Journées du Groupe de Travail en Modélisation Géométrique 2014, Lyon

Calcul d'un diagramme de Voronoï restreint en dimension quelconque

Vincent Nivoliers[1] et Christophe Geuzaine[1]

[1]Université de Liège, Institut Montefiore B28, 4000 Liège, Belgique

Résumé

Un diagramme de Voronoï restreint est un outil fondamental pour l'étude de l'échantillonnage d'une surface. Il permet de partitionner une surface en fonction d'un ensemble de sites, en associant à chaque site la portion de surface pour laquelle ce site est le plus proche. Les applications sont nombreuses, en particulier dans le domaine de la génération de maillages. Nous nous intéressons ici au calcul d'un tel diagramme en dimension quelconque : étant donné un maillage $\mathcal{S}$ constitué de triangles dont les sommets sont des points de $\mathbb{R}^d$ et un ensemble de sites $\mathbf{V} = \{\mathbf{v}_k\}_{k=1}^n \subset \mathbb{R}^d$, il s'agit de déterminer pour chaque point de $\mathcal{S}$ le site le plus proche dans $\mathbf{V}$. Le coût de calcul du diagramme de Voronoï complet de $\mathbf{V}$ dans $\mathbb{R}^d$ devient vite prohibitif lorsque la dimension d augmente. Nous étendons ici les travaux de Lévy et Bonneel [LB12], qui approximent les voisinages du diagramme de Voronoï à l'aide de requêtes de plus proche voisinage. Nous proposons deux nouvelles formulations permettant de réduire le nombre de sites à considérer pour chaque triangle de $\mathcal{S}$, en particulier lorsque les triangles de $\mathcal{S}$ sont éloignés des sites de $\mathbf{V}$.

Mots-clés : Modélisation géométrique, Géométrie algorithmique, maillage, diagramme de Voronoï restreint

1. Introduction

Les diagrammes de Voronoï ont de nombreuses applications, et leur calcul a fait l'objet de nombreuses recherches. Des algorithmes efficaces existent pour les calculer, et leur complexité reste abordable en dimension deux et trois. Leur calcul en dimension plus élevée devient cependant rapidement prohibitif, car la complexité en espace et en temps des algorithmes existants augmente exponentiellement avec la dimension. Nous présentons ici une méthode pour calculer des diagrammes de Voronoï *restreints* efficacement en dimension quelconque. Il s'agit, étant donné une surface maillée dont les sommets sont dans $\mathbb{R}^d$, de calculer l'intersection entre un diagramme de Voronoï et cette surface. Pour y parvenir, nous évitons le calcul global du diagramme de Voronoï, et suivons la technique de Lévy et Bonneel [LB12], qui utilisent à la place des requêtes de plus proche voisinage, qui restent abordables en dimension élevée. Pour rappel, nous détaillons leur méthode en Section 2.4.

Notre contribution consiste à améliorer la méthode qu'ils proposent pour gérer des configurations de sites plus générales. Lévy et Bonneel utilisent leur algorithme à des fins de remaillage, et s'intéressent donc à la situation où les sites du diagramme de Voronoï sont situés sur la surface avec laquelle l'intersection est calculée. Si leur approche fonctionne bien dans cette situation, nous montrons en Section 2.4.3 qu'elle peut rapidement devenir très coûteuse lorsque les sites s'éloignent de la surface. Nous proposons donc en Section 3 et Section 4 deux nouvelles méthodes pour sélectionner les points où réaliser les requêtes de plus proche voisinage, ainsi que pour limiter le nombre de voisins à considérer. Nous montrons en Section 5 les bénéfices de ces approches par rapport à celle de Lévy et Bonneel.

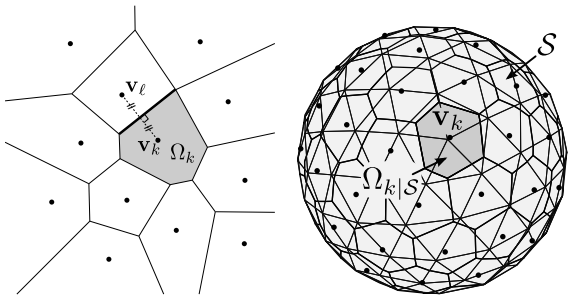
2. Fondements

2.1. Diagramme de Voronoï

Étant donné un ensemble de sites $\mathbf{V} = \{\mathbf{v}_k\}_{k=1}^n \subset \mathbb{R}^d$, la *cellule de Voronoï* Ω_k d'un site $\mathbf{v}_k$ est définie comme

$$\Omega_k = \left\{ \mathbf{p} \in \mathbb{R}^d, \|\mathbf{v}_k - \mathbf{p}\| \leq \|\mathbf{v}_\ell - \mathbf{p}\| \text{ pour tout } \mathbf{v}_\ell \in \mathbf{V} \right\}.$$

Étant donné une paire de sites $(\mathbf{v}_k, \mathbf{v}_\ell) \in \mathbf{V}^2$, l'ensemble des points $\mathbf{p} \in \mathbb{R}^d$ tels que $\|\mathbf{v}_k - \mathbf{p}\| = \|\mathbf{v}_\ell - \mathbf{p}\|$ est un plan perpendiculaire au segment $[\mathbf{v}_k, \mathbf{v}_\ell]$ appelé le *bissecteur* de $\mathbf{v}_k$ et



(a) classique (b) restreint

Figure 1: *Diagrammes de Voronoï*

$\mathbf{v}_\ell$. Les cellules de Voronoï sont donc des polyèdres convexes de $\mathbb{R}^d$, dont les faces correspondent à des bissecteurs. La Figure 1(a) donne un exemple de diagramme de Voronoï en dimension 2.

Le *diagramme de Voronoï* de $\mathbf{V}$ est l'ensemble des cellules de Voronoï. Son calcul est un problème classique ayant fait l'objet de nombreux travaux. Nous nous référons à la synthèse d'Okabe et al. [OBSC09] pour une revue détaillées de ces travaux et des applications des diagrammes de Voronoï. Des bibliothèques génériques existent pour calculer des diagrammes de Voronoï en dimension quelconque, en particulier CGAL [CGA] et QHull [BDH96].

2.2. Diagramme de Voronoï restreint

Étant donné un maillage $\mathcal{S}$, la cellule de Voronoï d'un site $\mathbf{v}_k$ restreinte à $\mathcal{S}$ est donnée par

$$\begin{aligned}\Omega_{k|\mathcal{S}} &= \Omega_k \cap \mathcal{S} \\ &= \{\mathbf{x} \in \mathcal{S}, \|\mathbf{v}_k - \mathbf{x}\| \leq \|\mathbf{v}_\ell - \mathbf{x}\| \text{ pour tout } \mathbf{v}_\ell \in \mathbf{V}\}.\end{aligned}$$

Le diagramme de Voronoï de $\mathbf{V}$ restreint à $\mathcal{S}$ est l'ensemble de ces cellules restreintes (Figure 1(b)).

La notion de diagramme de Voronoï restreint apparaît dans plusieurs domaines. Elle est notamment utilisée en reconstruction et en génération de maillages pour calculer, étant donné un ensemble de points, un ensemble de triangles reliant ces points pour former une surface [ES97]. Plus récemment, les travaux de Yan et al. [YLL*09] proposent une méthode pour remailler une surface, en généralisant l'algorithme de Lloyd [Llo82] via le calcul de diagrammes de Voronoï restreints *barycentriques*. Nivoliers et al. [NYL12] approximent la *distance au carré* entre deux surfaces en utilisant deux diagramme de Voronoï restreints, et utilisent cette approximation pour élaborer un algorithme d'ajustement de surfaces.

Dans tous ces travaux, les diagrammes de Voronoï restreints sont utilisés comme une approximation d'un diagramme de Voronoï *géodésique* sur la surface : la distance géodésique entre deux points sur $\mathcal{S}$ est remplacée par la distance euclidienne dans l'espace de plongement (ici $\mathbb{R}^3$). Il est à noter que cette approximation peut être très mauvaise lorsque $\mathcal{S}$ est fine : lorsque deux portions éloignées de $\mathcal{S}$ selon la distance géodésique se trouvent proches dans l'espace de plongement par exemple. Les travaux d'Amenta et Bern [AB99] étudient en particulier les conditions à respecter pour que cette approximation soit correcte.

D'autres approches existent pour calculer exactement ou approximer un diagramme de Voronoï géodésique sur un maillage. Eck et al. [EDD*95] utilisent un procédé itératif d'agrégation des faces du maillage, pour associer chaque face au site dont elle est le plus proche. Peyré et Cohen [PC06] utilisent un mécanisme de propagation sur le maillage pour résoudre l'équation eikonale et obtenir une approximation de la distance géodésique au site le plus proche sur la surface. Ces deux méthodes sont rapides, mais requièrent un maillage de bonne qualité suffisamment fin pour pouvoir approximer la forme des cellules de Voronoï. Pour calculer le diagramme de Voronoï géodésique exact, Kunze et al. [KWR97] fournissent les équations permettant de calculer les courbes délimitant les cellules dans le cas où $\mathcal{S}$ est une surface paramétrée par une fonction C^2. Liu et al. [LCT11] décrivent un algorithme calculant le diagramme de Voronoï géodésique dans le cas d'un maillage. Ces deux dernières approches sont cependant beaucoup plus complexes, et restent trop lentes pour les applications actuelles.

2.3. Calcul d'un diagramme de Voronoï restreint

L'opération de base pour le calcul d'un diagramme de Voronoï restreint est le calcul de l'intersection entre un triangle et une cellule de Voronoï. Étant donné un site $\mathbf{v}_k \in \mathbf{V}$, soit $\mathbf{N}_k$ le *voisinage de Voronoï* $\mathbf{N}_k$ de $\mathbf{v}_k$, défini par

$$\mathbf{N}^k = \{\mathbf{v}_\ell \in \mathbf{V}, \Omega_\ell \cap \Omega_k \neq \emptyset\}.$$

Un site $\mathbf{v}_\ell$ est dans $\mathbf{N}^k$ si le bissecteur de $\mathbf{v}_k$ et $\mathbf{v}_\ell$ contribue au bord de Ω_k. Pour calculer l'intersection entre un triangle et la cellule de Voronoï Ω_k, il suffit donc de considérer successivement chaque site $\mathbf{v}_\ell$ de $\mathbf{N}^k$, et de découper le triangle selon le bissecteur de $\mathbf{v}_k$ et $\mathbf{v}_\ell$. Yan et al. [YLL*09] utilisent l'algorithme de Sutherland et Hodgman [SH74] pour réaliser ces découpes. Afin d'obtenir $\mathbf{N}^k$, ils utilisent la librarie CGAL [CGA] pour calculer le diagramme de Voronoï de $\mathbf{V}$ dans $\mathbb{R}^3$.

À partir de l'intersection entre un triangle et une cellule de Voronoï, Yan et al. [YLL*09] proposent un mécanisme de propagation pour obtenir de nouveaux couples triangle – site à traiter. Étant donné le polygone P correspondant à l'intersection entre un triangle T et une cellule Ω_k, chaque arête de P peut être :

une portion d'une arête de T : Ω_k intersecte donc le triangle T' voisin de T par cette arête ; le calcul peut être propagé au couple $(T', \mathbf{v}_k)$ (symbole ⧋ sur la Figure 2) ;

l'intersection entre T et le bissecteur de $\mathbf{v}_k$ et $\mathbf{v}_\ell$: la cellule Ω_ℓ intersecte donc également le triangle T ; le cal-

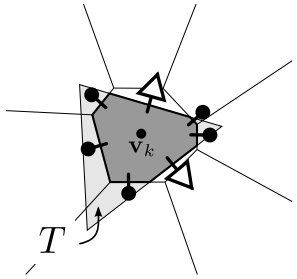


Figure 2: *Propagation du calcul d'un diagramme de Voronoï restreint. Les arêtes marquées ⍋ sont une portion d'une arête de T et permettent de propager le calcul au triangle voisin avec $\mathbf{v}_k$. Les arêtes marquées 📍 correspondent à une découpe du triangle par le bissecteur de $\mathbf{v}_k$ et d'un autre site $\mathbf{v}_\ell$ et permettent de propager le calcul au même triangle avec le site $\mathbf{v}_\ell$.*

cul peut être propagé au couple $(T, \mathbf{v}_\ell)$ (symbole 📍 sur la Figure 2).

L'organisation efficace de cette propagation est détaillée dans la thèse de Nivoliers [Niv12]. Si m est le nombre de triangles de $\mathcal{S}$ et n le nombre de sites de $\mathbf{V}$, le nombre de couples triangle – site à traiter dans le pire des cas est de l'ordre de mn : lorsque chaque triangle coupe toutes les cellules de Voronoï. L'intersection entre un triangle et un ensemble de k bissecteurs peut être réalisée en $O(k \log k)$. Dans le cas d'un diagramme de Voronoï à n sites en dimension au moins trois, il est possible que chaque cellule ait $O(n)$ faces, et l'intersection entre un triangle et une cellule peut donc atteindre une complexité en $O(n \log n)$. Dans le pire des cas, la complexité du calcul d'un diagramme de Voronoï restreint est donc $O(mn^2 \log n)$. Expérimentalement, le comportement de cet algorithme pour les cas d'application classiques semble cependant plus se rapprocher d'une complexité du type $O(m+n)$ [Niv12], même si aucune étude théorique à notre connaissance ne prouve ce résultat.

Dans la suite, nous utiliserons toujours le même principe de propagation pour calculer le diagramme de Voronoï restreint. Nous nous concentrons donc sur le cœur de l'algorithme : le calcul de l'intersection entre un triangle de $\mathcal{S}$ et une cellule de Voronoï Ω_k. Plus généralement, nous cherchons à calculer, étant donné un polygone P, l'intersection $\Omega_{k|P} = \Omega_k \cap P$.

2.4. Calcul par les plus proches voisins

Dans leurs travaux, Lévy et Bonneel [LB12] calculent des diagrammes de Voronoï restreints sur des surfaces plongées dans un espace de dimension plus élevée (les sommets sont des points de $\mathbb{R}^6$). Dans ce contexte, il n'est plus envisageable d'obtenir le voisinage de chaque site en calculant directement le diagramme de Voronoï dans $\mathbb{R}^6$ car la complexité explose. Leur stratégie consiste donc à utiliser les plus proches voisins d'un site à la place de ses voisins de Voronoï. Les plus proches voisins sont calculés efficacement en utilisant une structure de données adaptée [MA97].

2.4.1. Rayon de sécurité

Étant donné un polygone P et un site $\mathbf{v}_k$, l'idée clé de la méthode de Lévy et Bonneel [LB12] pour calculer $\Omega_{k|P} = \Omega_k \cap P$ réside dans la définition d'un critère pour déterminer le nombre de sites voisins de $\mathbf{v}_k$ voisins nécessaires. Soit $\mathbf{V}^k$ une séquence de sites distincts deux à deux, définie par

$$\mathbf{V}^k = \left\{\mathbf{v}_\ell^k\right\}_{\ell=1}^{n-1}, \text{ avec } \|\mathbf{v}_\ell^k - \mathbf{v}_k\| \leq \|\mathbf{v}_{\ell+1}^k - \mathbf{v}_k\|. \quad (1)$$

$\mathbf{V}^k$ est donc un ordonnancement de tous les sites de $\mathbf{V} \setminus \{\mathbf{v}_k\}$ par distance croissante par rapport à $\mathbf{v}_k$. Soit P_i^k le polygone défini par

$$P_i^k = \left\{\mathbf{x} \in P, \|\mathbf{x} - \mathbf{v}_k\| \leq \|\mathbf{x} - \mathbf{v}_\ell^k\| \text{ pour tout } \ell \leq i\right\}. \quad (2)$$

Moins formellement, P_i^k est la portion de P restante après une découpe par les bissecteur de $\mathbf{v}_k$ et des i sites les plus proches de lui. Soit r la fonction prenant en argument un polygone P et un point $\mathbf{p} \in \mathbb{R}^d$

$$r(P, \mathbf{p}) = \max_{\mathbf{x} \in P} \|\mathbf{x} - \mathbf{p}\|. \quad (3)$$

Le *rayon de sécurité* de P_i^k est défini comme $r(P_i^k, \mathbf{v}_k)$. L'idée sous-jacente est que si $\mathbf{v}_\ell$ est un site tel que $\|\mathbf{v}_\ell - \mathbf{v}_k\| > 2r(P_i^k, \mathbf{v}_k)$, alors le bissecteur de $\mathbf{v}_k$ et $\mathbf{v}_\ell$ ne coupe pas P_i^k, car tous les points de P_i^k sont plus proches de $\mathbf{v}_k$ que de $\mathbf{v}_\ell$. Cette remarque permet alors d'obtenir le théorème suivant, définissant le nombre de voisins nécessaire au calcul de $\Omega_{k|P}$.

Théorème 1 (Lévy and Bonneel [LB12]) Soient P un polygone et $\mathbf{v}_k$ un site de $\mathbf{V}$. Soient $\mathbf{V}^k = \{\mathbf{v}_\ell^k\}_\ell$ la séquence de sites définie par l'Équation 1, P_i^k le polygone défini par l'Équation 2 et $r(P_i^k, \mathbf{v}_k)$ le rayon de sécurité défini par l'Équation 3. Alors :

$$\|\mathbf{v}_{i+1}^k - \mathbf{v}_k\| > 2r(P_i^k, \mathbf{v}_k) \Rightarrow P_i^k = \Omega_{k|P}. \quad (4)$$

Ce théorème permet d'élaborer un algorithme permettant de calculer les cellules de Voronoï restreintes en utilisant la proximité spatiale entre les sites, sans avoir à calculer le diagramme de Voronoï complet de $\mathbf{V}$.

2.4.2. Algorithme

Le Théorème 1 permet de construire un premier algorithme pour calculer $\Omega_{k|P}$ étant donné un polygone P et un site $\mathbf{v}_k$ de $\mathbf{V}$: en itérant sur les sites de $\mathbf{V}_k$ pour calculer P_i^k et $r(P_i^k, \mathbf{v}_k)$, et stopper l'itération dès que $\|\mathbf{v}_{i+1}^k - \mathbf{v}_k\| > 2r(P_i^k, \mathbf{v}_k)$. Calculer intégralement $\mathbf{V}^k$ reste néanmoins une opération de complexité $O(n \log n)$ à éviter. Lévy et Bonneel utilisent donc la bibliothèque ANN [MA97]

pour calculer efficacement les p premiers sites de $\mathbf{V}_k$. Friedman et al. [FBF77] montrent que pour p constant, la complexité moyenne d'une telle requête pour des points distribués uniformément est de $O(\log n)$ où n est le nombre de sites. Dans le cas de Lévy et Bonneel [LB12], les points n'ont pas une distribution uniforme, car ils sont répartis à proximité de la surface $\mathcal{S}$, mais l'efficacité des requêtes reste expérimentalement bonne.

Entrées: un polygone P, un site $\mathbf{v}_k$ de $\mathbf{V}$, et p le nombre de plus proches voisins utilisés
Sorties: $\Omega_{k|P} = \Omega_k \cap P$
Données: un marqueur par site indiquant s'il a été traité
Algorithme

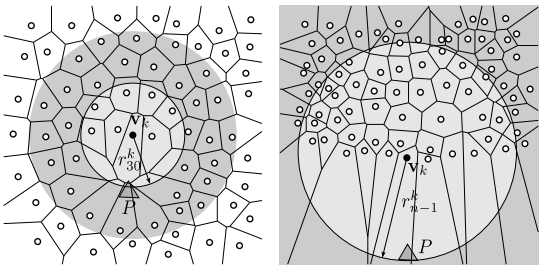
```
nbVoisins ← p
P^k ← P
r^k ← r(P^k, v_k)
marquer v_k comme traité
tant que tous les sites ne sont pas traités faire
    V^k ← recherche_ann(v_k, nbVoisins)
    pour chaque site v_ℓ^k de V_p^k dans l'ordre faire
        si ||v_ℓ^k − v_k|| > 2r^k alors
            // le rayon de sécurité est dépassé
            retourner P^k
        si v_ℓ^k n'est pas traité alors
            marquer v_ℓ^k comme traité
            P^k ← découper(P^k, v_k, v_ℓ^k)
            r^k ← r(P^k, v_k)
    // nombre de voisins insuffisant
    nbVoisins ← nbVoisins + p
retourner P^k
```

Algorithme 1: voronoi_kP_sites(P,$\mathbf{v}_k$,p) : calcul de $\Omega_{k|P}$ par les plus proches voisins. La fonction découper(P,$\mathbf{v}_k$,$\mathbf{v}_\ell$) découpe le polygone P par le bissecteur de $\mathbf{v}_k$ et $\mathbf{v}_\ell$, pour ne garder que la portion la plus proche de $\mathbf{v}_k$.

L'Algorithme 1 réalise le calcul de $\Omega_{k|P}$ en utilisant les plus proches voisins. Lorsque le nombre de plus proches voisins calculé initialement n'est pas suffisant, une nouvelle requête est effectuée en augmentant le nombre de voisins cherchés, jusqu'à obtenir l'assurance que $\Omega_{k|P}$ est calculé correctement via le critère fourni par le Théorème 1. Dans l'Algorithme 1, le nombre de voisins est augmenté linéairement, mais il est également possible de l'augmenter exponentiellement en le doublant à chaque étape. Il est également possible de stocker globalement les plus proches voisins calculés pour chaque site, pour ne pas avoir à les recalculer pour chaque triangle.

2.4.3. Limitations

Dans leurs travaux, Lévy et Bonneel s'intéressent au remaillage. Dans cette configuration, les sites sont répartis à proximité de la surface $\mathcal{S}$. Lorsque cette condition n'est plus respectée, l'Algorithme 1 peut devenir inefficace. En effet, pour un site $\mathbf{v}_k$ et un polygone P, le rayon de sécurité $r(P_i^k, \mathbf{v}_k)$ défini par l'Équation 3 est minoré par la distance entre $\mathbf{v}_k$ et le plan contenant P. Ainsi, lorsque $\mathbf{v}_k$ est loin du polygone P, le nombre de voisins à considérer pour dépasser le rayon de sécurité peut devenir très important, voire atteindre l'ensemble des sites (Figure 3).



(a) cas courant (b) cas pathologique

Figure 3: *Limitation du rayon de sécurité de Lévy et Bonneel lorsque les sites ne sont pas proches des triangles. Dans le cas courant, 37 sites sont utilisés alors que v_k n'a que 8 voisins. Dans le cas pathologique, tous les sites sont utilisés.*

3. Boule de sécurité centrée sur le maillage

Dans nos travaux, nous cherchons à trouver une formulation similaire à celle du Théorème 1, mais ayant des limitations moins fortes. Il s'agit en particulier de pouvoir calculer un diagramme de Voronoï restreint dans la situation où les sites sont éloignés de la surface. Dans leurs travaux par exemple, Nivoliers et al. [NYL12] calculent des diagrammes de Voronoï restreints pour évaluer une distance entre deux surfaces. Cette distance est ensuite utilisée dans un algorithme d'ajustement de surfaces. Dans leur contexte, les sites sont répartis sur une des deux surfaces, et la seconde surface intervient dans le calcul du diagramme de Voronoï restreint. L'ajustement de surfaces en dimension supérieure à trois est par exemple envisageable dans un contexte de reconstruction ou de vision, où plusieurs vues partielles d'une même surfaces doivent être mises en correspondance. Les dimension en plus correspondent alors à des informations supplémentaires disponibles sur la surface, comme la couleur.

Notre première idée consiste à définir un nouveau rayon de sécurité centré sur le polygone P à découper plutôt que sur le site $\mathbf{v}_k$.

3.1. Formulation

Étant donné un polygone P, soit $\mathbf{g}_P$ sont barycentre. Nous définissons cette fois un critère d'arrêt fondé sur un rayon centré sur $\mathbf{g}_P$.

Théorème 2 Soient P un polygone, $\mathbf{v}_k$ et $\mathbf{v}_\ell$ deux sites de $\mathbf{V}$. Soit $\mathbf{g}_P$ le barycentre de P. Pour tout point $\mathbf{x}$ de P,

$$\|\mathbf{v}_\ell - \mathbf{g}_P\| > r(P,\mathbf{v}_k) + r(P,\mathbf{g}_k) \Rightarrow \|\mathbf{x}-\mathbf{v}_\ell\| > \|\mathbf{x}-\mathbf{v}_k\|. \quad (5)$$

En particulier, le bissecteur de $\mathbf{v}_k$ et $\mathbf{v}_\ell$ ne coupe pas P, et P est du même côté de ce bissecteur que $\mathbf{v}_k$.

Démonstration Nous avons :

$$\|\mathbf{x}-\mathbf{v}_k\| \leq r(P,\mathbf{v}_k)$$
$$\|\mathbf{x}-\mathbf{v}_\ell\| \geq \|\mathbf{g}_P-\mathbf{v}_\ell\| - r(P,\mathbf{g}_P).$$

Ainsi, si nous avons

$$\|\mathbf{g}_P-\mathbf{v}_\ell\| > r(P,\mathbf{v}_k) + r(P,\mathbf{g}_P),$$

alors nous obtenons

$$\|\mathbf{x}-\mathbf{v}_\ell\| > r(P,\mathbf{v}_k) \geq \|\mathbf{x}-\mathbf{v}_k\|,$$

ce qui prouve le résultat. □

Le Théorème 2 permet, étant donné un polygone P et un site $\mathbf{v}_k$, de définir une boule centrée sur le polygone P permettant d'assurer que tous les sites en dehors de cette boule ne contribuent pas à l'intersection $\Omega_{k|P}$.

3.2. Algorithme naïf

L'Algorithme 2 utilise directement le Théorème 2 pour calculer $\Omega_{k|P}$. Le principe consiste à requérir les plus proches voisins du barycentre $\mathbf{g}_P$ du polygone P. Ces plus proches voisins sont examinés par distance croissante vis-à-vis de $\mathbf{g}_P$. Le Théorème 2 est utilisé pour interrompre le calcul en assurant que plus aucun site ne peut participer à la découpe du triangle. Si le calcul n'est pas terminé à l'issue de l'examen des plus proches voisins de $\mathbf{g}_P$, le barycentre est mis à jour en utilisant le polygone découpé, et une nouvelle itération est lancée en utilisant plus de voisins.

Ce fonctionnement peut néanmoins s'avérer peu efficace lorsque le nombre de sites $\mathbf{v}_\ell$ tels que $\Omega_{\ell|T}$ est non vide est élevé. Dans ce cas, même si la suite des barycentres $\mathbf{g}_P$ de l'Algorithme 2 se rapprochera progressivement de $\Omega_{k|T}$, le nombre de sites traités avant de considérer les sites réellement utiles peut être important (Figure 4). Ce comportement transparaît au niveau des performances en Section 5.4, sur la Figure 10. Nous proposons donc une formulation mixte, utilisant partiellement l'Algorithme 1.

3.3. Algorithme mixte

L'approche mixte consiste à utiliser l'Algorithme 3 pour initialiser le polygone fourni ensuite à l'Algorithme 2. L'Algorithme 3 est très proche de l'Algorithme 1, mais il n'itère pas pour terminer le calcul de $\Omega_{k|P}$. À la place, il renvoie un marqueur indiquant si le calcul de $\Omega_{k|P}$ est terminé, ou s'il doit se poursuivre. L'idée sous-jacente est que même si le polygone P est grand, en utilisant les p plus proches

Entrées: un polygone P_0, un site $\mathbf{v}_k$ de $\mathbf{V}$ et p le nombre de plus proches voisins utilisés
Sorties: $\Omega_{k|P_0} = \Omega_k \cap P_0$
Données: un marqueur par site indiquant s'il a été traité
Algorithme

```
nbVoisins ← p
P ← T
g_P ← barycentre(P)
r_P ← r(P, g_P)
r^k ← r(P, v_k)
marquer v_k comme traité
tant que nbVoisins ≤ n faire
    V^P ← recherche_ann(g_P, nbVoisins)
    pour chaque site v_ℓ^P de V^P dans l'ordre faire
        si ||v_ℓ^P − g_P|| > r^k + r_P alors
            // le rayon de sécurité est dépassé
            retourner P
        si v_ℓ^P n'a pas été traité alors
            marquer v_ℓ^P comme traité
            P ← découper(P, v_k, v_ℓ^P)
            // le barycentre n'est pas mis à jour
            r_P ← r(P, g_P)
            r^k ← r(P, v_k)
    g_P ← barycentre(P)
    r_P ← r(P, g_P)
    nbVoisins ← nbVoisins + p
retourner P
```

Algorithme 2: voronoi_kP_polygone (P,$\mathbf{v}_k$,p) : calcul de $\Omega_{k|P}$ via les plus proches voisins en utilisant un critère d'arrêt centré sur le polygone P découpé, via le Théorème 2.

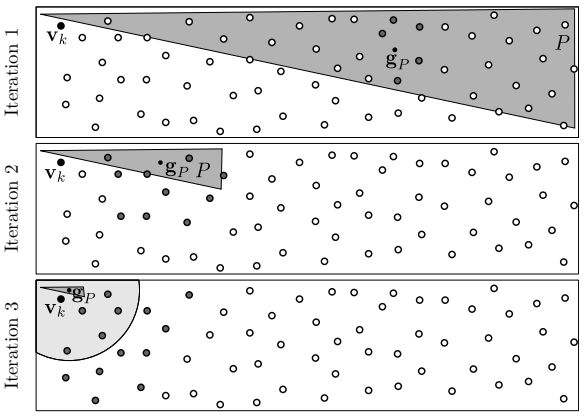


Figure 4: *Trois itérations sont nécessaires à l'Algorithme 2 pour calculer $\Omega_{k|P}$ en utilisant $p = 5$ voisins. Les sites plus sombres sont ceux calculés via les plus proches voisins. Le cercle à l'itération 3 est le rayon de sécurité du Théorème 2.*

voisins de $\mathbf{v}_k$ le polygone P_p^k (Équation 2) réduira déjà fortement sa taille, et son barycentre sera directement situé dans la zone d'intérêt.

Entrées: un polygone P, un site $\mathbf{v}_k$ de $\mathbf{V}$ et p le nombre de plus proches voisins utilisés
Sorties: P_p^k et un marqueur indiquant si $P_p^k = \Omega_{k|P}$.
Données: un marqueur par site indiquant s'il a été traité
Algorithme

```
P^k ← P
r^k ← r(P, v_k)
marquer v_k comme traité
V_p^k ← recherche_ann(v_k, p)
pour chaque site v_ℓ^k de V_p^k dans l'ordre faire
    si ‖v_ℓ^k − v_k‖ > 2r^k alors
        // le rayon de sécurité est dépassé
        retourner P^k, vrai
    marquer v_ℓ^k comme traité
    P^k ← découper(P^k, v_k, v_ℓ^k)
    r^k ← r(P, v_k)
retourner P^k, faux
```

Algorithme 3: voronoi_kP_init (P,$\mathbf{v}_k$,p) : initialisation du calcul de $\Omega_{k|P}$ via les p plus proches voisins de $\mathbf{v}_k$.

En utilisant l'Algorithme 3, le cas illustré sur la Figure 4 sera résolu par la phase d'initialisation, et l'Algorithme 2 n'a même pas besoin d'être lancé. Sur le cas pathologique illustré sur la Figure 3, le nouveau rayon de sécurité permettra rapidement d'arrêter le calcul.

3.4. Limitations

Le critère fondé sur le Théorème 2 a également quelques limitations. Tout d'abord, l'approche mixte ne permet pas toujours de réduire significativement la taille du polygone P et ainsi de recentrer le calcul dans la zone d'intérêt. Lorsque les p sites les plus proches de $\mathbf{v}_k$ sont alignés, le polygone P_5^k défini par l'Équation 2 correspondant au résultat de l'Algorithme 3 peut rester allongé, comme illustré sur la Figure 5 (en haut). Les alignements de sites peuvent régulièrement se produire, en particulier lorsque l'ensemble des sites est optimisé, comme dans les travaux de Nivoliers et Lévy [NL13]. Nous testons ce comportement en Section 5.5.

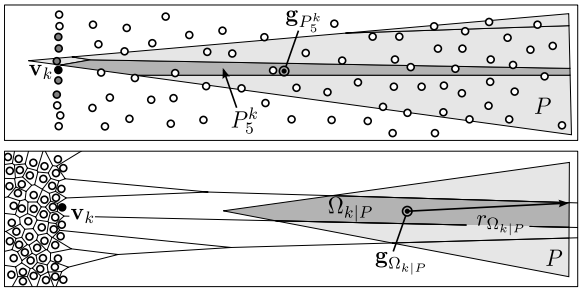


Figure 5: *Limitations de l'Algorithme 3. En haut, l'initialisation mixte ne fonctionne pas à cause de l'alignement des plus proches voisins de $\mathbf{v}_k$. En bas, la cellule restreinte à calculer est de grande taille, et le rayon du polygone $\Omega_{k|P}$ reste élevé.*

Une autre limitation apparaît lorsque $\Omega_{k|P}$ est de toute façon de grande taille par rapport à la densité des sites (Figure 5). Dans cette situation, le rayon $r(P, \mathbf{g}_P)$ calculé au fur et à mesure de l'Algorithme 2 reste minoré par le rayon de $\Omega_{k|P}$, et le test permettant d'arrêter le calcul n'éliminera que les sites très éloignés du barycentre de $\Omega_{k|P}$. Ainsi, le calcul de $\Omega_{k|P}$ peut dans le pire des cas considérer tous les sites de $\mathbf{V}$, même si le voisinage de Voronoï de $\mathbf{v}_k$ ne contient que quelques sites.

4. Découpe des coins

Nous formulons ici un nouveau critère, qui vise à limiter le nombre de sites inutiles considérés. Avec ce nouveau critère, chaque site $\mathbf{v}_\ell$ examiné lors du calcul de $\Omega_{k|P}$ sera tel que le bissecteur de $\mathbf{v}_k$ et $\mathbf{v}_\ell$ coupe le polygone en cours de calcul.

4.1. Formulation

Étant donné un polygone P et une cellule de Voronoï Ω_k, l'idée générale consiste à remarquer que si P n'est pas entièrement à l'intérieur de Ω_k, alors un de ses coins est en dehors de Ω_k. Ainsi, en calculant le point le plus proche de ce coin, nous obtenons un site $\mathbf{v}_\ell$ tel que le bissecteur de $\mathbf{v}_k$ et $\mathbf{v}_\ell$ coupe P.

Théorème 3 Soient P un polygone et $\mathbf{v}_k$ un site de $\mathbf{V}$. Soit $W \subset V$ un sous-ensemble des sites. Soit P_W^k le polygone défini par

$$P_W^k = \{\mathbf{x} \in P, \|\mathbf{x} - \mathbf{v}_k\| \leq \|\mathbf{x} - \mathbf{v}_\ell\| \text{ pour tout } \mathbf{v}_\ell \in W\}. \quad (6)$$

$P_W^k = \Omega_{k|P}$ si et seulement si, pour chaque sommet $\mathbf{x}_i$ de P_W^k, l'un des sites les plus proches de $\mathbf{x}_i$ est $\mathbf{v}_k$ ou dans W.

Démonstration Par définition, nous avons que tout point de $\Omega_{k|P}$ est au moins aussi proche de $\mathbf{v}_k$ que de tous les autres sites de V. Ainsi, si $P_W^k = \Omega_{k|P}$, alors $\mathbf{v}_k$ fait partie des plus proches voisins de tous ses points, et en particulier de ses sommets. Nous nous focaliserons donc sur l'implication inverse.

Par définition, nous savons que $\Omega_{k|P} \subset P_W^k$. Nous allons donc montrer que si tous les sommets de P_W^k ont parmi leurs sites les plus proches $\mathbf{v}_k$ ou un point de W, alors $P_W^k \subset \Omega_{k|P}$. Soit $\mathbf{x}_i$ un sommet de P_W^k. Par définition, si un site de W fait partie des sites les plus proches de $\mathbf{x}_i$, alors $\mathbf{v}_k$ également, et $\mathbf{x}_i \in \Omega_k$. Par convexité des cellules de Voronoï, tout point d'une arête de P_W^k est alors également dans Ω_k. Le bord de

P_W^k étant intégralement dans Ω_k, toujours par convexité de Ω_k, tout point de P_W^k est alors dans Ω_k. Nous avons donc $P_W^k \subset \Omega_k \cap P = \Omega_{k|P}$. □

4.2. Algorithme

Étant donné un polygone P et un site $\mathbf{v}_k \in \mathbf{V}$, calculer $\Omega_{k|P}$ en utilisant le Théorème 3 revient à couper successivement tous les coins de P en utilisant leur plus proche voisin. Dès que les plus proches voisins de tous les coins du polygone obtenu ont été traités, le calcul est terminé. L'Algorithme 4 détaille cette idée. Comme dans la Section 3, pour éviter de commencer avec un polygone trop grand, nous utilisons une formulation mixte, en utilisant l'Algorithme 3.

Entrées: un polygone P_0, un site $\mathbf{v}_k$ de $\mathbf{V}$ et p le nombre de plus proches voisins initiaux
Sorties: $\Omega_{k|P_0} = \Omega_k \cap P_0$
Données: un marqueur par site indiquant s'il a été traité et un tableau siteProche.
Algorithme

```
P ← voronoi_kP_init(P_0, v_k, p)
pour chaque sommet x_i ∈ P faire
    siteProche[x_i] ← recherche_ann(x_i, 1)
tant que tous les sites ne sont pas traités faire
    si siteProche[x_i] est traité pour tout x_i ∈ P alors
        retourner P
    sinon
        soit x_i tel que siteProche[x_i] n'est pas traité
        v_ℓ ← siteProche[x_i]
        P ← découper(P, v_k, v_ℓ)
        pour chaque nouveau sommet x_j ∈ P faire
            siteProche[x_j] ← recherche_ann(x_j, 1)
```

Algorithme 4: voronoi_kP_coins($P, \mathbf{v}_k, p$) : calcul de $\Omega_{k|P}$ par découpe successive des coins. L'arrêt de l'algorithme est dicté par le critère défini dans le Théorème 3.

4.3. Limitations

Dans les situations où l'Algorithme 2 se comporte bien, cette nouvelle approche peut se trouver plus lente, car un plus grand nombre de requêtes de plus proche voisinage sera effectué. En effet, si le nombre de voisins demandé reste faible, chaque requête a une complexité de $O(\log n)$ quel que soit le nombre de voisins requis. Ainsi, dix requêtes de plus proches voisins seront environ dix fois plus coûteuses qu'une seule requête demandant dix plus proches voisins. Expérimentalement, le comportement de l'Algorithme 4, semble toutefois rester compétitif.

En terme de pire cas, il peut également arriver que pour chaque site $\mathbf{v}_k$, le nombre d'autres sites à traiter pour calculer $\Omega_{k|P}$ soit $O(n)$. La base du problème est illustrée sur la Figure 6. Imaginons un repère (x,y) du plan 2D, avec $\mathbf{v}_k$ en $(0,0)$, et le polygone P est un triangle dont un sommet est proche de $\mathbf{v}_k$ et les deux autres ont des abscisses très grandes. Si p est le nombre de sites utilisés pour l'initialisation via l'Algorithme 3, plaçons p sites sur la droite $x = 0$ avec $y \in [-1,1]$. De cette façon, l'approche mixte échouera à réduire la taille de P. Plaçons ensuite le reste des sites sur la droite $y = 2$, aux abscisses $x \in \{2^i\}_{i=0}^n$. Le polygone étant très long, les coins ayant des sites les plus proches non traités auront pour site le plus proche le site situé en $(2,2^n)$. Le bissecteur de ce site avec $\mathbf{v}_k$ coupera grossièrement le polygone en deux dans le sens des abscisses, et les nouveaux coins créés auront pour site le plus proche $(2,2^{n-1})$, etc. Le nombre de sites utilisés pour calculer $\Omega_{k|P}$ est donc $O(n)$. Cette construction peut se généraliser pour faire en sorte que tous les sites soient dans cette situation : en répartissant les sites par groupes de p le long de deux droites perpendiculaires, avec la même répartition exponentielle que précédemment.

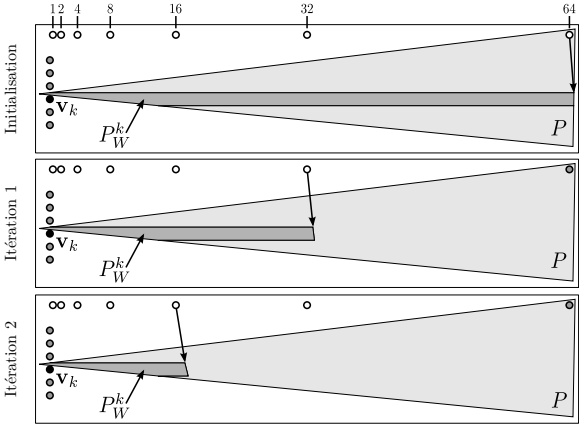


Figure 6: *Pire cas pour l'Algorithme 4. La répartition des sites fait que l'initialisation par l'Algorithme 3 est inutile, et si $O(n)$ sites sont disposés le long d'une droite horizontale, avec une répartition exponentielle, le nombre de sites traités pour calculer $\Omega_{k|P}$ est $O(n)$ car tous les sites de la droite sont considérés.*

5. Performances

Nous montrons ici différents scénarios de test, et les performances associées des différents algorithmes proposés. Notre analyse est principalement focalisée sur l'évolution des performances en fonction du nombre et de la disposition des sites, plutôt que sur la taille du maillage fourni en entrée. La taille du maillage intervient plutôt au niveau global de la propagation, et le comportement des algorithmes que nous proposons en fonction de ce paramètre reste similaire à celui des algorithmes classiques pour le calcul de diagrammes de Voronoï restreints [Niv12].

Pour les résultats de Lévy et Bonneel [LB12], nous utili-

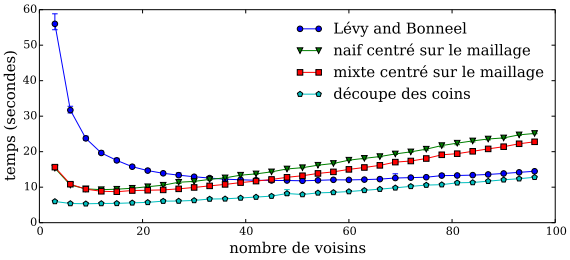


Figure 7: *Performances des algorithmes en fonction du nombre de plus proches voisins choisis. Le maillage utilisé est le David, avec 20000 sites.*

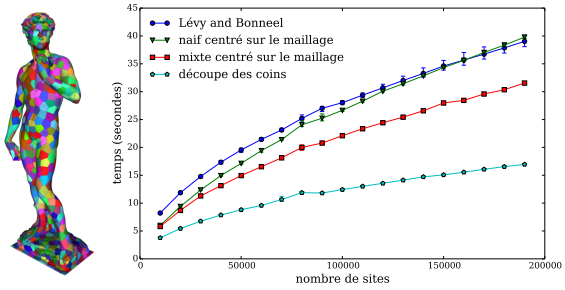


Figure 8: *Performances pour les applications de Lévy et Bonneel [LB12] : le David est maillé par 60k triangles.*

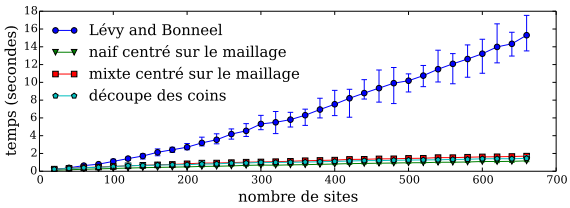


Figure 9: *Performances des algorithmes dans le cas de sites répartis uniformément dans la boîte englobante de la surface. Le maillage utilisé est le même que pour la Figure 8*

sons notre propre implémentation de leurs travaux. Il est à noter que notre implémentation n'est pas parallèle, contrairement à la leur, et qu'en pratique, leurs performances sont à diviser grossièrement par le nombre de processeurs. Nous nous intéressons ici plutôt au comportement des algorithmes présentés, et ces algorithmes étant très similaires, nos approches pourraient se paralléliser de la même manière que la leur. Pour calculer les plus proches voisins, nous utilisons la bibliothèque FLANN [ML09] qui est un peu plus rapide que la bibliothèque ANN [MA97]. Nous ne prenons pas en compte dans nos mesures le temps de construction de la structure spatiale associée à la recherche de plus proches voisins.

5.1. Paramètres

Tous les tests que nous présentons sont réalisés dans $\mathbb{R}^6$. Les deux surfaces que nous utilisons (un cube, et le David du projet Michelangelo) sont plongées dans $\mathbb{R}^6$ en ajoutant aux coordonnées spatiales de $\mathbb{R}^3$ les coordonnées de la normale moyenne de la surface au niveau du sommet. Pour chaque test, nous lançons dix fois le calcul sur dix ensembles de sites de même taille tirés au hasard. Les courbes présentent la valeur moyenne, et au niveau de chaque mesure, un intervalle vertical montre l'écart entre le temps minimal et le temps maximal obtenu sur les dix itérations.

Un paramètre utilisé dans tous les algorithmes est le nombre p de plus proches voisins initiaux utilisés pour les requêtes de plus proche voisinage. La Figure 7 illustre les performances des algorithmes en fonction de ce paramètre. À partir de cette étude, nous avons donc choisi 15 plus proches voisins pour nos algorithmes, et 50 pour celui de Lévy et Bonneel.

5.2. Sites répartis sur la surface

Lévy et Bonneel ont conçu leur algorithme pour répartir des sites sur une surface, à des fins de remaillage. Nous commençons donc par comparer nos algorithmes au leur dans ce cas de figure. Les sites sont tirés indépendamment au hasard sur la surface avec une probabilité uniforme. Les résultats en utilisant le David sont présentés sur la Figure 8. D'une manière générale, l'Algorithme 4 est le plus efficace, et ne semble pas handicapé par le fait que plus de requêtes de plus proche voisinage sont réalisées. Les algorithmes centrés sur le maillage de la Section 3 sont grossièrement équivalents à celui de Lévy et Bonneel. Les performances des algorithmes restent toutefois dans le même ordre de grandeur.

5.3. Site répartis dans la boîte englobante de la surface

Comme évoqué en Section 2.4.3, le rayon de sécurité de l'algorithme de Lévy et Bonneel reste grand lorsque les sites sont éloignés de la surface. Nous illustrons ce cas simplement en tirant chaque site indépendamment au hasard dans la boîte englobante de la surface. De nouveau, nous utilisons le David. Nous utilisons peu de sites pour ce test, car les performances de l'algorithme de Lévy et Bonneel se dégradent rapidement. Les résultats sont illustrés sur la Figure 9. En comparaison, tous les algorithmes que nous proposons continuent à présenter des performances acceptables.

5.4. Faces de grande taille

Une limitation de l'approche naïve de l'Algorithme 2 évoquée en Section 3.2 est le cas des faces de grande taille. Dans ce cas, plusieurs itérations sont nécessaires avant d'atteindre les sites de la zone d'intérêt. Pour exhiber ce défaut, nous

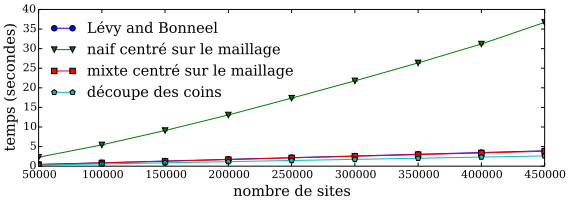


Figure 10: *Performances des algorithmes dans le cas de sites répartis uniformément sur un cube, ayant des faces de grande taille par rapport à la densité des sites.*

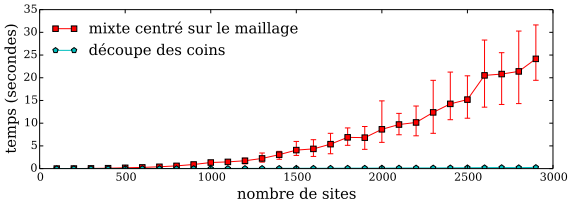


Figure 11: *Performances de l'approche mixte centrée sur le maillage par rapport à l'approche par découpe de coins, dans le cas de sites répartis uniformément sur dix droites alignées sur les axes.*

utilisons comme maillage un simple cube, et répartissons les sites uniformément sur sa surface. La Figure 10 montre que l'Algorithme 2 est effectivement bien plus coûteux, et que l'approche mixte proposée en Section 3.3 permet effectivement de remédier à ce problème. Même avec cette modification, l'algorithme de Lévy et Bonneel et celui par découpe de coin restent plus rapides dans ce cas de figure, avec des performances très similaires. Par rapport à la Figure 8, le comportement de tous nos algorithmes semble impacté, par rapport à celui de Lévy et Bonneel qui redevient compétitif.

5.5. Alignements de sites

En Section 3.4, nous mentionnons que l'approche mixte fondée sur l'Algorithme 2 reste inopérante lorsque les sites sont alignés. Nous utilisons le cube comme surface car les faces sont grandes par rapport à la densité des sites. Pour générer des sites alignés, nous tirons au hasard dix droites alignées sur les axes. Chaque droite est définie en tirant au hasard l'indice de la coordonnée restant libre, et les $d-1$ coordonnées fixées. Chaque point est ensuite généré en tirant au hasard une droite, et en générant au hasard la coordonnées manquante le long de l'axe de la droite. Le résultat est présenté sur la Figure 11. Nous ne testons ici que l'approche mixte centrée sur le maillage et l'approche par découpe des coins, les deux autres approches étant hors course du fait de la taille des faces et de l'éloignement des sites par rapport aux faces. Le résultat montre bien les difficultés de l'approche mixte, et les performances de l'algorithme mixte centré sur le maillage se dégradent rapidement.

6. Conclusion

Les deux nouvelles approches que nous proposons rendent accessible le calcul de diagrammes de Voronoï restreints pour des configurations de sites plus génériques. L'Algorithme 4 en particulier, fondé sur la découpe successive des coins des polygones, a le comportement le plus robuste pour les cas de figure que nous avons envisagés.

Plusieurs pistes semblent envisageables pour poursuivre ces travaux. Tout d'abord, tous nos tests sont réalisés dans $\mathbb{R}^6$ en ajoutant les coordonnées de leur normale aux coordonnées des sommets. Il serait nécessaire d'étudier le comportement de nos algorithmes sur des surfaces plus générales, en particulier pour des dimensions plus élevées. Du point de vue théorique, il serait également nécessaire de déterminer si des garanties en terme de complexité moyenne ou dans le pire des cas peuvent être obtenues en fixant des hypothèses sur la surface et l'ensemble de sites. Nous ne fournissons ici en effet que des performances expérimentales, sur les cas de figure que nous avons imaginés. Enfin, il reste beaucoup de place pour améliorer les algorithmes proposés. Par rapport à l'Algorithme 4, il serait par exemple pertinent de voir si le calcul peut être accéléré en testant les plus proches voisins ailleurs que sur les coins. Il est également envisageable de chercher à déterminer globalement, étant donné un polygone, l'ensemble des cellules le coupant, sans se focaliser sur le calcul d'une intersection en particulier. Cette approche éviterait de calculer plusieurs fois les mêmes plus proches voisins.

7. Remerciements

Ce projet a été supporté en partie par la Politique Scientifique Belge via le projet PAI P7/02, et par la région wallonne WIST3 via le projet DOMHEX.

Références

[AB99] AMENTA N., BERN M. : Surface reconstruction by Voronoi filtering. *Discrete and Computational Geometry. Vol. 22*, Num. 4 (1999), 481–504.

[BDH96] BARBER C., DOBKIN D., HUHDANPAA H. : The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS). Vol. 22*, Num. 4 (1996), 469–483.

[CGA] CGAL, Computational Geometry Algorithms Library. http ://www.cgal.org.

[EDD*95] ECK M., DEROSE T., DUCHAMP T., HOPPE H., LOUNSBERY M., STUETZLE W. : Multiresolution analysis of arbitrary meshes. In *Proceedings of the 22nd*

annual conference on Computer graphics and interactive techniques (1995), ACM, pp. 173–182.

[ES97] EDELSBRUNNER H., SHAH N. : Triangulating topological spaces. *International Journal of Computational Geometry and Applications. Vol. 7*, Num. 4 (1997), 365–378.

[FBF77] FRIEDMAN J. H., BENTLEY J. L., FINKEL R. A. : An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software (TOMS). Vol. 3*, Num. 3 (1977), 209–226.

[KWR97] KUNZE R., WOLTER F., RAUSCH T. : Geodesic voronoi diagrams on parametric surfaces. In *Computer Graphics International, 1997. Proceedings* (1997), IEEE, pp. 230–237.

[LB12] LÉVY B., BONNEEL N. : Variational anisotropic surface meshing with voronoi parallel linear enumeration. In *IMR-21st International Meshing Roundtable* (2012).

[LCT11] LIU Y., CHEN Z., TANG K. : Construction of iso-contours, bisectors and voronoi diagrams on triangulated surfaces. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, Num. 99 (2011), 1–1.

[Llo82] LLOYD S. : Least squares quantization in pcm. *IEEE Transactions on Information Theory. Vol. 28*, Num. 2 (1982), 129–137.

[MA97] MOUNT D. M., ARYA S. : ANN : A library for approximate nearest neighbor searching. In *CGC Workshop on Computational Geometry* (1997), pp. 33–40.

[ML09] MUJA M., LOWE D. G. : Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP'09)* (2009), INSTICC Press, pp. 331–340.

[Niv12] NIVOLIERS V. : *Échantillonnage pour l'approximation de fonctions sur des maillages*. PhD thesis, LORIA – Université de Lorraine, Novembre 2012.

[NL13] NIVOLIERS V., LÉVY B. : Approximating functions on a mesh with restricted voronoï diagrams. *Computer Graphics Forum. Vol. 32*, Num. 5 (2013), 83–92.

[NYL12] NIVOLIERS V., YAN D.-M., LÉVY B. : Fitting Polynomial Surfaces to Triangular Meshes with Voronoi Squared Distante Minimization. *Engineering with Computers* (2012).

[OBSC09] OKABE A., BOOTS B., SUGIHARA K., CHIU S. N. : *Spatial tessellations : concepts and applications of Voronoi diagrams*, vol. 501. Wiley. com, 2009.

[PC06] PEYRÉ G., COHEN L. : Geodesic remeshing using front propagation. *International Journal of Computer Vision. Vol. 69*, Num. 1 (2006), 145–156.

[SH74] SUTHERLAND I., HODGMAN G. : Reentrant polygon clipping. *Communications of the ACM. Vol. 17*, Num. 1 (1974), 32–42.

[YLL*09] YAN D., LÉVY B., LIU Y., SUN F., WANG W. : Isotropic remeshing with fast and exact computation of restricted voronoi diagram. In *Computer graphics forum* (2009), vol. 28, Wiley Online Library, pp. 1445–1454.