Comment dénombrer le nombre de faces, arêtes, sommets pour un objet 3D discretisé de manière optimale ?

I. Blasquez et J.-F. Poiraudeau

LICN – IUT du Limousin – Allée André Maurois – 87065 LIMOGES Cedex
isabelle.blasquez@unilim.fr ; poiraudeau@unilim.fr

Résumé : *L'analyse morphologique d'image permet de caractériser la forme (géométrie) et la connectivité (topologie) des images binaires 3D en utilisant les fonctionnelles de Minkowski. En 3D, ces fonctionnelles correspondent au volume, à la surface, à la largeur moyenne et à la caractéristique d'Euler. Pour les calculer, il suffit de compter le nombre de cubes ouverts, faces ouvertes, arêtes ouvertes et sommets ouverts de l'objet. Pour dénombrer rapidement ces éléments géométriques, nous proposons dans cet article un algorithme qui s'appuie sur les diagrammes de décision binaires et sur des notions de topologie en introduisant la structure du triple-ADD réduit et ordonné. Nous montrons que cet algorithme est 17 fois plus rapide que l'algorithme proposé récemment dans la littérature par Michielsen.*

Mots-clés : *Diagrammes de Décision Binaires, topologie, fonctionnelles de Minkowski, analyse d'image*

1 Introduction

De très importants volumes de données, souvent obtenues à partir d'images, sont utilisés dans de nombreux domaines scientifiques. Un des buts principaux de l'analyse d'image est de fournir une caractérisation quantitative de la forme et de la connectivité des constituants. La mise en place d'un algorithme rapide est donc importante pour les applications [1]. Dans cet article, nous proposons une méthode efficace pour calculer les propriétés morphologiques des images.
La caractérisation des formes 3D à l'aide de quatre mesures [2] est un des domaines de la morphologie mathématique. Ces mesures, appelées aussi fonctionnelles de Minkowski (ou volumes intrinsèques) sont respectivement, le volume (V), la surface (S), la largeur moyenne (B) et la caractéristique d'Euler-Poincaré (χ)[1]. A l'origine, ces quatre fonctionnelles ont été définies pour des objets convexes dans le cadre de la géométrie intégrale. Dans les années cinquante, Hadwiger a montré que chaque mesure sur un ensemble fini de

[1]Pour un parallélépipède rectangle plein de longueur L, de largeur l et de hauteur h, les fonctionnelles de Minkoswki sont : $V = Llh$, $S = 2(Ll + Lh + lh)$, $B = (L + l + h)/2$ et $\chi = 1$. La notion de largeur moyenne est utilisée depuis longtemps par la poste pour dimensionner les colis : «les dimensions maximales sont le total des trois dimensions inférieur ou égal à 100 centimètres». Un colis est dit standard si $L + l + h < 100$ cm, de 100 cm à 150 cm il est qualifié de volumineux. χ est un invariant topologique qui vise à décrire la morphologie des ensembles en fonction de leur connexité et de l'existence de cavité ou de tunnels. Une cavité incrémenterait χ de 1, un tunnel décrémenterait χ de 1.

convexes compacts peut être écrit comme une combinaison linéaire des quatre fonctionnelles de Minkowski. Ces fonctionnelles ont une propriété commune : elles sont additives. La fonctionnelle $\mathcal{M}_\nu$ de l'union $A \cup B$ de deux ensembles convexes A et B est la somme des fonctionnelles des ensembles convexes moins la fonctionnelle de leur intersection : $\mathcal{M}_\nu(A \cup B) = \mathcal{M}_\nu(A) + \mathcal{M}_\nu(B) - \mathcal{M}_\nu(A \cap B)$. Cette relation généralise la règle pour l'addition du volume de deux ensembles convexes au cas d'une mesure morphologique générale. Dans le cas des images discrètes, la propriété d'additivité simplifie le traitement des fonctionnelles. Au cours de ces dernières années, ces fonctionnelles ont été largement utilisées dans de nombreux domaines qui vont de la détermination des très grandes structures de l'univers [3] à la modélisation des milieux poreux [4]. Cependant, pour l'instant, elles restent relativement peu employées par les «traiteurs d'image».

2 Position du problème

Une image binaire 3D est une grille de $\mathbb{Z}^3$ représentée par une matrice de voxels, où chaque voxel est affectée d'une valeur binaire. L'objet est l'ensemble des voxels qui ont pour valeur 1 (appelés aussi voxels noirs, pleins ou actifs). Le fond est l'ensemble des voxels qui ont pour valeur 0 (appelés aussi voxels blancs, vides ou inactifs).
En considérant chaque voxel comme une union de son intérieur, de ses 6 faces ouvertes, de ses 12 arêtes ouvertes et de ses 8 sommets ouverts, il est possible de déterminer les quatre fonctionnelles de Minkowski pour un objet 3D discrétisé : le volume intérieur ($V = n_3$), la surface ($S = -6n_3 + 2n_2$), la largeur moyenne ($2B = 3n_3 - 2n_2 + n_1$) et la caractéristique d'Euler ($\chi = -n_3 + n_2 - n_1 + n_0$) où n_3 est le nombre de cubes ouverts (ou voxels ouverts), n_2 le nombre de faces ouvertes, n_1 le nombre d'arêtes ouvertes et n_0 le nombre de sommets ouverts [4]. La caractérisation morphologique d'une forme 3D se réduit donc au dénombrement des objets géométriques élémentaires (cubes ouverts, faces ouvertes, arêtes ouvertes et sommets ouverts) qui constituent cette forme. Leur dénombrement requiert d'examiner tout d'abord les 26 voisins de chaque voxel pour calculer ensuite une fonction discrète à variables booléennes. Dans la littérature, il n'existe à notre connaissance qu'une seule implémentation proposée par Michielsen et Raedt [4]. Dans cet article, ils décrivent des équations qui montrent comment n_2, n_1, n_0 évoluent lorsqu'un voxel noir $V(i, j, k)$ est ajouté à une forme 3D donnée. En utilisant ces équations, il calculent les fonctionnelles de Minkowski pour une forme donnée, simplement en ajoutant les voxels noirs un à un à un fond initialement complètement blanc. Cependant, nous estimons que la solution proposée dans [4] a deux inconvénients. Le premier point concerne l'algorithme lui-même et plus particulièrement le nombre de voisins à explorer : tous les 26 voisins du voxel courant doivent être examinés pour pouvoir résoudre les équations. Le second point concerne l'implémentation de l'algorithme. Deux tableaux sont utilisés : le premier pour stocker toute l'image 3D à analyser, l'autre pour traiter l'image 3D en cours d'analyse (il est rempli voxel par voxel), ce qui est coûteux en mémoire et en temps d'accès. A notre connaissance, cet algorithme a été implémenté en Fortran 90 sur un monoprocesseur et aucune indication de temps de calcul n'est indiquée. Pour limiter le nombre d'accès aux voisins d'un voxel courant et supprimer l'utilisation de deux tableaux, nous avons décidé d'envisager le problème différemment.
Tout d'abord, nous nous sommes intéressés à des considérations algorithmiques et nous avons pris en compte les symétries du problèmes pour réduire notre problème en sous-

problèmes. De tels sous-problèmes sont généralement résolus en utilisant des tables de références (*lookup−table*) ou des *quadtrees* [5] que nous n'avons pas retenu à cause du coût mémoire. Un compromis doit être envisagé entre l'espace mémoire et la complexité algorithmique. Dans [6], une approche différente à base de Diagrammes de Decision Binaires (DDB) (*Binary Decision Diagrams*) est proposée pour générer automatiquement un code efficace pour des algorithmes de traitement d'image. Nous avons également utilisé les DDBs pour calculer les fonctions discrètes à variables booléennes de notre dénombrement de faces ouvertes, arêtes ouvertes et sommets ouverts.

3 Vers un dénombrement rapide des caractéristiques morphologiques d'une image 3D discrète

Dans notre algorithme, nous décomposons la matrice de voxels en plans et la parcourons en colonne de gauche à droite, en ligne de bas en haut, et d'un plan en avant de l'image vers un plan en arrière. Sur la figure 1.a, nous avons représenté une sous-matrice autour d'un voxel courant noté V. Les 26 voxels voisins sont notés N_{ip} où i indique la position du voxel dans le plan ($0 \leq i \leq 8$) et p indique le plan dans lequel se situe le voxel voisin considéré (1 : plan avant, 2 : plan courant, 3 : plan arrière). Pour chaque nouveau voxel courant plein, nous cherchons à déterminer le nombre de faces ouvertes (Δn_2), d'arêtes ouvertes (Δn_1), et de sommets ouverts (Δn_0) introduit par l'ajout de ce voxel dans la matrice. Dans un premier temps, nous réduisons le problème en n'examinant que les 13 voxels voisins précédant le voxel courant plein ($Ni1$ pour i=0..8, $Nj2$ pour j=1,2,3,8). L'existence d'un voxel courant plein amènera au moins au dénombrement suivant : 3 faces ouvertes, 3 arêtes ouvertes et 1 sommet ouvert ; ceci quels que soient les 13 autres voxels voisins, examinés à leur tour dans la suite du parcours de la matrice.
Pour chaque élément géométrique restant du voxel courant, il faut pouvoir déterminer si cet élément a déjà été compté. Pour cela, tous les voxels précédents susceptibles de partager cet élément doivent être examinés : si un seul voxel est plein, l'élément ne doit pas être pris en compte, puisqu'il fait déjà partie de l'énumération existante. Le nombre maximal de faces ouvertes susceptibles d'être rajoutées est 3, celui d'arêtes ouvertes est 9, et celui de sommets ouverts est 7. On arrive donc à 3 équations discrètes qui dépendent des 13 voisins précédents. Nous ne donnerons que celle relative aux sommets ouverts :

$$\Delta n_0 = 1+Q82+Q22.Q32+Q12.Q22.Q82+Q01.Q41.Q51.Q61+Q81.Q01.Q61.Q71.Q82+ Q21.Q31.Q41.Q01.Q22.Q32 + Q11.Q21.Q01.Q81.Q12.Q22.Q82 \quad (1)$$

avec $Qij = 1 - Nij$, où $Nij = 1$ pour un voxel noir et $Nij = 0$ pour un voxel blanc.

L'évaluation expérimentale montrera que le gain sur la durée d'exécution par rapport à l'algorithme de Michielsen est déjà important en utilisant uniquement ces équations. Dans un second temps, nous souhaitons optimiser la solution précédente. Nous proposons un algorithme qui permet d'examiner seulement les voxels dont les valeurs affectent le résultat en utilisant des diagrammes de decision binaires. Cet algorithme revient à utiliser des branchement conditionnels judicieusement choisis afin d'examiner le plus faible nombre de voxels voisins pour chaque voxel courant et d'arrêter les tests dès qu'un élément géométrique restant a déjà été partagé par un voxel voisin.

4 Les diagrammes de décision binaires (DDBs)

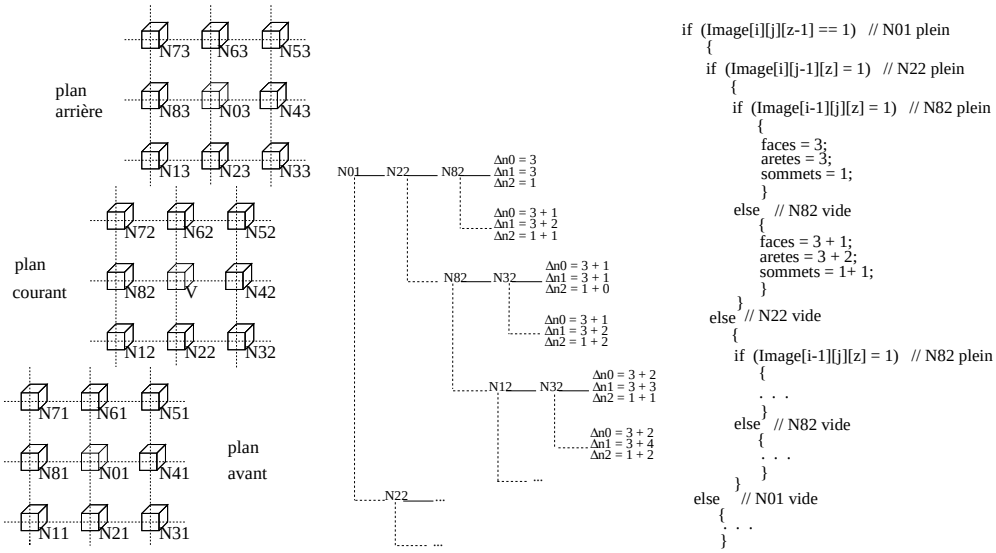
Les DDBs sont une représentation compacte et efficace pour la manipulation symbolique des fonctions booléennes. Leur concept a été introduit par Lee et Akers [7]. Lors de ces dernières années, les DDBs ont montré leur efficacité dans différents domaines et dans de nombreuses applications aussi bien en conception de systèmes numériques [8], qu'en traitement d'image [9].

4.1 Fonctions booléennes à variables booléennes

Les variables booléennes peuvent prendre les valeurs dans $\mathbb{B} = \{0,1\}$. Un diagramme de decision binaire (DDB) représente une fonction booléenne $f(x_1, x_2, ..., x_n) : \mathbb{B}^n \rightarrow \mathbb{B}$ comme un graphe acyclique direct dans lequel chaque noeud est soumis à un test d'une variable booléenne x_i (représentation de Shannon) : $f = \bar{x}_i f_{x_i=0} + x_i f_{x_i=1}$ $(1 \leq i \leq n)$
Chaque noeud a deux fils qui sont eux aussi des DDBs. A chaque noeud, un des fils est choisi en fonction de la valeur de la variable représentée associée à ce noeud. La valeur de la fonction est déterminée en traversant le graphe à partir de la racine jusqu'à un noeud terminal. Les noeuds terminaux du graphe sont uniquement des valeurs de $\mathbb{B}$.
Un DDB est dit *ordonné* si chaque variable n'est comptée au plus qu'une fois sur chaque chemin allant de la racine au noeud terminal et si on rencontre les variables toujours dans le même ordre sur chaque chemin $(x_1 < x_2 < ... < x_n)$.
Un DDB est dit *réduit* s'il ne contient pas de sommets : soit avec un graphe isomorphe, soit avec deux arêtes qui pointent vers le même noeud.
Les DDBs réduits et ordonnés sont uniques. Les DDBs n'ont été réellement développés qu'à la suite de la publication d'un article de Bryant [8] qui définit alors les diagrammes de décision binaires ordonnés (*OBDD : Ordered Binary Decision Diagrams*) comme un sous-ensemble des DDBs. Les OBDDs sont des DDBs dans lesquels les variables sont strictement ordonnées à partir de la racine jusqu'aux terminaux. Un des avantages des *OBDDs* est la canonicité : si une variable ordonnée est fixée et que les règles de réduction sont appliquées, deux fonctions booléennes équivalentes ont le même *OBDD*. Un *OBDD* qui est sous forme canonique est aussi appelé diagramme de décision binaire ordonné réduit (*ROBDD*).
Des algorithmes de manipulation de graphes s'appliquent aux DDBs. Leur complexité est polynomiale par rapport à la taille des DDBs et ils délivrent des graphes canoniques. Mais un des inconvénients des *ROBDDs* est que l'ordre des variables influe beaucoup sur leur efficacité. La détermination a priori de l'ordre le plus efficace est un problème NP-difficile.

4.2 Fonctions discrètes à variables booléennes

Une solution au problème de l'explosion combinatoire de certaines représentations avec des *ROBDDs* est d'étendre le concept et d'utiliser des fonctions à valeur numériques et non plus booléennes. Ainsi une seule structure peut représenter toutes les sorties au lieu d'utiliser un DDB par sortie. Les *ADDS(Algebraic Decision Diagrams)* comme les *MTBDDS (Multi-Terminal BDD)* [10] sont des structures dérivées des DDBs dans lesquelles les noeuds terminaux représentent des valeurs numériques arbitraires dans $\mathbb{Z}$ et non plus restreintes à $\mathbb{B}$. Un ADD représente une fonction discrète $f(x_1, x_2, ..., x_n) : \mathbb{B}^n \rightarrow \mathbb{Z}$ où chaque noeud est soumis à un test d'une variable booléenne x_i.



(a) *26 Voisins d'un voxel* (b) *Arbre binaire* (c) *Algorithme*

FIG. 1 – Triple ADD

5 Choix d'un DDB pour le dénombrement des éléments géométriques

L'efficacité des diagrammes de décision est fortement lié à leur taille et à leur coût de construction. Ces paramètres dépendent essentiellement de trois critères : la fonction à représenter, l'ordre des variables booléennes et la stratégie de construction.

5.1 Choix d'un arbre binaire de décision

Dans [6], Robert et Malandain proposaient d'utiliser les ROBDDs pour des techniques classiques de traitement d'image qui dépendent d'une seule fonction booléenne à variable booléenne. Notre problème consiste à implémenter dans le même DDB les 3 fonctions (Δn_2, Δn_1, Δn_0) présentées dans la partie 3. Ces fonctions sont des fonctions discrètes à variables booléennes. Notre choix se porte donc sur l'utilisation d'un ADD. Bien que ces 3 fonctions discrètes dépendent de la valeur des 13 voisins, elles restent indépendantes. En associant à chaque équation un ADD, nous devons implémenter 3 ADDs. Pour accélérer notre algorithme, nous nous sommes focalisés sur le critère suivant : minimiser le nombre d'accès à une image. Nous avons donc choisi de n'implémenter qu'un seul ADD pour les 3 fonctions discrètes. Nous l'avons appelé : *triple-ADD*. Chaque noeud terminal de ce triple-ADD n'incrémente plus une seule équation discrète, mais trois. Il est donc impossible d'utiliser un module «déjà existant» pour créer ce triple-ADD. L'utilisation d'un triple-ADD pour 3 fonctions discrètes permet certes un gain de temps par rapport à l'utilisation de 3 ADDs simples. Mais le choix de l'ordre des variables pour un ADD simple est plus facile à mettre en oeuvre, puisqu'il suffit d'écrire la fonction sous sa forme canonique. Pour un triple-ADD, nous devons définir un ordre qui tienne compte à la fois des variables booléennes utilisées dans les 3 équations et de leur fréquence d'apparition. Nous nous sommes appuyés sur la topologie 3D, et notamment sur la connexité pour ordonner les variables de notre triple-ADD.

5.2 Choix de l'ordre des variables

5.2.1 Notion de topologie discrète 3D

L'étude topologique des images binaires nécessite l'utilisation des différentes connexités discrètes. La connexité consiste à définir des relations d'adjacence entre les noeuds de la grille. Pour une image 3D, il existe trois manières de définir la notion de voisinage entre voxels. Deux voxels sont dits 6-connexes s'ils partagent une face. Deux voxels sont dits 18-connexes s'ils partagent une face ou une arête. Deux voxels sont dits 26-connexes s'ils partagent une face ou une arête ou un sommet. En utilisant la notation de la figure 1.a, nous proposons la classification suivante pour les 26 voisins de V en fonction de leur connexité :

- 6-connexe : N01, N22, N42, N62, N82, N03
- 18-connexe : N21, N41, N61, N81, N12, N32, N52, N72, N23, N43, N63, N83
- 26-connexe : N11, N31, N51, N71, N31, N33, N53, N73

5.2.2 Détermination d'un ordre topologique

Comme notre critère consiste à minimiser le nombre d'accès aux voxels de l'image, nous devons choisir un ordre de branchement de telle sorte qu'à chaque test, le plus grand nombre d'éléments géométriques soit éliminé. Un voxel 6-connexe adjacent au voxel courant V par 1 face élimine immédiatement 1 face, 4 arêtes et 4 sommets. C'est le cas de N01, N22 et N82. La fréquence d'apparition de tels voxels est de 1 fois pour Δn_2, 4 pour Δn_1 et 4 pour Δn_0. Un voxel 18-connexe adjacent à V par 1 arête élimine immédiatement 1 arête et 2 sommets. C'est le cas de N21, N81, N41, N61, N12 et N32. Un voxel 26-connexe adjacent à V par 1 sommet élimine 1 sommet. C'est le cas de N11, N31, N71 et N51.
Nous avons donc choisi la connexité comme stratégie pour définir un ordre entre nos 13 variables booléennes. Les premiers tests sont effectués sur les voxels 6-connexes, puis 18-connexes, et enfin 26-connexes. Les voxels de même connexité sont ordonnés en fonction du sens de parcours de la matrice. L'ordre de présentation des 13 variables booléennes pour notre triple-ADD est donc le suivant : N01 $<$ N22 $<$ N82 $<$ N21 $<$ N81 $<$ N41 $<$ N61 $<$ N12 $<$ N32 $<$ N11 $<$ N31 $<$ N71 $<$ N51.
Sur la figure 1.b, nous avons représenté le début de notre triple-ADD ordonné et réduit. Par exemple, si les 3 voxels 6-connexes sont pleins, il n'est pas nécessaire de continuer les tests après avoir analysé la valeur de ces voxels (Q01=Q22=Q82=0). Δn_2 reste égal à 3, Δn_1 reste égal à 3 et Δn_0 reste égal à 1. Le diagramme obtenu est composé de 181 noeuds terminaux dont la répartition est donnée dans le tableau 1. Nous avons implémenté le triple-ADD en C++. Lors de la programmation, nous avons choisi de respecter la structure du DDB en transformant le diagramme en une suite de branchements conditionnels sous forme de «si imbriqué» comme le montre la figure 1.c. Certes le code source obtenu est un peu long, mais très efficace : à chaque étape de l'exécution, l'algorithme garantit n'examiner que les données d'entrées pertinentes, c'est à dire les valeurs qui affecte le résultat. Pour chaque valeur, trois tests sont au moins réalisés : la longueur minimale des branchements est donc de 3 lorsque les 3 voxels 6-connexes sont pleins. La longueur maximale des branchements est de 13. Dans [6], en moyenne pour chaque voxel, 8.7 voisins étaient examinés pour traiter une seule fonction booléenne. Dans notre algorithme, il suffit en moyenne pour chaque voxel d'examiner 9.5 voisins pour mettre à jour en même

temps les 3 fonctions booléennes. On peut estimer que le nombre de branchements le plus fréquemment rencontré pour un voxel plein est de longueur 3. Ce cas correspond à un voxel plein situé à l'intérieur d'un objet 3D.

Longueur du branchement	3	4	5	6	7	8	9	10	11	12	13	Total
Nombre de branchements	2	2	7	7	15	14	34	32	36	16	16	181
Longueur × Nombre	6	8	35	42	105	112	306	320	396	192	208	1730

TAB. 1 – *Répartition du nombre de branchements conditionnels dans le triple-ADD*

6 Évaluation Expérimentale

Dans cette partie, nous allons montrer l'efficacité de l'utilisation d'un triple-ADD ordonné et réduit pour dénombrer les faces ouvertes, arêtes ouvertes et sommets ouverts. Nous avons implémenté et comparé trois algorithmes. Le premier est l'«algorithme de Michielsen et de Raedt» tel qu'il a été donné dans [4]. Le second implémente les 3 équations Δn_2, Δn_1, Δn_0 sans optimisation ; il est nécessaire de connaître pour chaque voxel plein la valeur des 13 voxels voisins précédents. On l'appellera «algorithme des équations». Le dernier algorithme a le plus long code source puisqu'il implémente le triple-ADD réduit et ordonné présenté précédemment. On l'appellera «algorithme du triple-ADD». Ces programmes ont été compilés sous Visual C++ 6.0 avec un PC Duron 1 GHz (mémoire cache de niveau 1 de 128 Ko et 256 Mo de RAM).
De nombreux systèmes observés dans la nature peuvent être modélisés par un ensemble de points [11]. Pour étudier leurs caractéristiques (répartition aléatoire, regroupement, ...), les systèmes de points peuvent être considérés comme des images en noir. Pour comparer les temps d'exécution des algorithmes, nous avons utilisé un modèle booléen de germination. Dans un domaine cubique, considérons un ensemble de N voxels noirs. Ces voxels, qui sont les germes du modèle, sont positionnés de manière aléatoire (loi de Poisson) et correspondent à des cubes de côté 1. Nous appelons grains, tous cubes élargis de côté $2r + 1$, $r \geq 0$. En utilisant le processus de germination qui consiste à transformer un modèle de N germes en un modèle de grains, nous allons étudier l'évolution des temps de calcul de n_2, n_1 et n_0 en fonction r. Les premiers tests ont été réalisés sur un modèle de $N = 1024$ germes pour une image de $128 \times 128 \times 128$ ($L_x = 128$). Le nombre maximal de voxels noirs dans une telle image 3D sera de $126 \times 126 \times 126 = 2\ 000\ 376$ parce que nous laissons une frontière d'un voxel blanc tout autour de la matrice 3D. La figure 2 représente les temps d'exécution des 3 algorithmes au cours du processus de germination. En moyenne, l'algorithme du triple-ADD est 17 fois plus rapide que l'algorithme de Michielsen et deux fois plus rapide que l'algorithme des équations. Nous avons utilisé un second modèle de $N = 4000$ germes dans une image de $500 \times 500 \times 500$ ($L_x = 500$). La figure 3 représente les temps d'exécution pour l'algorithme des équations et l'algorithme du triple-ADD. Pour de plus gros volumes de données, l'algorithme du triple-ADD reste en moyenne $2,2$ fois plus rapide que l'algorithme des équations. On peut décomposer ces deux courbes en trois parties. Tout d'abord, r varie de 0 à 5 : les cubes sont petits et isolés les uns des autres. Ensuite, le rayon r augmente jusqu'à 25 et les grains se rejoignent. Enfin, toute la matrice image est remplie.

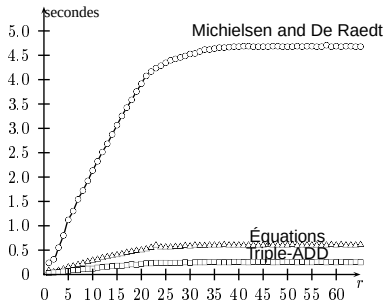


FIG. 2 – *Temps d'exécution ($N = 1024$).*

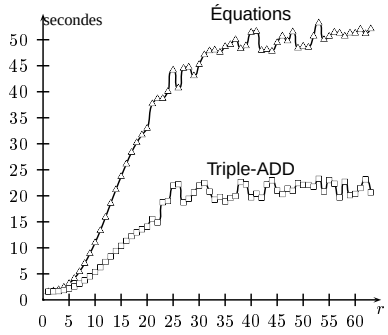


FIG. 3 – *Temps d'exécution ($N = 4000$).*

7 Conclusions et perspectives

Dans ce papier, nous avons proposé un algorithme qui s'appuie sur la topologie et utilise des diagrammes de décision binaires. L'utilisation d'un triple-ADD réduit permet ainsi de dénombrer rapidement le nombre de faces ouvertes, d'arêtes ouvertes et de sommets ouverts d'une image discrètes 3D, même sur des volumes de données importants. Le nombre de cubes ouverts correspond aux nombre de voxels noirs dans l'image. Une fois, ce dénombrement réalisé, il est possible de calculer les fonctionnelles de Minkowski.
Dans la suite, nous souhaitons utiliser la méthode présentée pour accélérer la classification de corps convexes et non convexes [12].

Références

[1] P.J. FLYNN, A. HOOVER, and P.J. PHILLIPS. Special issue on empirical evaluation of computer vision algorithms. *Computer Vision and Image Understanding*, 84 :1–4, 2001.

[2] J. SERRA. *Image analysis and mathematical morphology.* Academic Press Inc, London 1982.

[3] A.S. SZALAY, J. GRAY, and J. VANDENBERG. Petabyte scale data mining : dream or reality. *Proc. SPIE Conference on Advanced Telescope Technologies*, 4836, August 2002, Hawaii.

[4] K. MICHIELSEN and H. DE RAEDT. Aspects of integral-geometry. *Advances in Imaging and Electron Physics*, 125, Academic Press, 2002.

[5] R.W. HALL and C.-Y. HU. Time-efficient computation of 3d topological functions. *Pattern Recognition Letters*, 17 :1017–1033, 1996.

[6] L. ROBERT and G. MALANDAIN. Fast binary image processing using binary decision diagrams. *Computer Vision and Image Understanding*, 72(1) :1–9, October 1998.

[7] C.Y. LEE. Representing of switching functions by binary decision programs. *Bell Systems Technical Journal*, 38 :985–999, 1959.

[8] R.E. BRYANT. Symbolic boolean operation with ordered binary decision diagrams. *ACM Computing Surveys*, 24(3) :293–317, September 1992.

[9] S. BISCHOFF and L. KOBBELT. Isosurface reconstruction with topology control. *Pacific Graphics 2002 Proceedings*, October 2002, Tsinghua University, Beijing.

[10] R.I. BAHAR, E.A. FROHM, C.M. GAONA, G.D. HACHTEL, E. MACII, A. PARDO, and F. SOMENZI. Algebraic Decision Diagrams and Their Applications. *Proc. ACM/IEEE International Conference on CAD*, pages 188–191, 1993.

[11] U. BRODATZKI and K. MECKE. Simulating stochastic geometries : morphology of overlapping grains. *Computer Physics Communications*, 147 :218–221, 2002.

[12] J.-F. POIRAUDEAU and I. BLASQUEZ. Analysis of sets of convex bodies in 3d space with Minkowski functionals. *Proceedings of SPIE-Vision Geometry XI*, 4794, 7-8 July 2002, Seattle, WA.