Université de Nantes — UFR Sciences et Techniques
Année académique 2018-2019

Représentation de données immersives basée sur la quantification vectorielle algébrique de nuages de points

Stella Pagot - Inès Dussuet

23 mai 2019

Résumé

The ever evolving demand in virtual and augmented reality content leads to the use of a new type of data : 3D point clouds. Because of their significant size, there is a need to compress this type of unstructured data. A previous research team used an approach for static 3D point clouds based on an adaptive Tree-Structure Point-Lattice Vector Quantization (TSPLVQ). In this paper, we address the possibility to apply this method on dynamic 3D cloud points. We propose a way to display the tree structure, and compare two tree structures to visualize their common parts and differences. Results show that successive point cloud structures tend to have nodes and leaves in common, with little difference from one frame to another. This outlook on tree structures imply that compressing a sequence of dynamic 3D point clouds could possibly be faster by keeping the already-coded parts that are common to successive trees.

Table des matières

1 Contexte

La capture et visualisation de contenus 3D est un domaine en rapide évolution suite à l'apparition d'outils de réalité virtuelle ou de réalité augmentée pour en citer quelques uns, dont l'utilisation est de plus en plus demandée [1]. Un acteur majeur autour de cette problématique est le MPEG (Moving Picture Experts Group)[2]. En Janvier 2017, le MPEG à lancé un appel à proposition pour la réalisation d'un standard pour les codecs de compression de nuages de points 3D [3].

L'acquisition de nuages de points 3D se fait entre autre à partir de capteurs LIDAR (Laser Imaging Detection And Ranging) fixes ou mobiles. La capture résulte en un nuage de points 3D, qui se caractérise par un ensemble de points non structurés, sans lien entre eux, avec pour chaque point des attributs spécifiques : sa géométrie, ou sa position dans l'espace représentée par des coordonnées (x,y,z), sa couleur en RGB et sa réflectance [1].

Dû à la densité élevée en points des nuages capturés, pouvant atteindre plusieurs millions pour les outils de captures les plus précis, Il est possible d'obtenir des nuages de points pesant plusieurs gigabyte. Le travail sur les codecs de compression permettrait de réduire l'espace utilisé pour représenter de telles données. La compression de nuages de points 3D voit aussi des domaines d'application dans la "transmission de données en temps réel, la réalité virtuelle ou des applications d'héritage culturel"[3].

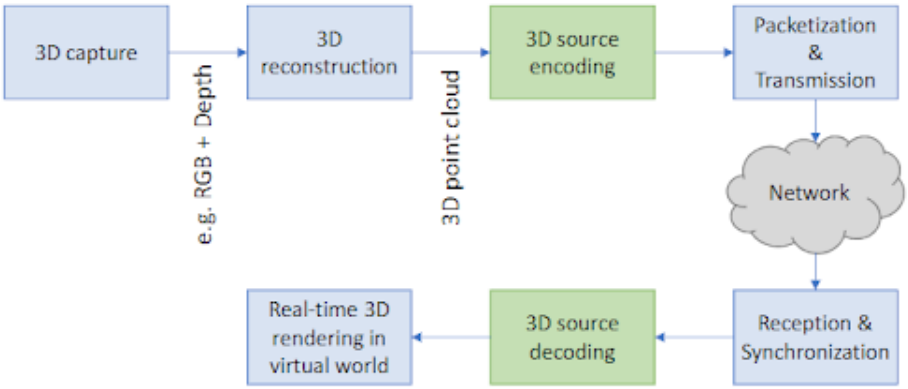

FIGURE 1 – *Processus de capture, encodage et décodage de données 3D [1]*

Le travail réalisé par l'équipe de recherche nous précédant s'est majoritairement axé sur l'encodage et le décodage de la source 3D. Pour encoder des nuages de points il existe plusieurs bases de comparaisons,

— L-PCC : LIDAR Point Cloud Compression. Pour des nuages de points capturés avec un LIDAR. Le sujet de la capture peut être mobile.
— V-PCC : Video-Based Point Cloud Compression. Adapté aux caméras mobiles, des images 2D +t
— S-PCC : Surface Point Cloud Compression. Le S-PCC s'utilise sur des structures statiques. Utilisé pour des images.

Les deux techniques les plus aptes à être utilisées sur des nuages de points 3D sont le L-PCC et S-PCC car correspondent à des codages géométriques, des G-PCC. La TSLVQ (Tree-Structured Lattice Vector Quantization [4]) utilise les deux codecs précédemment mentionnés comme point de référence. Les nuages de points 3D ne possédant pas de structure, il est difficile d'encoder un tel type de données. Le but de la TSLVQ est d'organiser les points capturés du nuage en une structure d'arbre afin de procéder à l'encodage du nuage de points.

2 Etat de l'art

2.1 TSLVQ

La représentation du nuage de points 3D avec un arbre peut se traduire simplement par la construction d'un large cube (la racine) contenant tous les points du nuage puis le découpage de celui-ci en cubes de même taille. Ces mêmes cubes seront ensuite découpés à leur tour selon des critères que nous allons voir par la suite et ainsi de suite jusqu'à une condition d'arrêt établie. Afin de construire l'arbre, la méthode TSLVQ [4] utilise un principe hiérarchique d'ensemble de cube qui se fait grâce à la possibilité d'intégrer un cube tronqué d'une plus petite échelle dans une cellule du prochain cube tronqué d'une échelle plus grande. Un facteur b de mise à l'échelle entre les cubes successifs est donc mis en place.

La construction de l'arbre se fait de manière itérative telle que à chaque boucle une feuille est découpée, elle est choisie selon un critère de débit-distorsion.

L'équipe avec laquelle nous avons travaillé propose une version améliorée de la méthode TSLVQ. En effet, deux façons de partitionner les cellules sont utilisées : le découpage en 2x2x2 cubes ou en 3x3x3 cubes, afin de mieux s'adapter aux différents types de nuage de points et rendre par conséquent plus efficace l'encodage et la compression.

L'algorithme [4] suivi par la méthode proposée par l 'équipe se décline de la manière suivante :

(0) L'initialisation, lors de laquelle l'ensemble des points est normalisé pour rentrer dans la cellule Voronoï initiale (la racine) en les centrant et en les mettant à l'échelle avec le facteur $F = \frac{1}{\sqrt{E_{max}}}$ avec E_{max} la distance euclidienne maximale entre un point et le centre. Les points du nuage sont donc projetés dans un cube Voronoï qui sera la racine de l'arbre.

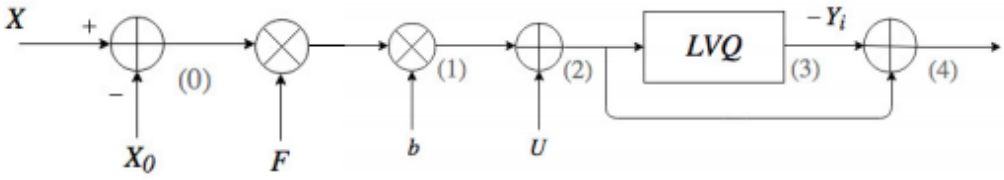

FIGURE 2 – *Schéma de la méthode TSLVQ (initialisation(0) et procédure basique(1 à 4))* [4],[5]

Puis la procédure basique qui sera répétée à chaque découpage d'un cube, pour chaque point du nuage contenu dans le cube choisi :

(1) et (2) On entre un point X contenu dans un cube Voronoï donné, il est mis à l'échelle par le facteur b puis il est déplacé selon le vecteur U . U et b sont déterminés selon le mode de découpage, si c'est du 2x2x2 alors b = 2 et U = (1/2,1/2,1/2) sinon pour du 3x3x3 b =3 et U = (0,0,0).

(3) Un point Y correspondant est calculé selon un algorithme de quantification par une opération d'arrondi [10]

(4) On centre le point Y afin de permettre une autre quantification

A chaque tour de boucle de l'algorithme, le mode de découpage (2x2x2 ou 3x3x3) et le cube à découper est choisi selon un critère débit-distorsion que nous verrons par la suite. La condition d'arrêt choisie dans notre méthode est l'atteinte d'un débit maximal par la construction de l'arbre.

2.2 Méthodes lagrangiennes

La finalité de la compression d'un nuage de point 3D est de minimiser la distorsion avec un coût minimal. A chaque itération de l'algorithme, on applique la méthode TSLVQ sur toutes les feuilles de l'arbre existant. Pour chaque découpage de feuille, on obtient deux mesures : le débit r et la distorsion d. Afin de sélectionner la meilleure feuille à chaque itération de l'algorithme, on retrouve un problème d'optimisation Lagrangienne. Dans notre cas d'étude, l'algorithme BFOS (algorithme Breiman, Friedman, Olshen et Stone)[6] est utilisé dans une approche où l'on cherche a minimiser

le critère de la formule générale. La formule utilisée est adaptée à une échelle locale, donc un noeud, ou l'on cherche à maximiser la baisse de distorsion (Δd) et minimiser l'augmentation du coût (Δr). On obtient donc un critère à maximiser, dans la formule,

$$\lambda = \frac{\Delta d}{\Delta r}$$

Dans le cadre de notre recherche, deux méthodes de partitionnement étaient mis en concurrence, comme vu dans la section précédente. La meilleure feuille est choisie en prenant en compte le obtenu par les deux modes de découpage, et ce à chaque itération. Le noeud possédant le minimal est conservé.

Pour déterminer la fin de l'exécution de l'algorithme, plusieurs conditions d'arrêt sont possible. On peut par exemple déterminer un débit limite maximum, ou encore un nombre de points maximum à conserver. Dans notre cas, le découpage s'arrête quand le débit atteint une valeur précise arbitraire. Une fois le nouveau noeud conservé, on choisit un représentant pour chaque cellule. Plusieurs méthodes existent, mais ici le représentant est le centre de chaque nuage de points.

2.3 Débit distorsion

Dans la section précédente nous avons introduit deux notions, la distorsion et le débit. La distorsion est la mesure d qui caractérise la distance entre deux nuages de points. Cette mesure peut être obtenue avec deux métriques.

La métrique Point-to-Point, $\mathrm{D}_{P2Point}$, correspond à la somme des distances euclidiennes entre les points du nuage original et le représentant dans le nuage de points de la cellule Voronoï associée.[4]

$$D_{P2Point}(X, Y_i) = \sum_{X \in C_i} d(X, Y_i)$$

La métrique Point-to-Plane, $\mathrm{D}_{P2Plane}$, correspond à la distance euclidienne entre les points du nuage initial et les normes du plan représentant associé à chaque point dans la cellule Voronoï associée, obtenu par projection du point initial sur le plan représentant.[4]

$$D_{P2Plane} = \sum_{X \in C_i} ||d(X, Y_i).\vec{N}||$$

Les deux métriques ci dessus sont prises en compte lors de l'application de l'algorithme BFOS, mais ne sont pas mises en concurrence pour le choix d'un noeud.

Le débit correspond à la quantité d'informations nécessaires (le coût en bit) pour encoder l'arbre. Il est lié au codage entropique (plus précisément pour notre méthode, le codage arithmétique). En effet, l'arbre sera encodé sous la forme d'un flux binaire. Lorsqu'un nœud est découpé, 8 ou 27 feuilles sont créés (selon le découpage 2x2x2 ou 3x3x3), le nœud prendra comme chaîne binaire 8 ou 27 fois 0 et apparaîtra avec un 1 au lieu de 0 dans la chaîne binaire de son nœud parent. Les feuilles, elles sont les 0 et le flux binaire suit un ordre pour les enfants d'un nœud. L'arbre sera donc de la forme : 00001000 00000000 pour un arbre avec une racine qui a comme cinquième fils un nœud découpé en 2x2x2 dont les enfants ne sont que des feuilles. Avec toutefois une remarque, la manière de présenter ce flux binaire est en réflexion du fait qu'il y ait deux méthodes de découpage et qu'il y ait donc besoin d'informations pour les distinguer. C'est sur ce flux binaire que le codage arithmétique est appliqué.

3 Notre travail

Notre travail se base sur le travail précédemment effectué par l'équipe de Vincent Ricordel et Amira Filali. L'approche abordée dans les sections précédentes est fonctionnelle pour des nuages de points statiques. Une étape suivante dans leur recherche était d'appliquer cette méthode sur des nuages de points dynamiques.

L'angle de recherche se base sur l'idée que un nuage de point capturé à un instant t et un nuage de point capturé à un instant t+1 sont susceptibles d'avoir une partie commune dans leur structure d'arbre. En effet, par analogie avec une vidéo, un nuage de point représentant un mouvement d'un instant t à un instant t+1 devraient avoir un grand nombre de point commun, tandis que les points au niveau des parties en mouvement devraient se différencier.

FIGURE 3 – *Trois frames consécutives frame0000, frame0001 et frame0002 [7]*

Dans la structure d'arbre étudiée, les nuages de points devraient se caractériser par un "tronc" commun, tandis que des différences apparaîtraient à un niveau de profondeur plus élevé, au niveau des feuilles. Dans la finalité, l'objectif serait, de réduire le temps nécessaire pour compresser une séquence de nuages de points dynamiques. Pour une scène donnée, il s'agirait de calculer un arbre pour chaque état du nuage afin de comparer les arbres obtenus. Il serait possible de voir les points communs et différences entre chaque nuage. Il suffirait ensuite de compresser le nuage de point à un instant t, et de conserver les parties communes. Les parties différentes du nuages de point à l'instant t+1 seraient les parties qu'il reste à coder.

Une première étape était de pouvoir visualiser l'arbre résultant de l'application de l'algorithme appliqué pour compresser le nuage de points initial. Nous avons décidé de représenter l'arbre à l'aide d'un graphe de visualisation. Notre approche s'intègre au sein de l'algorithme de compression pour obtenir une construction du graphe de visualisation incrémentale.

Nous avons créé une structure d'arbre simplifiée qui se base sur la structure d'arbre originale. L'arbre initial contient entre autre une liste de noeuds découpés, ainsi qu'une liste des feuilles contenant de l'information, c'est à dire des points. Chaque noeud est représenté par une structure possédant de nombreux attributs. Avec notre approche, nous avons déterminé que la majeure partie des informations donnée par cette structure d'arbre n'étaient pas nécessaire. C'est pourquoi nous avons décidé de simplifier la structure existante, et d'ajouter les informations manquantes, comme les feuilles vides.

Notre nouvelle structure est un dictionnaire contenant en clé les index dans l'arbre initial des noeuds découpés, et en valeur une liste d'enfants (liste des indexes des enfants ou 0 si c'est une feuille). La liste d'enfant est trié par ordre croissant des représentants des nœuds enfants dans l'espace de la cellule Voronoï correspondant au nœud parent. Lors du découpage d'un nœud dans l'arbre initial, on remplace le 0 dans la liste d'enfants de son parent par l'index puis on ajoute la nouvelle paire index/liste d'enfants. Pour visualiser cet arbre, nous avons donc utilisé une structure de graphe de visualisation adaptée utilisant des outils de python.

Par la suite, nous devions chercher comment comparer deux nuages de points à l'aide de notre structure. Nous nous sommes penchés vers la méthode xor qui applique le ou exclusif sur les flux binaires (représentant la combinaison du découpage d'un nœud) de deux nœuds étant à la même position dans chacun des arbres avec 0 pour indiquer que le nœud est une feuille ou 1 pour indiquer qu'il est découpé.

Nous avons adapté cette méthode de comparaison à notre structure. La méthode pour construire l'arbre initial utilisant deux modes de découpage nous avons dû établir un sens de comparaison. Notre algorithme de comparaison se décrit de la manière suivante : Deux arbres A et B sont donnés en paramètres. On construit un graphe de visualisation à partir de l'arbre A. Soient les racines ra et rb de chacun des deux arbres. On applique les points suivants pour ra et rb :

(1) on regarde la liste des enfants de chacun des deux nœuds donnés na et nb dans les dictionnaires de l'arbre A et l'arbre B.
(2) Si les deux listes ne sont constituées que de feuilles (c'est-à-dire que de 0) ou si leur liste est vide, on s'arrête.
(3) Si la taille des deux listes est la même (c'est-à-dire si les deux noeuds sont découpés tous les deux de la même façon soit en mode 2x2x2 soit en mode 3x3x3) alors :
    (4) Pour chaque paire de noeuds (ea,eb) obtenue en parcourant en parallèle les deux listes d'enfants, on applique les points (5) et (5')
    (5) Si ea et eb sont des nœuds découpés, on applique à nouveau les

points à partir de (1) pour les nœuds ea et eb.

(5') Si seulement ea ou eb est découpé alors on indique sur le graphe de visualisation que le sous-arbre partant de ea est différent.

(3') Si la taille des deux listes n'est pas la même (ce qui veut dire que soit un des deux noeuds est une feuille soit l'un est découpé en 2x2x2 et l'un en 3x3x3) alors on indique sur le graphe de visualisation que le sous-arbre partant de na est différent.

On obtient ainsi le graphe de visualisation de la comparaison, indiquant ce qu'il existe de commun entre les deux arbres et ce qu'il faut changer sur l'arbre B pour passer à l'arbre A, on notera ArbreA ← Arbre B la comparaison.

4 Résultats

Nous avons ici utilisé le fichier longdress [9] pour illustrer le graphe que l'on obtient en résultat. Par souci de clarté, nous avons limité le débit d'arrêt à 2000, les graphes que l'on obtient étant très larges.

FIGURE 4 – *Longdress, nuage d'origine*

FIGURE 5 – *Nuage de point obtenu par l'algorithme de quantification avec un débit d'arrêt de 2000*

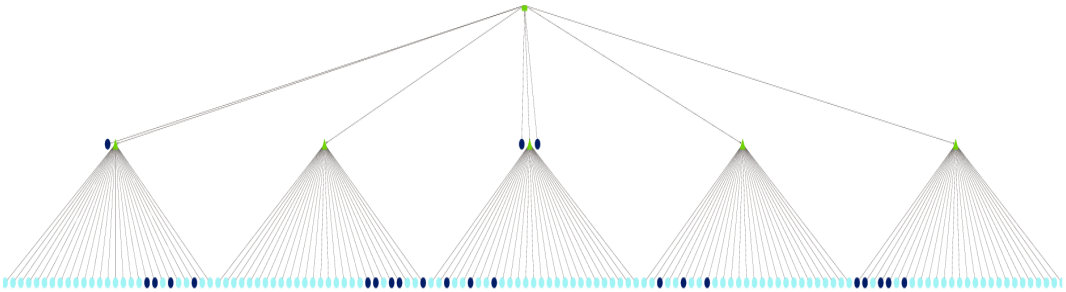

FIGURE 6 – *Arbre de visualisation avec un critère d'arrêt du débit à 2000*

L'état des noeuds et feuilles est indiqué par une forme et une couleur. Les noeuds, en vert, possèdent une forme de carré lors d'un découpage 2x2x2, et d'un triangle avec un découpage 3x3x3. Les feuilles, en bleu, sont représentées par des cercles. Si une feuille contient de l'information, elle est alors colorée en bleu foncé. Les feuilles vides sont en bleu clair.

Comme mentionné précédemment, nous avons décidé de faire des comparaisons d'arbres deux à deux. Les parties communes entre deux arbres sont affichés en noir, tandis que les parties différentes sont colorées. L'exécution du programme nous donne deux images en résultat. La première image possède les points communs entre l'arbre 1 et l'arbre 2, avec les éléments présents uniquement dans l'arbre 1. La deuxième image possède elle aussi les points communs entre l'arbre 1 et l'arbre 2, ainsi que les éléments présents dans l'arbre 2.

Nous avons illustré ce résultat sur les fichiers frame0000, 0001 et 0002.

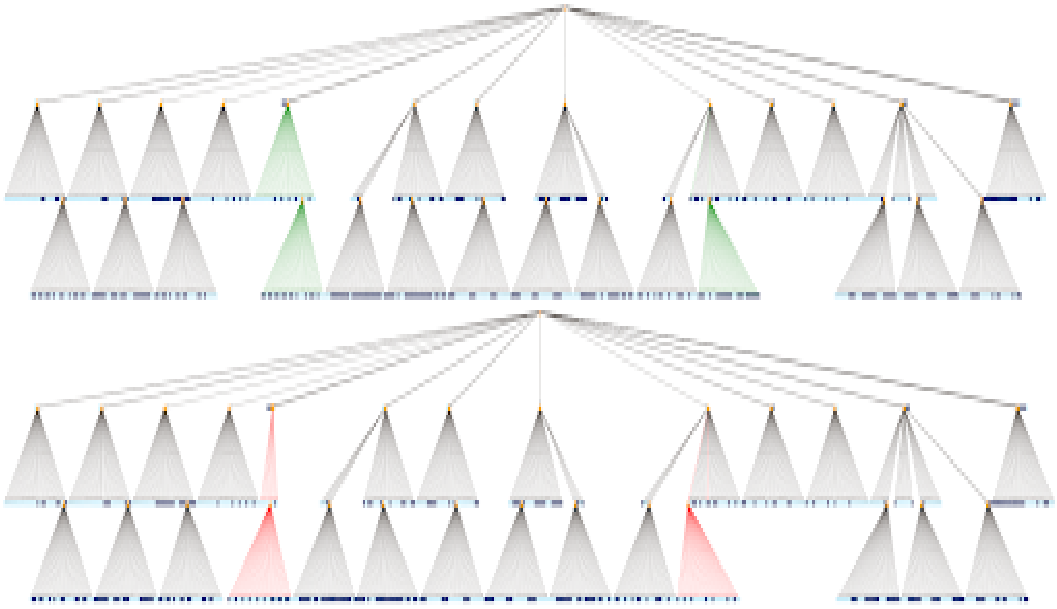

FIGURE 7 – *Vues d'ensemble de la comparaison des arbres issus de frame0000, en vert et frame 0001, en rouge*

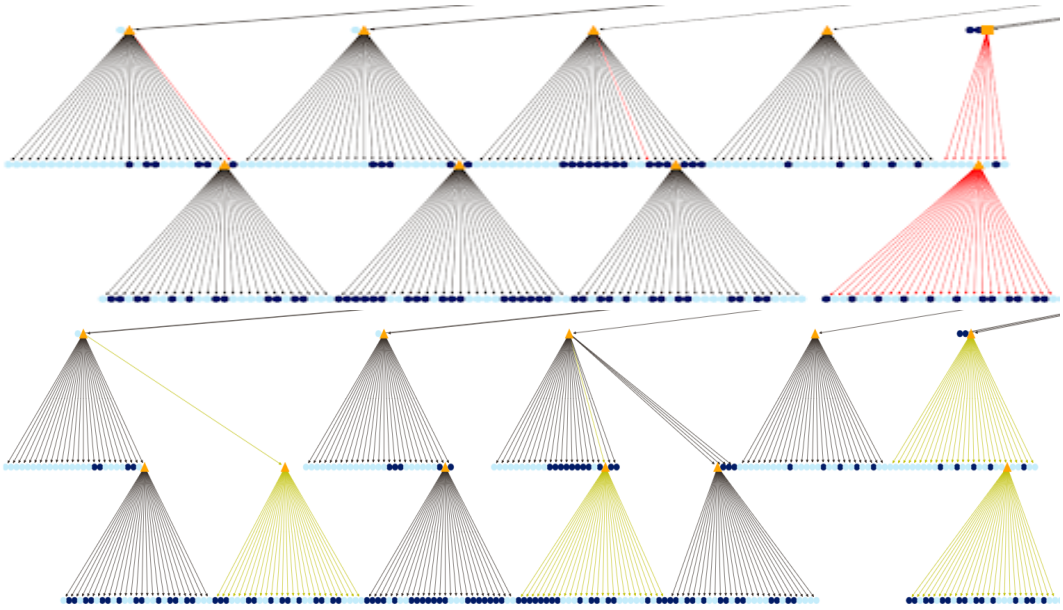

FIGURE 8 – *Extrait de la comparaison des arbres de frame0001, en rouge et frame0002, en jaune, pour les enfants 0 à 8 de la racine*

Nous avons fait ce choix d'avoir deux images plutôt qu'une car un noeud peut avoir deux découpages possibles. Si l'on se trouve dans un cas ou un même noeud est découpé dans l'arbre 1 et l'arbre 2, il aurait fallu trouver un moyen de représenter ce noeud découpé de deux manières ainsi que leurs enfants sur le même graphe. On aurait donc eu une superposition de leurs différences, rendant le graphe peu lisible.

5 Discussion

Notre travail tournant autour du problème de compression de nuage de points 3D dynamique, nous avons proposé une structure d'arbre permettant de mettre à jour les différences entre les nuages de points compressés des différentes images d'une scène dynamique. En construisant une visualisation de ces arbres, nous avons pu voir pour quelques images que l'hypothèse d'un tronc commun était réaliste. Ce qui signifie qu'il y a la possibilité d'optimiser la compression d'une scène en enlevant les éléments redondants. Il faut prendre en compte que nos résultats n'ont été obtenus que sur quelques images, il faudrait étendre les essais notamment sur des nuages de points dynamiques ayant plus d'images. Cette hypothèse de tronc commun est notamment plus probable entre des images successives que entre la première image et la dernière d'un nuage de point représentant un mouvement rapide. Un petit changement dans la géométrie du nuage de points peut entraîner cependant un changement de mode de découpage dès les premiers niveaux de l'arbre de l'image suivante et être similaire par la suite ce qui rend d'autant plus compliquée la tâche de comparaison.

La mise en évidence des modes de découpage et de l'occupation des feuilles sur notre visualisation permet de mieux comprendre comment a été construit l'arbre et donc la géométrie du nuage de points correspondant.

6 Conclusion et Perspectives futures

Ce papier a présenté un outil pour visualiser les structures d'arbres associées aux nuages de points à encoder. Cet outil se base sur une structure d'arbre déjà mise en place, que nous avons du adapter. Nous avons aussi établi un algorithme permettant de comparer deux arbres, dont les points communs et différences peuvent être appréhendés visuellement.

Dans le futur, il pourrait être intéressant d'établir une métrique pour évaluer numériquement la différence entre deux nuages de points. On peut penser à la distance de Hamming, qui est une distance entre deux mots binaires de même taille. Aussi, dans le prolongement de la recherche sur le codage entropique pour des nuages de points statiques, la mise en application du codage sur les nuages de points dynamiques reste à voir. Nous espérons que l'outil présenté dans ce papier facilitera cette tâche.

Bien que nous comparions les images deux à deux dans notre méthode, nous réfléchissons à une manière de généraliser cette comparaison pour toutes les images d'une scène afin de visualiser seulement les parties communes et voir s'il existe un tronc commun pour toute la scène et ainsi améliorer l'efficacité de la compression en réduisant l'espace pris par les informations encodées.

Références

[1] *Sebastian Schwarz, Marius Preda, Vittorio Baroncini, Madhukar Budagavi, Pablo Cesar, Philip A. Chou,Robert A. Cohen, Maja Krivoku ca, S ebastien Lasserre, Zhu Li, Joan Llach, Khaled Mammou, Rufael Mekuria,Ohji Nakagami, Ernestasia Siahaan, Ali Tabatabai, Alexis M. Tourapis, Vladyslav Zakharchenko*, "Emerging MPEG Standards for Point Cloud Compression".

[2] The Moving Picture Experts Group website. https ://mpeg.chiariglione.org/ [consulté le 07/03/19]

[3] *MPEG 3DG*, "Call for Proposals for Point Cloud Compression". INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, Jan. 2017, Geneva, CH.

[4] *Amira Filali, Vincent Ricordel, Nicolas Normand*, "Tree-Structured Point-Lattice Vector Quantization for 3-D Point Cloud Geometry Compression". CORESA, Nov 2018, Poitiers, France.

[5] *Vincent Ricordel, Claud Labit and Moncef Gabbouj*, "Tree-Structured Lattice Vector Quantization for Video Coding". 1997

[6] *E. Riskin*, "Optimal bit allocation via the generalized BFOS algorithm". IEEE Transactions on Information Theory, 1991.

[7] *Microsoft*, "David". Microsoft Voxelized Upper Bodies - A Voxelized Point Cloud Dataset

[8] *Julius Kammerl, Nico Blodow, Radu Bogdan Rusu, Suat Gedikli, Michael Beetz, and Eckehard Steinbach*, "Real-time Compression of Point Cloud Streams". IEEE International Conference on Robotics and Automation, May 2012, RiverCentre, Saint Paul, Minnesota, USA.

[9] *Eugene d'Eon, Bob Harrison, Taos Myers, and Philip A. Chou*, "8i Voxelized Full Bodies - A Voxelized Point Cloud Dataset," ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006, Geneva, January 2017.

[10] *J.H. Conway and N.J.A. Sloane* "Sphere packings lattices and groups".1993.