LSL3D : Etiquetage en Composantes Connexe par segments pour volumes 3D

Nathan Maurice, Florian Lemaitre, Julien Sopena, Lionel Lacassagne

Compas'2022 : Parallélisme/ Architecture/ Système
MIS/UPJV - Amiens France, 5-8 juillet 2022

LSL3D : Étiquetage en Composantes Connexe par segments pour volumes 3D

Nathan Maurice, Florian Lemaitre, Julien Sopena, Lionel Lacassagne

LIP6, Sorbonne Université, CNRS
nom.prénom@lip6.fr

Résumé
L'Étiquetage en Composantes Connexe (*ECC*) est une opération fondamentale de la vision numérique depuis des décennies. Bien que la majeure partie de la littérature traite des algorithmes en 2D, sur des applications telles que la vidéo-surveillance ou la conduite autonome, il y a de plus en plus besoin d'algorithmes pour traiter des images en 3D, notamment pour des applications médicales.
Bien que les algorithmes d'*ECC* en 2D génèrent un nombre important d'accès mémoire et de comparaisons, le problème empire avec algorithmes en 3D. Il s'agit de la *malédiction de la dimension*. La mise en place d'un algorithme efficace doit donc résoudre ce problème. Ce papier introduit un algorithme utilisant des segments pour étiqueter des volumes en 3D. Les équivalences entre les étiquettes sont accélérées à l'aide d'une nouvelle stratégie atténuant l'impact de la dimension supérieure. Ce nouvel algorithme surpasse les algorithmes de l'État-de-l'art d'un facteur allant de $\times 1.5$ à $\times 3.1$ sur des images médicales et sur des images aléatoires.

1. Introduction

L'étiquetage en composantes connexes (*ECC*) est un problème fondamental du traitement d'images depuis plusieurs décennies [26][10]. Il consiste à trouver les composantes connexes (ensembles de pixels adjacents) dans une image binaire et à leur assigner un identifiant unique appelé *étiquette*. Les applications de ce traitement sont nombreuses : on y retrouve la conduite autonome [29][7], la vidéo-surveillance [13][27], l'imagerie médicale [6][25][1][20][14] ainsi que d'autres domaines [24] pour lesquels une implémentation cadence temps-réel est importante.
Très étudié en 2D, des algorithmes d'*ECC* existent également pour la 3D, mais des progrès restent à faire, notamment pour atteindre des performances permettant d'assurer un traitement temps-réel. En effet, avec le passage à la 3D, le voisinage s'agrandit à 6 voisins sans diagonales (6-connexité) et à 26 avec (26-connexité), ce qui d'une part fait exploser le nombre d'opérations effectuées et d'autre part pose des problèmes de localité mémoire.
Dans cet article, un nouvel algorithme d'étiquetage nommé *LSL3D* est proposé pour le traitement d'images 3D en 26-connexité. Les contributions sont 1) l'utilisation d'une machine à état (FSM) pour traiter efficacement les segments en utilisant un encodage par *Run-Length Encoding* (RLE) et 2) un mécanisme de cache permettant une réutilisation de résultats partiels et réduisant le nombre de calculs.
Une étude de performance a montré que notre algorithme par segments est de $1.8\times$ à $2.3\times$ plus rapide que l'État-de-l'Art sur des bases d'images médicales. De plus, sur des images aléatoires en 3D, davantage stressantes, à granularité faible, le *LSL3D* est de $1.5\times$ jusqu'à $3.1\times$ plus rapide.
L'article se compose de la façon suivante : La Section 2 donne une vue d'ensemble de l'*ECC* ainsi que les difficultés pour la 3D. La Section 3 fait un état-de-l'art de la littérature existante. Ensuite, la Section 4 introduit trois stratégies pour gérer les équivalences entre étiquettes tout en atténuant la *malédiction de la dimension*. Pour finir, la Section 5 étudie l'impact du SIMD sur les algorithmes d'*ECC* en 3D

2. Approches classiques et évaluation de l'étiquetage en composantes connexe

Fondamentalement, les algorithmes d'*ECC* regroupent les pixels d'avant-plan (pixels blancs lorsque l'image est sur fond noir) connectés. Pour toute image binaire, il n'existe qu'une unique solution à son étiquetage. Ainsi, il ne peut y avoir de gain qualitatif entre algorithmes et la recherche s'est donc concentrée sur l'accélération du traitement.
On parle de 8-connexité lorsque l'on considère tous les voisins d'un point et de 4-connexité lorsque l'on ignore les diagonales. Dans la suite de cet article, on ne considère que la 8-connexité, celle-ci trouvant davantage d'applications.
Dans cette section, nous présenterons les principes de base de ces algorithmes ainsi que la méthodologie d'évaluation : métriques, jeux de données et plateforme d'exécution.

2.1. Principes généraux des algorithmes d'*ECC* modernes

Tous les algorithmes modernes suivent la structure initialement proposée par Rosenfeld en 1966 [26] ou Haralick et Shapiro en 1981 [10]. Ils sont ainsi composés de 3 étapes :

1. un **premier étiquetage** qui créé des étiquettes temporaires, tout en capturant les connections entre voisins dans une table d'équivalence. Cette table peut être vue comme un graphe orienté du voisinage.
2. une **résolution d'équivalence** qui détermine la *Fermeture Transitive* (FT) du graphe associé à la table d'équivalence,
3. un **étiquetage final** pour remplacer les étiquettes temporaires par les étiquettes finales (habituellement, la plus petite de chaque composante).

Les algorithmes modernes utilisent des optimisations algorithmiques pour accélérer ces étapes. Comme ces algorithmes sont en général limités par leur *control-flow* plutôt que par les accès mémoire ou les calculs, les banque d'images utilisées pour l'évaluation des performances ont un impact sur celles-ci.

2.2. Procédure de benchmark et jeux de données

Afin d'évaluer la performance des algorithmes d'*ECC*, la communauté a mis en place une suite de benchmarks avec *YACCLAB* [4]. On y retrouve deux jeux d'images médicales en 3D (domaine applicatif de l'*ECC*) (Figures 5 et 6) : *mitochondria* [21] qui contient 3 images et *OASIS* [22] qui contient 373 images. Les images de *mitochondria* sont composées de *blobs* (volumes pleins) de grandes tailles ainsi que de quelques petites composantes connexes. À l'inverse, les images de *OASIS* sont composées d'un volume creux et complexe.
Outre ces jeux de données tirés de cas réels, s'ajoutent des générateurs d'images aléatoires qui permettent de faire varier à la fois la densité (proportion de pixels blancs dans l'image) et la granularité (taille de chaque bloc au moment de la génération) (Figure 7).
Ainsi, les résultats de l'article peuvent être reproduits en utilisant le générateur MT19937 [23] et en faisant varier la densité (notée d dans la suite de l'article) de 0% à 100% ainsi que la granularité (notée g) de 1 à 16. Tous les résultats présentés par la suite ont été obtenu sur un Intel Xeon Gold 6126.

3. État de l'art des algorithmes d'*ECC* en 3D

Si tous les algorithmes efficaces reprennent le principe évoqué en Section 2.1, de nombreuses propositions ont étés faites pour accélérer le traitement en trois phases. Ces travaux ont ciblé, dans la majorité, les images 2D ; les travaux 3D adaptant souvent ces propositions sans proposer de mécanisme spécifique. Ainsi, dans cette section, nous présenterons les trois principales évolutions et leur adaptation à la 3D.
Dans cet état de l'art, seront notés en gras tous les algorithmes retenus comme points de comparaison.

3.1. Algorithmes par pixel

De façon naïve, la première phase considère tous les voisins pour chaque pixel, ce qui est très couteux. Ainsi, en utilisant un parcours séquentiel, *Rosenfeld* [26] montre qu'il est possible de diviser par deux le nombre de voisins considérés. On appelle cet ensemble de voisins *masque de Rosenfeld*.
Cette approche par masque a été améliorée par Wu [30] dans l'algorithme *SAUF* en remarquant qu'il n'est pas toujours nécessaire de consulter tout le masque. Il est par exemple inutile de regarder les autres

pixels du masque lorsque celui du dessus est présent. *SAUF* repose donc sur un arbre de décision qui capture les différents cas. Cet algorithme a été adapté à la 3D par Bolelli [2] avec l'algorithme *SAUF 3D*. Plusieurs travaux ont par ailleurs cherché à optimiser cet arbre de décision, notamment par He en 2D [11] et en 3D [19] (***LEB 3D***).
Si les propositions basées sur le masque ou sur l'arbre de décision exploitent le fait que le traitement a déjà été fait pour certains points du voisinage, elles ne tiennent pas compte des décisions précédentes. L'idée de He [12] est de s'appuyer sur la décision précédente de façon à simplifier l'arbre. Il propose ainsi un algorithme à états basé sur un graphe d'arbre de décisions pouvant être vu comme un automate. Cette approche est généralisée par Grana avec l'algorithme *PRED* [8], ensuite étendu à la 3D par Bolelli avec *PRED 3D* [2]. L'introduction de *Direct Rooted Acyclic Graphs* (*DRAG*) par Bolelli [3] a permis de réduire l'empreinte du code de décision. Les *DRAG* ont ainsi étés utilisés par Bolelli [2] pour optimiser des algorithmes existants comme avec *SAUF++* et *PRED++*, avec des versions adaptées à la 3D (***SAUF++ 3D*** et ***PRED++ 3D***).

3.2. Algorithmes par blocs

Afin de réduire le nombre de comparaisons dans la première phase, Grana [9] propose une approche par bloc (les pixels dans un même bloc 2×2 étant nécessairement dans la même composante). L'arbre de décision utilisé pour les algorithmes par bloc est amélioré par Chabardes [5] avec une forêt d'arbres de décision, qui est elle-même ensuite remplacée par un *DRAG* [3].
Étendre cette approche à un cube $2 \times 2 \times 2$ n'est en pratique pas évident, car cela complexifie énormément l'automate. L'algorithme *Entropy Partition Decision Tree* [28] repose sur une approche intermédiaire avec des blocs $2 \times 1 \times 1$ (***EPDT_19c*** et ***EPDT_22c***, avec prise en compte de 19 et 22 pixels voisins respectivement) ou $2 \times 2 \times 1$ (***EPDT_26c***).

3.3. Algorithmes par segments

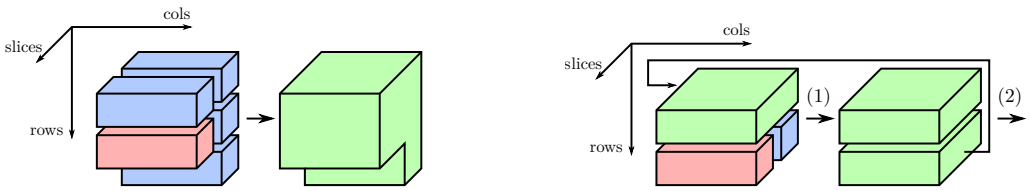
Bien que les approches par blocs se montrent efficaces, il est également possible de grouper les pixels en segments : suites continues de pixels. Un pré-traitement d'encodage en segments (*Run Length Encoding*, noté RLE par la suite) est alors appliqué avant la recherche des composantes. Ensuite, l'algorithme d'étiquetage établi les adjacences entre segments (superpositions entre segment actuel et segment de la ligne précédente) et effectue les fusions de composantes si nécessaire.
Cette approche par segments est employée par He dans *Run Based Two Scans* (*RBTS*) sur images 2D [18] et en 3D avec ***RBTS 3D*** [19]. L'utilisation de segments a également été proposée par Lacassagne avec le *Light Speed Labeling* (*LSL*)[15][16] dans le cadre des images 2D. Outre un encodage RLE, il introduit plusieurs structures de données intermédiaires afin de simplifier le calcul des voisinages, en compensant la perte des coordonnées des pixels par une position relative des segments. Bien que le *LSL* soit aujourd'hui l'un des algorithmes les plus efficaces en 2D, aucune adaptation à la 3D n'a été proposée. Il est pourtant un très bon candidat puisque la compression en segments permettrait d'atténuer l'explosion combinatoire qui accompagne l'ajout d'une troisième dimension.

4. LSL3D : stratégie d'unification efficace pour volumes 3D

Cette section présente, pas à pas, la transformation et l'amélioration du *LSL* classique 2D en un nouvel algorithme 3D optimisé :

- Une première étape étend le *LSL* vers la 3D en gardant un étiquetage relatif par ligne (version **LSL_ER**). Une difficulté de ce travail est la réorganisation des structures de données pour exploiter la localité spatiale.
- La deuxième étape consiste à remplacer la table ER par une Machine à États Finie (FSM) (**LSL_FSM**) (et les calculs associés)
- la troisième étape repose sur l'ajout d'un mécanisme de cache de résultats partiels, pour réaliser l'union/unification avec des double-lignes (**LSL_FSM_DOUBLE**), afin de tirer parti de la localité temporelle (là où la première étape ciblait la localité spatiale)

Les implémentations successives du *LSL* ont été testé suivant le benchmark de la Section 2.2. Elles sont comparées à 7 algorithmes de l'État-de-l'Art : *LEB*, *RBTS*, *PRED++ 3D*, *SAUF++ 3D*, *EPDT_19c*, *EPDT_22c* et *EPDT_26c*. Parmi les algorithmes *EPDT*, seul le meilleur est présenté (*EPDT_22c*).
Les Figures 2 et 3 présentent les résultats, ceux-ci sont analysés dans les sections qui suivent.



(a) Unification 3D classique (b) Unification 3D avec double-lignes

FIGURE 1 – Unification par segments en 3D

4.1. Extension de l'unification par segments pour la 3D

Afin de trouver les superpositions de segments sans parcourir la ligne courante plusieurs fois, le premier *LSL* [16] en 2D utilise deux tables ER (ligne courante et précédente) pour stocker les positions relatives des segments. Dans la 3D, en reprenant l'idée du masque de Rosenfeld, une unification est réalisée entre la ligne courante et les 4 lignes voisines au moyen de 5 tables ER (Figure 1a), ce qui n'est pas sans poser des problèmes de localité spatiale puisque ces 5 tables correspondent à deux plans différents.
La performance de notre algorithme *LSL_ER* en 3D est visible sur la Figure 2. *LSL_ER* devient plus rapide pour $g > 4$ sur les images aléatoires et creuse l'écart pour des granularités plus élevées (jusqu'à $\times 1.3$ pour $g = 16$)
Pour les images médicales, *LSL_ER* est dans l'ensemble plus rapide que l'Etat-de-l'Art (Figure 3) : bien que *RBTS* soit $\times 1.1$ plus rapide sur *mitochondria*, il est également $\times 1.3$ plus lent sur *OASIS*. À l'inverse, bien que *PRED++ 3D* soit aussi rapide sur *OASIS*, il est $\times 1.3$ plus lent sur *mitochondria*.
Les limites du *LSL* s'expliquent par la durée de l'étape de RLE, particulièrement sur les grandes images (de 60% à 70% du temps d'exécution sur *mitochondria*). Non seulement le RLE créé une table de segment (RLC) mais en plus, l'initialisation de la table ER est couteuse, car celle-ci contient 1 élément par pixel.

4.2. Une unification par machine à état finie

La recherche de superpositions entre segments peut également être réalisée sans ER via l'utilisation d'une machine à états finie (FSM). Dans l'algorithme 2D [17], chaque état de la FSM encode une configuration de segments entre la ligne courante et la ligne précédente. La fusion de deux lignes revient donc à itérer sur ces deux lignes en même temps : une nouvelle étiquette est créée pour chaque segment isolé alors que deux segments en contact voient leur composante fusionnées.
Si cette amélioration de l'algorithme se montre performante sur des images simples, il n'en va pas de même pour des images plus complexes. Ainsi, sur la Figure 2, qui présente les résultats des expérimentations sur les images aléatoires, le passage à une machine à état accélère d'un facteur $\times 1.3$ le traitement d'images à forte granularité ($g = 16$), mais est $\times 0.61$ plus lent avec une image pixelisée ($g = 1$).
Une tendance similaire peut être observée sur les images médicales (Figure 3) : sur *mitochondria* (granularite elevée), *LSL_FSM* améliore le temps d'exécution par rapport à *LSL_ER* d'un facteur $\times 1.1$, et à l'inverse *LSL_FSM* est plus lent sur *OASIS* d'un facteur $\times 0.95$.
Ces résultats s'expliquent essentiellement par le surcoût engendré dans la phase d'unification. En effet, même si la phase RLE est accélérée par l'absence du calcul de ER, la complexité de la machine à états 3D (27 états et 55 transitions, contre 8 états et 14 transitions pour la FSM en 2D) dégrade fortement les performances du prédicteur de branchement lors de cette deuxième phase. Cela est naturellement plus prégnant sur des images complexes, où il y a plus d'unifications avec une variété de configurations (et donc de transitions empruntées) plus grande.

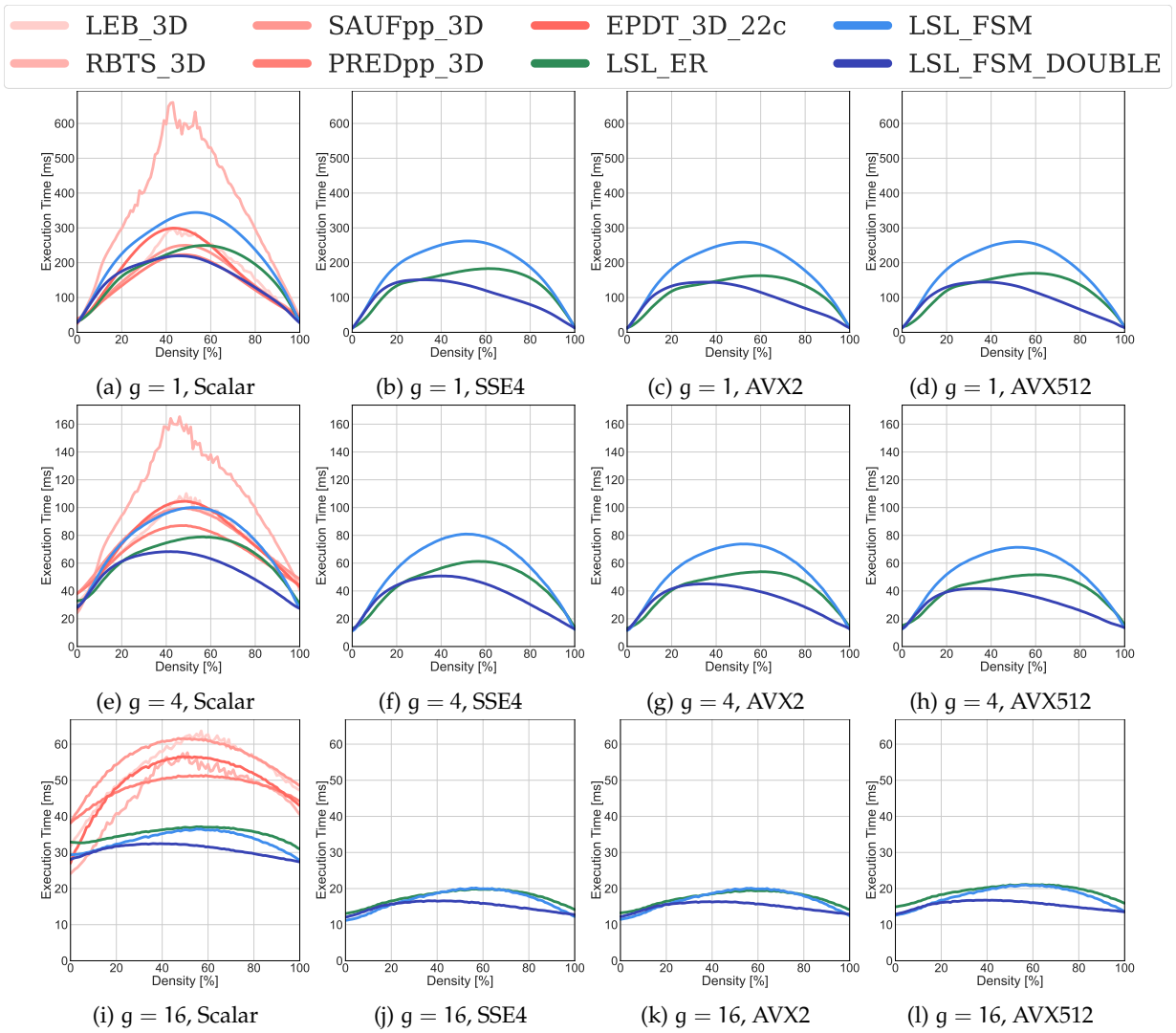
4.3. Cache partiel de lignes fusionnées

La complexité de la machine à état posant des problèmes de performance sur les images complexes, nous avons repensé l'algorithme pour tenir compte de la séquentialité des traitements en conservant une partie des calculs intermédiaires. Outre la simplification des calculs, ceci va permettre de simplifier la machine à états (qui passe de 27 états et 55 transitions à 9 états et 18 transitions).
Plus précisément, comme le montre la Figure 1, deux itérations consécutives sur la même tranche traitent trois lignes plusieurs fois : la ligne courante (en rouge) et deux lignes voisines (en bleu) vont être retraitées dans la fusion suivante. On peut donc supprimer les calculs redondants en mettant en cache les résultats partiels (lignes vertes) dans un tableau de *double-lignes*.

Dans un premier temps, il est possible de réduire le masque de lignes adjacentes à un masque 2×2. Bien que cela complexifie le traitement (l'adjacence avec le 4ème voisin devant se faire à l'itération suivante), cela permet d'établir une symétrie avec le plan précédent. Cette symétrie permet de réutiliser le calcul effectué (Figure 1b). L'unification se fait ainsi en deux étapes : une première (1) fusionne la ligne actuelle (en rouge) et sa voisine du plan précédent (en bleu). Cela produit une *double-ligne* (en vert) qui contient la superposition des segments des deux lignes. Ensuite, une seconde étape (2) fusionne la double-ligne avec celle créée durant l'itération précédente. La double-ligne produite à l'étape (1) est réutilisée pour l'itération suivante pour éviter de la recalculer.
Le test des performances du *LSL_DOUBLE* sur images aléatoire (Figure 2) se montre concluant. Il garde en effet l'efficacité du *LSL_FSM* sur les images simples ($g = 4$ et $g = 16$) tout en se montrant performant sur des images plus complexes ($g = 1$). Il est ainsi $\times 1.3 - 1.5$ plus rapide que le meilleur algorithme sur $g = 4$ et $g = 16$ et seulement $\times 0.94$ moins rapide que le meilleur sur $g = 1$.
Outre ces bons résultats, on remarque sur la Figure 2 que *LSL_DOUBLE* résiste bien mieux à l'augmentation de la densité (écart entre la courbe verte et violette, au-delà d'une densité de 25%). En effet, pour ces niveaux de densités, le nombre de segments est statistiquement élevé, ce qui dégrade d'autant les algorithmes d'*ECC* basés sur un encodage RLE (par segments). Dans le cas de *LSL_DOUBLE*, ce phénomène est atténué par la fusion des segments au sein d'une double-ligne : plus le nombre de segments est élevé, plus il y aura de segments superposés, plus il y aura de fusions et moins il y aura de segments au final dans la double-ligne. Le mécanisme de double-ligne est donc particulièrement efficace pour des images complexes.
Sur les images de *mitochondria*, *LSL_DOUBLE* se montre presque aussi rapide que *LSL_FSM* et que *RBTS 3D*, qui est le meilleur algorithme de l'état de l'art en scalaire sur des images simples. Sur le benchmark plus complexe de *OASIS*, *LSL_DOUBLE* est le plus performant (Figure 3). Ce nouvel algorithme est à la fois efficace sur des images simples comme sur des images complexes, là où *RBTS 3D* est peu performant sur *OASIS* ($\times 1.3$ plus lent).

5. Prise en compte d'optimisations spécifiques à l'architecture

Comme vu dans la section précédente, l'unification par double-ligne est efficace et réduit le temps d'exécution du *LSL3D* : le temps d'unification (qui ne requiert plus de table ER supplémentaire) se retrouve très optimisé. Le goulot d'étranglement devient alors la phase de pré-traitement (encodage RLE) et le ré-étiquetage final (phase 3, voir Section 2.1). Ainsi, sur la Figure 3, ils représentent 70% du temps d'exécution sur les images complexes (*OASIS*) et jusqu'à 90% sur les images simples (*mitochondria*).
Heureusement, ces phases se prêtent toutes deux bien à des travaux de parallélisation au niveau instruction avec le SIMD. Plusieurs propositions ont d'ailleurs étés faites dans ce sens pour des images 2D [17]. Nous avons donc repris l'idée de ces travaux pour les adapter aux images 3D et ainsi réalisé trois implémentations parallèles pour différents jeux d'instruction SIMD, à savoir le `SSE4`, l'`AVX2` et l'`AVX512`.
Les résultats présentés sur la Figure 2 (une colonne par implémentation) montrent que quelque soit le jeu d'instructions, l'utilisation du SIMD réduit le temps d'exécution de toutes nos versions sur les images aléatoires. À lui seul le `SSE4` suffit à rendre *LSL_ER* plus rapide que tous les algorithmes de l'état de l'art d'un facteur $\times 1.1$ à $\times 2.6$. On retrouve aussi ces bons résultats sur les benchmarks de la littérature (Figures 3a et 3b), avec un speedup de $\times 1.4$ à $\times 2.0$ plus rapide que l'état de l'art sur *mitochondria* et *OASIS* (Figure 3).
Le passage à des instructions plus complexes traitant des registres plus grand n'apportent pas de performances sur des images simples, mais permettent tout de même d'améliorer de 10% les performances sur les images complexes d'*OASIS*. On peut d'ailleurs observer ce même phénomène sur les images aléatoires où l'on observe une amélioration des performances avec une forte densité pour une granularité $g = 4$. Plus précisément, nous avons été confrontés, pour l'`AVX2` à l'absence d'instruction *compress* (sélection d'éléments à partir d'un masque) nous forçant à utiliser l'implémentation *SSE* de la *look-up table* durant l'étape de RLE, limitant d'autant le speed-up apportés par l'`AVX2`. Dans le cas de l'`AVX512`, l'instruction *compress* est bien présente. Malheureusement, sur le Xeon, elle n'est disponible que pour des éléments de 32-bits, nous obligeant à ajouter une conversion supplémentaire vers le 16-bits pour le traitement des segments (pour des raisons de localité mémoire, la table RLC utilise des entiers de 16-bits). Malgré toutes ces limites, notre algorithme *LSL_DOUBLE* et son implémentation sur `AVX512` se montre en moyenne $\times 1.5$ à $\times 3.0$ plus rapide que le meilleur algorithme de l'État-de-l'Art.



(a) g = 1, Scalar (b) g = 1, SSE4 (c) g = 1, AVX2 (d) g = 1, AVX512

(e) g = 4, Scalar (f) g = 4, SSE4 (g) g = 4, AVX2 (h) g = 4, AVX512

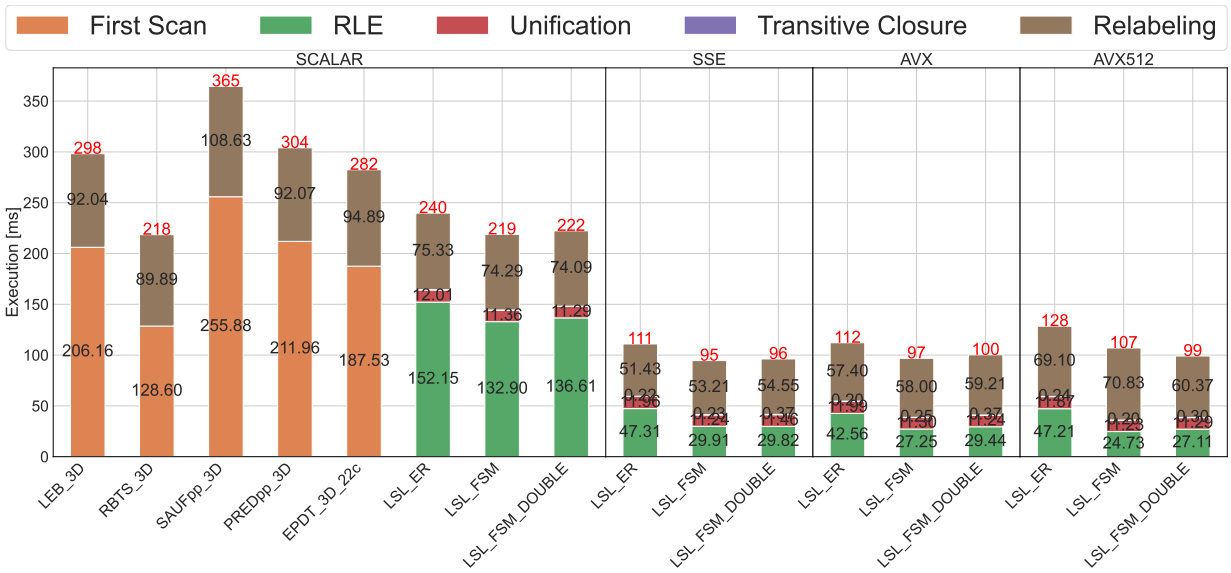
(i) g = 16, Scalar (j) g = 16, SSE4 (k) g = 16, AVX2 (l) g = 16, AVX512

FIGURE 2 – Temps d'exécution des algorithmes sur images aléatoires à granularité (g) et densité variables sur Xeon

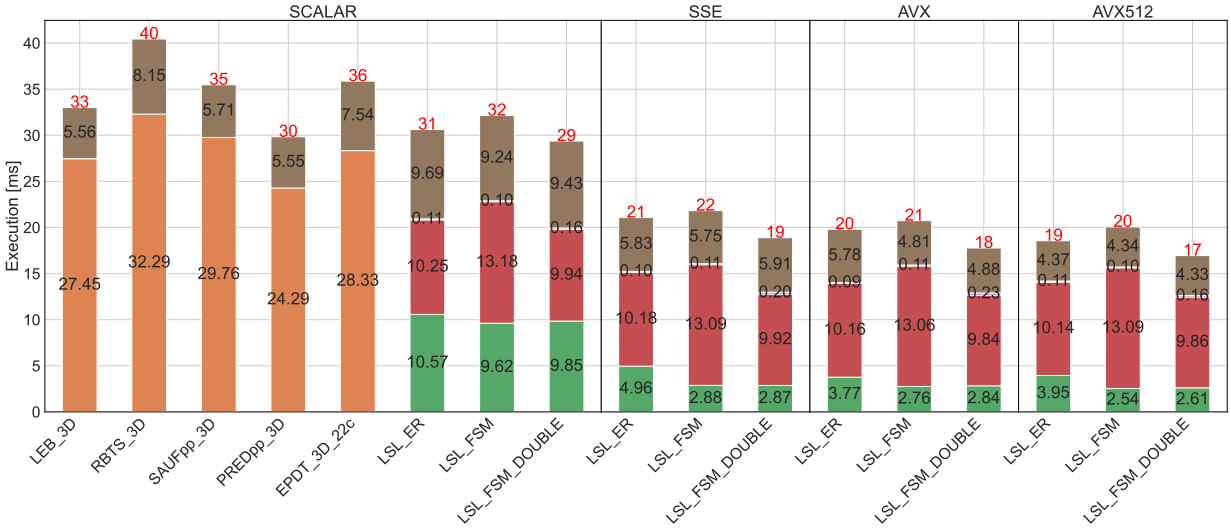
Malgré toutes ces limites, notre implémentation `AVX512` de notre algorithme *LSL3D* se montre beaucoup plus rapide que les meilleurs algorithmes de l'état de l'art, comme on peut le vérifier sur la Figure 4. on retrouve pour chacun des benchmarks (*mitochondria* et *OASIS*), les performances comparées des implémentations `AVX512` des différentes versions de *LSL3D* par rapport aux meilleurs temps des algorithmes de l'État de l'art. Ainsi, pour chaque image des deux benchmarks, on retient en ordonnée, le plus petit temps d'exécution parmi les meilleurs algorithmes de l'état de l'art (*LEB*, *RBTS*, *PRED++ 3D*, *SAUF++ 3D*, *EPDT_19c*, *EPDT_22c* et *EPDT_26c*), la valeur en abscisse correspond, elle, à la performance de notre algorithme. On peut y voir que nos algorithmes sont meilleurs dans tous les (points toujours au-dessus de la première bissectrice) et que la version *LSL_DOUBLE* (points bleu foncés) est la plus performante des trois pour des images complexes (*OASIS*).

6. Conclusion et travaux futurs

Cet article introduit un nouvel algorithme d'étiquetage en composantes connexes, *LSL3D*, qui combine un encodage par segment (RLE) pour diminuer globalement la complexité en 3D, une approche d'unification par machine à états finie pour être efficace sur les images simples, et un mécanisme de cache de double-lignes pour les images complexes. En plus d'une implémentation scalaire, nous utilisons les



(a) *mitochondria*, Scalar



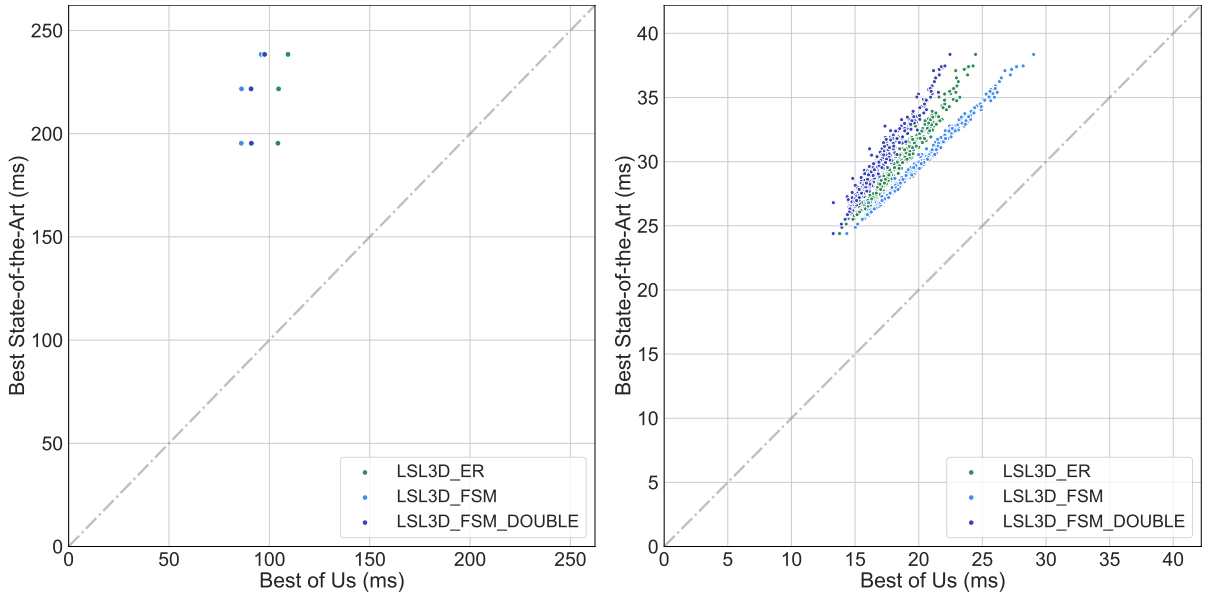
(b) *OASIS*, Scalar

FIGURE 3 – Temps d'exécution sur *mitochondria* et *OASIS* sur Intel Xeon

instructions SIMD (`SSE4`, `AVX2`, `AVX512`) pour en augmenter encore les performances.
L'évaluation des performances sur des images aléatoires ainsi que sur des bases d'images médicales, montre que *LSL3D* est $\times 1.5$ à $\times$ 3.1 plus rapide que les algorithmes de l'État-de-l'Art sur un Intel Xeon, comme le montre la Table 1.
Les travaux futurs vont traiter de la parallélisation de cet algorithme sur CPUs multicœurs et sur GPUs. La complexité d'une telle parallélisation se situe au niveau des stratégies de découpage (par tranches ou par cubes) et de la synchronisation des traitements des zones ainsi découpées. En effet, les composantes connexes recherchées peuvent être de forme arbitraire et donc s'étendre sur deux zones de travail. Deux threads sont alors amenés à traiter chacun une partie d'une même composante connexe.

Bibliographie

1. Abuzaghleh (O.), Barkana (B. D.) et Faezipour (M.). – Noninvasive real-time automated skin lesion analysis system for melanoma early detection and prevention. *IEEE journal of translational engineering in health and medicine*, vol. 3, 2015, pp. 1–12.



(a) *mitochondria* (b) *OASIS*

FIGURE 4 – Temps d'exécution des meilleurs *LSL3D* par rapport aux meilleurs algorithmes de l'État-de l'art

dataset	mitochondria	oasis		random g=1	random g=4	random g=16
speedup	× 2.3	× 1.8		× 1.5	× 2.2	× 3.1

TABLE 1 – Meilleurs speedups de *LSL3D* versus meilleur algorithme de l'Etat-de-l'Art

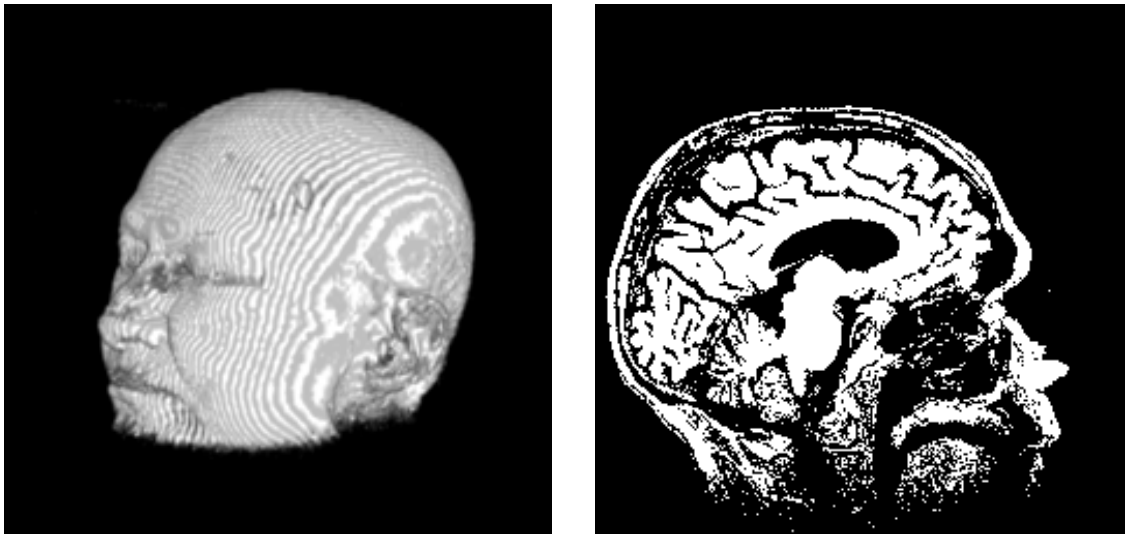
2. Bolelli (F.), Allegretti (S.) et Grana (C.). – One DAG to rule them all. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, janvier 2021.
3. Bolelli (F.), Baraldi (L.), Cancilla (M.) et Grana (C.). – Connected Components Labeling on DRAGs. – In *2018 24th International Conference on Pattern Recognition (ICPR)*, pp. 121–126, Beijing, août 2018. IEEE.
4. Bolelli (F.), Cancilla (M.), Baraldi (L.) et Grana (C.). – Toward reliable experiments on the performance of Connected Components Labeling algorithms. *Journal of Real-Time Image Processing*, vol. 17, n2, avril 2020, pp. 229–244.
5. Chabardès (T.), Dokládal (P.) et Bilodeau (M.). – A labeling algorithm based on a forest of decision trees. *Journal of Real-Time Image Processing*, vol. 17, n5, octobre 2020, pp. 1527–1545.
6. Chen (W.), Giger (M. L.) et Bick (U.). – A fuzzy c-means (FCM)-based approach for computerized segmentation of breast lesions in dynamic contrast-enhanced mr images. *Academic radiology*, vol. 13, n1, 2006, pp. 63–72.
7. Farhat (W.), Faiedh (H.), Souani (C.) et Besbes (K.). – Real-time embedded system for traffic sign recognition based on ZedBoard. *Journal of Real-Time Image Processing*, vol. 16, n5, 2019, pp. 1813–1823.
8. Grana (C.), Baraldi (L.) et Bolelli (F.). – Optimized Connected Components Labeling with Pixel Prediction. *In : Advanced Concepts for Intelligent Vision Systems*, éd. par Blanc-Talon (J.), Distante (C.), Philips (W.), Popescu (D.) et Scheunders (P.), pp. 431–440. – Cham, Springer International Publishing, 2016.
9. Grana (C.), Borghesani (D.) et Cucchiara (R.). – Optimized Block-Based Connected Components Labeling With Decision Trees. *Transactions on Image Processing*, vol. 19, n6, juin 2010, pp. 1596–1609.
10. Haralick (R.). – Some neighborhood operations. – In *Real-Time Parallel Computing Image Analysis*, pp. 11–35. Plenum Press, 1981.
11. He (L.), Chao (Y.) et Suzuki (K.). – A Linear-Time Two-Scan Labeling Algorithm. – In *2007 IEEE International Conference on Image Processing*, pp. V – 241–V – 244, San Antonio, TX, USA, 2007. IEEE.
12. He (L.), Zhao (X.), Chao (Y.) et Suzuki (K.). – Configuration-Transition-Based Connected-Component Labeling. *IEEE Transactions on Image Processing*, vol. 23, n2, février 2014, pp. 943–951.
13. Joshi (K. A.) et Thakore (D. G.). – A survey on moving object detection and tracking in video surveillance system. *International Journal of Soft Computing and Engineering*, vol. 2, n3, 2012, pp. 44–48.
14. Khan (N.), Ahmed (I.), Kiran (M.), Rehman (H.), Din (S.), Paul (A.) et Reddy (A. G.). – Automatic segmentation of liver & lesion detection using h-minima transform and connecting component labeling. *Multimedia Tools and*

dataset	# subsets	taille	densité	granularité	# runs	# CCs
mitochondria	3	1024×768×165	5.9	26.4	197,878	40
OASIS	373	256×256×128	19.8	4.2	236,718	3,200
random	16	256×256× 256	0 - 100	1 -16		

TABLE 2 – Caractéristiques moyennes des jeux de données en 3D

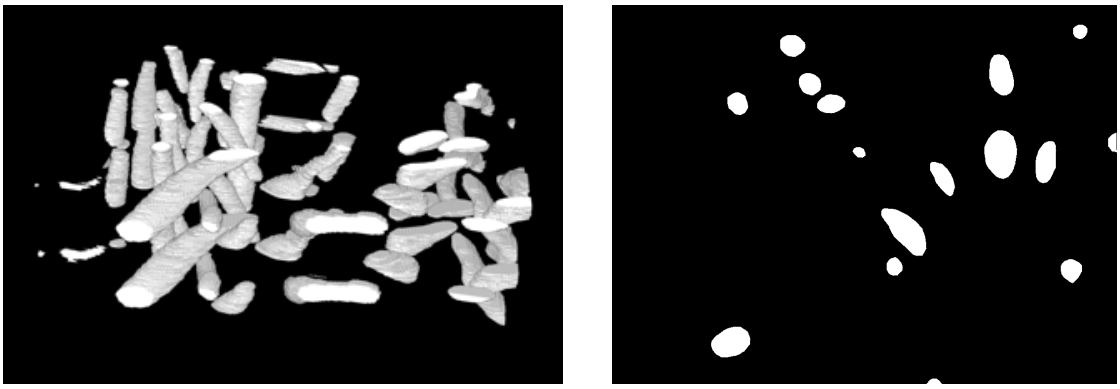
Applications, vol. 79, n13, 2020, pp. 8459–8481.
15. Lacassagne (L.) et Zavidovique (A. B.). – Light speed labeling for RISC architectures. – In *IEEE International Conference on Image Analysis and Processing (ICIP)*, 2009.
16. Lacassagne (L.) et Zavidovique (B.). – Light speed labeling : Efficient connected component labeling on RISC architectures. *Journal of Real-Time Image Processing*, vol. 6, n2, juin 2011, pp. 117–135.
17. Lemaitre (F.), Hennequin (A.) et Lacassagne (L.). – How to speed Connected Component Labeling up with SIMD RLE algorithms. – In *Proceedings of the 2020 Sixth Workshop on Programming Models for SIMD/Vector Processing*, pp. 1–8, San Diego CA USA, février 2020. ACM.
18. Lifeng He, Yuyan Chao et Suzuki (K.). – A Run-Based Two-Scan Labeling Algorithm. *IEEE Transactions on Image Processing*, vol. 17, n5, mai 2008, pp. 749–756.
19. Lifeng He, Yuyan Chao et Suzuki (K.). – Two Efficient Label-Equivalence-Based Connected-Component Labeling Algorithms for 3-D Binary Images. *IEEE Transactions on Image Processing*, vol. 20, n8, août 2011, pp. 2122–2134.
20. Litjens (G.), Sánchez (C. I.), Timofeeva (N.), Hermsen (M.), Nagtegaal (I.), Kovacs (I.), Hulsbergen-Van De Kaa (C.), Bult (P.), Van Ginneken (B.) et Van Der Laak (J.). – Deep learning as a tool for increased accuracy and efficiency of histopathological diagnosis. *Scientific reports*, vol. 6, 2016.
21. Lucchi (A.), Li (Y.) et Fua (P.). – Learning for Structured Prediction Using Approximate Subgradient Descent with Working Sets. – In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1987–1994, Portland, OR, USA, juin 2013. IEEE.
22. Marcus (D. S.), Fotenos (A. F.), Csernansky (J. G.), Morris (J. C.) et Buckner (R. L.). – Open Access Series of Imaging Studies : Longitudinal MRI Data in Nondemented and Demented Older Adults. *Journal of Cognitive Neuroscience*, vol. 22, n12, décembre 2010, pp. 2677–2684.
23. Matsumoto (M.) et Nishimura (T.). – Mersenne twister web page : http ://www.math.sci.hiroshima-u.ac.jp/ m-mat/MT/emt.html.
24. Millet (M.), Rambaux (N.), Petreto (A.), Lemaitre (F.) et Lacassagne (L.). – A new processing chain for detection and tracking of meteors from space. – In *International Meteor Conference*, septembre 2021.
25. Nazlibilek (S.), Karacor (D.), Ercan (T.), Sazli (M. H.), Kalender (O.) et Ege (Y.). – Automatic segmentation, counting, size determination and classification of white blood cells. *Measurement*, vol. 55, 2014, pp. 58–65.
26. Rosenfeld (A.) et Platz (J.). – Sequential operator in digital pictures processing. *Journal of ACM*, vol. 13,4, 1966, pp. 471–494.
27. Salau (J.) et Krieter (J.). – Analysing the space-usage-pattern of a cow herd using video surveillance and automated motion detection. *Biosystems Engineering*, vol. 197, 2020, pp. 122–134.
28. Sochting (M.), Allegretti (S.), Bolelli (F.) et Grana (C.). – A Heuristic-Based Decision Tree for Connected Components Labeling of 3D Volumes. – In *International Conference on Pattern Recognition*, p. 8, 2021.
29. Weng (H.-M.) et Chiu (C.-T.). – Resource efficient hardware implementation for real-time traffic sign recognition. – In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1120–1124. IEEE, 2018.
30. Wu (K.), Otoo (E.) et Shoshani (A.). – Optimizing connected component labeling algorithms. – In Fitzpatrick (J. M.) et Reinhardt (J. M.) (édité par), *Medical Imaging*, p. 1965, San Diego, CA, avril 2005.

A. Appendix



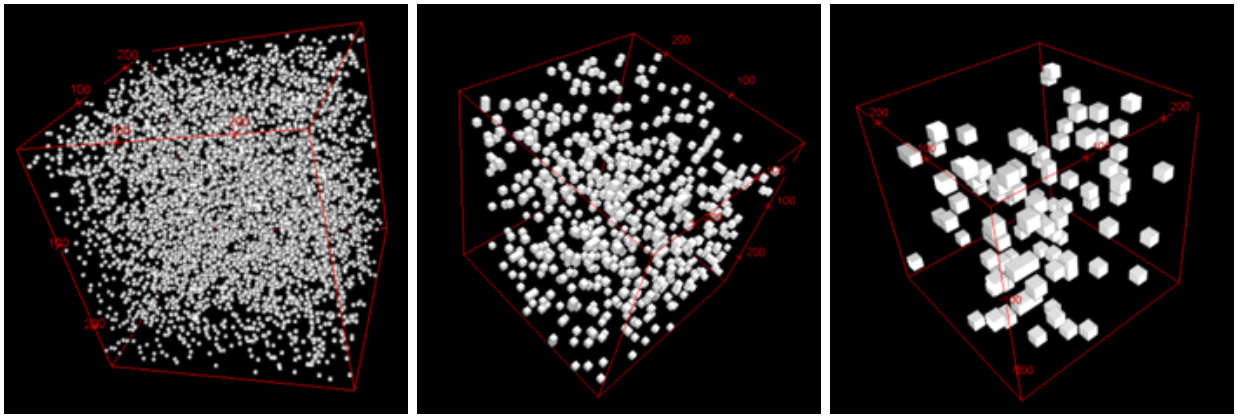
(a) Vue 3D d'une image de *OASIS* (b) Tranche d'une image de *OASIS*

FIGURE 5 – *OASIS* dataset



(a) Vue 3D d'une image de *mitochondria* (b) Tranche d'une image de *mitochondria*

FIGURE 6 – *mitochondria* dataset



(a) $g = 4$ (b) $g = 8$ (c) $g = 16$

FIGURE 7 – Images aléatoires pour différentes granularités (taille des cubes)