dans* U*,*
3. $d_H(\ \underline{\mathsf{U}}^{\,C},\ \mathsf{U}^{\circ\,C}) < \sqrt{2}$*,*
4. $|\underline{\mathsf{U}}| < |\mathsf{U}| + k + h$*, où* k *est le nombre de sommets de* R *non représentables sur la grille entière et* h *désigne le nombre d'intersections entre les arêtes de* R *et les arêtes de* I*.*

4.2.6 Généralisation au cas de régions polygonales quelconques

À partir des lemmes et des algorithmes présentés dans cette section, le calcul d'un arrondi par excès et par défaut de régions polygonales quelconques (c'est-à-dire dont on ne dispose d'aucune hypothèse sur la représentation des sommets, par exemple, une région polygonale issue d'une opération de rotation) peut être obtenu de la manière suivante.

Soit P une région polygonale quelconque, considérons S_c (respectivement S_r) l'ensemble des sommets convexes (respectivement concaves) de P non représentables sur la grille entière. Soit I_c (respectivement I_r) l'ensemble des quadrilatères isothétiques unitaires à coordonnées entières contenant l'ensemble des sommets de S_c (respectivement de S_r), c'est-à-dire l'ensemble des quadrilatères ayant $(\lfloor v_x \rfloor, \lfloor v_y \rfloor)$ et $(\lceil v_x \rceil, \lceil v_y \rceil)$ comme coin inférieur gauche et coin supérieur droit où $v = (v_x, v_y)$ est un sommet de I_c (respectivement de I_r). On définit alors l'arrondi par défaut $\underline{\mathsf{P}}$ de P comme le résultat du calcul de l'intersection par défaut de P et I_r, c'est-à-dire $\underline{\mathsf{P}} = (\mathsf{P} \underline{\cap}\ I_r)$. Similairement, on définit l'arrondi par excès $\overline{\mathsf{P}}$ de P comme le complémentaire du résultat du calcul de l'intersection par défaut de P^C et ${I_c}^C$, c'est-à-dire $\overline{\mathsf{P}} = (\mathsf{P}^C \underline{\cap}\ {I_c}^C)^C$.

Les opérations d'intersection arrondies par défaut intervenant dans ces définitions se calculent directement à partir de l'algorithme décrit à la section 4.2.3. On note en effet que la totalité des sommets concaves des régions d'intersection $(\mathsf{P} \cap I_r)$ et $(\mathsf{P}^C \cap {I_c}^C)$ sont à coordonnées entières, condition qui suffit à la vérification des propriétés de la carte de visibilité verticale énoncées par les lemmes 4.2.2 et 4.2.3, et donc à la validité de l'algorithme. On remarque cependant que l'absence de segment support à coordonnées entières aux arêtes de P nécessite selon le cas l'utilisation d'un type de nombre et d'une arithmétique adaptés afin d'évaluer les prédicats et les constructeurs intervenant dans l'algorithme. Un type de nombre et une arithmétique permettant la manipulation de nombres algébriques peuvent par exemple être nécessaires si la région originale P est issue d'opérations de rotation.

Les propriétés vérifiées par les régions $\underline{\mathsf{P}}$ et $\overline{\mathsf{P}}$ sont à quelques exceptions près similaires à celles énoncées par les théorèmes 4.2.16 et 4.2.18 et se déduisent directement des lemmes de la section. Précisément, la région arrondi par défaut $\underline{\mathsf{P}}$ est une région polygonales à coordonnées

entières valide contenu dans P telle que $d_H(\underline{\mathsf{P}}^C, \mathsf{P}^{\circ\,C}) < \sqrt{2}$. De plus, on garantit que le correspondant arrondi d'un sommet convexe de P est (si il existe) un sommet convexe de $\underline{\mathsf{P}}$. Enfin, on a $|\underline{\mathsf{P}}| < |\mathsf{P}| + r + h_r$ où r est le nombre de sommets concaves de P et h_r est le nombre d'intersections entre les arêtes de P et les arêtes de I_r. La région arrondi par excès $\overline{\mathsf{P}}$ est une région polygonale valide à coordonnées entières contenant la région P telle que $d_H(\,\overline{\mathsf{P}}\,,\,\mathsf{P}\,) < \sqrt{2}$ et $|\overline{\mathsf{P}}| < |\mathsf{P}| + c + h_c$ où c désigne le nombre de sommets convexes P et h_c le nombre d'intersections entre les arêtes de P et les arêtes de I_c.

On remarque finalement que le nombre de sommets nécessaires à la représentation de la région $\overline{\mathsf{P}}$ peut être réduit en appliquant le même type de technique que celles proposées à la fin de la section 4.2.4. L'application de la dernière solution présentée garantit, par exemple, que $\overline{\mathsf{P}}$ est convexe et $|\overline{\mathsf{P}}| < |\mathsf{P}| + 4$ si P se réduit à un unique polygone convexe.

Conclusion

Les travaux de cette thèse ont porté sur la recherche de méthodes pour résoudre les problèmes de robustesse posés par la construction d'objets géométriques. Plus particulièrement, nous avons abordé le problème de l'arrondi sur la grille entière du résultat d'opérations booléennes sur des régions polygonales du plan. Nous avons ainsi présenté différentes notions d'arrondi de régions polygonales, telles que l'arrondi *par défaut* et l'arrondi *par excès*, offrant des garanties sur la qualité de l'arrondi. Plus spécifiquement, étant donné P le résultat d'une opération d'intersection ou de réunion sur des régions polygonales à coordonnées entières, nous avons défini un arrondi par défaut $\underline{\mathsf{P}}$ et un arrondi par excès $\overline{\mathsf{P}}$ de la région P ayant l'ensemble de leurs sommets sur la grille entière et offrant, en plus de la conservation de la planarité du plongement géométrique et de l'inclusion de $\underline{\mathsf{P}}$ dans P (respectivement de l'inclusion de P dans $\overline{\mathsf{P}}$), des garanties sur le nombre de sommets nécessaires à leur représentation, ainsi que des garanties sur la proximité, en termes de distance de Hausdorff, de $\underline{\mathsf{P}}$ et $\overline{\mathsf{P}}$ relativement à P. Pour un de ces modes d'arrondi, nous avons de plus la garantie de la préservation de la convexité possible de la région P. D'autre part, nous avons proposé deux algorithmes pratiques pour calculer ces arrondis, utilisant des primitives numériques ayant un degré algébrique au plus égal à quatre et ayant chacun une complexité asymptotique raisonnable. Enfin, nous avons montré comment ces méthodes de calcul pouvaient se généraliser au cas de régions polygonales simples à coordonnées réelles, c'est-à-dire à des régions qui ne résultent pas forcément d'opérations booléennes.

Par la définition de ces modes d'arrondi et l'élaboration d'algorithmes permettant leur calcul, nous autorisons ainsi, en plus de l'obtention d'une représentation à précision fixée cohérente de constructions géométriques dans le plan, la détermination d'intervalles géométriques encadrant selon une relation d'inclusion le résultat exact de constructions. Cette isolation du résultat dans un intervalle permet notamment de cascader diverses constructions géométriques telles que les opérations booléennes, le calcul d'enveloppes convexes, les rotations, etc... Ce type de résultat est en un certain sens similaire à l'arithmétique d'intervalle offert par l'arithmétique IEEE grâce à l'utilisation de l'arrondi vers $-\infty$ et vers $+\infty$ du résultat d'opérations arithmétique élémentaires.

Certaines restrictions inhérentes au problème sont toutefois à considérer. Une première limitation concerne l'absence de canonicité dans la spécification de ces arrondis et est directement liée à l'absence de relation d'ordre total sur l'ensemble des polygones du plan. Pour une région polygonale et un mode d'arrondi donné, il existe en effet un ensemble de régions à coordonnées entières vérifiant les propriétés énoncées sans qu'il soit possible de déterminer laquelle constitue le *meilleur* arrondi.

D'autre part, les méthodes d'arrondi proposées ne garantissent pas toujours la préservation de la topologie originale de l'objet. Dans certains cas, cette perte est inévitable, par exemple l'arrondi par défaut d'un polygone vide de point entier est forcément nul. Dans d'autres cas, cette perte pourrait être évitée mais s'avère en définitive souvent souhaitable, par exemple lorsque celle-ci provient de la fusion ou de l'introduction d'une incidence entre des sommets ou des arêtes très proches et correspond à la perte de détails négligeables par rapport à la précision de la grille.

Dans ce mémoire, seul l'arrondi sur une grille uniforme a été abordé. L'emploi d'une grille non-uniforme (comme la grille des flottants par exemple) est toutefois envisageable mais ne nous a pas semblé plus approprié, l'utilisation de la grille flottante disposant de plus de précision lorsque l'on se rapproche de l'origine n'étant pas forcément justifiable pour notre problème.

Par ailleurs, les méthodes d'arrondi proposées dans cette thèse s'appliquent uniquement au cas d'objets linéaires plongés dans l'espace bidimensionnel. Les perspectives de recherche envisageables au terme de ce travail portent donc naturellement sur la généralisation de ces méthodes à des structures plus complexes. L'extension des algorithmes au cas tridimensionnel, c'est-à-dire à l'arrondi de polyèdres, semble raisonnablement possible. Cependant, le problème du calcul effectif du plus proche point entier visible depuis un sommet d'un polyèdre convexe reste à régler. La généralisation directe de la méthode proposée à la section 3.7 aux cas des polyèdres nécessite par exemple la détermination d'une base de $\mathbb{Z}^3$ telle que l'un des vecteurs de cette base se dirige vers l'intérieur du cône défini par les faces incidentes au sommet à arrondir.

La généralisation aux cas d'objets courbes (dans le plan ou dans l'espace) est beaucoup moins évidente et dépend de plus fortement de la représentation de ces objets. Pour ce type de problèmes, la mise au point de méthodes au cas par cas semble nécessaire.

Bibliographie

[1] P. Alliez, O. Devillers, and J. Snoeyink. Removing degeneracies by perturbing the problem or the world. *Reliable Computing*, 6 :61–79, 2000. Special Issue on Computational Geometry.

[2] F. Avnaim, J-D. Boissonnat, O. Devillers, F. P. Preparata, and M. Yvinec. Evaluating signs of determinants using single-precision arithmetic. *Algorithmica*, 17 :111–132, 1997.

[3] I. J. Balaban. An optimal algorithm for finding segment intersections. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages 211–219, 1995.

[4] J. L. Bentley and T. A. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Trans. Comput.*, C-28(9) :643–647, September 1979.

[5] H. Brönnimann, C. Burnikel, and S. Pion. Interval arithmetic yields efficient dynamic filters for computational geometry. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, pages 165–174, 1998.

[6] H. Brönnimann, I. Emiris, V. Pan, and S. Pion. Sign determination in Residue Number Systems. *Theoret. Comput. Sci.*, 210(1) :173–197, 1999. Special Issue on Real Numbers and Computers.

[7] J. Canny. *The Complexity of Robot Motion Planning*. ACM – MIT Press Doctoral Dissertation Award Series. MIT Press, Cambridge, MA, 1987.

[8] *The CGAL Manual*, 2003. Release 3.0.

[9] B. Chazelle and D. P. Dobkin. Detection is easier than computation. In *Proc. 12th Annu. ACM Sympos. Theory Comput.*, pages 146–153, 1980.

[10] B. Chazelle and D. P. Dobkin. Intersection of convex objects in two and three dimensions. *J. ACM*, 34(1) :1–27, January 1987.

[11] B. Chazelle and H. Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. *J. ACM*, 39(1) :1–54, 1992.

[12] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry : Algorithms and Applications*. Springer-Verlag, Berlin, Germany, 2nd edition, 2000.

[13] O. Devillers and F. P. Preparata. A probabilistic analysis of the power of arithmetic filters. *Discrete and Computational Geometry*, 20 :523–547, 1998.

[14] O. Devillers and P. M. Gandoin. Rounding Voronoi diagram. In *Proc. 8th Discrete Geometry and Computational Imagery conference (DGCI99)*, volume 1568 of *Lecture Notes in Computer Science*, pages 375–387. Springer-Verlag, 1999.

[15] O. Devillers and P. Guigue. Finite precision elementary geometric constructions. Rapport de recherche 4559, INRIA, 2002.

[16] O. Devillers and S. Pion. Efficient exact geometric predicates for Delaunay triangulations. In *Proc. 5th Workshop Algorithm Eng. Exper.*, pages 37–44, 2003.

[17] O. Devillers and F. P. Preparata. Further results on arithmetic filters for geometric predicates. *Comput. Geom. Theory Appl.*, 13 :141–148, 1999.

[18] D. P. Dobkin and D. G. Kirkpatrick. Fast detection of polyhedral intersection. *Theoret. Comput. Sci.*, 27(3) :241–253, December 1983.

[19] H. Edelsbrunner and E. P. Mücke. Simulation of simplicity : a technique to cope with degenerate cases in geometric algorithms. In *Proc. 4th Annu. ACM Sympos. Comput. Geom.*, pages 118–133, 1988.

[20] I. Emiris and J. Canny. A general approach to removing degeneracies. *SIAM J. Comput.*, 24 :650–664, 1995.

[21] D. Eppstein. http ://www.ics.uci.edu/~eppstein/junkyard/circumcenter.html.

[22] A. R. Forrest. Computational geometry in practice. In R. A. Earnshaw, editor, *Fundamental Algorithms for Computer Graphics*, volume F17 of *NATO ASI*, pages 707–724. Springer-Verlag, 1985.

[23] A. R. Forrest. Computational geometry and software engineering : Towards a geometric computing environment. In D. F. Rogers and R. A. Earnshaw, editors, *Techniques for Computer Graphics*, pages 23–37. Springer-Verlag, 1987.

[24] S. Fortune. A sweepline algorithm for Voronoi diagrams. In *Proc. 2nd Annu. ACM Sympos. Comput. Geom.*, pages 313–322, 1986.

[25] S. Fortune. Stable maintenance of point set triangulations in two dimensions. In *Proc. 30th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 494–505, October 1989.

[26] S. Fortune. Robustness issues in geometric algorithms. In M. C. Lin and D. Manocha, editors, *Applied Computational Geometry : Towards Geometric Engineering*, volume 1148 of *Lecture Notes Comput. Sci.*, pages 9–14. Springer-Verlag, 1996.

[27] S. Fortune. Vertex-rounding a three-dimensional polyhedral subdivision. *Discrete Comput. Geom.*, 22(4) :593–618, 1999.

[28] S. Funke and K. Mehlhorn. Look : A lazy object-oriented kernel for geometric computation. In *Proc. 16th Annu. ACM Sympos. Comput. Geom.*, pages 156–165, 2000.

[29] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi. A fast procedure for computing the distance between complex objects. *Internat. J. Robot. Autom.*, 4(2) :193–203, 1988.

[30] D. Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Comput. Surv.*, 23(1) :5–48, March 1991.

[31] J. E. Goodman, R. Pollack, and B. Sturmfels. Coordinate representation of order types requires exponential storage. In *Proc. 21st Annu. ACM Sympos. Theory Comput.*, pages 405–410, 1989.

[32] M. Goodrich, L. J. Guibas, J. Hershberger, and P. Tanenbaum. Snap rounding line segments efficiently in two and three dimensions. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 284–293, 1997.

[33] R. L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Inform. Process. Lett.*, 1 :132–133, 1972.

[34] T. Granlund. *GMP, The GNU Multiple Precision Arithmetic Library*. http ://www.swox.com/gmp/.

[35] D. H. Greene and F. F. Yao. Finite-resolution computational geometry. In *Proc. 27th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 143–152, 1986.

[36] L. J. Guibas and D. Marimont. Rounding arrangements dynamically. *Internat. J. Comput. Geom. Appl.*, 8 :157–176, 1998.

[37] L. J. Guibas, D. Salesin, and J. Stolfi. Epsilon geometry : building robust algorithms from imprecise computations. In *Proc. 5th Annu. ACM Sympos. Comput. Geom.*, pages 208–217, 1989.

[38] P. Guigue and O. Devillers. Fast and robust triangle-triangle overlap test using orientation predicates. *Journal of graphics tools*, 8(1) :39–52, 2003.

[39] P. Guigue and O. Devillers. Fast and robust triangle-triangle overlap test using orientation predicates. Web page, August 2003. http ://www.acm.org/jgt/papers/GuigueDevillers03/.

[40] S. Har-Peled. An output sensitive algorithm for discrete convex hulls. *Comput. Geom. Theory Appl.*, 10 :125–138, 1998.

[41] M. Held. ERIT – A collection of efficient and reliable intersection tests. Technical Report, University at Stony Brook, 1996.

[42] J. D. Hobby. Practical segment intersection with finite precision output. *Comput. Geom. Theory Appl.*, 13(4) :199–214, October 1999.

[43] C. M. Hoffmann. The problems of accuracy and robustness in geometric computation. *IEEE Computer*, 22(3) :31–41, March 1989.

[44] *IEEE Standard for binary floating point arithmetic, ANSI/IEEE Std* $754-1985$. New York, NY, 1985. Reprinted in SIGPLAN Notices, 22(2) :9–25, 1987.

[45] V. Karamcheti, C. Li, I. Pechtchanski, and C. Yap. A CORE library for robust numeric and geometric computation. In *15th ACM Symp. on Computational Geometry*, 1999.

[46] F. Klein. Sur une représentation géométrique du développement en fraction continue ordinaire. *Nouv. Ann. Math.*, 15(3) :327–331, 1896.

[47] H. S. Lee and R. C. Chang. Approximating vertices of a convex polygon with grid points in the polygon. In *Proc. 3rd Annu. Internat. Sympos. Algorithms Comput.*, volume 650 of *Lecture Notes Comput. Sci.*, pages 269–278. Springer-Verlag, 1992.

[48] Z. Li and V. Milenkovic. Compaction and separation algorithms for nonconvex polygons and their applications. *European J. Oper. Res.*, 84 :539–561, 1995.

[49] G. Liotta, F. P. Preparata, and R. Tamassia. Robust proximity queries : An illustration of degree-driven algorithm design. *SIAM J. Comput.*, 28(3) :864–889, 1998.

[50] L. Lovasz. *An Algorithmic Theory of Numbers, Graphs, and Convexity*. CBMS-NSF Regional Conference Series in Applied Mathematics. Society of Industrial and Applied Mathematicians, Philadelphia, PA, 1986.

[51] K. Mehlhorn and S. Näher. *LEDA : A Platform for Combinatorial and Geometric Computing*. Cambridge University Press, Cambridge, UK, 2000.

[52] D. Michelucci. *Les représentations par les frontières : quelques constructions ; difficultées rencontrées*. Thèse de doctorat en sciences, École Nationale Supérieure des Mines de Saint-Étienne, 1987.

[53] D. Michelucci. Arithmetic issues in geometric computations. In *Proc. 2nd Real Numbers and Computer Conf.*, pages 43–69, 1996.

[54] V. Milenkovic and L. R. Nackman. Finding compact coordinate representations for polygons and polyhedra. In *Proc. 6th Annu. ACM Sympos. Comput. Geom.*, pages 244–252, 1990.

[55] V. J. Milenkovic. Shortest path geometric rounding. *Algorithmica*, 27(1) :57–86, 2000.

[56] T. Möller. A fast triangle-triangle intersection test. http ://www.acm.org/jgt/papers/Moller97/.

[57] T. Möller. A fast triangle-triangle intersection test. *Journal of Graphics Tools*, 2(2) :25–30, 1997.

[58] E. Packer and D. Halperin. Snap rounding revisited. In *Abstracts 17th European Workshop Comput. Geom.*, pages 82–85. Freie Universität Berlin, 2001.

[59] S. Pion. *De la géométrie algorithmique au calcul géométrique*. Thèse de doctorat en sciences, université de Nice-Sophia Antipolis, France, 1999.

[60] S. Pion. Interval arithmetic : an efficient implementation and an application to computational geometry. In *Workshop on Applications of Interval Analysis to systems and Control*, pages 99–110, 1999.

[61] F. P. Preparata and M. I. Shamos. *Computational Geometry : An Introduction*. Springer-Verlag, 3rd edition, October 1990.

[62] D. Priest. Algorithms for arbitrary precision floating point arithmetic. In *Proc. 10th Symp. on computer arithmetic*, pages 132–143, 1991.

[63] D. Priest. *On properties of floating point arithmetics : numerical stability and the cost of accurate computations*. Ph.D. thesis, Dept. of mathematics, Univ. of California at Berkeley, 1992.

[64] S. Robbins. package for intersection. Web page, July 2001. http ://cgm.cs.mcgill.ca/~stever/CGAL/.

[65] S. Robbins and S. Whitesides. Personal communication.

[66] J. T. Schwartz. Finding the minimum distance between two convex polygons. *Inform. Process. Lett.*, 13 :168–170, 1981.

[67] R. Seidel. The nature and meaning of perturbations in geometric computing. In *Proc. 11th Sympos. Theoret. Aspects Comput. Sci.*, volume 775 of *Lecture Notes Comput. Sci.*, pages 3–17. Springer-Verlag, 1994.

[68] J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, London, UK, 1982.

[69] J. R. Shewchuk. Adaptive precision floating-point arithmetic and fast robust geometric predicates. *Discrete Comput. Geom.*, 18(3) :305–363, 1997.

[70] C. Yap. Towards exact geometric computation. *Comput. Geom. Theory Appl.*, 7(1) :3–23, 1997.

[71] C. Yap. A geometric consistency theorem for a symbolic perturbation scheme. *J. Comput. Syst. Sci.*, 40(1) :2–18, 1990.

[72] C. Yap and T. Dubé. The exact computation paradigm. In D.-Z. Du and F. K. Hwang, editors, *Computing in Euclidean Geometry*, volume 4 of *Lecture Notes Series on Computing*, pages 452–492. World Scientific, Singapore, 2nd edition, 1995.

[73] J. Yu. Exact arithmetic solid modeling. Technical Report CSD-TR-92-037, Comput. Sci. Dept., Purdue University, June 1992.

UNIVERSITE DE NICE SOPHIA ANTIPOLIS
UFR SCIENCES
SERVICE SCOLARITE 3EM CYCLE

AVIS DU JURY SUR LA REPRODUCTION DE LA THESE SOUTENUE

1. TITRE DE LA THESE : **Construction géométriques à précision fixée**

2. NOM ET PRENOM DE L'AUTEUR : **Monsieur GUIGUE Philippe**

3. MEMBRES DU JURY : J-M Moreau, J-M Muller
O. Devillers, A. Lieutier, S. Pion.

4. PRESIDENT DU JURY : A. GALLIGO

5. DATE DE LA SOUTENANCE : 5 Dec 03

6. REPRODUCTION DE LA THESE SOUTENUE :

- [x] Thèse pouvant être reproduite en l'état
- [ ] Thèse ne pouvant pas être reproduite
- [ ] Thèse pouvant être reproduite après correction suggérées au cours de la soutenance

Signature du Président du jury

A. Galligo

Secrétarat ED STIC
250 rue Albert Einstein
Lucioles 1 bat 4
06560 Valbonne

CONSTRUCTIONS GÉOMÉTRIQUES À PRÉCISION FIXÉE

Les problèmes de robustesse liés à la substitution du calcul exact sur les réels par le calcul flottant approché sont souvent un obstacle à l'implantation pratique des algorithmes géométriques. Si l'adoption du paradigme exact apporte une solution satisfaisante à ce type de problèmes pour les algorithmes ayant un résultat purement combinatoire, cette solution ne permet cependant pas de résoudre en pratique le cas des algorithmes qui réutilisent voire cascadent la construction de nouveaux objets géométriques.

Cette thèse aborde le problème de l'arrondi sur la grille entière du résultat d'opérations booléennes sur des régions polygonales et propose plusieurs notions d'arrondi permettant de garantir certaines propriétés métriques et topologiques intéressantes entre le résultat exact et sa version arrondie telles que la garantie de relations d'inclusion et la préservation de la convexité du résultat. Nos méthodes sont basées sur l'utilisation de constructeurs élémentaires arrondis pour lesquels nous présentons également plusieurs algorithmes efficaces. Nous proposons enfin des tests rapides permettant la détection robuste d'intersection entre plusieurs types d'objets convexes dans le plan et dans l'espace. L'ensemble de ces solutions trouvent une application directe en CAO et en graphisme.

Mots-clés : géométrie algorithmique, arrondi géométrique, opérations booléennes, polygone, grille uniforme, test d'intersection, graphisme, robustesse, détection de collision.

GEOMETRIC CONSTRUCTIONS WITH FINITE PRECISION

Robustness problems resulting from the substitution of floating-point arithmetic for exact arithmetic on real numbers are often an obstacle to the practical implementation of geometric algorithms. For algorithms that are purely combinatorial, the use of the exact computation paradigm gives a satisfactory solution to this problem. However, this paradigm does not allow to practically solve the case of algorithms that reuse or cascade the construction of new geometric objects.

This thesis treats the problem of rounding the result of set operations on polygonal regions onto the integer grid. We propose several rounding modes that allow to guarantee some interesting metric and topological properties between the exact result and its rounded counterpart such as inclusion properties and convexity preservation. Our methods are based on the rounding of elementary geometric constructions, e.g. rounding a vertex of a convex polygon to its nearest interior grid point, for which we propose efficient algorithms. We finally present some fast overlap tests that allow to detect in a robust manner the intersection of several kinds of convex objects in the plane and in the three dimensional space. All methods have a direct application in domains such as CAD and computer graphics.

Keywords : computational geometry, geometric rounding, set operations, polygon, uniform grid, intersection test, computer graphics, robustness, collision detection.