École Doctorale Sciences et Ingénierie (SI), ED 653 Institut XLIM UMR CNRS n°7252

Construction efficace de géométrie pour l'analyse structurelle de grands systèmes moléculaires

Thèse pour obtenir le grade de

Docteur de l'Université de Limoges en Informatique

Présentée et soutenue par

Cyprien Plateau–Holleville

Le 21 Novembre 2024

Devant le jury composé de

Président du jury
Loïc Barthe — Professeur des Universités, *IRIT, Université Paul Sabatier Toulouse*

Rapporteur et Rapporteure
Julie Digne — Directrice de Recherche, *CNRS, LIRIS*
Bruno Lévy — Directeur de Recherche, *INRIA Saclay, Université Paris-Saclay*

Examinateur et Examinatrice
Nathalie Lagarde — Maîtresse de Conférences, *GBCM, CNAM*
Jonathan Sarton — Maître de Conférences, *ICube, Université de Strasbourg*

Directeurs de thèse
Maxime Maria — Maître de Conférences, *XLIM, Université de Limoges*
Stéphane Mérillou — Professeur des Universités, *XLIM, Université de Limoges*

Remerciements

J'exprime une très grande gratitude envers mes directeurs de thèse Stéphane Mérillou et Maxime Maria pour m'avoir accordé leur confiance et une importante liberté au cours de ces travaux. Plus particulièrement, je remercie le premier pour sa disponibilité et ses nombreux conseils ainsi que le second pour son accueil et son soutien, mais aussi pour son engagement et sa rigueur.

Je souhaite aussi remercier Julie Digne et Bruno Lévy pour avoir accepté de rapporter cette thèse, mais aussi les autres membres du jury : Loïc Barthe, Nathalie Lagarde et Jonathan Sarton.

Je suis extrêmement reconnaissant envers les co-auteurs des travaux de cette thèse. De façon plus précise, je remercie Matthieu Montes pour m'avoir aidé à comprendre les enjeux des thématiques traitées et liées à la biochimie. Je souhaite aussi remercier Benjamin Stamm pour son soutien sur les parties les plus théoriques de ce travail ainsi que son équipe pour leur accueil chaleureux à Stuttgart. Je suis enfin très reconnaissant envers Vincent Nivoliers pour son aide importante et son accueil à Lyon.

Je souhaite de plus remercier mes co-bureaux et amis, Heinich Porro Sufan ainsi que Vincent Larroque, pour la qualité de nos échanges et l'ambiance chaleureuse que nous avons réussi à installer ensemble.

C'est ensuite mes collègues de l'équipe SIR, de l'équipe de développement de VTX, de l'axe ASALI, d'XLIM, de la faculté des sciences, mais aussi de la communauté française d'informatique graphique que je remercie pour leur accueil et avec qui j'ai eu très grand plaisir à échanger.

Je témoigne d'une grande reconnaissance envers ma famille et mes amis. Plus particulièrement, je remercie mes parents pour leur confiance, soutien et détermination à qui je dois une part importante de mon parcours.

Enfin, je souhaite remercier Claudia Gauthier pour son soutien inébranlable, son infinie patience et ses encouragements intarissables.

Table des Matières

Chapitre 1

Introduction

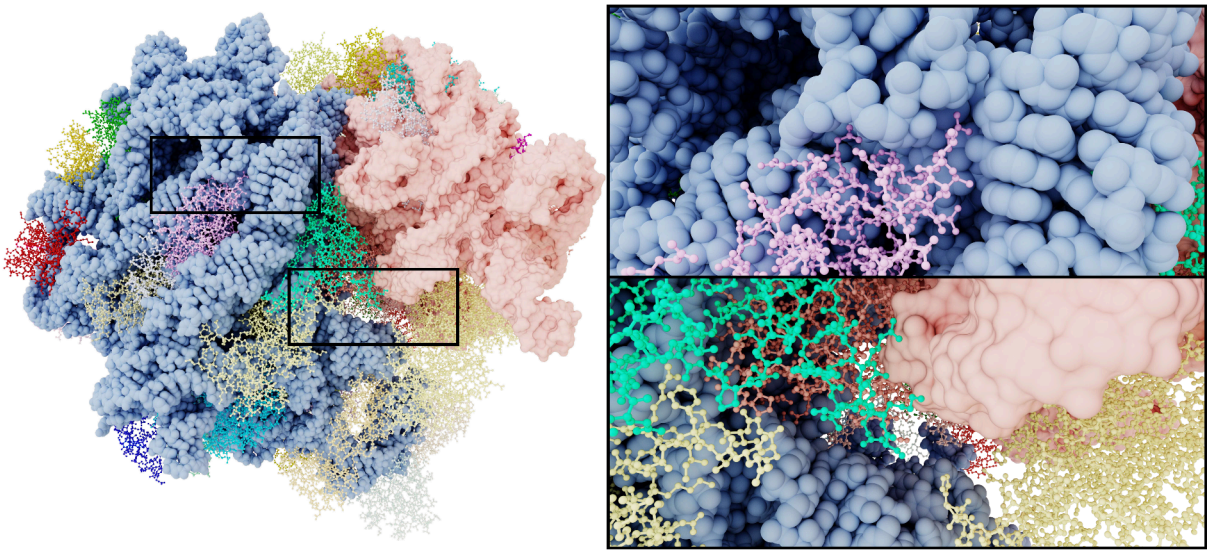


Figure 1.1 – Illustration d'un complexe moléculaire d'un ribosome à partir d'un assemblage de différentes représentations (PDB : 6GXO).

Sommaire

Depuis les débuts de l'informatique graphique, la visualisation scientifique représente une application fondamentale en soutien à d'autres sciences et est aujourd'hui un outil incontournable des logiciels dédiés. La représentation de données aussi bien concrètes qu'abstraites est nécessaire à leur exploration par un utilisateur. À la différence d'autres domaines de l'informatique graphique où l'affichage est dirigé par le soutien de la créativité des artistes, la visualisation scientifique est souvent guidée par la recherche de la meilleure perception des informations délivrées à l'utilisateur. Celles-ci sont parfois contraintes par les méthodes d'acquisition et de simulation dont les capacités et la précision varient en fonction du contexte.

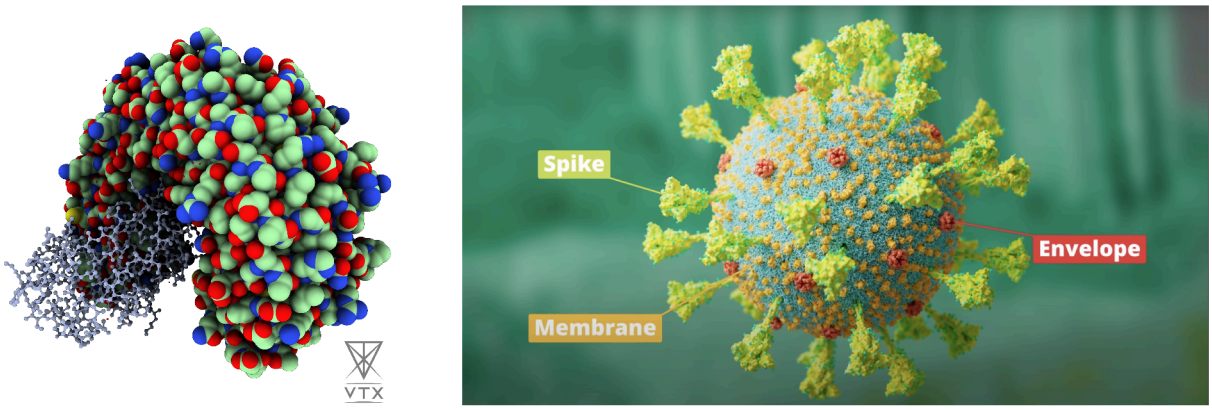
Il est commun en informatique graphique de distinguer l'affichage interactif de l'affichage illustratif par leurs objectifs. Tandis que le premier vise d'abord le calcul rapide afin de permettre à l'utilisateur d'explorer la scène et manipuler ses objets, le second est orienté vers la production d'une image de la meilleure qualité possible. Ces différences sont intrinsèquement liées à la qualité des géométries cibles, mais aussi à la méthode de rendu utilisée. Même si ces aspects semblent secondaires pour la visualisation scientifique, ils restent cependant importants afin de proposer une présentation claire des informations ciblées. De plus, lorsqu'on souhaite afficher des entités spatiales, une simulation des interactions lumineuses permet une mise en valeur intuitive et esthétique. Ainsi, même si la visualisation scientifique est aujourd'hui majoritairement liée à l'affichage interactif, de plus en plus de travaux se sont concentrés sur l'illustration de haute qualité pour la communication de résultats scientifiques [Sla+22 ; Joh23]. Comme illustré figure 1.2, cela permet une bonne perception de la géométrie, elle-même de bonne qualité.

Les données moléculaires sont l'une des premières cibles abordées par la visualisation scientifique [AB02]. D'abord dessinées à la main sous la forme de schémas, la représentation par ordinateur des molécules est devenue un outil important à la base de leur analyse [Koz+17]. Leurs structures complexes représentent ainsi un défi pour les algorithmes de visualisation dont l'applicabilité est toujours remise en question par l'amélioration des stratégies d'acquisition de données. Ces dernières permettant un niveau de précision et des échelles plus importants résultant en de plus grandes quantités de données, les techniques de visualisation doivent s'adapter afin de proposer des temps de calcul appropriés aux fonctionnalités souhaitées.

Cette thèse présente nos travaux visant le calcul efficace de géométrie pour l'analyse structurelle de protéines à travers notamment la visualisation moléculaire. Dans cette introduction, nous décrivons donc les structures ciblées par nos études ainsi que leurs représentations principales (section 1.1). Nous présentons ensuite les objectifs et contributions de nos travaux (section 1.2).

1.1 Structures moléculaires

Les complexes moléculaires sont caractérisés par des données de différentes natures. Celles-ci peuvent définir leur structure, mais aussi certaines de leurs propriétés locales et globales guidant la visualisation. Dans cette section, nous introduisons donc les cibles de nos travaux ainsi que certaines de leurs propriétés.



(a) Visualisation temps réel **(b)** Illustration pour la communication scientifique

Figure 1.2 – Deux types de visualisation souhaités pour l'exploration de structures moléculaires. (a) Capture d'une visualisation interactive d'une protéine avec le logiciel de visualisation VTX [Mar+22]. (b) Capture du film de vulgarisation scientifique ≪The lifecycle of SARS-CoV-2 ≫, par Raimond Ravelli, Kèvin Knoops et Phospho Biomedical Animation.

1.1.1 Description générale

Les molécules sont des ensembles d'atomes appelés biomolécules lorsqu'elles structurent le vivant comme l'ADN ou les protéines. En tant qu'ensemble fonctionnel, elles représentent la cible d'études cherchant à déterminer leurs rôles dans les systèmes qu'elles composent. L'assemblage chimique de leurs atomes et leur arrangement spatial (appelé conformation) définissent leurs structures et sont à la base de leurs fonctionnements. Leurs tailles étant particulièrement variées, les stratégies d'acquisition représentent un enjeu important qui a bénéficié de progrès remarquables depuis la fin du 20^{e} siècle. Notamment, les méthodes basées sur la congélation des échantillons pour leur observation à l'aide d'un microscope électronique, appelées cryomicroscopie, représentent un standard permettant l'acquisition de grandes structures, comme des virus, avec une précision importante [Lév+21]. Une fois un volume obtenu, les données structurelles de la molécule comme la position des atomes, leur élément et leurs liaisons sont dérivées en fonction de règles physico-chimiques. Les travaux de cette thèse visant la visualisation de données biochimiques, les molécules prises pour exemple seront majoritairement des protéines obtenues à partir de la *Protein Data Bank* (PDB) [Ber+00]. Elles seront donc nommées par leur identifiant sous la forme d'un quadruplet alphanumérique (*e.g* : ≪ PDB : 1AGA ≫).

Comme illustré figure 1.3, la structure d'une protéine est principalement définie par les atomes qui la composent, caractérisés par leur position dans un espace tridimensionnel, souvent donnée en Ångström Å (0.1 nanomètre), et leur élément correspondant selon le tableau périodique. En plus de ces informations essentielles, il peut être intéressant de connaitre les liaisons entre les atomes ainsi que leurs charges électriques. Tandis que les premières permettent d'identifier des schémas structurels tels que les chaînes carbonées qui représentent le squelette des protéines, les secondes offrent des informations locales et globales, notamment utilisées dans des simulations numériques. Ces dernières décrivent l'évolution d'un système donné, souvent visualisable sous la forme d'une séquence de conformation, et sont à la base de l'étude d'un système moléculaire.

Puisque ces simulations sont souvent coûteuses en temps et en puissance de calcul, des outils additionnels permettent de préfiltrer les combinaisons possibles. La visualisation moléculaire s'inscrit notamment en soutien de cette recherche.

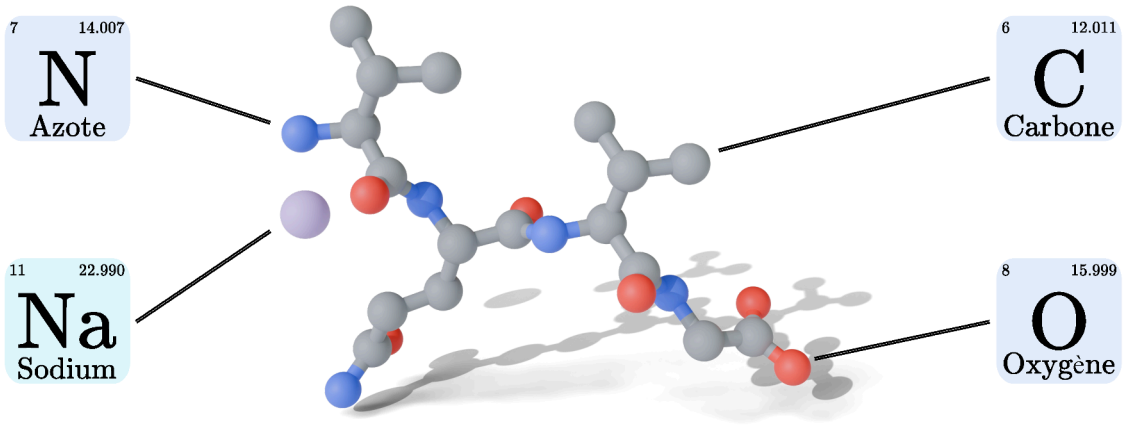
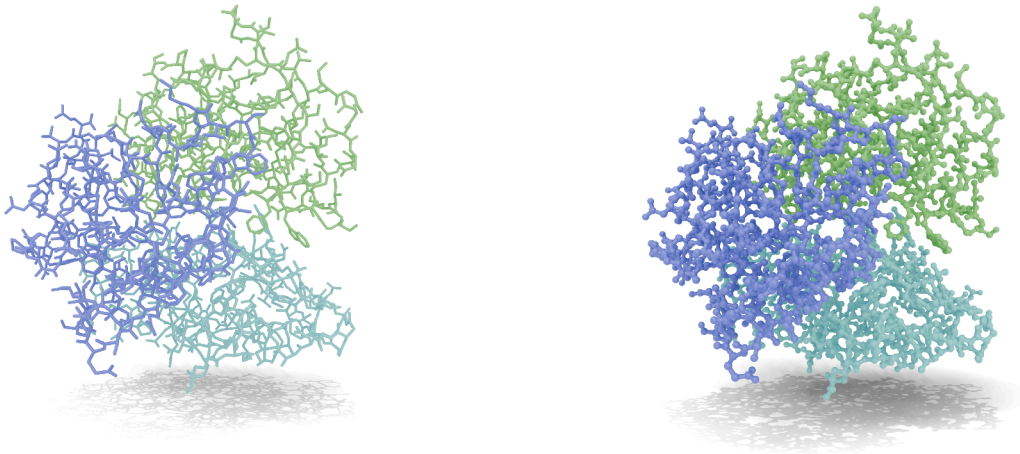


Figure 1.3 – Structure d'une petite molécule (PDB : 5ZCK).

1.1.2 Représentations moléculaires

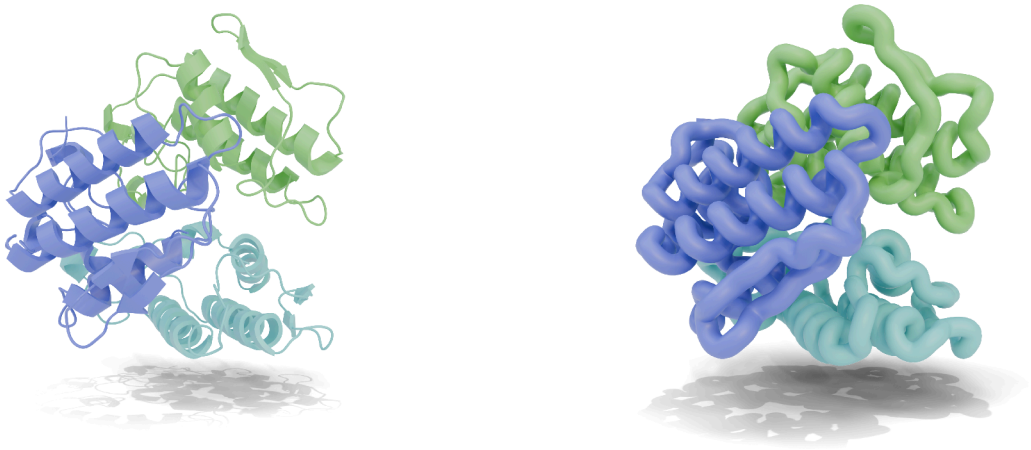
Même s'il est possible de décrire des molécules sous la forme de texte, de schéma, ou en deux dimensions, notre étude ne concerna que la visualisation en trois dimensions, particulièrement intéressante pour la manipulation et l'exploration. Les principales représentations couramment disponibles dans les logiciels de visualisation peuvent être divisées en trois catégories visant différents niveaux de lecture : orientées liaisons, abstraites et surfaciques. Tandis que la première est basée sur la représentation explicite des liaisons interatomiques, la seconde met en valeur la structure de la protéine de façon abstraite et la dernière expose l'interface entre la molécule visée et son environnement.

Les liaisons entre les atomes sont de différents types et caractérisent la structure d'une molécule. Les représentations basées sur celles-ci sont ainsi très couramment utilisées, notamment pour des molécules de petites tailles. Comme illustré figure 1.4, une molécule peut être représentée par ses liaisons seules (*Stick*, figure 1.4a), mais aussi en les associant à une représentation explicite



(a) *Stick* **(b)** *Ball & Stick*

Figure 1.4 – Représentations moléculaires orientées liaisons communes (PDB : 1A3F).



(a) *Cartoon* **(b)** *Ribbons*

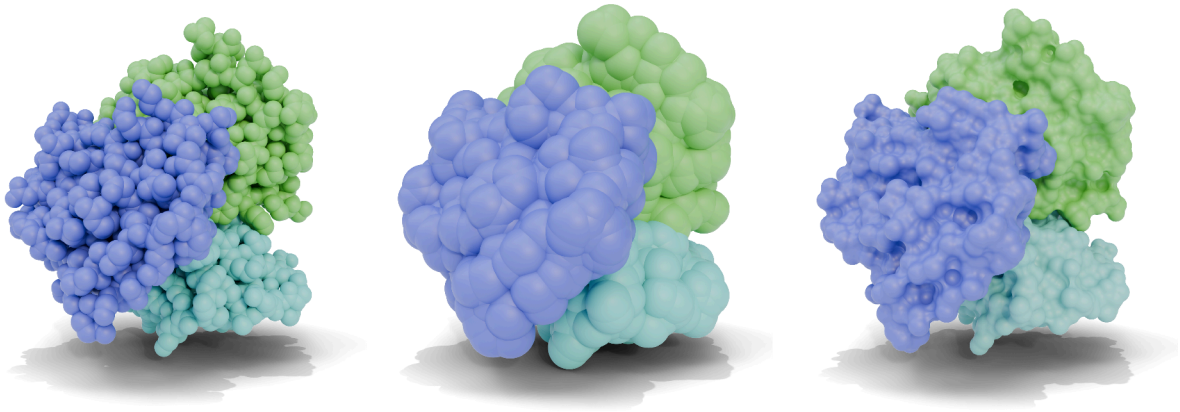
Figure 1.5 – Représentations moléculaires abstraites communes (PDB : 1A3F).

de ses atomes sous la forme de petites sphères (*Ball & Stick*, figure 1.4b). Ces visualisations offrent cependant peu d'informations structurelles de haut niveau sémantique.

La structure moléculaire est particulièrement importante puisqu'elle donne de nombreuses informations sémantiques. Il est donc intéressant de visualiser certains schémas structurels, comme la structure secondaire. Cette dernière est caractérisée par des sous-ensembles d'atomes peu mobiles lors du repliement d'une protéine. Ainsi, différentes représentations abstraites ont été développées visant à mettre en valeur ces aspects, comme illustré figure 1.5 avec les représentations *Cartoon* (figure 1.5a) et *Ribbons* (figure 1.5b).

Les représentations surfaciques sont principalement basées sur le modèle compact définissant la surface de *van der Waals*, comme illustré figure 1.6a. Dans celui-ci, chaque atome est représenté par une sphère centrée sur son noyau et dont le rayon est proportionnel à la force de van der Waals associée à son élément [Ric77]. La surface est ainsi définie par l'union de toutes les sphères. L'un des principaux avantages de cette formulation est la représentation implicite de l'interaction des atomes du système à travers la définition d'autres surfaces. La *Surface Accessible au Solvant* (SAS, figure 1.6b) et la *Surface Exclue au Solvant* (SES, figure 1.6c) caractérisent l'accessibilité d'une petite molécule appelée solvant, souvent de l'eau. Les endroits accessibles à ce solvant jouant un rôle sur les caractéristiques fonctionnelles de la protéine, ces surfaces sont utilisées pour la visualisation, mais aussi pour des calculs scientifiques ainsi que des simulations. Les représentations surfaciques sont particulièrement intéressantes puisqu'elles permettent de calculer des quantités soutenant l'analyse des systèmes comme leur hydrophobie [Ric84]. Les surfaces de van der Waals et exclue au solvant sont de plus couramment utilisées afin de représenter les interactions potentielles entre corps moléculaires. Notamment, les cavités de la SES sont beaucoup étudiées [Kro+16] et représentent de potentiels sites dans lesquels une autre molécule peut être positionnée. De plus, sa surface, propre à une entité, permet de représenter des informations comme l'énergie à travers des nuances de couleurs. Ainsi, elle soutient directement l'étape préalable à certaines simulations.

En raison de leur géométrie squelettique, les représentations orientées liaisons et abstraites sont généralement peu adaptées aux complexes moléculaires de grandes tailles. En effet, tandis que les représentations surfaciques permettent une bonne visualisation indépendamment du point



(a) *Surface de van der Waals* **(b)** *Surface Accessible au Solvant* **(c)** *Surface Exclue au Solvant*

Figure 1.6 – Représentations moléculaires surfaciques communes (PDB : 1A3F).

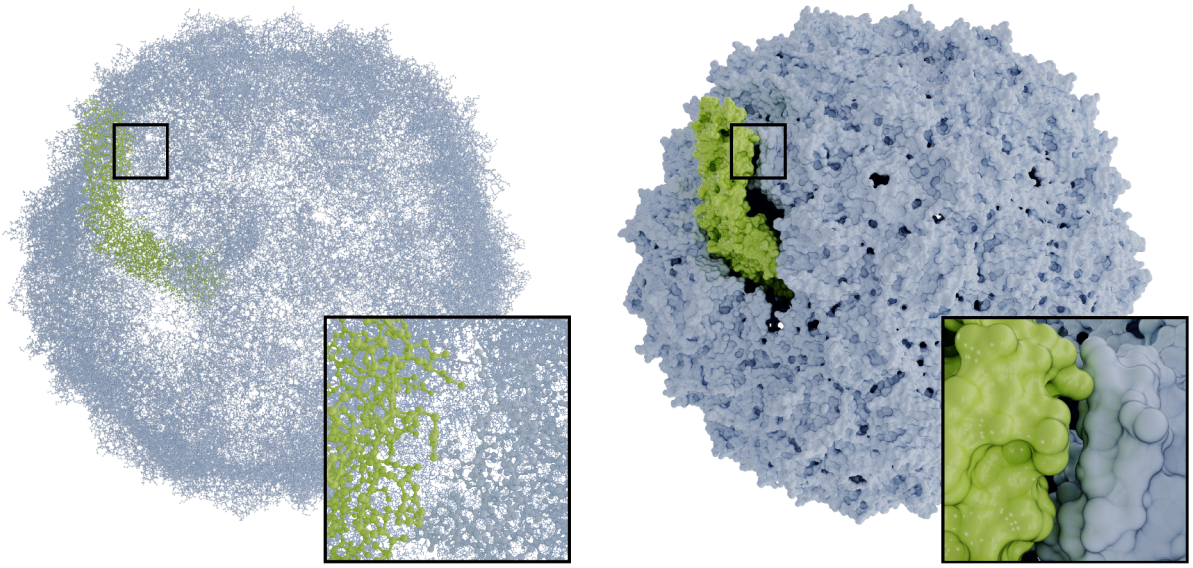
de vue (figure 1.7b), les autres ne permettent pas une perception précise de la géométrie à une grande distance (figure 1.7a). Les objets moléculaires de grande taille sont donc communément représentés par des approches surfaciques.

1.2 Objectifs de la thèse

Les représentations surfaciques sont beaucoup utilisées pour la visualisation de données moléculaires, mais aussi comme base pour le calcul scientifique en tant que modèle des interactions potentielles. Elles représentent cependant un enjeu en termes de temps de construction, mais aussi de mise à l'échelle sur des structures de grande taille. De plus, en raison de leurs nombreuses utilisations possibles, la qualité de la géométrie proposée par les algorithmes impacte les usages, ce qui rend difficiles des optimisations liées à l'utilisation de plusieurs niveaux de détails. La définition des protéines en tant qu'ensemble d'atomes se prête particulièrement bien à leur traitement par des composants matériels proposant des environnements massivement parallèles, comme les GPUs, permettant ainsi d'effectuer un nombre très important de petites tâches indépendantes et supportant donc bien les mises à l'échelle. Cette thèse vise alors à proposer des algorithmes innovants pour les représentations surfaciques en soutien à la visualisation, mais aussi au calcul scientifique. Son objectif principal est de profiter d'environnements parallèles afin de mettre en place des traitements rapides tout en ne diminuant pas l'applicabilité de la structure construite. Enfin, il est souhaitable que les stratégies développées soient compatibles avec de grandes structures comme celle présentée figure 1.7. En effet, celles-ci remettent actuellement en question les algorithmes de l'état de l'art en raison de la quantité de données à laquelle elles sont liées.

Au cours de cette thèse, nous nous sommes en premier lieu concentrés sur le calcul efficace de la SES de façon parallèle et exhaustive. Ainsi, nous proposons un processus dédié avec les contributions suivantes :

— nous définissons une structure de données adaptée aux environnements parallèles modernes et basée sur un schéma de calcul plutôt que stockage ;
— notre méthode permet une réduction importante de la quantité de mémoire nécessaire à la construction de la surface ;



(a) *Ball & Stick* **(b)** *SES*

Figure 1.7 – Exemples de visualisation de grandes protéines, ici un virion (PDB : 7LHD). Les représentations orientées liaisons sont squelettiques et ne permettent pas une bonne visualisation générale de la géométrie. La SES, au contraire, permet une perception précise à faible et haute distance.

— nous introduisons une stratégie de calcul coopératif et parallèle permettant un traitement rapide.

Ces développements permettent le calcul rapide de la SES de très grands complexes moléculaires compatible avec des GPUs standards. De plus, en comparaison avec la méthode précédente la plus rapide, mais limitée à un calcul partiel de la surface, notre stratégie propose des performances légèrement plus rapides tout en calculant la surface de façon exhaustive et nécessitant beaucoup moins de mémoire. Ce travail a fait l'objet d'une présentation aux *journées Françaises de l'Informatique Graphique 2023* (jFIG 2023) [Pla+23b] ainsi que d'une publication dans le journal *IEEE Transactions on Visualization and Computer Graphics* [Pla+24].

Nous avons ensuite souhaité généraliser l'approche de façon non restreinte à un type de surface en particulier. Le diagramme d'Apollonius, ou diagramme de Voronoï additivement pondéré, représente les fondations de nombreux travaux visant le calcul de la SES, mais aussi l'analyse de complexes moléculaires. Ainsi, il représente le cœur de différentes méthodes utilisant ses propriétés. Cependant, il est plus complexe à calculer que d'autres diagrammes de Voronoï généralisés et ses caractéristiques sont moins bien connues. Nous nous sommes donc intéressés à sa caractérisation formelle afin d'en étudier ses propriétés. Nous proposons une étude systématique du diagramme en deux et trois dimensions. Sur la base de ces résultats, nous présentons une paramétrisation permettant son maillage.

Enfin, sur la base de notre caractérisation du diagramme d'Apollonius, nous proposons une méthode de calcul sur GPU basée sur une construction parallèle des cellules. Celle-ci est soutenue par les contributions suivantes :

— nous présentons une méthode de construction itérative d'une cellule du diagramme d'Apollonius ;

— afin de prendre en charge des distributions spatiales variées, nous proposons plusieurs méthodes d'exploration spatiale ;
— nous proposons une structure de données légère adaptée à la construction efficace dans des environnements avec une quantité de mémoire limitée.

Notre méthode permet un calcul parallèle du diagramme d'Apollonius sur GPU. En comparaison avec les méthodes précédentes visant un calcul efficace, notre stratégie n'est pas limitée aux composantes connectées du diagramme tout en proposant, en moyenne, des temps de calcul deux fois inférieurs.

En parallèle de ces travaux, nous avons développé une application, appelée UDock2, soutenant la préparation de simulations moléculaires. Notamment, elle permet le calcul rapide de l'énergie d'un complexe moléculaire afin de tester de façon interactive des conformations intéressantes. Ce travail a été présenté dans le journal *Bioinformatics* [Pla+23a].

Nous présentons tout d'abord les travaux précédents visant le calcul de surface moléculaire ainsi que des diagrammes de Voronoï et leurs généralisations (chapitre 2). Nous décrivons ensuite la méthode parallèle de calcul efficace de la SES ainsi que ses performances (chapitre 3) qui sont suivies par notre étude formelle du diagramme d'Apollonius permettant sa paramétrisation et le calcul de sa géométrie (chapitre 4) ainsi que notre stratégie de calcul parallèle et ses performances (chapitre 5). Nous concluons ensuite cette thèse en donnant des perspectives de travaux futurs (chapitre 6). Enfin, nous présentons en annexe UDock2 et ses fonctionnalités (annexe A).

Chapitre 2

État de l'art

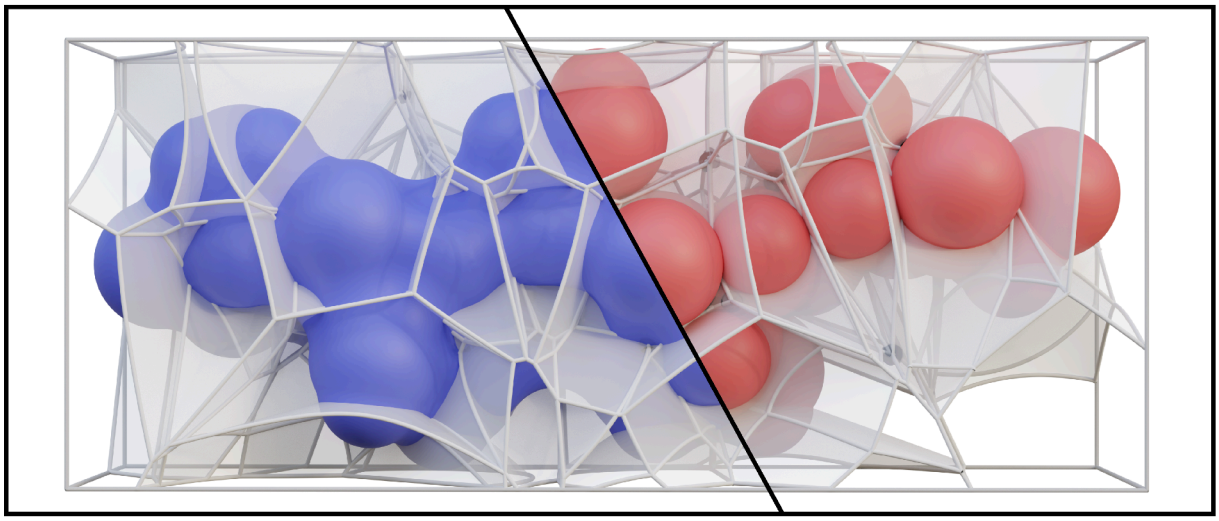


Figure 2.1 – Illustration de la SES (en bleu), de la surface de van der Waals (en rouge) et du diagramme d'Apollonius (en blanc) d'une protéine (PDB : 6EEX).

Sommaire

Ce chapitre présente les méthodes précédant les recherches effectuées au cours de cette thèse. Celles-ci étant orientées sur la recherche de performance sur GPU, la première section présente les caractéristiques de ces architectures, mais aussi leurs avantages et restrictions (section 2.1). Nous introduisons ensuite les définitions formelles des surfaces moléculaires ciblées, plus particulièrement de la Surface Exclue au Solvant, et présentons les principaux travaux permettant leurs constructions (section 2.2). Les diagrammes de Voronoï et leurs généralisations pouvant servir à construire les surfaces moléculaires, nous présentons leurs caractéristiques et méthodes de calcul principales (section 2.3).

2.1 Architectures massivement parallèles

La principale unité de calcul des ordinateurs modernes est le *Central Processing Unit* (CPU) qui est conçu de façon à proposer une rapidité d'exécution séquentielle très importante. Même s'il offre une simplicité relative d'implémentation ainsi qu'une vitesse de traitement intéressante, il est peu adapté lorsque les tâches sont à effectuer sur un grand nombre de données. Ainsi, il dispose couramment de caractéristiques additionnelles visant l'exécution rapide d'un nombre important d'opérations, et particulièrement en parallèle. Notamment, il est en général composé de quelques cœurs de traitement permettant d'effectuer des opérations totalement parallèles. De plus, il est souvent capable d'exécuter des jeux d'instructions supplémentaires fonctionnant en *Simple Instruction Multiple Data* (SIMD) selon la taxonomie de Flynn [Fly66] et permettant d'effectuer la même opération sur un petit ensemble de données de façon parallèle, améliorant ainsi les performances de traitement.

Depuis leur démocratisation au début des années 2000, les *Graphics Processing Units* (GPU) sont des composants essentiels à la base de nombreuses applications. À la différence des CPUs, ils mettent à la disposition du développeur de nombreux cœurs de calcul permettant l'exécution d'un grand nombre d'opérations parallèles. Même s'ils étaient au départ spécialisés dans l'affichage de

Figure 2.2 – Schéma de l'architecture matérielle ADA des cartes graphiques NVIDIA [NVI23]. Les cœurs du GPUs (SM) sont principalement composés d'unité de calcul et permettent de les répartir à leurs threads respectifs.

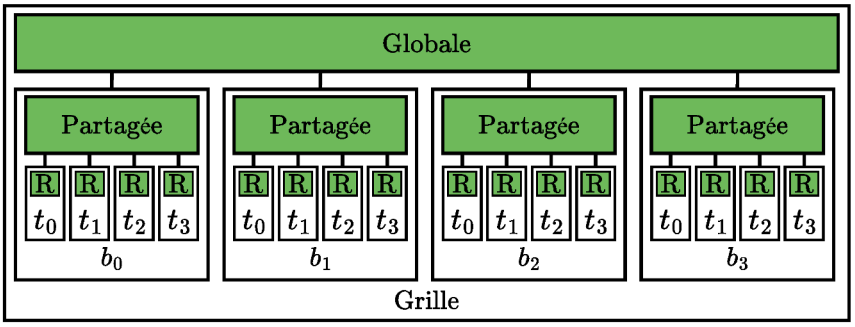


Figure 2.3 – Illustration de l'architecture logique d'un GPU. Celle-ci est hiérarchique (la grille contient des blocs b_i, qui contiennent des threads t_i) et met à disposition différents niveaux de mémoire accessibles en fonction du contexte (mémoire globale, partagée et registres).

primitives géométriques, leur usage n'est pas limité à l'affichage et de plus en plus d'algorithmes sont développés afin de tirer profit de leur puissance de calcul. Ce champ d'application, appelé *General-purpose Processing on Graphics Processing Units* (GPGPU), est central dans de très différents contextes comme en apprentissage machine ou pour des simulations moléculaires.

Comme illustré figure 2.2, l'architecture d'un GPU est construite autour d'un nombre très important d'unités de calcul (*e.g.* 16384 cœurs pour une NVIDIA RTX 4090) permettant l'exécution parallèle des fils d'exécution, ou *threads*, souhaités. Ces unités, à la différence d'un cœur de CPU, fonctionnent sur un modèle *Single Instruction, Multiple Threads* (SIMT). Lorsque le cœur charge une nouvelle instruction, plusieurs threads l'exécutent de façon concurrente et avec des données différentes. Ce paradigme est particulièrement intéressant puisqu'il permet de mutualiser les latences telles que les accès mémoires. Il est cependant très sensible aux divergences d'exécution notamment induites par des branches conditionnelles. Le paradigme de programmation des GPUs n'étant pas séquentiel, il est basé sur un vocabulaire spécifique. Dans cette thèse, nous utilisons celui du langage CUDA, lié aux GPU NVIDIA [NVIb], qui définit une architecte abstraite. Une fonction à exécuter de façon parallèle sur le GPU est appelée un *noyau* qui est configuré à partir d'un nombre de threads désiré. L'ensemble des threads est contenu dans une *grille*, elle-même décomposée en *blocs*, illustrés figure 2.3. Le *warp* représente l'unité d'exécution parallèle matérielle. Il est lui-même composé de quelques threads (*e.g.* 32 sur les cartes NVIDIA) théoriquement totalement parallèles et synchrones dans l'exécution de l'instruction actuelle par le cœur GPU. Un bloc est composé de plusieurs warps dont l'exécution est déterminée par leurs disponibilités. Ainsi, comme illustré figure 2.4, le cœur GPU diminue

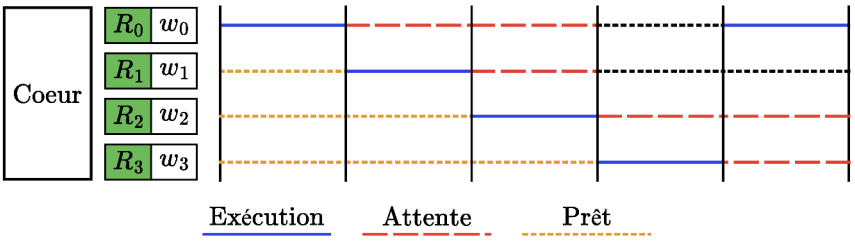


Figure 2.4 – Illustration du fonctionnement des warps. Ceux-ci disposent de 3 états déterminant leurs potentielles exécutions par le cœur. Certaines opérations peuvent placer un warp en attente plus longtemps que d'autres.

les latences en exécutant les instructions des warps en fonction de leurs disponibilités. Ainsi, lorsqu'un warp effectue une opération demandant plusieurs cycles, le cœur reste actif en exécutant les instructions des autres warps du bloc. Cela permet d'éviter que le cœur reste inoccupé et ainsi de bénéficier de meilleures performances.

À la différence des CPUs, dont les caches sont gérés de façon implicite en fonction des instructions du programme, les APIs GPUs permettent un accès explicite à certains niveaux de cache de leurs architectures, comme illustré figure 2.3. Ainsi, la mémoire est segmentée en trois grandes principales catégories. La mémoire globale est le niveau de plus grande taille dont les cartes modernes disposent de quelques dizaines de *Gigaoctets* (Go). Accessible par tous les threads de la grille, la mémoire globale est cependant la plus lente et très sensible à la façon avec laquelle les threads accèdent aux données pouvant impacter de façon importante les performances de lecture et d'écriture. La mémoire partagée est quant à elle un cache accessible par tous les threads d'un même bloc pouvant aller jusqu'à 48 *Kilo-octets* (Ko). Elle est au cœur de beaucoup de stratégies visant à réduire la latence d'accès à la mémoire globale à travers des schémas de coopération des threads du bloc. Enfin, chaque thread possède une quantité de registres dépendante du programme exécuté. Ces derniers sont propres à un thread, mais peuvent être partagés entre ceux d'un même warp. Lorsqu'un programme requête beaucoup de mémoire partagée et/ou de registres par threads, il diminue la possibilité du matériel à exécuter beaucoup de warp en même temps.

Les threads des warps partagent différentes propriétés et fonctionnalités. Par exemple, lorsqu'ils requêtent un accès à la mémoire globale, le cœur accède à un segment complet qui est mis à disposition du warp. Si toutes les requêtes sont localisées dans le segment accédé, aucune autre requête n'est nécessaire, réduisant ainsi la latence induite. Ces dernières années, les APIs de programmation GPU ont permis l'accès à des fonctionnalités de coopération entre threads d'un même warp. Celles-ci sont particulièrement intéressantes puisqu'elles permettent la mutualisation des registres des threads et un calcul coopératif [Col]. Les instructions dédiées permettent différentes fonctionnalités comme l'échange de données (`shfl_up`, `shfl_down`, `shfl_xor`), tester une valeur pour tous les threads (`all`, `any`), etc. Un exemple de l'utilisation de ces instructions est donné code source 2.1 pour la somme d'un ensemble de valeurs stockées dans les registres des threads. Cette opération, primitive de nombreux algorithmes, permet de ne pas avoir à explicitement mettre en cache les données afin de calculer la somme tout en profitant d'un accès à la valeur finale par tous les threads du warp.

Une autre composante importante des GPUs est la rastérisation matérielle. Celle-ci permet l'affichage de triangles, de lignes et de points de façon performante grâce à une implémentation

```
1  __device__ uint32_t warpSum( cg::thread_block_tile<32> warp, uint32_t value )
2  {
3      for ( uint32_t i = 1; i < warpSize; i *= 2 )
4          value += warp.shfl_xor( value, i );
5
6      // Tous les threads du warp obtiennent la valeur finale dans le registre de  value.
7      return value;
8  }
```

Code Source 2.1 – Exemple de code CUDA pour la somme coopérative d'un ensemble de valeurs stockées dans les registres des threads d'un warp.

matérielle et à partir d'une succession d'étapes programmables, appelées *shader*. Deux principaux shaders sont ainsi nécessaires à l'affichage : le vertex et le fragment shader. Le premier est destiné à transformer les positions des sommets des triangles en fonction des besoins de l'application tandis que le second est appelé pour chacun des éléments composant l'image et à l'intérieur de l'un des triangles. La programmabilité de ces étapes permet des usages très différents et des innovations adaptées à des contextes spécifiques. Notamment, l'affichage de triangle est parfois trop limitant pour certains types de géométries. Dans ce contexte, il est possible d'utiliser un *imposteur*, composé de deux triangles faisant face à la caméra. Ceux-ci, une fois rastérisés, permettent d'appeler le fragment shader sur leur surface complète et donc d'afficher une géométrie procédurale ou d'utiliser une stratégie différente d'affichage comme le lancer de rayon. En plus de ces composantes, les GPUs disposent depuis quelques années d'unités de calcul dédiées au lancer de rayon. Celles-ci sont principalement conçues pour la construction et le parcours de la structure accélératrice ainsi que pour le calcul d'intersection entre la géométrie et le rayon. Grâce à leurs fonctionnalités, elles contribuent à de meilleures performances pour ce type de stratégie d'affichage qui est compatible avec l'illustration de haute qualité.

L'accélération matérielle des GPUs est particulièrement utilisée dans les domaines cherchant une interactivité importante comme la visualisation scientifique. Elle est notamment indispensable pour afficher de grandes quantités de données [Zel+23]. Plus particulièrement, leur paradigme se prête aux données moléculaires d'une façon similaire aux algorithmes traitant des maillages qui peuvent bénéficier d'une parallélisation par sommet. Les calculs effectués sur une protéine peuvent par exemple être réalisés en associant un thread par atome permettant ainsi une mise à l'échelle performante.

2.2 Surfaces moléculaires

La surface de van der Waals, la Surface Accessible au Solvant (SAS) et la Surface Exclue au Solvant (SES) sont au centre de multiples études en biochimie. Ces représentations pouvant être utilisées pour des calculs scientifiques, mais aussi la visualisation de structures, de nombreux travaux ont été dédiés à leur construction. C'est cependant la SES qui est visée de façon générale en raison de ses potentielles utilisations variées. Cette partie décrit donc tout d'abord

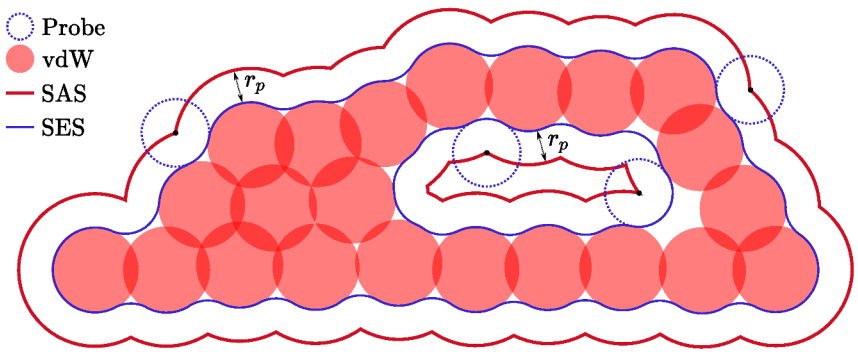


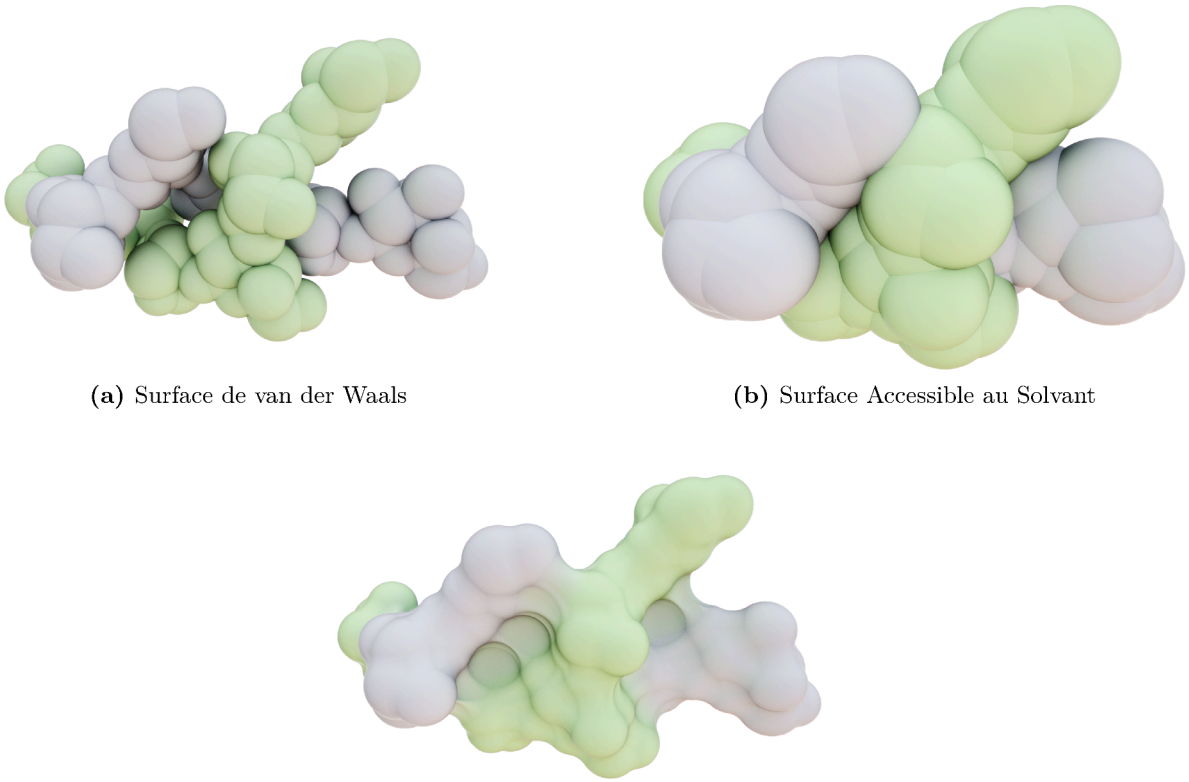
Figure 2.5 – Schéma structurel des surfaces moléculaires sur la base du trajet de la sphère du probe sur la surface de van der Waals (vdW). Son centre décrit la SAS et son interaction avec la surface de van der Waals décrit la SES.

les caractéristiques formelles des surfaces moléculaires principales (section 2.2.1), puis présente les travaux majeurs pour leurs calculs (section 2.2.2 et section 2.2.3).

2.2.1 Préambule

La surface de van der Waals joue un rôle fondateur pour les modèles de surfaces moléculaires. Notamment, c'est sur sa base que sont caractérisées la SAS et la SES dans les travaux les ayant introduits par Richards [Ric77] et Greer et Bush [GB78]. Dans cette partie, nous rappelons les caractéristiques de la SES.

L'union des sphères centrées sur les atomes $a_i = (c_i, r_i) \in \mathbb{R}^3 \times \mathbb{R}$ d'une molécule $\mathcal{M}$, composés d'un centre c_i et de rayon de van der Waals r_i, décrit la surface de van der Waals. Un *probe* est défini par son rayon r_p comme la sphère englobante de la molécule du solvant souhaité. La plupart du temps, le solvant considéré est une molécule d'eau de rayon $r_p = 1.4$Å [DAr78]. La SAS et la SES sont caractérisées par toutes les positions de la sphère du probe en contact avec la surface de van der Waals sans l'intersecter, comme illustré figure 2.5. On parle alors souvent du probe roulant sur celle-ci et dont le centre décrit la SAS. La surface obtenue peut ainsi être vue de façon alternative comme une surface de van der Waals dilatée dont les rayons des atomes sont donnés par $r_i + r_p$. La SES est, quant à elle, décrite par la surface d'interaction entre le probe et la surface de van der Waals. Les primitives de la SAS et de la SES sont liées et l'une peut donc être définie à partir de l'autre. Les trois surfaces sont illustrées figure 2.6.



(a) Surface de van der Waals

(b) Surface Accessible au Solvant

(c) Surface Exclue au Solvant

Figure 2.6 – Illustration des principales surfaces moléculaires.

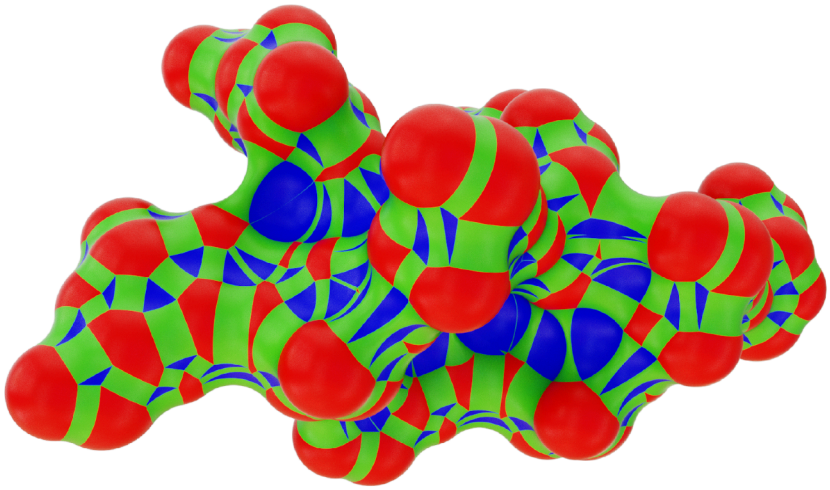
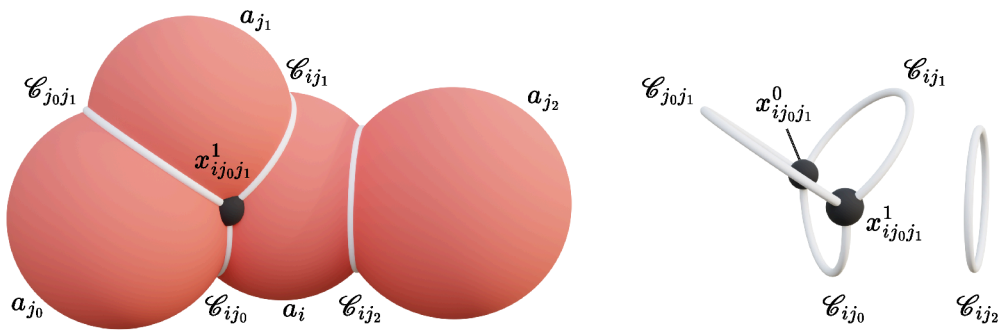


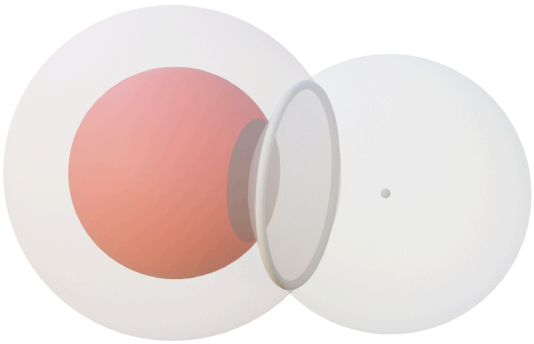
Figure 2.7 – Illustration des patchs de la SES caractérisants, une fois assemblés, la surface d'une petite protéine (PDB : 1AGA).

Différentes caractérisations de la SES ont été proposées depuis son introduction par Richards [Ric77]. Notamment, elle est caractérisée comme un ensemble de primitives par Connolly, souvent appelées *patchs*, qui, une fois assemblées, donnent la surface, comme illustré figure 2.7. Une conséquence directe de la formulation de la SES à partir du probe permet de définir ces patchs à partir de primitives de la SAS, illustrées figure 2.8. Ces dernières sont issues de l'intersection entre les sphères de la SAS, centrées sur chacun des centres des atomes c_i et de rayon $r_i + r_p$. L'intersection de deux sphères de la SAS associées aux atomes a_i et a_j, si elle existe, est un cercle de la SAS noté $\mathscr{C}_{ij}$. Lorsque trois cercles de la SAS $\mathscr{C}_{ij}$, $\mathscr{C}_{ik}$ et $\mathscr{C}_{jk}$ s'intersectent, ils produisent deux points appelés intersections de la SAS et notés x^0_{ijk} et x^1_{ijk}, et un point s'ils sont exactement tangents. Comme illustré figure 2.9, Les patchs de la SES sont de trois types basés sur les éléments de la SAS :

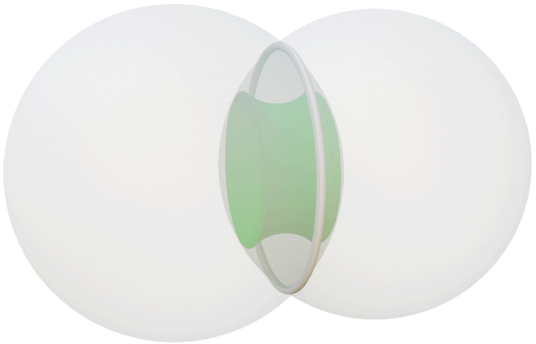


(a) Surface Accessible au Solvant (SAS) **(b)** Structure

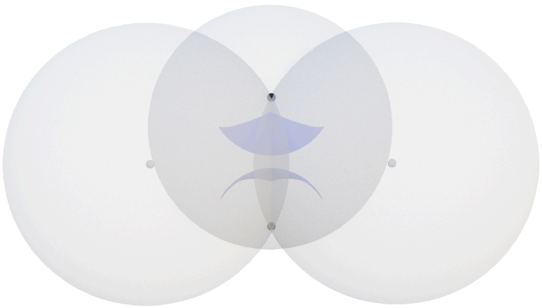
Figure 2.8 – Illustration des deux types de primitives de la SAS : les cercles (en blanc) et les intersections (en noir). Ils sont tous les deux dérivés des intersections entre les sphères de la SAS et décrivent sa structure.



(a) Patch sphérique convexe (en rouge)



(b) Patch toroïdal (en vert)



(c) Deux patchs sphériques concaves (en bleu)

Figure 2.9 – Illustration du découpage par patch de la SES et les configurations de la SAS liées (en gris et transparent). Les primitives de la SAS décrivent un chemin possible pour le probe associé à un type de patch.

— *Sphérique convexe* : chaque sphère de la SAS contribuant à sa surface permet deux degrés de liberté au probe. Le patch de la SES correspondant est alors sphérique et convexe (figure 2.9a) ;
— *Toroïdal* : les cercles de la SAS $\mathscr{C}_{ij}$ décrivent un chemin pour le probe avec un seul degré de liberté. Son interaction avec la surface de van der Waals résulte en un patch de la SES sous la forme de l'intérieur d'un tore (figure 2.9b) ;
— *Sphérique concave* : un probe positionné sur une intersection de la SAS est tangent à trois sphères de van der Waals et est immobile. Sa surface d'interaction avec la surface de van der Waals est donc caractérisée par un triangle sphérique qualifié patch concave de la SES (figure 2.9c).

Ces patchs permettent ainsi de complètement caractériser la SES et représentent les fondations d'algorithmes de construction. Une étude alternative présentant certains avantages supplémentaires a été plus récemment proposée [QS16] et, en tant que base de nos travaux, sera présentée en détail chapitre 3.

Sur la base des études de la SES, des approches variées ont été développées afin de la construire efficacement. Nous présentons donc dans les parties suivantes les méthodes principales visant son calcul. Celles-ci peuvent être séparées en deux catégories : les méthodes approximantes et les méthodes de construction analytique. Tandis que les premières visent à produire une approximation de la SES permettant une construction rapide, elles sont couramment contraintes

par un compromis entre niveau de détail et consommation mémoire. À l'inverse, les méthodes analytiques sont basées sur une construction topologique de la structure de la SES, mais requièrent généralement des calculs plus importants.

2.2.2 Méthodes approximantes et représentations alternatives

La SES étant utilisée pour visualiser de grandes structures, mais aussi des séquences animées de simulation, les performances de calcul représentent un enjeu particulièrement important. Cependant, le nombre de patchs de la SES augmentant linéairement avec le nombre d'atomes de la molécule visée, des méthodes ont été développées visant une sensibilité moins importante à ce facteur. Celles-ci peuvent être rassemblées en deux catégories : les méthodes approximantes visant une représentation approchée de la SES par sa discrétisation, et les représentations alternatives s'approchant des caractéristiques de la SES tout en étant souvent plus simple à construire.

Représentations approximées

La définition de la SES, à partir de la SAS et de la surface de van der Waals, permet de la discrétiser en classifiant tout point de l'espace comme à l'intérieur ou à l'extérieur de sa structure et ainsi de définir un champ scalaire. Cette stratégie a été utilisée à de nombreuses reprises [CCW06 ; Yu09 ; LBH14 ; Her+17] notamment pour sa simplicité d'implémentation, mais aussi de parallélisation permettant une construction rapide. Comme illustré figure 2.10, ces méthodes se basent sur un champ de distance signée. Ainsi, un ensemble uniforme de points est sélectionné puis classifié en fonction de sa proximité avec la SES. Trois catégories sont définies en fonction de la distance du point échantillonné x_g avec l'atome le plus proche $a_i = (c_i, r_i) \in \mathcal{M}$, le rayon du probe r_p et le rayon d'échantillonnage r_g :

- $||x_g - c_i|| - r_i > r_p$, $\forall a_i \in \mathcal{M}$: la distance entre le point et la surface de van der Waals est supérieure au rayon du probe. Le probe n'intersecte pas la surface de van der Waals, le point est donc en dehors de la SES et sa distance avec elle est au moins de r_g ;
- $\exists a_i \in \mathcal{M}$ tel que $||x_g - c_i|| - r_i < -r_g$: le point est localisé à l'intérieur de la surface de van der Waals. Le probe ne peut être centré sur le point, il est donc à l'intérieur de la SES et sa distance avec elle est au maximum de $-r_g$;

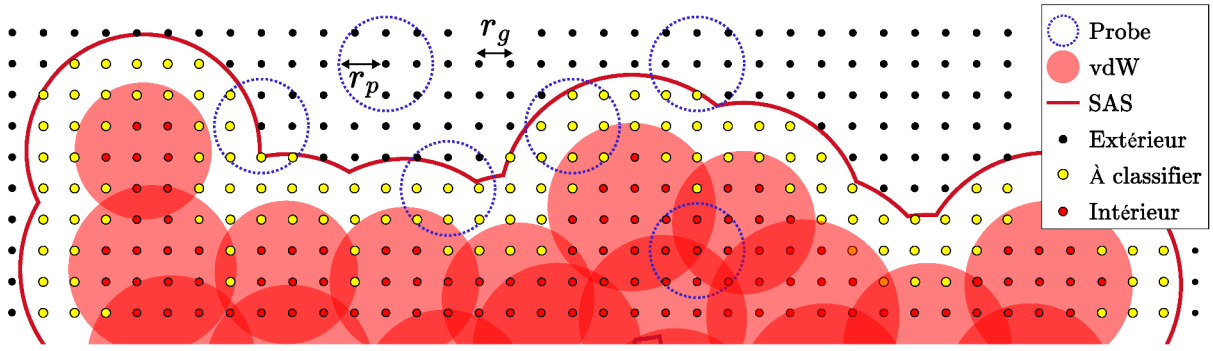


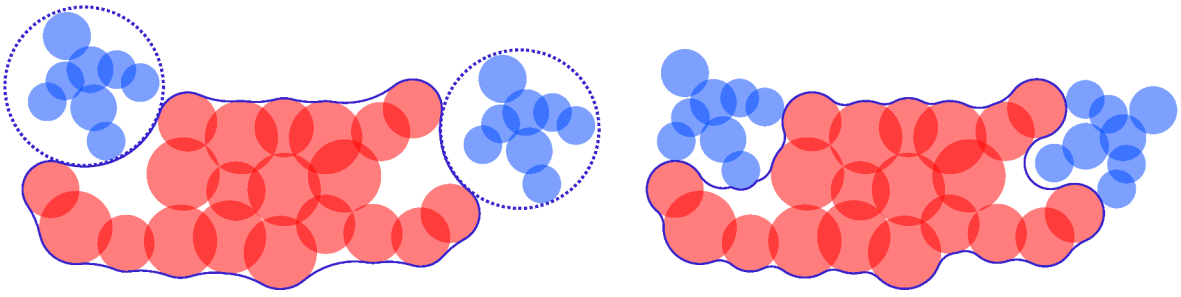
Figure 2.10 – Illustration du processus d'échantillonnage de l'espace pour la construction de la SES. La proximité du point échantillonné avec la surface de van der Waals permet de définir deux zones respectivement à l'intérieur (rouge) et à l'extérieur (noir) de la SES. La frontière (jaune) entre ces deux ensembles contient donc la SES. Il reste alors à assigner en chaque point échantillonné la distance à la surface.

— $(\exists a_i \in \mathcal{M}$ tel que $||x_g - c_i|| < r_i + r_p) \wedge (||x_g - c_j|| - r_j > -r_g,\ \forall a_j)$: le point est localisé dans une zone proche de la SES.

Dans le dernier cas, la distance avec la SES peut être approchée de façon plus précise en utilisant par exemple le point échantillonné extérieur le plus proche [Her+17]. La structure volumique en résultant peut ensuite être maillée avec des stratégies classiques comme l'algorithme des *Marching Cubes* [LC87] ou affichée avec un lancer de rayon [Har96].

Ce type de méthode a permis différentes avancées liées au calcul de la SES. Notamment, sa simplicité d'implémentation est associée à de bonnes performances, ainsi qu'à une bonne configurabilité en fonction du matériel à disposition et de la qualité de la surface souhaitée. En effet, la fréquence d'échantillonnage permet un compromis intéressant entre qualité d'affichage, vitesse de calcul et capacité mémoire. Ainsi, elle est distribuée dans des logiciels de visualisation moléculaire [Pet+20], mais aussi des bibliothèques de calcul [MKB19], qui tirent profit de sa configurabilité afin de l'adapter au plus grand nombre d'utilisateurs. Ces caractéristiques permettent de plus un raffinement progressif de la qualité de la surface [Her+17]. Il est alors possible de calculer en premier lieu une surface de basse qualité, et d'augmenter itérativement la fréquence d'échantillonnage. Ainsi, l'utilisateur bénéficie d'une visualisation de la séquence de façon interactive, et d'une bonne qualité de la surface si celle-ci est immobile.

La SES est définie à partir de la sphère englobante de la molécule de solvant. Cependant, lorsqu'on souhaite visualiser l'accessibilité d'une plus grande molécule, comme un ligand, la sphère donne une très mauvaise approximation. Dans ce contexte, l'échantillonnage de l'espace permet de représenter avec plus de précision la forme du solvant que les définitions analytiques. Ainsi Lindow et al. [LBH14] ont présenté la Surface Exclue au Ligand (LES) comme une généralisation de la SES (figure 2.11). La LES est construite à partir du même échantillonnage de l'espace que la SES. Cependant, une fois l'espace classifié, on teste en chacun des points proches de la frontière si le ligand peut être positionné sans intersecter la surface de van der Waals. Cette dernière étape est intrinsèquement discrète et achevée à partir d'un ensemble de conformations et rotations possibles du ligand. La surface qui résulte de ce processus bénéficie ainsi d'une caractérisation plus précise de l'espace qui peut être atteint par le ligand et particulièrement vers les parties étroites et les potentielles cavités. Cependant, les étapes additionnelles nécessaires



(a) Surface Exclue au Solvant (SES) **(b)** Surface Exclue au Ligand (LES)

Figure 2.11 – En construisant la surface moléculaire par échantillonnage de l'espace, on peut utiliser un probe dont la forme est dynamique. Ainsi, on représente d'une façon plus précise la surface exclue à une protéine dont la structure peut être quelconque. (a) La sphère englobante d'un ligand est de grande taille, ce qui produit une SES avec peu de détail et représentant mal l'espace véritablement exclu. (b) En discrétisant les potentielles configurations d'amarrage, on obtient une surface qui décrit d'une meilleure façon l'accessibilité du ligand.

à la LES impactent drastiquement ses performances de calcul la rendant moins adaptée à de moyennes et grandes structures ou des séquences d'animation.

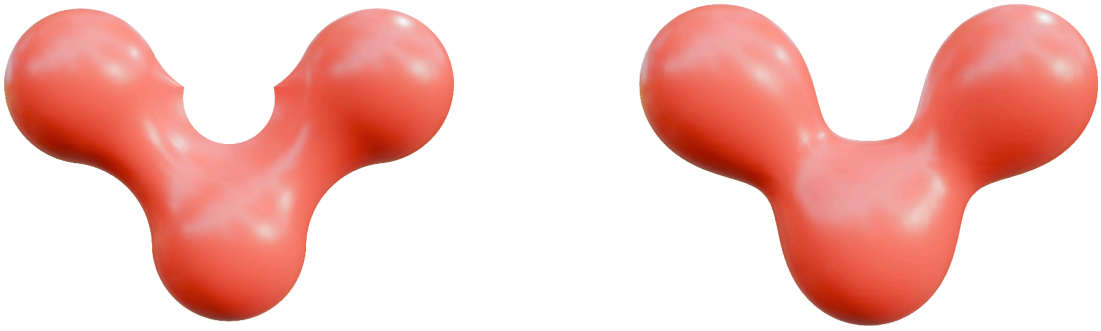
Représentations surfaciques alternatives

Les caractéristiques de la SES pouvant contraindre de façon importante les méthodes de calcul, d'autres représentations surfaciques ont été développées afin de bénéficier de meilleure performance sans proposer toutes les fonctionnalités de la SES. Notamment, cette dernière est principalement visuellement caractérisée par ses patchs toroïdaux et concaves accentuant la proximité entre les atomes. Ces patchs nécessitant le calcul des cercles et intersections de la SAS, ils représentent un coût important. La surface gaussienne, aussi appelée Meta-Balls [Bli82], est alors souvent utilisée comme alternative à la SES. Celle-ci est définie comme une fonction implicite donnée par la somme des densités de fonctions gaussiennes paramétrées à partir des sphères de van der Waals

$$f(x) \coloneqq \sum_i e^{\frac{-||x-c_i||^2}{r_i^2}}.$$

Cette formule a pour principal avantage d'être moins coûteuse à évaluer et plus compacte que celle de la SES tout en étant cependant plus grossière, comme illustré figure 2.12. Sa caractéristique compacte a ainsi permis des optimisations intéressantes à partir de structures accélératrices dédiées [Gou+10], ou en utilisant la rastérisation matérielle des GPUs [Bru19]. Cette dernière stratégie permet notamment l'affichage en temps réel de la surface, sans précalcul.

D'autres surfaces alternatives de la SES ont été proposées tout en offrant plus de fonctionnalités que la surface gaussienne. Ainsi, la Molecular Skin Surface (MSS) est basée sur un diagramme de Puissance [Ede99] et présente l'avantage d'être continue, à la différence de la SES qui présente des singularités géométriques. De plus, elle peut aussi être représentée à partir de patchs associés à des surfaces quadriques et compatibles avec son affichage par lancer de rayon [CLM08 ; Lin+10]. Cette surface reste cependant moins populaire que la SES pour la visualisation moléculaire et le calcul de quantité.



(a) Surface Exclue au Solvant (SES) **(b)** Surface gaussienne

Figure 2.12 – Comparaison entre la SES (a) et la surface gaussienne (b). Cette dernière est très simple à calculer grâce à son modèle compact, mais présente moins de fonctionnalités. Notamment, elle conserve une représentation explicite de la proximité entre atomes, mais perd les détails de la SES que l'on peut observer sur les patchs toroïdaux et concaves.

Malgré les nombreux avantages de la construction approchée de la SES évoqués dans cette partie, ce type de méthode est directement lié à un compromis entre qualité de la surface, performance, et utilisation de la mémoire. Ainsi, en tant qu'approximations ou surfaces alternatives, elles ne présentent pas le même niveau de détails que la SES analytique ce qui les rend peu compatibles avec le calcul de la surface avec un haut niveau de détail, notamment nécessaire pour l'illustration de haute qualité.

2.2.3 Méthodes de construction analytique

La caractérisation de la SES telle que présentée section 2.2.1 permet une représentation compacte en stockant uniquement les primitives de la SAS suffisantes pour représenter les patchs correspondants. Ainsi, de nombreuses méthodes de construction analytiques de la SES ont été proposées ne souffrant pas des compromis liés à la qualité et à la consommation mémoire des méthodes approximantes. Plusieurs algorithmes fondateurs [VBW94 ; SOS96 ; TA96] ont été proposés à la suite des travaux de caractérisation de Connolly. Ces derniers permettent un calcul structurel de la SES à partir de laquelle une représentation de la géométrie peut être dérivée sous la forme d'un maillage ou autre. Les méthodes actuelles sont ainsi principalement basées sur Reduced Surface [SOS96] et Contour-Buildup [TA96].

Reduced Surface [SOS96], proposé par Sanner et al., fait partie des algorithmes de construction de la SES les plus populaires. Notamment, il est distribué avec le programme MSMS permettant le calcul d'une triangulation, de l'aire et du volume de la SAS et de la SES. Permettant un calcul simple de ces informations sans réimplémentation de Reduced Surface, il est toujours utilisé par des logiciels de visualisation moléculaire [HDS96 ; Pet+04 ; SD20]. Comme illustré figure 2.13, Reduced Surface construit la SAS et la SES à partir de leur structure combinatoire, souvent comparée à une α-shape [EKS83]. Celle-ci est composée de sommets définis par le centre des atomes, d'arêtes liant les sommets si deux sphères de la SAS ont une intersection non vide, et enfin une face dans le cas de trois sphères. L'algorithme commence en trouvant une première face valide et donc un triplet de sphère de van der Waals sur lesquelles le probe peut être immobile. À partir de cette face, on explore ses arêtes afin de trouver de nouvelles faces de façon récursive. Lorsque toutes les arêtes ont été explorées, un sous-ensemble de la structure a été calculé. On cherche alors pour chacun des atomes a_i du sous-ensemble si un autre atome a_j, non lié à a_i

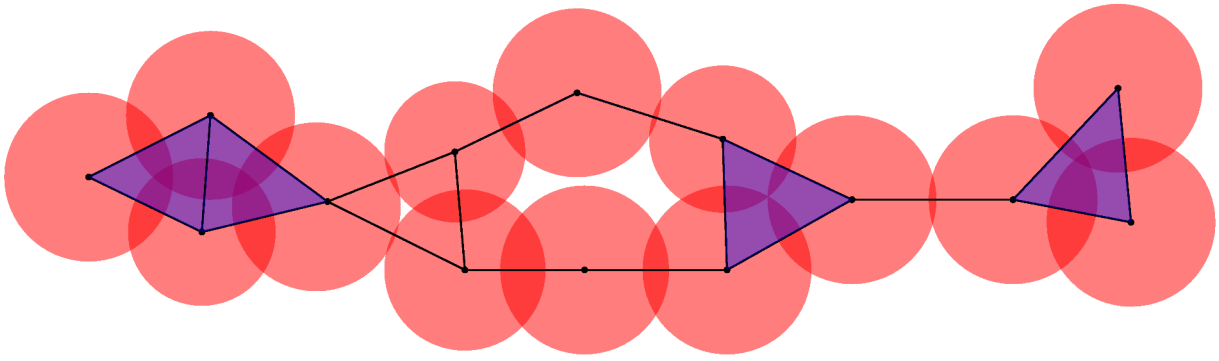
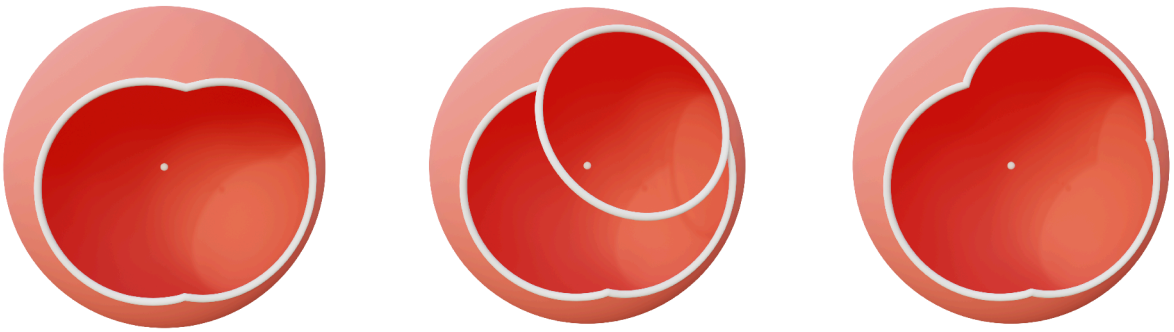


Figure 2.13 – Illustration de la structure combinatoire au cœur de la construction de la SAS et de la SES avec Reduced Surface [SOS96] et comparable à une α-shape. Les centres des sphères représentent ses sommets (points noirs). Si deux sphères ont une intersection non vide et visible, il existe une arête entre elles. De la même façon, si trois sphères ont une intersection non vide et visible, il existe une face entre elles.



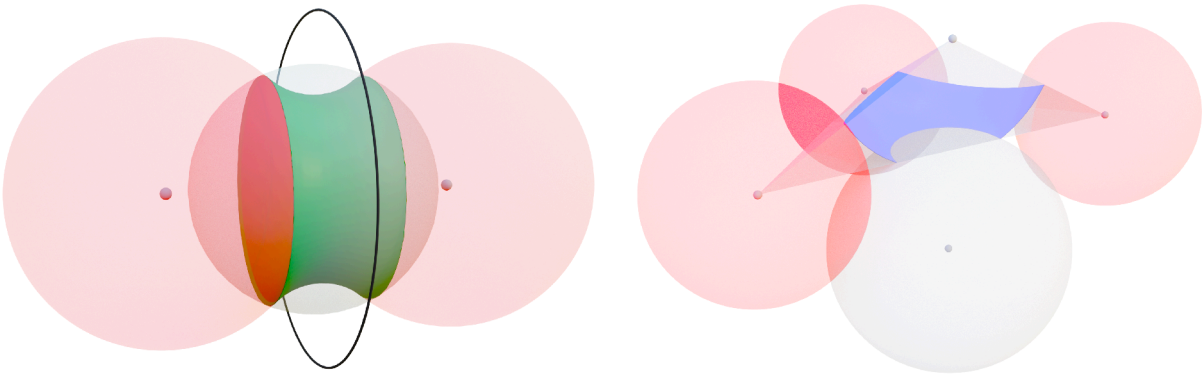
(a) Contour original **(b)** Insertion d'un nouveau cercle **(c)** Contour modifié

Figure 2.14 – Illustration du processus de modification du contour d'un patch convexe P_+ avec Contour-Buildup [TA96]. (a) Le contour est composé de l'union de deux cercles. (b) On ajoute un nouveau cercle de la SAS au contour. (c) Le nouveau cercle est intégré au contour.

par une arête, peut former une nouvelle arête avec celui-ci. Si un nouveau voisin est trouvé, il tente enfin de trouver un autre atome a_k pouvant participer à une face. Si une face est trouvée, on recommence jusqu'à trouver tous les sous-ensembles constituant la structure, sinon, l'arête correspond à un cercle de la SAS sans intersection. À la fin du traitement, chaque sommet est associé à un patch convexe, chaque arête à un patch toroïdal et chaque face à un patch concave. La structure résultante peut enfin être maillée en triangulant chacun des patchs associés aux sommets, arêtes et faces. Malgré sa popularité, Reduced Surface est intrinsèquement séquentiel, ce qui freine son utilisation, notamment dans des environnements parallèles.

Contour-Buildup [TA96] est proposé par Totrov et Abagyan la même année que Reduced Surface. À sa différence, ce second algorithme est directement basé sur la construction des primitives de la SAS. Celle-ci est effectuée à partir des contours des sphères de la SAS définis par leurs cercles et permettant de dériver leurs connectivités. Cette stratégie est particulièrement intéressante puisqu'elle permet un calcul indépendant par atome et donc plus facilement parallélisable. Les bordures des sphères de la SAS sont ainsi stockées dans une structure représentant l'union de leurs cercles de la SAS sous la forme d'un ensemble de listes d'arcs de cercle. Un nouveau cercle est inséré de façon itérative à la structure pour chacune des sphères voisines intersectant la sphère traitée, comme illustré figure 2.14. Cette dernière étape est permise par des prédicats permettant de tester si une intersection est possible entre chaque arc de la structure et le nouveau cercle. Dans ce cas, le nouvel arc de cercle est mis à jour en calculant ses intersections. Une fois que tous les arcs connus ont été testés, les nouvelles intersections sont triées de façon angulaire afin de reconstruire les nouveaux arcs et les ajouter à la structure. À la fin du traitement, il est possible d'extraire les primitives de la SAS correspondantes à partir des contours des sphères et ainsi de construire les patchs de la SES.

Une fois la description analytique et structurelle de la SES calculée à partir de la SAS, il est nécessaire de définir une stratégie pour son affichage. Tandis que les travaux fondateurs ont proposé des processus dédiés pour le maillage des patchs [Con83 ; VBW94 ; TA96 ; SOS96 ; AE96 ; LB02 ; Ryu+ ; RCK09] permettant d'obtenir un échantillonnage de la surface plus proche que les méthodes approximées, ce type de stratégie est tout de même contraint par un compromis entre consommation mémoire et qualité de la surface. Ainsi, Krone et al. propose d'afficher les patchs de la SES de façon individuelle et par lancer de rayon sur GPU en utilisant des



(a) Patchs toroïdaux **(b)** Patchs concaves

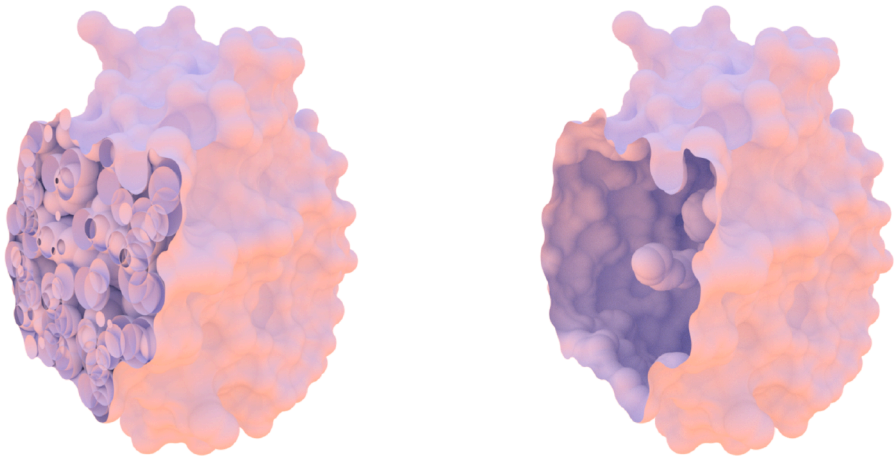
Figure 2.15 – Illustration des primitives utilisées pour le rendu rapide de la SES extérieure et des formes les bornant [KBE09]. (a) L'intérieur d'un tore peut être utilisé pour l'affichage d'un patch toroïdal en conservant uniquement sa partie interne avec une sphère. (b) Un triangle sphérique représente un patch concave qui, en cas d'intersection avec des patchs voisins, est aussi intersecté par la sphère de ceux-ci.

imposteurs [KBE09]. Cette stratégie, déjà utilisée pour d'autres représentations basées sur des quadriques [Sig+06], permet de bénéficier d'un affichage rapide tout en obtenant une surface de la meilleure qualité possible au pixel près. Les patchs convexes sont affichés à partir d'une sphère, les patchs toroïdaux à partir d'une partie d'un tore et les patchs concaves à partir d'un triangle sphérique. Cette méthode bénéficie ainsi d'une accélération matérielle tout en ne subissant pas les mêmes contraintes que les surfaces approximées. Elle est cependant limitée à un affichage extérieur de la surface. Comme illustré figure 2.15, les patchs toroïdaux sont affichés avec l'intérieur d'un tore complet tandis que les patchs convexes utilisent des sphères complètes, ne prenant ainsi pas en compte les bordures délimitant la surface. Elle permet un affichage rapide de la surface sans autoriser l'affichage avec de la transparence, notamment intéressant pour la visualisation de cavités. Enfin, Parulek et Viola proposent une représentation des patchs de la SES à partir de fonctions distance signées permettant l'affichage sans précalcul et par lancer de rayon [PV12]. Elle nécessite cependant une évaluation à la volée plus coûteuse que les quadriques utilisées par Krone et al. basée sur le voisinage de chaque atome.

Deux implémentations parallèles ont été proposées à la suite de la méthode de Krone et al. [KBE09] qui est basée sur une implémentation séquentielle de Reduced Surface. La première utilise le processus de Reduced Surface en calculant les intersections possibles de façon parallèle par atomes [KDE10]. Afin de réduire les temps de calcul, les traitements sont basés sur l'identification des données qui seront à priori affichées à l'écran, pouvant provoquer des instabilités à l'affichage. La deuxième stratégie [KGE11], basée sur Contour-Buildup, calcule en premier lieu les voisinages des sphères de la SAS, équivalents à tous les cercles de la SAS possibles, et calcule de façon parallèle à chacun des cercles les intersections avec tous les autres voisins de son atome [TA96]. Ce traitement permet ainsi d'obtenir toutes les intersections de la SAS, mais perd cependant l'information des arcs de la SAS, à l'origine des patchs toroïdaux. Ce dernier aspect peut ainsi être vu comme une optimisation, puisque dans la méthode d'affichage visée, les patchs issus de cercles et d'arcs de cercle sont traités de la même façon. Même si cette dernière méthode permet de très bonnes performances, elle est cependant limitée de façon importante par sa consommation mémoire. Ainsi, Schäfer et Krone [SK19] proposent de calculer toutes les positions de probe possibles et de tester si elles n'intersectent pas un atome voisin. Les cercles de la

SAS sont ensuite déterminés à partir de la structure combinatoire des positions de probes valides. Même si cette stratégie utilise de façon intéressante les capacités de calcul des GPUs, elle requiert beaucoup plus de traitement que les autres et est plus lente que la méthode de Krone et al. [KGE11]. Elle permet cependant de calculer la surface de plus grosses protéines puisqu'elle nécessite 10 fois moins de mémoire que l'implémentation GPU de Contour-Buildup sur une protéine de 50000 atomes [KGE11].

Le calcul et l'affichage de la SES complète (figure 2.16b) de façon efficace impliquent de retrouver les bornes des patchs convexes constituées de la projection des cercles de la SAS, mais aussi de reconstruire tous les arcs de la SAS, en comparaison avec la SES extérieure uniquement (figure 2.16a). Cette dernière étape est particulièrement cruciale dans un environnement parallèle puisqu'un arc de la SAS est borné par deux intersections pouvant être créées par des threads différents. Kauker et al. [Kau+13] affinent l'affichage de Krone et al. [KBE09] en reconstruisant les patchs de la SES à partir d'opération ensembliste et de la structure utilisée pour Reduced Surface. Cette stratégie permet l'affichage avec transparence de la SES, mais n'est pas compatible avec celui des potentielles cavités comme expliqué par Jurcik et al. [Jur+16]. Ces derniers proposent de calculer explicitement la forme complète des patchs concernés afin d'obtenir une visualisation cohérente de la surface. Les arcs de la SAS sont alors obtenus à partir d'un hash des intersections, obtenues avec Contour-Buildup, ce qui permet ainsi de les retrouver par cercle. La liste d'intersections de chaque cercle est ensuite triée de façon angulaire par rapport à son axe. Les cycles dans le graphe composé des intersections et arcs de la SAS sont ensuite identifiés puisqu'ils composent les bordures des patchs convexes. Cette stratégie permet un calcul complet de la SES, mais est cependant limitée par ses performances sur de grandes structures. D'une façon similaire, Rau et al. [Rau+19] proposent une étude pour l'affichage de la SES de très grandes structures sur CPU. Celle-ci se base sur Contour-Buildup et introduit plusieurs optimisations pour l'affichage de la SES complète à partir de ses patchs. Même si leur méthode permet l'affichage de la SES de



(a) Surface extérieure [KBE09] **(b)** Surface complète

Figure 2.16 – Comparaison entre la visualisation de la SES proposée par Krone et al. [KBE09] limitée à un affichage extérieur, mais permettant de meilleures performances que l'affichage de la surface complète. (a) Afficher les patchs toroïdaux provenant d'arc de cercles comme des cercles complets et les patchs convexes à partir de sphères complètes donne lieu à une cavité de la surface remplie de géométries non désirées. (b) En utilisant les patchs de la caractérisation, l'intérieur de la surface est correctement défini.

grandes protéines, ses performances de construction sont inférieures d'un ordre de magnitude à celles de la méthode de Krone et al. [KGE11]. De plus, puisque les structures moléculaires sont répétitives, elle utilise des instances à l'affichage nécessitant moins de mémoire, mais n'étant pas compatible avec des structures issues de simulations moléculaires.

Les stratégies présentées dans cette partie permettent de calculer la SES dans des contextes variés soutenant la visualisation. Elles sont cependant contraintes par la possibilité de calculer la surface complète de façon performante et plus particulièrement celles de grandes protéines. Ces dernières contraignent les algorithmes existants qui nécessitent une importante quantité de mémoire afin de proposer un haut niveau de détail. Ainsi, la plupart des logiciels se basent sur des méthodes approximées et ne permettant pas une visualisation fine de la géométrie à de grandes échelles rapidement. De plus, même si les surfaces moléculaires sont à la base de nombreuses études, elles représentent une formulation restreinte de l'espace qui est dépendante du probe. D'autres caractérisations structurelles de protéines à partir de diagramme de Voronoï ont ainsi été présentées permettant notamment des analyses comprenant des informations plus générales que celles obtenues avec les surfaces moléculaires. Ainsi, nous présentons leurs caractéristiques dans la section suivante.

2.3 Diagrammes de Voronoï et généralisations

Les caractérisations de la SAS et de la SES sont directement liées aux diagrammes de Voronoï et leurs généralisations [Aur91]. Ces derniers caractérisant les points en fonction de leurs distances à un ensemble de sites, différents travaux ont montré qu'ils peuvent être vus comme une généralisation de la SAS et de la SES, indépendante du rayon du probe [MJK16 ; DQS20]. Le diagramme d'Apollonius, ou diagramme de Voronoï additivement pondéré, est la généralisation des diagrammes de Voronoï la plus utilisée pour la construction de surfaces [Joo+05 ; MJK16 ; Kim+19] et l'analyse de structures [GPF97 ; LBH11 ; Lin+13] moléculaires. Ainsi, nous introduisons dans cette partie certaines des caractéristiques notables des diagrammes de Voronoï et leurs liens avec les surfaces moléculaires (section 2.3.1) qui sont suivis par les analyses et méthodes de construction du diagramme d'Apollonius (section 2.3.2). La parallélisation de la construction des diagrammes de Voronoï est nécessaire pour de nombreuses applications nécessitant de grandes échelles, comme en biochimie, mais aussi en cosmologie ou en science des matériaux. Nous présentons donc enfin les méthodes principales de construction parallèles des diagrammes de Voronoï (section 2.3.3).

2.3.1 Préambule

Définitions

Les diagrammes de Voronoï sont des partitions de l'espace associant chaque point de celui-ci à son *site* le plus proche. Ils sont ainsi définis par un ensemble de sites $\mathcal{P} = \{p_i\}_{i=1}^{n}$, avec $p_i \in \mathbb{R}^d$ et d'une fonction distance euclidienne entre un point x et un site $d(p_i,\ x) \coloneqq ||p_i - x||$. Une *cellule* de Voronoï $\mathcal{V}(p_i)$ d'un site p_i est alors donnée par la minimisation de la fonction $d(p_i,\ x)$

$$\mathcal{V}(p_i) \coloneqq \{x \ \mid \ d(p_i,\ x) \leq d(p_j,\ x),\ \forall p_j \in \mathcal{P} \setminus \{p_i\}\}\,.$$

Comme illustré figure 2.17, celle-ci est définie par un polytope convexe contenant p_i et borné par un ensemble de faces de $d - k$ dimensions, $k \in \{1, \ldots, d\}$, nommées en fonction de leur degré :

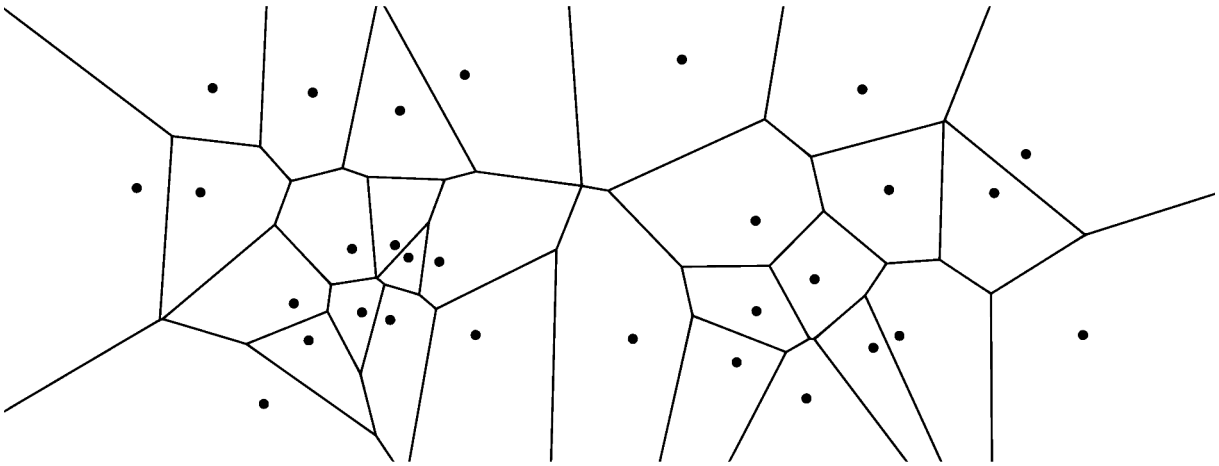


Figure 2.17 – Illustration du diagramme de Voronoï d'un ensemble de points dans $\mathbb{R}^2$.

bisecteur $(d-1)$, *trisecteur* $(d-2)$, *quadrisecteur* $(d-3)$, etc. Les bisecteurs de Voronoï Π_{ij} sont des hyperplans de dimensions $d-1$ et caractérisés par les points à égale distance des deux sites

$$\Pi_{ij} \coloneqq \{x \mid d(p_i,\, x) = d(p_j,\, x) \leq d(p_k,\, x),\; p_k \in \mathcal{P} \setminus \{p_i, p_j\}\}\,.$$

Ils représentent ainsi la frontière entre deux demi-espaces Π_{ij}^{+} et Π_{ij}^{-} tels que $p_i \in \Pi_{ij}^{+}$ et $p_j \in \Pi_{ij}^{-}$. Cette dernière définition permet une caractérisation alternative de la cellule à partir de l'intersection de tous les demi-espaces Π_{ij}^{+} contenant p_i

$$\mathcal{V}(p_i) \coloneqq \bigcap_{i \neq j}^{n} \Pi_{ij}^{+}.$$

La triangulation de Delaunay est la structure combinatoire des diagrammes de Voronoï et a fait l'objet de nombreuses études visant une construction exacte. La géométrie du diagramme de Voronoï en elle-même est cependant aussi au cœur de nombreuses méthodes, notamment pour sa caractérisation de l'espace.

Les diagrammes de Puissance, aussi appelés diagrammes de Laguerre-Voronoï, sont une généralisation des diagrammes de Voronoï. Ainsi, ils sont basés sur un ensemble de sites pondérés $\mathcal{S} = \{s_i\}_{i=1}^{n}$, où un site pondéré $s_i = (p_i, r_i) \in \mathbb{R}^d \times \mathbb{R}$ est composé d'une position p_i et d'un poids r_i. Les diagrammes de Puissance utilisent la distance quadratique pondérée $\delta^{(2)}(s_i,\, x) \coloneqq ||p_i - x||^2 - r_i^2$ permettant de les caractériser comme une généralisation des diagrammes de Voronoï. Leurs bisecteurs Π_{ij}^{2} sont ainsi des hyperplans, comme illustré figure 2.18, et caractérisés par

$$\Pi_{ij}^{2} \coloneqq \left\{x \mid \delta^{(2)}(s_i,\, x) = \delta^{(2)}(s_j,\, x) \leq \delta^{(2)}(s_k,\, x),\; s_k \in \mathcal{S} \setminus \{s_i, s_j\}\right\}.$$

Si tous les poids des sites sont nuls, on retrouve alors le diagramme de Voronoï de leurs positions. La forme d'une cellule de Puissance $\mathcal{P}(s_i)$ est aussi caractérisée par un polytope convexe et a pour particularité qu'elle ne contient pas forcément son site associé s_i. De plus, si les sphères représentant les sites concernés par un bisecteur Π_{ij}^{2} s'intersectent, Π_{ij}^{2} est caractérisé par

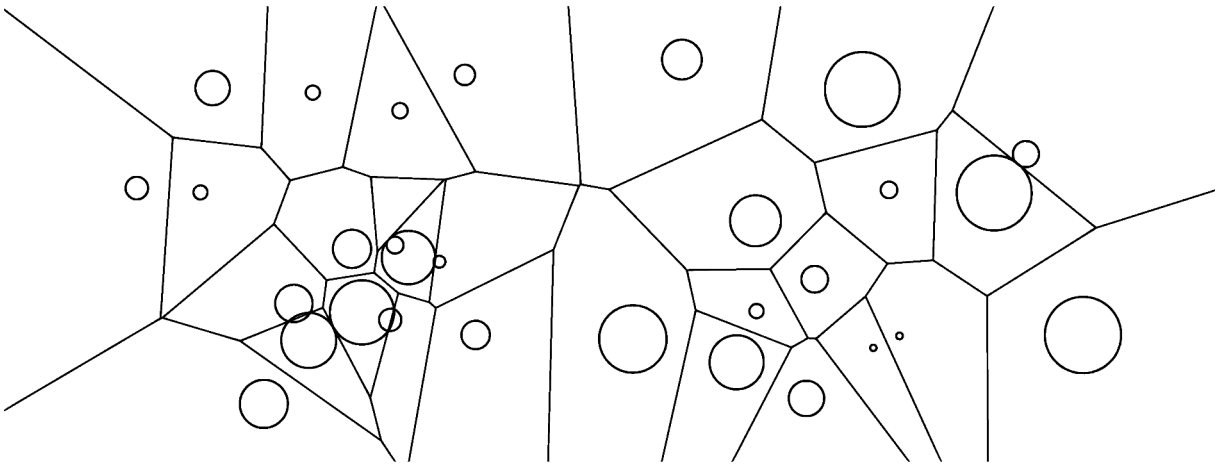


Figure 2.18 – Illustration du diagramme de Puissance d'un ensemble de cercles dans $\mathbb{R}^2$.

l'hyperplan passant par leur intersection. Leurs caractéristiques sont particulièrement utiles, et notamment pour trouver des solutions numériques du transport optimal semi-discret [BD23].

Les diagrammes d'Apollonius, ou diagrammes de Voronoï additivement pondérés, sont une autre généralisation des diagrammes de Voronoï à des sites pondérés. À la différence des diagrammes de Puissance, ils sont basés sur la distance euclidienne pondérée $\delta(s_i,\ x) \coloneqq ||p_i - x|| - r_i$ ou distance à la sphère. Un bisecteur d'Apollonius H_{ij} est alors caractérisé par

$$H_{ij} \coloneqq \{x \ \mid \ \delta(s_i,\ x) = \delta(s_j,\ x) \leq \delta(s_k,\ x),\ s_k \in \mathcal{S} \setminus \{s_i, s_j\}\}\,.$$

H_{ij} n'est cependant pas nécessairement un hyperplan, mais il peut être également caractérisé par des coniques [Aur87]. Une cellule d'Apollonius $\mathcal{A}(s_i)$ contient toujours le centre de son site p_i. En revanche, elle n'existe pas si le site est totalement contenu à l'intérieur d'un autre.

Les diagrammes d'Apollonius apparaissent naturellement dans l'étude d'ensemble de sphères. Comme illustré figure 2.19, les faces du diagramme caractérisent l'espace vide tangent aux sphères décrites par les sites. Dans $\mathbb{R}^3$, les faces sont caractérisées par des hyperboloïdes tandis que les

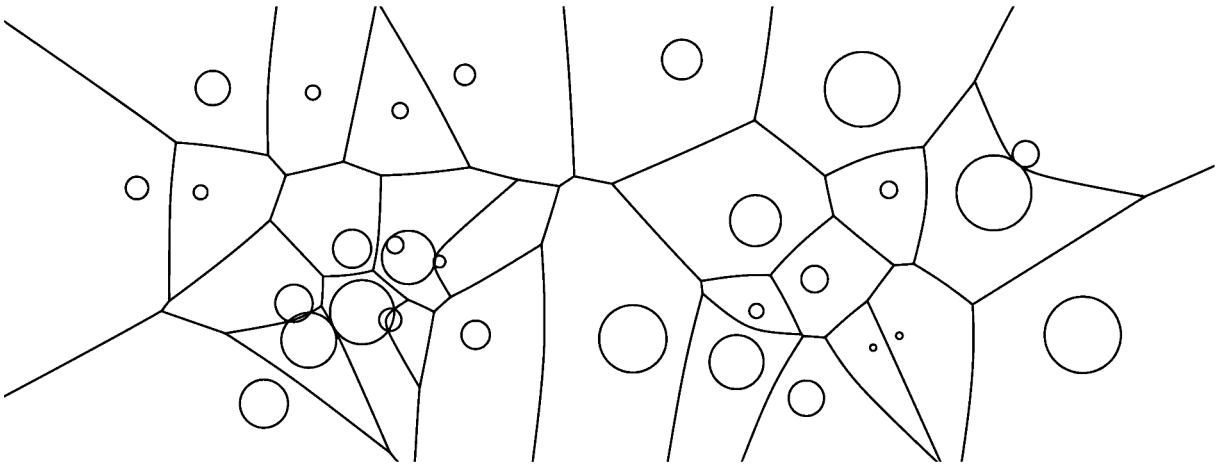


Figure 2.19 – Illustration du diagramme d'Apollonius d'un ensemble de sphères dans $\mathbb{R}^2$.

arêtes sont des coniques, hyperboles ou ellipses dans la majorité des cas, et parfois des paraboles ou lignes dans des cas particuliers [Wil99]. Enfin, contrairement aux diagrammes affines (*i.e.* avec des faces planaires), les arêtes ne sont pas nécessairement connectées.

La modélisation de protéines à partir de sphères, comme pour la surface de van der Waals, se prête ainsi à leur étude avec les diagrammes d'Apollonius. Ils sont ainsi utilisés pour leur analyse structurelle, comme la recherche de cavité, mais aussi pour le calcul de surface moléculaire.

Diagrammes d'Apollonius et surfaces moléculaires

Les surfaces moléculaires, définies à partir d'intersection de sphères (section 2.2), sont directement liées aux fonctions de distance au cœur des diagrammes de Voronoï. Ainsi différents travaux étudient les similarités entre la structure combinatoire des surfaces moléculaires et celle des diagrammes de Voronoï [Kim+19]. Des partitions de l'espace pour l'analyse structurelle de protéines ont de plus été définies à partir d'une combinaison de plusieurs diagrammes de Voronoï. Celle-ci est caractérisée à l'intérieur de l'union des sphères par des bisecteurs de puissances et des bisecteurs d'Apollonius à l'extérieur [Man+17 ; DQS20]. Les diagrammes de Voronoï sont aussi au cœur de nombreuses méthodes visant la recherche efficace de cavités dans des structures moléculaires [EFL98 ; Lin+13 ; Kro+16].

Dans les méthodes de construction de surfaces moléculaires basées sur la SAS [SOS96 ; TA96], le nombre de voisins devant être considérés augmente avec le rayon du probe. L'utilisation de diagramme de Voronoï permet ainsi de simplifier de façon importante ces traitements. Il est classique de représenter les sommets d'un diagramme de Voronoï à partir de sphères tangentes aux sites les plus proches [Del34]. Les faces du diagramme d'Apollonius peuvent alors être vues comme la famille des sphères tangentes aux sites. En comparaison avec la modélisation de la SAS et de la SES comme des surfaces décrites par le trajet du probe, le diagramme d'Apollonius donne alors toutes les positions et tous les rayons possibles de probe, comme illustré figure 2.20. Ce type de caractérisation permet ainsi le calcul de surfaces moléculaires avec des probes de grande

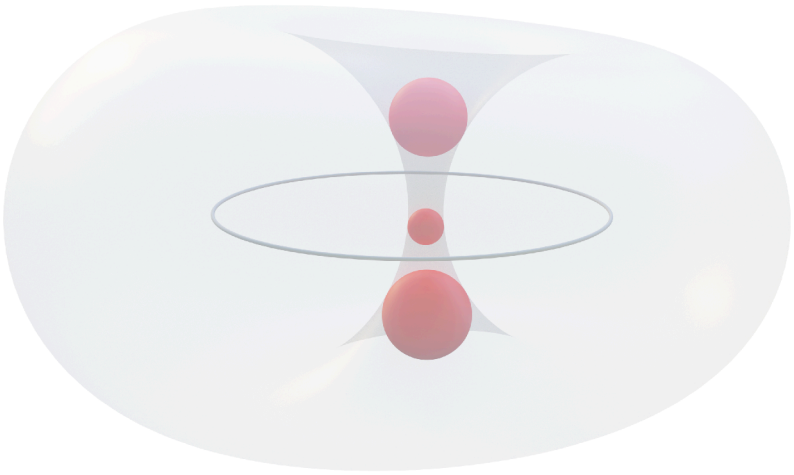


Figure 2.20 – Illustration de la famille de sphères centrées sur un trisecteur d'Apollonius, tangente aux trois sites et correspondant à une cyclide de Dupin. En tant que famille de sphères tangentes à un triplet d'atome, le trisecteur d'Apollonius contient toutes les positions possibles du probe.

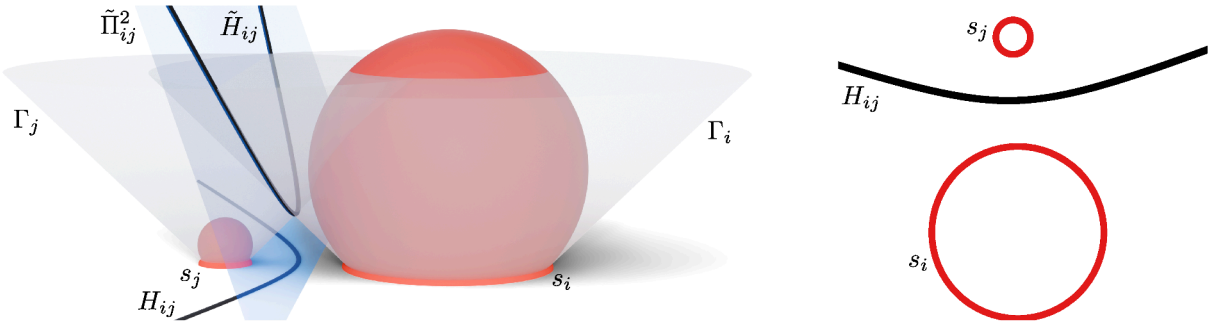
taille rapidement. Une fois le diagramme d'Apollonius construit, il suffit en effet d'échantillonner ses faces afin d'obtenir la surface [Joo+05 ; MJK16].

Les caractéristiques des diagrammes d'Apollonius les rendent particulièrement attractifs pour l'étude de structures moléculaires. Ces ensembles de sphères sont cependant uniquement contraints par des règles physico-chimiques pouvant donner lieu à des configurations variées et de grandes tailles nécessitant des algorithmes de construction à la fois robustes et performants.

2.3.2 Analyses et méthodes de calcul des diagrammes d'Apollonius

Les diagrammes d'Apollonius ont bénéficié de moins d'attention que les diagrammes de Voronoï ou les diagrammes de Puissance. Leur topologie courbe n'autorise pas le même genre d'optimisation que celles couramment utilisées sur les diagrammes affines, ce qui rend leur construction particulièrement difficile. Les besoins des domaines comme la biochimie ont cependant mené à la présentation d'algorithmes variés dans l'état de l'art [Aur87 ; Luc+99 ; Wil99 ; KY02 ; KCK05 ; Kam20]. Dans $\mathbb{R}^2$, des méthodes éprouvées et robustes [KY02] sont disponibles publiquement dans des bibliothèques logicielles facilement accessibles [The23]. Dans $\mathbb{R}^3$, même si différentes méthodes ont été proposées, il n'y a pas encore, à notre connaissance, d'équivalent proposant une construction robuste et accessible. Dans cette section, nous présentons les principaux travaux précédents visant au calcul des diagrammes d'Apollonius dans $\mathbb{R}^3$.

Plusieurs études fondatrices décrivent les bases théoriques sur lesquelles s'appuient les principales méthodes de construction de l'état de l'art. Aurenhammer [Aur87], dans son étude des diagrammes de Puissance, établit un lien entre les diagrammes d'Apollonius dans $\mathbb{R}^d$ et les diagrammes de Puissance dans $\mathbb{R}^{d+1}$, illustré figure 2.21. Soit un site pondéré $s_i \in \mathbb{R}^d \times \mathbb{R}$, Γ_i, le cône d'axe x_{d+1} et positionné tel que $s_i = \Gamma_i \cap \mathbb{R}^d \times \{0\}$, et le site pondéré $\tilde{s}_i \in \mathbb{R}^{d+1} \times \mathbb{R}$ tel que $s_i = \Gamma_i \cap \tilde{s}_i$. Le bisecteur d'Apollonius H_{ij} est alors caractérisé par la projection sur $\mathbb{R}^d \times \{0\}$ de l'intersection $\Pi^2_{ij} \cap \Gamma_i$ entre le bisecteur de puissance Π^2_{ij} des deux sites $\tilde{s}_i$ et $\tilde{s}_j$, et le cône Γ_i. La cellule d'Apollonius $\mathcal{A}(s_i)$ peut donc être obtenue par la projection de l'intersection de la cellule de puissance $\mathcal{P}(\tilde{s}_i)$ avec le cône Γ_i sur $\mathbb{R}^d \times \{0\}$. Cette caractérisation du diagramme d'Apollonius a été par la suite approfondie par Boissonnat et al. [BWY06]. Elle permet ainsi une construction élégante des diagrammes d'Apollonius, mais qui requiert cependant le calcul d'un diagramme de Puissance dans $\mathbb{R}^{d+1}$. Ce dernier aspect implique un coût non



(a) Formulation d'Apollonius comme une projection depuis $\mathbb{R}^3$ sur $\mathbb{R}^2$.

(b) Diagramme d'Apollonius de deux sites dans $\mathbb{R}^2$.

Figure 2.21 – Illustration de la projection verticale de l'intersection entre une cellule de Puissance dans $\mathbb{R}^3$ et un cône d'axe vertical (a) donnant le diagramme d'Apollonius en $\mathbb{R}^2$ (b).

négligeable en raison de la dépendance exponentielle du nombre de voisins d'une cellule par rapport à la dimensionnalité du diagramme.

Will introduit, dans sa thèse [Wil99], une caractérisation et une paramétrisation des arêtes du diagramme d'Apollonius. Il présente ainsi une construction d'une cellule d'Apollonius à partir de la projection de ses faces sur la sphère. Gavrilova et Rokne, dans leurs études sur la maintenance d'un diagramme d'Apollonius dans un contexte dynamique, étudient les configurations géométriques du diagramme à partir de systèmes d'équations non linéaires [GR03]. Enfin, Kamarianakis propose dans sa thèse une étude complète des prédicats nécessaires à la construction des diagrammes d'Apollonius dans $\mathbb{R}^3$ [Kam20] sur la base des travaux effectués dans $\mathbb{R}^2$ [KY02].

Bien que les études fondatrices présentées jusqu'ici ont parfois été accompagnées d'algorithmes de construction dédiés, elles ne visent pas de hautes performances de calcul et requièrent des processus complexes. Dans ce contexte, Luchnikov et al. proposent de calculer un premier sommet du diagramme et chercher de nouveaux sommets à partir de ses trisecteurs correspondants [Luc+99]. Cette stratégie, appelée Edge-Tracing, a été reprise et améliorée par de nombreux travaux [KCK04 ; KCK05 ; Med+06 ; OV14]. Elle est caractérisée par un concept simple qui a participé à sa popularité, illustré figure 2.22. Afin de trouver un premier sommet valide de façon performante, des sites positionnés à l'extérieur de l'ensemble d'entrée peuvent être ajoutés artificiellement de façon à garantir qu'ils produisent un sommet valide [KCK05]. Une fois ce sommet trouvé, ses quatre arêtes correspondantes peuvent être ajoutées à une pile afin de trouver leurs sommets finaux. Contrairement aux diagrammes affines, les diagrammes d'Apollonius peuvent être composés de plusieurs segments d'arêtes. Comme illustré figure 2.22, l'arête e est composée d'un premier segment entre v_1 et v_2 et un second entre v_3 et v_4. D.-S. Kim et al. proposent de tester si le sommet trouvé sur l'arête actuelle est valide et n'intersecte donc pas de sites de l'ensemble d'entrée [KCK05]. Si plusieurs sommets sont valides, ils sont ordonnés de façon angulaire sur l'arête et seul le sommet le plus proche du départ est conservé. Pour ne pas créer un sommet plusieurs fois, il est courant d'utiliser un dictionnaire basé sur une fonction de hachage. Même si Edge-Tracing permet de bonnes performances d'exécution, cet algorithme ne prend cependant pas en compte les diagrammes contenant des arêtes déconnectées. Manak et Kolingerova proposent une extension permettant de découvrir toutes les composantes, mais requérant cependant plus de traitement [MK16].

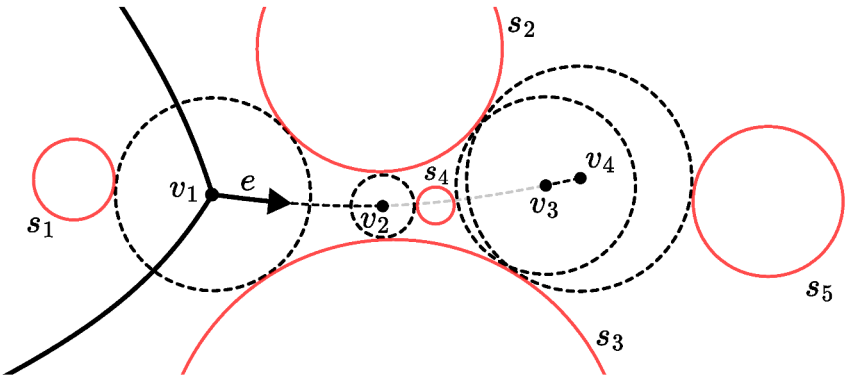


Figure 2.22 – Illustration du processus de l'algorithme Edge-Tracing. Le traitement débute avec le sommet v_1 et cherche à trouver le prochain sommet de l'arête e entre s_2 et s_3. s_4 produit deux sommets valides v_2 et v_3 avec e. Les segments finaux sont entre v_1 et v_2 ainsi qu'entre v_3 et v_4.

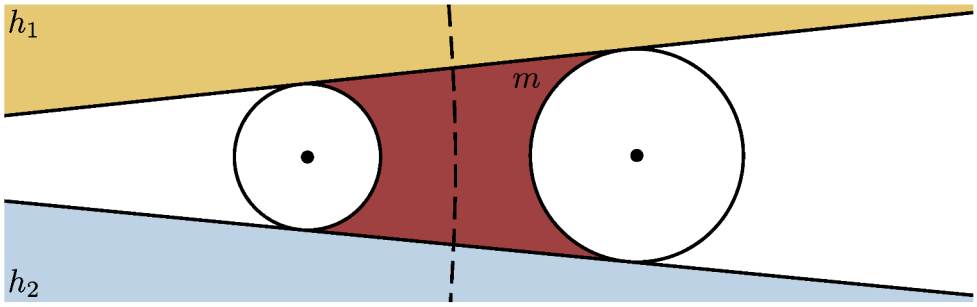


Figure 2.23 – Illustration des zones de recherches de sommets dans Voronota [OV14]. Un trisecteur sous la forme d'une hyperbole (en pointillé) peut être divisé en trois zones : deux demi-espaces h_1 et h_2, ainsi qu'une zone centrale m. h_1 et h_2 ne peuvent contenir qu'un seul sommet valide, ce qui facilite la recherche. m peut contenir plusieurs sommets, mais sa zone de recherche est plus petite.

Olechnovič et Venclovas proposent une implémentation d'Edge-Tracing optimisée et parallélisée sur CPU, appelée Voronota [OV14]. Afin d'améliorer les performances de recherche de sommet, ils divisent l'espace de recherche en trois zones, comme illustré figure 2.23. L'enveloppe convexe des sites générateurs d'une arête sous la forme d'une hyperbole est bornée par deux plans tangents aux sites, définissant chacun deux demi-espaces. Nous pouvons alors observer que, si un sommet valide est positionné dans un de ces demi-espaces, aucun autre sommet n'existe dans ce demi-espace. La zone interne à ces plans m peut cependant inclure un nombre indéfini de sommets. Ainsi, la méthode consiste à rechercher le premier sommet valide dans chacun des demi-espaces, puis à effectuer une recherche localisée sur m. Cette stratégie facilite la résolution d'une arête sous la forme d'une hyperbole, mais n'est pas adaptée à celles décrites par d'autres formes, telles que les ellipses. Ces dernières étant peu présentes sur les ensembles de sites provenant de structures moléculaires, il est possible de les traiter à partir d'une recherche linéaire sur l'ensemble des sites [OV14].

Les limitations d'Edge-Tracing le rendent peu adapté pour traiter des ensembles présentant beaucoup de trisecteurs elliptiques ou des cas déconnectés. Ainsi, d'autres stratégies ont été proposées afin de construire ces diagrammes de façon exhaustive et performante. P. Wang et al. proposent une méthode basée sur une subdivision récursive de l'espace permettant la localisation des composantes du diagramme en fonction des sites les plus proches [Wan+20]. Cette approche est particulièrement intéressante puisqu'elle permet la construction des arêtes non connectées du diagramme. Elle est cependant contrainte par ses performances puisqu'elle requiert plusieurs dizaines de secondes pour le traitement de quelques milliers de sites avec une configuration matérielle standard. Plus récemment, Lee et al. ont introduit une stratégie robuste de calcul des diagrammes d'Apollonius [LSK22]. Basée sur la topologie du diagramme ainsi qu'un ensemble de règles permettant de détecter et maintenir la structure, cette méthode supporte des cas complexes déconnectés qui ne peuvent pas être calculés avec les implémentations d'Edge Tracing [OV14]. Les auteurs proposent de plus une analyse des performances de leur méthode sur des ensembles variés de sphères réparties aléatoirement, mais aussi de protéines [Son+22]. Ces deux cas présentent des configurations différentes de variations de rayons et de répartitions des sites. Ainsi, les protéines proposant des sites répartis de façon hétérogène, mais avec des rayons similaires, leurs bisecteurs sont peu courbés. Cela rend les configurations complexes à traiter avec Edge-Tracing, telles que les trisecteurs elliptiques et les configurations déconnectées, plus rares. Au contraire, les répartitions plus uniformes, avec des variations de rayon importantes, requièrent une prise en compte efficace

de ces configurations. La méthode de Lee et al. permet ainsi de meilleurs temps de calcul sur des ensembles uniformes que Voronota, qui est cependant 10 fois plus rapide sur des protéines.

Même si différents travaux ont été proposés afin de construire les diagrammes d'Apollonius, ils restent difficiles à calculer de façon performante et exhaustive. Notamment, les méthodes permettant une construction complète du diagramme sont basées sur des processus complexes qui sont peu adaptés à de grands ensembles. À l'inverse, Edge-Tracing permet de meilleures performances dans ce dernier cas [OV14], mais est restreint au calcul des arêtes connectées du diagramme. La construction complète et rapide de diagrammes d'Apollonius de grands ensembles reste complexe. Cette dernière permettrait pourtant l'analyse de structures biochimiques efficacement, mais pourrait aussi soutenir d'autres domaines variés [MLW07 ; SS07 ; WZ10 ; GKP12 ; MGM12]. Ces fonctionnalités étant souhaitées dans de nombreux contextes non limités aux diagrammes d'Apollonius, différents travaux ont proposé des méthodes de construction de diagramme de Voronoï visant une exécution hautement parallèle.

2.3.3 Construction parallèle des diagrammes de Voronoï et généralisations

De nombreuses applications nécessitent le calcul rapide de diagramme de Voronoï de grande taille. Ainsi, différentes méthodes ont été présentées afin de proposer une construction parallèle de diagrammes de très grands ensembles de sites. Dans cette étude, nous nous intéressons aux méthodes visant une exécution parallèle sur des GPUs.

La construction indépendante de chacune des cellules d'un diagramme est particulièrement adaptée à un calcul parallèle et constitue la base de plusieurs méthodes visant une exécution sur GPU. Ce type de construction, basé sur le voisinage des sites, a été introduit par Rycroft [Ryc09] pour des diagrammes de Voronoï d'ensembles de points. Les cellules sont initialisées avec la boite englobante de l'ensemble d'entrée et sont ensuite itérativement raffinées à partir des bisecteurs des nouveaux sites insérés dans l'ordre croissant de leur distance par rapport au site traité. Le rayon de sécurité est couramment utilisé afin de borner la recherche et détecter la validation de la cellule [LB13], comme illustré figure 2.24. Celui-ci est basé sur la construction de la cellule de Voronoï comme un ensemble de demi-espaces. Dans ce contexte, un site voisin est contribuant si et seulement si la cellule n'est pas totalement comprise dans le demi-espace Π_{ij}^{+}. Autrement dit, si le bisecteur Π_{ij} intersecte le polytope de la cellule. En construisant

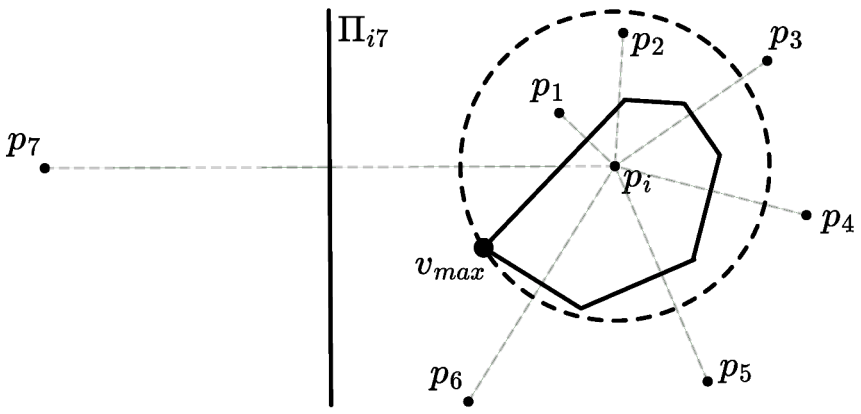


Figure 2.24 – Illustration du rayon de sécurité. p_7 ne respecte pas le rayon de sécurité, le bisecteur Π_{i7} ne contribue donc pas à la cellule de $\mathcal{V}(p_i)$.

la cellule de p_i on définit alors v_{max}, le sommet le plus éloigné de p_i. Un nouveau voisin p_j contribue à la cellule de p_i si et seulement

$$||p_i - p_j|| < 2||p_i - v_{max}||.$$

Ainsi, la cellule est complète si le prochain voisin dans l'ordre de la distance ne respecte pas ce critère. Cette stratégie élégante permet la construction performante de diagrammes de Voronoï lorsque les sites sont distribués uniformément dans l'espace, car l'ensemble des sites contribuants est équivalant à l'ensemble des plus proches voisins.

Différentes méthodes de construction sur GPU utilisant le rayon de sécurité ont ainsi été proposées. Ray et al. proposent de calculer la géométrie de la cellule pour des applications ne nécessitant pas une robustesse combinatoire importante telles que les simulations basées sur des méthodes de transport optimal [Ray+19]. Afin de ne pas stocker toute la géométrie explicitement et respecter les contraintes de mémoire des GPUs, ils proposent une structure de données basées sur les sommets de la cellule, à partir desquelles les arêtes et les faces peuvent être retrouvées. Basselin et al. ont ensuite présenté une adaptation pour les diagrammes de Puissance basée sur des méthodes de construction et d'intégration du volume de la cellule plus performantes [Bas+21]. Ces travaux visent cependant des ensembles de sites répartis de façon homogène dans l'espace. Dans le cas de distributions hétérogènes, les cellules peuvent être très déformées, ce qui impacte la pertinence du rayon de sécurité. Ainsi, il est nécessaire de traiter plus de voisins que nécessaire puisque l'ensemble des voisins les plus proches n'est plus équivalent à l'ensemble des voisins contribuant à la cellule, ce qui impacte les performances de calcul.

2.4 Conclusion

Les surfaces moléculaires comme la SAS et la SES sont au cœur de nombreuses études et ainsi présentes dans de nombreux logiciels de visualisation moléculaire. Ces derniers ont cependant des contraintes importantes liées à leur distribution pouvant contraindre leur développement. Notamment, la diversité des configurations matérielles des utilisateurs requiert l'utilisation de méthodes compatibles avec des environnements variés tout en proposant les fonctionnalités souhaitées.

Les surfaces discrétisées sont majoritaires dans les logiciels de visualisation scientifique. La disponibilité de méthodes de calcul comme MSMS [SOS96] ainsi que les facilités d'implémentations des surfaces approximées permettent une intégration simple de ce type de géométrie. Elles requièrent cependant un compromis important entre capacité mémoire et précision de la surface finale. Elles sont ainsi incompatibles avec l'illustration de grandes structures qui pourrait aider la communication de nombreuses études récentes.

Le calcul analytique des surfaces moléculaires propose de nombreux avantages en comparaison à leur calcul approximé. Ces stratégies permettent notamment de dissocier les liens entre consommation mémoire et qualité de la surface. Elles sont cependant limitées par leur performance sur de grandes structures ainsi que pour le calcul de surfaces complètes. En effet, même si elles ne souffrent pas des limites concernant la précision des surfaces discrétisées, elles sont soumises à une consommation mémoire et des temps de calcul importants limitant leur distribution et leur applicabilité à des protéines de grandes tailles.

Les diagrammes de Voronoï contiennent des informations particulièrement utiles à l'analyse de la structure moléculaire et sont à la base de nombreux travaux dans des domaines variés. La description de l'espace vide à l'intérieur de la structure permet une caractérisation alternative

des molécules dont les propriétés de leur conformation sont très étudiées. Ils restent cependant complexes à calculer à de grandes échelles, de façon parallèle et exhaustive.

Dans ce contexte, les travaux présentés dans cette thèse visent à répondre aux besoins en construction de géométrie moléculaire de façon performante et parallèle tout en ne restreignant pas les potentielles applications.

Chapitre 3

Construction parallèle efficace de la SES de grandes protéines

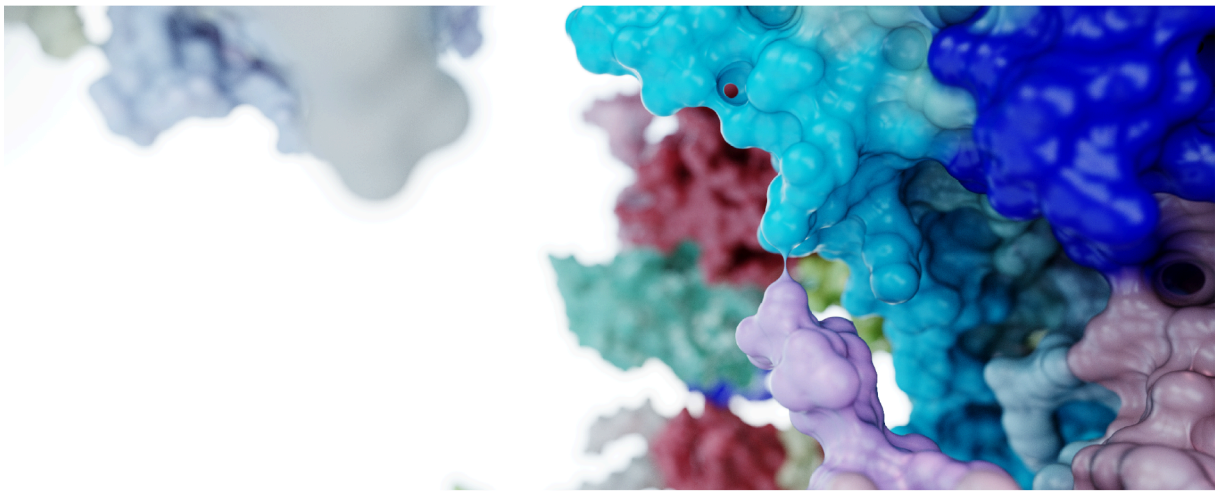


Figure 3.1 – Illustration de la SES d'une grande protéine (PDB : 5GMK), calculée avec notre méthode et affichée avec notre système de rendu illustratif.

Sommaire

Les caractéristiques de la SES permettent de répondre à de nombreux besoins liés à la visualisation moléculaire. Elle permet, entre autres, le support de calcul scientifique [QS16] mais aussi la visualisation de cavités de façon préliminaire à des simulations [Jur+16] ce qui la rend centrale à l'analyse de protéines. Ainsi, ses propriétés sont particulièrement intéressantes pour la visualisation de grands complexes moléculaires, devenus courants par l'amélioration des stratégies d'acquisition, ainsi que de séquences d'animation résultant de simulations moléculaires, mais qui nécessitent cependant une construction rapide et consommant peu de mémoire.

Elle reste cependant difficile à construire, notamment de façon complète, précise, et sur de grandes structures. Ainsi, la méthode la plus rapide est limitée à un calcul extérieur de la surface ne permettant pas la visualisation interne de la surface ainsi que les calculs de quantités, comme son aire et volume. De plus, cette dernière est contrainte par sa consommation mémoire importante qui la rend incompatible avec de très grandes structures. Enfin, même si les protéines peuvent être composées de schémas structurels répétitifs qu'il est possible de factoriser afin de limiter les calculs et l'utilisation de mémoire [Rau+19], les séquences d'animation de simulations empêchent tout à priori sur la conformation des structures cibles.

Dans ce contexte, nous présentons une méthode de construction parallèle de la SES analytique et complète. Notre stratégie est soutenue par les contributions suivantes :

— nous proposons une structure de données légère visant un calcul efficace sur GPU de la SES complète ;
— afin de supporter de grands complexes moléculaires, nous introduisons différentes opérations permettant une réduction importante de la consommation mémoire en comparaison avec les méthodes précédentes ;

Symbole	Signification
$\mathcal{M}$	Une molécule.
a_i, c_i, r_i, $\mathcal{N}(a_i)$	Le $i^{\text{ème}}$ atome, son centre, rayon de van der Waals et son voisinage.
r_p	Le rayon du probe.
$\mathscr{C}_{ij}$, c_{ij}, r_{ij}, n_{ij}	Un cercle de la SAS entre a_i et a_j, son centre, rayon et normale.
$\mathcal{I}$	L'ensemble des cercles de la SAS intersectés.
x, $\mathcal{X}$	Une intersection de la SAS et l'ensemble des intersections de la SAS.
$\mathcal{X}(\mathscr{C})$	L'ensemble des intersections de la SAS d'un cercle donné $\mathscr{C}$.
P_+	Un patch convexe.
P_-, T	Un patch concave et son tétraèdre.
P_t, P_{tf}	Un patch toroïdal sous la forme d'un arc et complet.

Table 3.1 – Nomenclature utilisée dans ce chapitre.

— nous introduisons plusieurs traitements coopératifs et parallèles pour le calcul efficace.

Notre méthode permet ainsi le calcul efficace de la SES complète de très grandes structures de façon compétitive aux méthodes précédentes tout en ne souffrant pas des mêmes limites.

Ce chapitre commence par la présentation des éléments mathématiques visant le calcul des primitives de la SAS ainsi que la définition géométrique de la SES (section 3.1) qui sont suivis par notre processus de construction (section 3.2). Nous donnons ensuite les détails d'implémentation GPU (section 3.3). Enfin, nous présentons les résultats et analysons les performances de notre méthode (section 3.4) ainsi que ses limites (section 3.5). Enfin, nous concluons ce chapitre (section 3.6). La nomenclature utilisée dans ce chapitre est détaillée tableau 3.1.

3.1 Préambule

Nos travaux se basent sur les caractérisations de la SAS et de la SES décrivant leurs fondations structurelles. Cependant, à la différence des méthodes précédentes, nous utilisons l'étude de Quan et Stamm [QS16] qui permet notamment une gestion implicite des singularités de la SES. Ainsi, dans cette section, nous rappelons les éléments mathématiques nécessaires à la construction des primitives de la SAS ainsi que la définition formelle des patchs de la SES.

3.1.1 Primitives de la SAS

Les primitives de la SAS étant au cœur de la caractérisation de la SES, comme présenté section 2.2.1, elles représentent la cible des algorithmes de construction analytique de ses patchs. Avant de présenter les étapes de notre méthode, nous définissons formellement les primitives de la SAS sur la base des différentes caractérisations [TA96 ; QS16 ; QS17].

Les cercles de la SAS sont à la base de la définition des patchs de la SES. Ainsi, le centre c_{ij} d'un cercle $\mathscr{C}_{ij}$ entre les atomes a_i et a_j est défini par

$$c_{ij} = c_i + (c_j - c_i)\frac{(r_i + r_p)^2 - (r_j + r_p)^2 + ||c_i - c_j||^2}{2\,||c_i - c_j||^2},$$

avec c_i et c_j les centres de a_i et a_j, r_i et r_j leurs rayons et r_p le rayon du probe [TA96]. Son rayon r_{ij} est alors obtenu avec

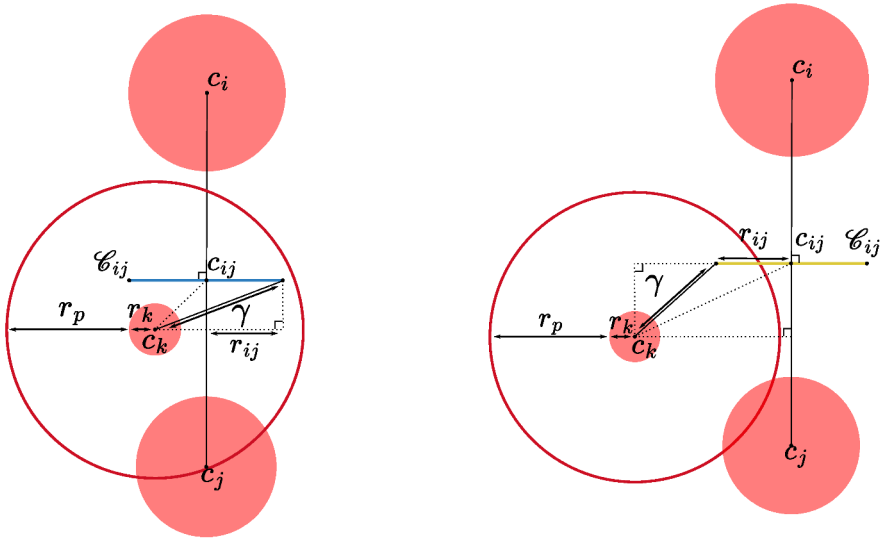
$$r_{ij} = \sqrt{(r_i + r_p)^2 - ||c_i - c_{ij}||^2}.$$

Une fois un cercle connu, il est intéressant de tester s'il est modifié par une sphère voisine. Ainsi, un cercle $\mathscr{C}_{ij}$ est totalement localisé à l'intérieur de la sphère de la SAS de l'atome a_k si et seulement si

$$(\sin(\theta)||c_{ij} - c_k|| + r_{ij})^2 + (\cos(\theta)||c_{ij} - c_k||)^2 < (r_k + r_p)^2$$

avec θ l'angle entre $c_{ij} - c_k$ et le plan sur lequel repose $\mathscr{C}_{ij}$ [QS17]. Le cercle est alors occulté et ne contribue pas à la SAS. De la même façon, $\mathscr{C}_{ij}$ est intersecté par la sphère voisine de l'atome a_k si et seulement si

$$(-\sin(\theta)||c_{ij} - c_k|| + r_{ij})^2 + (\cos(\theta)||c_{ij} - c_k||)^2 < (r_k + r_p)^2.$$



(a) $\mathscr{C}_{ij}$ (en bleu) est occulté par la sphère de la SAS de a_k

(b) $\mathscr{C}_{ij}$ (en jaune) est intersecté par la sphère de la SAS de a_k

Figure 3.2 – Illustration vue du dessus des tests d'occlusions (a) et d'intersections (b) du cercle de la SAS $\mathscr{C}_{ij}$ avec la sphère de la SAS voisine liée à l'atome a_k. Si le cercle n'est ni intersecté ni occulté, il est visible et complet.

Comme illustré figure 3.2, ces deux tests sont basés sur la distance entre le cercle et c_k : si le point le plus éloigné de c_k sur le cercle est dans la sphère de la SAS, le cercle est totalement à l'intérieur de la sphère. De façon similaire, si le point le plus proche de c_k sur $\mathscr{C}_{ij}$ est dans la sphère de la SAS, alors elle l'intersecte.

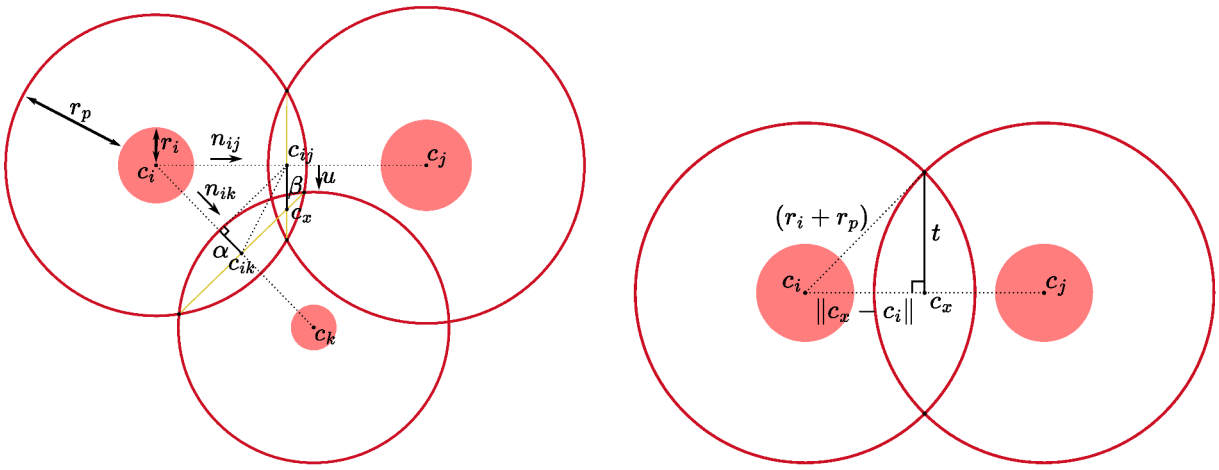
Il est nécessaire de calculer explicitement les intersections de la SAS pour tester leur visibilité ou afficher les patchs associés. Comme illustré figure 3.3a, les intersections entre les atomes a_i, a_j et a_k peuvent être obtenues en calculant tout d'abord leur projection c_x sur le plan défini par les centres des atomes c_i, c_j et c_k

$$c_x = c_{ij} + u\gamma,\ \gamma = \frac{\alpha}{u \cdot n_{ik}}$$
$$\alpha = (c_{ik} - c_{ij}) \cdot n_{ik},\ u = n_{ik} - (n_{ik} \cdot n_{ij})\, n_{ij}.$$

On projette ainsi α sur u afin d'obtenir la distance entre c_{ij} et c_x dans la direction de u. Les positions des deux intersections x_0 et x_1 peuvent finalement être calculées à partir de la distance entre c_x et le centre des atomes ainsi que du rayon des sphères de la SAS (figure 3.3b)

$$x_{0,1} = c_x \pm (n_{ik} \times n_{ij})\, t \tag{3.1}$$
$$t = \sqrt{(r_i + r_p)^2 - ||c_x - c_i||^2}.$$

Ces opérations compactes permettent le calcul de composantes importantes de la SAS, nécessaires à la construction de la SES. Ainsi, elles sont aux centres de la méthode que nous présentons dans ce chapitre.



(a) Centre c_x des intersections sur le plan $c_i c_j c_k$ **(b)** Hauteur par rapport au plan

Figure 3.3 – Illustration du calcul des intersections de la SAS à partir de leur projection c_x sur le plan formé par c_i, c_j et c_k (a) et de leur distance par rapport à celui-ci (b).

3.1.2 Définition géométrique de la SES

Nous présentons la définition géométrique des patchs de la SES. Pour cela, nous nous référons à l'étude proposée par Quan et Stamm [QS16] qui permet une caractérisation complète de la surface à partir de fonctions distance signées. Celles-ci, de la forme $d : \mathbb{R}^3 \to \mathbb{R}$, décrivent la distance entre un point quelconque $x \in \mathbb{R}^3$ et la SES. Ainsi, elles se prêtent particulièrement bien à l'affichage en utilisant des stratégies de lancer de rayon dédiées comme le *Sphere Tracing* [Har96], et donc à la visualisation de cette surface précise au pixel près. Ces fonctions proposent la distance entre un point et les patchs de la SES (figure 3.4) à partir des primitives de la SAS. Cela est notamment effectué en remarquant que lorsqu'un point x est localisé dans la SAS, sa distance à la SES est donnée par sa distance à la SAS à laquelle on ajoute le rayon du probe.

Patchs convexes P_+ Lorsque le probe roule sur la surface d'un seul atome sans en intersecter d'autres, il possède deux degrés de liberté. Son centre dessine alors la surface d'une sphère de la SAS. De la même façon, la surface du probe en contact avec la sphère de van der Waals produit un patch convexe P_+ de la SES, illustré figure 3.5. Soit un point $x \in \mathbb{R}^3$ positionné à l'extérieur de la SAS. Le point de la SES le plus proche de x est alors positionné sur un patch

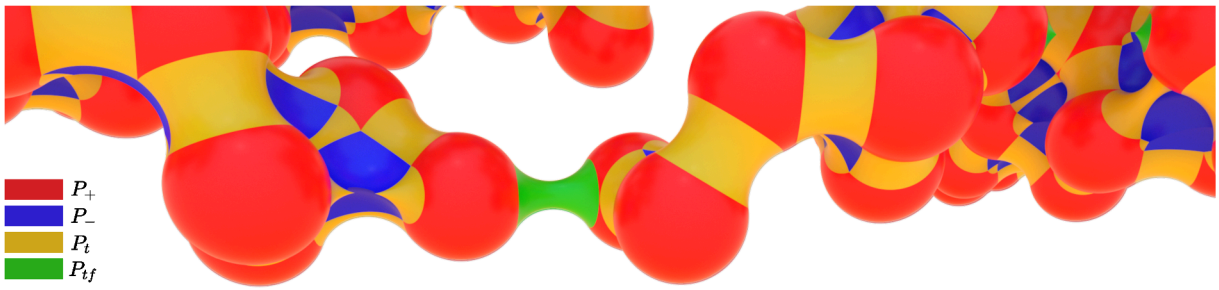


Figure 3.4 – Illustration des patchs correspondant à la SES d'une protéine (PDB : 3DIK). De quatre types, ils caractérisent complètement la SES une fois assemblés.

convexe P_+. La distance $d(P_+,\ x)$ entre x et la SES est alors

$$d(P_+,\ x) \coloneqq ||x - c_i|| - (r_i + r_p) + r_p = ||x - c_i|| - r_i.$$

avec c_i et r_i le centre et le rayon de l'atome le plus proche $a_i = \arg\min_{a_i} ||x - c_i|| - r_i$. De la même façon, si x est localisé dans la SAS et que son point le plus proche sur celle-ci appartient à une seule sphère, sa distance à la SES est donnée par

$$d(P_+,\ x) \coloneqq -||x - c_i|| + (r_i + r_p) - r_p = -||x - c_i|| + r_i,$$

La surface du patch convexe P_+ est donc délimitée par l'union de la projection des cercles de la SAS $\mathscr{C}_{ij}$ sur la sphère de van der Waals correspondante.

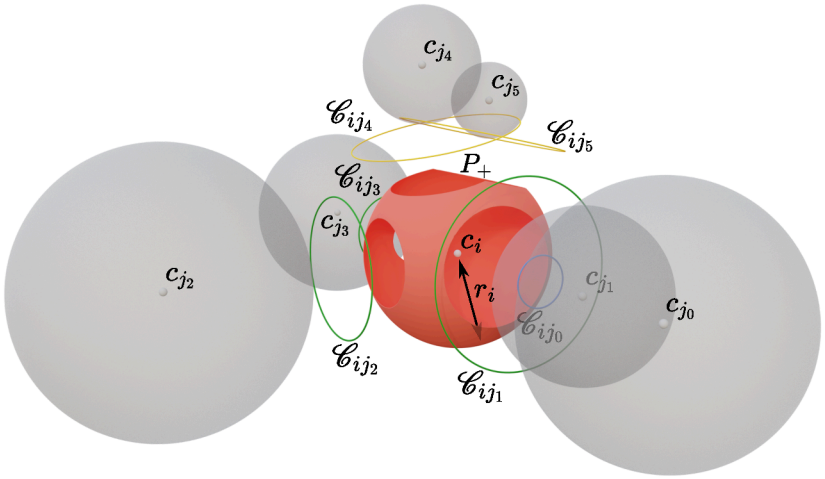


Figure 3.5 – Illustration d'un patch convexe (en rouge) et de ses cercles complets (en vert) et intersectés (en jaune) correspondants. Il est défini par la sphère de van der Waals associée à son atome ainsi que des cercles de la SAS. La projection de ces derniers sur la surface de la sphère définie les bordures du patch.

Patchs toroïdaux P_{tf}, P_t Lorsque le probe roule exactement sur deux sphères de van der Waals, il n'a qu'un seul degré de liberté et son centre dessine un cercle de la SAS. Soit un point $x \in \mathbb{R}^3$ localisé sur le triangle $\triangle(x_{ij}c_ic_j)$, défini par x_{ij}, la projection de x sur le cercle de la SAS $\mathscr{C}_{ij}$, et c_i et c_j, les centres des deux atomes a_i et a_j à l'origine de $\mathscr{C}_{ij}$. Le point de la SAS le plus proche de x est alors localisé sur $\mathscr{C}_{ij}$ et son point de la SES le plus proche appartient à un patch toroïdal P_t. La distance entre x et la SES est alors donnée par $d(P_t,\ x)$

$$d(P_t,\ x) \coloneqq -||x - x_{ij}|| + r_p.$$

x_{ij} peut être obtenu en projetant x sur le cercle $\mathscr{C}_{ij}$ à partir de

$$x_{ij} = c_{ij} + r_{ij}\frac{v}{||v||}$$
$$v = x + ((c_{ij} - x) \cdot n_{ij})n_{ij} - c_{ij},$$

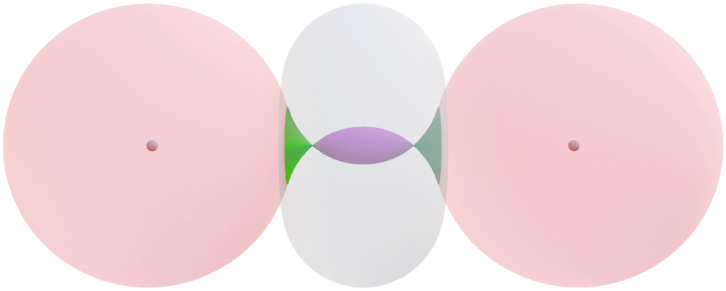


Figure 3.6 – Illustration d'une singularité géométrique sur un patch toroïdal. Celle-ci se produit lorsque le cercle principal du tore à un rayon inférieur à celui du cercle interne. Les caractérisations historiques de la SES [Con83] nécessitent de prendre en compte ce cas afin de supprimer la partie réentrante du tore (en violet) ce qui n'est pas le cas avec la caractérisation de Quan et Stamm.

avec c_{ij}, n_{ij} et r_{ij} le centre, la normale, et le rayon de $\mathscr{C}_{ij}$. Par convention, la normale n_{ij} est donnée par $c_i\vec{c}_j$. Le cercle $\mathscr{C}_{ij}$ peut être intersecté et donc constitué de plusieurs arcs, chacun délimité par les deux intersections x^l et x^m, comme illustré figure 3.7. Dans ce cas, plusieurs patchs P_t peuvent résulter d'un même cercle. Le point de la SAS le plus proche de x est sur un arc de cercle donné s'il respecte les conditions précédentes et si sa projection x_{ij} est localisée entre les bornes de l'arc x^l et x^m. Ce test peut être réalisé à partir de l'angle entre x_{ij} et les deux intersections sur $\mathscr{C}_{ij}$

$$\measuredangle\left(x^l - c_{ij}, x_{ij} - c_{ij}\right) < \measuredangle\left(x^l - c_{ij}, x^m - c_{ij}\right),$$

avec $\measuredangle(\vec{v_1}, \vec{v_2})$, l'angle entre les vecteurs $\vec{v_1}$ et $\vec{v_2}$. Nous différencions les notations des patchs issus d'un cercle complet P_{tf} et ceux issus d'un arc de cercle P_t. Les précédentes caractérisations de la SES définissent les patchs toroïdaux comme la partie centrale d'un tore [Con83], comme illustré figure 3.6. Lorsque le rayon du probe est supérieur à celui du cercle, la géométrie du tore s'intersecte elle-même et il est nécessaire de supprimer la zone intérieure (en violet figure 3.6). La caractérisation de Quan et Stamm supporte implicitement cette propriété et ne nécessite donc pas ce traitement additionnel [QS16].

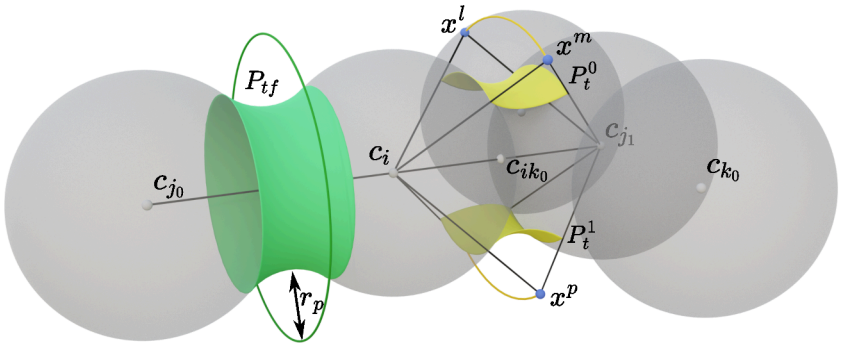
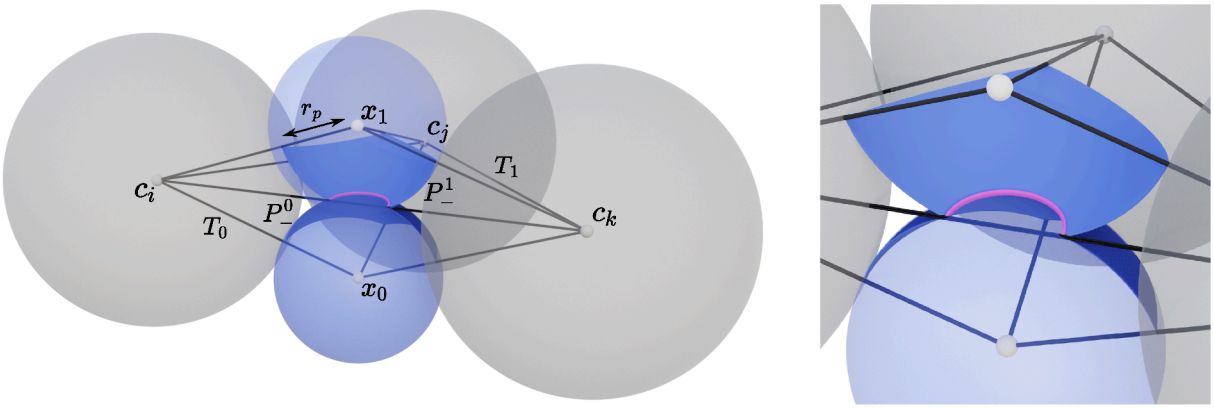


Figure 3.7 – Deux patchs toroïdaux sous la forme d'arcs P_t^0 et P_t^1 (en jaune) et un patch toroïdal complet P_{tf} (en vert).

Patchs concaves P_- Lorsque le probe est tangent à trois sphères de van der Waals, il est immobile et son centre est positionné sur une intersection de la SAS, provenant de l'intersection de trois cercles de la SAS. La région non couverte par les patchs précédents est caractérisée par l'union de tous les tétraèdres $\mathcal{T} = \bigcup T^t$ définis par les intersections de la SAS x^t_{ijk} et les centres des atomes associés c_i, c_j, c_k. Soit un point $x \in \mathbb{R}^3$ localisé dans $\mathcal{T}$. Le point de la SAS le plus proche de x est alors positionné sur une intersection tandis que son point de la SES le plus proche est localisé sur un patch concave P_- (figure 3.8). Ainsi, sa distance à la SES $d(P_-,\ x)$ est donnée par

$$d(P_-,\ x) \coloneqq -||x - x^t_{ijk}|| + r_p,$$

avec $x^t_{ijk} = \arg\min_{x^t_{ijk}} ||x - x^t_{ijk}||$. Cette représentation caractérise implicitement les singularités produites par l'intersection entre deux positions de la sphère du probe (figure 3.8b) qui nécessitaient un traitement particulier dans la formulation de Connolly [Con83].



(a) Patchs concaves **(b)** Singularité géométrique

Figure 3.8 – Deux patchs concaves (en bleu) associés aux intersections x_0 et x_1. Leur intersection résulte en une singularité géométrique (en violet).

Cette définition à partir des données structurelles de la SAS est particulièrement intéressante afin de représenter les patchs de façon compacte. Elle est donc à la base de notre méthode de construction, présentée dans la prochaine section.

3.2 Construction structurelle de la SES

Les éléments détaillés dans la partie précédente permettent le calcul géométrique des primitives de la SAS. Puisque celles-ci décrivent les fondations des patchs de la SES, ils sont au centre de sa construction. Ces opérations sont cependant dépendantes d'un accès aux données sur lesquelles elles sont basées, ce qui représente un aspect particulièrement crucial dans un contexte de performance en environnement parallèle. Leurs performances sont ainsi directement liées aux traitements et stockages associés.

Ainsi, dans cette section, nous présentons notre structure de données permettant de représenter la SES en fonction de composants de la SAS de façon compacte ainsi que les étapes nécessaires à sa construction visant une exécution parallèle performante.

3.2.1 Structure de données

La définition de la structure de données est centrale afin de répondre aux enjeux d'exhaustivité et de performance de la représentation complète de la SES et le support de très grandes protéines. Ses potentielles utilisations sont de plus directement dépendantes de sa définition puisqu'elles peuvent nécessiter l'accès performant à certaines données. Nous proposons ainsi une structure de données légère permettant le stockage des patchs de la SES de façon performante et sur laquelle repose notre méthode de construction.

Notre structure est une adaptation de celle présentée par Quan et Stamm [QS17]. Celle-ci est construite à partir de l'analyse mathématique détaillée en section 2.2 et représente explicitement les caractéristiques des patchs de la SES grâce à des composantes structurelles de la SAS. Elle n'est cependant pas adaptée à un traitement hautement parallèle puisque stockant un nombre important de propriétés pouvant notamment être recalculée efficacement. Nous proposons donc une adaptation permettant une construction, mais aussi un affichage efficace tout en limitant sa consommation mémoire.

Patch convexe P_+ Un patch convexe P_+ est représenté par une portion de la sphère de van der Waals correspondant à son atome a_i. Celle-ci est bornée par la projection des arcs formés par les cercles de la SAS auxquels elle contribue (figure 3.5). Certaines méthodes précédentes représentent explicitement ces polygones à partir de séquences d'arcs de petits cercles et de leurs intersections [TA96 ; QS17]. Cependant, ces éléments requièrent un calcul dédié, une agrégation ainsi qu'un stockage coûteux. Afin de représenter ces bornes, nous utilisons la stratégie de Rau et al. [Rau+19] qui stockent un ensemble de secteurs sphériques par patch équivalent à ses cercles de la SAS. Ils peuvent être totalement décrits à partir de l'axe du cercle de la SAS et de son angle d'ouverture permettant ainsi un stockage plus léger, comme illustré code source 3.1.

```
1 struct Secteur
2 |   float3 axeCercle
3 |   float angle
4 struct P+
5 |   uint32 i                    ▷ Indice de l'atome
6 |   uint32 nombreDeSecteurs
7 |   Secteur secteurs[]
```

Code Source 3.1 – Structure de données des patchs convexes P_+

Patch toroïdaux P_{tf}, P_t Notre dissociation des patchs toroïdaux complets P_{tf} avec ceux sous la forme d'arcs P_t permet de directement représenter les P_{tf} à partir des deux indices de leurs atomes correspondants, comme illustré code source 3.2. Le cercle de la SAS peut alors être recalculé efficacement en cas de besoin. Sur cette base, nous représentons les patchs P_t en ajoutant les indices des deux intersections les bornant.

```
1 struct P_tf
2 |   uint32 i, j              ▷ Indices des atomes
3 struct P_t
4 |   uint32 i, j              ▷ Indices des atomes
5 |   uint32 d, f              ▷ Indices des intersections
```

Code Source 3.2 – Structure de données des patchs toroïdaux P_t et P_{tf}

Patchs concaves P_- Les intersections de la SAS et les indices des atomes correspondants i, j et k sont suffisants pour totalement représenter la surface de contact du probe. Cependant, comme présenté section 2.2, deux positions du probe peuvent s'intersecter et produire un trou dans la surface. Il est ainsi courant de stocker explicitement les voisins des probes situés à une distance inférieure à $2r_p$ afin de faciliter l'affichage. Nous reproduisons ce stockage afin de bénéficier de performances d'affichage similaires, comme illustré code source 3.3.

```
1 struct P_-
2 |   float4 position
3 |   uint32 i, j, k           ▷ Indices des atomes
4 |   uint32 nombreDeVoisins
5 |   float4 voisins[]
```

Code Source 3.3 – Structure de données des patchs concaves P_-

La structure de données présentée est orientée vers un compromis entre légèreté et performance de construction et d'affichage. Elle est donc à la base de notre processus de calcul de la SES. Ainsi, les parties suivantes présentent les étapes de notre méthode de construction visant un calcul efficace de cette structure.

3.2.2 Cercles de la SAS

La construction de la SAS débute par ses cercles, à l'origine de toute sa structure. Ils résultent ainsi directement de la recherche de voisinage de chacun des atomes. Une sphère de la SAS d'un atome a_j d'une molécule $\mathcal{M}$ est caractérisée comme voisine de celle d'un atome $a_i \in \mathcal{M}$ si elles s'intersectent, et donc si leur distance est inférieure à la somme des rayons de leurs sphères de la SAS $||c_i - c_j|| < r_i + r_j + 2r_p$. Ces recherches de voisinage nécessitent ainsi une exploration spatiale performante.

Grille accélératrice

Les requêtes spatiales de ce type sont couramment réalisées à partir d'une grille accélératrice qui permet d'obtenir de bonnes performances, notamment lorsque les sphères sont de rayons similaires [Hoe14]. Nous construisons celle-ci efficacement en calculant un hash sur la position des sphères, qu'il suffit de trier afin de trouver les débuts et fins des cases de la grille. Une fois cette structure obtenue, elle peut être requêtée pour trouver les centres des sphères positionnés dans une case. Ainsi, afin de trouver tous les cercles d'une sphère d'un atome a_i, nous explorons toutes les cases localisées dans un rayon de $r_i + r_{max} + 2r_p$ à partir de son centre,

avec $r_{max} = \max(r_0, \ldots, r_n)$. Chaque voisin est alors caractérisé comme à l'origine d'un cercle de la SAS et stocké dans son voisinage $\mathcal{N}(a_i)$.

Puisque le parcours de chacune des cases de la grille représente un coût potentiel, il peut être intéressant d'adapter la taille de la grille en fonction du nombre et de la taille de sphères concernées. Afin de ne pas nécessiter un réglage manuel de ce paramètre, nous déterminons expérimentalement une taille de case en puissance de 2 et fonction du nombre d'atomes $\#\mathcal{M}$ avec $\min 2^m > \lceil \log_2(\#\mathcal{M}) \rceil$.

Classification

Chacune des sphères voisines dans $\mathcal{N}(a_i)$ participe à un cercle de la SAS. Celui-ci n'est cependant pas forcément contributeur de la SAS et donc de la SES. Comme illustré figure 3.9, trois classes de cercles sont possibles en fonction de la configuration d'un cercle $\mathcal{C}_{ij}$ par rapport à une troisième sphère de la SAS correspondant à l'atome a_k :

— cercle occulté : $\mathcal{C}_{ij}$ est complètement à l'intérieur de la sphère de la SAS de a_k et ne participe donc pas à la SAS ;
— cercle intersecté : $\mathcal{C}_{ij}$ est intersecté par la sphère de la SAS de a_k. Il participe à un patch convexe P_+ et potentiellement à un ou plusieurs patchs toroïdaux P_t et concaves P_- ;
— cercle complet : $\mathcal{C}_{ij}$ est complètement visible et participe alors à un patch convexe P_+ ainsi qu'à un patch toroïdal complet P_{tf}.

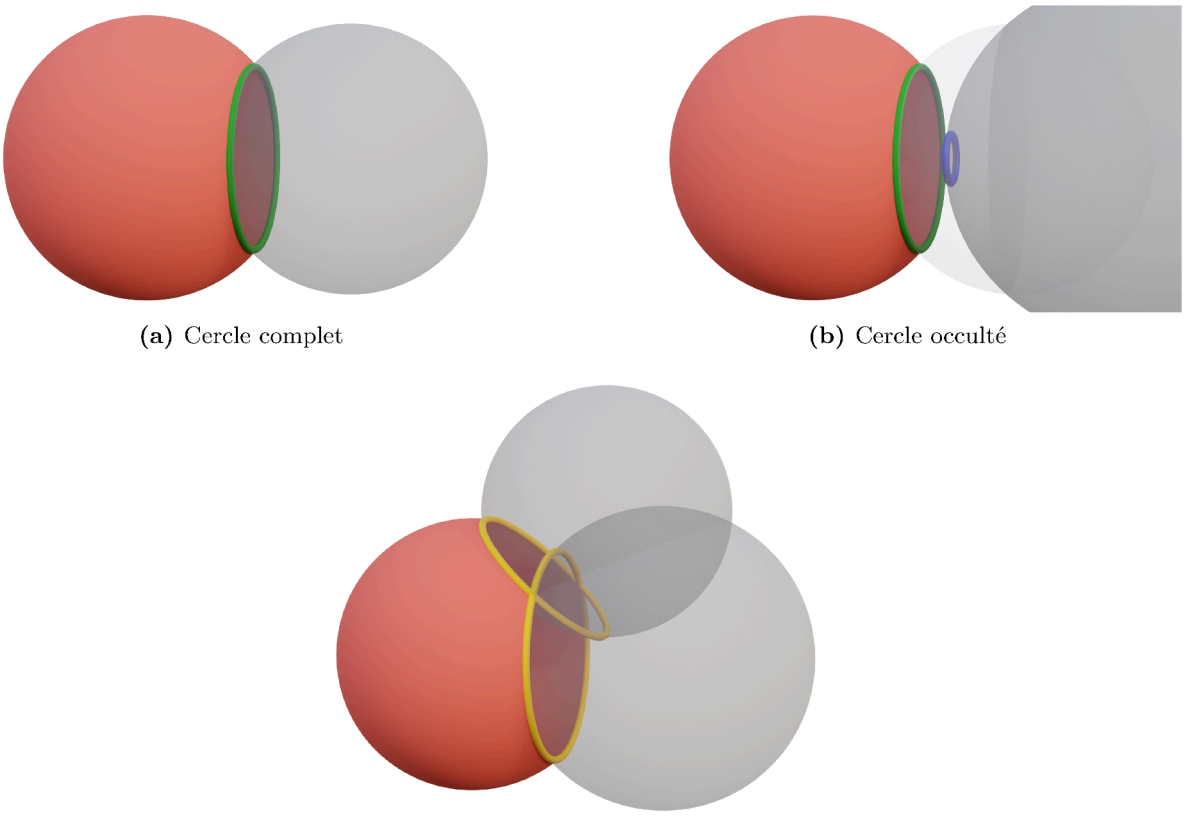
Ces configurations peuvent ainsi être directement dérivées des équations présentées section 3.1. Ainsi, nous cherchons les collisions entre chaque cercle $\mathscr{C}_{ij} \in \mathcal{N}(a_i)$ d'un atome a_i et un autre atome du voisinage $a_k \in \mathcal{N}(a_i) \setminus \{a_j\}$. Une classification similaire est utilisée par Krone et al. afin de détecter les cercles occultés [KGE11].

Cette classification est particulièrement intéressante puisqu'elle permet d'exclure un nombre important de cercles pour les calculs futurs. Ainsi, il n'est pas nécessaire de tester si un cercle occulté ou complet participe à une intersection de la SAS. Il est cependant important de noter qu'un cercle intersecté peut finalement ne pas contribuer à la SAS si aucune de ses intersections n'est visible.

3.2.3 Intersections de la SAS

Le traitement des intersections de la SAS est central au calcul de la SES et représente souvent la partie la plus consommatrice en termes de temps de calcul. En effet, il nécessite le parcours des cercles intersectés, semblable à un graphe de voisinage, qui est un type d'opération complexe à effectuer dans des environnements parallèles. Certains travaux précédents requièrent de plus la maintenance d'une structure pouvant représenter un coût important en mémoire ainsi qu'en temps de calcul. Dans ce contexte, nous utilisons une stratégie permettant une homogénéité d'exécution et ne nécessitant pas de stockage conséquent.

Les intersections de la SAS caractérisent chaque position du probe n'intersectant pas de sphères de van der Waals. Ainsi, une intersection est valide si et seulement si elle n'est pas positionnée à l'intérieur d'une sphère de la SAS. Comme certains travaux précédents [KDE10 ; QS16 ; SK19], nous utilisons ainsi un traitement basé sur la visibilité de l'intersection par rapport au voisinage de ses atomes. Pour chacun des cercles intersectés $\mathcal{C}_{ij} \in \mathcal{I}$ où $a_j \in \mathcal{N}(a_i)$, nous testons si une intersection existe avec un autre cercle $\mathcal{C}_{ik} \in \mathcal{I}$ où $a_k \in \mathcal{N}(a_i) \setminus \{a_j\}$. Dans cette éventualité, l'intersection appartient à la SAS si et seulement si son centre x n'est pas à l'intérieur d'une sphère de la SAS voisine $||x - c_l|| > r_l + r_p$, $\forall a_l \in \mathcal{N}(a_i) \setminus \{a_j, a_k\}$.



(a) Cercle complet

(b) Cercle occulté

(c) Cercle intersecté

Figure 3.9 – Illustration des différents types de cercles de la SAS et la façon avec laquelle ils limitent la surface de l'une de leurs sphères de la SAS (en rouge). Les cercles sont modifiés par des sphères de la SAS voisines (en gris). (a) Cercles complets (vert). (b) Cercles occultés (bleu) sans impact sur la sphère de la SAS. (c) Cercles intersectés (jaune).

3.2.4 Arcs de la SAS

Afin de représenter complètement la SAS, il est enfin nécessaire de reconstruire ses arcs à partir des intersections visibles trouvées. Ce traitement est souvent évité par les méthodes précédentes [KGE11] puisqu'il requiert, pour chacun des cercles, d'agréger toutes les intersections qui lui sont liées. Cette étape est particulièrement contraignante lorsque les intersections sont créées de façon parallèle. D'autres méthodes [QS16 ; MJK16] proposent de rassembler toutes les intersections d'un cercle afin de les traiter localement. Nous nous basons ainsi sur ce type de stratégie permettant de dissocier les traitements des intersections de ceux des arcs de la SAS.

Une fois que nous connaissons l'ensemble des intersections $\mathcal{X}(\mathscr{C}_{ij})$ d'un cercle donné $\mathscr{C}_{ij}$, le nombre d'arcs peut être déterminé à partir de

$$\frac{1}{2}\sum \#\mathcal{X}(\mathscr{C}_{ij}),$$

puisque chacun des arcs est borné par deux intersections de la SAS.

Les arcs peuvent ensuite être calculés en triant les intersections de façon angulaire. Comme illustré figure 3.10, deux intersections bornent le début et la fin d'un arc sur chacun des cercles

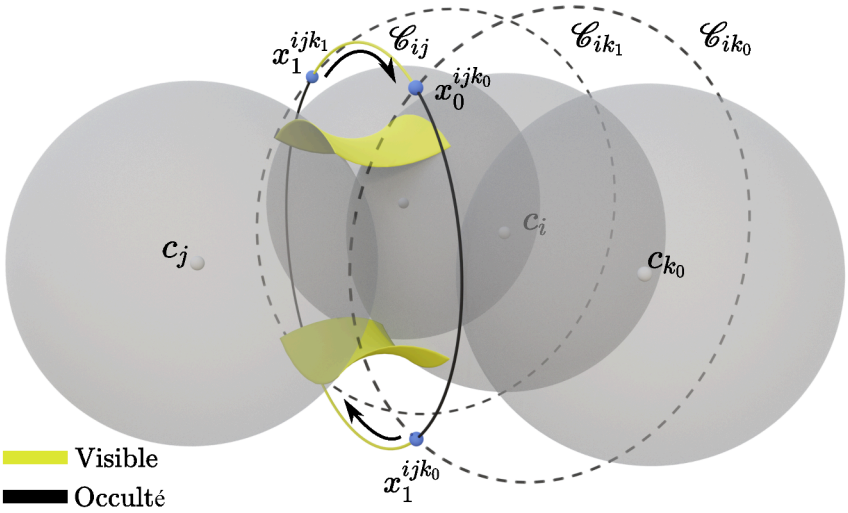


Figure 3.10 – Illustration de la reconstruction des arcs d'un cercle $\mathscr{C}_{ij}$ à partir de ses intersections. Une première intersection $x_1^{ijk_0}$ est sélectionnée de façon arbitraire et fixe l'ordre angulaire dans lequel la liste est triée. $x_1^{ijk_1}$ forme ainsi un arc avec $x_0^{ijk_0}$.

de la SAS de façon dépendante à l'ordre utilisé dans leurs équations, décrites section 3.1. Ainsi, en fonction de l'identifiant de la première intersection sélectionnée (*i.e.* si c'est la première ou seconde solution de (3.1)), nous définissons un sens de rotation à partir duquel les angles des autres intersections peuvent être calculés. Une fois ces derniers triés, chaque couple d'intersections successives constitue un arc de la SAS.

À la suite du calcul des arcs de la SAS, toutes les composantes nécessaires à la construction de la SES ont été obtenues. Ce processus définit ainsi un ensemble d'étapes compactes qui, comme présenté dans la section suivante, se prête à une implémentation massivement parallèle et un traitement coopératif visant de hautes performances de calcul.

3.3 Traitement parallèle et coopératif

Les opérations décrites dans la partie précédente permettent de construire la SES à partir de la structure de la SAS. Elles requièrent la définition de processus adaptés afin de répondre aux besoins de calcul performant et économe en mémoire. Ainsi, notre traitement est basé sur une mise en coopération des ressources de calcul permettant d'utiliser les capacités des architectures modernes des GPUs. Son architecture est illustrée figure 3.11. Celle-ci est comparable à celles des travaux précédents visant une exécution GPU [KGE11] mais présente plusieurs ajouts et modifications importantes permettant une construction efficace de la surface complète.

3.3.1 Construction des patchs convexes P_+ et toroïdaux P_{tf}

Les patchs toroïdaux complets P_{tf} et convexes P_+ peuvent être directement construits à partir de la classification des cercles de la SAS.

En effet, chaque cercle complet de la SAS est associé à un patch toroïdal complet P_{tf}. Lors de la classification, peu de ces cercles sont réellement identifiés. Nous utilisons donc une opération atomique afin de les enregistrer pendant cette étape et ne pas nécessiter de processus additionnel.

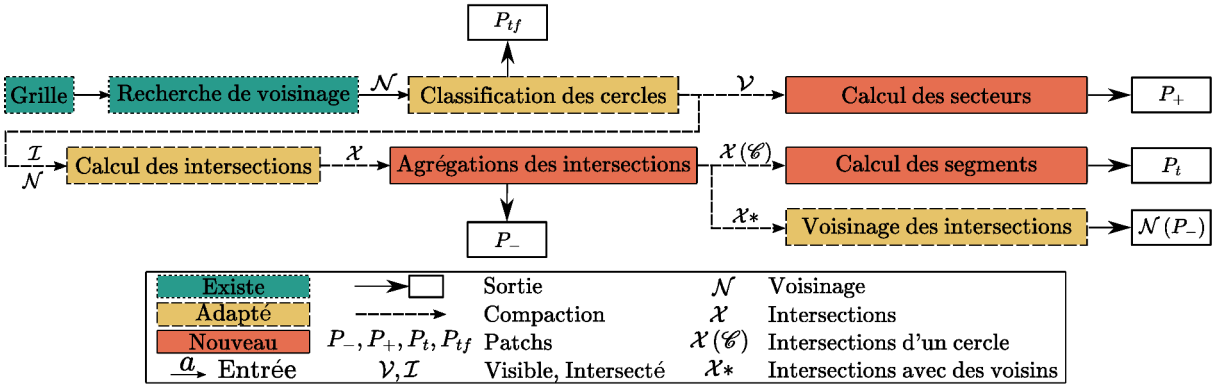


Figure 3.11 – Illustration du processus de construction de la SES. Nous proposons différentes modifications et nouvelles étapes par rapport à la méthode de Krone et al. [KGE11] afin de construire la SES complète de façon performante. Les données des patchs sont directement sauvegardées à partir du moment où elles sont disponibles dans la chaine de traitement.

Les performances de ce type d'instruction sont directement liées au nombre de threads les utilisant. Dans ce contexte, il est rare que deux cercles complets soient sauvegardés au même moment, ce qui a donc peu d'impact sur les performances.

Enfin, chaque cercle intersecté ou complet d'un atome a_i est utilisé afin de calculer les secteurs de son patch convexe. Leurs axes sont obtenus à partir de celui de leurs cercles tandis que leurs rayons sont calculés avec

$$\cos^{-1}\left(n_{ij} \cdot \frac{o - c_i}{r_i + r_p}\right), o \in \mathscr{C}_{ij}.$$

Grâce à ces deux opérations, les patchs P_+ et P_{tf} sont créés de façon efficace et parallèle tout en nécessitant peu de traitement dédié.

3.3.2 Traitement coopératif parallèle des intersections de la SAS

Le calcul des intersections de la SAS constitue l'un des principaux enjeux de la construction de la SES. Puisqu'elles sont obtenues à partir du graphe de voisinage des sphères de la SAS, elles sont difficiles à calculer de façon uniforme et parallèle sur GPU. Nous présentons donc un traitement visant à utiliser les propriétés coopératives des capacités modernes de ce type d'architecture parallèle.

Le principal enjeu de notre traitement est la distribution de la puissance de calcul à travers les différentes étapes nécessaires au calcul des intersections. En effet, il est complexe de répartir de façon uniforme les tâches lors du parcours du graphe de façon parallèle. Ainsi, on souhaite identifier les couples de cercles $\mathscr{C}_{ij}$ et $\mathscr{C}_{ik}$ de l'atome courant a_i produisant une intersection avec $i < j < k$ afin d'éviter toute redondance. En associant un thread par cercle de la SAS de a_i, la majorité des threads ne trouveront pas d'intersection tandis qu'une minorité devra identifier si celles trouvées sont visibles, entrainant ainsi une divergence d'exécution importante, des threads inactifs et donc des ralentissements. Pour pallier ce problème, nous établissons un processus dynamiquement parallèle.

Notre traitement est ainsi basé sur la division des tâches à effectuer à l'échelle d'un warp et est détaillé algorithme 1. Une illustration est enfin proposée figure 3.12. Afin d'éviter que la plupart des threads attendent qu'une minorité teste la visibilité de quelques intersections effectivement possibles, nous associons chacun des cercles intersectés $\mathscr{C}_{ij} \in \mathcal{I}$ de a_i à un warp. Au cours de ce processus, nous souhaitons répartir la charge des traitements simples effectués pour trouver les intersections possibles, et ceux qui sont plus lourds permettant de tester si elles sont visibles ou non. Pour cela, nous associons à chaque thread du warp une combinaison $(a_i, a_j, a_k), a_k \in \mathcal{N}(a_i) \cap \mathcal{I}$ ainsi qu'une valeur booléenne restant vraie tant qu'une combinaison liée à des intersections de la SAS reste possible. Afin de mettre en commun ces valeurs et distribuer les tâches à effectuer de façon efficace, nous utilisons un masque binaire noté $\mathcal{K}$ stockant les valeurs d'existences booléennes des threads.

Notre traitement commence ainsi par répartir les combinaisons possibles en fonction des indices des threads (l. 8-10 de algorithme 1). Les threads calculent ensuite les intersections à partir des combinaisons qui leur ont été associées. Si les intersections ne sont pas possibles (*i.e.* les trois sphères de la SAS ne s'intersectent pas), les intersections sont invalidées dans $\mathcal{K}$

Algorithme 1 : Traitement coopératif des intersections

Input : Le voisinage de l'atome $\mathcal{N}(a_i)$ et les cercles intersectés $\mathcal{I}$
Output : Les intersections de la SAS $\mathcal{X}$

```
1  block.load(N(a_i))                                   ▷ En mémoire partagée
2  X' ← ∅                                               ▷ En mémoire partagée
3  t ← thread.id()                                      ▷ Index relatif au warp
4  block.sync()
5  forall C_ij ∈ I, a_j ∈ N(a_i), i < j do              ▷ Un cercle par warp
6  |  warp.load(N(a_j))                                ▷ Depuis la mémoire globale
7  |  K ← ¬0                                           ▷ Initialise le masque
8  |  forall k ∈ {0,...,size(warp)} and K[k] ≠ 0 do
9  |  |  K[k] ← warp.test(a_k ∈ N(a_j))
10 |  end
11 |  if K[t] ≠ 0 then
12 |  |  K[t] ← exists(x_ijk_t)
13 |  end
14 |  forall k ∈ {0,...,size(warp)} et K[k] ≠ 0 do
15 |  |  if warp.test(visible(x_ijk, N(a_i))) then
16 |  |  |  X' ← X' ∪ {x_ijk}                          ▷ Sauvegarde temporaire
17 |  |  end
18 |  end
19 end
20 block.sync()
21 forall x_ijk ∈ X' do                                ▷ Sauvegarde en mémoire globale
   |  ▷ Opérations atomiques par cercle
22 |  #X(C_ij) ← #X(C_ij) + 1
23 |  #X(C_ik) ← #X(C_ik) + 1
24 |  #X(C_jk) ← #X(C_jk) + 1
25 |  X ← X ∪ {x_ijk}
26 end
```

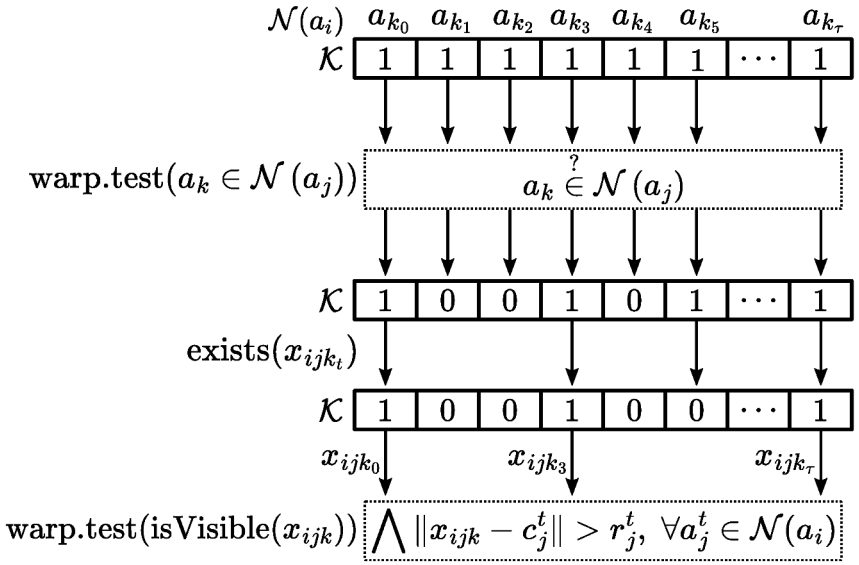


Figure 3.12 – Illustration du processus coopératif de calcul des intersections de la SAS par warp. La possibilité qu'un voisin a_{k_t} du cercle $\mathscr{C}_{ij}$ produise une intersection visible est stockée dans le masque $\mathcal{K}$. Ce processus bénéficie de différents niveaux de parallélisme : par threads (flèches pleines) et séquentiel au warp (cases en pointillés).

(l. 11-13 de algorithme 1). Il reste enfin à tester si les intersections restantes sont visibles et donc ne sont pas à l'intérieur d'une autre sphère de la SAS. Pour chacune des valeurs restantes actives de $\mathcal{K}$, on associe à tous les threads du warp un sous-ensemble du voisinage $\mathcal{N}(a_i)$. Les threads testent alors si l'intersection courante est occultée (l. 14-17 de algorithme 1). Si cette dernière est visible, elle est ajoutée à une liste temporaire en mémoire partagée avant sauvegarde en mémoire globale. Le calcul des intersections bénéficie ainsi d'une répartition dynamique de la charge de traitement et, même si des divergences sont possibles, les tâches les plus lourdes sont partagées afin de réduire leur impact.

Puisque nous ne calculons les intersections d'un cercle $\mathscr{C}_{ij}$ que si $i < j < k$, il est nécessaire de retrouver l'ensemble des intersections $\mathcal{X}(\mathscr{C}_{ij})$ d'un cercle donné $\mathscr{C}_{ij}$ afin de reconstruire ses arcs de la SAS. Pour cela, nous incrémentons un compteur $\#\mathcal{X}(\mathscr{C}_{ij})$ pour chaque cercle concerné par l'enregistrement d'une intersection (l. 22-24 de algorithme 1). Une fois cette information obtenue, nous réservons l'espace nécessaire au tableau des indices des intersections d'un cercle donné $\mathcal{X}(\mathscr{C}_{ij})$. Celui-ci est rempli dans un second temps en utilisant un thread par intersection écrivant son indice correspondant dans $\mathcal{X}(\mathscr{C}_{ij})$. Ce traitement, aussi basé sur des opérations atomiques, bénéficie du faible nombre d'intersections par cercle permettant peu de ralentissement au cours de la modification des valeurs.

Les intersections de la SAS calculées peuvent directement être utilisées pour créer les patchs P_-. Il est cependant nécessaire de calculer leur voisinage afin de supporter les singularités de ces patchs au moment de l'affichage. Comme noté par des travaux précédents [SOS96], seuls les patchs dont le probe dépasse le plan décrit par les centres de ses atomes tangents peuvent comprendre une singularité. Nous identifions donc ces intersections et les positionnons au début du tableau les contenant. La recherche de voisinage n'est alors effectuée que sur ce sous-ensemble, permettant ainsi des accès mémoires performants et une réduction des calculs inutiles.

3.3.3 Traitement coopératif parallèle des arcs de la SAS

Une fois les intersections de chaque cercle agrégées, il reste à reconstruire l'ensemble des arcs, dont le traitement est illustré algorithme 2.

Grâce à la construction de l'ensemble des intersections liées à un cercle donné, nous utilisons un thread par cercle. Celui-ci charge une première intersection qui définit le sens de calcul des angles des intersections suivantes (trigonométrique ou inverse). Il est ensuite nécessaire d'effectuer un tri angulaire afin de retrouver les paires d'intersections donnant les arcs. La nature séquentielle des algorithmes de tri représente un facteur de divergence très important. Afin de pallier ce problème, les bibliothèques de traitement GPU [NVIa] proposent couramment des alternatives utilisant plusieurs threads et bénéficiant de leur coopération. Ces algorithmes requièrent cependant la mise en commun des informations à trier ce qui ne permet pas de retrouver celles qui sont liées à un seul thread. Puisqu'il est nécessaire de ne pas mélanger les intersections de cercle appartenant à différents threads entre elles, nous appliquons un facteur de décalage aux valeurs angulaires des intersections calculées en fonction de l'indice du thread correspondant (l. 4 et 12 de algorithme 2). Ainsi, le tri peut être effectué par warp en mettant en commun les ensembles de données de chaque thread tout en conservant les valeurs liées à chacun d'entre eux dans des zones définies.

Algorithme 2 : Traitement coopératif des arcs de la SAS

Input : Les intersections d'un cercle $\mathcal{X}(\mathscr{C}_{ij})$
Output : Les arcs de la SAS $\mathcal{S}$

1 $x \leftarrow \mathcal{X}(\mathscr{C}_{ij})$
2 inverted $\leftarrow$ isInverted($\mathscr{C}_{ij}$, $x[0]$)
3 $t \leftarrow$ thread.id() ▷ Index relatif dans le warp
4 $\delta_t \leftarrow t\delta,\ \delta > 2\pi$
5 angles $\leftarrow \{0\}$ ▷ Vers la mémoire du thread
6 indices $\leftarrow \{0\}$ ▷ Vers la mémoire du thread
7 **for** $b \in \{1, \ldots, \#\mathcal{X}(\mathscr{C}_{ij}) - 1\}$ **do**
8 $\quad \alpha, \beta \leftarrow x[b-1], x[b]$
9 $\quad$ **if** *inverted* **then**
10 $\quad\quad \alpha, \beta \leftarrow \beta, \alpha$
11 $\quad$ **end**
12 $\quad \theta \leftarrow \delta_t + \text{angle}(\alpha, \beta)$
13 $\quad$ angles $\leftarrow$ angles $\cup \{\theta\}$
14 $\quad$ indices $\leftarrow$ indices $\cup \{b\}$
15 **end**
16 indices $\leftarrow$ warp.sortByKey(angles, indices)
17 **for** $s \in \{0, \ldots, \#\mathcal{X}(\mathscr{C}_{ij})/2 - 1\}$ **do**
18 $\quad b, c \leftarrow$ indices$[2s]$, indices$[2s+1]$
19 $\quad \alpha, \beta \leftarrow x[b], x[c]$
20 $\quad$ **if** *inverted* **then**
21 $\quad\quad \alpha, \beta \leftarrow \beta, \alpha$
22 $\quad$ **end**
23 $\quad \mathcal{S} \leftarrow \mathcal{S} \cup \{i, j, \alpha, \beta\}$ ▷ Vers la mémoire globale
24 **end**

Enfin, notre méthode visant l'affichage de la SES, il est intéressant de proposer une structure de données nécessitant peu de calcul supplémentaire au moment de son utilisation. Dans l'éventualité d'un sens trigonométrique inverse, nous inversons chacune des intersections. Elles peuvent ainsi être utilisées de façon uniforme sans nécessiter de traitement spécifique.

3.3.4 Estimation des besoins en mémoire

Le stockage des composantes de la SES représente un enjeu important pour le support d'équipements matériels disposant de faibles capacités mémoires ainsi que de grandes structures. Notamment, l'implémentation de Krone et al. [KGE11] utilise une estimation très large du nombre d'intersections possibles $\#\mathcal{M}(\max \#\mathcal{N})^2$, avec $\max \#\mathcal{N}$ la taille maximale du voisinage et $\#\mathcal{M}$ le nombre d'atomes dans la molécule.

Afin d'obtenir une estimation plus précise, nous utilisons la classification des cercles présentés section 3.1. Comme illustré figure 3.13, le nombre de cercles intersectés représente de façon stable moins de la moitié de l'ensemble des cercles. La majorité d'entre eux ne contribue cependant pas à la SAS à travers la création de segments. Ainsi, nous estimons le nombre d'intersections $\#\mathcal{X}$ à partir du nombre de cercles intersectés et d'un facteur empirique $\max(\#\mathcal{X}(\mathscr{C}_{ij}))$

$$\#\mathcal{X} \leq \#\mathcal{I} \max(\#\mathcal{X}(\mathscr{C}_{ij})).$$

Dans notre implémentation, nous bornons $\max(\#\mathcal{X}(\mathscr{C}_{ij})) = 2$, une valeur permettant de calculer la surface de molécules très variées dans nos expériences. Cette limite ne représente cependant qu'une valeur moyenne puisque, dans les faits, notre implémentation supporte des cercles contenant jusqu'à 16 intersections.

Les étapes présentées dans cette section permettent un calcul coopératif et parallèle des composantes structurelles de la SAS à partir desquelles nous construisons la structure de données

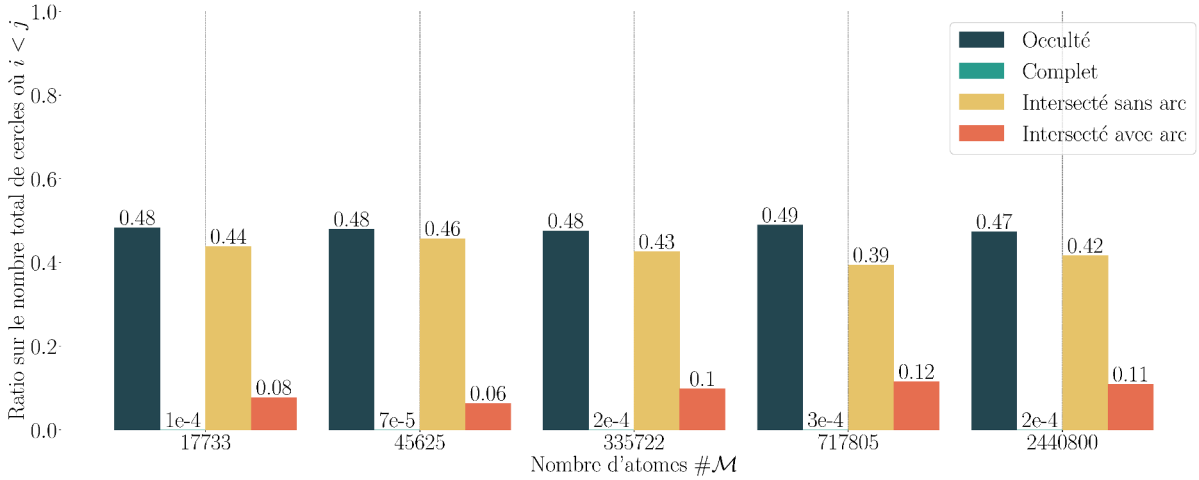


Figure 3.13 – Ratios des nombres de cercles par types sur le nombre total de cercles de la SAS pour lesquels $i < j$ pour un rayon du probe de 1.4Å. Une moitié des cercles sont occultés tandis qu'une grande majorité des cercles de l'autre moitié sont intersectés. Cette dernière partie est constituée de cercles ne contribuant finalement pas à la SAS parce que toutes leurs intersections sont occultées. La proportion de cercles intersectés contribuant à la SAS varie en fonction de la structure de la protéine visée.

nécessaire à l'affichage de la SES. Dans la section suivante, nous évaluons les capacités de notre stratégie et les comparons avec la méthode précédente la plus rapide.

3.4 Résultats

Dans cette section, nous présentons les résultats de notre méthode de calcul de la SES. Nous la comparons à l'implémentation de la méthode de Krone et al. [KGE11], disponible publiquement dans le logiciel Megamol [Gra+19], qui propose les meilleures performances de l'état de l'art pour la construction de la SES extérieure et dont la principale limite se situe sur la consommation mémoire. Puisque cette dernière ne permet que la construction de la surface extérieure, nous proposons aussi une adaptation de notre méthode pour le calcul de la surface extérieure en supprimant le traitement des patchs P_+ et P_t. Cette dernière permet ainsi une meilleure comparaison avec Contour-Buildup sur GPU. Nous utilisons un jeu de données contenant des molécules de tailles très différentes afin d'étudier la mise à l'échelle des méthodes étudiées. Notre méthode est configurée avec un nombre maximal de voisins par atome de 128 afin de supporter toutes les protéines testées. Même si notre méthode supporte de plus gros rayons, comme illustré figure 3.14, toutes les expériences utilisent un rayon de probe de 1.4Å, couramment utilisé comme approximation d'une molécule d'eau. Enfin, tous les tests présentés sont effectués sur un AMD Ryzen 5 1600 et une NVIDIA RTX 2080 disposant de 8Go de mémoire.

Le tableau 3.2 présente les performances de calcul ainsi que la consommation mémoire des méthodes testées. Nous remarquons tout d'abord que notre méthode complète propose des temps de construction similaires à celle de Krone et al. tandis que le calcul restreint à la surface extérieure permet un gain constant. Notre méthode permet ainsi de calculer la SES complète sans impact sur les performances de calcul, ce qui n'est pas possible avec les méthodes précédentes. La faible différence entre le calcul de la SES complète et extérieure illustre ainsi le faible coût des opérations liées à la SES complète que nous proposons. Enfin, la répartition spatiale des atomes a un impact sur les performances de notre méthode, comme constaté par Schäfer et

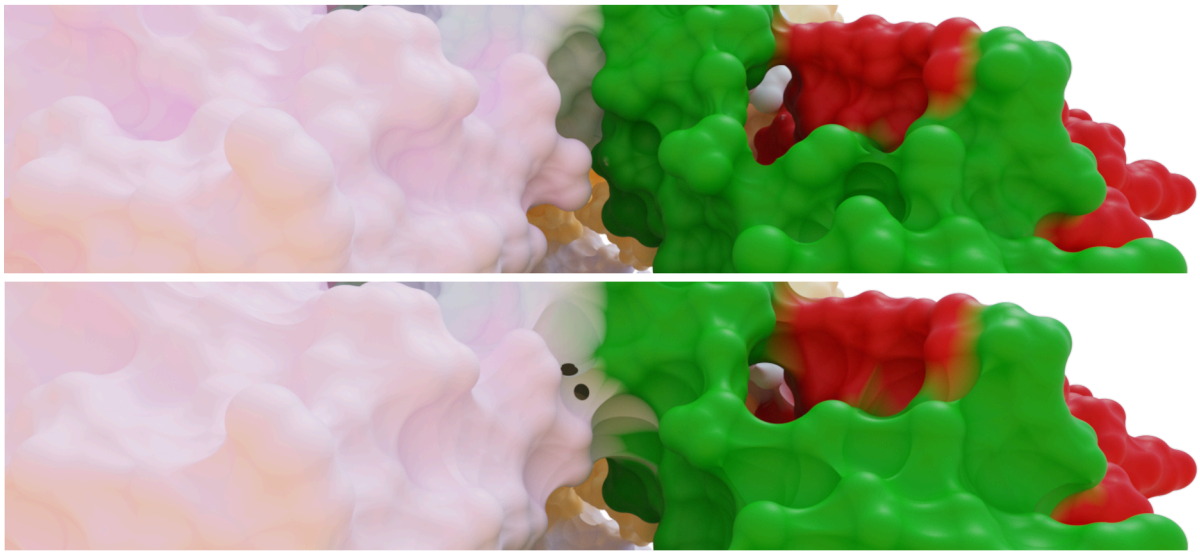


Figure 3.14 – Illustration de la SES d'une protéine (PDB : 1AON) calculée avec notre méthode en utilisant des rayons de probe différents : 1.4Å (haut) et 2.4Å (bas).

PDB	#Atomes	Krone et al. [KGE11]		Notre méthode		Notre méthode extérieure	
		Temps	Mémoire	Temps (ms)	Mém. (Mb)	Temps (ms)	Mém. (Mb)
1AGA	126	4.50	11.92	3.22	0.39	2.21	0.15
101M	1413	4.51	133.55	3.75	4.50	2.51	1.76
1VIS	2531	4.64	238.66	3.67	8.01	2.65	3.52
7SC0	11638	5.64	1108.02	4.70	36.17	3.54	16.16
3EAM	13505	6.44	1282.09	5.20	42.69	4.01	19.63
7DBB	17733	7.00	1675.43	5.80	56.18	4.49	25.50
1A8R	45625	11.14	4307.71	9.48	144.92	7.36	65.79
7O0U	55758	12.39	5263.23	10.71	177.25	8.49	80.29
1AON	58870	11.95	5552.49	10.75	185.33	8.37	86.86
7RGD	65008	16.86	6123.51	18.06	214.33	14.64	105.35
3JC8	107640	-	-	14.99	324.03	11.60	147.57
7CGO	335722	-	-	47.64	1054.85	36.34	486.03
4V4G	717805	-	-	104.72	2249.47	81.64	1130.35
6U42	1358547	-	-	200.79	4287.12	157.23	2162.99
3J3Q	2440800	-	-	700.17	7631.24	324.22	3744.73

Table 3.2 – Comparaison de notre méthode avec l'implémentation de Contour-Buildup sur GPU [KGE11] mise à disposition dans Megamol [Gra+19] sur un AMD Ryzen 5 1600 et une NVIDIA RTX 2080. Les temps de calcul sont donnés en millisecondes (ms) et représentent la moyenne de 1000 itérations. Chacune des expériences est précédée de 100 itérations non comptées dans le résultat final. L'utilisation mémoire est quant à elle donnée en mégaoctet (Mo).

Krone [SK19]. En effet, une protéine comme 3JC8 est ainsi calculée plus rapidement qu'une plus petite protéine comme 7RGD présentant pourtant 1.6 fois moins d'atomes.

Nous remarquons de plus que la méthode de Krone et al. [KGE11] requiert beaucoup plus de mémoire ce qui la limite à une protéine de 65000 atomes sur le GPU utilisé. En comparaison, les deux versions de notre méthode permettent le calcul de structures allant jusqu'à 2.5 millions d'atomes. Notre méthode complète consomme ainsi en moyenne 30 fois moins de mémoire que la méthode de Krone et al. tandis que le calcul de la surface extérieure consomme 60 fois moins de mémoire sur la plus grande protéine supportée par les trois méthodes. Ces économies sont notamment permises par notre structure de données ainsi que notre stratégie d'estimation des besoins. Ce gain est particulièrement intéressant pour la construction de surface de grandes protéines, comme celle présentée figure 3.16, ou pour l'exécution sur des GPUs disposant de moins de mémoire que celui utilisé. La différence entre la consommation mémoire de la construction complète et extérieure est principalement due au stockage explicite des cercles des patchs convexes P_+ sous la forme de secteur, mais aussi des patchs toroïdaux P_t nécessitant les indices des intersections de la SAS qui leur sont liés.

Nous analysons de façon plus précise les performances de notre méthode figure 3.15. Nous pouvons tout d'abord remarquer que la majorité du temps de calcul est dédiée au traitement des cercles et des intersections, de façon indépendante à la molécule traitée. Ainsi, même s'ils sont effectués de façon performante, la construction et le parcours du graphe des cercles représentent toujours les parties les plus complexes du traitement. Notamment, les nombreux cercles détectés comme intersectés, mais ne contribuant finalement pas à la SAS requièrent des temps de calcul importants avant d'être invalidés dans la suite des traitements. La création des arcs ainsi que du voisinage des intersections sont cependant effectués en une fraction du temps de calcul global grâce à la compacité des opérations nécessaires. Nous remarquons enfin encore une fois que les répartitions spatiales des atomes des protéines impactent les temps de calcul. Ainsi,

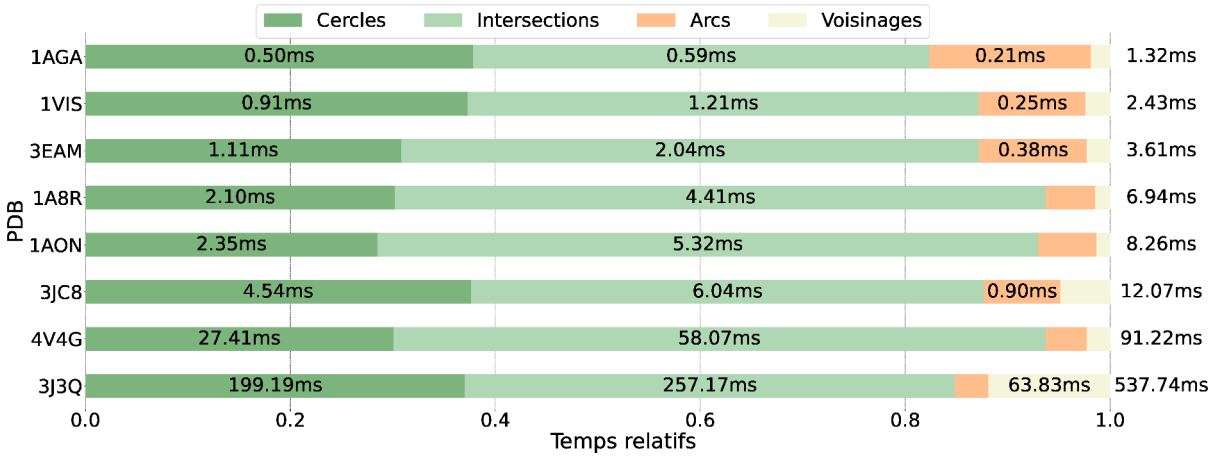


Figure 3.15 – Comparaison des temps de calcul GPU de notre méthode sur différentes protéines sur un AMD Ryzen 5 1600 et une NVIDIA RTX 2080. La majorité du temps de calcul est consacrée au traitement des cercles et au calcul des intersections. La création des arcs et du voisinage des intersections sont réalisés en une fraction du temps de calcul global.

certaines protéines proposent plus d'intersections de la SAS que d'autres résultant en des temps de calcul de leur voisinage plus importants.

Même si notre méthode vise uniquement le calcul de la SES, nous étudions l'affichage de la structure calculée avec deux systèmes de rendu. Le premier est basé sur une stratégie de rastérisation matérielle d'une façon similaire à Krone et al. [KBE09]. Ainsi, nous calculons à la volée des quadrilatères couvrant la sphère englobante du patch concerné, qui est ensuite intersecté avec un rayon. Cette dernière étape utilise les mêmes primitives que Krone et al. dans le cadre de la surface extérieure, et les fonctions distance signées (présentées section 3.1.2) pour les patchs toroïdaux de la surface complète. Les performances de ce système sont présentées tableau 3.3. Les deux types de surface permettent un affichage interactif, même avec de grandes protéines. La surface complète implique cependant un coût restant modéré sur la plupart du jeu de données, mais qui requiert des temps de calcul trois fois plus importants sur la plus grande protéine. Cela peut être expliqué par la prise en compte des patchs toroïdaux P_t, nécessitant une recherche d'intersection qui n'est pas analytique, mais aussi des bordures des patchs convexes P_+

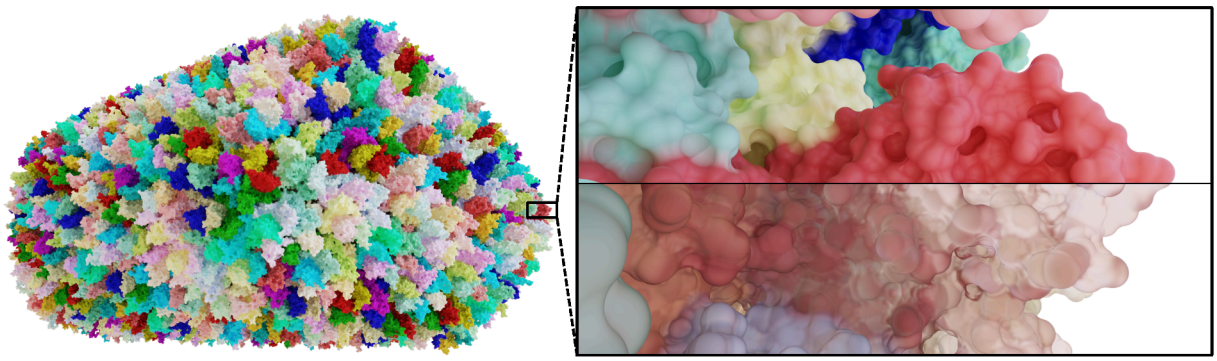


Figure 3.16 – Illustration de la SES complète de la plus grande protéine (PDB : 3J3Q) de notre jeu de données et obtenue avec notre méthode. Elle est calculée en 700ms sur une NVIDIA RTX 2080 et est compatible avec un affichage haute qualité comme présenté.

PDB	#Atomes	Surface complète (ms)	Surface extérieure (ms)
1AGA	126	2.08	2.07
101M	1413	3.44	2.78
1VIS	2531	3.78	3.50
7SC0	11638	5.40	4.73
3EAM	13505	6.97	6.31
7DBB	17733	4.89	4.05
1A8R	45625	6.41	5.35
7O0U	55758	10.72	8.41
1AON	58870	9.95	8.59
7RGD	65008	12.74	9.05
3JC8	107640	7.60	7.74
7CGO	335722	20.74	17.57
4V4G	717805	19.31	17.05
6U42	1358547	37.81	36.59
3J3Q	2440800	94.42	32.04

Table 3.3 – Performance de l'affichage avec notre système de rendu basé sur la méthode de Krone et al. [KBE09]. Les surfaces sont calculées avec un rayon de probe de 1.4Å. Les temps sont donnés en millisecondes et représentent la moyenne de 1000 itérations effectuées à partir de point de vue aléatoire orienté vers le centre de la protéine et dans une sphère englobant la boite englobante de la protéine. Les rendus sont effectués avec un matériau diffus lambertien.

dépendant du nombre de cercles des sphères de la SAS. Ce coût pouvant freiner une application visant des configurations matérielles modérées, l'utilisation de la surface extérieure peut ainsi servir en remplacement dans ce type de cas.

La deuxième méthode de rendue est basée sur le système OptiX [Par+10] permettant l'accélération matérielle du lancer de rayon. Elle vise ainsi la production d'image haute qualité comme celles présentées dans certaines illustrations de cette partie et notamment figure 3.14 ainsi que figure 3.16. Comme illustré, la surface proposée permet une bonne visualisation de la structure de la protéine. Vue de loin, la forme générale est correctement définie et permet d'identifier les principaux composants tandis que de près elle permet de visualiser les détails de la surface. Enfin, le calcul complet de la surface permet une représentation transparente, particulièrement utile afin de visualiser les potentielles cavités [Jur+16].

Notre méthode est ainsi compatible avec le calcul de la surface complète de très grandes protéines et permet une visualisation rapide et détaillée. Elle est cependant limitée par différents facteurs pouvant donner lieu à des améliorations possibles.

3.5 Limites et travaux futurs

La stratégie présentée dans cette section répond à différents besoins de la visualisation et illustration moléculaire pour lesquelles les précédentes méthodes sont trop limitées. Elle est cependant contrainte par certaines de ses caractéristiques.

La classification des cercles présentée section 3.2.2 permet de borner efficacement la mémoire nécessaire au calcul de la surface. Cependant, de nombreux cercles sont classifiés comme intersectés

sans pour autant contribuer à une intersection visible. Ainsi, le traitement pourrait être effectué de façon plus performante en identifiant ces cercles avant la recherche d'intersection.

Les différentes étapes proposées sont prévues pour une exécution sur GPU et permettent des temps de calcul similaires, ou meilleurs, que les méthodes précédentes. Cependant, de nombreux transferts aux CPUs sont nécessaires ainsi que des allocations dynamiques restreignant la parallélisation. De meilleures performances pourraient alors être obtenues en évitant ce type de transfert mémoire.

Enfin, de nombreux outils sont basés sur l'utilisation de maillage pour la représentation de la surface moléculaire, notamment pour des simulations moléculaires. Une adaptation parallèle de la méthode de Quan et Stamm sur la base de notre stratégie pourrait permettre la construction efficace de ce type de géométrie [QS17].

3.6 Conclusion

La méthode présentée dans ce chapitre permet un calcul complet de la SES de grandes protéines sur GPU de façon performante. Ces fonctionnalités sont permises grâce à l'introduction d'opérations dédiées au calcul d'éléments structurels de la SES satisfaisant les contraintes des environnements parallèles.

Les performances de notre méthode bénéficient particulièrement des opérations coopératives. Notamment, elles permettent une gestion dynamique de la puissance de calcul adaptée à la complexité variable des tâches à traiter. Ces dernières pouvant être à l'origine de divergence importante sur GPU, notre méthode permet ainsi une répartition plus fine des besoins en calcul.

La structure construite est légère et adaptée au rendu de la surface. Elle peut être directement utilisée afin d'afficher une surface précise dont les détails sont correctement visualisables. Ainsi, notre méthode souffre moins des compromis entre sa consommation mémoire, ses performances de calcul, et de la qualité de la surface obtenue. L'affichage de notre géométrie permet une visualisation de haute qualité compatible avec une exploration en temps réel, mais aussi pour l'illustration de grands complexes moléculaires.

Nous avons testé notre méthode sur un rayon de probe de 1.4Å, comme couramment utilisé. Cependant, il est parfois nécessaire de le faire varier afin d'étudier son impact sur la surface moléculaire. Même si de potentielles optimisations sont envisageables, ce type d'opération nécessite un recalcul complet de la surface avec notre méthode. De plus, la complexité de cette dernière est directement liée aux nombres de voisins des sphères de la SAS et donc à la taille du probe. Les cercles de la SAS sont contenus dans les caractéristiques des diagrammes d'Apollonius qui ne nécessitent pas de calculer le voisinage des sphères de la SAS. Ils sont ainsi couramment utilisés afin de proposer un calcul efficace de la SES, même avec des probes de grands rayons [MJK16].

Nos travaux se sont donc orientés vers la construction de diagrammes d'Apollonius afin de profiter des nombreuses fonctionnalités qu'ils proposent. Puisque ces diagrammes restent moins bien connus que les diagrammes de Voronoï ou de Puissance, et complexes à calculer sur des ensembles de grandes tailles, notre étude commence par leur caractérisation mathématique, présentée dans le chapitre suivant (chapitre 4) qui est à la base de notre méthode de construction efficace sur GPU (chapitre 5).

Chapitre 4

Caractérisation mathématique du diagramme d'Apollonius

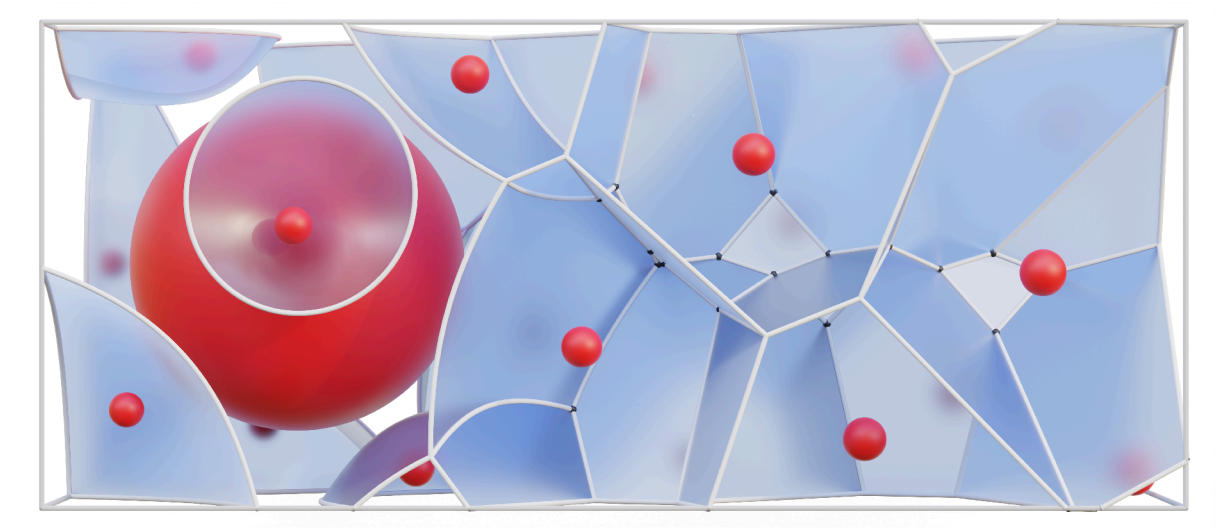


Figure 4.1 – Illustration du diagramme d'Apollonius d'un ensemble de sites dans $\mathbb{R}^3$ à partir de notre paramétrisation.

Sommaire

L'étude des diagrammes d'Apollonius a permis le développement de ses algorithmes de construction. En effet, les diagrammes d'Apollonius présentent des composantes plus diverses que les diagrammes de Voronoï affines tels que les diagrammes de Voronoï ou de Puissance. Notamment, leurs faces courbes complexifient de façon très importante les a priori possibles sur leurs topologies. Ainsi, la connaissance des propriétés de ces diagrammes est nécessaire au développement d'algorithmes de construction exhaustif.

Différents travaux étudiant les caractéristiques des diagrammes d'Apollonius ont été détaillés chapitre 2. Cependant, malgré le nombre de ces derniers, il n'existe pas à notre connaissance d'études complètes des diagrammes d'Apollonius en deux et trois dimensions. Ainsi, quelques études ont été proposées visant la caractérisation de certains éléments du diagramme comme les faces [Wan+20], les arêtes [Wil99] ou les sommets [GR03]. Certains résultats sont cependant encore manquants dans la littérature.

Sur la base de ces constats, et afin de présenter une fondation théorique solide permettant le développement d'une méthode de construction rapide et complète des diagrammes d'Apollonius, nous présentons dans ce chapitre une caractérisation complète permettant la paramétrisation et le calcul de la géométrie dans $\mathbb{R}^2$ et $\mathbb{R}^3$. Notre étude vise des ensembles de sites en position générale et donc qui ne présentent aucun sous-ensemble de $d+1$ sites contenus dans un même hyperplan de $(d-1)$-dimensions. De plus, nous utilisons l'hypothèse qu'aucun site n'est localisé

Symbole	**Signification**
d	Distance euclidienne.
$\delta^{(2)}$	Distance de puissance.
δ	Distance euclidienne additivement pondérée ou distance à la sphère.
x	Un point.
Γ	Un cône.
p, $\mathcal{P}$	Un site et un ensemble de sites.
r, s, $\mathcal{S}$	Un poids, un site pondéré et un ensemble de sites pondérés.
$\tilde{\square}$	L'objet correspondant à $\square$ dans $\mathbb{R}^{d+1}$.
$\mathcal{V}(\mathcal{P})$, $\mathcal{V}(p)$, Π	Un diagramme de Voronoï, une cellule de Voronoï et une face de Voronoï.
$\mathcal{P}(\mathcal{S})$, $\mathcal{P}(s)$, Π^2	Un diagramme de Puissance, une cellule de Puissance et une face de Puissance.
$\mathcal{A}(\mathcal{S})$, $\mathcal{A}(s)$, H	Un diagramme d'Apollonius, une cellule d'Apollonius et une face d'Apollonius.

Table 4.1 – Nomenclature utilisée dans ce chapitre.

totalement à l'intérieur d'un autre. Ce type de site n'a pas de cellule d'Apollonius et peut donc être identifié avant les traitements. Ainsi, dans ce chapitre, nous fixons

$$||p_i - p_j|| > |r_i - r_j|. \tag{4.1}$$

Notre étude débute avec l'analyse proposée par des travaux précédents [Aur87 ; BWY06] (section 4.1) à partir de laquelle nous proposons une caractérisation et paramétrisation complète du diagramme d'Apollonius (section 4.2). Sur la base de cette analyse, nous dérivons une méthode de calcul de la géométrie du diagramme (section 4.3). Enfin, nous concluons ce chapitre (section 4.4). La nomenclature utilisée dans ce chapitre est détaillée tableau 4.1.

4.1 Apollonius dans $\mathbb{R}^d$ depuis $\mathbb{R}^{d+1}$

Comme présenté dans chapitre 2, des travaux précédents ont montré que le diagramme d'Apollonius peut être vu comme une projection depuis $\mathbb{R}^{d+1}$ sur $\mathbb{R}^d$ [Aur87 ; BWY06]. Puisque cette formulation est à la base de notre caractérisation, nous rappelons dans cette section ces développements.

Théorème 4.1.1. *Soient $s_i = (p_i, r_i)^T \in \mathbb{R}^d \times \mathbb{R}$ et $s_j = (p_j, r_j)^T \in \mathbb{R}^d \times \mathbb{R}$, deux sites pondérés, et les cônes Γ_i et Γ_j définis par*

$$\Gamma_k \coloneqq \left\{ \tilde{x} = (x, x_{d+1}) \in \mathbb{R}^{d+1} \mid x_{d+1} + r_k = ||x - p_k|| \right\}, \; k \in \{i, j\}. \tag{4.2}$$

Le bisecteur d'Apollonius H_{ij} entre s_i et s_j est donné par la projection sur $\mathbb{R}^d \times \{\emptyset\}$ de l'intersection $\tilde{H}_{ij} = \Gamma_i \cap \Gamma_j$. Soient $\tilde{s}_i = (s_i, \sqrt{2}r_i)^T \in \mathbb{R}^{d+1} \times \mathbb{R}$ et $\tilde{s}_j = (s_j, \sqrt{2}r_j)^T \in \mathbb{R}^{d+1} \times \mathbb{R}$, deux sites pondérés dans $\mathbb{R}^{d+1}$, et Π^2_{ij} leurs bisecteurs de Puissance. $\Gamma_i \cap \Gamma_j$ peut être alternativement caractérisé par l'intersection $\tilde{H}_{ij} = \Gamma_i \cap \Pi^2_{ij}$.

Démonstration. Nous commençons par étudier l'intersection des cônes Γ d'axe x_{x+1}, donnés par la fonction distance à s et de sommets $(p, -r)$. Les points $\tilde{x}$ positionnés sur $\tilde{H}_{ij} = \Gamma_i \cap \Gamma_j$ sont ainsi donnés par

$$x_{d+1} = ||x - p_i|| - r_i = ||x - p_j|| - r_j.$$

La projection de $\tilde{H}_{ij}$ sur $\mathbb{R}^d \times \{\emptyset\}$ satisfait l'égalité $\delta(s_i, x) = \delta(s_j, x)$. Elle est ainsi équivalente au bisecteur d'Apollonius H_{ij} entre s_i et s_j.

En étudiant les points $\tilde{x} \in \tilde{H}_{ij}$, nous remarquons qu'ils satisfont

$$||x - p_i||^2 - (x_{d+1} + r_i)^2 = ||x - p_j||^2 - (x_{d+1} + r_j)^2.$$

Cette égalité peut alors être reformulée en supprimant la dépendance quadratique à $\tilde{x}$ et donnant

$$x \cdot (p_j - p_i) + x_{d+1}(r_j - r_i) + \tfrac{1}{2}\left(||p_i||^2 - ||p_j||^2 - r_i^2 + r_j^2\right) = 0. \tag{4.3}$$

L'équation affine résultante caractérise alors un hyperplan Π^2_{ij} de dimension d d'intersection $\Gamma_i \cap \Gamma_j$ sur lequel repose $\tilde{H}_{ij}$. Ce dernier est alors aussi donné par $\tilde{H}_{ij} = \Gamma_i \cap \Pi^2_{ij}$. Π^2_{ij} correspond

enfin au bisecteur de Puissance entre les deux sites pondérés $\tilde{s}_i = (s_i, -r_i)$ et $\tilde{s}_j = (s_i, -r_i)$. Les points positionnés sur celui-ci satisfont $\delta^{(2)}(\tilde{s}_i, \tilde{x}) = \delta^{(2)}(\tilde{s}_j, \tilde{x})$. En résulte donc

$$||\tilde{x} - \tilde{s}_i||^2 - r_i^2 = ||\tilde{x} - \tilde{s}_j||^2 - r_j^2.$$

En supprimant encore une fois la dépendance quadratique à $\tilde{x}$, on obtient

$$x \cdot (p_j - p_i) + x_{d+1}(r_j - r_i) + \tfrac{1}{2}\left(||p_i||^2 + r_i^2 - \tilde{r_i}^2 - ||p_j||^2 - r_j^2 + \tilde{r_j}^2\right) = 0, \tag{4.4}$$

qui décrit aussi un hyperplan. En insérant $\tilde{r}_k = \sqrt{2}r_k$, on obtient l'égalité

$$\tfrac{1}{2}\left(||p_i||^2 - ||p_j||^2 - r_i^2 + r_j^2\right) = \tfrac{1}{2}\left(||p_i||^2 + r_i^2 - \tilde{r_i}^2 - ||p_j||^2 - r_j^2 + \tilde{r_j}^2\right),$$

permettant ainsi de prouver que (4.3) est égal à (4.4) et donc que Π_{ij}^2 est le bisecteur de Puissance entre $\tilde{s}_i$ et $\tilde{s}_j$ ainsi que le plan d'intersection des cônes Γ_i et Γ_j. □

Ce résultat, dû à Aurenhammer [Aur87], permet une caractérisation élégante de la cellule d'Apollonius d'un site pondéré s_i. Cette dernière peut en effet être directement construite à partir de la projection sur $\mathbb{R}^d \times \{\emptyset\}$ de l'intersection $\Gamma_i \cap \mathcal{P}(\tilde{s}_i)$ entre la cellule de Puissance de $\tilde{s}_i$ et le cône Γ_i. Nous montrons dans la prochaine section comment, sur la base de cette formulation, nous pouvons caractériser complètement le diagramme d'Apollonius.

4.2 Caractérisation du diagramme d'Apollonius

Sur la base de la formulation de la partie précédente, nous proposons dans cette section une caractérisation des faces des diagrammes d'Apollonius. Celle-ci est notamment nécessaire à une paramétrisation, mais aussi à l'étude des propriétés de ces diagrammes.

Cette section présente donc les caractérisations des faces de $d-k$-dimensions où $k \in \{1, \ldots, d\}$ et $d \in \{2, 3\}$. Ces développements sont, pour chacune des faces, suivis d'une paramétrisation permettant leur maillage. Ces analyses sont basées sur l'étude de Aurenhammer [Aur87], mais aussi sur l'extraction de la transformation des éléments du diagramme permettant ensuite une caractérisation analytique des équations en résultant.

4.2.1 Bisecteur

Un bisecteur d'Apollonius H_{ij} est défini par l'ensemble des points dont les sites les plus proches sont s_i et s_j

$$H_{ij} \coloneqq \left\{x \in \mathbb{R}^d \mid \delta(s_i, x) = \delta(s_j, x) \leq \delta(s_k, x),\ \forall s_k \in \mathcal{S} \setminus \{s_i, s_j\}\right\}.$$

Dans cette section, nous présentons la caractérisation de ces faces ainsi que leur paramétrisation.

Caractérisation

Théorème 4.2.1. *Soient deux sites pondérés $s_i = (p_i, r_i)^T \in \mathbb{R}^d \times \mathbb{R}$ et $s_j = (p_j, r_j)^T \in \mathbb{R}^d \times \mathbb{R}$ avec $r_j \geq r_i$. Si $r_i \neq r_j$, le bisecteur d'Apollonius H_{ij} est la nappe s'enveloppant autour de s_i d'un hyperboloïde de révolution à deux nappes, sinon, H_{ij} est un hyperplan de normale $p_i\vec{p}_j$.*

Démonstration. Soient $\tilde{x} = (x, x_{d+1})^T \in \tilde{H}_{ij} = \Gamma_i \cap \Pi^2_{ij}$, avec

$$\Pi^2_{ij} \coloneqq \left\{ \tilde{x} = (x, x_{d+1})^T \in \mathbb{R}^{d+1} \mid \tilde{x} \cdot \tilde{n} + c_{ij} = 0 \right\},$$

le plan d'intersection entre Γ_i et Γ_j où $\tilde{n} = s_j - s_i = (p_j - p_i, r_j - r_i)^T = (n, n_{d+1})^T$ et $c_{ij} = \frac{1}{2}\left(||p_i||^2 - r_i^2 - ||p_j||^2 + r_j^2\right)$. Nous commençons par étudier le cas où $r_i \neq r_j$, donc où $n_{d+1} \neq 0$ ainsi que $||n|| > |n_{d+1}|$ (4.1). Afin de définir l'intersection $\Gamma_i \cap \Pi^2_{ij}$, nous exprimons $\tilde{\Pi}^2_{ij}$ en fonction de x_{d+1}. Soit $x' = p_i - x$ et $x = p_i - x'$, nous obtenons

$$\begin{aligned} -x' \cdot n + x_{d+1} n_{d+1} + c_{ij} + p_i \cdot n = 0 \\ \Leftrightarrow x_{d+1} = \frac{x' \cdot n - w_{ij}}{n_{d+1}}, \text{ avec } w_{ij} = c_{ij} + p_i \cdot n. \end{aligned}$$

Nous pouvons alors directement utiliser ce résultat dans l'équation du cône de la fonction distance (4.2) donnant

$$\tilde{H}_{ij} \coloneqq \tilde{\Pi}^2_{ij} \cap \Gamma_i = \left\{ (x', x_{d+1}) \in \mathbb{R}^{d+1} \mid ||x'|| = x_{d+1} + r_i = \frac{x' \cdot n - w_{ij}}{n_{d+1}} + r_i \right\}.$$

Nous cherchons alors à caractériser $\tilde{H}_{ij}$ à partir de cette équation

$$\begin{aligned} ||x'|| &= \frac{x' \cdot n - w_{ij}}{n_{d+1}} + r_i \\ \Leftrightarrow ||x'|| n_{d+1} &= x' \cdot n - w_{ij} + r_i n_{d+1} \\ \Leftrightarrow ||x'||^2 n^2_{d+1} &= (x' \cdot n - w_{ij} + r_i n_{d+1})^2 \end{aligned}$$

Nous développons la partie droite de l'égalité afin d'en extraire la partie constante

$$\begin{aligned} &(x' \cdot n - w_{ij} + r_i n_{d+1})^2 \\ &= (n_{d+1} r_i - w_{ij})^2 + (x' \cdot n)^2 + 2(n_{d+1} r_i - w_{ij}) x' \cdot n \\ &= k_{ij}^2 + (x' \cdot n)^2 + 2 k_{ij} x' \cdot n, \text{ avec } k_{ij} = n_{d+1} r_i - w_{ij}. \end{aligned}$$

Nous pouvons alors exprimer les points appartenant à l'intersection $(x', x_{d+1})^T \in \Gamma_i \cap \tilde{\Pi}^2_{ij}$ avec

$$n^2_{d+1} ||x'||^2 - (x' \cdot n)^2 - 2 k_{ij} x' \cdot n = k_{ij}^2.$$

Cette dernière peut alors être exprimée sous forme matricielle

$$x'^T A x' - 2 k_{ij} x' \cdot n = k_{ij}^2, \tag{4.5}$$

avec $A = n_{d+1}^2 I - nn^T$, où I est la matrice identité.

Afin d'étudier l'équation analytique de l'expression de $\tilde{H}_{ij}$, nous extrayons les transformations linéaires de son expression. Soient $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_d) = \text{diag}(n_{d+1}^2 - ||n||^2, n_{d+1}^2, \ldots, n_{d+1}^2)$ et $V = (\frac{n}{||n||}, v_2, \ldots, v_d)$ avec $\{v_2, \ldots, v_d\}$ une base orthonormée de $n^\perp = \left\{x \in \mathbb{R}^d \mid x \cdot n = 0\right\}$. En nous basant sur (4.1), nous savons que $\lambda_1 < 0$ et $\lambda_i > 0, \forall i = \{2, \ldots, d\}$. Dans ce contexte, si $r_i \neq r_j$, nous obtenons

$$\begin{aligned} Av_1 &= n_{d+1}^2 \frac{n}{||n||} - \frac{nn \cdot n}{||n||} = (n_{d+1}^2 - ||n||^2)v_1 = \lambda_1 v_1 \\ Av_i &= n_{d+1}^2 v_i - nn \cdot v_i = n_{d+1}^2 v_i = \lambda_i v_i, \forall i \in \{2, \ldots, d\}. \end{aligned}$$

Ces développements montrent ainsi que $A = V\Lambda V^T$ est la décomposition de A en éléments propres avec V, la matrice des vecteurs propres, et Λ la matrice diagonale des valeurs propres. Cette décomposition permet ainsi d'isoler la transformation linéaire appliquée sur le bisecteur en effectuant un changement de base à partir des vecteurs propres. Soit $y = V^T x'$ et donc $x' = Vy$, nous pouvons alors exprimer (4.5) tel que

$$\begin{aligned} & y^T \Lambda y - 2k_{ij} Vy \cdot n = k_{ij}^2 \\ \Leftrightarrow\ & y^T \Lambda y - 2k_{ij} V^T n \cdot y = k_{ij}^2 \\ \Leftrightarrow\ & y^T \Lambda y - 2By = k_{ij}^2, \text{ avec } B = k_{ij} V^T n = (b_1, b_2, \ldots, b_d)^T. \end{aligned}$$

Puisque $v_i \cdot n = 0, \forall i = \{2, \ldots, d\}$, nous en déduisons que $b_i = 0$, tandis que $b_1 = k_{ij}||n||$. Nous pouvons alors simplifier l'expression

$$\begin{aligned} & y^T \Lambda y - 2b_1 y_1 = k_{ij}^2 \\ \Leftrightarrow\ & -|\lambda_1| y_1^2 + \sum_{i=2}^{d} \lambda_i y_i^2 - 2b_1 y_1 = k_{ij}^2 \\ \Leftrightarrow\ & -\left(\alpha_1^2 y_1^2 + 2\alpha_1 y_1 \frac{b_1}{\alpha_1} + \frac{b_1^2}{\alpha_1^2}\right) + \sum_{i=2}^{d} \lambda_i y_i^2 - 2b_1 y_1 = k_{ij}^2 - \frac{b_1^2}{\alpha_1^2}, \text{ avec } \alpha_1 = \sqrt{|\lambda_1|} \\ \Leftrightarrow\ & \left(\alpha_1 y_1 + \frac{b_1}{\alpha_1}\right)^2 - \sum_{i=2}^{d} \lambda_i y_i^2 = e, \text{ avec } e = \frac{b_1^2}{\alpha_1^2} - k_{ij}^2 = \frac{k_{ij}^2 n_{d+1}^2}{||n||^2 - n_{d+1}^2} \geq 0 \end{aligned}$$

En normalisant l'expression, nous obtenons finalement que l'intersection $\tilde{H}_{ij}$ est équivalente à

$$\frac{(\alpha_1 y_1 + \frac{b_1}{\alpha_1})^2}{e} - \frac{\sum_{i=2}^{d} y_i^2}{\frac{e}{n_{d+1}^2}} = 1,$$

qui décrit une hyperbole sur les deux coordonnées $\alpha_1 y_1 + \frac{b_1}{\alpha_1}$ et $\sqrt{\sum_{i=2}^{d} y_i^2}$. L'expression décrit ainsi dans $\mathbb{R}^3$ un hyperboloïde de révolution à deux nappes et une hyperbole dans $\mathbb{R}^2$. Puisque $x' = Vy$, l'axe selon x' est alors donné par la première colonne de V, $v_1 = \frac{n}{||n||}$.

En étudiant $r_i = r_j$, nous remarquons que $n_{d+1} = 0$ et que $c_{ij} = \frac{1}{2}(||p_i||^2 - ||p_j||^2)$. Puisque l'hyperplan $\tilde{\Pi}_{ij}^2$ est alors de normale $\tilde{n} = (n, n_{d+1})^T = (p_j - p_i, 0)^T$, il est perpendiculaire à

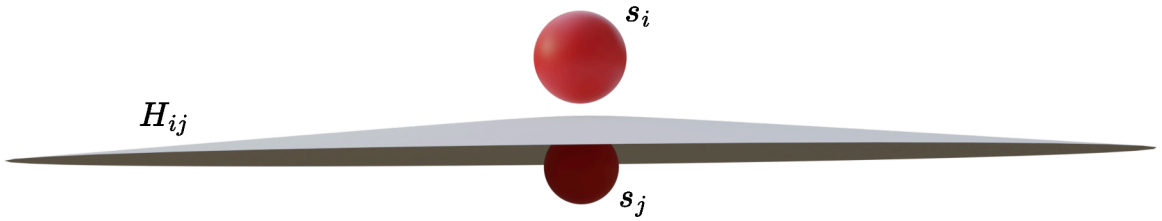


Figure 4.2 – Illustration d'un bisecteur d'Apollonius H_{ij} dans $\mathbb{R}^3$ entre les sites s_i et s_j. Ce dernier est caractérisé par la nappe d'un hyperboloïde de révolution à deux nappes s'enveloppant autour du site avec le plus petit rayon s_j.

$\mathbb{R}^d \times \{\emptyset\}$. Sa projection sur ce dernier donne alors un hyperplan de dimension $d-1$ et de normale n caractérisant le bisecteur d'Apollonius H_{ij}. □

Paramétrisation

La caractérisation proposée précédemment permet de connaitre la nature des bisecteurs d'Apollonius. Sur cette base, nous présentons une paramétrisation de ces derniers, illustrée figure 4.2.

Afin d'exprimer l'hyperbole obtenue (4.2.1), nous définissons $(\overline{y}_1(t), \overline{y}_2(t))$ comme sa paramétrisation. Ceux-ci doivent ainsi satisfaire $\overline{y}_1(t)^2 - \overline{y}_2(t)^2 = 1$ avec par exemple $\overline{y}_1(t) = \cosh(t)$ et $\overline{y}_2(t) = \sinh(t)$.

Dans $\mathbb{R}^2$ En définissant l'équivalence

$$\begin{aligned} \overline{y}_1(t) &= \frac{1}{\sqrt{e}}(\alpha_1 y_1 + \tfrac{b_1}{\alpha_1}) \\ \overline{y}_2(t) &= \frac{n_{d+1}}{\sqrt{e}} y_2, \end{aligned}$$

nous obtenons la paramétrisation

$$\begin{aligned} \tfrac{1}{\alpha_1}(\overline{y}_1(t)\sqrt{e} - \tfrac{b_1}{\alpha_1}) &= y_1 \\ \frac{\sqrt{e}}{n_{d+1}} \overline{y}_2(t) &= y_2. \end{aligned}$$

Ces résultats permettent de définir l'équation paramétrique

$$\overline{y}(t) = \left(\tfrac{1}{\alpha_1}(\overline{y}_1(t)\sqrt{e} - \tfrac{b_1}{\alpha_1}), \frac{\sqrt{e}}{n_{d+1}} \overline{y}_2(t) \right)^T.$$

Nous définissons enfin la paramétrisation complète $\overline{x}(t)$ de H_{ij} en transférant $\overline{y}(t)$ dans l'espace physique avec

$$\overline{x}(t) = p_i - V\overline{y}(t).$$

Dans $\mathbb{R}^3$ Avec une dimension additionnelle, les deux fonctions $\overline{y}_1(t)$ et $\overline{y}_2(t)$ ne suffisent plus à paramétrer H_{ij} puisque nous obtenons

$$\overline{y}_1(t) = \frac{1}{\sqrt{e}}(\alpha_1 y_1 + \tfrac{b_1}{\alpha_1})$$
$$\overline{y}_2(t) = \frac{n_{d+1}}{\sqrt{e}}\sqrt{y_2^2 + y_3^2}.$$

Nous pouvons cependant remarquer que

$$y_2^2 + y_3^2 = \frac{e}{n_{d+1}^2}\overline{y}_2(t)^2$$

décrit un cercle. Ce dernier peut alors être paramétré en utilisant $\overline{s}_1(\theta)$ et $\overline{s}_2(\theta)$ tels que $\overline{s}_1(\theta)^2 + \overline{s}_2(\theta)^2 = 1$. Nous pouvons, sur cette base, introduire la paramétrisation complète avec

$$\overline{y'}_1(t,\theta) = \tfrac{1}{\alpha_1}(\overline{y}_1(t)\sqrt{e} - \tfrac{b_1}{\alpha_1})$$
$$\overline{y'}_2(t,\theta) = \frac{\sqrt{e}}{n_{d+1}}\overline{y}_2(t)\,\overline{s}_1(\theta)$$
$$\overline{y'}_3(t,\theta) = \frac{\sqrt{e}}{n_{d+1}}\overline{y}_2(t)\,\overline{s}_2(\theta).$$

En définissant $\overline{y'}(t,\theta) = \left(\tfrac{1}{\alpha_1}(\overline{y}_1(t)\sqrt{e} - \tfrac{b_1}{\alpha_1}), \tfrac{\sqrt{e}}{n_{d+1}}\overline{y}_2(t)\,\overline{s}_1(\theta), \tfrac{\sqrt{e}}{n_{d+1}}\overline{y}_2(t)\,\overline{s}_2(\theta)\right)$, la paramétrisation finale $\overline{x}(t,\theta)$ dans l'espace physique peut être exprimée de la même façon que dans $\mathbb{R}^2$ avec

$$\overline{x}(t,\theta) = p_i - V\overline{y'}(t,\theta).$$

4.2.2 Trisecteur

Les trisecteurs d'Apollonius bornent les bisecteurs qui leur sont tangents. À la frontière de trois bisecteurs, ils sont définis par

$$H_{ijk} \coloneqq \left\{x \in \mathbb{R}^d \mid \delta(s_i,x) = \delta(s_j,x) = \delta(s_k,x) \leq \delta(s_l,x),\ \forall s_l \in \mathcal{S} \setminus \{s_i,s_j,s_k\}\right\}.$$

La caractérisation des bisecteurs comme de potentielles courbes implique que les trisecteurs puissent aussi être non linéaires. Puisqu'ils sont caractérisables comme l'intersection de plusieurs bisecteurs, ils présentent ainsi de nombreuses configurations possibles. Ces propriétés rendent leur étude difficile, alors même qu'ils sont à la base de certaines méthodes de constructions [KCK04 ; Wil99]. Notre étude s'inscrit ainsi en soutien de ce type de travaux et commence par la caractérisation des trisecteurs d'Apollonius puis présente leur paramétrisation.

Caractérisation

Théorème 4.2.2. *Soient $s_i = (p_i, r_i)^T \in \mathbb{R}^d \times \mathbb{R}$, $s_j = (p_j, r_j)^T \in \mathbb{R}^d \times \mathbb{R}$ et $s_k = (p_k, r_k)^T \in \mathbb{R}^d \times \mathbb{R}$. S'il existe, le trisecteur d'Apollonius H_{ijk} entre s_i, s_j et s_k dans $\mathbb{R}^2$ est composé de 0 à 2 points. Dans $\mathbb{R}^3$, il est défini par une section conique.*

Démonstration. Soit Π^2_{ijk}, l'hyperplan supportant $\tilde{H}_{ijk}$ et caractérisé par l'intersection $\Pi^2_{ij} \cap \Pi^2_{ik}$ des deux hyperplans portant $\tilde{H}_{ij}$ et $\tilde{H}_{ik}$ définis par

$$\Pi^2_{ij} \coloneqq \{\tilde{x} \mid \tilde{x} \cdot \tilde{n}_{ij} + c_{ij} = 0\}$$
$$\Pi^2_{ik} \coloneqq \{\tilde{x} \mid \tilde{x} \cdot \tilde{n}_{ik} + c_{ik} = 0\},$$

où $n_{il} = s_l - s_i = (p_l - p_i, r_l - r_i)^T$ et $c_{il} \coloneqq \frac{1}{2}(||p_i||^2 - r_i^2 - ||p_l||^2 + r_l^2)$, $l \in \{j, k\}$.

Avant de caractériser H_{ijk}, nous exprimons Π^2_{ijk} comme l'intersection $\Pi^2_{ij} \cap \Pi^2_{ik}$. Celle-ci est caractérisée par tous les points $\tilde{x} = (\hat{x}, x_d, x_{d+1})^T$ satisfaisant

$$\begin{cases} \hat{x} \cdot \hat{n}_{ij} + x_d n_{d,ij} + x_{d+1} n_{d+1,ij} + c_{ij} & = 0 \\ \hat{x} \cdot \hat{n}_{ik} + x_d n_{d,ik} + x_{d+1} n_{d+1,ik} + c_{ik} & = 0, \end{cases}$$

avec $\tilde{n} = (\hat{n}, n_d, n_{d+1})^T$. L'intersection peut alors être exprimée sous forme matricielle

$$A \begin{pmatrix} x_d \\ x_{d+1} \end{pmatrix} = - \begin{pmatrix} \hat{n}_{ij}^T \\ \hat{n}_{ik}^T \end{pmatrix} \hat{x} - \begin{pmatrix} c_{ij} \\ c_{ik} \end{pmatrix}, \text{ avec } A = \begin{pmatrix} n_{d,ij} & n_{d+1,ij} \\ n_{d,ik} & n_{d+1,ik} \end{pmatrix}.$$

Si A est singulière, nous pouvons utiliser une autre combinaison de coordonnées $a, b \in \{1, \ldots, d+1\}$, $a \neq b$. Nous pouvons de plus remarquer que A n'est singulière avec n'importe quelles combinaisons a, b que si et seulement si $\tilde{n}_{ij}$ et $\tilde{n}_{ik}$ sont parallèles, et donc si le trisecteur H_{ijk} n'existe pas. Dans l'éventualité où A est inversible, nous obtenons

$$\begin{aligned} & \begin{pmatrix} x_d \\ x_{d+1} \end{pmatrix} = -A^{-1} \begin{pmatrix} \hat{n}_{ij}^T \\ \hat{n}_{ik}^T \end{pmatrix} \hat{x} - A^{-1} \begin{pmatrix} c_{ij} \\ c_{ik} \end{pmatrix} \\ \Leftrightarrow & \begin{pmatrix} x_d \\ x_{d+1} \end{pmatrix} = N\hat{x} - A^{-1} \begin{pmatrix} c_{ij} \\ c_{ik} \end{pmatrix}, \text{ avec } N = \begin{pmatrix} N_{1\bullet} \\ N_{2\bullet} \end{pmatrix} = -A^{-1} \begin{pmatrix} \hat{n}_{ij}^T \\ \hat{n}_{ik}^T \end{pmatrix} \in \mathbb{R}^{2\times(d-1)} \\ \Leftrightarrow & \begin{pmatrix} x_d \\ x_{d+1} \end{pmatrix} = N\hat{x} + b, \text{ avec } b = -A^{-1} \begin{pmatrix} c_{ij} \\ c_{ik} \end{pmatrix} \in \mathbb{R}^2, \end{aligned}$$

permettant d'exprimer Π^2_{ijk} en fonction des paramètres x_d et x_{d+1}. Puisque Π^2_{ij} et Π^2_{ik} portent tous deux les intersections $\Gamma_i \cap \Gamma_j$ et $\Gamma_i \cap \Gamma_k$ (section 4.1), les points $\tilde{x} = (x, x_{d+1})^T$ appartenant à l'intersection $\Pi^2_{ijk} \cap \Gamma_i$ respectent

$$\delta(s_i, x) = \delta(s_j, x) = \delta(s_k, x) \leq \delta(s_l, x),\ s_l \in \mathcal{S} \setminus \{s_i, s_j, s_k\}.$$

La projection de $\tilde{H}_{ijk} = \Pi^2_{ijk} \cap \Gamma_i$ sur $\mathbb{R}^d \times \{\emptyset\}$ est donc H_{ijk}. Nous exprimons maintenant $\tilde{H}_{ijk}$ à partir de l'intersection $\Pi^2_{ijk} \cap \Gamma_i$. Tout point $\tilde{x} = (\hat{x}, x_d, x_{d+1})^T \in \Pi^2_{ijk} \cap \Gamma_i$ satisfait

$$\begin{cases} x_d = N_{1\bullet}\hat{x} + b_1 \\ x_{d+1} = N_{2\bullet}\hat{x} + b_2 \\ (x_{d+1} + r_i)^2 = ||x - p_i||^2. \end{cases} \tag{4.6}$$

Afin de combiner les expressions, nous effectuons un changement de coordonnées. Soit $\tilde{x'} = (\hat{x'}, x'_d, x'_{d+1})^T = (x - p_i, x_{d+1} + r_i)^T$, nous obtenons

$$\begin{cases} x'_d = N_{1\bullet}\hat{x} + b_1 - p_{d,i} = N_{1\bullet}(\hat{x'} + \hat{p}_i) + b_1 - p_{d,i} = N_{1\bullet}\hat{x'} + e_1 \\ x'_{d+1} = N_{2\bullet}\hat{x'} + b_2 + r_i = N_{2\bullet}(\hat{x'} + \hat{p}_i) + b_2 + r_i = N_{2\bullet}\hat{x'} + e_2 \\ x'^2_{d+1} = \sum_{\alpha=1}^{d-1} {x'_\alpha}^2 + x'^2_d, \end{cases} \tag{4.7}$$

avec $e_1 = N_{1\bullet}\hat{p}_i + b_1 - p_{d,i}$ et $e_2 = N_{2\bullet}\hat{p}_i + b_2 + r_i$. À partir de ce changement de coordonnées, nous pouvons combiner ces contraintes en une seule équation

$$\begin{aligned} & ||\hat{x'}||^2 + (N_{1\bullet}\hat{x'} + e_1)^2 = (N_{2\bullet}\hat{x'} + e_2)^2 \\ \Leftrightarrow\ & ||\hat{x'}||^2 + (N_{1\bullet}\hat{x'})^2 + e_1^2 + 2N_{1\bullet}\hat{x'}e_1 = (N_{2\bullet}\hat{x'})^2 + e_2^2 + 2N_{2\bullet}\hat{x'}e_2 \\ \Leftrightarrow\ & ||\hat{x'}||^2 + (N_{1\bullet}\hat{x'})^2 + 2N_{1\bullet}\hat{x'}e_1 - 2N_{2\bullet}\hat{x'}e_2 - (N_{2\bullet}\hat{x'})^2 = e_2^2 - e_1^2 \\ \Leftrightarrow\ & ||\hat{x'}||^2 + (N_{1\bullet}\hat{x'})^2 - (N_{2\bullet}\hat{x'})^2 + 2(N_{1\bullet}e_1 - N_{2\bullet}e_2)\hat{x'} = e_2^2 - e_1^2 \\ \Leftrightarrow\ & \hat{x'}^T\hat{x'} + \hat{x'}^T N_{1\bullet}^T N_{1\bullet}\hat{x'} - \hat{x'}^T N_{2\bullet}^T N_{2\bullet}\hat{x'} + 2(N_{1\bullet}e_1 - N_{2\bullet}e_2)\hat{x'} = e_2^2 - e_1^2 \\ \Leftrightarrow\ & \hat{x'}^T(I + N_{1\bullet}^T N_{1\bullet} - N_{2\bullet}^T N_{2\bullet})\hat{x'} + 2(N_{1\bullet}e_1 - N_{2\bullet}e_2)\hat{x'} = e_2^2 - e_1^2 \\ \Leftrightarrow\ & \hat{x'}^T E\hat{x'} + 2M \cdot \hat{x'} = e_2^2 - e_1^2, \end{aligned}$$

avec $E = I + N_{1\bullet}^T N_{1\bullet} - N_{2\bullet}^T N_{2\bullet} \in \mathbb{R}^{(d-1)\times(d-1)}$ et $M = N_{1\bullet}e_1 - N_{2\bullet}e_2 \in \mathbb{R}^{d-1}$.

Dans $\mathbb{R}^2$ $E, M \in \mathbb{R}$. Ainsi, l'expression du second degré

$$E{\hat{x'}}^2 + 2M\hat{x'} - e_2^2 + e_1^2 = 0$$

peut être résolue classiquement. Le trisecteur H_{ijk} est alors composé d'un ou deux points, s'il existe.

Dans $\mathbb{R}^3$ $\hat{x'} \in \mathbb{R}^2$ ce qui implique que l'intersection $\Pi^2_{ijk} \cap \Gamma_i$ est une courbe de deux dimensions. Soit $E = V\Lambda V^T$ avec $\Lambda = \text{diag}(\lambda_1, \lambda_2), \lambda_1 \leq \lambda_2$ la décomposition de E en éléments propres. D'une façon similaire aux bisecteurs, nous extrayons la transformation appliquée à la courbe en changeant de coordonnées pour les vecteurs propres avec $\hat{z} = V^T{\hat{x'}}^T = (z_1, z_2)^T, \hat{x'} = Vz$ permettant d'obtenir

$$\lambda_1 z_1^2 + \lambda_2 z_2^2 + 2\hat{z} \cdot \beta = e,$$

avec $\beta = V^T M$ et $e = e_2^2 - e_1^2$. L'expression obtenue est quadratique, et décrit donc une section conique. Puisque cette dernière peut donner lieu à différents types de coniques, nous étudions les cas possibles :

— Tout d'abord, si $\lambda_1 = \lambda_2 = 0$, l'expression devient linéaire puisqu'elle ne contient plus de termes quadratiques

$$2\hat{z} \cdot \beta = e$$

et le trisecteur prend alors la forme d'une droite.

— Si $0 = \lambda_1 < \lambda_2$ ou $\lambda_1 < \lambda_2 = 0$, il ne reste plus qu'un terme quadratique à l'expression. Afin de réécrire l'expression indépendamment des cas, nous introduisons plusieurs variables supplémentaires

$$\begin{cases} z'_1 = z_2, z'_2 = z_1, \beta'_1 = \beta_2, \beta'_2 = \beta_1, \lambda' = \lambda_2, e' = e & \text{si } \lambda_1 = 0 \\ z'_1 = z_1, z'_2 = z_2, \beta'_1 = -\beta_1, \beta'_2 = -\beta_2, \lambda' = -\lambda_1, e' = -e & \text{si } \lambda_2 = 0. \end{cases}$$

Nous pouvons alors reformuler l'expression avec

$$\lambda'_1 z'^2_1 + 2\beta'_1 z'_1 + 2\beta'_2 z'_2 = e'.$$

Nous isolons ensuite le terme quadratique z'_1 à partir de

$$\begin{aligned} \lambda' z'^2_1 + 2\beta'_1 z'_1 + 2\beta'_2 z'_2 + \frac{\beta'^2_1}{\lambda'} &= e' + \frac{\beta'^2_1}{\lambda'} \\ \Leftrightarrow \left(\sqrt{\lambda'} z'_1 + \frac{\beta'_1}{\sqrt{\lambda'}}\right)^2 + 2\beta'_2 z'_2 &= e' + \frac{\beta'^2_1}{\lambda'} \\ \Leftrightarrow \left(\sqrt{\lambda'} z'_1 + \frac{\beta'_1}{\sqrt{\lambda'}}\right)^2 &= e' + \frac{\beta'^2_1}{\lambda'} - 2\beta'_2 z'_2 \end{aligned}$$

décrivant ainsi une parabole de la forme $u^2 = 4v$ avec $u = \sqrt{\lambda'} z'_1 + \frac{\beta'_1}{\sqrt{\lambda'}}$ et $v = \frac{1}{4}(e' + \frac{\beta'^2_1}{\lambda'} - 2\beta'_2 z'_2)$.

— Enfin, l'expression peut contenir deux termes quadratiques avec $0 < \lambda_1 \leq \lambda_2$, $\lambda_1 < 0 < \lambda_2$ ou $\lambda_1 \leq \lambda_2 < 0$. Une nouvelle fois, nous introduisons des variables permettant de traiter tous les cas d'une même façon avec $\mu_1 = \text{sign}(\lambda_1)$, $\mu_2 = \text{sign}(\lambda_2)$, $\lambda'_1 = |\lambda_1|$ et $\lambda'_2 = |\lambda_2|$. Nous obtenons alors

$$\mu_1 \lambda'_1 z_1^2 + \mu_2 \lambda'_2 z_2^2 + 2\beta_1 z_1 + 2\beta_2 z_2 = e.$$

Nous rassemblons les termes quadratiques comme pour le cas précédent

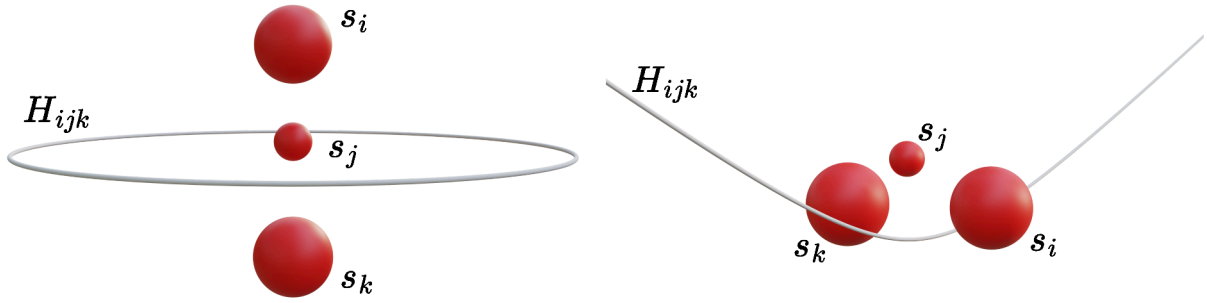
$$\mu_1 \lambda'_1 z_1^2 + \mu_2 \lambda'_2 z_2^2 + 2\beta_1 z_1 + 2\beta_2 z_2 + \frac{\beta_1^2}{\lambda_1} + \frac{\beta_2^2}{\lambda_2} = e', \text{ avec } e' = e + \frac{\beta_1^2}{\lambda_1} + \frac{\beta_2^2}{\lambda_2}$$

$$\mu_1 \left(\sqrt{\lambda'_1} z_1 + \mu_1 \frac{\beta_1}{\sqrt{\lambda'_1}}\right)^2 + \mu_2 \left(\sqrt{\lambda'_2} z_2 + \mu_2 \frac{\beta_2}{\sqrt{\lambda'_2}}\right)^2 = e'.$$

Cette expression peut enfin être classifiée en introduisant $z'_\alpha = \lambda'_\alpha z_\alpha + \mu_\alpha \beta_\alpha$, permettant ainsi d'obtenir

$$\frac{\mu_1}{e'} \frac{z'^2_1}{\lambda'_1} + \frac{\mu_2}{e'} \frac{z'^2_2}{\lambda'_2} = 1 \Leftrightarrow \frac{z'^2_1}{e'\lambda_1} + \frac{z'^2_2}{e'\lambda_2} = 1$$

Le type de l'expression est ainsi décrit par les signes de $e'\lambda_1$ et $e'\lambda_2$. Le trisecteur H_{ijk} est une hyperbole si $e'\lambda_1 < 0 < e'\lambda_2$ ou $e'\lambda_2 < 0 < e'\lambda_1$, une ellipse si $0 < e'\lambda_1$, $0 < e'\lambda_2$ avec $\lambda_1 \neq \lambda_2$, et un cercle si $0 < e'\lambda_1 = e'\lambda_2$. Enfin, si $e'\lambda_1 < 0$ et $e'\lambda_2 < 0$, le trisecteur H_{ijk} n'existe pas.



(a) Trisecteur circulaire **(b)** Trisecteur hyperbolique

Figure 4.3 – Illustration de deux configurations d'un trisecteur d'Apollonius H_{ijk} entre les sites s_i, s_j et s_k dans $\mathbb{R}^3$. Elles sont toutes les deux caractérisées par une section conique.

□

Paramétrisation

Comme pour les bisecteurs, nous fournissons maintenant la paramétrisation des trisecteurs d'Apollonius. Puisque ces développements sont similaires pour tous les types de trisecteurs, nous les présentons pour les trisecteurs hyperboliques et elliptiques qui représentent les cas les plus généraux. En effet, les autres types de trisecteurs sont des cas particuliers qui peuvent être évités en perturbant l'ensemble d'entrée.

Nous introduisons les fonctions $\overline{z}_1(t)$ et $\overline{z}_2(t)$ supportant les trisecteurs elliptiques (si $0 < e'\lambda_1 < e'\lambda_2$) et hyperboliques (si $e'\lambda_1 < 0 < e'\lambda_2$) grâce à

$$\begin{cases} -\overline{z}_1(t)^2 + \overline{z}_2(t)^2 = 1, & \text{si } e'\lambda_1 < 0 < e'\lambda_2 \\ \overline{z}_1(t)^2 + \overline{z}_2(t)^2 = 1, & \text{si } 0 < e'\lambda_1 < e'\lambda_2. \end{cases}$$

Il reste ensuite à appliquer les développements obtenus lors de la caractérisation

$$\begin{cases} \overline{z}_1(t)^2 = \frac{z_1'^2}{e'\lambda_1} \\ \overline{z}_2(t)^2 = \frac{z_2'^2}{e'\lambda_2} \end{cases} \Leftrightarrow \begin{cases} \overline{z}_1(t) = \frac{1}{\sqrt{|e'\lambda_1|}}(\lambda_1' z_1 + \mu_1\beta_1) \\ \overline{z}_2(t) = \frac{1}{\sqrt{|e'\lambda_2|}}(\lambda_2' z_2 + \mu_2\beta_2) \end{cases} \Leftrightarrow \begin{cases} \frac{1}{\lambda_1'}(\overline{z}_1(t)\sqrt{|e'\lambda_1|} - \mu_1\beta_1) = z_1 \\ \frac{1}{\lambda_2'}(\overline{z}_2(t)\sqrt{|e'\lambda_2|} - \mu_2\beta_2) = z_2. \end{cases}$$

Nous exprimons ensuite la paramétrisation dans l'espace physique avec la matrice des vecteurs propres V

$$\overline{x'}(t) = V \begin{pmatrix} \frac{1}{\lambda_1'}(\overline{z}_1(t)\sqrt{|e'\lambda_1|} - \mu_1\beta_1) \\ \frac{1}{\lambda_2'}(\overline{z}_2(t)\sqrt{|e'\lambda_2|} - \mu_2\beta_2) \end{pmatrix}.$$

Afin d'obtenir les coordonnées finales, il reste enfin à ajouter la coordonnée manquante et

à retirer la translation avec

$$\overline{x}(t) = p_i + \begin{pmatrix} \overline{x'}(t) \\ N_{1\bullet}\overline{x'}(t) + e_1 \end{pmatrix}.$$

Des exemples de cette paramétrisation sont illustrés figure 4.3.

Ces résultats illustrent la complexité des diagrammes d'Apollonius et la nécessité de la prise en compte des cas possibles afin de proposer des algorithmes de construction exhaustifs.

4.2.3 Quadrisecteur

Les quadrisecteurs d'Apollonius sont la cible de plusieurs méthodes de construction puisque, en tant que faces de plus petite dimension dans $\mathbb{R}^3$, ils permettent de dériver la structure combinatoire des trisecteurs et bisecteurs qui leur sont liés. Ils sont ainsi caractérisés par

$$H_{ijkl} \coloneqq \{x \mid \delta(s_i, x) = \delta(s_j, x) = \delta(s_k, x) = \delta(s_l, x) \leq \delta(s_m, x),\ \forall s_m \in \mathcal{S} \setminus \{s_i, s_j, s_k, s_l\}\}.$$

Même si les quadrisecteurs d'Apollonius sont étudiés dans de précédents travaux [GR03 ; Wan+20], nous présentons leurs caractérisations pour des raisons d'exhaustivité.

Théorème 4.2.3. *Soient $s_i = (p_i, r_i)^T \in \mathbb{R}^d \times \mathbb{R}$, $s_j = (p_j, r_j)^T \in \mathbb{R}^d \times \mathbb{R}$, $s_k = (p_k, r_k)^T \in \mathbb{R}^d \times \mathbb{R}$ et $s_l = (p_l, r_l)^T \in \mathbb{R}^d \times \mathbb{R}$. Dans $\mathbb{R}^3$, le quadrisecteur d'Apollonius H_{ijkl} est composé de 1 ou 2 points, s'il existe.*

Démonstration. Soit Π^2_{ijkl}, l'hyperplan sur lequel est positionné l'intersection $\tilde{H}_{ijkl} = \Gamma_i \cap \Pi^2_{ijkl}$. Π^2_{ijkl} est ainsi caractérisé par l'intersection $\Pi^2_{ij} \cap \Pi^2_{ik} \cap \Pi^2_{il}$ donnée par

$$\begin{aligned}
\Pi^2_{ij} &\coloneqq \left\{\tilde{x} \in \mathbb{R}^{d+1} \mid \tilde{x} \cdot \tilde{n}_{ij} + c_{ij} = 0\right\} \\
\Pi^2_{ik} &\coloneqq \left\{\tilde{x} \in \mathbb{R}^{d+1} \mid \tilde{x} \cdot \tilde{n}_{ik} + c_{ik} = 0\right\} \\
\Pi^2_{il} &\coloneqq \left\{\tilde{x} \in \mathbb{R}^{d+1} \mid \tilde{x} \cdot \tilde{n}_{il} + c_{il} = 0\right\},
\end{aligned}$$

avec $\tilde{x} = (\hat{x}, x_{d-1}, x_d, x_{d+1})^T$ et $\tilde{n}_{im} = (\hat{n}_m, n_{d-1,m}, n_{d,m}, n_{d+1,m})^T, m \in \{j, k, l\}$. Les points $\tilde{x} \in \Pi^2_{ijkl}$ localisés sur l'hyperplan d'intersection Π^2_{ijkl} doivent donc satisfaire

$$\begin{cases}
\hat{x} \cdot \hat{n}_{ij} + x_{d-1} n_{d-1,ij} + x_d n_{d,ij} + x_{d+1} n_{d+1,ij} + c_{ij} &= 0 \\
\hat{x} \cdot \hat{n}_{ik} + x_{d-1} n_{d-1,ik} + x_d n_{d,ik} + x_{d+1} n_{d+1,ik} + c_{ik} &= 0 \\
\hat{x} \cdot \hat{n}_{il} + x_{d-1} n_{d-1,il} + x_d n_{d,il} + x_{d+1} n_{d+1,il} + c_{il} &= 0.
\end{cases}$$

Nous exprimons ensuite ces contraintes sous formes matricielles

$$A \begin{pmatrix} x_{d-1} \\ x_d \\ x_{d+1} \end{pmatrix} = - \begin{pmatrix} \hat{n}_{ij}^T \\ \hat{n}_{ik}^T \\ \hat{n}_{il}^T \end{pmatrix} \hat{x} - \begin{pmatrix} c_{ij} \\ c_{ik} \\ c_{il} \end{pmatrix}, \text{ avec } A = \begin{pmatrix} n_{ij,d-1} & n_{ij,d} & n_{ij,d+1} \\ n_{ik,d-1} & n_{ik,d} & n_{ik,d+1} \\ n_{il,d-1} & n_{il,d} & n_{il,d+1} \end{pmatrix}.$$

Si A n'est pas inversible, nous sélectionnons un autre triplet de coordonnées $a, b, c \in \{1, \dots, d+1\}, a \neq b \neq c$. Si, pour tout triplet (a, b, c), A est singulière, cela veut dire qu'au moins deux des hyperplans Π^2_{ij}, Π^2_{ik} et Π^2_{il} sont parallèles, et donc que Π^2_{ijkl} n'existe pas. En supposant maintenant que A est inversible, nous obtenons

$$\begin{pmatrix} x_{d-1} \\ x_d \\ x_{d+1} \end{pmatrix} = -A^{-1} \begin{pmatrix} \hat{n}_{ij}^T \\ \hat{n}_{ik}^T \\ \hat{n}_{il}^T \end{pmatrix} \hat{x} - A^{-1} \begin{pmatrix} c_{ij} \\ c_{ik} \\ c_{il} \end{pmatrix}$$

$$\Leftrightarrow \begin{pmatrix} x_{d-1} \\ x_d \\ x_{d+1} \end{pmatrix} = N\hat{x} + b, \text{ avec } N = \begin{pmatrix} N_{1\bullet} \\ N_{2\bullet} \\ N_{3\bullet} \end{pmatrix} = -A^{-1} \begin{pmatrix} \hat{n}_{ij}^T \\ \hat{n}_{ik}^T \\ \hat{n}_{il}^T \end{pmatrix} \text{ et } b = -A^{-1} \begin{pmatrix} c_{ij} \\ c_{ik} \\ c_{il} \end{pmatrix}$$

qui permet de paramétriser l'intersection Π^2_{ijkl} à travers les paramètres x_{d-1}, x_d et x_{d+1}. Tout comme pour les trisecteurs, en tant que plan porteur de l'intersection des cônes Γ_i, Γ_j, Γ_k, et Γ_l, les d premières coordonnées de l'intersection $\tilde{H}_{ijkl} = \Gamma_i \cap \Pi^2_{ijkl}$ satisfont

$$\delta(s_i) = \delta(s_j) = \delta(s_k) = \delta(s_l) \leq \delta(s_m), \forall s_m \in \mathcal{S} \setminus \{s_i, s_j, s_k, s_l\},$$

ce qui prouve que la projection de $\tilde{H}_{ijkl}$ sur $\mathbb{R}^d \times \{\emptyset\}$ est H_{ijkl}. À partir de cette paramétrisation, nous exprimons $\tilde{H}_{ijkl} = \Gamma_i \cap \Pi^2_{ijkl}$ avec l'équation du cône Γ_i

$$\Gamma_i \coloneqq \left\{ \tilde{x} = (\hat{x}, x_{d-1}, x_d, x_{d+1})^T \mid ||x - p_i|| = x_{d+1} + r_i \right\}.$$

Afin de combiner les expressions, nous reformulons la paramétrisation de $\tilde{H}_{ijkl}$ en fonction de p_i avec $x' = (\hat{x'}, x'_{d-1}, x'_d, x'_{d+1})^T = (\hat{x} - \hat{p}_i, x_{d-1} - p_{d-1,i}, x_d - p_{d,i}, x_{d+1} + r_i)^T$. Le système devient alors

$$\begin{cases} x'_{d-1} = N_{1\bullet} \cdot \hat{x} + b_1 - p_{d-1,i} = N_{1\bullet} \cdot (\hat{x'} + \hat{p}_i) + b_1 - p_{d-1,i} = N_{1\bullet}\hat{x'} + e_1 \\ x'_d = N_{2\bullet} \cdot \hat{x} + b_2 - p_{d,i} = N_{2\bullet} \cdot (\hat{x'} + \hat{p}_i) + b_2 - p_{d,i} = N_{1\bullet}\hat{x'} + e_2 \\ x'_{d+1} = N_{3\bullet} \cdot \hat{x} + b_3 + r_i = N_{3\bullet} \cdot (\hat{x'} + \hat{p}_i) + b_3 + r_i = N_{1\bullet}\hat{x'} + e_3, \end{cases}$$

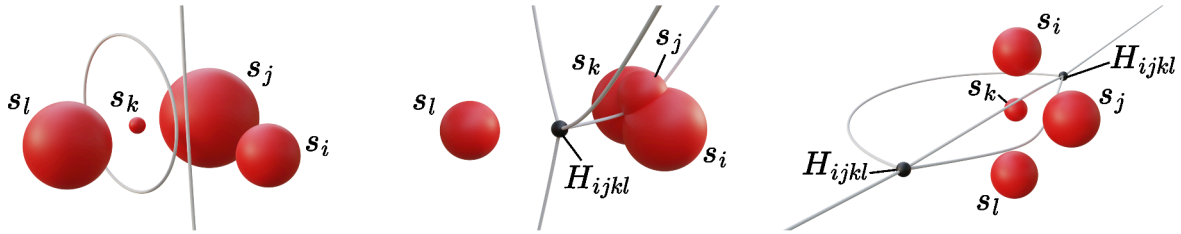
$$\Leftrightarrow \begin{pmatrix} x'_{d-1} \\ x'_d \\ x'_{d+1} \end{pmatrix} = N\hat{x'} + e, \text{ avec } e = \begin{pmatrix} e_1 \\ e_2 \\ e_3 \end{pmatrix} = \begin{pmatrix} N_{1\bullet} \cdot \hat{p}_i + b_1 - p_{d-1,i} \\ N_{2\bullet} \cdot \hat{p}_i + b_2 - p_{d,i} \\ N_{3\bullet} \cdot \hat{p}_i + b_3 + r_i \end{pmatrix}.$$

Nous pouvons maintenant combiner cette dernière expression avec celle décrivant le cône Γ_i

$$||\hat{x'}||^2 + x'^2_{d-1} + x'^2_d = x'^2_{d+1}.$$

Les points $\tilde{x} \in \tilde{H}_{ijkl}$ satisfont donc

$$\begin{aligned} ||\hat{x'}||^2 + (N_{1\bullet}\hat{x'} + e_1)^2 + (N_{2\bullet}\hat{x'} + e_2)^2 &= (N_{3\bullet}\hat{x'} + e_3)^2 \\ \Leftrightarrow\ \hat{x'}^T E\hat{x'} + 2M\hat{x'} &= e_3^2 - e_1^2 - e_2^2, \end{aligned}$$



(a) Pas de solutions **(b)** Une solution **(c)** Deux solutions

Figure 4.4 – Illustrations de trois différentes configurations d'un quadrisecteur d'Apollonius H_{ijkl} entre les sites s_i, s_j, s_k et s_l dans $\mathbb{R}^3$.

avec $E = I + N_{1\bullet}^T N_{1\bullet} + N_{2\bullet}^T N_{2\bullet} - N_{3\bullet}^T N_{3\bullet} \in \mathbb{R}^{(d-2)\times(d-2)}$ et $M = N_{1\bullet} e_1 + N_{2\bullet} e_2 - N_{3\bullet} e_3 \in \mathbb{R}^{(d-2)}$. Le quadrisecteur H_{ijkl} n'existe donc pas dans $\mathbb{R}^2$. Dans $\mathbb{R}^3$, nous obtenons alors une expression du second degré caractérisant H_{ijkl}

$$E\hat{x'}^2 + 2M\hat{x'} + e_1^2 + e_2^2 - e_3^2 = 0,$$

résolvable classiquement et donnant une ou deux solutions si le quadrisecteur H_{ijkl} existe (figure 4.4). □

Les développements présentés caractérisent complètement les diagrammes d'Apollonius dans $\mathbb{R}^2$ et $\mathbb{R}^3$. Ainsi, ils permettent l'étude analytique de leurs propriétés, mais aussi leur paramétrisation. Cette dernière est cependant locale et ne prend pas en compte les potentielles bordures dues à des sites voisins, nécessaire au diagramme complet. C'est donc sur cette base que nous présentons dans la section suivante une méthode de calcul de la géométrie d'un diagramme d'Apollonius.

4.3 Calcul de la géométrie des diagrammes d'Apollonius

Les développements analytiques de la partie précédente permettent une caractérisation complète des éléments d'Apollonius dans $\mathbb{R}^2$ et $\mathbb{R}^3$. Celles-ci sont cependant locales et ne prennent pas en compte les faces voisines pouvant potentiellement les borner, nécessaires au calcul de la géométrie du diagramme. Nous visons une stratégie permettant une implémentation simple et rapide supportant les particularités du diagramme. Dans cette section, nous présentons donc une méthode de calcul de la géométrie du diagramme à partir de notre paramétrisation.

Comme remarqué par des travaux précédents [KCK05], les trisecteurs peuvent être utilisés comme la limite des faces voisines. Étant donné qu'un trisecteur d'Apollonius H_{ijk} est porté par un plan Π_{ij}^2 dans $\mathbb{R}^3$, il décrit ainsi les limites de H_{ij}, H_{ik} et H_{jk}. Ces plans peuvent alors être utilisés afin de borner les paramétrisations proposées.

Soit un bisecteur d'Apollonius H_{ij}, entre s_i et s_j, et sa géométrie locale non bornée H_{ij}^*, obtenue à partir de la paramétrisation. La géométrie globale de H_{ij} bornée par des faces voisines peut ainsi être obtenue à partir de

$$H_{ij} := H_{ij}^* \bigcap_{s_m \in \mathcal{S}\setminus\{s_i,s_j\}} \Pi_{ijm}^{2+},$$

avec Π^{2+}_{ijm} le demi-espace contenant H_{ij} et décrit par le plan d'intersection Π^{2}_{ijm} entre les cônes Γ_i, Γ_j et Γ_k. Il est cependant nécessaire de prendre en compte le cas où l'intersection Π^{2}_{ijm} n'existe pas. Cette éventualité n'est possible que si et seulement si H_{ij} est totalement localisé dans H^{+}_{im} ou H^{-}_{im}. Tandis que le premier cas laisse la géométrie inchangée, si $H_{ij} \cap H^{+}_{im} = \emptyset$, le bisecteur H_{ij} n'existe alors pas. Afin de différencier ces configurations, nous testons l'appartenance d'un point de la géométrie $x_{ij} \in H^{*}_{ij}$ au demi-espace défini par l'autre bisecteur $x_{ij} \overset{?}{\in} \Pi^{2+}_{ik}$.

Cette stratégie est particulièrement adaptée puisqu'elle est aussi applicable aux trisecteurs et quadrisecteurs. Ainsi, nous dérivons un algorithme de construction itératif du diagramme donné algorithme 3. Celui-ci nécessite uniquement l'implémentation de notre paramétrisation avec la fonction *parametrize* et de la limitation des faces à partir des demi-espaces avec *intersect*. Cette dernière fonction peut être implémentée à partir du plan Π^{2}_{ijk} et en utilisant un algorithme d'intersection entre celui-ci et un maillage, couramment disponible dans les bibliothèques de traitement géométrique [The23]. Un exemple de diagramme est illustré figure 4.5. Enfin, même si les développements présentés dans cette section visent des maillages obtenus à partir de la paramétrisation, cette méthode est aussi utilisable avec une stratégie de lancer de rayon. Ainsi, il suffit d'appliquer les transformations des caractérisations à l'intersection entre un rayon et les formes souhaitées, comme des hyperboloïdes à deux nappes pour les bisecteurs.

Algorithme 3 : Construction itérative de la géométrie d'un diagramme d'Apollonius dans $\mathbb{R}^3$

Input : Un ensemble de sites pondérés $\mathcal{S} \in \mathbb{R}^3$
Output : Le diagramme d'Apollonius $\mathcal{A}(\mathcal{S})$

forall $s_i \in \mathcal{S}$ **do**
  forall $s_j \in \mathcal{S},\ i < j$ **do**
    $H_{ij} \leftarrow \text{parametrize}\,(s_i, s_j)$
    forall $s_k \in \mathcal{S},\ k \notin \{i, j\}$ **do**
      $H_{ij} \leftarrow \text{intersect}\,(s_i, s_j, s_k, H_{ij})$
      $H_{ijk} \leftarrow \text{parametrize}\,(s_i, s_j, s_k)$
      forall $s_l \in \mathcal{S},\ l \notin \{i, j, k\}$ **do**
        $H_{ijk} \leftarrow \text{intersect}\,(s_i, s_j, s_l, H_{ijk})$
        $H_{ijkl} \leftarrow \text{parametrize}\,(s_i, s_j, s_k, s_l)$
        forall $s_m \in \mathcal{S},\ m \notin \{i, j, k, l\}$ **do**
          $H_{ijkl} \leftarrow \text{intersect}\,(s_i, s_j, s_m, H_{ijkl})$
        end
        $\mathcal{A}(\mathcal{S}) \leftarrow \mathcal{A}(\mathcal{S}) \cup \{H_{ijkl}\}$
      end
      $\mathcal{A}(\mathcal{S}) \leftarrow \mathcal{A}(\mathcal{S}) \cup \{H_{ijk}\}$
    end
    $\mathcal{A}(\mathcal{S}) \leftarrow \mathcal{A}(\mathcal{S}) \cup \{H_{ij}\}$
  end
end

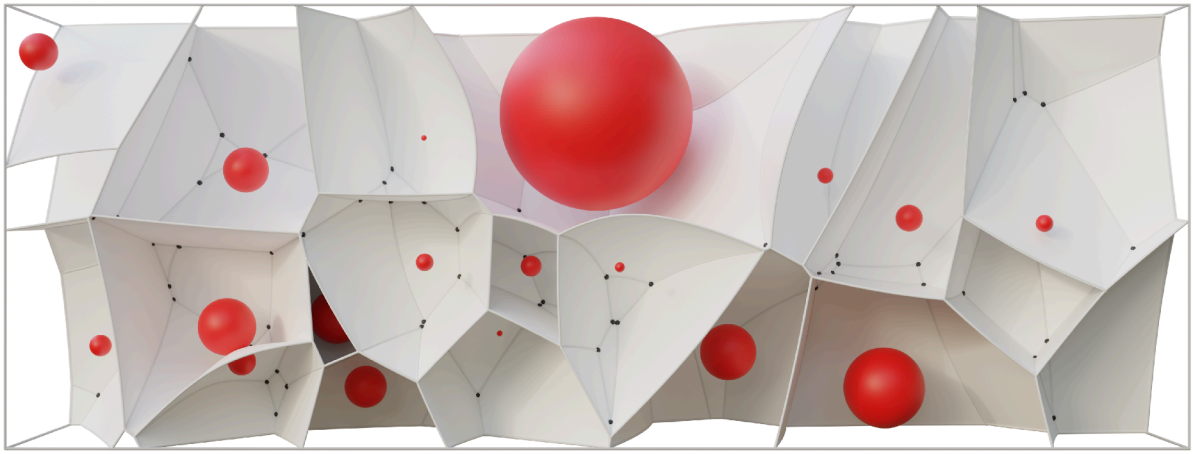


Figure 4.5 – Illustration de la géométrie obtenue à partir de algorithme 3. Il est intéressant de remarquer que les rayons des deux sites et leur proximité influent sur la courbure de leur bisecteur.

4.4 Conclusion

Dans ce chapitre, nous avons proposé une caractérisation complète des diagrammes d'Apollonius en $\mathbb{R}^2$ et $\mathbb{R}^3$. Celle-ci est effectuée dans $\mathbb{R}^d$ sur la base des travaux de Aurenhammer [Aur87] ainsi que Boissonnat et al. [BWY06] qui permet une analyse élégante à partir des diagrammes de Puissance en $\mathbb{R}^{d+1}$.

Le calcul de la géométrie d'un diagramme de façon naïve et simple d'implémentation supporte le développement d'algorithmes visant sa construction de façon performante. Ainsi même si la stratégie présentée est lente et nécessite de prendre en compte tous les sites de l'ensemble de départ, elle permet de calculer de façon exhaustive la géométrie de ces diagrammes.

Les caractérisations des faces d'Apollonius présentées permettent l'analyse des propriétés des diagrammes d'Apollonius et constituent une base fondatrice à d'autres études. Ainsi, c'est sur cette base que nous présentons dans le chapitre suivant notre méthode de calcul de diagramme d'Apollonius sur GPU.

Chapitre 5

Construction parallèle du diagramme d'Apollonius

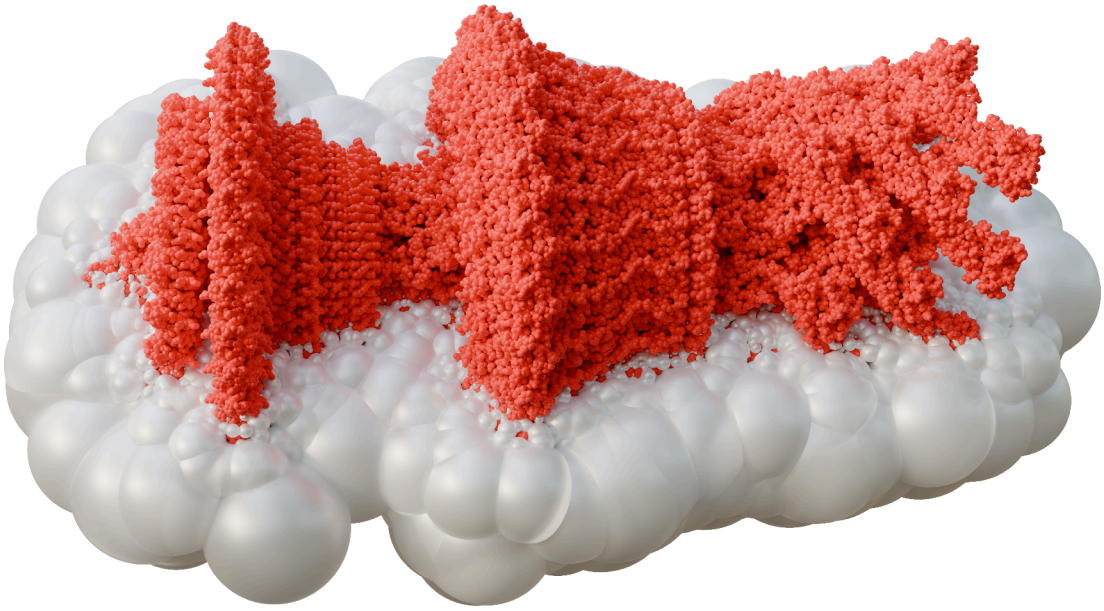


Figure 5.1 – Illustration d'une grande protéine (en rouge, PDB : 7CGO) composée de 335722 atomes et d'une partie de ses quadrisecteurs d'Apollonius (en blanc) calculés avec notre méthode de construction parallèle.

Sommaire

La caractérisation du diagramme d'Apollonius présentée dans le chapitre précédent permet l'étude en détail de ses faces, à l'origine de la complexité de sa construction. En effet, les multiples configurations remettent en cause les performances ou l'exhaustivité des algorithmes de construction qui sont souvent sensibles à des cas particuliers dépendant de la distribution spatiale de l'ensemble de sites et des poids associés. Notamment, les arêtes pouvant être composées de multiples segments et sans sommets (*i.e.* déconnectés), ils représentent un enjeu pour la robustesse et l'efficacité des algorithmes.

Dans ce contexte, les travaux précédents proposant des méthodes de calcul sont souvent basés sur des stratégies complexes à implémenter (section 2.3). Même s'ils permettent parfois le calcul rapide ou exhaustif des diagrammes d'Apollonius, la définition d'algorithmes combinant ces deux fonctionnalités reste un défi pour de nombreuses applications.

La construction de diagrammes d'Apollonius de données moléculaires, uniquement limitées par des règles physico-chimiques, est complexe. Notamment, leurs distributions spatiales hétérogènes requièrent le support d'analyse spatiale robuste et performante, mais aussi de cas particuliers difficiles à borner. De plus, même si les rayons des atomes sont majoritairement similaires, des variations importantes de la topologie peuvent toujours se produire, notamment en considérant les atomes d'hydrogène (1.20Å) qui sont de petites tailles en comparaison avec les autres éléments comme le potassium (2.75Å).

Nous présentons dans ce chapitre notre méthode de construction parallèle de diagrammes d'Apollonius sur GPU. Celle-ci est basée sur notre caractérisation et vise le support exhaustif

Symbole	**Signification**
δ	Distance euclidienne additivement pondérée.
x	Un point.
p_i, r_i, s_i, $\mathcal{S}$	Un site, un poids, un site pondéré et un ensemble de sites pondérés.
$\mathcal{A}(\mathcal{S})$, $\mathcal{A}(s)$, $\mathcal{A}^t(s)$	Un diagramme d'Apollonius, une cellule, et une cellule calculée à partir d'un sous-ensemble de t voisins de $\mathcal{S}$.
$\mathcal{A}(\sigma)$	Un ensemble de points dont l'ensemble de sites les plus proches est σ.
H_{ij}	Un bisecteur ou une face.
e_{ijk}, $\mathcal{E}$	Un trisecteur ou une arête et un ensemble d'arêtes.
v_{ijkl}, $\mathcal{V}$	Un quadrisecteur ou un sommet et un ensemble de sommets.
$\tau(x)$	La plus grande boule ouverte centrée sur x et d'intersection vide avec les sites de $\mathcal{S}$.

Table 5.1 – Nomenclature utilisée dans ce chapitre.

des caractéristiques de ces diagrammes jusqu'à la limite de la précision des nombres réels utilisés et de façon performante. Ce dernier critère est primordial pour les grands ensembles de sites et nécessite un processus compatible avec les restrictions des GPUs. Ainsi, notre méthode est basée sur une structure de données légère, mais aussi un traitement visant à réduire la divergence d'exécution. Comme certains travaux précédents [Ray+19 ; Bas+21], elle est fondée sur une construction itérative des cellules d'Apollonius en parallèle offrant une exécution efficace. Enfin, notre méthode propose plusieurs traitements additionnels répondant aux enjeux de calcul rapide de ces diagrammes pour des ensembles de sites spatialement hétérogènes, mais aussi homogènes.

La méthode présentée vise des ensembles de sites en position générale de la même façon que le chapitre précédent (chapitre 4). Ce critère peut être assuré à partir d'une perturbation aléatoire légère et permet de limiter les traitements aux cas généraux des arêtes d'Apollonius qui ne peuvent être qu'hyperboliques ou elliptiques. Les autres cas, en tant que limites, ne peuvent être satisfaits avec cette perturbation.

Nous introduisons tout d'abord les notions nécessaires à la définition de notre traitement (section 5.1) puis présentons notre algorithme de construction (section 5.2) et certaines particularités de notre implémentation GPU (section 5.3). Nous analysons ensuite ses performances (section 5.4) et concluons ce chapitre (section 5.5). La nomenclature de ce chapitre est présentée tableau 5.1.

5.1 Préambule

Dans cette section, nous présentons différentes caractéristiques des diagrammes d'Apollonius et outils mathématiques à la base de notre méthode de construction parallèle.

5.1.1 Diagrammes d'Apollonius et espace vide

La définition des diagrammes de Voronoï à partir de fonctions distance permet de reformuler leurs composantes en fonction de leur proximité. Soit $\mathit{nearest}(x)$, le prédicat retournant l'ensemble des sites les plus proches d'un point $x \in \mathbb{R}^d$ et défini par

$$\mathit{nearest}(x) \coloneqq \underset{s_i \in \mathcal{S}}{\arg\min}\, \delta(s_i,\ x).$$

Sur sa base, nous pouvons ainsi caractériser toutes les faces d'Apollonius en fonction de leur structure combinatoire. Soit $\mathcal{A}(\sigma)$, les points dont l'ensemble des sites les plus proches est σ et caractérisé par

$$\mathcal{A}(\sigma) \coloneqq \{x \mid \sigma \subseteq \mathit{nearest}(x)\}.$$

Ainsi, une cellule d'Apollonius $\mathcal{A}(s_i)$ d'un site pondéré $s_i = (p_i, r_i) \in \mathbb{R}^d \times \mathbb{R}$ est donnée par l'ensemble des points $\mathcal{A}(\{s_i\})$, une face H_{ij} par $\mathcal{A}(\{s_i,\ s_j\})$, etc. C'est à partir de cette modélisation que nous pouvons remarquer qu'une sphère centrée sur n'importe quel point x localisé sur une composante $\mathcal{A}(\sigma)$ du diagramme d'Apollonius et de rayon $\delta(s_i, x),\ s_i \in \sigma$, est tangente à σ, comme observé pour le diagramme de Voronoï [Del34]. Sa boule ouverte $\tau(x \in \mathcal{A}(\sigma))$ correspondante présente alors une intersection vide avec $\mathcal{S}$. Ainsi, un point x appartient à un diagramme d'Apollonius si et seulement si

$$\tau(x \in \mathcal{A}(\sigma)) \cap \mathcal{S} = \emptyset.$$

Cette formulation est au cœur de notre méthode puisqu'elle permet, comme nous le montrons dans la prochaine section, de tester l'appartenance des composantes au diagramme.

5.1.2 Bornes des arêtes

Avant de détailler notre méthode, nous introduisons un élément mathématique supplémentaire, nécessaire au calcul des composantes des diagrammes d'Apollonius.

Tandis que les sommets d'Apollonius peuvent être calculés à partir de la caractérisation développée chapitre 4, les arêtes d'Apollonius requièrent une paramétrisation coûteuse incompatible avec une implémentation performante. Afin de répondre au besoin d'analyse de ces composantes, nous introduisons un nouvel outil permettant de calculer les minimas et maximas des arêtes d'Apollonius et suffisant à leur construction.

Comme remarqué par Medvedev et al. [Med+06], les points d'une arête H_{ijk} minimisant et maximisant la distance aux sites peuvent être obtenus à partir du plan $p_ip_jp_k$ défini par les centres des sites associés. Si l'arête est ouverte, l'intersection $p_ip_jp_k \cap H_{ijk}$ est alors composée du point $\arg\min_x(\delta(s_i, x)), x \in H_{ijk}$. Si l'arête est fermée, elle est composée des deux points $\{\arg\min_x(\delta(s_i, x)), \arg\max_x(\delta(s_i, x))\}, x \in H_{ijk}$. Le calcul de ces points peut être réalisé en résolvant un système similaire à celui utilisé pour le calcul des sommets (section 4.2.3). Soit $\tilde{x} = (x, x_{d+1})^T \in \mathbb{R}^4$, le système est alors défini par

$$\begin{cases} \tilde{x} \cdot \tilde{n}_{ij} + c_{ij} = \tilde{x} \cdot \tilde{n}_{ik} + c_{ik} = \tilde{x} \cdot \tilde{n}_{ijk} + c_{ijk} = 0 \\ ||x - p_i|| = x_{d+1} + r_i, \end{cases}$$

avec

$$\tilde{n}_{ijk} = \left(\frac{(p_j - p_i) \times (p_k - p_i)}{||(p_j - p_i) \times (p_k - p_i)||}, 0 \right)^T \in \mathbb{R}^4, \text{ et } c_{ijk} = -(n_{ijk} \cdot p_i) \in \mathbb{R}.$$

L'équation du second degré obtenue à partir de ce système résulte en une solution dans le cas d'une arête ouverte, et deux solutions pour une arête fermée. Ainsi, en plus de servir à calculer les bornes des arêtes, elle peut être utilisée afin d'obtenir leurs types sans nécessiter une paramétrisation complète.

Dans la section suivante, nous utilisons les deux résultats présentés afin de construire un algorithme efficace et compact permettant la construction des diagrammes d'Apollonius.

5.2 Construction d'une cellule d'Apollonius

La construction de diagrammes de Voronoï par leur cellule permet une parallélisation efficace et simple d'implémentation, comme montré par des travaux précédents pour des diagrammes affines [Ray+19 ; Bas+21]. Ce traitement est couramment basé sur un processus itératif permettant la mise à jour de la cellule qui représente ainsi le principal enjeu. Cependant, tandis que les cellules affines peuvent être construites en modifiant la structure d'un polytope en fonction de faces planaires, les cellules d'Apollonius requièrent une prise en compte des faces hyperboliques.

Soit s_i, le site dont on souhaite construire la cellule d'Apollonius $\mathcal{A}(s_i)$, et $s_m \in \mathcal{S}$, un voisin nouvellement considéré. Celui-ci est dit *contribuant* à $\mathcal{A}(s_i)$ si et seulement si la face H_{im} existe. L'insertion de nouveaux voisins et la mise à jour de la structure de données sont ainsi des étapes fondatrices pour ce type d'algorithme. De plus comme illustré figure 5.2, ce traitement ne peut pas

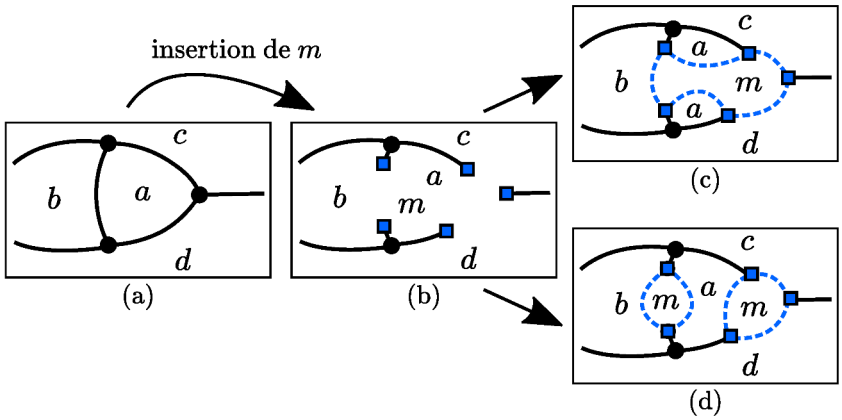


Figure 5.2 – Illustration d'une mise à jour topologique d'une cellule d'Apollonius. Les faces sont représentées par les indices des voisins associés. Lors de l'insertion d'un nouveau voisin, différentes configurations topologiques sont possibles et dépendent uniquement de critères géométriques. (a) Topologie initiale. (b) Insertion de s_m créant 5 nouveaux points de sommets. (c, d) Deux configurations possibles de la topologie.

uniquement se baser sur la structure topologique à la différence des diagrammes de Voronoï affine. En effet, les arêtes pouvant être composées de différents segments, les connexions entre les points des sommets sont dépendantes de facteurs géométriques potentiellement complexes à calculer.

Enfin, la construction d'une cellule implique l'exploration de l'espace afin de trouver efficacement le sous-ensemble de $\mathcal{S}$ contribuant. Les derniers travaux de construction de diagrammes de Voronoï visant des ensembles homogènes, ils sont basés sur le rayon de sécurité [LB13], qui n'est pas compatible avec des ensembles hétérogènes comme les protéines (section 2.3.3). Ainsi, notre méthode propose différents processus d'exploration spatiale additionnels pour le support efficace de ce type de distribution.

Dans cette section, nous introduisons notre algorithme de construction d'une cellule d'Apollonius, à la base de notre calcul parallèle. Nous présentons tout d'abord la structure de données de notre méthode (section 5.2.1), qui est suivie par l'initialisation de chaque cellule (section 5.2.2), et le processus de mise à jour topologique (section 5.2.3). Enfin, nous proposons plusieurs traitements additionnels basés sur une exploration spatiale permettant la construction efficace de diagrammes de distributions hétérogènes (section 5.2.4).

5.2.1 Structure de données

La construction exhaustive et efficace de diagrammes d'Apollonius repose notamment sur les caractéristiques de la structure de données utilisée. Notamment, comme illustré figure 5.2, la représentation compacte de leurs caractéristiques est complexe en raison des multiples cas possibles. Afin de répondre aux contraintes de mémoire des GPUs, nous proposons une structure basée sur un stockage minimal des informations nécessaires au recalcul en cas de besoin. Celle-ci est illustrée figure 5.3.

Faces Les faces d'Apollonius, H_{ij}, étant totalement définies par la structure combinatoire de leurs arêtes, nous ne les stockons pas explicitement. Elles peuvent être retrouvées en cas de besoin à partir du reste de la structure.

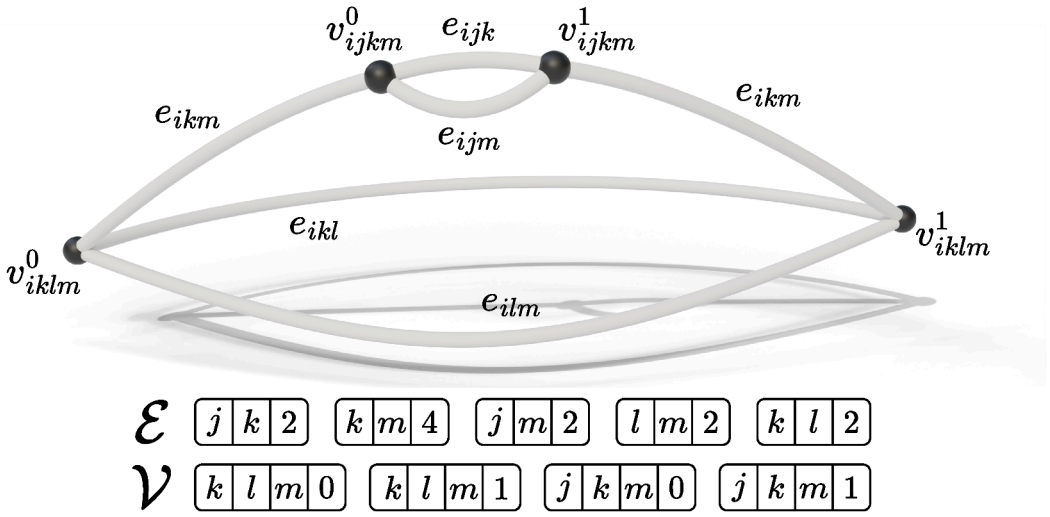


Figure 5.3 – Illustration de notre structure de données. Celle-ci caractérise complètement la structure d'une cellule d'Apollonius malgré sa complexité. Une arête composée de plusieurs segments comme e_{ikm} est représentée par les deux indices k et m ainsi que son nombre de points de sommets (4). Un point d'un sommet comme v^1_{iklm} est représenté par les trois indices k, l et m de même que l'indice de sa solution (1).

Arêtes Les arêtes d'Apollonius, e_{ijk}, ne sont pas entièrement caractérisées par leurs sommets. En effet, puisqu'elles peuvent être composées de multiples segments, elles représentent un enjeu en termes de compacité. Ainsi, nous ne stockons que les deux indices des voisins correspondants, le premier étant implicitement donné par la cellule. De plus, nous maintenons le nombre de sommets positionnés sur l'arête. Puisqu'un sommet peut être composé de plusieurs points, nous maintenons le nombre de points total plutôt que de sommets qui peuvent alors être traités indépendamment.

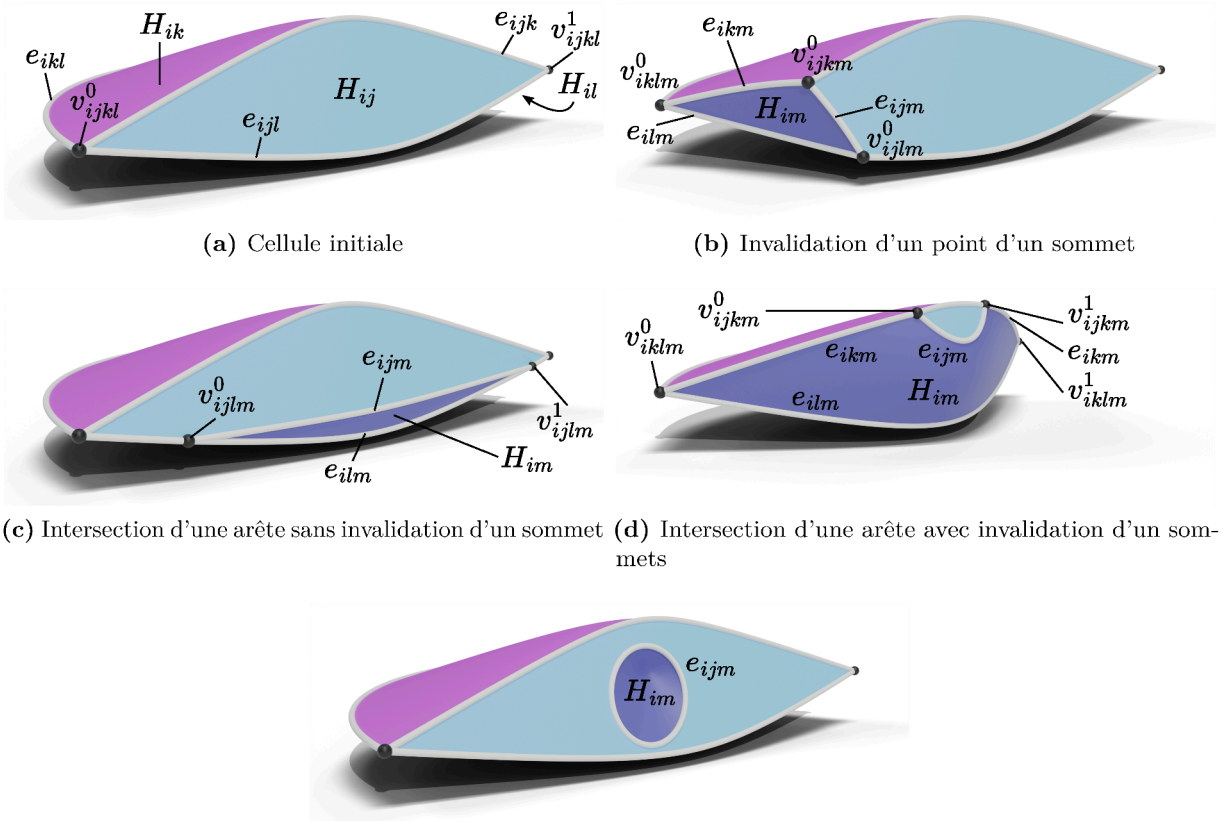
Sommets Les sommets d'Apollonius, v_{ijkl}, bornant les arêtes, nous les stockons à partir des trois indices de leurs voisins correspondants. D'une façon similaire aux arêtes, nous stockons explicitement chaque point valide des sommets pour un traitement uniforme. Ainsi, en plus des indices des voisins, nous stockons celui qualifiant la solution correspondante à un point.

Les composantes présentées sont stockées dans des listes dédiées aux sommets $\mathcal{V}$ et arêtes $\mathcal{E}$. Cette structure de données, orientée vers un stockage minimal, permet de représenter de façon exhaustive la structure des diagrammes d'Apollonius tout en étant suffisamment compacte pour être compatible avec les contraintes mémoires des GPUs. Notamment, elle ne stocke pas explicitement la connectivité des faces, pouvant être complexe à maintenir (figure 5.2). À défaut, celle-ci est retrouvée en cas de besoin à partir des indices des composantes permettant un compromis entre performance de calcul et stockage minimal. Cette structure représente le support de notre stratégie de construction.

5.2.2 Initialisation

La structure de données présentée précédemment nécessite une initialisation avant d'être itérativement construite en fonction de son voisinage. En effet, les faces et arêtes non bornées ne sont pas compatibles avec un traitement uniforme qui devrait prendre en compte de nombreux cas particuliers.

Si la caractérisation des diagrammes affines comme polytopes permet l'utilisation de la boite englobante de l'ensemble d'entrée comme valeur d'initialisation de chaque cellule, notre structure de données nécessite cependant des indices de sites correspondants à chaque face. Nous utilisons alors une méthode basée sur l'ajout de nouveaux sites à $\mathcal{S}$. Comme remarqué par Boissonnat et Karavelas [BK03], les arêtes non bornées des diagrammes d'Apollonius sont topologiquement équivalentes à l'enveloppe convexe de l'ensemble de sites d'entrée $\mathcal{S}$. Ainsi, nous ajoutons de nouveaux sites modifiant l'enveloppe convexe de façon à ce qu'aucun site de $\mathcal{S}$ ne contribue à des arêtes non bornées. Pour cela, nous introduisons 6 sites artificiels positionnés sur les axes à une distance de $\sqrt{3}r_{max}$ avec r_{max}, le rayon de la sphère englobante de $\mathcal{S}$. Cette distance permet d'assurer que la nouvelle enveloppe convexe, de la forme d'un octahédron, n'intersecte pas $\mathcal{S}$. Chaque cellule peut alors être initialisée uniformément et de façon performante en calculant ses arêtes et sommets liés aux sites artificiels.



(a) Cellule initiale

(b) Invalidation d'un point d'un sommet

(c) Intersection d'une arête sans invalidation d'un sommet

(d) Intersection d'une arête avec invalidation d'un sommets

(e) Intersection d'une face et création d'une arête fermée

Figure 5.4 – Illustration d'une cellule initiale $\mathcal{A}(s_i)$ (a) et des cas possibles de modifications topologiques après l'insertion d'un nouveau voisin s_m contribuant à l'arête H_{im} (b, c, d, e). Pour des raisons de lisibilité, seules les nouvelles faces et arêtes sont nommées. (b) Le point v^0_{ijkl} est invalidé et s_m contribue aux nouveaux sommets v_{iklm}, v_{ijkm} et v_{ijlm} et leurs arêtes. (c) H_{im} intersecte e_{ijl} ce qui produit le nouveau sommet v_{ijlm} composé de deux points et leurs arêtes. (d) s_m invalide complètement le sommet v_{ijkl} et contribue aux nouveaux sommets v_{iklm}, v_{ijkm}, composés de deux points, et leurs arêtes. (e) H_{im} intersecte la face H_{ij} et contribue à la nouvelle arête fermée e_{ijm}. Notre méthode de mise à jour topologique supporte tous ces cas.

5.2.3 Mise à jour topologique

La mise à jour topologique d'une cellule représente la composante principale de notre traitement. Soit s_i, un site pondéré dont on souhaite construire la cellule $\mathcal{A}(s_i)$, et $\mathcal{A}^t(s_i)$, la cellule d'Apollonius de s_i construite à partir d'un sous-ensemble $\mathcal{N}^t \subset \mathcal{S}$ de t sites. On souhaite alors construire $\mathcal{A}^{t+1}(s_i)$, la cellule de s_i, à partir du sous-ensemble $\mathcal{N}^{t+1} = \mathcal{N}^t \cup \{s_m\}$, construit avec $\mathcal{N}^t$ et un voisin nouvellement considéré s_m.

Cette étape requiert la prise en compte des multiples cas possibles résultant des configurations du diagramme d'Apollonius qui sont illustrées figure 5.4. Ainsi, nous présentons dans cette section notre méthode réalisant la mise à jour topologique d'une cellule et supportant tous les cas présentés.

Prédicat Nous définissons un unique prédicat permettant de tester l'existence d'une composante d'Apollonius. Celui-ci est basé sur l'observation qu'un nouveau voisin contribue à une cellule si et seulement s'il existe une sphère qui leur est tangente dont la boule ouverte n'a pas d'intersection avec $\mathcal{N}^t$. Ainsi, le prédicat *valid* teste l'existence d'une composante $\mathcal{A}(\sigma)$ par rapport à un site s_j et est défini par

$$valid\,(\mathcal{A}(\sigma),\ s_j) \coloneqq \exists x \in \mathcal{A}(\sigma) \text{ tel que } \tau(x) \cap s_j = \emptyset,$$

avec $\tau(x)$ la boule ouverte de la sphère centrée sur x et tangente à σ. L'implémentation de ce prédicat pour les sommets requiert ainsi le calcul d'un point du sommet (section 4.2.3) tandis que celle des arêtes peut être effectuée à partir de l'analyse de leurs bornes (section 5.1.2). Une composante $\mathcal{A}(\sigma)$ d'une cellule $\mathcal{A}^t(s_i)$ est alors dite *valide*, si elle satisfait $valid\,(\mathcal{A}(\sigma),\ s_j)$ pour tout $s_j \in \mathcal{N}^t$.

Invalidation de sommets Notre traitement débute par le test d'existence des sommets de la structure de données, détaillé algorithme 4. Ainsi, si un sommet v_{ijkl} est invalidé par le nouveau voisin s_m, il est supprimé de la liste de sommet $\mathcal{V}$ (l.3-6 de algorithme 4). Dans

Algorithme 4 : Invalidation des sommets

```
   Input : Une cellule A^t(s_i) et un nouveau site s_m
 1 E, V ← A^t(s_i) ▷ Lecture de la structure de données
 2 forall v_ijkl ∈ V do
 3 |   if valid(v_ijkl, s_m) then
 4 |   |   continue
 5 |   end
 6 |   Tag v_ijkl as invalid
 7 |   forall e_ij'k' ∈ E do
   |   |   ▷ Si v_ijkl n'est pas sur e_ij'k'
 8 |   |   if {j', k'} ∉ {j, k, l} then
 9 |   |   |   continue
10 |   |   end
11 |   |   Decrement e_ik'l''s vertex number
12 |   end
13 end
```

ce cas, nous décrémentons aussi le nombre de sommets des arêtes sur lesquelles il est positionné (l.7-12 de algorithme 4).

Calcul des nouveaux sommets De nombreux cas peuvent conduire à la création de nouveaux sommets d'Apollonius (figure 5.4b, figure 5.4d, figure 5.4c). Pour proposer un traitement uniforme malgré ces possibilités, nous testons l'intersection entre chaque arête e_{ijk} et la face H_{im}. La validité du sommet en résultant, v_{ijkm}, est testée à partir du prédicat *valid* et chaque site de la structure de données (l.4-9 de algorithme 5). Si un point du sommet est valide, nous l'ajoutons à la liste $\mathcal{V}$. De plus, nous créons les nouvelles arêtes e_{ijm} et e_{ikm} qui sont ajoutées à une liste temporaire $\mathcal{E}*$ (l.10-14 de algorithme 5). Puisqu'une même arête peut être créée plusieurs fois, cette liste est ainsi utilisée afin d'éviter toute redondance avant l'ajout dans la structure de données $\mathcal{E}$. Enfin, nous ajoutons toutes les valeurs uniques de $\mathcal{E}*$ à $\mathcal{E}$ et supprimons les arêtes ouvertes sans sommet (l.22-23 de algorithme 5).

Arêtes fermées Les arêtes fermées nécessitent un traitement particulier puisqu'elles ne peuvent être découvertes à partir des sommets (figure 5.4e). Nous remarquons cependant qu'une arête fermée n'existe que si un de ses points est valide, qu'elle est elliptique et qu'aucun sommet ne peut être partagé avec un autre site de la structure combinatoire. En effet, si un sommet

Algorithme 5 : Calcul des nouveaux sommets

```
   Input : Une cellule A^t(s_i) et un nouveau site s_m
 1 E* ← {} ▷ Liste temporaire des nouvelles arêtes
 2 forall e_ijk ∈ E do
 3 |   v_ijkm ← (H_im ∩ e_ijk)
   |   ▷ Valide le nouveau sommet v_ijkm
 4 |   forall s_l ∈ N^t \ {s_j, s_k} do
 5 |   |   if ¬valid(v_ijkm, s_l) then
 6 |   |   |   Tag v_ijkm as invalid
 7 |   |   |   break
 8 |   |   end
 9 |   end
10 |   if v_ijkm is invalid then
11 |   |   E* ← E* ∪ {e_ijm, e_ikm}
12 |   |   V ← V ∪ {v_ijkm}
13 |   |   Increment e_ijk's vertex number
14 |   end
   |   ▷ Teste si l'arête e_ijk est complètement invalidée
15 |   if e_ijk is not a closed elliptic edge then
16 |   |   continue
17 |   end
18 |   if H_im ∩ e_ijk = ∅ ∧ ¬valid(e_ijk, s_m) then
19 |   |   E ← E \ {e_ijk}
20 |   end
21 end
22 Add all unique value in E* to E
23 Remove open edges without vertices from E
```

est possible, mais invalide, l'arête est forcément intersectée ou totalement invalide. Ainsi, nous testons l'existence d'un arête fermée et vérifiant ces caractéristiques entre le site de la cellule s_i, le nouveau site s_m et chaque site de la structure $s_j \in \mathcal{N}^t$ (l.2-8 de algorithme 6). Si une arête e_{ijm} satisfait ces critères, nous validons son existence à partir de *valid* et de la structure combinatoire (l.9-14 de algorithme 6). Si l'arête est valide, nous l'ajoutons à la liste $\mathcal{E}$ (l.15-18 de algorithme 6). Enfin, une arête elliptique fermée e_{ijk} peut être totalement invalidée par un nouveau voisin s_m. Afin de supporter ce cas, si aucun sommet n'est possible entre e_{ijk} et s_m, nous vérifions sa validité (l.15-20 algorithme 5). Dans le cas où une boule ouverte centrée sur l'un des points de e_{ijk} intersecte s_m, e_{ijk} est totalement invalide et supprimée de $\mathcal{E}$.

À la fin de ce traitement, la structure de données de la cellule est totalement mise à jour. Même si celui-ci nécessite de nombreuses opérations, il se prête particulièrement bien au compromis entre puissance de calcul et consommation mémoire des GPUs. Afin de proposer de bonnes performances, il nécessite cependant une sélection précise des voisins sélectionnés.

5.2.4 Exploration spatiale

Afin de ne pas considérer la totalité de $\mathcal{S}$ et trouver le sous-ensemble contribuant d'une cellule de façon performante, plusieurs traitements dédiés sont nécessaires. Notamment, même si le rayon de sécurité permet le support de diagrammes affines (section 2.3.3), il n'est pas directement compatible avec les diagrammes d'Apollonius. De plus, il ne suffit pas au traitement efficace de distributions hétérogènes qui nécessitent une exploration spatiale plus précise. Nous présentons ainsi plusieurs méthodes dédiées à ces caractéristiques.

Algorithme 6 : Calcul des arêtes fermées

Input : Une cellule $\mathcal{A}^t(s_i)$, un nouveau site s_m et la liste des nouvelles arêtes $\mathcal{E}*$

```
1  forall s_j ∈ N^t do
2      e_ijm ← H_ij ∩ H_im
3      if s_j can contribute to a vertex with s_m then
4          continue
5      end
6      if e_ijm is not elliptic then
7          continue
8      end
       ▷ Valide la nouvelle arête e_ijm
9      forall s_k ∈ N^t \ {s_j} do
10         if ¬valid(e_ijm, s_l) then
11             e_ijm does not exist
12             break
13         end
14     end
15     if e_ijm exists then
16         Tag e_ijm as closed
17         E ← E ∪ {e_ijm}
18     end
19 end
```

Rayon de sécurité Le rayon de sécurité [LB13], est basé sur l'observation que la distance maximale entre un site et une composante de sa cellule d'un diagramme affine est toujours atteinte sur un sommet. Cette formulation n'est cependant pas compatible avec les diagrammes d'Apollonius, dont les composantes ne sont, par exemple, pas forcément bornées par des sommets. Le rayon de sécurité peut ainsi être reformulé de façon plus générale pour le diagramme d'Apollonius. Soit $x_{max}(\mathcal{A}(s_i))$, le point satisfaisant

$$x_{max}(\mathcal{A}(s_i)) \coloneqq \arg\max_x \left(\delta(s_i,\ x)\right), x \in \mathcal{A}(s_i).$$

Sur la base de notre caractérisation (section 4.2.1), nous pouvons remarquer qu'une face d'Apollonius H_{ij} est bornée, de la même façon que les diagrammes affines, par le minimum

$$\min \delta(x, s_i) = \tfrac{1}{2}\delta(s_i, s_j),\ x \in H_{ij}.$$

En définissant le rayon de sécurité r_{max} avec

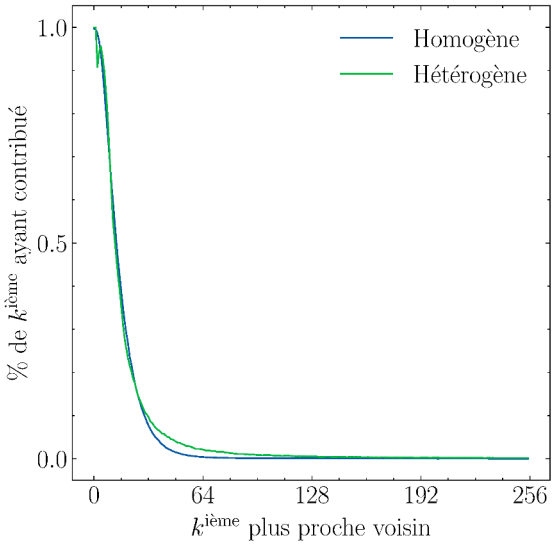
$$r_{max}(\mathcal{A}(s_i)) \coloneqq 2\delta(s_i,\ x_{max}(\mathcal{A}(s_i))),$$

nous pouvons alors assurer qu'un site s_m peut contribuer à une cellule d'Apollonius si et seulement s'il respecte

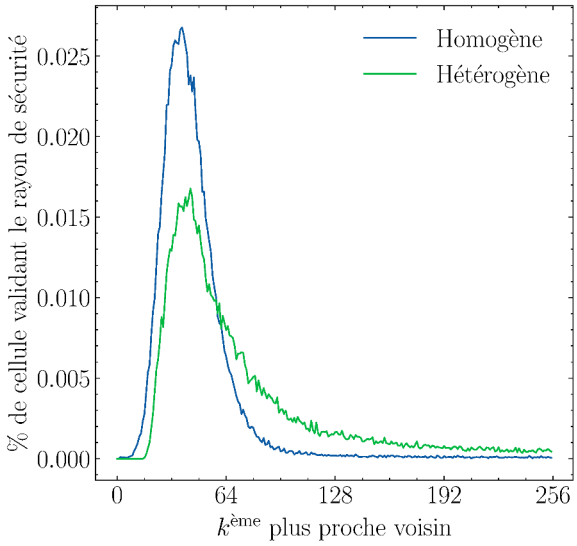
$$\delta(s_i, s_j) \leq r_{max}(\mathcal{A}(s)).$$

Le rayon de sécurité est ainsi compatible avec les cellules d'Apollonius. Afin de trouver $x_{max}(\mathcal{A}(s_i))$, nous remarquons que notre initialisation (section 5.2.2) permet d'assurer que la distance maximale d'une arête hyperbolique est toujours bornée par ses sommets. De plus, une arête elliptique est soit bornée par son maximum (section 5.1.2), s'il est compris dans le segment d'arête, soit par ses sommets. Tester si le maximum appartient à l'arête effective représente cependant un traitement complexe. Afin de le simplifier, nous calculons dans les faits $x_{max}(\mathcal{A}(s_i))$ à partir de la distance des sommets et du maximum des arêtes elliptiques (section 5.1.2) permettant ainsi moins de divergences d'exécution.

Validation des coins Même si le rayon de sécurité permet un traitement adapté aux distributions uniformes, il ne convient pas aux distributions hétérogènes ne permettant pas d'associer plus proches voisins et voisins contribuants, comme illustré figure 5.5. Ainsi, comme remarqué par Sainlot et al., une cellule de diagramme de Voronoï peut être construite en explorant l'espace à partir des sphères tangentes centrées sur chacun de ses sommets, ou coins [SNA17]. La cellule est alors construite en ajoutant à sa topologie les voisins intersectant la boule ouverte correspondant à ces sphères. Cela permet ainsi une exploration de l'espace plus fine que celle basée sur le rayon de sécurité et de meilleures performances dans un contexte hétérogène. Cependant, elle ne permet pas la validation complète de la cellule lorsqu'elle n'est pas complètement couverte par ses sommets. Notamment, les faces et arêtes courbées peuvent comprendre de grandes zones qui ne sont pas présentes à l'intérieur des sphères centrées sur les faces. Afin de profiter des avantages de la validation des coins tout en proposant une construction exhaustive, nous définissons plusieurs traitements additionnels adaptés aux particularités des faces et arêtes d'Apollonius.



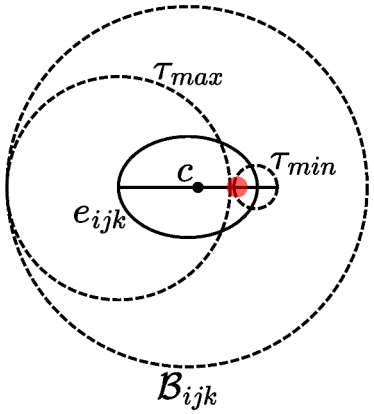
(a) Pourcentage de cellule dont le $k^{\text{ième}}$ plus proche voisin a contribué

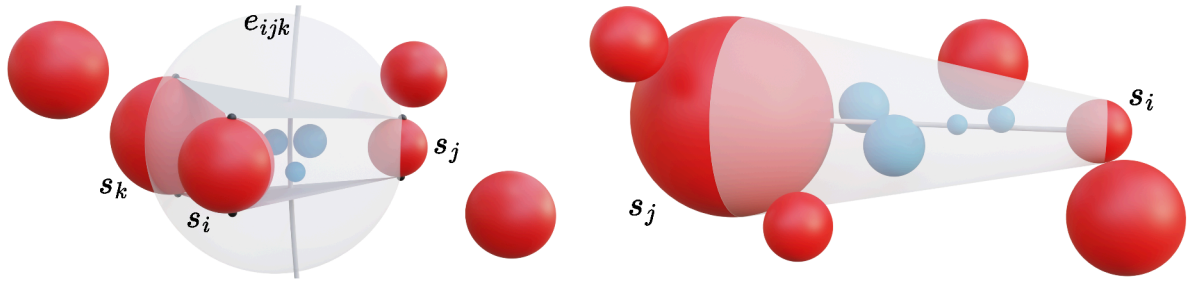


(b) Pourcentage de cellule ayant atteint le rayon de sécurité

Figure 5.5 – Analyse des performances du rayon de sécurité pour une distribution homogène (100000 sites aléatoires de rayon compris entre 1 et 2) et hétérogène (PDB : 1AON de 58870 sites). (a) Tandis que k augmente, le pourcentage de $k^{\text{ième}}$ voisin contribuant diminue pour les deux types de distributions. Cela implique que la pertinence de la construction d'une cellule en fonction de ses plus proches voisins diminue fortement après un seuil donné (ici, $k = 32$), quelle que soit la distribution. (b) Le nombre de cellules atteignant le rayon de sécurité augmente beaucoup plus vite pour une distribution homogène que pour une distribution hétérogène. Dans ce dernier cas, les cellules requièrent la prise en compte de beaucoup plus de voisins pour s'assurer de la validité des cellules.

Validation des arêtes Afin de valider les arêtes, il est nécessaire de définir la zone non validée par les sommets. Ainsi, puisque les arêtes hyperboliques sont toujours bornées par l'initialisation (section 5.2.2), nous pouvons assurer qu'une partie de l'arête est déjà validée. Celle-ci, comme remarqué par Olechnovič et Venclovas [OV14], est au minimum composée des deux demi-espaces définis par les plans tangents au triplet de sphères (section 2.3.2). Cette configuration représente le cas limite définissant la zone non validée la plus grande et apparait lorsque les deux sommets bornant les arêtes hyperboliques sont localisés à l'infini. La zone restante à valider est alors caractérisée par la cyclide de Dupin passant par les sites de l'arête. Afin de l'explorer de façon performante, nous effectuons une recherche spatiale à partir des plans tangents ainsi que de la sphère passant par les points des plans localisés sur les sites (figure 5.6a). Ce traitement permet ainsi de limiter de façon importante la zone de recherche, mais n'est cependant pas compatible avec les arêtes elliptiques. En effet, ces dernières peuvent être composées de multiples segments qui demanderaient un traitement important afin d'être reconstruits. À la place, nous utilisons la sphère englobant toutes les sphères tangentes à $\mathcal{S}$ localisées sur l'arête. Celle-ci peut être calculée efficacement à partir des sphères de rayon minimum et maximum appartenant à l'arête (section 5.1). Enfin, il





(a) Validation des arêtes **(b)** Validation des faces

Figure 5.6 – Illustration des zones de validation utilisées pour les arêtes et des faces ainsi que des sites qui n'ont pas encore été intégrés à la structure (en bleu). (a) La zone non validée des arêtes est approchée par les plans tangents aux trois sphères et une sphère passant par les points d'intersection. (b) La zone non validée des faces est caractérisée par un tronc du cône passant par les deux sites de la face.

est nécessaire d'effectuer une nouvelle validation des sommets à chaque fois que de nouveaux sont trouvés pendant la validation des arêtes.

Validation des faces D'une façon similaire aux arêtes hyperboliques, nous dérivons un cas limitant les possibilités de zones validées des faces. Ainsi, puisque toutes les faces sont bornées (section 5.2.2), la plus grande zone à valider est caractérisée par une arête elliptique bornant la face H_{ij} à l'infini. Après la validation des arêtes, la zone restant à valider pour une face est alors caractérisée par l'enveloppe convexe des deux sites s_i et s_j, comme illustré figure 5.6b. Afin de valider cette forme, nous utilisons un cylindre de rayon $\max(r_i, r_j)$ permettant un traitement uniforme tout en supportant le cas particulier de $r_i = r_j$.

La méthode présentée dans cette section permet la construction d'une cellule d'Apollonius ainsi que l'exploration de l'espace pour la recherche des voisins contribuants. Celle-ci, compatible avec des distributions variées, permet une construction exhaustive. De plus, ses caractéristiques la rendent adaptée à une parallélisation sur GPU.

5.3 Implémentation GPU

Notre méthode permet une implémentation efficace sur GPU basée sur une parallélisation par cellule. Cependant, elle nécessite différents choix d'implémentation particuliers afin de proposer de hauts niveaux de performance avec des configurations de sites variées. Ainsi, nous présentons dans cette section différentes caractéristiques de notre implémentation. Celle-ci est basée sur le langage CUDA, comme la méthode présentée chapitre 3.

5.3.1 Détails d'implémentation

Les étapes de notre construction itérative de cellule requièrent plusieurs opérations sur la structure de données. Notamment, nous utilisons des traitements nécessitant des itérations à travers l'entièreté de la structure de données. De plus, le type de distribution influe de façon importante sur la complexité de la structure de chaque cellule que l'on souhaite stocker dans un cache rapide. Afin de pallier ces deux types de problèmes, nous utilisons un traitement coopératif des cellules permettant ainsi une mise en cache.

Afin de réduire les potentielles divergences d'exécution entre threads traitant des cellules différentes, nous utilisons un warp complet pour chaque cellule. Ainsi, les opérations coûteuses sont partagées permettant de réduire les latences. Ce type de traitement est particulièrement adapté à l'implémentation du prédicat *valid*. Puisque notre structure ne contient pas explicitement la liste des voisins contribuants $\mathcal{N}$, celle-ci nécessite le parcours de la structure combinatoire à partir de la liste d'arêtes $\mathcal{E}$. Grâce à ce traitement coopératif, ce calcul est ainsi amorti et ne provoque pas de divergence d'exécution (code source 5.1).

Ce traitement coopératif permet aussi de limiter la pression sur la mémoire partagée, celle-ci n'étant plus que partagée entre warp d'un bloc et non plus par thread. Ainsi, nous disposons d'une quantité de cache importante permettant de stocker des cellules complexes, nécessaires à la construction de diagrammes de distribution hétérogènes.

Enfin, dans notre implémentation, nous utilisons des nombres flottants 64 bits comme les travaux précédents [Ray+19 ; Bas+21]. Cette configuration impacte fortement les performances sur GPU puisqu'ils proposent beaucoup moins d'unités de calcul pour nombres flottants 64 bits que 32 bits [NVI23]. Cependant, elle permet de limiter de façon importante les imprécisions numériques sans utiliser d'arithmétique exacte qui requiert de nombreux traitements particuliers incompatibles avec les besoins de cohérences d'exécution des GPUs.

5.3.2 Validation spatiale et structure accélératrice

Les étapes de validation spatiale présentées dans la partie précédente permettent la construction des cellules d'Apollonius. Cependant, afin de proposer de bonnes performances de calculs, nous présentons certaines particularités de notre implémentation.

Notamment, même si la validation des composantes de la cellule de façon itérative permet une construction exhaustive, elle nécessite de nombreuses requêtes spatiales qui ne sont pas toujours adaptées. Comme illustré figure 5.7, les performances sont meilleures lorsque le traitement commence par traiter quelques plus proches voisins plutôt qu'en utilisant directement une

```
__device__ bool valid( Site* sites, Edge* edges, Real4 sphere )
{
    const auto warp    = cg::tiled_partition<WarpSize>( block );
    bool       isValid = true;

    uint32_t offset = warp.thread_rank();
    for ( uint8_t e = offset; e < edges.size; e += warp.size )
    {
        const Edge & edge = edges[ e ];
        isValid &= !intersect( sites[ edge.j ], sphere );
        isValid &= !intersect( sites[ edge.k ], sphere );
    }
    return warp.all( isValid );
}
```

Code Source 5.1 – Implémentation coopérative en CUDA du prédicat *valid*. Chaque thread teste si la sphère intersecte une sous-partie de la structure combinatoire. Le résultat est ensuite partagé de façon efficace via `all`.

validation des composantes. Ainsi, nous permettons de sélectionner le nombre de plus proches voisins à considérer avant d'effectuer une construction par validation dans notre implémentation.

Enfin, pour effectuer les nombreuses requêtes spatiales de recherche de voisinage, nous utilisons une structure accélératrice. Dans notre implémentation, nous utilisons un BVH qui permet l'exécution de requête de tailles très variées de façon plus performante que les grilles accélératrices utilisées par les méthodes précédentes [Ray+19 ; Bas+21]. Plus précisément, nous utilisons un LBVH qui offre une construction rapide de la structure tout en permettant des requêtes efficaces [Kar12].

Les caractéristiques de notre implémentation permettent d'utiliser les fonctionnalités des GPU afin de bénéficier de traitement adapté à des distributions variées. Ainsi, nous étudions ses performances dans la prochaine section.

5.4 Résultats

Dans cette section, nous présentons les résultats obtenus à partir de notre méthode. Nous évaluons notamment ses performances de calcul en comparaison avec la méthode la plus rapide de l'état de l'art (section 5.4.1), mais aussi la validité de la topologie obtenue (section 5.4.2). Enfin, nous présentons une application utilisant notre méthode et soutenant l'illustration moléculaire (section 5.4.3) ainsi que ses limites (section 5.4.4).

5.4.1 Performances

Afin d'évaluer la rapidité de calcul de notre méthode, nous la comparons avec l'implémentation de Edge-Tracing de Olechnovič et Venclovas [OV14]. Celle-ci présente en effet les meilleures performances de l'état de l'art et est de plus publiquement accessible. Nous définissons trois jeux de données visant à tester les capacités des deux méthodes avec des configurations très différentes :

— *Protéines* : Protéines de petites et grandes tailles dont les rayons des atomes sont compris entre 1.5Å et 2.4Å.

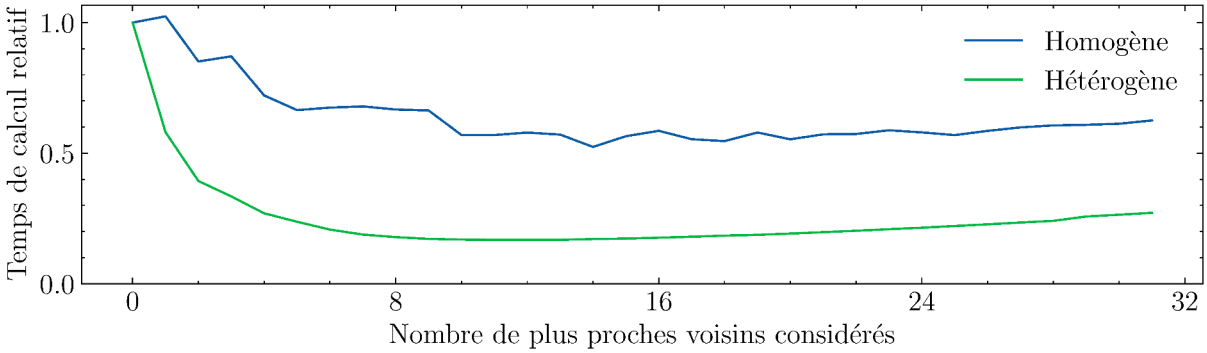


Figure 5.7 – Analyse des temps de calcul relatifs de construction d'un diagramme d'Apollonius en utilisant un nombre k de plus proches voisins avant la validation des composantes avec une distribution pour une distribution homogène (10000 sites aléatoires de rayon compris entre 0.1 et 10) et hétérogène (PDB : 1AON de 58870 sites). Ainsi, considérer quelques plus proches voisins permet d'obtenir de meilleures performances que de n'en considérer aucun. Cet effet est plus marqué sur les distributions hétérogènes qui sont moins adaptées à un traitement avec le rayon de sécurité que les distributions homogènes.

— *Bruit blanc modéré* : Sites répartis de façon aléatoire à partir d'un bruit blanc et dont les rayons sont compris entre 1 et 3.
— *Bruit blanc* : Sites répartis de façon aléatoire à partir d'un bruit blanc et dont les rayons sont compris entre 0.1 et 10.

Tandis que le jeu de données de protéines nécessite le traitement efficace de données spatialement très hétérogènes, mais de rayon similaire, ceux produits à partir de bruit blanc requièrent le support exhaustif des composantes d'Apollonius en raison de leur distribution spatiale plus uniforme. Le bruit blanc est généré à partir de PCG [ONe14 ; JO20]. Enfin, nous configurons

Protéines $r_i \in [1.5, 2.4]$				
PDB	**#Sites**	**Voronota (ms)**	**Notre méthode (ms)**	**Accélération**
5ZCK	31	0.17	2.12	0.08 x
1AGA	126	0.74	2.73	0.27 x
3DIK	219	1.49	2.83	0.53 x
101M	1413	12.38	6.53	1.90 x
1A3F	2784	26.27	10.20	2.58 x
1A2Z	7666	72.58	20.65	3.51 x
8ID8	8635	86.40	44.93	1.92 x
7DBB	17733	209.26	113.15	1.85 x
7P3W	37149	415.04	256.18	1.62 x
7O0U	55758	616.16	486.50	1.27 x
1AON	58870	634.33	169.42	3.74 x
6QZ9	71724	893.83	459.08	1.95 x
3JC8	107640	1392.98	727.00	1.92 x
4V8W	123082	1522.44	317.10	4.80 x
7LER	158430	1978.83	950.03	2.08 x
6RXU	211834	2913.51	1540.50	1.89 x
4V6X	237685	3680.48	1793.66	2.05 x
7CGO	335722	5136.54	2523.89	2.04 x
4V60	483912	7357.97	3412.78	2.16 x
6U42	1358547	30701.77	10325.29	2.97 x
Bruit blanc modéré $r \in [1, 3]$				
Échelle	**#Sites**	**Voronota (ms)**	**Notre méthode (ms)**	**Speedup**
25.0	100	1.08	2.96	0.36 x
50.0	1000	16.64	13.17	1.26 x
250.0	10000	637.94	81.54	7.82 x
500.0	100000	9858.05	1887.71	5.22 x
Bruit blanc $r \in [.1, 10.1]$				
Échelle	**#Sites**	**Voronota (ms)**	**Notre méthode (ms)**	**Accélération**
100.0	100	1.37	2.93	0.47 x
200.0	1000	29.32	16.54	1.77 x
500.0	10000	933.20	205.76	4.54 x
1000.0	100000	29970.22	7590.99	3.95 x

Table 5.2 – Performances de notre méthode comparées à celles de Voronota [OV14] sur nos trois jeux de données avec un Intel I9-13900K et une NVIDIA RTX 4090. Chaque temps de calcul représente la moyenne de 100 itérations précédées de 10 itérations qui ne sont pas prises en compte. La version de Voronota est basée sur une parallélisation avec OpenMP utilisant les 32 cœurs du CPU.

notre méthode avec des paramètres permettant le support de tous les éléments des jeux de données testés sans changement de paramètres. Ainsi, nous fixons empiriquement le nombre maximum d'arêtes à 264 et de sommets à 152. Sur la base de figure 5.7, nous utilisons 16 plus proches voisins avant la validation des composantes de la cellule permettant un compromis entre les distributions homogènes et hétérogènes. Les ensembles de sites d'entrée sont de plus perturbés aléatoirement [ONe14 ; JO20] avec des valeurs entre $[-1e^{-3}, 1e^{-3}]$ afin d'assurer des configurations en position générale.

Les performances sont présentées tableau 5.2 et sont effectuées à partir d'un CPU Intel I9 13900K d'un GPU NVIDIA RTX 4090. Nous remarquons tout d'abord que notre méthode permet de meilleures performances de calcul à partir du millier de sites, comme attendu avec une méthode GPU. L'accélération de notre méthode par rapport à Voronota est relativement constante sur le jeu de données de protéines et peut aller jusqu'à 4.80 fois. Nous remarquons de plus que la structure des protéines est un facteur impactant les performances. En effet, le diagramme de la protéine 4V8W (123082 sites) est calculé en 317.10ms tandis qu'une protéine plus petite comme 3JC8 (107640 sites) requiert 727.00ms. Ces résultats illustrent la pertinence de notre processus de validation des composantes des cellules permettant ainsi le support efficace de distribution hétérogène. Dans un contexte de distribution uniforme, les deux méthodes proposent des temps de calcul plus importants que pour le traitement de protéines. Cette caractéristique est accentuée lorsque les échelles de rayon sont plus importantes. Cela est notamment explicable par la plus grande fréquence des configurations complexes présentant des arêtes composées de multiples segments, mais aussi d'arêtes déconnectées. Une fois encore, notre méthode permet un gain important en termes de temps de calcul allant jusqu'à permettre le calcul de grands ensembles en 7.82 fois moins de temps que Voronota. Ainsi, même dans ce contexte plus complexe, les différentes étapes présentées permettent un traitement efficace des caractéristiques des diagrammes d'Apollonius.

Enfin, nous présentons figure 5.8 les performances relatives de chaque étape de notre traitement. Nous remarquons ainsi que le traitement le plus complexe est la validation des arêtes. Dans le cas de certains échantillons du jeu de données de protéines, cette validation requiert l'exploration de grandes zones et donc la prise en compte d'un possible grand nombre de voisins. Cela est d'autant plus visible lorsque les arêtes elliptiques sont plus nombreuses, nécessitant ainsi la validation de très grandes zones, dans le cas des jeux de données de bruit blanc. Les autres

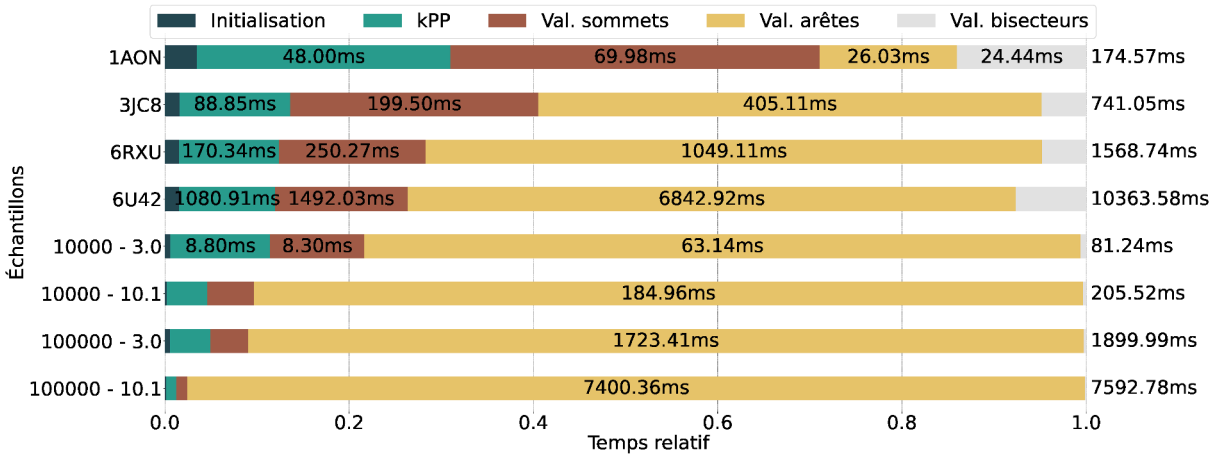


Figure 5.8 – Temps de calcul relatifs des différentes étapes du calcul du diagramme d'Apollonius avec notre méthode sur nos trois jeux de données avec un Intel I9-13900K et une NVIDIA RTX 4090.

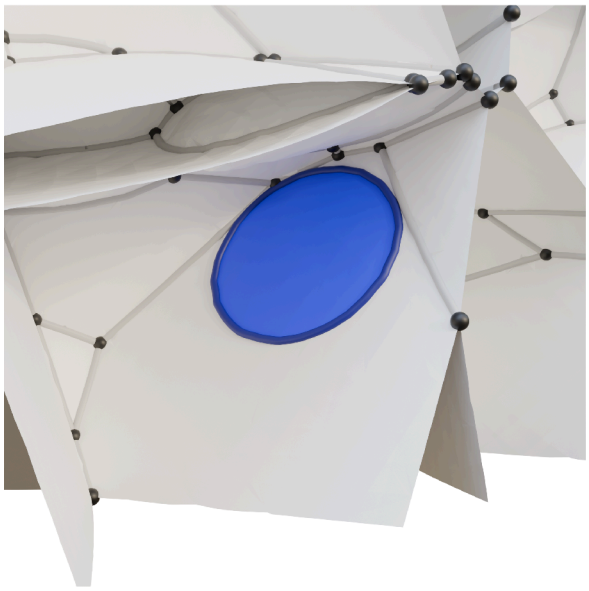
étapes sont cependant effectuées en une fraction du temps de calcul total et particulièrement la validation des faces qui est basée sur des zones plus précises.

Échantillon	*Bruit blanc*					
	Faces		**Arêtes**		**Sommets**	
	Voronota	Notre méth.	Voronota	Notre méth.	Voronota	Notre méth.
100-25-2-1	0	0	0	0	0	0
100-100-10-0.1	0	0	0	0	0	0
1000-50-2-1	8	0	4	0	15	15
1000-200-10-0.1	15	0	8	3	22	23
Échantillon	*Protéines*					
	Faces		**Arêtes**		**Sommets**	
	Voronota	Notre méth.	Voronota	Notre méth.	Voronota	Notre méth.
5ZCK	0	0	0	0	0	0
1AGA	0	0	0	0	2	2
3DIK	0	0	4	4	4	4
101M	6	6	69	69	127	127
1A3F	18	18	257	257	383	383
Échantillon	*Song et al. [Son+22]*					
	Faces		**Arêtes**		**Sommets**	
	Voronota	Notre méth.	Voronota	Notre méth.	Voronota	Notre méth.
Vis_I_10	0	0	0	0	0	0
Vis_II_5	2	0	1	0	0	0
Vis_III_5	6	0	3	0	0	0
Vis_IV_60	0	0	78	78	129	129
Vis_V_20	0	0	0	0	0	0
Vis_VI_6	0	0	0	0	4	4
Vis_VII_20	0	0	0	0	13	13
ANO1_0CONNECT	2	0	1	0	0	0
ANO2_0CONNECT	4	0	2	0	0	0
ANO3_3CONNECT	0	0	0	0	0	0
ANO4_4CONNECT	0	0	0	0	0	0
BALL_SMALL_1000	0	0	4	4	35	35
BALL_SMALL_2000	0	0	8	8	103	103
BALL_SMALL_3000	4	2	25	24	204	204
BALL_SMALL_4000	3	1	57	56	323	323
BALL_SMALL_5000	11	5	108	105	506	506
BALL_SMALL_6000	8	2	159	158	776	777
Ext_I_Congruent_300	0	0	4	4	165	167
Ext_II_Polysized_300	0	0	4	4	228	232

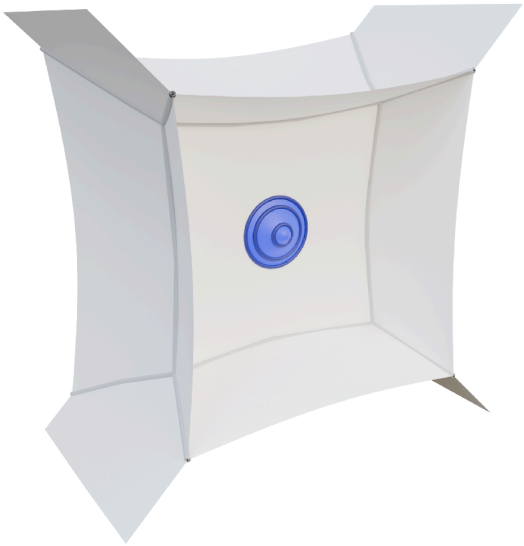
Table 5.3 – Validation des topologies extraites par Voronota [OV14] et notre méthode sur un sous-ensemble de nos jeux de données et de celui de Song et al. [Son+22]. Les valeurs représentent le nombre de composantes présentes dans la topologie obtenues avec la méthode de Du et al. [Du+22] et qui sont manquantes dans celles des méthodes comparées. Plus les valeurs sont réduites, plus les résultats sont bons.

5.4.2 Validité de la structure

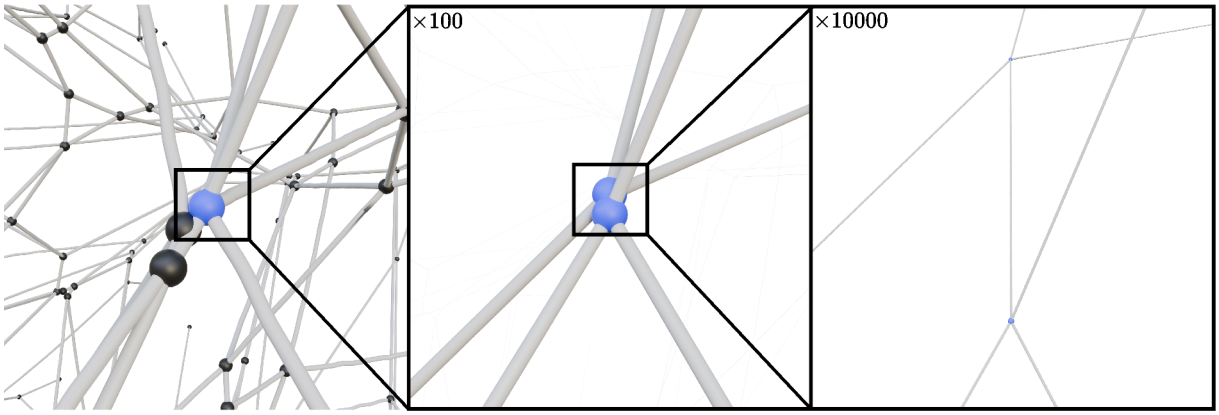
Nous étudions dans cette section la validité de la structure topologique extraite par notre méthode. Afin de produire des données de référence calculées de façon robuste, nous utilisons la méthode de Du et al. [Du+22]. Celle-ci garantit un calcul correct jusqu'à la précision des valeurs d'entrée de la structure combinatoire de réseaux de surface implicite et donc la construction du diagramme d'Apollonius. Sa complexité est cependant dépendante du nombre de sites ainsi que de la résolution de la discrétisation utilisée. Nous utilisons une grille de taille fixe composée de 1000000 de points [Du+22] permettant le calcul de diagrammes d'Apollonius d'ensembles allant



(a) Une arête elliptique et deux faces non calculés par Voronota (bruit blanc 1000-50-2-1)



(b) Trois arêtes elliptiques et six faces non calculés par Voronota (Vis_III_5)



(c) Deux sommets qui ne sont pas calculés par Voronota et notre méthode (PDB : 1AGA)

Figure 5.9 – Illustration du maillage produit par la méthode de Du et al. [Du+22] et des composantes non calculées par notre méthode ou Voronota (en bleu). (a, b) Voronota ne peut pas calculer les arêtes elliptiques fermées qui peuvent être imbriquées contrairement à notre méthode. (c) Voronota et notre méthode utilisant des nombres flottants sans prédicats robustes, elles ne peuvent pas calculer certaines configurations trop complexes.

jusqu'à 6000 sites sur un ordinateur de test muni de 64Go de mémoire vive.

Nous extrayons la structure combinatoire obtenue avec Voronota à partir des indices des sommets produits. En comparaison, nous enregistrons une composante produite avec notre méthode si son indice le plus petit est celui de la cellule calculée afin d'éviter toute redondance. Nous évaluons enfin la validité des structures des deux méthodes en testant si elles contiennent toutes les composantes présentes dans celle calculée par la méthode de Du et al. [Du+22].

Le nombre de composantes manquantes dans la topologie extraite par Voronota et notre méthode sont présentées tableau 5.3 sur un sous-ensemble de nos trois jeux de données, mais aussi de celui proposé par Song et al. [Son+22]. Celui-ci vise à tester la robustesse des méthodes de calcul de diagramme d'Apollonius et présente donc différentes configurations complexes. Nous remarquons tout d'abord que notre méthode propose en grande majorité une extraction plus robuste des faces et arêtes que Voronota. Cela est particulièrement visible sur les jeux de données de bruit blanc et de Song et al. [Son+22] qui proposent de nombreuses configurations comprenant des arêtes elliptiques. Cet aspect est une directe conséquence de l'absence de support des arêtes fermées par Voronota qui sont correctement calculées par notre méthode, comme illustré figure 5.9a et 5.9b. Les faces bornées par ces arêtes sont donc aussi absentes de la topologie extraite par Voronota. Nous remarquons de plus que dans la majorité des autres cas, les deux méthodes souffrent des mêmes erreurs de précision correspondantes à des cas nécessitant des prédicats plus robustes, comme illustré figure 5.9c. Voronota présente cependant une meilleure précision que notre méthode dans certains cas de calcul des sommets.

Notre méthode permet ainsi un support exhaustif des caractéristiques des diagrammes d'Apollonius en comparaison avec Voronota qui est limité aux arêtes ouvertes. Les deux méthodes présentent cependant des erreurs de précisions qui sont liées au manque de prédicats robustes et d'arithmétique exacte.

5.4.3 Exemple d'application

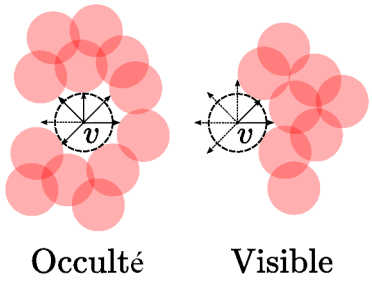
Nous proposons enfin un exemple d'application de notre méthode supportant l'illustration moléculaire. Lindow et al. ont montré que les arêtes et sommets d'Apollonius peuvent être utilisés afin de mettre en valeur les cavités et tunnels principaux des protéines [LBH11]. Celles-ci représentant l'une des cibles de la visualisation moléculaire, leur mise en valeur est ainsi particulièrement souhaitable, et notamment pour la communication scientifique.

Ainsi, nous présentons les étapes de ce traitement basé sur notre stratégie de calcul GPU. Celui-ci utilise les sommets d'Apollonius afin de positionner automatiquement des lumières dans les zones d'intérêt de la structure moléculaire. Il est de plus paramétrable permettant à l'utilisateur de configurer les zones sélectionnées.

Calcul des sommets Notre traitement débute par le calcul du diagramme d'Apollonius à partir de notre méthode. Nous extrayons ensuite les points des sommets de chaque cellule si l'indice du site de celle-ci est le plus petit du quadruplet qualifiant le sommet, permettant ainsi d'éviter les redondances.

Filtrage Afin de limiter les sommets utilisés à ceux qui représentent une cavité dans la structure moléculaire, nous les filtrons à partir de deux prédicats. Tout d'abord, Lindow et al. ont remarqué que les sommets positionnés dans des cavités moléculaires peuvent être différenciés

à partir d'une stratégie similaire au processus d'occlusion ambiante [ZIK98 ; LBH11]. Ainsi, nous calculons un niveau d'occlusion de chaque sommet en lançant un nombre configurable de rayons de façon uniforme sur la sphère. Nous permettons ainsi à l'utilisateur de filtrer les sommets dont le ratio de rayon ayant touché un atome est trop faible pour être localisé à l'intérieur d'une cavité. De plus, nous donnons la possibilité de limiter les sommets en fonction de leur distance à leurs sites permettant ainsi de limiter leurs nombres.



Notre implémentation GPU est développée en CUDA et utilise l'API OptiX [Par+10] pour le calcul du ratio d'occlusion des sommets.

Un rendu de plusieurs molécules est présenté figure 5.10. La structure des protéines, de différentes tailles, est accentuée grâce à ce traitement et permet de suggérer plusieurs éléments importants. Notamment, il est possible de visualiser les cavités de certaines d'entre elles qui ne sont pas directement visibles depuis l'extérieur sans éclairage. Nous présentons les performances de calcul des différentes étapes du traitement tableau 5.4 à des fins purement informatives. La majorité du temps de traitement est dédiée au calcul du diagramme d'Apollonius. Les deux autres étapes sont effectuées en une fraction du temps total, et bénéficient de l'implémentation GPU.

5.4.4 Limites

Même si notre méthode permet une construction exhaustive des diagrammes d'Apollonius de façon performante en comparaison avec l'état de l'art, elle est limitée par certaines de ses caractéristiques.

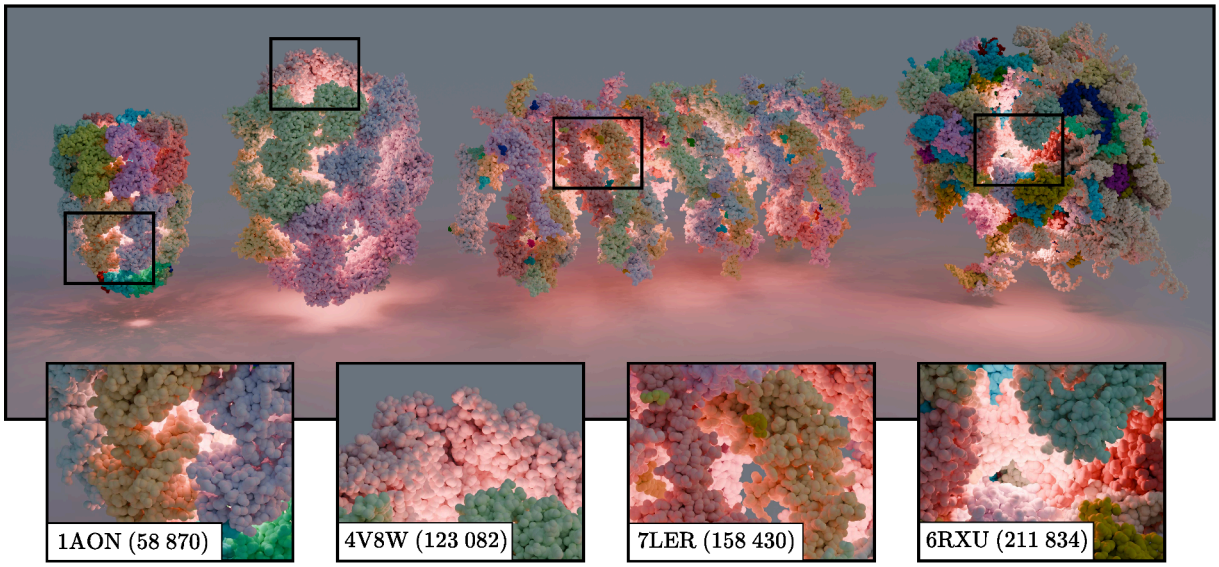


Figure 5.10 – Illustration de protéines avec la mise en valeur de cavités basée sur les sommets d'Apollonius. Les lumières sont positionnées sur les points des sommets calculés et filtrés permettant d'accentuer les cavités.

PDB	#Sites	#Sommets	*Étapes*		
			Calcul des sommets	**Ratio d'occlusion**	**Filtrage**
7P3W	37149	253431	247.53	5.85	0.13
4V8W	123082	838101	330.92	18.43	0.20
7LER	158430	1077978	921.70	23.11	0.33
6RXU	211834	1450138	1450.33	33.0	0.25
4V6X	237685	1646761	1765.53	37.55	0.25

Table 5.4 – Performances de filtrage des sommets d'Apollonius pour l'illustration moléculaire avec un Intel I9-13900K et une NVIDIA RTX 4090. Les temps proposés sont donnés en millisecondes et représentent la moyenne de 100 itérations précédées de 10 itérations qui ne sont pas prises en compte.

Tout d'abord, notre méthode est compatible avec la construction de cellules dans des contextes hétérogènes, notamment grâce au processus de calcul coopératif par warp utilisé (section 5.3.1). Cependant, il est possible de rencontrer des cellules dont la topologie requiert plus de mémoire que celle qui est allouée dans des cas très dégénérés. Ainsi, un processus CPU dédié est couramment envisagé [Ray+19 ; Bas+21] pour les cellules nécessitant plus de mémoire.

De plus, notre méthode permet le calcul d'une topologie exhaustive, compatible avec de nombreuses applications. Notre implémentation ne dispose cependant pas d'opérations basées sur de l'arithmétique exacte qui permettrait de hauts niveaux de précision ainsi que le calcul de configurations dégénérées. Notre stratégie peut cependant être adaptée en utilisant des prédicats visant ce type de cas [DKT17].

Enfin, nous avons proposé différents traitements permettant le calcul efficace de distributions hétérogènes. Cela est notamment rendu possible grâce aux zones de validation nécessaires à la construction exhaustive du diagramme. Cependant, notre traitement visant les arêtes elliptiques peut représenter l'un des facteurs ralentissants de notre méthode. Il représente ainsi une possible amélioration pour des applications visant un très haut niveau de performances.

5.5 Conclusion

Dans ce chapitre, nous avons présenté notre méthode de construction de diagramme d'Apollonius sur GPU. Celle-ci se base sur une construction par cellule qui permet une parallélisation efficace tout en supportant les caractéristiques particulières de ces diagrammes. En effet, le processus de mise à jour topologique proposé permet le calcul exhaustif de leurs composantes tout en étant compatible avec une structure de données légère permettant une exécution sur GPU. Nous avons de plus défini plusieurs adaptations visant une exploration spatiale performante pour le support efficace de distributions homogènes, mais aussi hétérogènes.

Notre méthode permet le calcul de diagrammes d'Apollonius avec des distributions variées, évaluée sans changement de configuration. Ainsi, ses performances sont meilleures que la méthode la plus rapide de l'état de l'art tout en proposant une construction exhaustive, ce qui n'était pas possible avant. Cette dernière caractéristique est notable et est une conséquence directe des fondations théoriques présentées dans le chapitre 4 et de notre méthode de mise à jour topologique d'une cellule présentée section 5.2.3. Ainsi, elle permet la construction de diagrammes de grandes tailles, sans a priori sur la distribution spatiale des sites. Cette caractéristique rend

notamment la géométrie produite par notre méthode utilisable dans des contextes plus variés que si elle était limitée aux arêtes fermées, comme pour des stratégies de transport optimal où la cellule doit souvent être complètement caractérisée.

Enfin, nous avons montré un cas d'utilisation de notre méthode dans un contexte d'illustration moléculaire. Celui-ci permet la mise en valeur d'une protéine de façon automatique et configurable par l'utilisateur. Même si nous avons choisi cet exemple, notre méthode est compatible avec d'autres traitements. Elle pourrait notamment être à la base de l'extraction des formes de cavités moléculaires [Lin+13] ou de construction de la SES [MJK16].

Notre stratégie s'inscrit dans une directe continuité de notre méthode de calcul de la SES pour de grandes protéines. Elle permet ainsi le calcul du diagramme d'ensemble de sites de grande taille à la base de l'analyse structurelle de protéines. Elle est cependant moins compatible avec des séquences d'animations que la stratégie présentée chapitre 3. Ainsi, elles peuvent être utilisées conjointement afin de proposer des applications visant une rapidité d'exécution sur ce type de données.

Chapitre 6

Conclusion

Sommaire

L'analyse structurelle de protéines représente un outil important pour leur étude en biochimie. Les performances des algorithmes à la base de celle-ci sont cependant contraintes par les tailles des structures cibles qui ont augmenté grâce aux progrès des stratégies d'acquisition. Dans cette thèse, nous avons présenté différentes méthodes visant la construction performante de géométrie pour l'analyse structurelle de grands complexes moléculaires tout en permettant le calcul de données exhaustives. Les stratégies développées ont visé à diminuer les compromis effectués afin de bénéficier d'un haut niveau de fonctionnalité, mais aussi d'une plus grande applicabilité.

6.1 Contributions

Nous avons proposé une stratégie de calcul analytique de la Surface Exclue au Solvant (SES) de grandes protéines (chapitre 3). Cette surface propose différentes fonctionnalités importantes pour la visualisation et l'analyse moléculaire, mais reste complexe à construire. Notre méthode utilise une analyse mathématique de la SES qui permet la définition d'une structure de données légère et compatible avec les limitations mémoires des GPUs. Nous avons de plus introduit différentes étapes nécessaires au calcul rapide de la surface complète, à la différence de la méthode précédente la plus rapide, mais limitée au calcul et l'affichage de l'extérieur de la surface uniquement. Notre stratégie permet une construction efficace de la surface complète ainsi qu'une réduction significative de la consommation mémoire en comparaison avec les méthodes de l'état de l'art. Ainsi, elle est compatible avec de plus grandes protéines, mais aussi avec des GPUs disposant de moins de mémoire. À partir de ces caractéristiques, elle pourra permettre la démocratisation de l'utilisation d'algorithme de construction analytique sur GPU dans les logiciels de visualisation moléculaire.

Nous nous sommes ensuite intéressés aux diagrammes d'Apollonius qui permettent le développement de nombreuses méthodes soutenant l'analyse de structures moléculaires. Les caractéristiques de leurs composantes étant cependant plus complexes que celles des diagrammes affines comme les diagrammes de Voronoï ou de Puissance, nous avons proposé une caractérisation et une paramétrisation complète du diagramme d'Apollonius dans $\mathbb{R}^2$ et $\mathbb{R}^3$ (chapitre 4). Celle-ci permet l'étude de leurs propriétés à travers leurs équations analytiques. Sur cette base, nous avons décrit un algorithme naïf permettant le maillage de la géométrie du diagramme de façon exhaustive. Notre étude représente donc des fondations pour le développement de stratégies plus complexes visant la construction efficace de diagrammes de plus grandes tailles.

Enfin, nous avons proposé une méthode de construction exhaustive et efficace de diagrammes d'Apollonius sur GPU (chapitre 5). Celle-ci vise le calcul en parallèle de toutes les composantes de ces diagrammes à partir de leurs cellules. Elle est basée sur une succession d'étapes et notamment un processus de mise à jour topologique permettant la modification itérative de la géométrie des cellules. De plus, afin de couvrir des usages larges de ces diagrammes, nous avons proposé différentes caractéristiques additionnelles pour le support de distributions spatiales de sites homogènes, mais aussi hétérogènes. L'évaluation de notre méthode montre qu'elle permet une construction exhaustive tout en proposant de meilleures performances que la stratégie la plus

rapide de l'état de l'art, limitée aux composantes connectées. Ainsi, elle pourra être considérée pour l'analyse de structure moléculaire, notamment pour la découverte de cavités moléculaires, mais aussi pour des applications plus générales nécessitant la construction de diagrammes d'Apollonius.

6.2 Perspectives

Les travaux présentés dans cette thèse visent le soutien de l'analyse structurelle de protéines à travers la visualisation et l'illustration. Nous avons proposé plusieurs méthodes permettant de hauts niveaux de performance sans compromis sur la qualité de la géométrie construite. Sur ces bases, de potentielles améliorations et travaux futurs peuvent être envisagés afin de bénéficier de fonctionnalités appropriées à des contextes particuliers. Nous discutons tout d'abord des perspectives liées aux caractéristiques des algorithmes présentés (section 6.2.1) puis de potentiels cas d'utilisation basés sur les données géométriques calculées (section 6.2.2).

6.2.1 Adaptations et améliorations

Les méthodes présentées sont orientées vers un calcul exhaustif et performant de la géométrie. Cependant, différents ajustements et améliorations peuvent être souhaités pour certains cas d'application.

Affichage de la SES Notre méthode de construction de la SES permet un affichage rapide après le calcul de sa géométrie. Cependant, il pourrait être intéressant de proposer un calcul interactif à la volée d'une sous-partie de la géométrie permettant de réduire les temps de traitements à ce qui est nécessaire à l'affichage d'une image. Les méthodes existantes de ce type requièrent cependant une recherche itérative des intersections [PV12] ou présentent des instabilités visuelles [KDE10]. Cela étant principalement causé par la caractérisation de la SES en fonction de la SAS, il pourrait être souhaitable de définir la SES plus directement.

Calcul du diagramme d'Apollonius La méthode de calcul de diagramme d'Apollonius que nous avons proposée permet une construction efficace tout en produisant une topologie complète. Elle est cependant contrainte par la quantité de mémoire nécessaire au stockage de chaque cellule, malgré notre structure de données légère. Il serait ainsi intéressant de limiter cette consommation mémoire à partir de processus plus dynamique, potentiellement basé sur notre mise à jour topologique, et permettant d'adapter la quantité de mémoire en fonction des besoins par cellule. Cela permettrait ainsi de limiter les allocations mémoire non utilisées.

Prédicats robustes sur GPU Même si notre méthode de construction du diagramme d'Apollonius permet une construction exhaustive d'une cellule, elle est limitée à la précision des nombres utilisés par l'implémentation. Ce type de limitation est courant dans des implémentations GPUs qui ne sont pas compatibles avec une exécution performante de prédicats robustes. Ces derniers nécessitent en effet une exécution séquentielle dans de rares cas impliquant des divergences d'exécutions. Notre méthode pourrait alors bénéficier de processus et prédicats satisfaisant ces contraintes avec un impact sur les performances limité.

Diagrammes d'Apollonius dans $\mathbb{R}^d$ Le calcul de diagramme de Voronoï dans $\mathbb{R}^2$ et $\mathbb{R}^3$ a représenté le cœur de nombreuses études. Leur construction dans de plus grandes dimensions, pour la science des données, représente cependant toujours un enjeu, notamment en raison de la complexité de la structure topologique croissante avec la dimension. Notre méthode étant basée sur une mise à jour séquentielle de la topologie, elle n'est pas compatible avec le calcul efficace dans de grandes dimensions. Le développement de méthodes n'étant pas directement dépendantes de la topologie complète d'une cellule pourrait alors permettre des applications plus générales.

6.2.2 Utilisation de la géométrie pour la biochimie

Les méthodes proposées dans cette thèse visent la construction de géométrie pour l'analyse structurelle de molécules. Il est possible d'envisager la définition d'algorithmes utilisant les données produites afin d'étudier les caractéristiques de protéines. Notamment, il est courant d'utiliser des descripteurs de ce type dans des processus d'optimisations afin de qualifier les structures cibles. Nous proposons ainsi quelques utilisations potentielles.

Géométrie moléculaire La structure extraite par notre méthode de calcul de la SES pourrait être utilisée afin d'obtenir la fonction distance signée proposée par Quan et Stamm [QS16]. Elle pourrait alors être à l'origine de méthodes utilisant cette surface pour sa caractérisation implicite des interactions moléculaires, notamment dans un contexte de simulation ou de prédiction de complexe. Notre stratégie ne calculant pas explicitement certaines parties de la géométrie comme les contours des patchs convexes, il serait cependant nécessaire d'adapter notre traitement afin de bénéficier d'un calcul performant des primitives souhaitées.

Diagrammes d'Apollonius pour l'analyse moléculaire De nombreux travaux ont montré que les diagrammes d'Apollonius peuvent être à la base de l'analyse structurelle de protéines notamment pour la découverte de cavités [Kro+16]. Ces caractéristiques les rendant particulièrement intéressants pour la visualisation et l'illustration moléculaire, ils pourraient aussi servir de fondation pour des méthodes de prédiction de complexes moléculaires. Ces derniers sont couramment évalués à partir de fonction de score prédisant la stabilité du système. La structure topologique des diagrammes d'Apollonius pourrait alors permettre la définition de descripteurs encodant la conformation d'une protéine, à la différence de sa structure chimique, comme cela peut déjà être réalisé avec des diagrammes de Voronoï affine [Ber+07]. De plus, leur caractérisation de l'espace tangent définit des zones d'intérêts, utilisées pour la mise en valeur de protéines (chapitre 5). Ces dernières représentant des caractéristiques physico-chimiques, elles définissent l'espace auquel il est alors possible d'attribuer des valeurs sémantiques. Ainsi, cette structure pourrait permettre de guider des processus d'optimisations à partir de ces informations additionnelles.

Annexe A

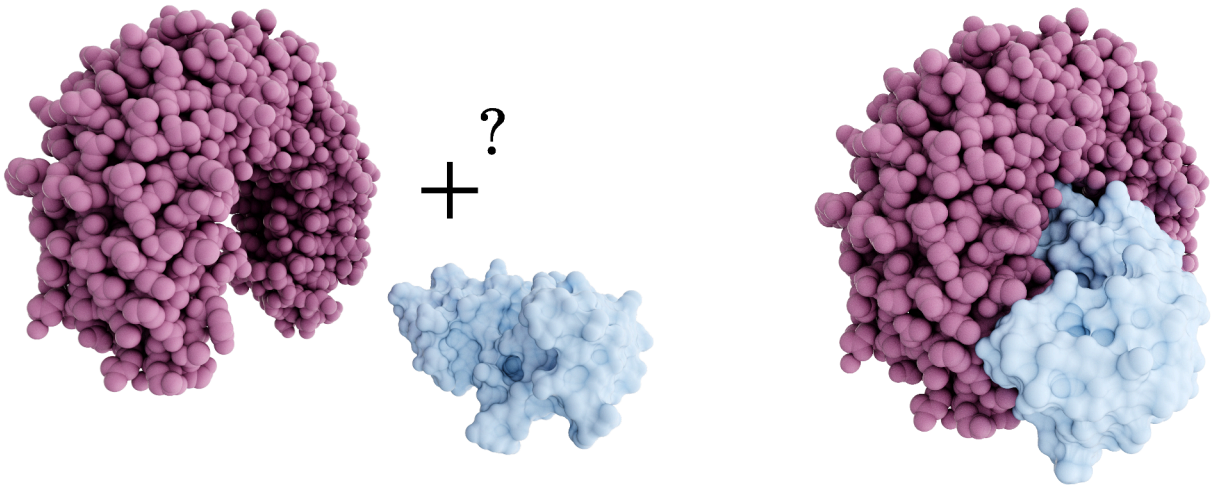
UDock2 : Amarrage moléculaire multi-corps et interactif

Sommaire

De nombreux types de simulations sont couramment utilisés pour l'étude de systèmes moléculaires. Une approche notable est celle de l'*amarrage moléculaire* aussi appelé *docking*, illustré figure A.1. Au cours de celui-ci, on va chercher à associer deux molécules afin de former un complexe moléculaire neutre énergétiquement. Ce type de simulation est notamment utilisé pour prédire les interactions entre une molécule (une protéine, un ligand ou un ion) et une protéine, par exemple pour la conception de médicaments.

De nombreux logiciels [Wan+16] proposent d'effectuer ce type de simulation automatiquement à partir de différents paramètres. Cependant, ils ne permettent généralement pas à l'utilisateur de manipuler facilement les systèmes moléculaires afin de guider la simulation vers de potentielles conformations intéressantes.

D'autres méthodes ont ainsi été introduites pour offrir un haut niveau de contrôle à l'utilisateur à travers des stratégies intuitives et interactives. Notamment, de nombreuses applications basées sur des dispositifs haptiques ont été développées afin de permettre à l'utilisateur de manipuler des corps moléculaires tout en proposant un retour sensoriel du champ de force [DMR07 ; Zon+08 ; Fér+09 ; Mat+19]. Ce type de matériel est cependant coûteux et moins couramment utilisé. D'autres logiciels proposent d'utiliser des environnements plus classiques à travers des stratégies novatrices de contrôle utilisateur [LDC05 ; Lev+14 ; Iak+20]. Ces dernières sont ainsi compatibles avec des configurations plus diverses tout en offrant la possibilité de manipuler les entités souhaitées.



(a) Deux protéines à partir desquelles on souhaite créer un complexe moléculaire stable

(b) Complexe moléculaire stable

Figure A.1 – Illustration d'un amarrage moléculaire. On cherche une conformation dans laquelle l'énergie potentielle du système est la plus faible (PDB : 1DFJ).

L'amarrage moléculaire interactif représente cependant toujours un enjeu en termes d'ergonomie, rapidité de calcul et pertinence des résultats. Ainsi, dans cette section, nous présentons la deuxième version du logiciel UDock, proposé initialement par Levieux et al. [Lev+14]. Nous proposons notamment le support de l'amarrage de plusieurs corps moléculaires à la fois, de nouveaux contrôles utilisateurs et une interface visant à améliorer son ergonomie, ainsi qu'une expérience ludique pour la popularisation de ce type d'application. UDock2 étant prévu pour un public très large, ce logiciel n'utilise pas de stratégie d'accélération GPU pouvant nécessiter des configurations particulières, mais propose néanmoins une rapidité d'exécution suffisante à l'exploration interactive. Ce projet a fait l'objet d'une publication dans le journal *Bioinformatics* [Pla+23a].

Dans ce chapitre, nous présentons tout d'abord la fonction d'évaluation utilisée ainsi que son optimisation (annexe A.1), puis les composants de l'interface utilisateur (annexe A.2). Enfin nous concluons ce chapitre (annexe A.3).

A.1 Évaluation de l'énergie moléculaire

L'énergie potentielle d'un système moléculaire est décrite par ses interactions interatomiques qui dépendent de sa conformation, mais aussi de la charge électrique de ses atomes. Elle est associée à un modèle servant à la simuler numériquement. Ainsi, l'évaluation d'une conformation donnée peut être effectuée à partir d'une fonction de score basée sur un champ de force [Men+11]. Celle-ci approxime l'énergie d'interaction afin de quantifier la stabilité d'un complexe. Dans cette section, nous présentons la fonction de score d'UDock2 et son optimisation afin de trouver automatiquement des conformations d'intérêt.

Les fonctions de score sont de différents types, et de nombreuses ont été proposées pour répondre à des besoins d'applications spécifiques effectuant un compromis entre capacité de calcul et précision [Men+11]. Nous réutilisons ainsi celle d'UDock [Lev+14] qui est composée de deux fonctions permettant de modéliser les interactions, illustrées figure A.2.

Nous utilisons le modèle de potentiel de Lennard-Jones pour caractériser les forces d'attraction et répulsion de courtes distances entre deux atomes. Soient a_i et a_j, deux atomes dont on souhaite calculer le potentiel, et $\epsilon = \sqrt{\epsilon_i \epsilon_j}$ la pondération empirique de Berthelot de leurs interactions. Leur potentiel $E_p\,(i,j)$ est alors défini par

$$E_p\,(a_i, a_j) = 3\epsilon \left(\frac{r_i + r_j}{||c_i - c_j||} \right)^8 - 4\epsilon \left(\frac{r_i + r_j}{||c_i - c_j||} \right)^6 ,$$

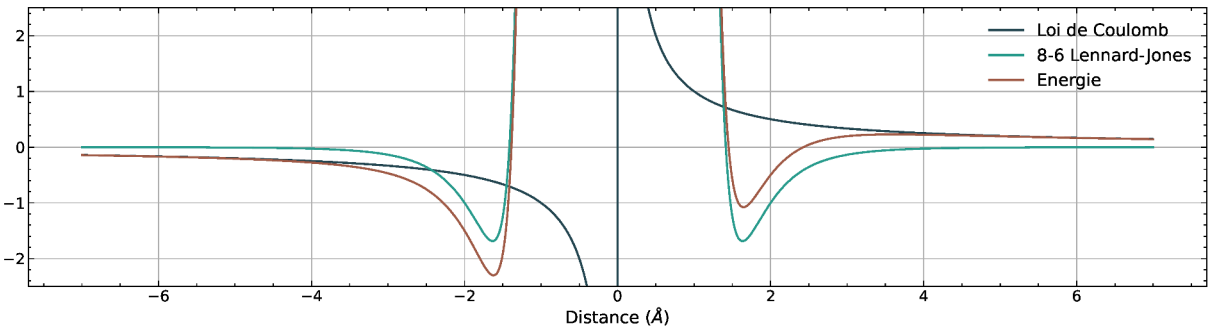


Figure A.2 – Illustration du potentiel moléculaire utilisé pour UDock2.

avec c et r les centre et rayon d'un atome. Nous utilisons de plus la loi de Coulomb afin de représenter les interactions électrostatiques de plus grandes distances qui est donnée par

$$F\left(a_i, a_j\right) = 332.0522 \frac{c_i c_j}{\mathcal{E}_0 ||c_i - c_j||},$$

où 332.0522 est un facteur de conversion en kcal/mol donné par AMBER23 [Cas+23], et $\mathcal{E}_0 = 20$, une constante diélectrique [Men+11 ; Lev+14]. La fonction de score $E\left(i, j\right)$ est alors définie par

$$E\left(a_i, a_j\right) = E_p\left(a_i, a_j\right) + F\left(a_i, a_j\right). \tag{A.1}$$

Le score d'une scène complète peut ensuite être calculé en sommant l'énergie de toutes les paires d'atomes

$$E = \sum_{i \neq j} E\left(a_i, a_j\right).$$

Cette définition implique cependant une complexité quadratique, incompatible avec le traitement rapide de protéines. En effet, afin de soutenir l'exploration de conformation, nous souhaitons proposer une indication interactive du score. Ainsi, nous limitons ce calcul pour chaque atome à ses voisins localisés dans un rayon r_{max}. L'énergie peut alors être exprimée avec

$$E = \sum_{a_i} E\left(a_i, a_j\right), \forall a_j \text{ tel que } ||c_i - c_j|| < r_{max}.$$

Dans les faits, nous utilisons $r_{max} = 12$Å comme classiquement utilisé pour des simulations moléculaires [SKV12 ; Lag+18].

Notre traitement se base ainsi sur une implémentation CPU visant un support peu limité par le matériel visé. Il utilise de plus une structure accélératrice pour effectuer cette requête spatiale basée sur un rayon. Nous nous servons ainsi, comme dans le chapitre 3, d'une grille accélératrice adaptée à ce type de requête lorsque les rayons des sphères sont similaires [Hoe14].

Sur la base de ces caractéristiques, nous utilisons de plus un processus d'optimisation de la fonction de score. Celui-ci commence par calculer aléatoirement une transformation spatiale pondérée de la protéine sélectionnée. Si, une fois qu'elle a été appliquée, le score diminue et donc s'améliore, nous la conservons pour la prochaine itération. Sinon, nous générons une nouvelle transformation. Cette stratégie débutant à partir de la conformation initiale choisie par l'utilisateur, elle est configurable et permet de raffiner automatique une configuration spatiale.

À partir de ces développements, nous définissons l'interface utilisateur d'UDock2 permettant une manipulation ergonomique des entités manipulées et guident l'exploration des conformations.

A.2 Interface utilisateur

L'interface utilisateur proposée dans UDock2, illustrée figure A.3, est basée sur le calcul de l'énergie présenté dans la section précédente. Nous détaillons ainsi dans cette section certaines des caractéristiques visant à proposer un contrôle ergonomique des configurations testées.

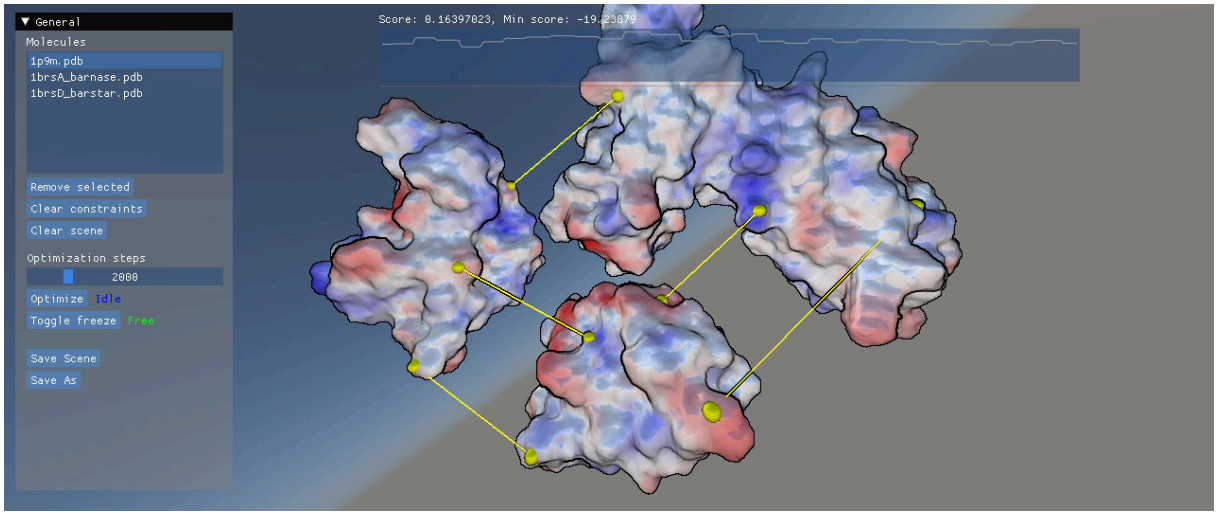


Figure A.3 – Illustration de l'interface d'UDock2.

A.2.1 Représentation moléculaire

En tant que représentation implicite des interactions (chapitre 2), la SES est particulièrement adaptée à la visualisation moléculaire au cours d'une simulation d'amarrage. Ainsi, elle est utilisée dans UDock2 pour toutes les molécules chargées par l'utilisateur.

Puisque nous visons un public très large, cette surface ne peut pas être calculée à partir d'une méthode visant une exécution sur GPU. Ainsi, nous nous basons sur un calcul approximé de la SES permettant un compromis entre temps de calcul et qualité du maillage extrait. Nous générons tout d'abord un champ de distance signé dans une grille de taille empirique $\epsilon \frac{1}{3} \sum_{a=1}^{3} s_a$, avec s, la taille de la boite englobante, et ϵ, un facteur permettant de configurer le niveau de précision. La surface est ensuite maillée à partir de l'algorithme des *Marching Cubes* [LC87].

Une fois la surface générée, nous l'utilisons afin de présenter des informations spatiales à l'utilisateur. Notamment, la charge électrique de la protéine intervenant dans le calcul de l'énergie de la scène, sa représentation permet de guider les manipulations de la conformation. Ainsi, elle est calculée pour chaque sommet à partir d'un potentiel électrostatique simplifié. La charge e_{v_i} d'un sommet v_i est alors obtenue avec

$$e_{v_i} \coloneqq \sum_{\forall a_j \in \mathcal{N}} \frac{e_j}{||c_j - v_i + r_p n_i||^2},$$

où $\mathcal{N} \coloneqq \left\{ a \mid ||c_a - v_i|| \leq 5\text{Å} \right\}$ représente les atomes dans le voisinage de v_i, r_p le rayon du probe, n_i la normale de v_i et e_j la charge de l'atome a_j. Les charges calculées sont ensuite moyennées à partir de celles des sommets localisés sur les triangles voisins permettant de limiter les artefacts liés à l'échantillonnage.

A.2.2 Contrôles

Afin de proposer une exploration intuitive des conformations testées, UDock2 propose deux caméras visant des types d'exploration différents, illustrées figure A.4. La première, notée $\mathcal{G}$, permet un contrôle standard à partir de l'orientation et la position. La seconde, notée $\mathcal{L}$,

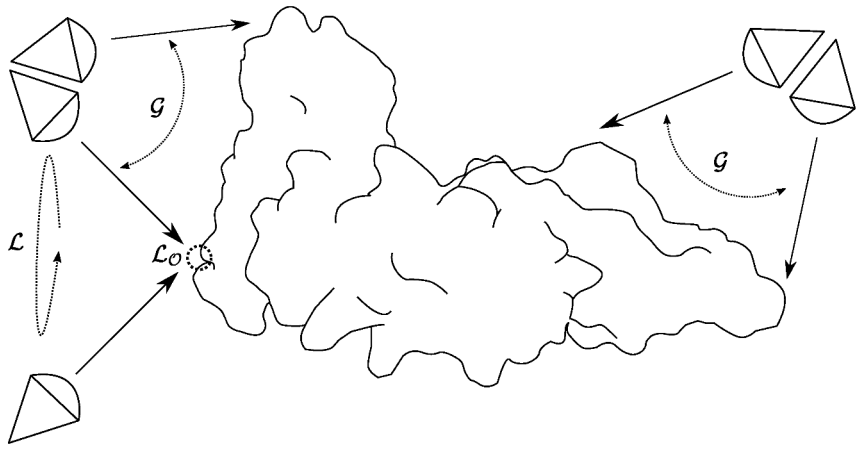


Figure A.4 – Illustration des différentes caméras proposées dans UDock2.

propose à l'utilisateur de tourner autour d'un point d'intérêt $\mathcal{L}_\mathcal{O}$. À la différence de caméras plus classiques, l'utilisateur peut modifier rapidement $\mathcal{L}_\mathcal{O}$ ce qui rend fluide la visualisation de multiples emplacements géométriques intéressants. Ainsi, tandis que la première permet une exploration générale de la scène, la seconde est orientée vers la visualisation proche des caractéristiques spatiales de la protéine.

En plus des caméras, UDock2 permet de contrôler les positions des molécules à partir d'un système de liens visant à approcher des zones d'intérêts de plusieurs corps moléculaires. L'utilisateur place ainsi une ancre sur un point de la géométrie moléculaire qu'il souhaite approcher d'une autre molécule. Une fois que des ancres sont positionnées sur deux protéines différentes, un lien est créé permettant d'appliquer une force d'attraction. Une fois qu'une conformation intéressante est fixée par l'utilisateur, il peut la raffiner à partir de l'optimisation automatique décrite dans la section précédente.

Enfin, l'équation du calcul d'énergie présentée requiert que la distance entre deux atomes soit supérieure à 0 (A.1). Afin d'éviter ce type de situation pouvant compliquer l'exploration des conformations par l'utilisateur, nous utilisons la SES comme enveloppe rigide des molécules, ce qui empêche que deux corps moléculaires s'intersectent et facilite l'interaction. Pour cela, nous utilisons la bibliothèque *Bullets* [CB21] qui permet de bonnes performances dans ce contexte.

A.2.3 Affichage

L'interface d'UDock2 est composée d'éléments visant la configuration de la scène, de l'optimisation, mais aussi l'affichage en temps réel du score. En plus de ces aspects, les corps moléculaires sont placés dans un environnement permettant une exploration intuitive et composé d'un ciel réaliste [HW12]. Celui-ci permet ainsi de donner une indication explicite de l'orientation de la caméra grâce, notamment, à l'affichage d'une ligne d'horizon.

Les molécules sont rendues à partir d'un matériau dont les reflets spéculaires sont désactivables par l'utilisateur. Leurs surfaces décrivent leur potentiel qui est identifié à partir de couleurs caractéristiques illustrant son signe et sa valeur. Par convention, la zone est affichée en rouge s'il est négatif, en bleu si positif, et blanc si neutre. Enfin, les contours des molécules sont mis en valeur à partir d'un filtre laplacien permettant d'accentuer les zones d'intérêts et faciliter la visualisation.

Enfin, nous proposons une scène additionnelle visant la vulgarisation. Celle-ci est composée d'un environnement spatial dans lequel l'utilisateur peut contrôler un vaisseau. Cette représentation est dotée d'une enveloppe rigide permettant d'entrer en collision avec les autres corps moléculaires pour une interaction plus intuitive. En plus des ancres et liens, l'utilisateur peut aussi déplacer les entités en projetant des sphères munies d'une enveloppe rigide poussant les corps moléculaires. Ces derniers ajouts permettent une utilisation ludique d'UDock2 visant la vulgarisation scientifique de ce type de simulation.

A.3 Conclusion

Nous avons présenté UDock2, la nouvelle version d'UDock [Lev+14]. Dans cette application, nous proposons différentes nouvelles fonctionnalités visant à améliorer son ergonomie, mais aussi ses performances. Ainsi, elle intervient préalablement à des simulations de plus grandes précisions permises par d'autres logiciels et qui pourront débuter avec la conformation optimisée et configurée avec UDock2.

Bibliographie

Références

[AB02] J. L. Atwood et L. J. Barbour. « Molecular Graphics : From Science to Art ». In : *Crystal Growth & Design* 3.1 (déc. 2002), p. 3-8 (Cité page : 8).

[AE96] N. Akkiraju et H. Edelsbrunner. « Triangulating the surface of a molecule ». In : *Discrete Applied Mathematics* 71.1–3 (déc. 1996), p. 5-22 (Cité page : 27).

[Aur87] F. Aurenhammer. « Power Diagrams : Properties, Algorithms and Applications ». In : *SIAM Journal on Computing* 16.1 (fév. 1987), p. 78-96 (Cité pages : 32, 34, 67, 68, 81).

[Aur91] F. Aurenhammer. « Voronoi diagrams—a survey of a fundamental geometric data structure ». In : *ACM Computing Surveys* 23.3 (sept. 1991), p. 345-405 (Cité pages : 30, 68).

[Bas+21] J. Basselin, L. Alonso, N. Ray, D. Sokolov, S. Lefebvre et B. Lévy. « Restricted Power Diagrams on the GPU ». In : *Computer Graphics Forum* 40.2 (mai 2021), p. 1-12 (Cité pages : 38, 85, 86, 96, 97, 104).

[BD23] N. Bonneel et J. Digne. « A survey of Optimal Transport for Computer Graphics and Computer Vision ». In : *Computer Graphics Forum* 42.2 (mai 2023), p. 439-460 (Cité page : 32).

[Ber+00] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov et P. E. Bourne. « The Protein Data Bank ». In : *Nucleic Acids Research* 28.1 (jan. 2000), p. 235-242 (Cité page : 9).

[Ber+07] J. Bernauer, J. Azé, J. Janin et A. Poupon. « A new protein–protein docking scoring function based on interface residue properties ». In : *Bioinformatics* 23.5 (jan. 2007), p. 555-562 (Cité page : 110).

[BK03] J.-D. Boissonnat et M. I. Karavelas. « On the Combinatorial Complexity of Euclidean Voronoi Cells and Convex Hulls of D-Dimensional Spheres ». In : *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '03. Baltimore, Maryland : Society for Industrial et Applied Mathematics, 2003, p. 305-312 (Cité page : 89).

[Bli82] J. F. Blinn. « A Generalization of Algebraic Surface Drawing ». In : *ACM Transactions on Graphics* 1.3 (juill. 1982), p. 235-256 (Cité page : 25).

[Bru19] S. Bruckner. « Dynamic Visibility-Driven Molecular Surfaces ». en. In : *Wiley* 38.2 (mai 2019), p. 317-329 (Cité page : 25).

[BWY06] J.-D. Boissonnat, C. Wormser et M. Yvinec. « Curved Voronoi Diagrams ». In : *Effective Computational Geometry for Curves and Surfaces*. Springer Berlin Heidelberg, 2006, p. 67-116 (Cité pages : 34, 67, 81).

[Cas+23] D. Case, H. Aktulga, K. Belfon, I. Ben-Shalom, J. Berryman, S. Brozell, D. Cerutti, T. Cheatham III, G. Cisneros, V. Cruzeiro, T. Darden, N. Forouzesh, G. Giambaşu, T. Giese, M. Gilson, H. Gohlke, A. Goetz, J. Harris, S. Izadi, S. Izmailov, K. Kasavajhala, M. Kaymak, E. King, A. Kovalenko, T. Kurtzman, T. Lee, P. Li, C. Lin, J. Liu, T. Luchko, R. Luo, M. Machado, V. Man, M. Manathunga, K. Merz, Y. Miao, O. Mikhailovskii, G. Monard, H. Nguyen, K. O'Hearn, A. Onufriev, F. Pan, S. Pantano, R. Qi, A. Rahnamoun, D. Roe, A. Roitberg, C. Sagui, S. Schott-Verdugo, A. Shajan, J. Shen, C. Simmerling, N. Skrynnikov, J. Smith, J. Swails, R. Walker, J. Wang, J. Wang, H. Wei, X. Wu, Y. Wu, Y. Xiong, Y. Xue, D. York, S. Zhao, Q. Zhu et P. Kollman. *AMBER 23*. Sous la dir. d'U. of California. San Francisco, 2023 (Cité page : 114).

[CB21] E. Coumans et Y. Bai. *PyBullet, a Python module for physics simulation for games, robotics and machine learning*. `http://pybullet.org`. 2016–2021 (Cité page : 116).

[CCW06] T. Can, C.-I. Chen et Y.-F. Wang. « Efficient molecular surface generation using level-set methods ». In : *Journal of Molecular Graphics and Modelling* 25.4 (déc. 2006), p. 442-454 (Cité page : 23).

[CLM08] M. Chavent, B. Levy et B. Maigret. « MetaMol : High-quality visualization of molecular skin surface ». In : *Journal of Molecular Graphics and Modelling* 27.2 (sept. 2008), p. 209-216 (Cité page : 25).

[Col] C. Collange. *Warp-synchronous programming with Cooperative Groups*. URL : `https://files.inria.fr/pacap/collange/talks/collange_warp_synchronous_19.pdf` (Cité page : 18).

[Con83] M. L. Connolly. « Analytical molecular surface calculation ». In : *Journal of Applied Crystallography* 16.5 (oct. 1983), p. 548-558 (Cité pages : 21, 26, 27, 47, 48).

[DAr78] J. S. D'Arrigo. « Screening of membrane surface charges by divalent cations : an atomic representation ». In : *American Journal of Physiology-Cell Physiology* 235.3 (sept. 1978), p. C109-C117 (Cité page : 20).

[Del34] B. Delaunay. « Sur la sphère vide ». In : *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et na* 1934.6 (1934), p. 793-800 (Cité pages : 33, 85).

[DKT17] O. Devillers, M. Karavelas et M. Teillaud. « Qualitative symbolic perturbation : two applications of a new geometry-based perturbation framework ». en. In : *Journal of Computational Geometry* (2017), Vol. 8 No. 1 (2017) (Cité page : 104).

[DMR07] B. Daunay, A. Micaelli et S. Regnier. « 6 DOF haptic feedback for molecular docking using wave variables ». In : *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, avr. 2007 (Cité page : 112).

[DQS20] X. Duan, C. Quan et B. Stamm. « A boundary-partition-based Voronoi diagram of d-dimensional balls : definition, properties, and applications ». In : *Advances in Computational Mathematics* 46.3 (mai 2020) (Cité pages : 30, 33).

[Du+22] X. Du, Q. Zhou, N. Carr et T. Ju. « Robust computation of implicit surface networks for piecewise linear functions ». In : *ACM Transactions on Graphics* 41.4 (juill. 2022), p. 1-16 (Cité pages : 100-102).

[Ede99] H. Edelsbrunner. « Deformable Smooth Surface Design ». In : *Discrete & Computational Geometry* 21.1 (jan. 1999), p. 87-115 (Cité page : 25).

[EFL98] H. Edelsbrunner, M. Facello et J. Liang. « On the definition and the construction of pockets in macromolecules ». In : *Discrete Applied Mathematics* 88.1–3 (nov. 1998), p. 83-102 (Cité page : 33).

[EKS83] H. Edelsbrunner, D. Kirkpatrick et R. Seidel. « On the shape of a set of points in the plane ». In : *IEEE Transactions on Information Theory* 29.4 (juill. 1983), p. 551-559 (Cité page : 26).

[Fér+09] N. Férey, J. Nelson, C. Martin, L. Picinali, G. Bouyer, A. Tek, P. Bourdot, J. M. Burkhardt, B. F. G. Katz, M. Ammi, C. Etchebest et L. Autin. « Multisensory VR interaction for protein-docking in the CoRSAIRe project ». In : *Virtual Reality* 13.4 (oct. 2009), p. 273-293 (Cité page : 112).

[Fly66] M. Flynn. « Very high-speed computing systems ». In : *Proceedings of the IEEE* 54.12 (1966), p. 1901-1909 (Cité page : 16).

[GB78] J. Greer et B. L. Bush. « Macromolecular shape and surface maps by solvent exclusion. » In : *Proceedings of the National Academy of Sciences* 75.1 (jan. 1978), p. 303-307 (Cité page : 20).

[GKP12] D. Geiß, R. Klein et R. Penninger. « Optimally Solving a Transportation Problem Using Voronoi Diagrams ». In : *Computing and Combinatorics*. Springer Berlin Heidelberg, 2012, p. 264-274 (Cité page : 37).

[Gou+10] O. Gourmel, A. Pajot, M. Paulin, L. Barthe et P. Poulin. « Fitted BVH for Fast Raytracing of Metaballs ». In : 29.2 (mai 2010), p. 281-288 (Cité page : 25).

[GPF97] A. Goede, R. Preissner et C. Frömmel. « Voronoi cell : New method for allocation of space among atoms : Elimination of avoidable errors in calculation of atomic volume and density ». In : *Journal of Computational Chemistry* 18.9 (juill. 1997), p. 1113-1123 (Cité page : 30).

[GR03] M. Gavrilova et J. Rokne. « Updating the topology of the dynamic Voronoi diagram for spheres in Euclidean d-dimensional space ». In : *Computer Aided Geometric Design* 20.4 (juill. 2003), p. 231-242 (Cité pages : 35, 66, 77).

[Gra+19] P. Gralka, M. Becher, M. Braun, F. Frieß, C. Müller, T. Rau, K. Schatz, C. Schulz, M. Krone, G. Reina et T. Ertl. « MegaMol – A Comprehensive Prototyping Framework for Visualizations ». In : *The European Physical Journal Special Topics* 227.14 (mars 2019), p. 1817-1829 (Cité pages : 59, 60).

[Har96] J. C. Hart. « Sphere tracing : a geometric method for the antialiased ray tracing of implicit surfaces ». In : *The Visual Computer* 12.10 (déc. 1996), p. 527-545 (Cité pages : 24, 45).

[HDS96] W. Humphrey, A. Dalke et K. Schulten. « VMD – Visual Molecular Dynamics ». In : *Journal of Molecular Graphics* 14 (1996), p. 33-38 (Cité page : 26).

[Her+17] P. Hermosilla, M. Krone, V. Guallar, P.-P. Vázquez, À. Vinacua et T. Ropinski. « Interactive GPU-based generation of solvent-excluded surfaces ». en. In : *The Visual Computer* 33.6-8 (mai 2017), p. 869-881 (Cité pages : 23, 24).

[Hoe14] R. Hoetzlein. « Fast Fixed-Radius Nearest Neighbors : Interactive Million-Particle Fluids ». In : *GPU Technology Conference (GTC)*. 2014 (Cité pages : 50, 114).

[HW12] L. Hosek et A. Wilkie. « An analytic model for full spectral sky-dome radiance ». In : *ACM Transactions on Graphics* 31.4 (août 2012), p. 1-9 (Cité page : 116).

[Iak+20] G. Iakovou, M. Alhazzazi, S. Hayward et S. D. Laycock. « DockIT : a tool for interactive molecular docking and molecular complex construction ». In : *Bioinformatics* 36.24 (déc. 2020). Sous la dir. d'Y. Ponty, p. 5698-5700 (Cité page : 112).

[JO20] M. Jarzynski et M. Olano. « Hash Functions for GPU Rendering ». In : *Journal of Computer Graphics Techniques (JCGT)* 9.3 (2020), p. 21-38 (Cité pages : 98, 99).

[Joh23] B. Johnston. *BradyAJohnston/MolecularNodes : MolecularNodes v2.2.0 for Blender 3.4.X*. 2023. URL : `https://zenodo.org/record/7498428` (Cité page : 8).

[Joo+05] R. Joonghyun, K. Donguk, C. Youngsong, P. Rhohun et K. Deok-Soo. « Computation of Molecular Surface Using Euclidean Voronoi Diagram ». In : *Computer-Aided Design and Applications* 2.1-4 (jan. 2005), p. 439-448 (Cité pages : 30, 34).

[Jur+16] A. Jurcik, J. Parulek, J. Sochor et B. Kozlikova. « Accelerated visualization of transparent molecular surfaces in molecular dynamics ». In : IEEE, avr. 2016 (Cité pages : 29, 42, 62).

[Kam20] M. Kamarianakis. « Predicates of the 3D Apollonius Diagram ». Thèse de doct. 2020 (Cité pages : 34, 35).

[Kar12] T. Karras. « Maximizing parallelism in the construction of BVHs, octrees, and k-d trees ». In : *Proceedings of the Fourth ACM SIGGRAPH / Eurographics Conference on High-Performance Graphics*. EGGH-HPG'12. Paris, France : Eurographics Association, 2012, p. 33-37 (Cité page : 97).

[Kau+13] D. Kauker, M. Krone, A. Panagiotidis, G. Reina et T. Ertl. « Rendering Molecular Surfaces using Order-Independent Transparency ». In : *Eurographics Symposium on Parallel Graphics and Visualization*. Sous la dir. de F. Marton et K. Moreland. The Eurographics Association, 2013 (Cité page : 29).

[KBE09] M. Krone, K. Bidmon et T. Ertl. « Interactive Visualization of Molecular Surface Dynamics ». In : 15.6 (nov. 2009), p. 1391-1398 (Cité pages : 27-29, 61, 62).

[KCK04] D. Kim, Y. Cho et D. Kim. « Edge-tracing algorithm for euclidean voronoi diagram of 3d spheres ». In : *Proceedings of the 16th Canadian Conference on Computational Geometry, CCCG'04, Concordia University, Montréal, Québec, Canada, August 9-11, 2004*. 2004, p. 176-179 (Cité pages : 35, 72).

[KCK05] D.-S. Kim, Y. Cho et D. Kim. « Euclidean Voronoi diagram of 3D balls and its computation via tracing edges ». In : *Computer-Aided Design* 37.13 (nov. 2005), p. 1412-1424 (Cité pages : 34, 35, 79).

[KDE10] M. Krone, C. Dachsbacher et T. Ertl. « Parallel computation and interactive visualization of time-varying solvent excluded surfaces ». In : *Proceedings of the First ACM International Conference on Bioinformatics and Computational Biology - BCB '10*. ACM Press, 2010 (Cité pages : 28, 51, 109).

[KGE11] M. Krone, S. Grottel et T. Ertl. « Parallel Contour-Buildup algorithm for the molecular surface ». In : IEEE, oct. 2011 (Cité pages : 28-30, 51-54, 58-60).

[Kim+19] D. Kim, M. Lee, Y. Cho et D.-S. Kim. « Beta-complex vs. Alpha-complex : Similarities and Dissimilarities ». In : *IEEE Transactions on Visualization and Computer Graphics* (2019), p. 1-1 (Cité pages : 30, 33).

[Koz+17] B. Kozlíková, M. Krone, M. Falk, N. Lindow, M. Baaden, D. Baum, I. Viola, J. Parulek et H.-C. Hege. « Visualization of Biomolecular Structures : State of the Art Revisited ». In : *Computer Graphics Forum* 36.8 (2017), p. 178-204 (Cité page : 8).

[Kro+16] M. Krone, B. Kozlíková, N. Lindow, M. Baaden, D. Baum, J. Parulek, H.-C. Hege et I. Viola. « Visual Analysis of Biomolecular Cavities : State of the Art ». In : *Computer Graphics Forum* 35.3 (juin 2016), p. 527-551 (Cité pages : 11, 33, 110).

[KY02] M. I. Karavelas et M. Yvinec. « Dynamic Additively Weighted Voronoi Diagrams in 2D ». In : *Algorithms — ESA 2002*. Springer Berlin Heidelberg, 2002, p. 586-598 (Cité pages : 34, 35).

[Lag+18] L. Lagardère, L.-H. Jolly, F. Lipparini, F. Aviat, B. Stamm, Z. F. Jing, M. Harger, H. Torabifard, G. A. Cisneros, M. J. Schnieders, N. Gresh, Y. Maday, P. Y. Ren, J. W. Ponder et J.-P. Piquemal. « Tinker-HP : a massively parallel molecular dynamics package for multiscale simulations of large complex systems with advanced point dipole polarizable force fields ». In : *Chemical Science* 9.4 (2018), p. 956-972 (Cité page : 114).

[LB02] P. Laug et H. Borouchaki. « Molecular Surface Modeling and Meshing ». In : *Engineering with Computers* 18.3 (oct. 2002), p. 199-210 (Cité page : 27).

[LB13] B. Lévy et N. Bonneel. « Variational Anisotropic Surface Meshing with Voronoi Parallel Linear Enumeration ». In : *Proceedings of the 21st International Meshing Roundtable*. Springer Berlin Heidelberg, 2013, p. 349-366 (Cité pages : 37, 87, 93).

[LBH11] N. Lindow, D. Baum et H.-C. Hege. « Voronoi-Based Extraction and Visualization of Molecular Paths ». In : *IEEE Transactions on Visualization and Computer Graphics* 17.12 (déc. 2011), p. 2025-2034 (Cité pages : 30, 102, 103).

[LBH14] N. Lindow, D. Baum et H.-C. Hege. « Ligand Excluded Surface : A New Type of Molecular Surface ». In : *IEEE Transactions on Visualization and Computer Graphics* 20.12 (déc. 2014), p. 2486-2495 (Cité pages : 23, 24).

[LC87] W. E. Lorensen et H. E. Cline. « Marching cubes : A high resolution 3D surface construction algorithm ». In : *ACM SIGGRAPH Computer Graphics* 21.4 (août 1987), p. 163-169 (Cité pages : 24, 115).

[LDC05] T.-C. Lu, J. Ding et S. Crivelli. « DockingShop : a Tool for Interactive Protein Docking ». In : *2005 IEEE Computational Systems Bioinformatics Conference - Workshops (CSBW'05)*. IEEE, 2005 (Cité page : 112).

[Lev+14] G. Levieux, G. Tiger, S. Mader, J.-F. Zagury, S. Natkin et M. Montes. « Udock, the interactive docking entertainment system ». In : *Faraday Discuss.* 169 (2014), p. 425-441 (Cité pages : 112-114, 117).

[Lév+21] D. Lévy, A. D. Cicco, A. Bertin et M. Dezi. « La cryo-microscopie électronique révèle une nouvelle vision de la cellule et de ses composants ». fr. In : *médecine/sciences* 37.4 (avr. 2021), p. 379-385 (Cité page : 9).

[Lin+10] N. Lindow, D. Baum, S. Prohaska et H.-C. Hege. « Accelerated Visualization of Dynamic Molecular Surfaces ». In : *Computer Graphics Forum* 29.3 (août 2010), p. 943-952 (Cité page : 25).

[Lin+13] N. Lindow, D. Baum, A.-N. Bondar et H.-C. Hege. « Exploring cavity dynamics in biomolecular systems ». In : *BMC Bioinformatics* 14.S19 (nov. 2013) (Cité pages : 30, 33, 105).

[LSK22] M. Lee, K. Sugihara et D.-S. Kim. « Robust Construction of Voronoi Diagrams of Spherical Balls in Three-Dimensional Space ». In : *Computer-Aided Design* 152 (nov. 2022), p. 103374 (Cité pages : 36, 37).

[Luc+99] V. A. Luchnikov, N. N. Medvedev, L. Oger et J.-P. Troadec. « Voronoi-Delaunay analysis of voids in systems of nonspherical particles ». In : *Physical Review E* 59.6 (juin 1999), p. 7205-7212 (Cité pages : 34, 35).

[Man+17] M. Manak, M. Zemek, J. Szkandera, I. Kolingerova, E. Papaleo et M. Lambrughi. « Hybrid Voronoi diagrams, their computation and reduction for applications in computational biochemistry ». In : *Journal of Molecular Graphics and Modelling* 74 (juin 2017), p. 225-233 (Cité page : 33).

[Mar+22] M. Maria, S. Guionnière, N. Dacquay, Y. Naïmi, J.-P. Piquemal, G. Levieux et M. Montes. « VTX : High-performance molecular structure and dynamics visualization software ». In : *8th chemoinformatics Strasbourg summer school*. Strasbourg, France, juin 2022 (Cité page : 9).

[Mat+19] N. Matthews, A. Kitao, S. Laycock et S. Hayward. « Haptic-Assisted Interactive Molecular Docking Incorporating Receptor Flexibility ». In : *Journal of Chemical Information and Modeling* 59.6 (avr. 2019), p. 2900-2912 (Cité page : 112).

[Med+06] N. N. Medvedev, V. P. Voloshin, V. A. Luchnikov et M. L. Gavrilova. « An algorithm for three-dimensional Voronoi S-network ». In : *Journal of Computational Chemistry* 27.14 (2006), p. 1676-1692 (Cité pages : 35, 86).

[Men+11] X.-Y. Meng, H.-X. Zhang, M. Mezei et M. Cui. « Molecular Docking : A Powerful Approach for Structure-Based Drug Discovery ». In : *Current Computer Aided-Drug Design* 7.2 (juin 2011), p. 146-157 (Cité pages : 113, 114).

[MGM12] P. Moreno-Regidor, J. García López de Lacalle et M.-Á. Manso-Callejo. « Zone design of specific sizes using adaptive additively weighted Voronoi diagrams ». In : *International Journal of Geographical Information Science* 26.10 (oct. 2012), p. 1811-1829 (Cité page : 37).

[MJK16] M. Manak, L. Jirkovsky et I. Kolingerova. « Interactive Analysis of Connolly Surfaces for Various Probes ». In : *Computer Graphics Forum* 36.6 (mai 2016), p. 160-172 (Cité pages : 30, 34, 52, 63, 105).

[MK16] M. Manak et I. Kolingerova. « Extension of the edge tracing algorithm to disconnected Voronoi skeletons ». In : *Information Processing Letters* 116.2 (fév. 2016), p. 85-92 (Cité page : 35).

[MKB19] X. Martinez, M. Krone et M. Baaden. « QuickSES : A Library for Fast Computation of Solvent Excluded Surfaces ». In : *Workshop on Molecular Graphics and Visual Analysis of Molecular Data* (2019) (Cité page : 24).

[MLW07] A. McLean, F. F. Leymarie et G. Wiggins. « Apollonius diagrams and the Representation of Sounds and Music ». In : *4th International Symposium on Voronoi Diagrams in Science and Engineering (ISVD 2007)*. IEEE, juill. 2007 (Cité page : 37).

[NVIa] NVIDIA. *CUB*. URL : `https://nvlabs.github.io/cub/` (Cité page : 57).

[NVIb] NVIDIA. *CUDA C++ Programming Guide*. URL : `https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html` (Cité page : 17).

[NVI23] NVIDIA. *NVIDIA ADA GPU ARCHITECTURE*. 2023. URL : `https://images.nvidia.com/aem-dam/Solutions/Data-Center/l4/nvidia-ada-gpu-architecture-whitepaper-V2.02.pdf` (Cité pages : 16, 96).

[ONe14] M. E. O'Neill. *PCG : A Family of Simple Fast Space-Efficient Statistically Good Algorithms for Random Number Generation*. Rapp. tech. HMC-CS-2014-0905. Claremont, CA : Harvey Mudd College, sept. 2014 (Cité pages : 98, 99).

[OV14] K. Olechnovič et Č. Venclovas. « Voronota : A fast and reliable tool for computing the vertices of the Voronoi diagram of atomic balls ». In : *Journal of Computational Chemistry* 35.8 (fév. 2014), p. 672-681 (Cité pages : 35-37, 94, 97, 98, 100).

[Par+10] S. G. Parker, J. Bigler, A. Dietrich, H. Friedrich, J. Hoberock, D. Luebke, D. McAllister, M. McGuire, K. Morley, A. Robison et M. Stich. « OptiX ». In : *ACM Transactions on Graphics* 29.4 (juill. 2010), p. 1-13 (Cité pages : 62, 103).

[Pet+04] E. F. Pettersen, T. D. Goddard, C. C. Huang, G. S. Couch, D. M. Greenblatt, E. C. Meng et T. E. Ferrin. « UCSF Chimera—A visualization system for exploratory research and analysis ». In : *Journal of Computational Chemistry* 25.13 (juill. 2004), p. 1605-1612 (Cité page : 26).

[Pet+20] E. F. Pettersen, T. D. Goddard, C. C. Huang, E. C. Meng, G. S. Couch, T. I. Croll, J. H. Morris et T. E. Ferrin. « UCSF ChimeraX : Structure visualization for researchers, educators, and developers ». In : *Protein Science* 30.1 (oct. 2020), p. 70-82 (Cité page : 24).

[Pla+23a] C. Plateau–Holleville, S. Guionnière, B. Boyer, B. Jiménez-Garcia, G. Levieux, S. Mérillou, M. Maria et M. Montes. « UDock2 : interactive real-time multi-body protein–protein docking software ». In : *Bioinformatics* 39.10 (oct. 2023). Sous la dir. d'Y. Ponty (Cité pages : 14, 113).

[Pla+23b] C. Plateau–Holleville, M. Montes, M. Maria et S. Mérillou. « Construction efficace de la surface de systèmes moléculaires complexes sur GPU ». In : *J.FIG 2023 - Journées Françaises de l'Informatique Graphique*. Montpellier, France, nov. 2023. URL : `https://hal.science/hal-04633888` (Cité page : 13).

[Pla+24] C. Plateau–Holleville, M. Maria, S. Mérillou et M. Montes. « Efficient GPU computation of large protein Solvent-Excluded Surface ». In : *IEEE Transactions on Visualization and Computer Graphics* (2024), p. 1-12 (Cité page : 13).

[PV12] J. Parulek et I. Viola. « Implicit representation of molecular surfaces ». In : *2012 IEEE Pacific Visualization Symposium*. ISSN : 2165-8773. Fév. 2012, p. 217-224 (Cité pages : 28, 109).

[QS16] C. Quan et B. Stamm. « Mathematical analysis and calculation of molecular surfaces ». en. In : 322 (oct. 2016), p. 760-782 (Cité pages : 22, 42, 43, 45, 47, 51, 52, 110).

[QS17] C. Quan et B. Stamm. « Meshing molecular surfaces based on analytical implicit representation ». In : 71 (jan. 2017), p. 200-210 (Cité pages : 43, 47, 49, 63).

[Rau+19] T. Rau, S. Zahn, M. Krone, G. Reina et T. Ertl. « Interactive CPU-based Ray Tracing of Solvent Excluded Surfaces ». en. In : *Eurographics Workshop on Visual Computing for Biology and Medicine* (2019) (Cité pages : 29, 42, 49).

[Ray+19] N. Ray, D. Sokolov, S. Lefebvre et B. Lévy. « Meshless voronoi on the GPU ». In : *ACM Transactions on Graphics* 37.6 (jan. 2019), p. 1-12 (Cité pages : 38, 85, 86, 96, 97, 104).

[RCK09] J. Ryu, Y. Cho et D.-S. Kim. « Triangulation of molecular surfaces ». In : *Computer-Aided Design* 41.6 (juin 2009), p. 463-478 (Cité page : 27).

[Ric77] F. M. Richards. « AREAS, VOLUMES, PACKING, AND PROTEIN STRUCTURE ». In : *Annual Review of Biophysics and Bioengineering* 6.1 (juin 1977), p. 151-176 (Cité pages : 11, 20, 21).

[Ric84] T. J. Richmond. « Solvent accessible surface area and excluded volume in proteins ». In : *Journal of Molecular Biology* 178.1 (sept. 1984), p. 63-89 (Cité page : 11).

[Ryc09] C. H. Rycroft. « VORO++ : A three-dimensional Voronoi cell library in C++ ». In : *Chaos : An Interdisciplinary Journal of Nonlinear Science* 19.4 (déc. 2009), p. 041111 (Cité page : 37).

[Ryu+] J. Ryu, R. Park, J. Seo, C. Kim, H. C. Lee et D.-S. Kim. « Real-Time Triangulation of Molecular Surfaces ». In : *Computational Science and Its Applications – ICCSA 2007*. Springer Berlin Heidelberg, p. 55-67 (Cité page : 27).

[SD20] L. Schrödinger et W. DeLano. *PyMOL*. Version 2.4.0. Mai 2020. URL : `http://www.pymol.org/pymol` (Cité page : 26).

[Sig+06] C. Sigg, T. Weyrich, M. Botsch et M. Gross. « GPU-Based Ray-Casting of Quadratic Surfaces ». In : *Symposium on Point-Based Graphics*. Sous la dir. de M. Botsch, B. Chen, M. Pauly et M. Zwicker. The Eurographics Association, 2006 (Cité page : 28).

[SK19] M. Schäfer et M. Krone. « A Massively Parallel CUDA Algorithm to Compute and Visualize the Solvent Excluded Surface for Dynamic Molecular Data ». In : *Workshop on Molecular Graphics and Visual Analysis of Molecular Data* (2019) (Cité pages : 28, 51, 59, 60).

[SKV12] R. Sinha, P. J. Kundrotas et I. A. Vakser. « Protein Docking by the Interface Structure Similarity : How Much Structure Is Needed ? » In : *PLoS ONE* 7.2 (fév. 2012). Sous la dir. d'O. Keskin, e31349 (Cité page : 114).

[Sla+22] A. Slater, N. Nair, R. Suétt, R. M. Donnchadha, C. Bamford, S. Jasim, D. Livingstone et E. Hutchinson. « Visualising Viruses ». In : *Journal of General Virology* 103.1 (jan. 2022) (Cité page : 8).

[SNA17] M. Sainlot, V. Nivoliers et D. Attali. « Restricting Voronoi diagrams to meshes using corner validation ». In : *Computer Graphics Forum* 36.5 (août 2017), p. 81-91 (Cité page : 93).

[Son+22] C. Song, M. Lee, S. Choi et D.-S. Kim. « Benchmark dataset for the Voronoi diagram of 3D spherical balls ». In : *Data in Brief* 45 (déc. 2022), p. 108605 (Cité pages : 36, 100, 102).

[SOS96] M. F. Sanner, A. J. Olson et J.-C. Spehner. « Reduced surface : An efficient way to compute molecular surfaces ». en. In : *Biopolymers* 38.3 (1996), p. 305-320 (Cité pages : 26, 27, 33, 38, 56).

[SS07] M. Sharifzadeh et C. Shahabi. « Processing Optimal Sequenced Route Queries Using Voronoi Diagrams ». In : *GeoInformatica* 12.4 (oct. 2007), p. 411-433 (Cité page : 37).

[TA96] M. Totrov et R. Abagyan. « The Contour-Buildup Algorithm to Calculate the Analytical Molecular Surface ». en. In : 116.1 (jan. 1996), p. 138-143 (Cité pages : 26-28, 33, 43, 49).

[The23] The CGAL Project. *CGAL User and Reference Manual*. 5.6. CGAL Editorial Board, 2023. URL : `https://doc.cgal.org/5.6/Manual/packages.html` (Cité pages : 34, 80).

[VBW94] A. Varshney, F. P. Brooks, Jr. et W. V. Wright. « Linearly Scalable Computation of Smooth Molecular Surfaces ». In : *IEEE Computer Graphics and Applications* 14.5 (sept. 1994), p. 19-25 (Cité pages : 26, 27).

[Wan+16] Z. Wang, H. Sun, X. Yao, D. Li, L. Xu, Y. Li, S. Tian et T. Hou. « Comprehensive evaluation of ten docking programs on a diverse set of protein–ligand complexes : the prediction accuracy of sampling power and scoring power ». In : *Physical Chemistry Chemical Physics* 18.18 (2016), p. 12964-12975 (Cité page : 112).

[Wan+20] P. Wang, N. Yuan, Y. Ma, S. Xin, Y. He, S. Chen, J. Xu et W. Wang. « Robust Computation of 3D Apollonius Diagrams ». In : *Computer Graphics Forum* 39.7 (oct. 2020), p. 43-55 (Cité pages : 36, 66, 77).

[Wil99] H.-M. Will. « Computation of additively weighted Voronoi cells for applications in molecular biology ». en. Thèse de doct. 1999 (Cité pages : 33-35, 66, 72).

[WZ10] R. Weller et G. Zachmann. « ProtoSphere : a GPU-assisted prototype guided sphere packing algorithm for arbitrary objects ». In : *ACM SIGGRAPH ASIA 2010 Sketches*. SA '10. ACM, déc. 2010 (Cité page : 37).

[Yu09] Z. Yu. « A list-based method for fast generation of molecular surfaces ». In : *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, sept. 2009 (Cité page : 23).

[Zel+23] S. Zellmann, Q. Wu, K.-L. Ma et I. Wald. « Memory-Efficient GPU Volume Path Tracing of AMR Data Using the Dual Mesh ». In : *Computer Graphics Forum* 42.3 (juin 2023), p. 51-62 (Cité page : 19).

[ZIK98] S. Zhukov, A. Iones et G. Kronin. « An ambient light illumination model ». In : *Rendering Techniques '98*. Springer Vienna, 1998, p. 45-55 (Cité page : 103).

[Zon+08] N. Zonta, I. J. Grimstead, N. J. Avis et A. Brancale. « Accessible haptic technology for drug design applications ». In : *Journal of Molecular Modeling* 15.2 (déc. 2008), p. 193-196 (Cité page : 112).

Liste des Travaux

Journaux nationaux à comité de lecture

- C. Plateau–Holleville, S. Guionnière, B. Boyer, B. Jiménez-Garcia, G. Levieux, S. Mérillou, M. Maria et M. Montes. ≪ UDock2 : interactive real-time multi-body protein–protein docking software ≫. In : *Bioinformatics* 39.10 (oct. 2023). Sous la dir. d'Y. Ponty.
- C. Plateau–Holleville, M. Maria, S. Mérillou et M. Montes. ≪ Efficient GPU computation of large protein Solvent-Excluded Surface ≫. In : *IEEE Transactions on Visualization and Computer Graphics* (2024), p. 1-12.

Construction efficace de géométrie pour l'analyse structurelle de grands systèmes moléculaires

Résumé : L'étude structurelle de complexes moléculaires est nécessaire à la compréhension de leurs fonctionnements, mais aussi leur analyse à travers leur visualisation et illustration. La Surface Exclue au Solvant (SES) représente implicitement l'interaction entre un corps moléculaire et un solvant ce qui permet d'analyser géométriquement certaines de ses propriétés. Cette surface reste cependant complexe à construire notamment pour des structures de grandes tailles. Dans cette thèse, nous présentons ainsi une méthode de calcul sur GPU de la SES de grandes protéines. Les diagrammes d'Apollonius, ou diagrammes de Voronoï additivement pondérés, peuvent servir à étudier la structure des protéines, mais aussi construire la SES efficacement. Nous présentons une caractérisation mathématique de ces diagrammes permettant leur analyse et leur paramétrisation pour un calcul naïf, mais exhaustif, de leur géométrie. Enfin, sur la base de notre étude, nous proposons une méthode de calcul GPU de diagrammes d'Apollonius dans $\mathbb{R}^3$ compatible avec des protéines de grandes tailles, mais aussi avec des distributions spatiales homogènes. Cette stratégie supporte les particularités des diagrammes d'Apollonius et permet le calcul exhaustif de leurs composantes.

Mots clés : Visualisation scientifique, surfaces moléculaires, diagrammes de Voronoï, diagrammes d'Apollonius, géométrie algorithmique, GPGPU.

Efficient computation of geometry for structural analysis of large molecular systems

Abstract : The study of the structure of large molecular systems through their visualization and illustration is needed to understand their features and the system they take part in. The Solvent Excluded Surface (SES) is the interaction surface between a protein and its environment. It then allows characterizing geometrically some of its properties. However, this surface is still complex to compute, especially for large molecular complex. In this thesis, we propose a dedicated GPU pipeline targeting fast and efficient computation of the SES of large proteins. Apollonius diagrams, or additively weighted Voronoi diagrams, can be used to analyze proteins structures and construct their surface. Then, we present a complete characterization of their facets which allows a parametrization supporting the exhaustive and naive computation of their geometry. Based on this study, we propose a method allowing the computation of Apollonius diagrams in $\mathbb{R}^3$ which support the study of large proteins but is also compatible with uniform spatial distributions. Additionally, it allows an exhaustive computation of their facets.

Keywords : Visualization, molecular surface, Voronoi diagrams, Apollonius diagrams, computational geometry, GPGPU.

Univ. de Limoges, CNRS, XLIM, UMR 7252, F-87000 Limoges, France