Table des matières

Chapitre 1 Introduction générale

1. Introduction

L'information spatiale aujourd'hui fait partie de notre quotidien, par exemple, lorsqu'on veut se rendre à un endroit, on programme machinalement le GPS de voiture. Lorsque l'on planifie ses vacances et qu'on recherche un hôtel autour duquel il y a les meilleures commodités en termes de transport, de proximité de la plage, du parc de loisir et des centres commerciaux. Lorsqu'on utilise sa *smartWatch* pour faire ses 10.000 pas quotidiens etc.

L'information spatiale n'est pas utilisée que par l'homme de tous les jours mais par les professionnels aussi. Par les ingénieurs des Directions de l'Urbanisme et de la Construction pour dresser le plan d'occupation du sol. Par la protection civile pour la gestion des risques majeurs. Par le Centre de Recherche en Astronomie Astrophysique et Géophysique (CRAAG) dans leur mission de surveillance des séismes sur le territoire. Par les publicitaires avec la publicité géolocalisée. Et bien entendue par les cartographes.

Parmi les opérations utilisées dans un système spatial, on retrouve la recherche de proximité, le calcul de l'itinéraire le plus court, la recherche de sa position ou encore la recherche d'intersection. Tous ces calcules relèvent de la géométrie ou ce qu'on appelle la géométrie algorithmique.

La géométrie algorithmique (en anglais : *computational geometry*), comme son nom l'indique, a pour but le développement et l'étude d'algorithmes pour résoudre des problèmes de géométrie. C'est une discipline très ancienne, car Euclide en faisait déjà sans le savoir ! De manière plus récente, la géométrie algorithmique s'est développée á partir des années 70 en réponse á des problèmes de conception assistée par ordinateur (C.A.O.), de robotique, de géographie (planification de trajectoires, . . .), d'informatique graphique, de conception de circuits intégrés, etc.

Les algorithmes géométriques les plus utilisés sont : l'enveloppe convexes, l'enveloppe concave, le triangle/rectangle englobant, la triangulation de Delaunay et le fameux diagramme de Voronoi.

Le diagramme de voronoi est un découpage du plan en cellules (régions adjacentes) à partir d'un ensemble discret de points appelés « germes ». Chaque cellule renferme un seul germe, et forme l'ensemble des points du plan les plus proches de ce germe que d'aucun autre. Il existe plusieurs algorithmes pour le calcul du diagramme de Voronoi, tel que L'algorithme de Green et Sibson qui est un algorithme incrémental. Il maintient un diagramme de Voronoï en ajoutant les points un à un. Sa complexité est O (n^2).

Shamos et Hoey ont montré en 1975 qu'il est possible de calculer le diagramme de Voronoï d'un ensemble de n points du plan dans le temps O (n log n). Ils utilisent pour cela un raisonnement par récurrence : supposons que l'on puisse séparer l'ensemble S en deux sous-ensembles de même cardinal n/2, séparés par une droite verticale : l'ensemble G des points à gauche et l'ensemble D des points à droite. Les diagrammes respectifs de ces sous-ensembles, V(G) et V(D), sont connus, et on peut les fusionner. On a ainsi un algorithme de type divisé pour régner, dont la complexité est O (n log n).

L'algorithme de Fortune (1987) a été démontré comme asymptotiquement optimal. Il est en O (n log n) en temps et en O(n) en espace mémoire. L'idée générale consiste à balayer le plan De haut en bas avec une ligne verticale (c'est un algorithme de sweep line) ; on construit le diagramme de Voronoï progressivement.

Quel est le meilleur algorithme ? Comment en choisir un ? Est-ce le moins complexe, le plus simple, le moins gourmand en espace mémoire, le plus flexible, etc. C'est les critères de sélection déterminant quel type d'algorithme fera le travail efficacement.

2. Problématique

La question naturelle que nous sommes en droit de nous poser avant d'aborder ce mémoire est la suivante : d'où vient l'intérêt porté au diagramme de Voronoi ?

Un premier élément de réponse peut être donné en considérant la simplicité de la définition du diagramme de Voronoi donnée dans Preparata [1] : Soit S, un ensemble de N points dans l'espace. Pour chaque point pi de S, quel est l'ensemble des points (x ;y ; z) dans l'espace qui sont plus proches de pi que de n'importe quel autre point de S ? La solution au problème posé est de partitionner l'espace en régions de Voronoi.

Le principe étant simple, la mise en pratique est plutôt complexe. Dans la littérature, il existe de nombreux algorithmes pour calculer le diagramme de Voronoi.

Mais les données géographiques étant en générales volumineuses. Quel est le meilleur algorithme en termes de temps de calcul et d'efficacité ?

3. Plan de travail

Tout d'abord on a commencé par la documentation afin d'acquérir des notions spatiales et géométrique. Deuxièmement on a appris à maitriser l'outil QGIS et le langage PyQGIS. Troisièmement on a étudié l'algorithme de fortune et Quatrièmement a implémenté l'algorithme de fortune en PyQGIS. Après on a testé et analysé les résultats que nous avons obtenus et finalement on a comparé ces résultats avec les autres

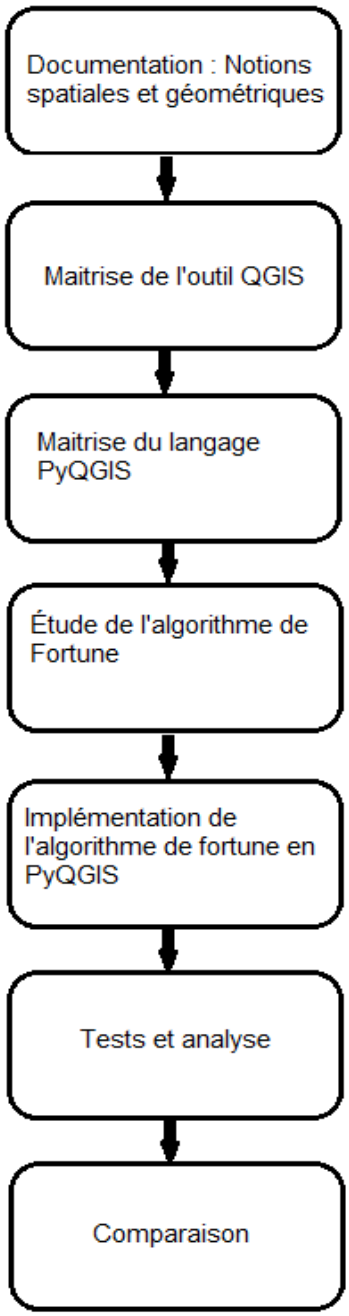


Figure 1. Plan de travaille

Figure 2. Plan de travaille

algorithmes qui existent.

4. Contenue des chapitres

Ce document est structuré comme suit : le deuxième chapitre donne un bref aperçu sur les notions spatiales et les algorithmes existants pour le calcul du diagramme de voronoi, le troisième chapitre est consacré à l'étude de l'algorithme de fortune et en présent les propriétés. On y présent les résultats ainsi qu'une comparaison avec les autres algorithmes enfin une conclusion générale conclut ce mémoire.

Chapitre 2 Généralités sur le spatial et les algorithmes géométriques

1. Introduction

Ce chapitre est dédié aux notions de base de l'information spatiale. On s'intéressera aussi la géométrie et surtout à l'une de ses branches à savoir la géométrie algorithmique. On analyse ensuite les problèmes de base de cette discipline. Enfin on s'attardera plus sur le diagramme de Voronoi

2. Notions spatiales

2.1. L'espace

L'espace se présente dans l'expérience quotidienne comme une notion de géométrie et de physique qui désigne une étendue, abstraite ou non, ou encore la perception de cette étendue. Conceptuellement, il est le plus souvent synonyme de contenant aux bords indéterminés. Le phénomène reste en lui-même indéterminé car nous ne savons pas s'il manifeste une structure englobante rassemblant toutes les choses et les lieux ou bien s'il ne s'agit que d'un phénomène dérivé de la multiplicité des lieux.

Avant d'être un concept physico-mathématique, l'espace a d'abord été une interrogation majeure des philosophes. De nos jours l'espace, qui semble s'être retiré du champ philosophique, prend de nombreux sens précis et propres à de multiples disciplines scientifiques dérivées de la géométrie. L'espace figure alors, de manière générale, un Tout ensembliste, mais structuré : le domaine de travail.

On parle encore d'espace pour désigner une certaine distance (l'espace entre deux personnes), une certaine surface (ce parc naturel couvre un espace considérable) ou un certain volume (ce placard occupe un grand espace). [C]

2.2. L'information spatiale

C'est une information sur la réalité localisée dans l'espace ; elle exprime les propriétés spatiales par le géo référencement et le voisinage, des propriétés thématiques ou temporelles. Elle est généralement modélisée dans les bases de données par :

- Un Point.
- Une Poly ligne : suite de points reliés par des segments.
- Un Polygone : suite de points fermée reliés par des segments.
- Un Champ (par exemple un lieu).

2.3. La couche

Est un ensemble d'informations sur le même thème, ex : les routes, les bâtiments, les points d'eau. Une couche représente une classe ou une table dans une BDD.

2.4. Organisation des données dans une couche

Les données sont organisées sous forme d'enregistrements. Chaque enregistrement est doté de deux types d'informations : alphanumériques dits attributaires(Nom, Surface, Nombre d'habitant) et spatiales (ensemble de coordonnées). En générale, ils sont conservés séparément.

2.5. Coordonnées géographiques

Par coordonnées géographiques (ou encore « repères géographiques ») d'un lieu sur la Terre, on entend un système de trois coordonnées qui sont le plus souvent : la latitude, la longitude et l'altitude (ou l'élévation) par rapport au niveau moyen de la mer (élévation ortho métrique) ou par rapport à une surface de référence, en général ellipsoïde (élévation ellipsoïdale).Ces coordonnées géographiques découlent d'un système géodésique qui modélise la forme de la Terre. Pour se repérer à la surface de la planète, on peut utiliser des systèmes de représentation graphique appelés « repères cartographiques ».

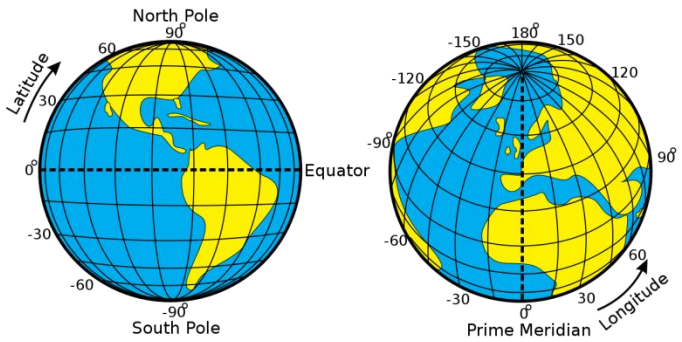


Figure 3-Repéres cartographiques [37]

2.6. QGIS

QGIS est un logiciel SIG (système d'information géographique) libre multiplateformes publié sous licence GPL (General Public License). QGIS dispose de toutes les fonctionnalités et performances que proposent les logiciels SIG payants. L'une des grandes forces de QGIS est de pouvoir utiliser des centaines de systèmes de projection géographiques à la volée tels que WGS84, Lambert 93 ou ED50. En outre, depuis la version 0.9, il possède un vrai moteur de scripts basé sur Python. Ceci permet tout à la fois de créer des modules plus simplement qu'en C++, mais aussi de construire de véritables applications. Cette possibilité passe par PyQt, le pont entre Python et la bibliothèque graphique Qt4.

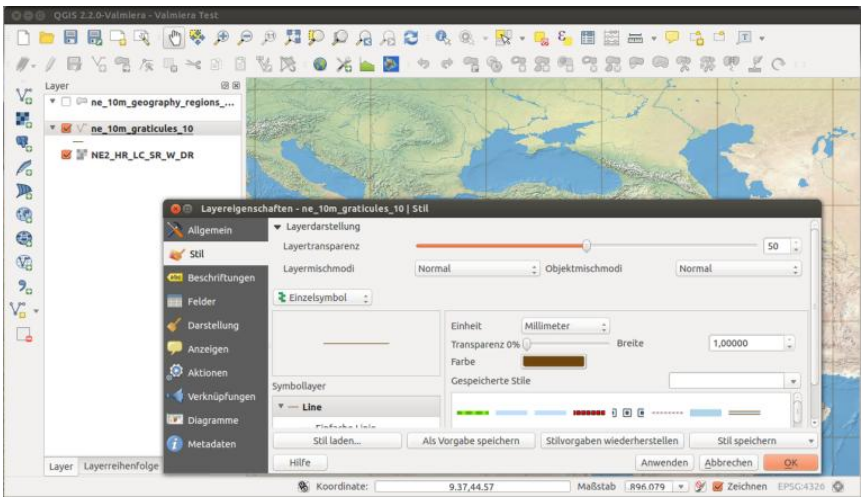


Figure 4-Interface QGIS [8]

3. La Géométrie

Dans sa forme la plus simple, la géométrie est l'étude mathématique des formes et de l'espace. La géométrie peut traiter des formes plates bidimensionnelles, telles que des carrés et des cercles, ou des formes tridimensionnelles avec une profondeur, telles que des cubes et des sphères ou même des formes a des dimensions supérieurs a trois dimensions.

3.1 Les Bases

Un point est représenté par un point et indique un emplacement dans l'espace. Une ligne est un ensemble de points droits qui s'étend à jamais dans les deux sens, comme indiqué par des flèches aux deux extrémités. Les rayons sont des lignes qui se terminent

sur un côté. Les segments se terminent des deux côtés. Les plans sont des surfaces qui s'étendent pour toujours dans toutes les directions.

Un plan s'étend infiniment en deux dimensions. Il n'a pas d'épaisseur. Un exemple de plan est un plan de coordonnées. Un plan est nommé par trois points du plan qui ne sont pas sur la même ligne. Ci-dessous, nous voyons le plan ABC.

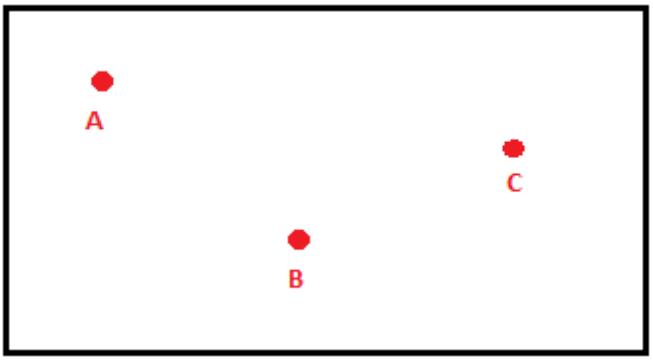


Figure 5. plan à deux dimensions

Un espace s'étend à l'infini dans toutes les directions et est un ensemble de tous les points en trois dimensions. Vous pouvez penser à un espace comme à l'intérieur d'une boîte [24].

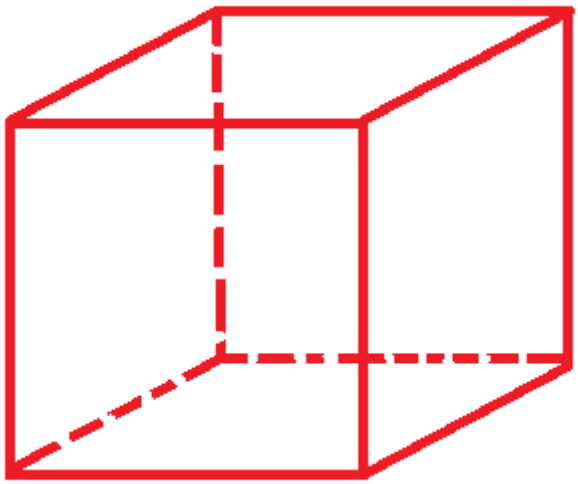


Figure 6. Cube

La géométrie algorithmique est une science encore jeune et génératrice de nombreux problèmes faisant intervenir des entités géométriques : points, polyèdres, sphères….ces problèmes sont souvent posés en termes simples et pour tant les solutions quand elles existent utilisent souvent des outils complexes issus des mathématiques.

L’optimalité recherchée dans les solutions est souvent la source de la principale difficulté. Mais ce souci d’efficacité est justifié par la grande taille des problèmes traités.

4. Géométrie algorithme

La géométrie algorithmique est une branche de l'informatique consacrée à l'étude d'algorithmes pouvant être énoncés en termes de géométrie. Certains problèmes purement géométriques découlent de l’étude des algorithmes de calcul géométrique, et de tels problèmes sont également considérés comme faisant partie de la géométrie de calcul. La géométrie informatique moderne est un développement récent, mais l’un des domaines informatiques les plus anciens, avec une histoire remontant à l’Antiquité.

La complexité de calcul est au cœur de la géométrie de calcul, et revêt une grande importance pratique si des algorithmes sont utilisés sur de très grands ensembles de données contenant des dizaines, voire des centaines de millions de points. Pour de tels ensembles, la différence entre O (n2) et O (n log n) peut être la différence entre les jours et les secondes de calcul.

Les progrès de l’infographie et de la conception et de la fabrication assistées par ordinateur (CAO / FAO) ont été les principaux moteurs du développement de la géométrie algorithmique en tant que discipline, mais de nombreux problèmes de géométrie computationnelle sont de nature classique et peuvent provenir de la visualisation mathématique.

La robotique (problèmes de planification de mouvement et de visibilité), les systèmes d’information géographique (SIG) (localisation et recherche géométriques, planification d’itinéraire), la conception de circuits intégrés (conception et vérification de géométrie IC), l’ingénierie assistée par ordinateur (CAE) sont d’autres applications importantes de la géométrie informatique. (Génération de maillage), vision par ordinateur (reconstruction 3D).

4.1 Enveloppe convexe

Qu'est-ce qu'une enveloppe convexe d'un ensemble de points ? C'est la surface qui minimise la zone qui contient tous les points, sans que l'angle entre deux arêtes voisines ne dépasse 180°.

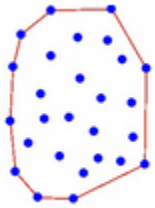


Figure 7. Enveloppe convexe

4.2 L'enveloppe concave :

Une enveloppe concave d'un ensemble de points obéit aux mêmes principes que l'enveloppe convexe, mais autorise tous les angles. Tous les ensembles d'enveloppes situés entre l'enveloppe convexe et celle qui minimise complètement la surface sont considérés comme concaves. [31]

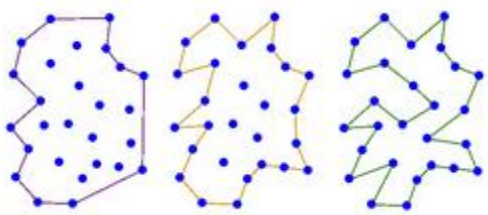


Figure 8. Diverses enveloppes concaves possibles

4.3. Le Rectangle englobant

En géométrie, le rectangle englobant pour un ensemble de points (S) en N dimensions est la boîte ayant les mesures minimales (surface, volume ou hyper volume dans les dimensions supérieures) dans laquelle se trouvent tous les points. [40]

4.4. Triangulation de Delaunay

En mathématiques et plus particulièrement en géométrie algorithmique, la triangulation de Delaunay d'un ensemble P de points du plan est une triangulation DT(P) telle qu'aucun point de P n'est à l'intérieur du cercle circonscrit d'un des triangles de DT(P).[41]

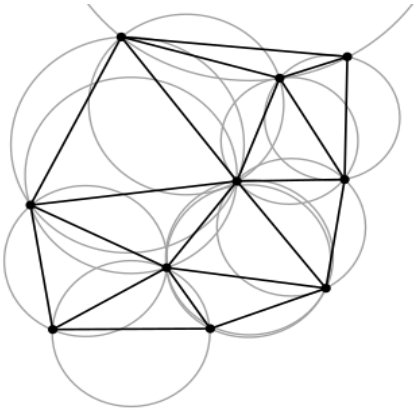


Figure 9. Triangulation de Delaunay [41]

4.5 Diagramme de Voronoi

Énoncé du problème : supposons que nous ayons un nombre fini de points distincts dans le plan. Nous nous référons à ces points en tant que sites. Nous souhaitons partitionner le plan en régions disjointes appelées cellules, chacune contenant exactement un site, de sorte que tous les autres points d'une cellule soient plus proches du site de cette cellule que de tout autre site. Un exemple de diagramme de Voronoï:

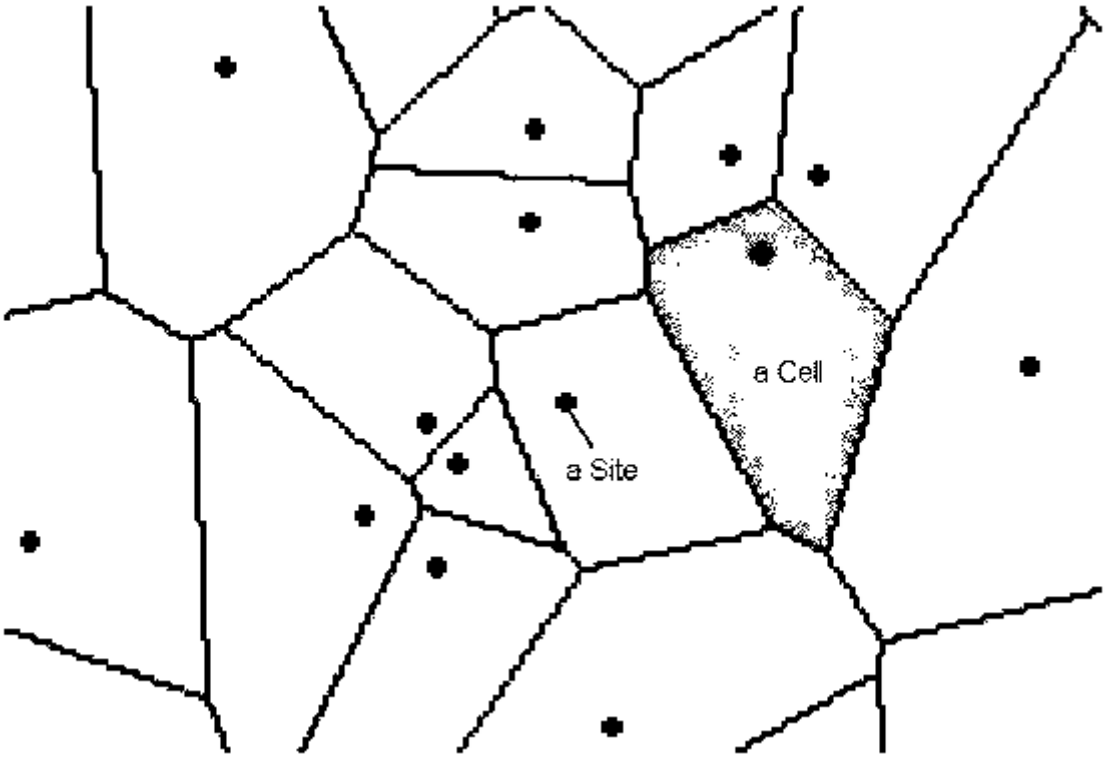


Figure 10. Diagramme de Voronoi

4.5.1 Histoire du Diagramme de Voronoi :

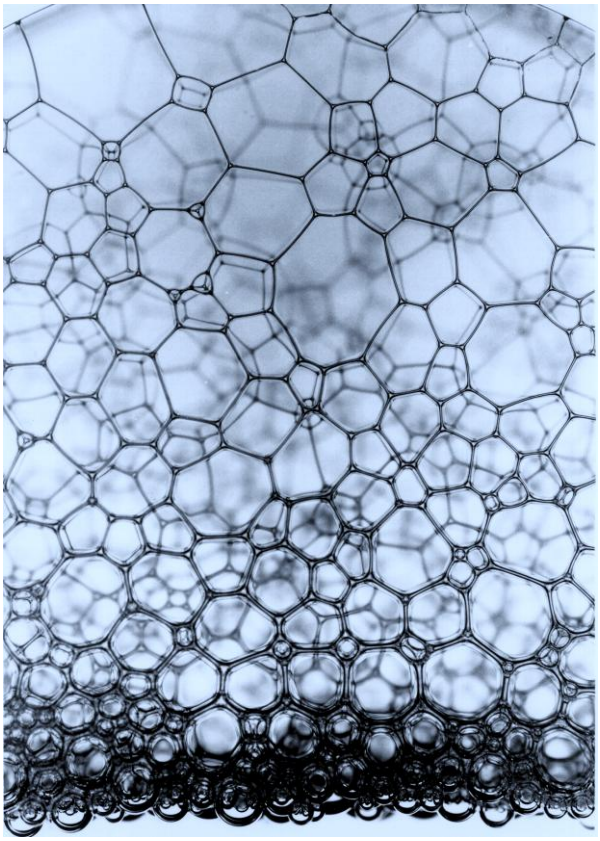
Les diagrammes de Voronoï ont une longue histoire, remontant au 17ème siècle. Les travaux de Descartes sur un partitionnement de l'univers en «tourbillons» constituent l'une des premières références connues au sujet. Même si Descartes ne définit pas explicitement ses vortexes de la même manière que les cellules de Voronoï, son travail est très similaire conceptuellement [8].

Les jeunes Dirichlet et M. G. Voronoi, deux mathématiciens allemands, ont été chargés de formaliser le concept moderne du diagramme de Voronoï [7]. Dirichlet est né en 1805 et, dans ses travaux sur les formes quadratiques, il a apporté l'une des premières contributions significatives au domaine des diagrammes de Voronoï. En effet, c'est grâce à lui que les diagrammes de Voronoï sont également connus sous le nom de mosaïques de Dirichlet. Avant sa mort en 1859, Dirichlet avait formalisé le concept du diagramme de Voronoï dans les cas à deux et trois dimensions [7]. Les travaux de M. G. Voronoi en 1908 ont formalisé le cas n dimensionnel et ont donné aux diagrammes de Voronoï le nom que nous utilisons couramment aujourd'hui. Le dual bidimensionnel du diagramme de Voronoï au sens théorique du graphe est la triangulation de Delaunay [8]. Les travaux sur les triangulations de Delaunay (ou, alternativement, sur les pavés de Delaunay) ont été réalisés par le mathématicien français Charles Delaunay avant 1872. Dans une triangulation de Delaunay, deux sites sont connectés s'ils partagent une limite de cellule de diagramme de voronoi.

Une autre définition, plus conforme à l'œuvre originale de Delaunay, est que deux sites sont connectés si et seulement s'ils se trouvent sur un cercle dont l'intérieur ne contient aucun autre site [14]. En 1909, BT Boldyrev, scientifique russe, utilisa les "polygones de zone d'influence" dans ses travaux de géologie [5]. Les diagrammes de Voronoï ont été utilisés en météorologie par Thiessen en 1911 pour aider à modéliser les précipitations moyennes [7]. Un travail influent en cristallographie a été réalisé à l'aide de diagrammes de Voronoï par un Allemand nommé Paul Niggli en 1927. En 1933, les physiciens EP Wigner et F. Seitz ont effectué d'importantes recherches à l'aide de diagrammes de Voronoï en physique. Les diagrammes de Voronoï jouent un rôle clé dans les recherches effectuées en physique, en écologie, en anatomie et en astronomie au cours des années 1900.

De nos jours le diagramme de voronoi forment une partie importante de la géométrie algorithmique. Le diagramme qui porte le nom de voronoi serait apparu pour la première fois dans des écrits de Descartes en 1664 pour l'étude des planètes de notre système solaire. Mais ce n'est que bien plus tard en 1850 sous l'impulsion de Dirichlet puis en 1980 sous l'impulsion de Voronoi que ce diagramme a trouvé ses lettres de noblesse.

Ce n'est qu'en 1934 que Delaunay introduit le diagramme qui porte son nom. Les diagrammes de voronoi et de Delaunay sont intimement liés puisqu'ils forment deux graphes duaux.



4.5.2. Diagramme de voronoi dans la géométrique

1. Les régions de Voronoi se retrouvent dans la nature : les cellules d'un tissu dans le plan forment approximativement une partition de Voronoi [14], ainsi que les alvéoles des abeilles [3] (Figure 1), les molécules chimiques [7] ou encore les cristaux [1].

2. Les régions de Voronoi ont des propriétés mathématiques intéressantes et surprenantes. C'est ce qui fait penser que les diagrammes de Voronoi sont l'une des plus fondamentales constructions géométriques découvertes pour un ensemble de points [8].

3. Il existe des algorithmes efficaces pour construire les diagrammes de Voronoi. De plus ces diagrammes sont très puissants pour résoudre en temps optimal de nombreux problèmes de géométrie algorithmique [1] : enveloppe convexe, problèmes de proximité, arbre de poids minimum (minimum spaning tree), graphe de Gabriel... [1].

4. La structure duale du diagramme de Voronoi (le diagramme de Delaunay) admet elle aussi de très bonnes propriétés géométriques et topologiques [1]. De plus le graphe

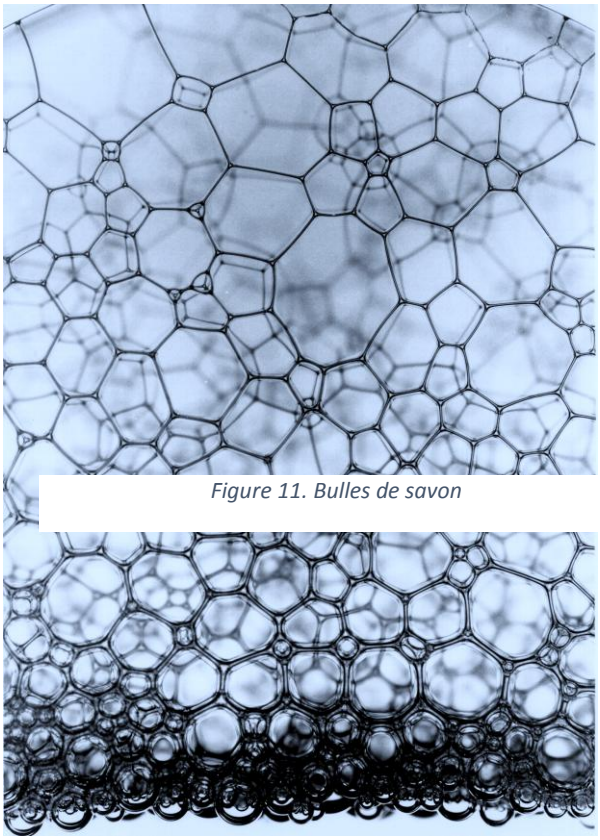


Figure 11. Bulles de savon

de Delaunay est le graphe qui contient toutes les informations locales [8].

4.5.3. Utilisations :

Dans de nombreux domaines. Nous pouvons citer par exemple :

Les diagrammes de Voronoi et de Delaunay sont très utiles

1. la sociologie cellulaire qui est l'étude structure-fonction des cellules au sein d'un tissu cellulaire [15, 16],

2. la reconstruction de surfaces á partir d'un modèle numérique de terrain [17,18, 19],
3. l'aide de la théorie des IFS (Iterated Function System la compression des images) [20, 21, 22],
4. la percolation au sein du diagramme de Delaunay ou de Voronoi [27],
5. la reconstruction d'un volume [24, 25, 26],
6. l'interpolation de données tridimensionnelles obtenues par stéréo [20],
7. la représentation et la segmentation des images 2D ou 3D [28],
8. la squelettisation d'objets bidimensionnels ou tridimensionnels [4],
9. la classification vectorielle [36],
10. la résolution des équations aux dérivées partielles en utilisant la méthode des Éléments finis sur des partitionnements en triangles ou en tétraèdres ayant des bonnes" propriétés [22],
11. la morphologie mathématique [32],
12. la physique et la chimie [7], [29], [1].

Cette liste est bien entendue non exhaustive, et nous ne doutons pas un instant que d'autres applications existantes ont échappé à notre attention.

4.5.4. Les Applications :

Les diagrammes de Voronoi et les triangulations de Delaunay ont de nombreuses interprétations et applications.

On présente ici les exemples du problème du bureau de poste, des problèmes de croissance et des arbres couvrants minimaux.

On renvoie á [8] pour d'autres applications.

Problème du bureau de poste. Soit S un ensemble de points dans le plan représentant les localisations des bureaux de poste dans une ville. Les habitants de cette ville ont pour habitude de se rendre au bureau de poste le plus proche de chez eux, la notion de proximité d'un bureau de poste étant mesurée par la distance euclidienne. Il est clair que la répartition des utilisateurs des bureaux de poste suit le diagramme de Voronoi de S. La connaissance des diagrammes de Voronoi permet alors de répondre á deux problèmes :

– étant donné un point, quel est le bureau de poste le plus proche. La réponse naïve (consistant á tester á chaque fois tous les points de S) requiert un temps linéaire pour chaque demande, tandis que si le calcul du diagramme de Voronoi est effectué comme prétraitement, on peut répondre en temps O (ln n).

– quelle va être la répercussion de l'installation d'un nouveau bureau de poste sur la répartition des utilisateurs.

Problème de croissance. Lorsqu'on dispose des échantillons de bactéries sur une planche nutritive, on observe une croissance centrifuge qui s'arrête lorsque deux échantillons se rejoignent. Si toutes les bactéries se développent à la même vitesse, on obtient ainsi le diagramme de Voronoi des points correspondant á l'emplacement initial des échantillons. On observe le même phénomène sur la carapace des tortues ou dans le cou des girafes réticulés.

Planification de trajectoire

Supposons que l'on dispose d'un robot devant évoluer dans un environnement jalonné d'obstacles ponctuels. Alors, afin de toujours être situé le plus loin possible d'un obstacle, le robot doit se déplacer sur les arêtes du diagramme de Voronoi des obstacles ((a)). Il est à noter que des généralisations du diagramme de Voronoi existent, qui permettent de traiter le cas d'obstacles non ponctuels (par exemples polygonaux, ou circulaires).

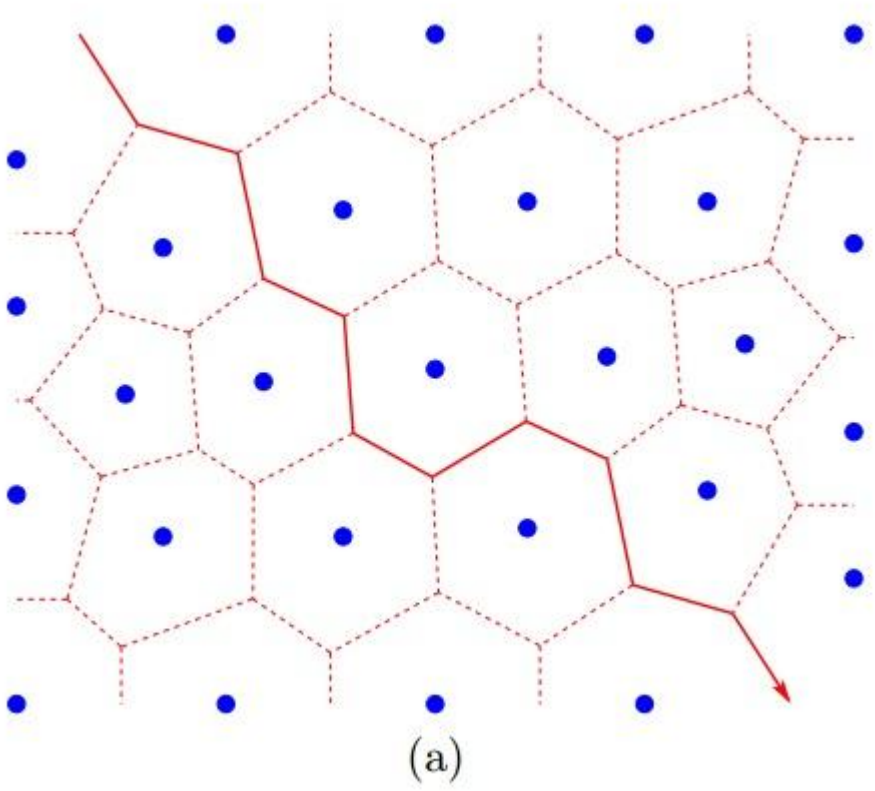


Figure 12.Planification de trajectoire

4.5.5. Les Algorithmes Existants

Nous présentons dans ce chapitre quelques algorithmes de calcul d'un diagramme de Voronoi, dans le cas du plan. L'objectif est bien sûr d'obtenir l'algorithme le plus rapide possible ; cependant l'algorithme doit pouvoir être implémentable !

De plus le calcul des diagrammes de Voronoi et de Delaunay permet de résoudre en temps optimal des problèmes de géométrie algorithmique comme, par exemple : la recherche de tous les plus proches voisins, le calcul de l'enveloppe convexe, le graphe de Gabriel.

Les calculs de la complexité et de la construction des diagrammes de Voronoi et de Delaunay donnent lieu à de nombreuses recherches, notamment en analyse [11, 12, 13], ou en moyenne [14]. Les techniques de calcul font appel à la théorie des probabilités et sont assez élégants.

Notons également qu'une fois des deux structures a été calculée, l'autre peut en être
déduite automatiquement, par dualité.

De nombreuses recherches portent sur la construction des diagrammes de Voronoi. Les deux classes d'algorithme les plus utilisées sont : La classe \split and merge, La classe \incrémentale.

a. Intersection de demi-plans

L'idée la plus naturelle pour construire un diagramme de Voronoi est de construire chaque cellule, Ci de manière indépendante, comme intersection des n-1 demi-plans définis par les médiatrices des segments $Pi\,Pj, \forall j \neq i$. Cette construction est duale de la construction de l'enveloppe convexe de n - 1 points du plan (en effet, le dual d'un point (x, y) est la droite d'équation $b = xa - y$ et le dual d'une droite $y = ax + b$ est le point (a, b), cette enveloppe convexe peut être calculée en temps $O(n \log n)$. La construction du diagramme de Voronoi se fait donc en temps $O(n2 \log n)$.[12]

b. L'algorithme Incrémentale

Une autre idée afin de construire un diagramme de Voronoi consiste á procéder de manière incrémentale : supposons que le diagramme de Voronoi d'ensemble $Pn - 1 =$

$\{p1, \ldots, pn-1\}$ de n - 1 points du plan a été construit ; que faut-il modifier ce diagramme pour construire le diagramme d'un ensemble $P = Pn-1 \cup \{pn\}$? Intuitivement, on comprend aisément que l'insertion d'un point pn ne va modifier le diagramme de Voronoi que localement. Quels sont les sommets de Voronoi qui vont disparaître lors de cette insertion ? Ce sont ceux dont le cercle associé (circonscrit aux germes voisins) contient pn, car on sait que l'intérieur des cercles centrés aux sommets de Voronoi et circonscrits aux germes voisins doit toujours être vide. Ces sommets sont dans une zone limitée du diagramme, ce qui conduit á un algorithme rapide (en O(n)) pour insérer pn et mettre á jour le diagramme de Voronoi. Cet algorithme, proposé par Green et Sibson en 1978, fonctionne de la manière suivante :

1. Trouver le germe pi tel que $pn \in Ci$;
2. Tracer la médiatrice du segment pi pn et calculer ses intersections x1 et x2 avec la frontière de Ci (il n'y en a que deux car Ci est convexe) ;
3. X1x2 est, par construction, une arête de Voronoi du diagramme de P ; c'est même l'arête séparant Ci de Cn. x2 est lui sur une arête de Voronoi du diagramme de Pn-1, séparant Ci d'une autre cellule Cj ;
4. Recommencer le processus en remplaçant pi par pj ;
5. Itérer jusqu'á retombé sur x1 ;
6. On a ainsi construit la frontière de Cn ; mettre á jour les arêtes de Voronoi qu'elle intersecte, en supprimant les morceaux á l'intérieur de Cn.

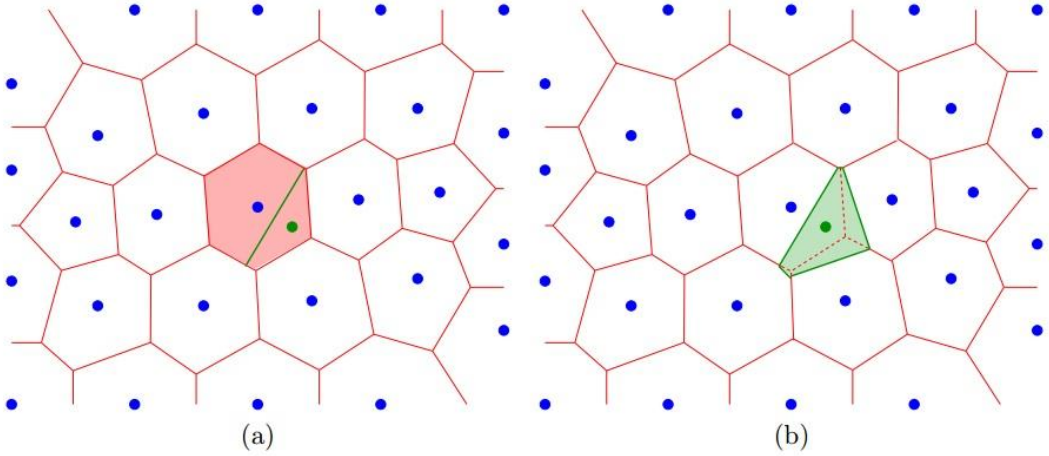


Figure 13 Constriction incrémentale de diagramme voronoi

La cellule Cn peut avoir au pire n - 1 arêtes ; le temps de calcul de cette cellule est donc en O(n). Puisqu'il y a n diagrammes de Voronoi successifs á calculé, la complexité totale de cet algorithme est en O (n2). [12]

c. L'algorithme Diviser pour régner

La complexité en O (n2) n'est pas optimale : un algorithme en O (n log n) a été proposé par Shamos et Hoey dès 1975 [25], avant celui de Green et Sibson. Cependant, cet algorithme n'est jamais utilisé en pratique car trop compliqué á implémenter. Il repose sur la stratégie bien connue "diviser pour régner".

L'ensemble P est divisé en deux sous-ensembles P1 et P2 de tailles identiques, regroupant les points respectivement les plus "á gauche" et les plus "á droite" du plan. Les diagrammes de Voronoi de P1 et de P2 sont calculés récursivement, et toute la difficulté réside ensuite en la fusion de ces deux diagrammes pour obtenir celui de P. Ceci peut être fait en temps linéaire,mais nécessite quelques précautions.[12]

d. L’algorithme par Balayage

L’algorithme le plus utilisé actuellement pour calculer un diagramme de Voronoi est de Steven Fortune. En effet, il est plus rapide que l’algorithme de Green et Sibson car il calcule un diagramme de Voronoi de n points en temps O (n log n) 1, plus simple et facilement implémentable.

Cet algorithme est dit par balayage car il “balaie” progressivement le plan par une ligne et selon une certaine direction, de telle manière qu’à tout moment, dans la zone déjá balayée, le diagramme de Voronoi est construit de manière définitive. Cela peut sembler impossible, car l’apparition d’un nouveau point de P lors du balayage va modifier certaines cellules de Voronoi avant ce point.

L’idée géniale de Fortune consiste á utiliser une troisième dimension, afin d’“anticiper” les modifications du diagramme (en quelque sorte, de “voir l’avenir”).

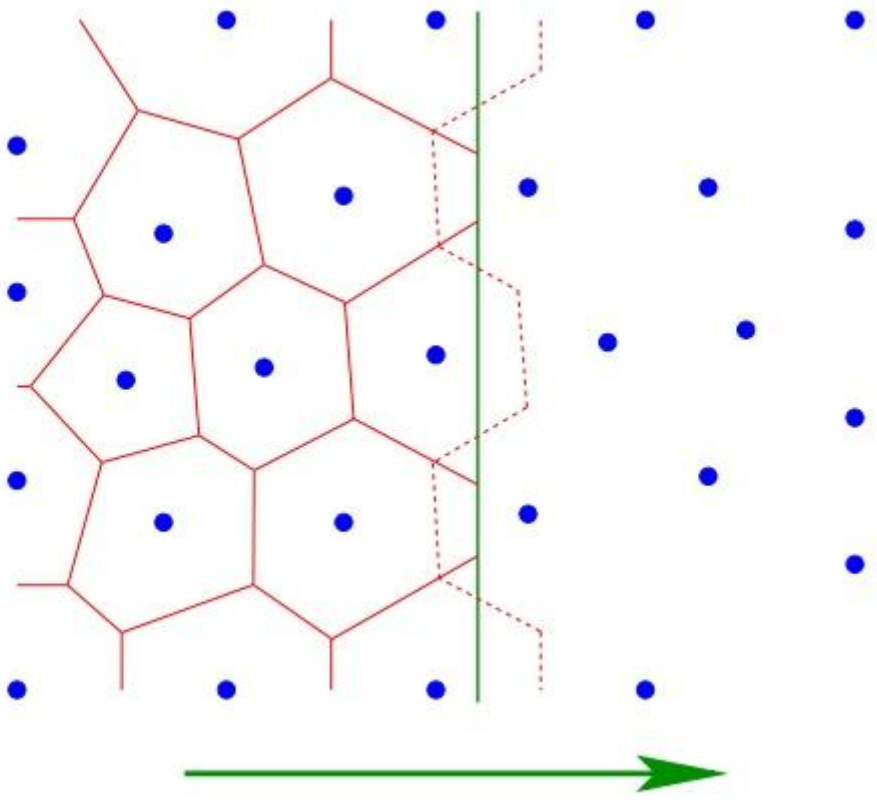


Figure 14.L’algorithme par balayage [12]

Notons $\underset{x}{\rightarrow}$ et $\underset{y}{\rightarrow}$les directions du plan, avec -⟶x la direction de balayage. A chaque germe pi, on associe le cône Ci de sommet pi, d'axe de troisième direction $\underset{z}{\rightarrow} = \underset{x}{\rightarrow} \Delta \underset{y}{\rightarrow}$, et d'angle $\frac{\pi}{4}$ (le choix de cet angle est très important !). Intuitivement, si on interprète la direction -⟶z comme étant le temps, un cône Ci représente un cercle centré sur pi et grossissant á vitesse constante : á z = t0, si on suppose que l'équation du plan est z = 0, son rayon est de t0 (car l'angle est de $\frac{\pi}{4}$ . . .). L'intersection de deux cônes voisins Ci et Cj sera une branche d'hyperbole, et comme les deux cônes ont une même direction et un même angle, celle-ci sera incluse dans le plan perpendiculaire au plan z = 0 et portant la médiatrice du segment pipj (voir figure 13 (a)). Ainsi, les arêtes de Voronoi correspondent aux intersections des cônes, projetées sur le plan z = 0. L'algorithme proposé par Fortune balaie les cônes de x = −∞ á x = +∞ avec un plan Π incliné de π4 par rapport au plan x = cst (voir figure 13 (b)). Le fait que Π ait la même inclinaison que les cônes est très important : Π rencontre un cône Ci exactement quand la droite de balayage

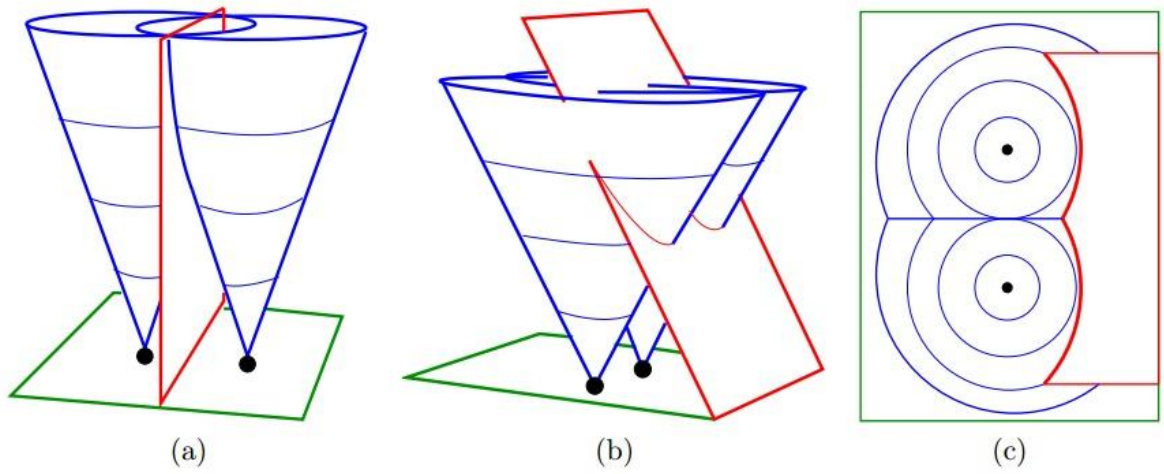


Figure 15 – Algorithme de Fortune. (a) L'intersection de deux cônes se projette sur la médiatrice [12]

(C'est-`a-dire l'intersection de Π et du plan z = 0) rencontre le germe pi. Ainsi une arête de Voronoi, qui correspond á l'intersection de deux cônes voisins Ci etCj, ne sera créée qu'á partir du moment où la droite de balayage a rencontré les deux germes correspondant pi et pj.

L'intersection d'un plan avec un cône étant une parabole, la projection orthogonale sur le plan z = 0 de l'intersection de Π avec l'ensemble des cônes donne ce qu'on appelle un front parabolique (voir figure 13(c)). Le diagramme de Voronoi n'est en fait pas construit á tout instant dans toute la zone déjà balayée, mais seulement "á gauche" du front parabolique. La mise á jour du diagramme ne concerne que ce front, qui est de taille O(n), et peut se faire en temps O (n log n), en représentant le front par un arbre binaire de recherche équilibré (les ajouts et suppressions se font alors en temps O (log n)). On a donc bien un temps de calcul en pire cas en O (n log n).[12]

5. Conclusion de chapitre :

Comme dans d'autres domaines de l'informatiques il existe quelques problèmes fondamentaux, suggérés souvent par les applications : souvent leur solution a motivé un certain nombre de structures de données et algorithmes de la géométrie algorithmique, dont ce chapitre on a concentré au problème de diagramme voronoi

Chapitre 3 Implémentation et analyse de l'algorithme de Fortune

1. Introduction

Dans ce chapitre, nous allons faire une étude pratique de l'algorithme de Fortune pour le calcul du diagramme de Voronoi. Nous verrons le cas des sites ponctuels. Ensuite, nous allons faire une implémentation de l'algorithme avec PyQGIS sous l'environnement QGIS. Enfin, nous aurons une comparaison avec les autres algorithmes de Voronoi.

2. Algorithme de fortune

C'est un algorithme basé sur le "balayage" du plan par une ligne L. Cette ligne se déplace de haut en bas ou de gauche à droite, c'est au choix. Nous choisissons le sens de haut en bas. Lorsque la ligne se trouve à un moment donné, tous ce qui pointe au-dessus de L appartiennent déjà du diagramme de Voronoi.

Si le point x est à la même distance de a et b (les 3 se situent au-dessus de la ligne de balayage), ce point x ne doit pas obligatoirement figurer dans le diagramme, car il peut y avoir le site suivant juste en bas de la ligne de balayage, qui est plus proche. Nous pouvons choisir un diagramme de Voronoï uniquement pour les points qui sont plus proches d'un site sous la ligne que de la ligne L. Ces points équidistants d'un site et de la

ligne de balayage se trouvent sur une parabole. Le site est son point focal et la ligne de balayage est sa directrice, en raison de la définition d'une parabole.

Un peu d'arrière-plan mathématique

J'ai mentionné ci-dessus que l'algorithme représente les cellules en croissance sous forme de paraboles. À l'école, on enseigne à la plupart des gens qu'une parabole est définie par l'équation ($y = ax^2 + bx + c$). Cependant, il existe une autre définition : prenons un point (que nous appellerons le focus) et une ligne droite (que nous appellerons la directrice). Maintenant, si nous devions marquer tous les points du plan pour lesquels la distance au foyer est la même que la distance au point le plus proche de la directrice, l'ensemble de points résultant ferait une parabole.

Pour nos besoins, la direction sera toujours une ligne horizontale et sera toujours en dessous du foyer.

Un problème avec cette définition d'une parabole est qu'il n'est pas nécessairement évident de savoir comment calculer la coordonnée y de la courbe pour une valeur de valeur x. Avec quelques manipulations (illustrées ci-dessous) de la définition standard de la parabole et de notre définition du focus et de la directrice, nous pouvons montrer que pour une directrice ($y = y_d$) et un focus ((x_f, y_f)), on peut obtenir la formule $y = \frac{1}{2(yf-yd)} + (x - x_f)^2 + \frac{yf+yd}{2}$.

La partie utile de cette définition est qu'elle nous donne une idée des distances entre les choses. Considérons pour un moment deux courbes, définies par deux points différents (f_1) *et* (f_2)mais la même directrice, qui se coupent en un point (p). Puisque les courbes partagent (p) et la directrice, leur distance entre (p) et la directrice est

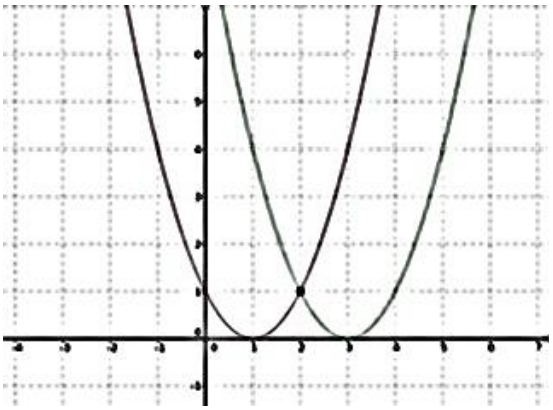


Figure 16. Deux paraboles

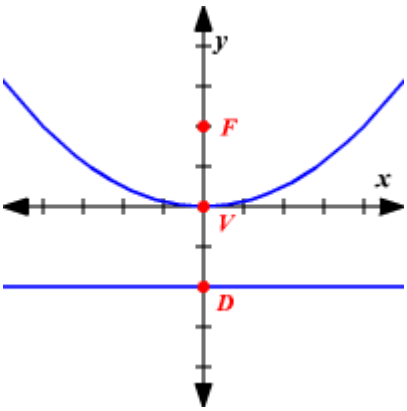


Figure 17. Relation entre le foyer, le sommet et la directrice

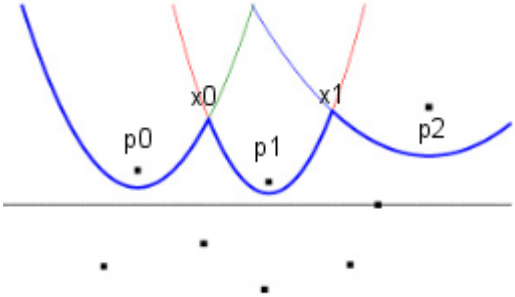
évidemment la même. Cependant, par définition, cela signifie que la distance de (p) à (f_1) est la même que la distance de (p) à (f_2). Maintenant, si (f_1) *et* (f_2) sont deux sites dans un diagramme de Voronoi, tous les points d'un côté de (p) sont plus proches de (f_1) qu'ils ne le sont à (f_2) (et devrait donc être dans la cellule autour de (f_1) et tous ceux de l'autre côté sont plus proches de (f_2)(et devraient être dans la cellule autour de (f_2).

Comme cela est vrai pour tous les points (p) au point d'intersection des deux courbes, nous pouvons trouver la limite entre les deux cellules en déplaçant la directrice et en traçant une ligne constituée des points d'intersection. C'est le raisonnement mathématique derrière l'algorithme de Fortune.

Une chose que je devrais mentionner cependant est que je n'ai pas encore montré pourquoi cette intersection sera toujours une ligne droite.

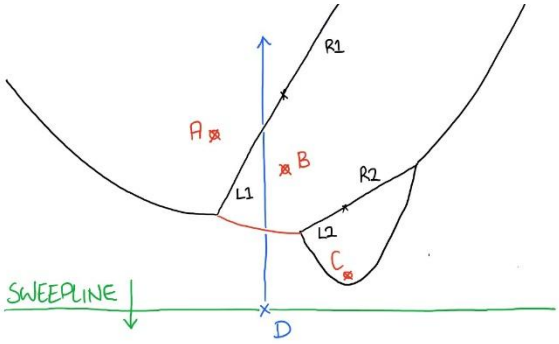
La bordure, qui dit, pour laquelle nous pouvons tracer un diagramme, est constituée d'arcs de paraboles. Nous appelons cela une plage ou une limite. Les intersections des arcs de cette parabole ont la même distance par rapport à quelque 2 sites (2 foyers de paraboles) et à la ligne de balayage. Ainsi, lorsque vous déplacez un axe haut vers le supérieur, ces intersections dessinent le diagramme.

La ligne de plage est une séquence d'arcs pet intersections x, que nous pouvons appeler "bords", car ils dessinent le bord du graphe de Voronoï.Beachline p_0, x_0,p_1, x_1,…, x_{n-1}, p_n. La ligne de plage change tandis que le sweepline monte. La

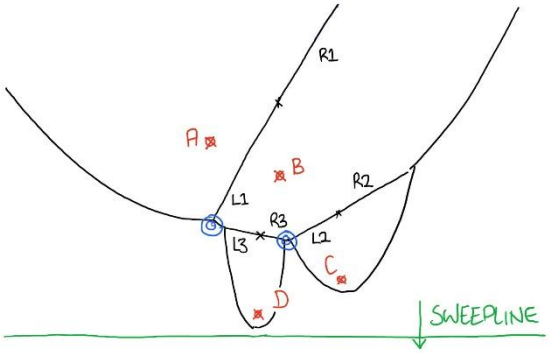


Beachline ne peut être changé qu'à deux reprises :

Événement sur site (Site évent) - la ligne de passage passe sur un nouveau site, ce qui ajoute une nouvelle parabole à la plage. La plage est divisée en 2 arcs et 2 bords (2 parties du bord final) commencent à venir dans des directions opposées sous ce nouveau site. Ainsi, la parabole pi est remplacé par une séquence pi, xi, pi + 1, xi + 1, pi + 2, où pi a pi + 2 ont le foyer d'une parabole précédente et pi + 1 se concentre sur un site nouvellement ajouté.



Événement de cercle - un arc disparaît, deux arcs voisins le "pressent". Un nouveau bord entre voisins est démarré. Ainsi, une suite xi-1, pi, xi est remplacée par un nouveau bord xi.



Un endroit, où l'arc disparaît et le nouveau bord commence, se trouve à la même distance des 3 foyers (son foyer et les foyers des voisins). Il s'agit donc du cercle circonscrit du cercle circonscrit, défini par ces 3 sites.

Nous allons stocker ces événements dans une file d'attente ou un tas triés, où ils sont triés par les coordonnées "y". Nous voyons que la plage ne change que quelques instants (événements), nous pouvons donc la déplacer discrètement d'un événement à l'autre.

3. Code de l'algorithme

L'entrée est un ensemble de points (sites). La sortie est un ensemble d'arêtes ou de polygones.

Distance (point1, point2) //calcule la distance entre deux points

d= $\sqrt{(\text{point1.x} - \text{point2.x})^2 + (\text{point1.y} - \text{point2.y})^2}$

return d

Shift (q File) // La méthode shift () Extrait le maximal évènement d'un la file d'attente.

```
Event_min=q[0]
    Si Event_min.y != None :
        Pour x dans q :
            If x.event=="circle" and Event_min.y>x.y :
                min=x
        Retirer Event_min de la file q
        Return min
    Sinon :
    Retirer Event_min de la file q
    Return min
```

Centre_Point(point1, point2,point3): // méthode qui calcul centre point entre 3 points

A=point2.x-point1.x

B=point2.y-point1.y

C=point3.x-point1.x

D=point3.y-point1.y

E = A*(point1.x+point2.x) + B*(point1.y+point2.y)

F = C*(point1.x+point3.x) + D*(point1.y+point3.y)

G = 2*(A*(pont3.y-point2.y) - B*(point3.x-point2.x))

Si G==0 : return //Les trois points sont debout

x = (D*E-B*F)/G ;

y = (A*F-C*E)/G

Return x,y

Point_Parbol(p, x): // méthode qui calcule y dans x dans un parabole

Intersection (p1, p2) :// méthode qui donnée les points d'intersection des deux parabole (p1,p2)

Supprimer _ Cercle_Event(p) ://méthode qui supprimer cercle évènement d'un point (p) dans la file

Ajouter_Parabole (p) :

p_au_dessus = au_dessus_du(beach,p)

y = point_parbol(p_au_dessus, p)

supprimer _ cercle_évent (p_au_dessus)

a,b,c = new point()

b = p

a = c = p_au_dessus

xl=[p.x,y,a,b] // un bord À partir du point d'intersection de p,au-dessus et appartenir á point a,b

xr=[p.x,y,b,c] // un bord À partir du point d'intersection de p,au-dessus et appartenir á point b,c

remplacer la p_ au_dessus avec la séquence a,xl,b,xr,c

CheckCircleEvent(a)

CheckCircleEvent(c)

Supprimer Parabole (p):

l = point vers la gauche de p

r = point vers la droit de p

xl = bord vers la gauche de p

xr = bord vers la droit de p

supprimer _ cercle_évent (l)

supprimer _ cercle_évent (r)

s = centre_point(l,p,r)

x = [s.x,s.y,l,r] // un bord À partir du point s et appartenir á point l,r

// Replacer a sequence xl, p, xr par le nouveau bord x

CheckCircleEvent(l);

CheckCircleEvent(r);

CheckCircleEvent(p):

xl = bord vers la gauche de p

xr = bord vers la droit de p

si l ==None ou r==None ou l==r : return

s = centre_point(l,p,r)

r = Distance (p, s):

Si s.y + r est toujours au-dessus de la sweepline return

e = crée une nouvelle cercle évènement

e.point = p;

e.y = s.y + r;

ajouter e dans la file d’attente

Pour tous les points (sites) :

Créer un événement E de site S

E. point = point (site), ajouter E dans la file F

E. Event = «événement sur site»

Tant que F pas vide :

E = Shift(F) // La méthode shift () supprime le premier élément d'un file.

Si E. Event == (événement sur site) : Ajouter_Parabole (E. Point)

Si non : supprimer Parabole (E. Parabole) ;

4. Résultats

Nous avons implémenté l'algorithme de Fortune en PyQGIS sous QGIS. L'entrée de l'algorithme est une couche de points. On calcule le rectangle englobant tout d'abord pour délimiter la zone, ensuite on calcule le diagramme de Voronoi. On peut voir ci-dessous les résultats de l'algorithme :

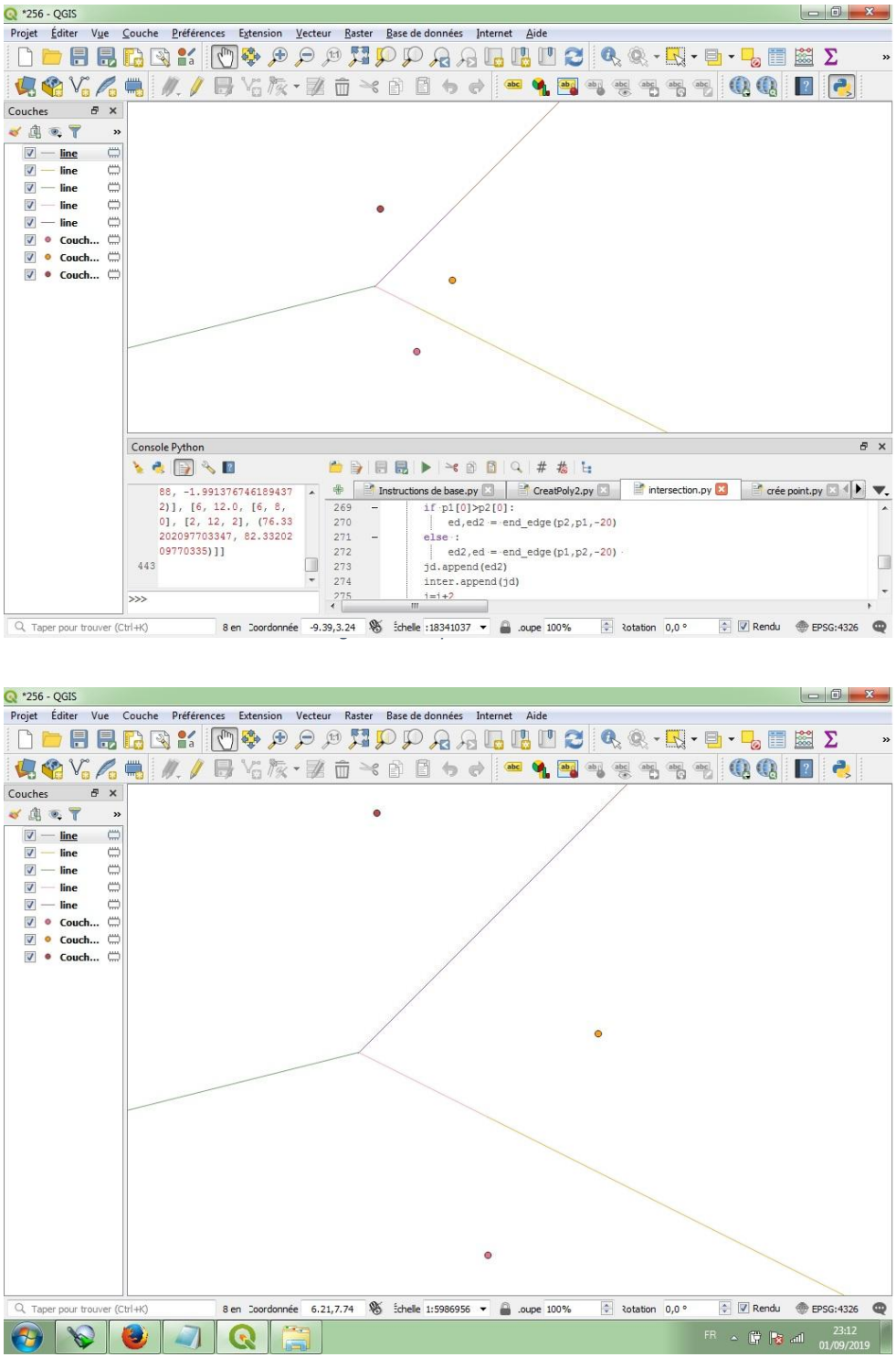


Figure 19. Capture d'écran 2

5. Complexité du diagramme de Voronoi

Théorème : Pour $n \geq 3$ sites, le diagramme de Voronoï contient au plus $2n - 5$ sommets et $3n - 6$ arêtes.

Preuve :

Pour les sites situés dans la ligne, c'est évidemment vrai.

Supposons que les sites ne soient pas dans la file.

Notons ces variables :

V : nombre de sommets dans le diagramme de Voronoï

E : nombre d'arêtes

N : nombre de faces internes = nombre de sites

D'après la formule d'Euler, nous savons que :

$$V - E + N = 2$$

Comme le diagramme de Voronoï contient des arêtes infinies, créons un nouveau sommet "infini" et connectons-lui toutes ces arêtes. Maintenant, c'est un graphe planaire.

$$(V + 1) - E + N = 2$$

Dans la graphie de Voronoï, nous savons que chaque sommet a un degré d'au moins 3 (y compris le sommet "infini"). Un bord si entre deux sommets, donc

$$3\,(V + 1) \leq 2 * E$$

Par une simple modification algébrique, nous obtenons un théorème précédent.

Les opérations primitives, telles que la recherche d'un élément dans l'arborescence ou dans la file d'attente, la suppression de la file d'attente, tout cela peut être fait- en O (log (n)).A chaque événement, nous effectuons un nombre constant de ces opérations primitives.

Le nombre d'événements sur site est N, les événements de cercle sont au plus $2N - 5$. A chacun d'eux, on fait des opérations c primitives. Ainsi, la complexité finale est O ($n * \log(n)$).

6. Les critères de comparaison

Puisqu'il existe plusieurs algorithmes qui résolvent un même problème, lequel préférer ? Répondre à cette question, c'est se demander ce qui fait la qualité d'un algorithme ou d'un programme informatique. Quels sont les critères qui permettent de juger ? C'est un vaste sujet mais nous voudrions aborder les principaux points.

La Complexité

L'objectif premier d'un calcul de complexité algorithmique est de pouvoir comparer l'efficacité d'algorithmes résolvant le même problème. Dans une situation donnée, cela permet donc d'établir lequel des algorithmes disponibles est le plus optimal.

Ce critère est un cas particulier de l'*efficience* qui traite de la gestion économe des ressources.

La Simplicité

La simplicité indique si une personne qui lit l'algorithme (ou le programme) peut facilement percevoir comment il fonctionne. C'est un critère très important qu'il ne faut surtout pas sous-évaluer. On peut en faire d'ailleurs l'amère expérience, si l'on néglige la simplicité d'un algorithme, cet algorithme ne sera jamais utilisé en pratique car trop compliqué á implémenter

La Généralisation

La généralisation des algorithmes signifie que l'algorithme est capable de traiter plusieurs formes de données sans modifier les performances. Et dans le domaine géographique un algorithme se doit de traiter les différents types d'objets pouvant se trouver sur la carte (point, ligne, surface).

7. Comparaison

Le tableau ci-dessous représente la comparaison des algorithmes pour le calcul du diagramme de Voronoi, en utilisant les critères décrits précédemment. Cependant, tous ces critères n'ont pas le même poids.

	Intersection de demi-plans	L'algorithme de Green et Sibson	L'algorithme de Shamos et Hoey	L'algorithme de Steven Fortune
La complicité	O (n^2 log n)	O (n)	O (n log n)	O (n log n)
La simplicité	Simple	Simple	Très compliqué et difficile a implémenter	Simple
La généralisation	Il marche avec tous les Objets géométriques	Il marche seulement avec les points	Il marche avec tous les Objets géométriques	Il marche avec tous les Objets géométriques

Tableau 1. La comparaison des algorithmes

L'algorithme de Steven Fortune est plus rapide que l'algorithme de Green et Sibson car il calcule un diagramme de Voronoi de n points en un temps O(n log n), il est plus simple que l'algorithme de Shamos et Hoey et facilement implémentable, en plus d'être généralisable.

8. Conclusion

Nous avons vu dans ce chapitre l'algorithme de Fortune plus en détails. Nous l'avons implémenté avec PyQGIS et nous l'avons comparé avec les algorithmes existants. Les résultats ont démontré que l'algorithme de Fortune répondait au mieux aux critères que nous avons sélectionné à savoir la complexité, la simplicité et la généralisation.

Conclusion générale

Dans ce mémoire nous nous sommes intéressés au diagramme de Voronoi qui est un des diagrammes les plus utilisés dans le domaine géographique. Il fait partie de la discipline de la géométrie algorithmique. Notre intuition s'est portait sur l'algorithme de Steven Fortune, un algorithme qui permet justement de calculer le diagramme de Voronoi. C'est un algorithme très élégant d'un point de vu mathématique et son idée est assez simple car basée simplement sur la distance.

Nous avons pu implémenter l'algorithme avec le langage PyQGIS qui lie la puissance du langage Python aux bibliothèques spatiales de l'environnement cartographique QGIS.

Nous avons pu confirmer notre intuition sur les qualités de l'algorithme de Fortune en le comparant avec les autres algorithmes pour le calcul du diagramme de Voronoi existants notamment quater algorithmes (Intersection de demi-plans, L'algorithme de Green et Sibson, L'algorithme de Shamos et Hoey et L'algorithme de Steven Fortune), Nous avons basé notre comparaison sur les critères suivants (la complicité, la simplicité, la généralisation).

Il s'est avéré que c'est l'algorithme de Fortune qui répond aux mieux aux critères énumérés car il possède non seulement une complexité de O (n log(n)) mais il permet la généralisation en plus d'être simple à implémenter. Nous pouvons dire alors que l'algorithme de Voronoi répond bien à la problématique des données cartographiques qui se caractérisent par des formes variés en plus d'être volumineuses.

Comme perspectives nous espérons que ce modeste travail pourra être amélioré avec une implémentation de l'algorithme pour le reste des objets cartographiques en plus des dimensions géométriques supérieurs.

Bibliographie :

[1] A.E. Galashev and V.P. Skripov. Stability of Lennars-Jones crystal structures in the molecular dynamics model. Sov. J. Low Temp. Phys., 6, 1985.

[2] Blog.ivank.net/fortunes-algorithm-and-implementation

[3] D.F. Stevens. Patterns in Nature. Little Brown, Boston, 1978.

[4] D.T. Lee. Concrete and abstract Voronoi diagrams. IEEE Trans. PAMI, 4(4):363{369, 1982.

[5] E. Bertin and J.-M. Chassery. 3D generalized Voronoi diagram. In Proc. of The second Congr es Curves and Surfaces, Chamonix-Mont-Blanc, June 1993.

[6] E. Bruzzone, G. Garibotto, and F. Mangili. Three-dimensional surface reconstruction using delaunay triangulation in the image plane. In C. Arcelli, editor,Visual Form, pages 99{108, New York, 1992. Plenum.

[7] E.E. David and C.W. David. Voronoi polyhedra as a tool for studying solvation. J. Chem. Phys., 76(9):4611{4614, 1982.

[8] F. Aurenhammer. Voronoi diagrams - a survey of a fundamental geometric data structure. ACM Computing Surveys, 33(3):345-405, 1991.

[9] F. Benoit, E. Bertin, M. M emier, and J.-M. Chassery. G en eration de courbesde niveau à partir de donn ees contraintes. In Proc. Du 9 eme Congr es RFIA, AFCET-INRIA ed. Paris, page In press, Janvier 1994.

[10] F. Davoine, E. Bertin, and J.-M. Chassery. From rigidity to adaptive tessellations for fractal image compression: comparative studies. In Proc. of the 8 eme Workshop on Image Multidimensional Signal Processing, IEEE Signal Processing Society, Cannes, pages 56{57, September 1993.

[11] F. Davoine, E. Bertin, and J.-M. Chassery. Fractal image coding based on Delaunay tessellation. In Proc. of the 17th OAGM - Meeting, Graz, Austria,June 1993.

[12] Franck Hétroy. Un petit peu de géométrie algorithmique.

[8] G.T. Toussaint. Pattern recognition and geometrical complexity. In Proc. Of the 5th ICPR, IEEE, Miami Beach, pages 1324{1347, 1989.

[13] Guendalina Palmirotta, ***Triangulation de Delaunay*** 2014-2015

[14] H. Honda. Description of cellular patterns by Dirichlet domains: the two dimensionnal case. J. Theor. Biol., 75:523{543, 1978.

[15] I. Bloch-Boulanger, F. Schmitt, and H. Matre. Calcul de l'enveloppe convexe D'un objet tridimensionnel. In Journées de géométrie algorithmique, INRIA

Sophia Antipolis, pages 791{800, Juin 1990.

[16] J.-D. Boissonnat and M. Devillers-Teillaud. On the randomized construction of the Delaunay tree. Technical Report 1140, INRIA, Sophia Antipolis, 1989.

[17] J.-D. Boissonnat. Representation of ob jects by triangulating points in 3D space. In Proc. of the ICPR 82, IEEE, Silver Springs, Munich, pages 830{832, 1982.

[18] J.-D. Boissonnat. Shape reconstruction from planar cross-sections. Technical Report 546, INRIA, Sophia Antipolis, 1986.

[19] J.-D. Boissonnat and B. Geiger. Three-dimensional reconstruction of complex shapes based on the Delaunay triangulation. Technical Report 1697, INRIA, Sophia Antipolis, 1992.

[20] J.-D. Boissonnat, O. D. Faugeras, and E. Le Bras-Mehlman. Representing stereo data with the Delaunay triangulation. Technical Report 788, INRIA,Sophia Antipolis, 1988.

[21] J.-M. Chassery, F. Davoine, and E. Bertin. Compression fractale par partitionnement de Delaunay. In Proc. Of the 14 eme Col loque GRETSI, Juan-les-Pins, volume 2, pages 819{922, September 1993.

[22] J.M. Talon. G en eration et am elioration de mail lage pour les el ements nis en deux et trois dimensions. PhD thesis, Universit e Joseph Fourier, Grenoble, 1989.

[23] L. Vincent. Mathematical morphology on graphs. Signal Processing, 16(4):365{388, 1989.

[24] Mathplanet.com/education/geometry/points,-lines,-planes-and-angles/an-introduction-to-geometry

[25] M. Shamos, D. Hoey. Closest-Point Problems. 16th Annual Symposium on Foundations of Computer Science, p. 151-162, 1975.

[26] M. Teillaud. Vers des algorithmes dynamiques randomis es en g eom etrie algorithmique. PhD thesis, Universit e de Paris XI, Orsay, 1992

[27] M.Q. Vahidi-asl. First-passage percolation on the Voronoi tesselation and theDelaunay triangulation. In Poeppl, editor, Mustererkennung 1993, L • ubeck, September 1993. Springer.

[28] N. Ahuja, B. An, and B. Schachter. Image representation using Voronoi tessellation. Computer Vision Graphics and Image processing, 29:286{295, 1985.

[29] N.N. Medvedev and Yu. I. Naberukhin. Delaunay simplexes of a simple liquid

And amorphous substances. Sov. Phys. Dokl., 31(6):465{466, 1985.

[30] O. Devillers. Randomization yields simple O(nlogn) algorithms for diécult W (n) problems. International Journal of Computational Geometry and Applications, 2:97{111, 1992.

[31] Portailsig.org/content/sur-la-creation-des-enveloppes-concaves-concave-hull-et-les-divers-moyens-d-y-parvenir-forme

[32] R.A. Dwyer. Average-case analysis of algorithms for convex hulls and Voronoi diagrams. Technical Report CMU-CS-88-132, Departement of ComputerScience, Carnegie Mellon University, 1988.

[33] R. Marcelpoil and Y. Usson. Methods for the study of cellular sociology: Voronoi diagrams and parametrization of the spatial relationships. J. Theor.Biol., 154:359{369, 1992.

[34] R. Marcelpoil and E. Bertin. Cellules en soci et e. Science et Vie, 184:68{74, 1993.

[35] S. Rippa. Minimal roughness property of the Delaunay triangulation. Computer Geometric Design, 7:489{497, 1990.

[36] T. Kohonen. The self-organizing map. Proc. of the IEEE, 78(9):1464{1480, 1990.

[37] Wikipedia.org/wiki/Coordonnographiques

[38] Wikipedia, 3 septembre 2019, [consulté le 7 septembre 2019], Disponible sur Wikipedia.org/wiki/QGIS

[39] Wikipedia, 15 juillet 2019, [consulté le 7 septembre 2019], Disponible sur Wikipedia.org/wiki/Espace_(notion)

[40] Wikipedia, 1 aout 2019, [consulté le 7 septembre 2019], Disponible sur Wikipedia.org/wiki/Minimum_bounding_box

[41] Wikipedia 15 février 2019, [consulté le 7 septembre 2019], Disponible sur Wikipedia.org/wiki/Triangulation_de_Delaunay