Ressources naturelles Canada
Natural Resources Canada

# Le GeoHashTree, une structure de données multirésolution pour la gestion des nuages de points

*N. Sabo, A. Beaulieu, D. Bélanger, Y. Belzile et B. Piché*

Géomatique Canada

Note technique 4

2014

Canada

**Géomatique Canada**

**Note technique 4**

# Le GeoHashTree, une structure de données multirésolution pour la gestion des nuages de points

*N. Sabo, A. Beaulieu, D. Bélanger, Y. Belzile et B. Piché*

**2014**

ISSN 1914-4229
N° de catalogue M103-1/4-2014F-PDF
ISBN 978-0-660-21445-0
doi:10.4095/293155

Les bibliothèques de dépôt d'un bout à l'autre du pays ont accès à la présente publication par l'intermédiaire du site Web du Programme des services de dépôt (http://dsp-psd.tpsgc.gc.ca).

On peut télécharger cette publication gratuitement à partir de GEOSCAN (http://geoscan.sst.rncan.gc.ca).

This publication is also available in English.

**Notation bibliographique conseillée**

Sabo, N., Beaulieu, A., Bélanger, D., Belzile, Y. et Piché, B., 2014. Le GeoHashTree, une structure de données multirésolution pour la gestion des nuages de points; Géomatique Canada, Note technique 4, 13 p. doi:10.4095/293155

***Lecture critique***

*J. Brodeur*

***Auteurs***

***N. Sabo (Nouri.Sabo@RNCan-NRCan.gc.ca)***
***A. Beaulieu (Alexandre.Beaulieu@RNCan-NRCan.gc.ca)***
***D. Bélanger (David.Belanger@RNCan-NRCan.gc.ca)***
***Y. Belzile (Yves.Belzile@RNCan-NRCan.gc.ca)***
***B. Piché (Benoit.Piche@RNCan-NRCan.gc.ca)***
*Centre d'information topographique de Sherbrooke*
*2144, rue King Ouest*
*Sherbrooke (Québec)*
*J1J 2E8*

Corrections faites le

# Le GeoHashTree, une structure de données multirésolution pour la gestion des nuages de points

N. Sabo, A. Beaulieu, D. Bélanger, Y. Belzile et B. Piché

Sabo, N., Beaulieu, A., Bélanger, D., Belzile, Y. et Piché, B., 2014. Le GeoHashTree, une structure de données multirésolution pour la gestion des nuages de points; Géomatique Canada, Note technique 4, 13 p. doi:10.4095/293155

**Résumé :** Depuis un certain nombre d'années, le lidar est devenu une des importantes technologies d'acquisition des données altimétriques. Cependant, la gestion des données lidar est très complexe, compte tenu de la quantité phénoménale de données que génère cette technologie. Pour faciliter la gestion des données lidar, cet article propose le GeoHashTree, une structure de données qui permet de gérer différents types de nuages de points, à de multiples résolutions. Le GeoHashTree est une structure hiérarchique qui permet de présenter les données de distribution régulière ou irrégulière sous différents niveaux d'abstraction. En plus de faciliter la gestion des nuages de points, cette structure permet de réduire considérablement l'espace de stockage de données tout en facilitant l'accès à ces données, ainsi que leur manipulation. Cet article présente le GeoHashTree, ainsi qu'un prototype basé sur cette structure.

**Abstract:** Over a number of years, lidar has become one of the major elevation-data acquisition technologies. However, the management of lidar data is extremely complex due to the phenomenal amount of data generated by this technology. To facilitate lidar-data management, this article proposes a GeoHashTree, which is a multiresolution data structure for managing different types of point clouds. The GeoHashTree is a hierarchical structure which can present data of regular or irregular distribution with various levels of abstraction. In addition to facilitating the management of point clouds, this structure reduces data storage space considerably, while also facilitating data access and handling. This article introduces the GeoHashTree and describes a prototype based on it.

## INTRODUCTION

La popularité sans cesse croissante des données altimétriques et leur utilisation dans plusieurs domaines montrent la nécessité d'une bonne connaissance de la topographie. Dans plusieurs pays, les données altimétriques sont parmi les données les plus demandées. C'est par exemple le cas du Canada, où les données altimétriques représentent 73 % de tous les téléchargements du portail GéoBase (http://www.geobase.ca/). Cette popularité grandissante des données altimétriques s'explique en grande partie par le nombre impressionnant d'applications requérant ce type de données. De nos jours, plusieurs applications utilisées pour la gestion des risques d'inondation, en télécommunication, en planification urbaine et régionale, et dans divers autres domaines ne peuvent se passer des données altimétriques. On peut aussi entrevoir des besoins plus importants dans le futur; par exemple, les systèmes de transport intelligent et d'aide à la conduite automobile intégreront bientôt les élévations sur les routes afin de réduire la consommation et le rejet des gaz à effet de serre de 4 à 12 % (L. Sugarbaker, G. Snyder et D. Maune, 2012, présentation intitulée « Results of the National Enhanced Elevation Assessment (NEEA) », donnée au 12th International LiDAR Mapping Forum, Denver (Colorado), 23-25 janvier 2012).

Dans plusieurs organisations, des données altimétriques de différentes sources, différents niveaux de précision et différentes époques cohabitent. On retrouve des données captées il y a plusieurs décennies à l'aide des méthodes conventionnelles et des données plus récentes et précises captées grâce aux technologies modernes comme le lidar. Dans de telles situations, même si, à première vue, le réflexe est de se demander pourquoi ne pas garder seulement les données les plus récentes et les plus précises, la réalité est tout autre. En effet, dans plusieurs régions et pays du monde, la couverture en données altimétriques est souvent une mosaïque de données de résolutions et qualités variées, captées à différentes époques. En outre, même lorsque la couverture est complète et homogène, le besoin de différents modèles altimétriques (p. ex. modèles de surface, de terrain, de couvert forestier, etc.) et différentes résolutions est toujours présent, compte tenu de la diversité des applications. En effet, d'une part, certaines applications n'ont pas besoin de données de très haute résolution et, d'autre part, les données historiques sont en demande croissante, surtout dans le domaine des changements climatiques. Cette cohabitation de données de différentes sources entraîne de gros défis d'intégration, surtout quand on sait que dans la plupart des applications, ces données doivent être intégrées avec des données autres que l'élévation (p. ex. couverture du sol).

En fait, les besoins de données altimétriques ne cessent de grandir et de se spécialiser, et les technologies d'acquisition pour ce type de données sont de plus en plus nombreuses, performantes et accessibles. Si le développement technologique de ces dernières années a engendré l'émergence de nouveaux capteurs capables d'acquérir des données de très haute précision et résolution, il n'en demeure pas moins que les données générées sont de plus en plus volumineuses et complexes, au point de poser de sérieux problèmes en termes de gestion et d'exploitation. Par exemple, les capteurs lidar actuels sont capables d'acquérir jusqu'à un million de points en une seule seconde avec des précisions verticale et horizontale centimétriques. Avec un million de points par seconde, on peut imaginer le nombre de données qui peuvent être collectées lors des grandes campagnes d'acquisition (p. ex. à l'échelle d'un pays). La gestion et l'exploitation d'une telle quantité de données nécessite des outils autres que ceux utilisés traditionnellement pour la gestion des données vectorielles ou matricielles.

Afin de permettre la gestion, l'exploitation et l'intégration de données altimétriques de différents types et résolutions, nous proposons dans cet article une nouvelle structure de données appelée GeoHashTree (GHT). Le GHT est une structure hiérarchique multirésolution qui permet de gérer les données de distribution régulière ou irrégulière (comme les données lidar) sous différents niveaux d'abstraction. En plus, compte tenu de sa capacité à indexer toutes les données, cette structure s'intègre facilement dans les systèmes de gestion des bases de données.

Cet article fournit une introduction aux données lidar et à leur gestion, suivie par une introduction à la nouvelle structure et la présentation d'un prototype fonctionnel basé sur cette structure. Les résultats des tests réalisés sont ensuite présentés et discutés avant les conclusions finales.

## LES DONNÉES LIDAR ET LEUR GESTION

### Les données lidar

Dans les dernières décennies, le lidar (***li**ght **d**etection **a**nd **r**anging*, ou détection et télémétrie par la lumière) est l'une des technologies qui a radicalement révolutionné le mode d'acquisition des données altimétriques. Bien que datant des années 60, le lidar est une technologie qui se développe à une vitesse fulgurante. Les types d'utilisation des données lidar sont de plus en plus nombreux et les capteurs deviennent de plus en plus performants, amenant continuellement de nouveaux défis en termes de gestion et d'exploitation de ces données. Contrairement à la photogrammétrie traditionnelle où l'élévation doit être extraite à partir des modèles stéréoscopiques, le lidar fournit directement l'élévation, ce qui résulte en un gain de temps. En fait, le lidar offre plus que l'élévation, car certains attributs sont utilisés à d'autres fins. Par exemple, grâce à la lidargrammétrie, certains éléments topographiques comme les infrastructures peuvent être extraits en utilisant des paires pseudostéréo créées à partir des images d'intensité et de l'élévation.

Si, d'une part, cette technologie permet d'acquérir des données de très haute précision, il n'en demeure pas moins que le stockage, la gestion et l'exploitation de ces données posent de sérieux défis à cause de leur distribution irrégulière, de leur densité et de la quantité d'information qu'elles contiennent. Contrairement aux données matricielles, les données lidar sont des données de distribution irrégulière sous forme de nuages de points qui ne suivent aucune organisation logique, ce qui rend impossible la création d'une fonction mathématique à l'image de la matrice, capable de prédire les coordonnées des points. D'autre part, le lidar produit une quantité phénoménale de points dont chacun comporte pas moins d'une douzaine d'attributs de formats souvent différents. Par exemple, le temps nécessaire pour réaliser la collecte d'un million de points est passé de plus de 15 ans (en utilisant des techniques d'arpentage) à quelques secondes en utilisant la technologie lidar (S. Daniel, 2011, notes du cours « LiDAR terrestre et aéroporté : principes et applications » donné à l'Université Laval, Québec (Québec), le 21 novembre 2011). De plus, le lidar est un mélange de plusieurs modèles (p. ex. modèles de surface et de terrain) à cause de la multiplicité des retours. Bien que le plus souvent deux principaux modèles soient gérés, à savoir les modèles de surface et de terrain, il faut considérer que chacun des retours pourrait représenter un modèle pouvant être utilisé pour d'autres applications. Comme mentionné ci-dessus, le volume de données générées par ces technologies, combiné à la complexité de ces types de données, rend leur gestion et leur exploitation très ardue.

Depuis longtemps, les données lidar sont stockées et gérées grâce à une approche de gestion par fichiers. Mais depuis un certain nombre d'années, on voit l'émergence d'une nouvelle approche, qui permet de les gérer dans un système de gestion de base de données. Dans les prochaines sections, nous présenterons les deux principales approches de gestion des données lidar.

## La gestion des données lidar par fichiers

Comme on l'a expliqué plus haut, les données lidar sont très complexes à cause de la multitude d'attributs qu'elles comportent et du grand volume de données générées par cette technologie. Cette complexité se traduit par une difficulté à gérer et manipuler ces données. Pour cette raison, les données lidar sont généralement gérées différemment selon les utilisateurs. Ainsi, les compagnies qui réalisent le levé gèrent les données sous forme de nuages de points. Quant aux utilisateurs finaux, ils se contentent souvent des modèles altimétriques numériques qui sont une interpolation des nuages de points. Les deux types de données, à savoir les nuages de points et les modèles numériques, sont généralement gérés sous forme de fichiers de différents formats.

Pour stocker les nuages de points, les formats ASCII ou LAS sont généralement utilisés (ASPRS, 2012; Huber, 2011). Le format ASCII a l'avantage d'être lisible par l'humain et de pouvoir être lu et modifié par n'importe quel éditeur de texte, et il présente moins de problèmes de compatibilité entre plates-formes. Par contre, ce format est très volumineux et non-performant lors de la manipulation, ce qui le rend parfois inutilisable. Compte tenu du volume astronomique des nuages de points, l'utilisation du format ASCII devient moins attrayante pour la gestion d'une grande quantité de données. C'est pourquoi les formats binaires comme le LAS ont fait leur apparition.

Depuis un certain nombre d'années, le format LAS est devenu le standard de facto pour stocker les nuages de points. Il s'agit d'un format binaire qui utilise un ensemble prédéfini d'attributs de taille fixe. Ce format est performant pour l'échange de données; malheureusement, en format LAS original, les points ne sont pas ordonnés, ce qui rend l'accès arbitraire aux données impossible (Graham, 2009). Or, l'accès arbitraire aux données est primordial lors de l'exploitation des données altimétriques (p. ex. pour trouver facilement l'élévation d'un point). Le format LAS représente donc un compromis entre l'efficacité lors de la transmission des nuages de points et la flexibilité (Graham, 2009). Ainsi, pour bien exploiter les nuages de points par des requêtes spatiales et attributives, ils doivent être indexés afin de faciliter un accès arbitraire.

Même si certains éditeurs de logiciels proposent leur propre système d'indexation, il n'en demeure pas moins que cette approche pose de sérieux problèmes. D'une part, l'indexation de milliards de points pourrait nécessiter un espace de stockage important, et d'autre part, l'index doit être mis à jour à chaque fois qu'un changement est apporté à la donnée. Pour ce qui est du format LAS, la spécification actuelle ne permet pas d'ajouter de nouveaux attributs à un fichier existant, car le format possède une structure attributive standardisée, à taille fixe, et exclusivement destinée à la gestion du lidar, ce qui le rend moins flexible, voire inutilisable, en termes d'intégration des données de types et de sources différents.

De plus, le format LAS ne possède aucun mécanisme permettant de généraliser les données. Cette capacité de généraliser les données lidar est primordiale lors de la visualisation. Sans cette capacité, toutes les données doivent être affichées quelle que soit l'échelle de visualisation, ce qui est inimaginable pour une grande étendue spatiale (p. ex. à l'échelle d'un pays ou une région), compte tenu de la quantité de points à afficher. Pour toutes ces raisons, bien que performant pour les opérations de production et pour son niveau de compression, le format LAS n'est pas bien adapté au processus d'analyse et d'exploitation des données dans le contexte où on doit pouvoir accéder arbitrairement aux données, les intégrer entre elles et les mettre à jour.

La difficulté de gérer et d'utiliser les nuages de points dans les formats ASCII ou LAS a souvent poussé certaines organisations à transformer les points lidar en données de distribution régulière (p. ex. matricielles) sous forme de modèles numériques afin de faciliter leur manipulation. Même si les données régulières ont leur place dans la gestion

des données altimétriques, il reste qu'une telle transformation est irréversible et ne permet pas de garder efficacement toute la richesse des données lidar, à savoir la précision et la diversité des attributs. En effet, les modèles numériques prennent seulement en compte l'élévation en laissant de côté tous les autres attributs du lidar, malgré leur utilité dans d'autres applications (p. ex. le RGB pour la visualisation ou l'intensité pour l'extraction des éléments topographiques).

En somme, les approches actuelles de stockage des données lidar par fichiers présentent de sérieuses limitations en termes de gestion, d'intégration et d'exploitation de données de différents types. Avec une approche par fichiers, il serait très difficile de mettre en place un système efficace pour gérer des données couvrant un grand territoire, car d'une part, beaucoup de systèmes d'exploitation ont une limitation en termes de la taille des fichiers, et d'autre part, partitionner les données dans plusieurs fichiers nécessiterait la mise en place d'un système qui permettrait de les indexer, d'assurer la continuité et l'intégrité des données, d'éviter les superpositions, d'assurer l'accès multiple et concurrent, etc.

Afin de faciliter la gestion et l'exploitation des nuages de points et leur intégration avec d'autres types de données, une des solutions serait de les stocker dans un système de gestion de base de données. En effet, les bases de données modernes intègrent déjà les principaux types de données spatiales existantes (données vectorielles et matricielles), ce qui faciliterait l'intégration et l'interaction des nuages de points avec ces types.

## Gestion des données lidar dans un système de gestion de base de données

Depuis quelques années, beaucoup d'efforts ont été déployés pour faciliter l'intégration des nuages de points dans une base de données, à l'image des autres types de données existantes comme les données vectorielles et matricielles, qui sont actuellement gérées dans différentes bases de données (Nandigam et al., 2010; Arias Prado, 2011; Ott, 2012). La gestion des nuages de points dans une base de données offre plusieurs atouts : 1) profiter des avantages qu'offre une base de données en termes de sécurité, d'accès concurrent, de gestion des utilisateurs, d'évolutivité, de gestion des mises à jour, d'accès rapide grâce aux systèmes performants d'indexation et de partitionnement, d'informatique en nuage, et de contrôle de versions; 2) faciliter l'utilisation grâce au langage SQL; 3) faciliter l'interaction avec d'autres types de données déjà inclus dans les bases de données (p. ex. données vectorielles et matricielles); 4) permettre une intégration harmonieuse entre des ensembles de données disparates tout en donnant accès à un riche éventail d'outils d'exploitation (Graham, 2009). La gestion des nuages de points par fichiers est viable lorsqu'il s'agit de projets qui couvrent un espace limité. Par contre, dans le cas d'une couverture d'envergure à une échelle régionale ou nationale, comme par exemple le cas du territoire canadien avec plus de 10 000 000 km$^2$, l'utilisation d'une base de données est plus appropriée, compte tenu de la flexibilité et des avantages mentionnés plus haut. De plus, les bases de données modernes fournissent déjà l'infrastructure de base pour faciliter une telle réalisation (p. ex. mécanisme d'extension).

Pour stocker les nuages de points dans une base de données, trois principales approches peuvent être envisagées : l'approche par points, l'approche multipoint et l'approche par tuiles.

L'approche par points consiste à stocker chaque point sous forme vectorielle, soit un enregistrement par point, et de les indexer. Chaque attribut du point constitue un champ distinct. Une telle solution est très facile à mettre en œuvre puisque le type de point et les outils capables de le manipuler existent dans la majorité des systèmes de gestion des bases de données. Par contre, une telle approche présente de sérieuses restrictions. En effet, l'indexation de milliards de points nécessite un espace de stockage conséquent, et la mise à jour de l'index suite à des changements est fastidieuse.

L'approche multipoint consiste à stocker un ensemble de points par enregistrement (sous forme de grand objet binaire, ou BLOB (*Binary Large OBject*)), au lieu de stocker chaque point individuellement. Dans une telle approche, les attributs et les géométries sont stockés dans des champs différents. Il y a donc un champ pour stocker les géométries de l'ensemble des points du bloc et autant de champs que d'attributs. Une telle approche peut être vue comme une optimisation de l'approche par points, car elle nécessite moins d'enregistrements et est par conséquent plus facile à indexer, ce qui résulte en un gain de performance et une amélioration de l'espace de stockage. Par contre, une telle approche nécessite l'ajout d'un champ « étendue spatiale » à chaque bloc. Ce champ permet d'indexer spatialement les blocs afin de faciliter les requêtes spatiales. De plus, le fait que les attributs soient dans des blocs et dissociés de la géométrie fait en sorte que certaines opérations peuvent être plus difficiles, comme par exemple une requête spatiale combinée à une requête attributive. Comme les points à l'intérieur d'un bloc ne sont pas indexés, l'accès arbitraire aux points à l'intérieur d'un bloc est impossible, ce qui peut poser des problèmes de performance dans le cas des blocs de grande taille.

L'approche par tuiles permet de subdiviser la donnée en tuiles de moindre taille et de stocker chaque tuile sous forme de grand objet binaire (*BLOB*). Contrairement à l'approche précédente, les attributs et la géométrie forment un tout. Les tuiles peuvent être stockées en utilisant en partie ou en totalité une structure ou un format existant (p. ex. LAS) afin d'englober les points et leurs attributs. Pour faciliter l'accès aux données, l'étendue de chaque tuile peut être indexée spatialement. Une telle approche offre plus d'avantages que les deux précédentes en termes de stockage; par contre, elle est tributaire de toutes les qualités et les défauts du format de la tuile. Par exemple,

dans le cas d'un format LAS original, bien que les tuiles puissent être indexées, les points à l'intérieur des tuiles ne le seront pas. Ainsi, pour faciliter l'accès aux données, la taille des tuiles doit être ramenée au strict minimum avec comme conséquence l'augmentation du nombre de tuiles, ce qui nous ramène au problème d'indexation.

Techniquement, certains systèmes de gestion des bases de données actuels offrent des mécanismes qui permettent d'ajouter de nouveaux types avec lesquels on peut interagir directement. Une bonne approche de stockage de données de type « nuages de points » dans une base de données doit prendre en compte plusieurs facteurs :

- permettre une mise à jour facile des données existantes;
- permettre l'intégration de données disparates (différentes sources, différentes résolutions, différents modèles, etc.);
- permettre un accès arbitraire aux données par des requêtes spatiales ou attributives;
- faciliter l'analyse et l'exploitation sans induire un coût excessif en termes de stockage et de performance;
- permettre de présenter les données à plusieurs niveaux d'abstraction (plusieurs résolutions). Cette caractéristique est très importante lors de la visualisation des données très volumineuses comme le lidar;
- faciliter la conversion d'un type vers un autre (p. ex. transformer les nuages de points en données matricielles et vice-versa).

La gestion par fichiers est une approche qui peut accommoder la situation où l'étendue des données est très restreinte et les interactions sont limitées. Pour une gestion de données couvrant une grande étendue où les interactions et les contraintes d'exploitation (p. ex. multiples accès) sont élevées, une approche de gestion par base de données est plus appropriée. Par contre, compte tenu de la nature des données lidar et du besoin de les intégrer avec d'autres types de données (parfois de moindre résolution et précision), les approches actuelles de gestion par base de données sont limitées (problématique d'indexation à cause du nombre élevé de points, manque de structure de données facilitant l'intégration des données, etc.), d'où la nécessité de développer une structure de données plus adaptée.

Pour faciliter la gestion et l'exploitation des nuages de points de différents types et résolutions (y compris les données lidar), nous avons développé une nouvelle structure appelée GeoHashTree.

## LA STRUCTURE GEOHASHTREE (GHT)

Le GeoHashTree (GHT) est une structure de données générique qui permet de stocker des nuages de points, d'y accéder et de les manipuler. Elle permet d'intégrer à la fois des données de différents types (données de distribution irrégulière, comme les nuages de points, et régulière, comme les données matricielles) et de différentes résolutions dans une même structure, et permet des interactions entre ces données. Le GHT est basé sur le Geohash (Wikipedia, 2012), un système global de géocodage du domaine public reposant sur une fonction de hachage qui subdivise la surface de la terre en une grille hiérarchique. Il permet d'encoder les coordonnées (longitude, latitude) en une chaîne de caractères. Le GHT fournit donc une structure de données géospatiales à laquelle on peut rattacher un nombre illimité d'attributs provenant de plusieurs types de données de résolutions différentes, sans dupliquer les coordonnées de chaque source. Par exemple, des données lidar peuvent être intégrées avec d'autres données altimétriques de résolution moindre. Ceci peut se faire sans ajouter de charge supplémentaire indue, car bien que les deux types de données soient différents, seules les coordonnées de la donnée de plus haute résolution sont retenues sous forme de Geohash, alors que pour la donnée avec la plus faible résolution, seuls les attributs sont utilisés. Donc, quels que soient les types de données combinés, une seule structure de données géospatiales sous forme de Geohash est utilisée, ce qui résulte en un gain énorme en terme de stockage lorsque plusieurs types de données sont intégrés. Les Geohash sont générés seulement là où la donnée existe. De plus, puisque le géocodage sous forme de Geohash est aussi un index spatial, aucun index spatial sur les données n'est nécessaire, quoiqu'un accès arbitraire à tous les points soit permis.

Un des grands avantages du GHT est sa flexibilité. Il permet de transformer des nuages de points, ou toutes autres données de type XY-attributs, en une structure hiérarchique où chaque point est indexé grâce à la structure en arbre du GHT et présenté sous une forme multiniveau. Le nombre de niveaux varie en fonction de la densité de l'information et de la précision des coordonnées à stocker. Ainsi, même les données de distribution irrégulière comme le lidar sont représentées sous forme de pyramide, à l'image des données matricielles. Une telle représentation pyramidale de la donnée combinée à l'indexation de tous les points d'un GHT permet de faciliter l'interpolation des nuages de points afin de les transformer en données raster. La figure 1 schématise une représentation multiniveau d'un seul attribut à résolution variable dans un GHT (chaque attribut d'un GHT peut être multirésolution et représenté sous forme multiniveau). En général, pour transformer des points en matriciel, trois principales étapes sont nécessaires : 1) la création de la grille; 2) la sélection des points à l'intérieur de chaque cellule; et 3) la généralisation des données sélectionnées à l'intérieur de chaque cellule en appliquant une fonction mathématique

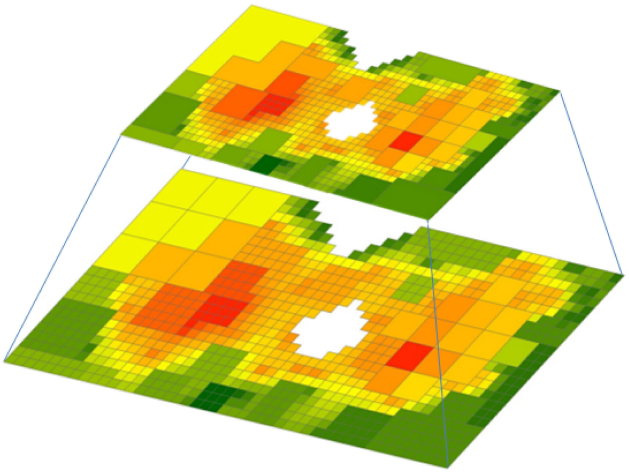

**Figure 1.** La représentation multiniveau d'un attribut à résolution variable dans un GeoHashTree.

(p. ex. pondération inverse à la distance). Or, dans un GHT, la structure elle-même est une grille dont on connaît tous les points à l'intérieur de chaque cellule. De plus, à chaque nœud d'un GHT, plusieurs statistiques liées à chaque attribut comme les valeurs minimum, maximum et moyenne sont présentes. Or, ces statistiques sont souvent les mêmes que celles utilisées pour généraliser les valeurs d'une cellule lors de l'interpolation des données lidar.

D'autre part, cette même structure pyramidale des GHT permettrait de faciliter la visualisation des nuages de points. En effet, compte tenu du nombre phénoménal de points dans les données lidar, la visualisation de tous les points à toutes les échelles n'est pas envisageable lorsqu'il s'agit d'un grand territoire. Grâce aux GHT, on peut entrevoir une visualisation des nuages de points où à chaque échelle, on afficherait le niveau de la structure qui est le plus approprié. Ainsi, à des échelles plus petites, des niveaux de données généralisés (p. ex. minimum, maximum ou moyenne) seront présentés et, au fur et à mesure que l'utilisateur fera un zoom avant dans la donnée, des niveaux plus détaillés seront affichés.

Le stockage des données dans une base de données sous forme de GHT est réalisé en quatre principales étapes : l'encodage des coordonnées, la création de l'arbre, son optimisation et son stockage dans une base de données.

## Encodage des coordonnées en Geohash

La première étape de création d'un GHT commence par l'encodage des coordonnées géographiques (longitude, latitude) des points en Geohash. Pour créer le Geohash d'un point (p. ex. –73,5, 45,4), on subdivise itérativement l'espace et on alloue des bits selon le quadrant dans lequel se situe le point. La subdivision commence à partir des coordonnées planétaires ([–180, 180] et [–90, 90]). Par exemple, pour notre point, le résultat de la première subdivision sera 01, car il se situe dans le quadrant gauche-supérieur (voir la figure 2). Le résultat de l'encodage de notre point sous forme binaire sera 011100001000101010110010111. Par la suite, le résultat binaire est converti en nombre décimal qui, à son tour, sera converti en caractères en utilisant une table de conversion à base 32. Ainsi, le résultat final sera f25dr.

Le Geohash est un système global qui utilise des précisions arbitraires. Le nombre de caractères détermine la précision des coordonnées, et l'élimination graduelle des caractères à partir de la fin de la chaîne de caractères permet de diminuer sa précision (p. ex. 6gkzwgjzn820 pour –25,382708, –49,265506 et 6gkzwgjz pour –25,383, –49,266). La précision d'un tel système d'encodage peut dépasser le femtomètre ($10^{-15}$ m). Les données lidar peuvent être amplement représentées avec un Geohash d'une résolution de 15 caractères, ce qui correspond à une précision métrique d'environ $10^{-7}$ m.

Dans un système basé sur le Geohash, en général, plus les préfixes des Geohash de deux endroits se ressemblent, plus ils sont proches spatialement; par exemple, 6gkzwgjz (–25,383, –49,266) et 6gkzmg1w (–25,427, –49,315). Cette propriété permet d'utiliser le Geohash lui-même comme un système d'indexation de données spatiales (tri simple). Par contre, cette propriété n'est pas toujours respectée (deux points spatialement rapprochés, situés de part et d'autre d'une ligne de subdivision lors du hachage, peuvent avoir des Geohash différents), d'où la nécessité de mettre en place des stratégies basées sur le voisinage pour aider à la sélection des Geohash. Ainsi, la sélection précise doit toujours se faire sur neuf Geohash : le Geohash en question et les huit autres qui l'entourent. Cette stratégie est facile à mettre en œuvre, surtout quand on sait que le tri des Geohash en ordre alphabétique suit un tracé en Z.

Il existe d'autres systèmes d'encodage semblables, comme le HHCode (*Helical Hyperspatial Code*) développé par le Service Hydrographique Canadien, qui permet d'encoder les données en format binaire (Varma et al., 1990). Le choix du Geohash a été guidé par le fait qu'il est dans le domaine public, qu'il est déjà employé dans plusieurs bases de données (p. ex. PostgreSQL, MySQL), et qu'il existe des librairies Open Source dans différents langages (C, Java, Python, JavaScript, etc.). Bien que le Geohash soit un système dont les coordonnées elles-mêmes peuvent être utilisées comme index, il présente plusieurs inconvénients quand il est utilisé ainsi, comme le coût élevé en stockage et un manque de performance. Pour minimiser tous ces inconvénients et apporter d'autres avantages comme l'aspect multiniveau ou la capacité d'intégrer plusieurs attributs dans une structure de données multisource, nous avons mis en place le GeoHashTree (GHT) à partir du Geohash.

## Construction de l'arbre GeoHashTree

Le GHT est un arbre inversé dans lequel tous les enfants d'un nœud partagent le même préfixe de Geohash et les feuilles comportent une liste d'attributs (fig. 3). Ainsi, les

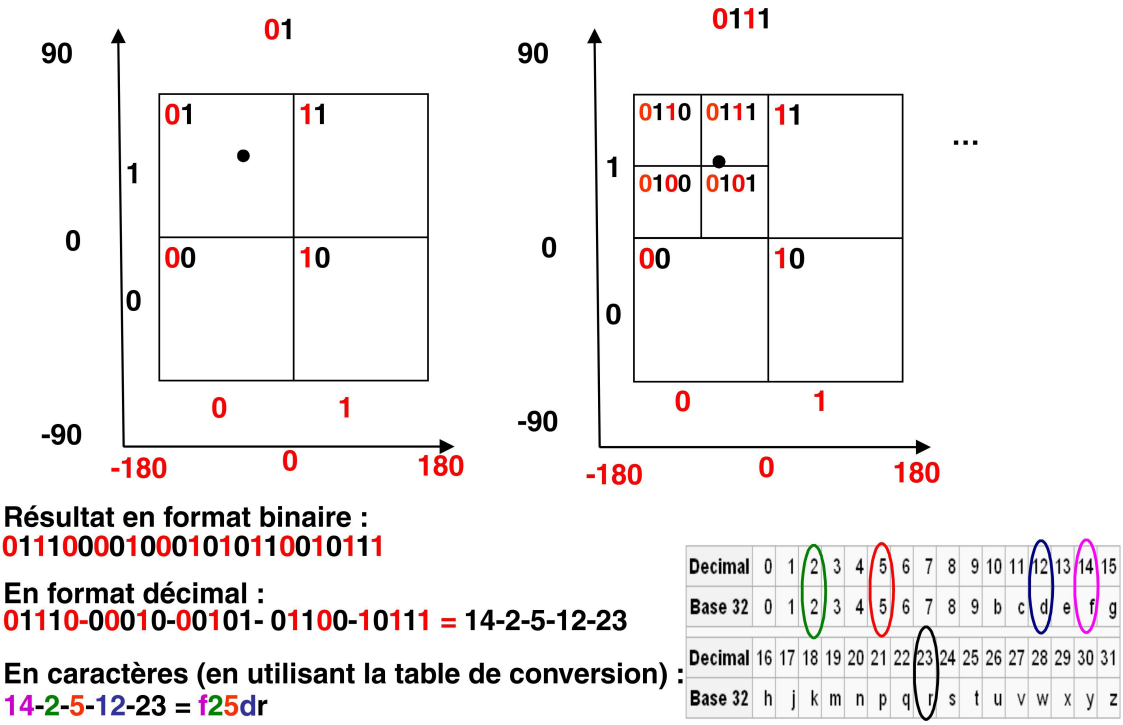


**Figure 2.** Encodage Geohash.

| Geohash | Z | C | T | A | I | N | R | P | E | D | U | r | g | b |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| f2j 2prt 49pjz | 251.15 | 2 | 78494.953266 | 12 | 20 | 2 | 2 | 1039 | 0 | 1 | 0 | 0 | 0 | 0 |
| f2j 2prt 031tc | 251.78 | 1 | 78494.953293 | 12 | 253 | 1 | 1 | 1039 | 0 | 1 | 0 | 0 | 0 | 0 |
| f2j 8025 pr1p2 | 255.98 | 1 | 78494.953358 | 13 | 255 | 1 | 1 | 1039 | 0 | 0 | 0 | 0 | 0 | 0 |
| f2j 8025 1pnz6 | 255.31 | 1 | 78494.953463 | 13 | 76 | 1 | 1 | 1039 | 0 | 0 | 0 | 0 | 0 | 0 |

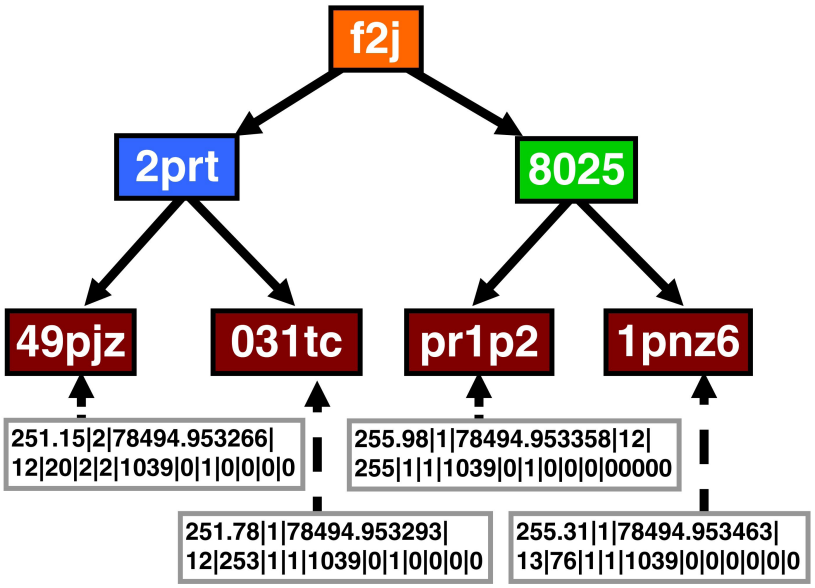


**Figure 3.** Structuration des GeoHashTree.

points qui se rapprochent spatialement se partageront le même parent. Le GHT est créé à partir des Geohash générés à l'étape précédente. À chaque nœud, des statistiques comme le minimum, la moyenne et le maximum de certains attributs pertinents sont attachés, au besoin. La structure en arbre complémentée par ces statistiques confère au GHT un caractère multirésolution (comparable aux pyramides des données matricielles). Lorsqu'un nouveau point est inséré dans la structure, une comparaison du Geohash du nouveau point par rapport aux Geohash des points existants est réalisée. Selon le résultat de cette comparaison (la proximité), le nouveau point est inséré au bon endroit de la structure. Dans un GHT, chaque attribut possède un nom et un type, ce qui permet un nombre quasi illimité d'attributs pour chaque point. Par exemple, on peut stocker plusieurs élévations qui proviennent de différentes sources sur un même point (p. ex. lidar, SRTM (*Shuttle Radar Topography Mission*), points cotés, etc.) ou même insérer des données autres que l'élévation (p. ex. pluviométrie, imagerie multibande) dans une même structure.

Lorsqu'un point de résolution moindre est injecté dans un GHT, le point s'insère au niveau de l'arbre qui correspond à sa résolution (selon la longueur de son Geohash), car chaque niveau d'un GHT correspond à une résolution donnée. La corrélation entre la longueur d'un Geohash et la résolution est due au fait que plus le Geohash est long, plus grand est le nombre de subdivisions réalisées pour sa création. Donc, le choix de la longueur du Geohash d'un point détermine la résolution avec laquelle il est stocké. Par exemple, un Geohash de longueur 15 pour un point situé à l'équateur représente une résolution d'environ 1 mm, tandis que pour une longueur de 10 caractères, la résolution est d'environ 1 m.

## Optimisation du GeoHashTree

Bien que l'étape précédente apporte un certain degré d'optimisation à cause de la structure en arbre qui permet de partager partiellement les préfixes des Geohash, l'espace de stockage reste encore élevé, d'où la nécessité d'optimiser d'avantage. Pour cela, les attributs de chaque point sont restructurés. Fraîchement créée, chaque feuille d'un GHT comporte tous les attributs du point; or, il est évident qu'à l'intérieur d'un GHT certains attributs sont récurrents et d'autres sont stockés dans des types de taille déraisonnable (p. ex. 123,5 stocké en double précision). Ces deux situations sont exploitées lors de l'optimisation. Selon les attributs, deux types d'optimisation sont appliqués (les deux méthodes ne sont pas exclusives) : migration d'attributs et changement de type.

- ***La migration des attributs*** est une forme d'optimisation appliquée aux attributs récurrents comme le nombre de retours ou la classification, dans le cas du lidar. Pour un attribut donné dans un nœud quelconque, lorsque la valeur de cet attribut est la même pour tous les enfants du nœud, cet attribut est stocké uniquement au niveau du nœud parent. Par exemple, lorsque pour un attribut toutes les feuilles d'un GHT ont la même valeur, cet attribut ne sera pas stocké dans les feuilles; il sera représenté au niveau des parents ou peut-être même à la racine. Dans le cas spécifique du lidar, beaucoup d'attributs se prêtent bien à une telle forme d'optimisation.
- ***Le changement de type*** est une forme d'optimisation appliquée aux attributs stockés dans des types de données de grande taille (p. ex. point flottant, double précision, etc.). Cette optimisation permet de changer le type d'un attribut afin de minimiser son espace de stockage lorsque cela est nécessaire. Pour les attributs de type « point flottant », on utilise une résolution afin de les transformer en nombres entiers. Pour chaque attribut, une seule résolution est utilisée pour tous les points du GHT. (Par exemple, la valeur 123,56 sera 12356 en utilisant la résolution 0,01.) Ensuite, pour chaque attribut qui sera optimisé, on calcule les minima et maxima. Puis, pour chaque point, on calcule la différence entre la valeur de son attribut et le minimum de cet attribut. En analysant cette différence, on détermine le type qui est plus approprié pour représenter l'attribut de ce point. Par exemple, lorsqu'un attribut d'un point vaut 123,56 en double précision, que la résolution de cet attribut est 0,01 et que la valeur minimale pour cet attribut est 122,35, l'écart sera (123,56 – 122,35)/0,01, soit 121. Dans ce cas, l'écart peut être stocké dans un type « caractère non signé » qui a une portée de 0 à 255 et une taille de un octet. Donc, cet attribut sera stocké sous la forme d'un octet au lieu de huit pour la donnée initiale en double précision. Le type initial et les statistiques (minimum, maximum) d'un attribut sont stockés au niveau de l'arbre, donc une seule fois, et utilisés pour tous les points du GHT. L'utilisation de la résolution permet aussi de contrôler le degré d'optimisation lorsqu'une précision élevée n'est pas requise (optimisation avec perte). La figure 4 montre le principe d'optimisation des GHT.

## Le stockage des GeoHashTree dans une base de données

Une fois créé et optimisé, le GHT peut être stocké dans un champ de type binaire d'une base de données. Pour cela, il doit être préalablement sérialisé. Lors du stockage du GHT dans une base de données, le Geohash de la racine peut être utilisé comme index spatial. Ainsi, on peut ajouter un champ supplémentaire qui recevra le Geohash de la racine de chaque GHT afin d'éviter de chercher inutilement dans les champs binaires lors des requêtes. Grâce à la propriété des Geohash selon laquelle plus les préfixes des index de deux endroits se ressemblent, plus ils sont proches dans l'espace, cette colonne peut devenir l'index spatial de la table une fois le contenu trié. De plus, cette même propriété des Geohash est utilisée pour partitionner la base de données lorsque nécessaire (*horizontal partitioning*). Pour faciliter les requêtes, la racine de chaque GHT contient des statistiques générales

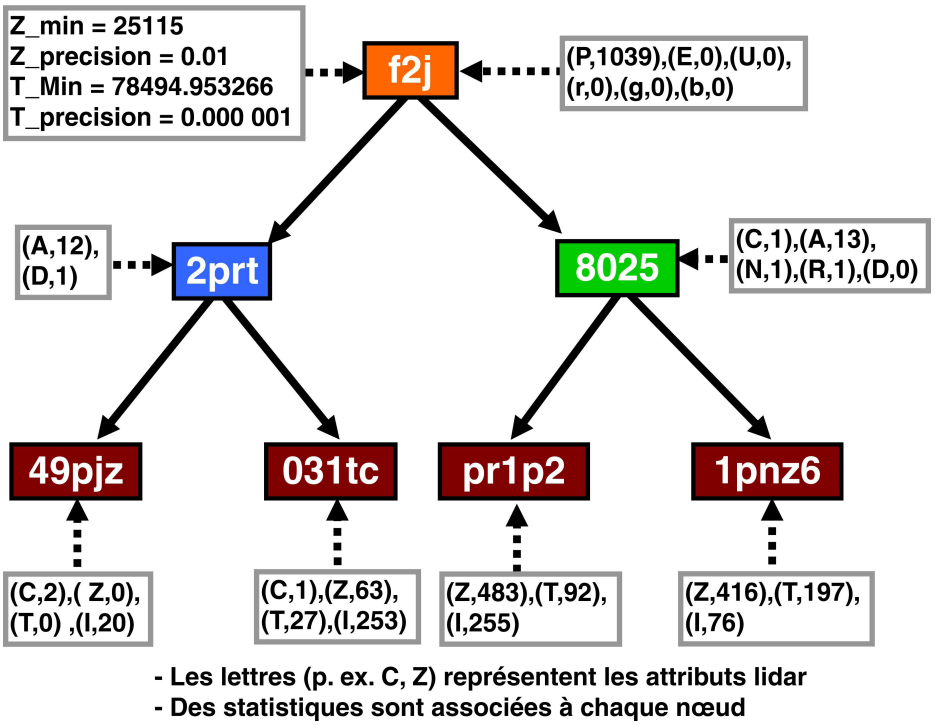


**Figure 4.** Optimisation des GeoHashTree.

pour chaque attribut (minimum et maximum). Lors des requêtes attributives, ces statistiques permettent de décider si le GHT doit être considéré et désérialisé ou pas.

# PROTOTYPE

## L'architecture du prototype

Sur la base de la structure présentée ci-dessus, un prototype fonctionnel a été développé en C++. Ce prototype permet de créer des GHT à partir des données de différents formats et de les stocker dans une base de données PostgreSQL (PostgreSQL, 2012). Ce prototype comprend trois principaux composants : le module « reader/writer », le module de création des GHT et le module « dumper/loader ».

- ***Le module « reader/writer »*** sert d'interface entre le module de création des GHT et les formats de données existants (p. ex. LAS, CSV) pour lire ou écrire dans ces formats. Il permet donc de manipuler des données lidar (en format LAS), des données matricielles (en n'importe quel format compatible avec la *Geospatial Data Abstraction Library*, ou GDAL) et n'importe quelle donnée de type XY-attribut en format texte. Le « reader » permet d'importer les données qui seront utilisées pour créer les GHT. Quant au « writer », il permet d'exporter des GHT en format LAS ou CSV. Le module « reader/writer » utilise les librairies libLAS (libLAS, 2012) pour la lecture et l'écriture des formats LAS, et GDAL pour les formats matriciels.
- ***Le module de création des GHT*** permet de prendre les données importées par le « reader », de les reprojeter au besoin, de transformer les coordonnées XY de chaque point en Geohash, de trier les points sur la base des Geohash et de créer les GeoHashTree. Ce tri permet de regrouper les points spatialement pour former des paquets de points. Chaque paquet composera un GeoHashTree. Pour la création des Geohash, la librairie Geohash développée par Kato (2012) sous licence MIT a été utilisée. En plus de permettre l'encodage et le décodage des Geohash, cette librairie permet la création du voisinage de n'importe quel Geohash, ce qui facilite la mise en place d'une stratégie de sélection des données.
- ***Le module « dumper/loader »*** sert d'interface entre le module de création des GHT et la base de données. Il permet d'une part de sérialiser les GHT créés grâce au module précédent et de les stocker dans la base de données, et d'autre part de sélectionner les GHT stockés dans la base de données (suite à une requête d'utilisateur) et de les désérialiser. Lors de la sélection, il utilise une requête SQL spécifiant à la fois l'étendue du territoire et les attributs à sélectionner. Ce module est basé sur la librairie Libpq contenue dans PostgreSQL. Libpq est une librairie C qui sert d'interface avec la base de données PostgreSQL. Ainsi, un programme client peut utiliser cette librairie pour se connecter à la base de données et réaliser des requêtes SQL.

En plus de créer des GHT à partir des nuages de points et de les stocker dans une base de données PostgreSQL, le prototype permet aussi d'interagir avec cette base de données grâce à des requêtes spatiales et attributives. Ainsi, un utilisateur peut par exemple sélectionner tous les points inclus dans un rectangle englobant dont l'élévation est supérieure à 150 m, et stocker les coordonnées XY, l'élévation et le nombre de retours dans un fichier CSV. Bien sûr, la requête attributive se fait sur n'importe quel attribut de la donnée et le résultat peut contenir n'importe quel attribut demandé par l'utilisateur. L'architecture simplifiée du prototype est présentée à la figure 5.

## Les données et matériels du test

Pour tester le prototype, des données lidar de l'état d'Indiana (É.-U.) en format LAS ont été utilisées. Les données d'essai couvrent environ 23 km$^2$ (sur la base du rectangle englobant) et contiennent 21 364 937 points dans un fichier LAS de 570,5 Mo. Chaque point du jeu de données porte 11 attributs. Ces données ont été transformées en GHT et stockées dans la base de données PostgreSQL. Pour faciliter la création des GHT (afin d'éviter le problème de mémoire), le fichier LAS a été scindé en six fichiers de taille plus petite (environ 100 Mo). Tous les tests ont été réalisés sur un ordinateur muni d'un processeur i5-3570@CPU3.4GHz et d'une mémoire vive de 8 Go sous Windows.

## Le choix de la taille des GHT et la création des GHT

Ainsi, à partir de chaque fichier LAS, les coordonnées XY de chaque point ont été transformées en Geohash d'une longueur de 14 caractères. Cette longueur permet de garantir une précision au millimètre près, ce qui est amplement suffisant car les données lidar initiales ont une précision centimétrique. Le résultat de cette étape est une liste dont chaque élément est composé du Geohash et de tous les attributs d'un point. À partir de cette liste, nous avons créé un GeoHashTree pour chaque ensemble de points qui partagent les sept premiers caractères de Geohash (le préfixe). La longueur du préfixe a une grande influence sur le nombre et la taille des GHT qui seront créés, et aussi sur le taux de compression et la performance du GHT. Le choix du préfixe de longueur 7 est basé sur des tests présentés dans le graphique de la figure 6. Selon ces tests, on peut constater que les préfixes de longueur inférieure ou égale à 7 présentent le meilleur taux de compression. On constate aussi sur le graphique de la figure 7 que lorsque le préfixe est trop court, la taille des GHT augmente de façon considérable, entraînant une grande utilisation de mémoire (*high memory footprint)* et par là-même d'éventuels problèmes de mémoire. Par contre, lorsque la taille du GHT est trop petite, on constate une dégradation du taux de compression, car l'optimisation utilisée est basée sur le partage des attributs entre différents points, donc efficace seulement lorsque le nombre de points d'un GHT est

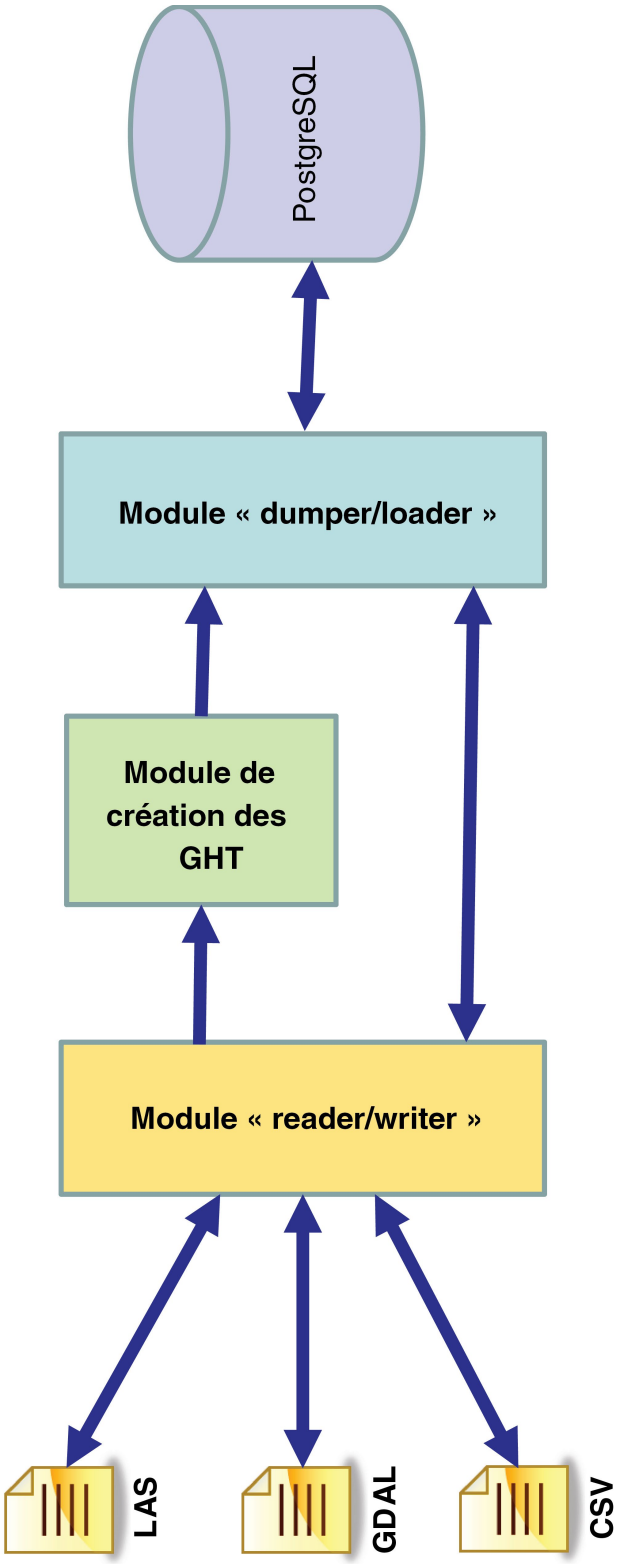


**Figure 5.** Architecture simplifiée du prototype.

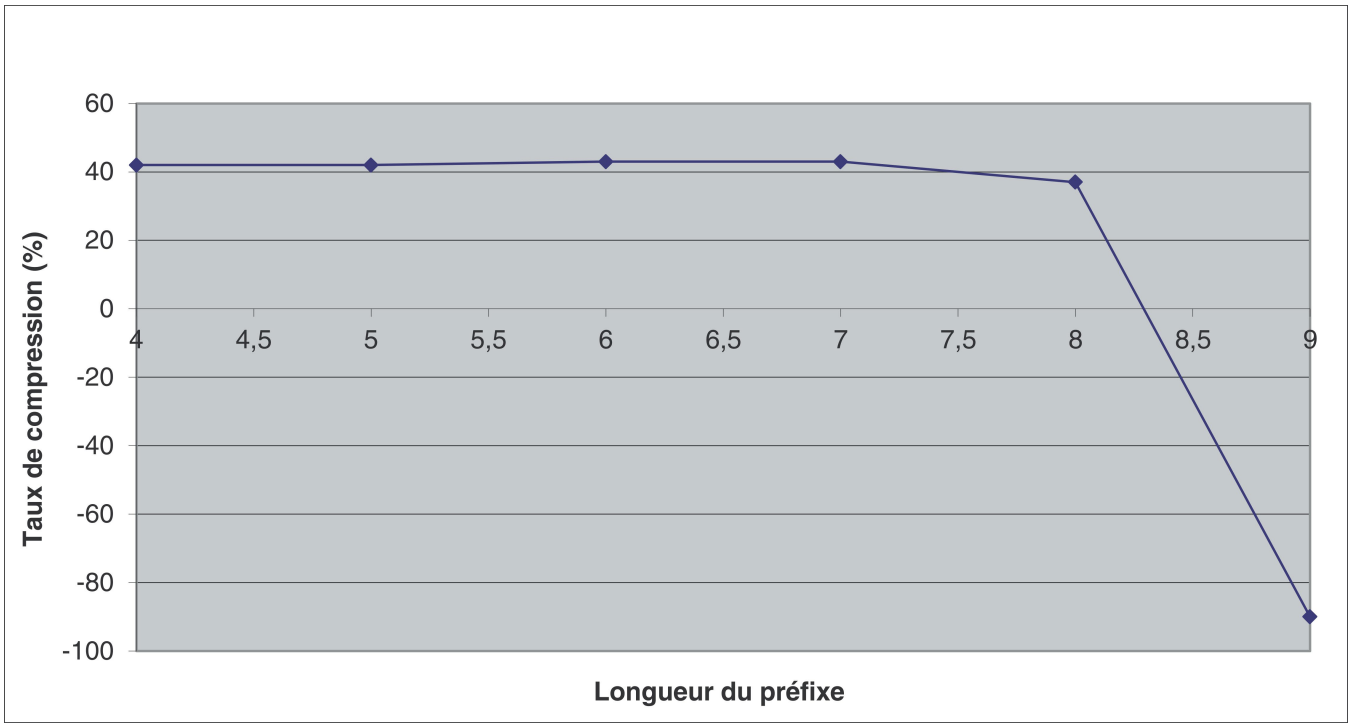


**Figure 6.** Taux de compression (par rapport au LAS) selon la longueur du préfixe dans un GeoHashTree.

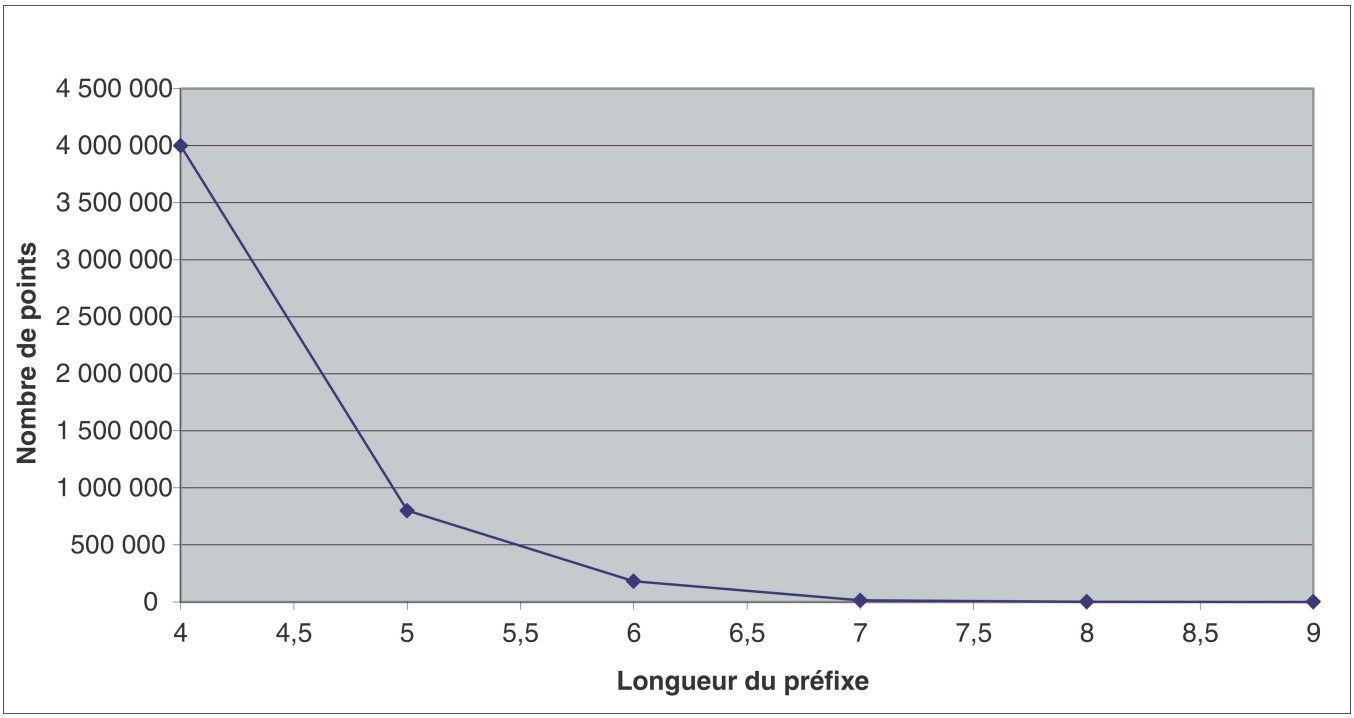


**Figure 7.** Nombre moyen de points dans un GeoHashTree.

relativement élevé. Pour toutes ces raisons, notre choix s'est arrêté sur un préfixe de sept caractères. Il est évident que cette longueur de préfixe, de même que le nombre de caractères des Geohash, sont spécifiques à nos données d'essai. Pour chaque type de données et type d'utilisation, ces paramètres doivent être choisis selon la résolution de la donnée, la performance et le taux de compression recherchés.

Comme les données initiales sont en coordonnées UTM, le processus de création des GHT inclut aussi une transformation préalable des coordonnées UTM en coordonnées géographiques tout en conservant le même système de référence (Système géodésique mondial de 1984, ou WGS84). Ainsi, 1 687 GHT ont été crées et stockés dans la base de données PostgreSQL 9.0.1, en format bytea, dans la colonne GHT de la table. De plus, pour chaque GHT, le Geohash racine (longueur 7) est stocké dans un deuxième champ du même enregistrement afin de faciliter les interrogations spatiales. Le temps moyen pour créer les 1 687 GHT, les optimiser, les sérialiser et les stocker dans la base de données est d'environ quatre minutes. Ceci inclut le temps de transfert réseau. Une fois stockés dans la base de données, les GHT nouvellement créés n'occupent que 321 Mo, soit un taux de compression de 44 % par rapport au fichier LAS initial. Comme nous l'avons mentionné plus haut, le GHT n'est pas seulement un moyen de stocker les données lidar; il permet entre autres de stocker les nuages de points de différents types et résolutions et d'avoir des données à plusieurs niveaux d'abstraction dans une même structure. Dans une telle structure, chaque point est indexé, ce qui facilite l'accès arbitraire aux données.

## Les tests d'exploitation

Pour tester l'exploitation des GHT stockés dans la base de données, des requêtes spatiales et attributives ont été appliquées. Deux requêtes spatiales ont été utilisées : 1) la sélection des données incluses dans une grande étendue spatiale (100 % des données d'essai); et 2) la sélection des données incluses dans une étendue limitée (couvrant environ 3 % de toutes les données d'essai). Par la suite, chacune des requêtes spatiales a été combinée aux requêtes attributives (le nombre de retours R = 2 et l'élévation Z ≥ 309). Pour chaque requête combinée, les GHT sélectionnés sont transformés en points et stockés dans un fichier CSV sous forme X,Y,Z. Le tableau 1 montre les résultats des tests. Par exemple, le temps nécessaire pour sélectionner et transformer en fichier CSV (X,Y,Z) tous les points ayant un nombre de retours égal à 2 est de 17 secondes. Ce temps est plus court que le temps nécessaire pour faire la même requête à partir d'un fichier LAS stocké sur le même réseau en utilisant l'application las2las (27 secondes pour sélectionner les données à l'aide de las2las plus 5 secondes pour transformer les points sélectionnés en CSV en utilisant l'application las2txt). Le but de ces tests n'est pas de comparer la performance du GHT à celle du LAS, mais tout simplement de donner une idée de la performance du GHT.

**Tableau 1.** Résultats des tests de performance

| **Sélection de données incluses dans tout le jeu de données** | | | **Sélection de données incluses dans une étendue limitée** | | |
|---|---|---|---|---|---|
| Condition « Where » | Nombre de points | Temps de traitement (s) | Condition « Where » | Nombre de points | Temps de traitement (s) |
| Tous | 21 364 937 | 73 | Tous | 580 150 | 2,1 |
| R = 2 | 617 214 | 17 | R = 2 | 5 079 | 0,4 |
| Z≥309 | 1 072 | 20 | Z≥309 | 2 | 0,3 |

# CONCLUSIONS ET DISCUSSION

Dans cet article, nous avons présenté une structure de données multirésolution et multisource novatrice, le GeoHashTree, qui permet de stocker des nuages de points ou tout autre type de données ponctuelles dans une base de données tout en facilitant la gestion et l'exploitation de ces données, ainsi que leur intégration avec d'autres types de données. Bien que la principale caractéristique visée lors du développement de cette structure ait été la flexibilité, il n'en demeure pas moins que le GeoHashTree optimise l'espace de stockage et présente une excellente performance lors de l'interaction avec la base de données. L'implantation actuelle est un compromis entre l'optimisation de l'espace de stockage, l'optimisation de la performance et la flexibilité de la structure. Malgré ce compromis, le GHT présente une performance et un degré de compression acceptables, surtout quand on sait qu'une grande quantité de données doit transiter par le réseau avec le temps de latence qu'on connaît. Un des avantages de cette approche est la flexibilité et la polyvalence de la solution. Cette flexibilité permet de gérer des données de différentes sources dans une même structure.

Vu les résultats des tests et la performance des systèmes existants de stockage des nuages de points par fichiers (p. ex. le fichier LAS), nous pensons que l'approche développée présente une excellente performance, surtout que celle-ci (en termes de stockage et de vitesse d'accès) peut encore être améliorée. En effet, dans l'implantation actuelle, la base de données sert seulement à stocker les données. Il n'existe aucun mécanisme de préfiltrage attributif des données lors des requêtes puisque les GHT ne sont pas intégrés comme type dans la base de données. La conséquence d'une telle situation est que, pour chaque requête spatio-attributive, tous les GHT inclus dans l'aire de sélection doivent être transférés vers le client même si beaucoup de ces GHT ne répondent pas aux critères attributifs et si les GHT possèdent une métadonnée d'en-tête qui nous permet de savoir s'ils répondent à la condition de sélection sans même les désérialiser au complet. Par exemple, pour la sélection des données incluses dans une grande étendue spatiale (100 % des données d'essai) ayant un Z ≥ 309, tous les GHT (1 687) ont dû être transférés vers le client même si, en réalité, seuls quatre d'entre eux contenaient des points répondant à la requête attributive. Concernant le stockage, des améliorations sont encore possibles car, actuellement, l'encodage du GeoHash utilise une table à base 32, ce qui résulte en un coût général

de 3 bits par caractère de Geohash, soit jusqu'à 37 % par Geohash d'un point. Cet coût général peut être minimisé en encodant les Geohash au moyen d'une table à base 64 ou en les stockant sous forme binaire (sans conversion en caractères). En plus d'améliorer le stockage, une telle approche améliorerait aussi la performance.

Actuellement, le prototype utilise seulement des données en coordonnées géographiques. Par contre, le principe d'encodage Geohash peut être vu comme une structure en quadrants arborescents (*quadtree*), ce qui permettrait d'utiliser facilement des coordonnées non géographiques. Pour cela, il suffirait de stocker dans chaque GHT ou dans une table à part une métadonnée sur les limites de l'étendue spatiale du territoire cartographié (p. ex. les xmin, xmax, ymin, ymax de la zone UTM). Pour faciliter l'interaction entre différents GHT, ils doivent être dans un même système, et pour cela leurs Geohash doivent être générés à partir d'une même étendue spatiale. Comme les GHT présentent les données sous forme de grille multirésolution et multiniveau dans laquelle tous les points sont indexés, une telle structure permettrait de faciliter la transformation des données de distribution irrégulière en données de distribution régulière (p. ex. données matricielles). En plus, cet aspect multiniveau des GHT pourrait faciliter la visualisation des nuages de points. Une thèse de maîtrise en collaboration avec l'Université Laval est en cours afin de développer une telle capacité de visualisation des GHT exploitant l'aspect multiniveau de la structure. Actuellement, le code de la structure GHT est disponible en OpenSource (https://github.com/pramsey/libght) et dans pointcloud (https://github.com/pramsey/pointcloud), une extension PostgreSQL pour la gestion des nuages de points.

## REMERCIEMENTS

Les auteurs tiennent à remercier le programme GéoConnexions pour son appui au projet national d'altimétrie, dans le cadre duquel la structure GeoHashTree a été développée. Les auteurs aimeraient aussi remercier Herman Varma, retraité du Service Hydrographique du Canada, pour ses conseils judicieux.

## RÉFÉRENCES

Arias Prado, D.A., 2011. Efficient LiDAR data bulk load in a PostGIS database; <http://ariasprado.name/2011/02/06/lidar-bulk-data-load.html> [consulté le 20 mai 2013].

ASPRS, 2012. LAS Specification, version 1.4–R12; disponible à l'adresse <http://www.asprs.org/Committee-General/LASer-LAS-File-Format-Exchange-Activities.html> [consulté le 15 mai 2013].

Graham, L., 2009. Management of LiDAR data; Chapter 10 *in* Topographic laser ranging and scanning: principles and processing, (ed.) J. Shan and C.K. Toth; CRC Press, Florida, p. 295–306.

Huber, D., 2011. The ASTM E57 file format for 3D imaging data exchange; *in* Proceedings, SPIE 7864, Three-dimensional imaging, interaction, and measurement, 78640A, January 27, 2011. doi:10.1117/12.876555

Kato, L., 2012. Geohash library; <https://github.com/lyokato/objc-geohash> [consulté le 16 mai 2013].

libLAS, 2012. <http://www.liblas.org> [consulté le 16 mai 2013].

Nandigam, V., Baru, C. et Crosby, C., 2010. Database design for high-resolution LIDAR topography data; *in* Proceedings, Scientific and statistical database management, 22nd International Conference, SSDBM 2010, (ed.) M. Gertz and B. Ludäscher; Lecture notes in Computer Science, v. 6187, p. 151–159. doi:10.1007/978-3-642-13818-8_12

Ott, M., 2012. Towards storing point clouds in PostgreSQL; HSR Hochschule für Technik Rapperswil, Rapperswil, Switzerland, 29 p.

PostgreSQL, 2012. <http://www.postgresql.org> [consulté le 15 mai 2013].

Varma, H., Boudreau, H. et Prime, W., 1990. A data structure for spatio-temporal databases; International Hydrographic Review, Monaco, v. 67, p. 71–92.

Wikipedia, 2012. Geohash; <http://en.wikipedia.org/wiki/Geohash> [consulté le 14 mai 2013].

Projet 362304NP23 de Géomatique Canada