المملكة المغربية

ROYAUME DU MAROC

INSTITUT AGRONOMIQUE ET VÉTÉRINAIRE HASSAN II

معهد الحسن الثاني للزراعة والبيطرة

Projet de Fin d'Études

Présenté pour l'obtention du diplôme

D'Ingénieur d'État en Topographie

CLASSIFICATION ET INTEGRATION DES NUAGES DE POINTS 3D DANS UN ENVIRONNEMENT DE REALITE VIRTUELLE

Présenté et soutenu publiquement par

KHARROUBI Abderrazzaq

Devant le JURY

Pr. K.AIT EL KADI	**Présidente**	**IAV HASSAN II**
Pr. R.HAJJI	**Rapporteuse**	**IAV HASSAN II**
Pr. R.BILLEN	**Rapporteur**	**Université de Liège**
Pr. T.TACHALLAIT	**Examinateur**	**IAV HASSAN II**
Pr. A.TAMTAOUI	**Examinateur**	**INPT**

SEPTEMBRE 2019

Adresse : Madinat Al Irfane, B.P. 6202. Rabat – Maroc
Tél : (00 212) 0537 77 17 58/59
Fax : (00 212) 0537 77 58 45
Site web : http://www.iav.ac.ma

العنوان: ص. ب 6202 الرباط المعاهد الرباط – المغرب
الهاتف: 59 / 58 (00 212) 0537 77 17
الفاكس: (00 212) 0537 77 58 45
موقع الأنترنت:http://www.iav.ac.ma

DEDICACES

À ma mère,

À mon père qui a disparu trop tôt,

Qui n'ont jamais cessé de formuler des prières à mon égard, de me soutenir et de m'épauler pour que je puisse atteindre mes objectifs.

À mes chers frères Youness et Ismail,

À ma chère sœur,

Pour leur soutien moral et leurs conseils précieux tout au long de mes études.

À mon cher grand père, et ma chère grande mère.

À toute ma famille,

Qui m'a doté d'une éducation digne.

À ma belle,

Qui m'a aidé et supporté dans les moments difficiles

À mes chers amis,

À ceux que je connais depuis l'enfance et ceux avec qui j'ai partagé les cinq ans de l'IAV Taha ayoub, Chafiq, Akhi. À mes chers amis, Abdellatif, Khalil, Noureddine, Abdelmoumine, Abdessamad, Aken, Asklou, ELkacimi, Les Mohammeds Banouni, Belbrik, et Abdelaoui. À mes anciens de promotion Mohammeds, Boumerouane et El Ansari.

Au doctorant Abdeljalil que je souhaite une bonne continuation dans ses projets futurs.

Aux membres de l'unité de géomatique à l'ULiège,

Pour leur accueil, et tous les échanges qu'on a eu

À mes conjoints de laboratoire, Quentin et Alexis,

Aux doctorants,

Cécile Deprez, Gilles-Antoine Nys, Romain Neuville

Aux étudiants de TFE de l'unité, à Florian pour toutes les discussions

À toute ma famille et tous mes amis

Dédié à ma mère, à mon père et à ma future petite famille

À moi, à tous ceux que j'aime et ceux qui m'aiment

REMERCIEMENTS

En termes de ce travail, je souhaite adresser mes remerciements les plus sincères à mon encadrante madame **Rafika Hajji** professeur chercheur au sein du département de Géodésie et Topographie à l'IAV HASSAN II. Je tiens à remercier également mes encadrants, monsieur **Roland Billen** professeur chercheur et **Florent Poux** docteur à l'unité de géomatique de l'université de Liège qui se sont toujours montrés disponibles et à mon écoute, tout au long de la réalisation de ce mémoire; je les remercie aussi pour l'inspiration, l'aide et le temps qu'ils ont bien voulu me consacrer.

Mes remerciements s'adressent à tous les enseignants, chercheurs et doctorants de l'unité de Géomatique pour leurs générosité et conseils distillés. À ce titre, j'exprime ma gratitude à **Quentin Valembois** développeur en C# et Unity, **Gilles-Antoine Nys** doctorant à l'unité, ainsi qu'à toute l'équipe de l'unité de Géomatique qui ont facilité mon intégration au sein de l'équipe et m'ont toujours encouragé et aidé à surpasser les difficultés.

Je remercie mes chers professeurs de la filière de formation en sciences géomatiques et ingénierie topographique qui m'ont apporté leur aide et qui m'ont accompagné le long de mon cursus.

J'adresse aussi mes profondes gratitudes et mes hommages les plus respectueux aux membres de jury qui ont accepté de participer à l'évaluation de ce travail.

Enfin, mes reconnaissances et mes respects à tous ceux qui ont contribué de près ou de loin à la réalisation de la présente étude.

PREAMBULE

Le présent travail intitulé « Classification et intégration des nuages de points 3D dans un environnement de réalité virtuelle », a été effectué au sein de l'unité de Géomatique qui fait partie du département de Géographie (faculté des sciences) à l'université de Liège et il s'insère dans le cadre du projet « Terra Mosana ». Ce dernier est une collaboration entre municipalités, sites patrimoniaux, universités et citoyens afin de renforcer l'attractivité touristique de l'Euregio Meuse-Rhin (EMR) et le sentiment d'appartenance à celle-ci grâce à l'exploitation numérique de son patrimoine culturel. Ce travail consiste à effectuer des mesures sur terrain par lasergrammétrie, photogrammétrie, prétraiter les données acquises, les traiter et les indexer spatialement en vue de leur exploitation dans un dispositif de réalité virtuelle.

RESUME

Ce mémoire d'ingénieur en sciences géomatiques et ingénierie topographique propose une application de réalité virtuelle pour la visualisation et l'interaction avec les nuages de points massifs dans un environnement virtuel. La solution s'inscrit dans un Framework global post classification s'appuyant sur des structures de données adaptées à la gestion des données massives en temps réel (plusieurs milliards de points). En général, les nuages de points constituent des données 3D de base pour le développement d'un bon nombre d'applications notamment celles liées à la réalité virtuelle. Cependant, ces données brutes n'intègrent pas d'information sémantique sur les objets qui composent une scène. Ainsi, les nuages de points nécessitent un prétraitement et une classification en vue d'une visualisation sémantique. À travers notre étude, nous présentons un processus de classification pour l'intégration de l'information sur le nuage de points. Ensuite, nous développons un système de rendu des nuages de points massifs en réalité virtuelle sous l'outil Unity. Dans ce travail, l'accent est principalement mis sur l'intégration de la classification dans un environnement virtuel et sur la gestion de ces nuages de points massifs, dépassant plusieurs milliards de points, ne pouvant pas être chargés directement dans la mémoire vive. Ainsi, une structure de données de type « Octree » est employée et des algorithmes « Out-of-Core » sont utilisés pour charger en temps réel et en continu, uniquement les points qui figurent dans le champ de vision de l'utilisateur. Nous proposons une solution de visualisation en réalité virtuelle avec une interface utilisateur permettant l'interaction avec le nuage de points et le changement des paramètres et des méthodes de rendu. Cette solution intégrée offre aussi un mode de déplacement dans le nuage de points afin de garantir à l'utilisateur une expérience immersive.

Mots clés : Nuage de point, Segmentation sémantique, Réalité virtuelle, Potree, Octree, Unity

Encadrant	**Candidat**	**Co-encadrants**
Pr. R. HAJJI (1)	KHARROUBI Abderrazzaq	Pr. R. BILLEN (2)
		Dr. F. POUX (2)

(1) Département de Géodésie et Topographie-IAV HASSAN II, Rabat

(2) Unité de Géomatique-ULiège, Liège

ABSTRACT

This engineering dissertation in Geomatics and Surveying Engineering offers a virtual reality solution for viewing and interacting with massive point clouds in a virtual environment. This solution is part of a global post classification framework based on data structures adapted to the management of massive data in real time (several billions points). In general, 3D data production processes initially rely on 3D point clouds. However, these raw data don't include semantic information about the objects that make up a scene. Thus, the considerable volume of point clouds requires pretreatment and classification for semantic visualization. Through our study, we present a classification process for the integration of point cloud information. Then, we develop a massive point cloud rendering system in virtual reality under the Unity tool. In this work, the main focus is on the integration of the classification in a virtual environment and on the management of these massive points cloud, exceeding several billion points, which can't be loaded directly into random access memory. Thus, an octree data structure is employed and Out-of-Core algorithms are used to load only the points that are in the user's field of view in real time and continuously. We offer a virtual reality visualization solution with a user interface for interacting with the points cloud, and changing the parameters and rendering methods. This integrated solution also offers a mode of movement in the cloud of points to guarantee the user an immersive experience.

Keywords: Point Cloud, semantic segmentation, virtual reality, Potree, Octree, unity,

TABLE DES MATIERES

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ABREVIATIONS

AR: Augmented reality

BRIEF: Binary Robust Independent Elementary Features

CAVE: Cave Automatic Virtual Environment

CLOD: Continuous Level of Detail

CSF: Cloth Simulation Filter

DLOD: Discreet Level of Detail

EMR: l'Euregio Meuse-Rhin

FPS: Frame per Second

GNSS: Global Navigation Satellite System

ICP: Iterative Closest Point

IU: Interface Utilisateur

LOD: Level of Detail

LPC: Layered Point Cloud

LRU: Least Recently Used

MNO: Modifiable Nested Octree

MR: Mixed reality

ORB: Oriented FAST and Rotated BRIEF

RAM: Random Access Memory

RANSAC: RANdom SAmple Consensus

SFM: Structure from Motion

SIFT: Scale-Invariant Feature Transform

SIG: Systéme d'information Géographique

SURF: Speeded Up Robust Features

VR: Virtual Reality

INTRODUCTION GENERALE

Contexte

Au cours de la dernière décennie, l'utilisation croissante des scanners lasers et des méthodes de reconstruction photogrammétrique, le développement des algorithmes de traitement et l'augmentation de la puissance de calcul des ordinateurs, ont conduit à la création de gigantesques « data sets » de nuage de points (Scheiblauer et al., 2014). En plus de l'information sur la position, les données acquises par lasergrammétrie comportent une composante colorimétrique associée à chaque point. Le modèle 3D résultant : modèle « tel que construit », peut ainsi être rendu réalistique par application des algorithmes de rendu des nuages de points (Mures et al., 2016). Aussi, les nuages de points peuvent être aussi sémantisés et rendus intelligents par des processus de segmentation (Poux, 2019). C'est le cas, par exemple, pour leur exploitation dans des « digital twins » qui constituent un lien en temps réel entre les mondes physique et numérique grâce à la synchronisation permanente des données via les différentes capteurs et intervenants.

Dans plusieurs domaines comme l'architecture, l'archéologie et l'ingénierie, la production de modèles 3D représentant la réalité terrain d'une manière détaillée et rapide revêt une grande importance. En effet, à travers l'application de nouvelles méthodes de visualisation 3D dont la principale pour ce travail est la réalité virtuelle, les ingénieurs peuvent visualiser et naviguer dans un environnent virtuel pour voir l'état d'avancement et contrôler la qualité des travaux faites sur chantier ; les archéologistes peuvent voir des copies originales des sites archéologiques et du patrimoine, qu'ils peuvent consulter et utiliser pour des besoins d'enseignement et de formation ; les architectes peuvent contrôler l'état des bâtiments ainsi qu'avoir des informations pour l'exploitation et la mise à jour de leurs maquettes numériques. Les urbanistes peuvent visualiser une ville entière pour des fins de planification et d'aménagement urbain (Mures et al., 2016). La réalité virtuelle qui consiste à remplacer l'environnement réel par un environnement complétement virtuel à l'aide d'un casque ou dans un CAVE (Cave Automatic Virtual Environment), constitue un nouveau mode de visualisation immersive dans lequel l'utilisateur peut interagir avec les objets sans les limitations que représente le monde réel. Ce nouvel mode offre plusieurs avantages dont le plus important réside dans l'immersion complète dans l'environnement virtuel par le biais

de retours sensoriels, le plus souvent visuel et stéréoscopique, mais également auditif et/ou haptique c'est-à-dire sur le canal du toucher (Novak, 2014).

Problématique

A défaut de leur nature d'ensembles discrètes de données spatiales, les points ne peuvent pas représenter un objet physique entier tel que vu et interprété par l'être humain (Poux, 2019). Vu leur grand volume, la visualisation et le rendu 3D présentent un grand défi, vu leur exigence en termes de capacités de stockage et de traitement dont ne disposent pas toutes les stations de travail actuelles (Scheiblauer et al., 2014). En outre, la visualisation sur un écran à deux dimensions donne moins d'effet d'immersion et moins de réalisme de l'environnement acquis.

Objectif

Ce travail, que j'ai réalisé au sein de l'unité de géomatique de l'Université de Liège en Belgique, tente clairement à résoudre ces problèmes de visualisation des nuages de points massifs, de combler le manque de l'information sémantique dans ces nuages de points par le processus de classification et vise l'intégration de ce résultat dans un environnement virtuel.

L'objectif principal du présent travail est de : **développer une solution qui permet d'enrichir un nuage de point massif par l'information sémantique, et l'intégrer dans un environnement de réalité virtuelle.**

Méthodologie

Pour atteindre les objectifs fixés par ce travail, une méthodologie propre à ce mémoire sera suivie.

- ✓ On commence par la segmentation des nuages de points, leurs classifications sur Cloud Compare et leur structuration sur PotreeConverter.
- ✓ Le résultat de la classification est adapté et visualisé sous le Viewer WebGL de Potree. Puis nous procédons à l'implémentation de la solution sous le Game Engine unity. Ensuite, l'application est déployée pour la visualisation avec le casque RV Oculus rift.
- ✓ À la fin, des tests de performances sont réalisés sur base des critères objectifs comme la consommation en mémoire, et le nombre d'images par seconde. Une analyse des résultats s'avère nécessaire afin de savoir les ponts d'innovation et la qualité de la solution développée.

Organisation du document

Le présent document est structuré en cinq chapitres :

Le premier chapitre est consacré à la mise en situation dans le contexte du travail, en présentant l'unité accueillante, en décrivant l'environnement du travail, et en donnant un aperçu global sur le projet dans le cadre duquel s'insère le présent travail.

Le deuxième chapitre présente les principes fondamentaux d'acquisition, de prétraitement, de segmentation et de classification des nuages de points. Les structures de données spatiales sont présentées en se focalisant sur la structure hiérarchique en Octree imbriquée. Un aperçu général sur les principes de réalité virtuelle est présenté.

Le troisième chapitre traite la méthodologie générale du présent travail qui consiste en la segmentation et la classification des nuages de points avec l'outil Cloud Compare[1] et la structuration de ces résultats en format hiérarchique d'Octree. Les étapes de traitement et de développement avec chaque outil sont détaillées et les résultats intermédiaires sont illustrés.

Le quatrième chapitre est dédié à l'implémentation de la solution avec C# sous l'outil Unity. L'implémentation sur unity est détaillée à partir de la lecture de la structure Octree de Potree, jusqu'à le rendu du nuage du point. Puis, l'intégration du dispositif de réalité virtuelle « Oculus rift » est détaillée.

Dans le dernier chapitre, plusieurs tests avec plusieurs nuages de points, et plusieurs utilisateurs sont faits pour valider les performances de la solution implémentée, et les résultats sont analysés et interprétés. Puis, l'apport de la solution dans l'intégration des nuages de points dans un environnement de réalité virtuelle est discuté et les perspectives de recherche pour la continuité de ce travail sont détaillées.

[1] https://www.danielgm.net/cc/

CHAPITRE I
CONTEXTE DE L'ETUDE

Introduction

Cette section est consacrée à la mise en situation dans le contexte du travail. Nous présentons l'unité accueillante, l'environnement du travail et nous donnons un aperçu sur le projet dans le cadre duquel s'insère le présent travail.

1.1 La structure d'accueil

Ce travail a été réalisé au sein de l'unité de Géomatique [2] à l'université de Liège en Belgique. Cette unité regroupe des enseignants-chercheurs et des chercheurs actifs dans le domaine de la Géomatique. Elle fait partie du département de Géographie (Faculté des Sciences) de l'Université de Liège. L'unité assure de nombreux enseignements (incluant les SIG ; systèmes d'information géographique, la cartographie, l'analyse spatiale, la Topographie, la géo métrologie, GNSS ; les systèmes de positionnement par satellites, la Géodésie, la Télédétection, la Photogrammétrie, etc.) pour plusieurs sections d'études aussi bien au niveau bachelier que master. Elle est plus spécifiquement concernée par la formation en Géomatique et Géo métrologie unique en communauté Wallonie-Bruxelles et donnant accès à la profession de Géomètre-Expert. Elle est également impliquée dans des cycles de formation continue.

La recherche occupe une place prépondérante dans les activités de l'unité, au travers de nombreux projets de recherche et de publications. Les domaines couverts sont les systèmes d'information géographique, les systèmes de positionnement par satellites et les méthodes d'acquisition de données spatiales. L'unité jouit d'une forte reconnaissance internationale. Elle joue également un rôle important auprès des institutions et des professionnels belges du domaine.

[2] http://www.geo.ulg.ac.be/fr/home.php

1.2 Le Projet Terra Mosana [3]

Le projet « Terra Mosana » est un projet de collaboration entre municipalités, sites patrimoniaux, universités et citoyens visant à renforcer l'attractivité touristique de l'Euregio Meuse-Rhin [4] (EMR). Le projet ambitionne d'accroître le sentiment d'appartenance à l'Euregio grâce à l'exploitation numérique de son patrimoine culturel. Cette exploitation se fera au travers de récits numériques de l'histoire partagée de plusieurs villes de l'EMR (*Figure 1. 1*). Ces récits numériques seront constitués en développant et exploitant, en réalité virtuelle ou en réalité augmentée, des modèles 3D réalisés à partir du patrimoine archéologique, urbain, matériel et immatériel. Les modèles 3D concerneront aussi bien le passé et le présent que les évolutions à venir. L'emploi de ces technologies favorise le partage de données et par conséquent permet de renforcer la cohésion entre les sites patrimoniaux de l'Euregio Meuse-Rhin tout en propulsant la valorisation du patrimoine dans l'ère du numérique.

Le présent travail est une contribution aux objectifs du projet, par la proposition d'une approche d'optimisation du rendu de grands nuages de points par classification et visualisation dans un environnement virtuel. La finalité de ce travail est de proposer une solution pour faciliter la gestion de ces nuages de points volumineux et les exploiter dans une application de réalité virtuelle en se passant de l'étape de modélisation.

Dans ce travail, nous avons utilisé les nuages de points 3D du château de JEHAY pour faire cette application. Ce château (*Figure 1. 2*) fait partie du patrimoine culturel immobilier classé de la Wallonie qui est l'une des trois régions de la Belgique et qui fait partie aussi de l'Euregio Meuse-Rhin, donc concerné directement par le projet TerraMosana. La contribution de notre travail dans le projet consiste dans la nouvelle méthode développée pour la visualisation des nuages de points massifs comme celui du château de Jehay via RV, après intégration des informations de classification.

[3] http://www.terramosana.org/

[4] Créée en 1976, est l'une des plus anciennes coopérations transfrontalières situées au cœur de la Communauté européenne et constituée sous la forme d'une eurorégion.

Figure 1. 1 *Carte des régions de l'Euregio Meuse-Rhin*

Figure 1. 2 *Chateau de Jehay*

1.3 L'environnement de travail

Ce travail s'est déroulé en commun dans le laboratoire de l'unité avec deux chercheurs travaillants sur le projet de TerraMosana, Quentin Valembois et Alexis Jacquemin, ainsi qu'avec Abdellatif Tabkha élève ingénieur de l'IAV Hassan II qui travaille sur une nouvelle approche de sémantisation des nuages de points dans son projet de fin d'études.

Les outils principaux de travail consistent en un scanner laser Leica P30, de deux caméras Canon 5D Mark III et Canon 70D pour la photogrammétrie et d'une station totale TPS 1200

de Leica. Il était disponible tous les accessoires topographiques pour faire une bonne mission de levé topographique, par station totale, par lasergrammétrie et par photogrammétrie.

Pour le traitement informatique, nous avons utilisé un ordinateur puissant muni d'un processeur i7, de 48 Go de RAM et une carte graphique Nvidia GeForce GTX 1080. Ce type de caractéristiques facilitera énormément les tests de visualisation en réalité virtuelle. La partie software englobe les logiciels de traitement des nuages de points (Cyclone, CloudCompare, ContextCapture), de génération des photos panoramiques (Auto pano Giga) et de visualisation et de rendu 3D (Potree, Unity 3D).

CHAPITRE II ETAT DE L'ART SUR LA SEGMENTATION, CLASSIFICATION, STRUCTURATION ET VISUALISATION DES NUAGES DE POINTS

Introduction

Le présent chapitre présente une revue bibliographique des méthodes d'acquisition d'un nuage de points, les étapes de post-traitement et les principales méthodes de segmentation et de classification. Il est présente aussi les différentes structures de données spatiales permettent le rendu de grands nuages de points. Ce chapitre se termine par un aperçu général sur les principes fondamentaux de la réalité virtuelle.

Les sections de ce chapitre suivent l'organigramme suivant :

Ordre des étapes :	Acquisition et prétraitement des nuages de points
	Segmentation
	Classification
	Structuration/Indexation spatiale
	La réalité virtuelle comme nouveau mode de visualisation immersive en 3D

Figure 2. 1 *Organigramme présentant l'ordre des étapes*

2.1 Acquisition et traitement des nuages de points

Un des formats de données résultant de la numérisation d'un espace 3D est le « nuage de points ». Cela peut être constitué principalement par lasergrammétrie qui est une méthode active basée sur la mesure de distance, ou par photogrammétrie qui est une méthode passive basée sur des images numériques.

La lasergrammétrie est une technique d'acquisition qui utilise le rayonnement laser pour balayer une zone directement et rapidement à partir d'une station laser fixée à bord d'une plate-forme aérienne ou terrestre, fixe ou mobile. Elle génère un nuage de points (X, Y, Z), dont la densité peut être très élevée. De plus, elle est capable d'enregistrer la composante

colorimétrique, et l'intensité laser de chaque point mesuré (Ait El Kadi, 2016; Poux, 2014) comme montré sur *Figure 2. 2*, où un bâtiment est visualisé avec sa composante colorimétrique.

Figure 2. 2 *Un nuage de point 3D crée par lasergrammétrie*

L'instrument de mesure est composé d'un télémètre laser pour transmettre et acquérir un faisceau, d'un ou plusieurs composants de déviation du faisceau laser, de systèmes de mesure des informations de distance et de direction et d'une partie pour traiter et enregistrer les informations transmises.

Les éléments de déviation assurent respectivement les mouvements horizontaux et verticaux des miroirs du scanner. Avant de passer à la phase de traitement, un premier produit est formé sous forme de matrice dont chaque point est positionné selon ses angles de vue (φ, Θ) et sa distance par rapport au scanner. La partie traitement consiste à extraire les coordonnées cartésiennes 3D (X, Y, Z) de chaque point en fonction de ses coordonnées sphériques mesurées (r, φ, Θ) (*Figure 2. 3*)

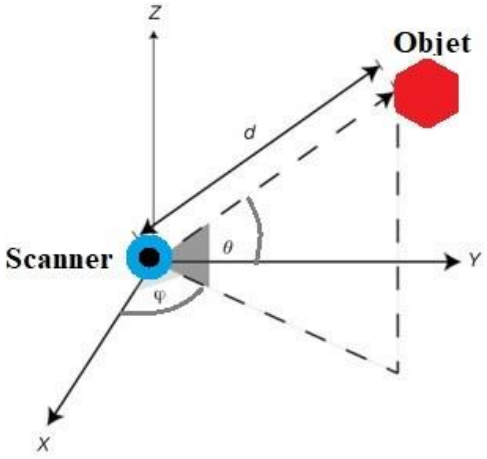


Figure 2. 3 *Coordonnées sphériques d'un scanner laser terrestre*

Le deuxième mode d'acquisition des nuages de points cité dans ce travail est la photogrammétrie. Elle consiste dans la création d'un modèle 3D (*Figure 2. 4*) basé sur la prise des photos à des emplacements variés, de la même façon qui est employée par notre cerveau pour obtenir l'image mentale en trois dimensions d'une scène. C'est ce qu'on appelle la stéréoscopie qui permet de créer une perception de relief à partir d'au moins deux images planes.

Figure 2. 4 *Exemple de modèle crée par photogrammétrie du statut nightmare roller coaster, château de Jehay, Liège*

A l'inverse d'une bulle de nuage de points issue d'un scanner positionné au centre d'un espace (un seul œil en quelque sorte) la photogrammétrie produit un nuage de points grâce à plusieurs points de vue (plusieurs yeux répartis dans la scène *Figure 2. 5*).

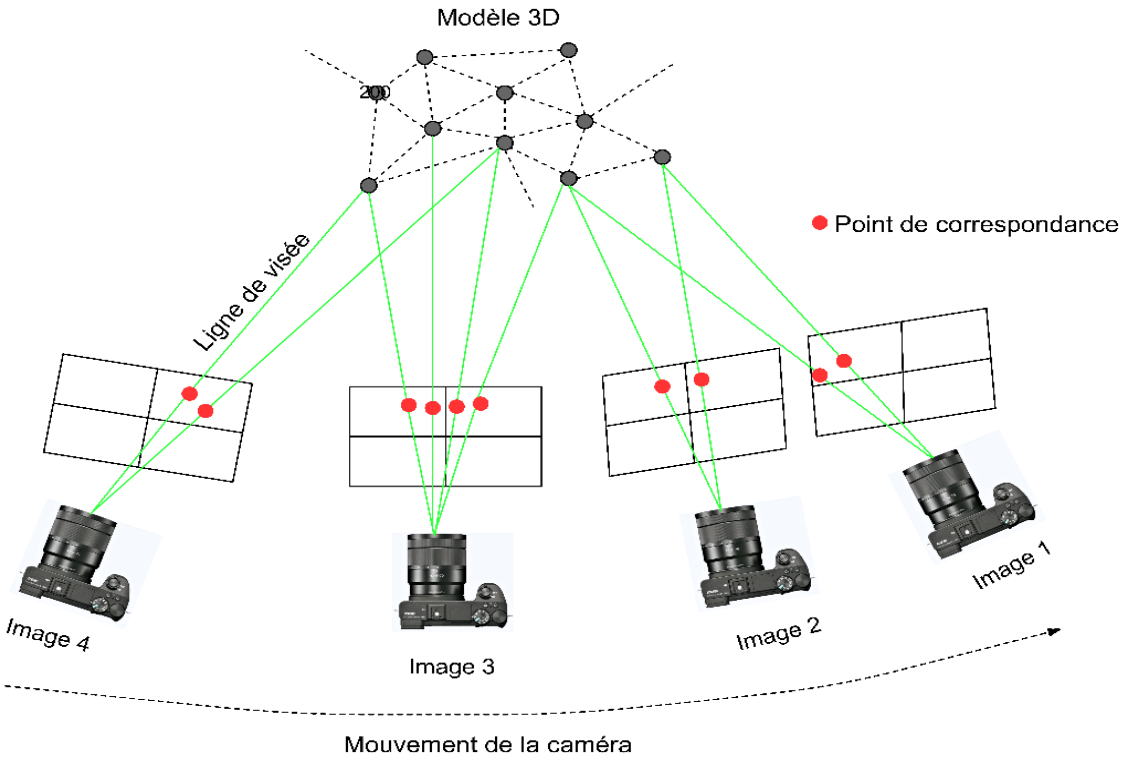


Figure 2. 5 *Le principe d'acquisition par reconstitution photogrammétrique*

la figure ci-dessus montre le principe de reconstitution photogrammétrique à partir de plusieurs images prisent à partir des positions différentes et des orientations variables, de multiple points de correspondances sont utilisées pour faire une orientation relative des images l'une par rapport à l'autre, sur base des algorithmes spécifiques comme le SIFT, SURF, BRIEF, ORB (Karami et al., 2017) (High and Topography, 2016)

Les points clés servent à calculer les emplacements relatifs des caméras afin de créer un nuage de points «faible densité» clairsemé. Un nuage de points de haute densité est ensuite généré sur la base des emplacements des points clairsemés et les emplacements de la caméra comme illustré avec la *Figure 2. 6*.

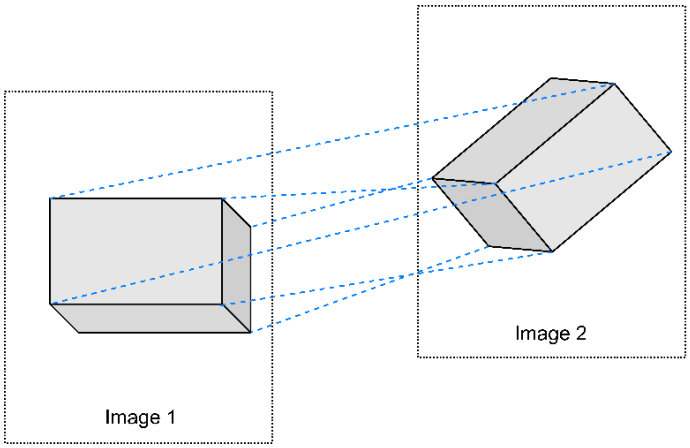


Figure 2. 6 *Recherche des points de liaison dans deux ou plusieurs images de l'objet sous différents points de vue*

Un modèle 3D généré par photogrammétrie résulte de l'assemblage tridimensionnel de la scène globale. La particularité d'un modèle photogrammétrique est qu'il est nécessaire de le remettre à l'échelle et de lui faire un géo référencement.

Assurément, un scanner laser est capable de mesurer de manière absolue les distances, ce qui, après assemblage, résulte en un nuage de dimensions réelles. Un appareil photo, quant à lui, ne peut pas mesurer de distances. Le modèle 3D construit par photogrammétrie est un modèle réaliste de la scène, les angles entre les faces sont réels, de même que les rapports de proportionnalité entre les distances. Pourtant, la distance absolue entre ces points ne pourra être connue qu'après une « mise à l'échelle » ou un géo référencement du modèle photogrammétrique. Cette dernière est réalisée par l'injection d'informations spatiales associées à des éléments du modèle (coordonnées des photos, points de calage visibles sur site, distance entre deux points...)

2.1.1 Consolidation

Après la phase d'acquisition du nuage du point 3D qui consiste à un balayage de la zone concernée (appelé aussi scannage, numérisation ou relevé par scanner), vient la phase d'interprétation des données acquises via un processus de traitement.

En effet, en cas de géo référencement indirect, le traitement commence par la consolidation (appelé aussi recalage 3D/3D, registration en anglais, ou mise en correspondance) qui consiste à effectuer un recalage relatif entre nuages (). La consolidation se fait par l'identification de « points homologues » dans les divers nuages à consolider qui peuvent être soit des cibles (cibles réfléchissantes aux caractéristiques physiques sous forme de sphères ou des cibles plates), soit des entités géométriques homologues(plans, sphères, cylindres), soit sur base des algorithmes spécifiques(méthode DARCES basée sur RANSAC, ICP, Hachage par la transformée de Hough, Hachage géométrique) (Landes et al., 2011)
La *Figure 2. 7* illustre le nuage de points issu de la consolidation et du géoréfrencement direct que nous avons utilisé dans notre travail, des scans de l'université RTWH, en Allemagne.

Figure 2. 7 *Nuage de points issu d'une consolidation avec un géo référencement direct des stations de scan, l'Université RWTH, Allemagne*

2.1.2 Nettoyage

Une fois les divers nuages de points assemblés dans le même repère et les résultats de la consolidation validés, une étape de nettoyage du nuage de points s'avère nécessaire pour éliminer les points indésirables présentant un bruit dans la mesure. En effet, ce bruit peut être dû à la réflectance trop faible de certaines surfaces ce qui donne un signal de retour faible augmentant par conséquent, l'erreur de mesure, ou bien à cause de la réception de plusieurs signaux retour pour un même signal émis, surtout aux bords des objets. Parfois, le bruit peut être formé par un opérateur qui passe au moment du scan, d'une voiture, ou tout autre objet mobile entre deux positions de scan.

L'opération de nettoyage est faite soit manuellement par la manipulation des outils présents sur des logiciels de traitement de nuage de points (sélection, coupe) ou automatiquement à l'aide des filtres qui permettent de supprimer les points isolés sur base de la distance par rapport à leurs entourages. Comme ça peut être fait avec des algorithmes de machine Learning basé sur l'apprentissage comme le PointCleanNet (Rakotosaona and Polytechnique, 2019).

Plus récemment, une partie de cette opération pourra être effectuée par mode opératoire sur terrain par l'acquisition de deux scans l'un après l'autre (RTC360 de Leica) afin de minimiser l'effet des objets qui bougent lors du relevé (le double scan).

2.1.3 Segmentation

La segmentation est une étape primordiale pour l'introduction de l'information sémantique sur les objets physiques contenus dans le nuage de points. Shi Pu et George Vosselman, (2006) la définissent comme étant un *« processus de labélisation de chaque point du nuage de points, de façon à ce que ces points appartenant à telle ou telle surface ou région, aient le même label* ». La segmentation d'un nuage de points est un partage/subdivision de l'ensemble des points 3D en sous- ensembles (sous-nuage de points) homogènes, suivant des critères prédéfinis (Landes et al., 2011)

Il existe deux principales familles de méthodes de segmentation (*Figure 2. 8*), que sont la segmentation par reconnaissance de primitives géométriques(dont la transformé de Hough, et l'algorithme de RANSAC (Fischler and Bolles, 1981)), et la segmentation par

regroupement de points (dont les algorithmes de croissance de région, regroupements en classes, profils de balayage, division fusion)(G. Vosselman, B.G.H. Gorte, G. Sithole, 2002)

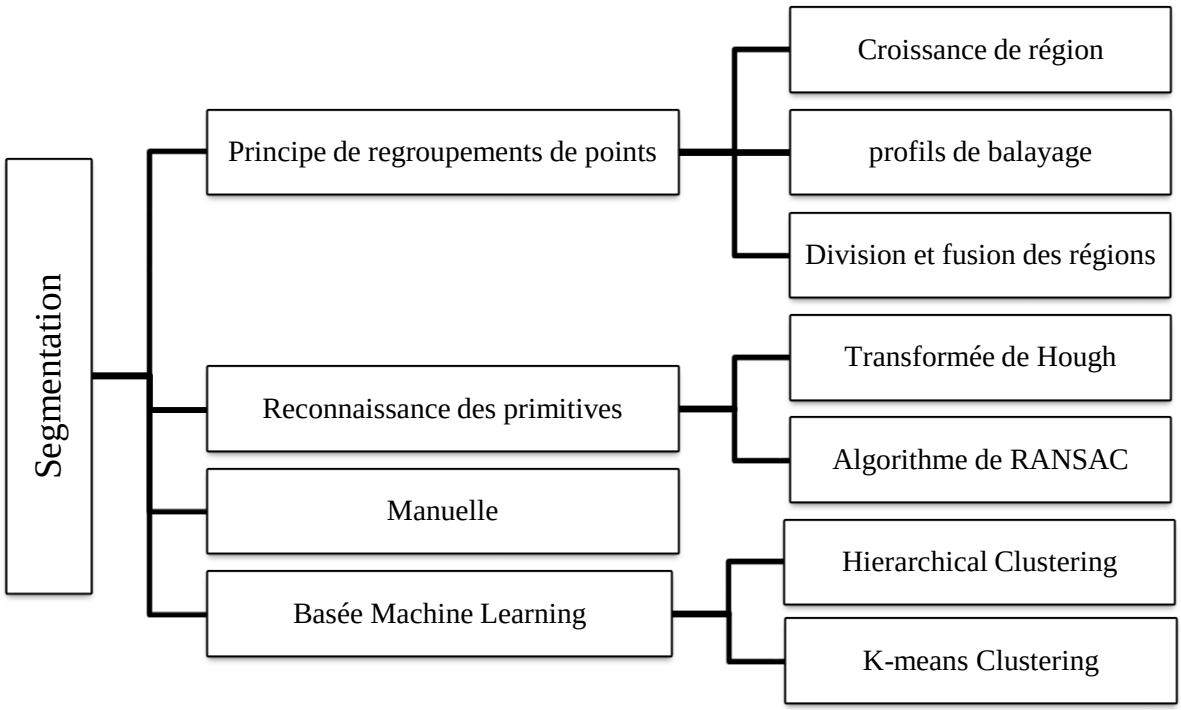


Figure 2. 8 *Techniques de segmentation (Grilli et al., 2017)*

La segmentation manuelle reste une étape chronophage et ardue qui dépend fortement des compétences de l'opérateur et son rapidité. Ainsi, le recours à des méthodes d'apprentissage automatique et d'apprentissage approfondie (Poux and Billen, 2019), ainsi qu'aux méthodes automatiques citées dans la *Figure 2. 8* s'avéré nécessaire.

Dans l'annexe 1 sont détaillés les outils de segmentation présents sur le logiciel de traitement des nuages de points CloudCompare à savoir RANSAC plugin, CSF(Cloth Simulation Filter), Scalar Field Filter, Connected-component labeling qui est un logiciel open source développé sous licence GNU GPL lors de la thèse de Daniel Girardeau-Montaut par Telecom Paris Tech et R&D division of EDF, développé en 2003.

2.1.4 Classification

Elle consiste à catégoriser les points/segments en différentes classes telles que les routes, le sol, la végétation, etc. afin de leur affecter une certaine information sémantique. C'est ainsi que la classification des nuages de points est appelée généralement la segmentation sémantique. Cette opération est effectuée soit manuellement, soit automatiquement avec/ou sans intervention de l'opérateur.

En général, comme le cas de données 2D, la classification peut avoir comme unité de base le point, le segment ou des entités multiples qui regroupent les deux premiers. La classification des nuages de points peut être effectuée selon les trois approches suivantes (Grilli et al., 2017) :

- Une approche supervisée, où les classes sont prises d'un ensemble de données annotées, puis un modèle est formé à partir de l'entrainement sur une base de données annotée, afin de fournir une classification à l'ensemble par la suite.
- Une approche non supervisée, où les données sont automatiquement partitionnées en segments basés sur les paramètres dans l'algorithme manipulé par l'utilisateur. Dans ce cas, aucune annotation ou donnée d'entrainement n'est exigée mais le résultat pourrait ne pas être vraiment conforme aux attentes de l'utilisateur.
- Une approche interactive, où l'utilisateur est activement impliqué dans la boucle de segmentation / classification en guidant l'extraction de segments via des signaux de retour. Cela nécessite un grand effort du côté de l'utilisateur mais le résultat peut s'améliorer en fonction des commentaires de celui-ci.(Grilli et al., 2017).

2.2 Structuration

Les nuages de points gigantesques qui ne peuvent pas être charger complétement dans la mémoire d'un système informatique peuvent être gérés de différentes manières(Scheiblauer et al., 2014)

Une première méthode consiste à scinder le nuage de points en plusieurs nuages de points plus petits, qui peuvent ensuite être traités individuellement. Cette méthode est trop utile si le nuage de point s'étend sur une large zone et qu'une partie du nuage de point complet est nécessaire (*Figure 2. 9*).

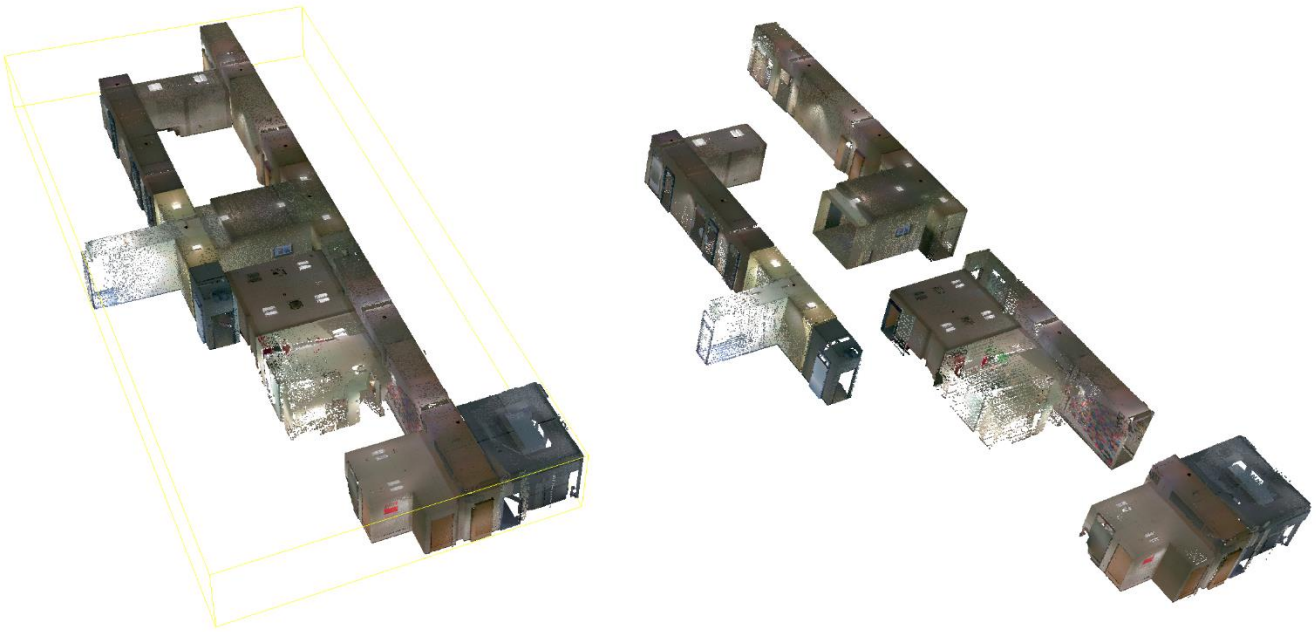


Figure 2. 9 *Nuage de point complet du RTWH (à gauche) scindé en plusieurs parties (à droite) pour faciliter le traitement et la visualisation*

Une autre stratégie consiste à procéder à un sous-échantillonnage du nuage de points, afin de réduire sa taille (*Figure 2. 10*). Cette méthode consiste à sous-échantillonner le nuage de points sur base du critère de la distance minimale entre deux points. Il doit être utilisé avec précaution pour créer une version moins haute résolution du nuage de points à des fins de visualisation ou pour accélérer le traitement des produits dérivés à une résolution plus grossière. Les points «en double» peuvent être supprimés par sous-échantillonnage.

Figure 2. 10 *Nuage de points sous-échantillonné avec distances minimales (dm) différentes. De gauche à droite, le nuage original (**264 448 295 points**), dm= 0.05cm (**257 491**), dm=0.10cm (**65 296**), dm= 0.15cm (**29 229**)*

Bien que ces deux méthodes peuvent être trop utiles pour voir le nuage de points, il y a l'inconvénient de perte de l'information et de réduction de la densité du nuage de points (qui peut être exigée pour la dérivation de certains produits comme les coupes et les plans d'étages) pour la méthode de sous-échantillonnage, et l'impossibilité de visualiser le nuage de points en entier pour la méthode de subdivision.

En plus des méthodes mentionnées précédemment, il est également possible de compresser les attributs des points, par exemple, la position ou la couleur. Les systèmes de compression avec perte sont les plus efficaces où certaines informations étant sacrifiées afin d'augmenter le taux de compression. Bien que cette option soit viable pour la visualisation des nuages de points, elle n'est généralement pas efficace si les nuages de points doivent également être édités, car le maintien du schéma de compression pendant les opérations d'édition peut devenir une tâche complexe et exigeante en calculs. Les points sont souvent compressés en groupes, de sorte qu'ils puissent devenir dépendants de leurs voisins. Ainsi, lors du changement d'attributs d'un point unique, les attributs des voisins doivent également être modifiés.

Une dernière façon de gérer les nuages de points massifs consiste à recourir à des algorithmes de mémoire externe (en anglais Out-Of-Core algorithmes). De tels algorithmes ne retiennent qu'une partie du jeu de données complet dans la mémoire principale (in-core) à tout moment, tout en changent de nouvelles données dans la mémoire principale si nécessaire, et permettent ainsi de visualiser, d'éditer et d'explorer ces jeux de données d'une manière interactive. Cette approche nécessite de structurer le nuage des points dans la structure appropriée (structure d'arbre Octree, *Figure 2. 11*) et de les charger à partir de la mémoire externe en cas de besoin. L'accès à la mémoire externe prend du temps, mais l'avantage est que l'ensemble de données complet est disponible en cas de besoin.

En effet, le choix de l'une ou de l'autre des méthodes dépend de la tâche à accomplir. L'approche la plus polyvalente semble être l'algorithme Out-of-core, car il n'est pas nécessaire de modifier les données d'origine et lorsqu'on utilise une structure de données appropriée (Octree), il est possible de modifier le jeu de données complet si nécessaire. C'est donc cette méthode qui sera utilisé par les algorithmes et les structures de données développés dans le présent travail.

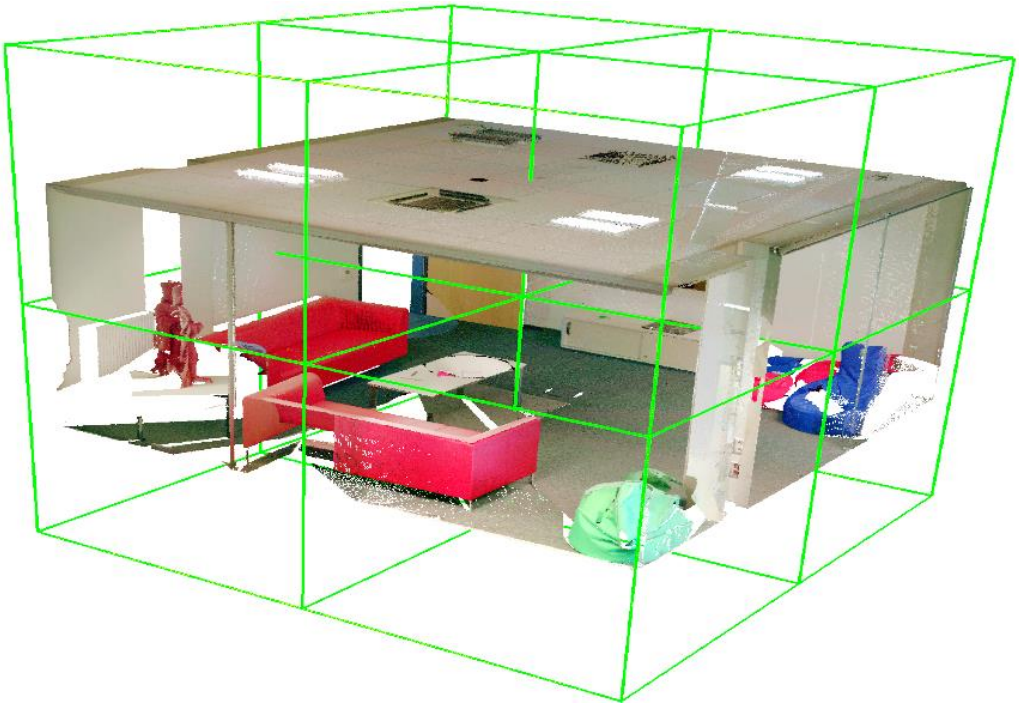


Figure 2. 11 *Un nuage de point du RTWH partitionné en structure Octree*

2.2.1 Out-of-core Algorithmes

La gestion des données massives nécessite des algorithmes out-of-core qui permettent de visualiser, d'éditer et de ne traiter qu'un sous-ensemble de données. La plupart des méthodes utilisent des variantes de structures hiérarchiques de partitionnement de l'espace, également appelées structures multi résolution, telles que kd-trees, octrees ou quadtrees. Elles peuplent tous les nœuds avec des données représentant le modèle d'origine à différentes résolutions. Certaines de ces méthodes redistribuent les données de points d'origine dans la structure hiérarchique, tandis que d'autres stockent les données d'origine uniquement dans feuilles et les moyennes sous-échantillonnées dans les nœuds internes.

Le premier système multi résolution permettant le rendu d'une centaine millions de points, proposé par (Rusinkiewicz and Levoy , 2005) est appelé QSplat. La méthode QSplat crée une hiérarchie d'une sphère de délimitation à partir d'un nuage de points ou d'un maillage pouvant être parcouru pour créer un rendu du modèle progressivement plus affiné.

Une autre structure multi-résolution appelé nuage de point en couches (en anglais Layred Point Cloud, LPC), permettant de réduire d'une manière significative le coût de la traversée coté CPU, et exploite les compétences du GPU dans le rendu d'une centaine de primitives géométriques en parallèle. Le coût de la traversée est réduit car il est seulement nécessaire

de descendre à l'un des sous-échantillons qui couvrent des zones plus vastes, et non à des points individuels qui couvrent de petites zones. (Gobbetti and Marton, 2004)

Une autre structure pour l'édition des nuages de points massifs est la structure octree imbriquée modifiable (MNO : Modifiable Nested Octree) (Scheiblauer and Wimmer, 2011). Il est détaillée dans la section suivante parce qu'elle est le point de départ de la structure Potree utilisée dans ce travail.

2.2.2 Structure Octree imbriquée modifiable

La structure de données octree imbriquée modifiable (MNO : Modifiable Nested Octree) est une amélioration de la structure de données octree imbriquée, présentée par (Wimmer and Scheiblauer, 2006), permettant des opérations d'édition efficaces sur les points stockés à l'intérieur de la structure de données tout en permettant un rendu rapide des points.

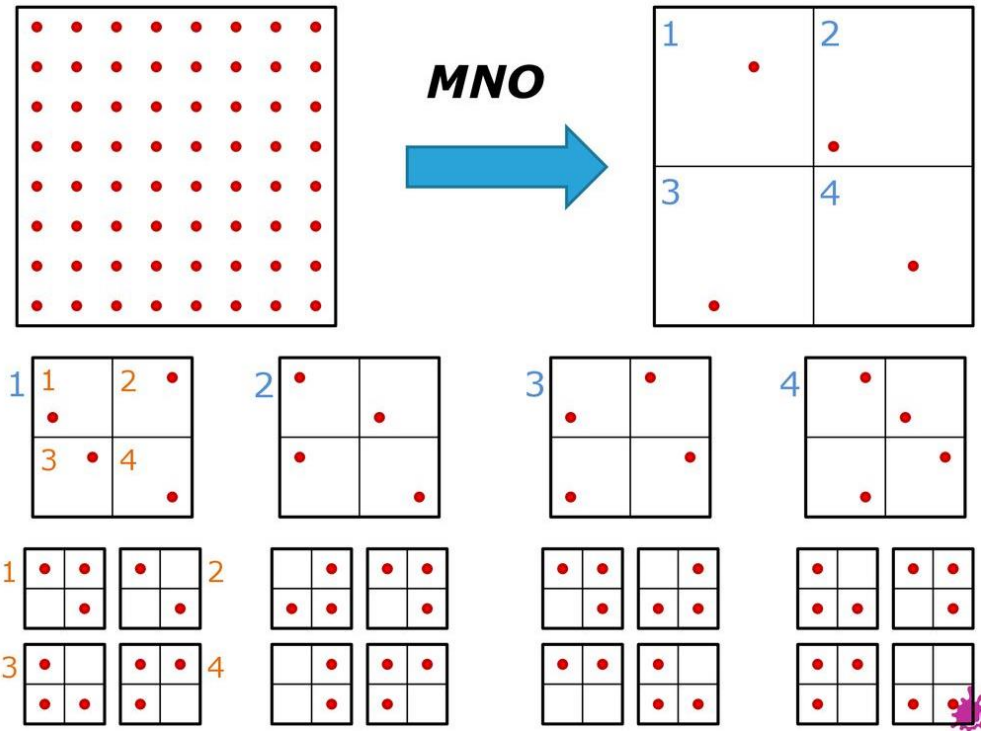


Figure 2. 12 *la structure MNO explicitée en 2D (Wimmer and Scheiblauer, 2006)*

En effet, nous décrivons brièvement cette structure d'octree imbriquée modifiable, sur laquelle est basée la structure octree de Potree. La structure MNO stocke des sous-échantillons de nuage de point d'origine dans chaque nœud. Les nœuds en bas niveau contiennent des sous-échantillons moins denses sur des volumes importants (*Figure 2. 12*). Donc, à chaque niveau, la taille d'un nœud se réduit tandis que la densité des points augmente (Schuetz, 2016).

Dans cette structure, chaque point du jeu de données d'origine est affecté à exactement un nœud octree. Cela signifie qu'aucun nouveau point ou doublons n'est créé et que la combinaison de tous les points de tous les nœuds donne le nuage de point d'origine.

2.2.3 Structure de Potree

La structure de Potree est une variante de la structure MNO avec une méthode de sous-échantillonnage différente et une partition de la hiérarchie en blocs plus petits. Pour la suite de ce travail et afin d'éviter toute confusion avec la structure MNO d'origine et parce que Potree n'offre pas de fonctionnalité pour modifier le nuage de points, nous l'appellerons « Structure octree de Potree».

La résolution d'un nœud est définie par la propriété espacement « Spacing », qui spécifie la distance minimale entre les points. L'espacement est initialement calculé pour le nœud racine, en fonction de la taille du « Bounding Box », puis divisé par deux à chaque niveau. Par exemple, un espacement de 1 mètre peut être utilisé pour un ensemble de données d'une étendue de 200 mètres dans chaque direction. Le nœud racine contiendra alors une version à basse résolution des données d'origine, dans laquelle chaque point est distant d'au moins 1 mètre du prochain. Les fils du nœud racine auront un espacement de 0,5 mètre, ce qui doublera effectivement la résolution, et ainsi de suite (Schuetz, 2016)

La valeur de l'espacement influence sur le nombre de points stockés par nœud, sur le nombre de nœuds nécessaires pour stocker tous les points et sur la profondeur de l'arbre. À part l'espacement, le nombre de points dans le nuage de points origine influence à son tour la profondeur de l'arbre et le nombre des nœuds qui résultent. Plus l'espacement est petit, moins on aura de nœuds et de niveaux et plus de points par nœuds et vice versa. Plus, on a de points en entrée plus en aura de nœuds en sortie pour le même espacement.

La hiérarchie de la structure Octree de Potree est stockée dans un fichier séparé des nœuds. L'utilisateur a besoin de cette hiérarchie pour trouver les nœuds à charger et qui doivent être visualisés. Afin de réduire le temps du chargement initial pour les grands nuages de points dont seulement le fichier de la hiérarchie peut plusieurs centaines de Mégabytes, la hiérarchie est subdivisée en plusieurs fichiers contenant seulement une partie de la hiérarchie selon un pas prédéfini *HierarchyStepSize*. La *Figure 2. 13* montre un exemple de hiérarchie d'octree multi résolution divisée en parties de 2 niveaux chacun.

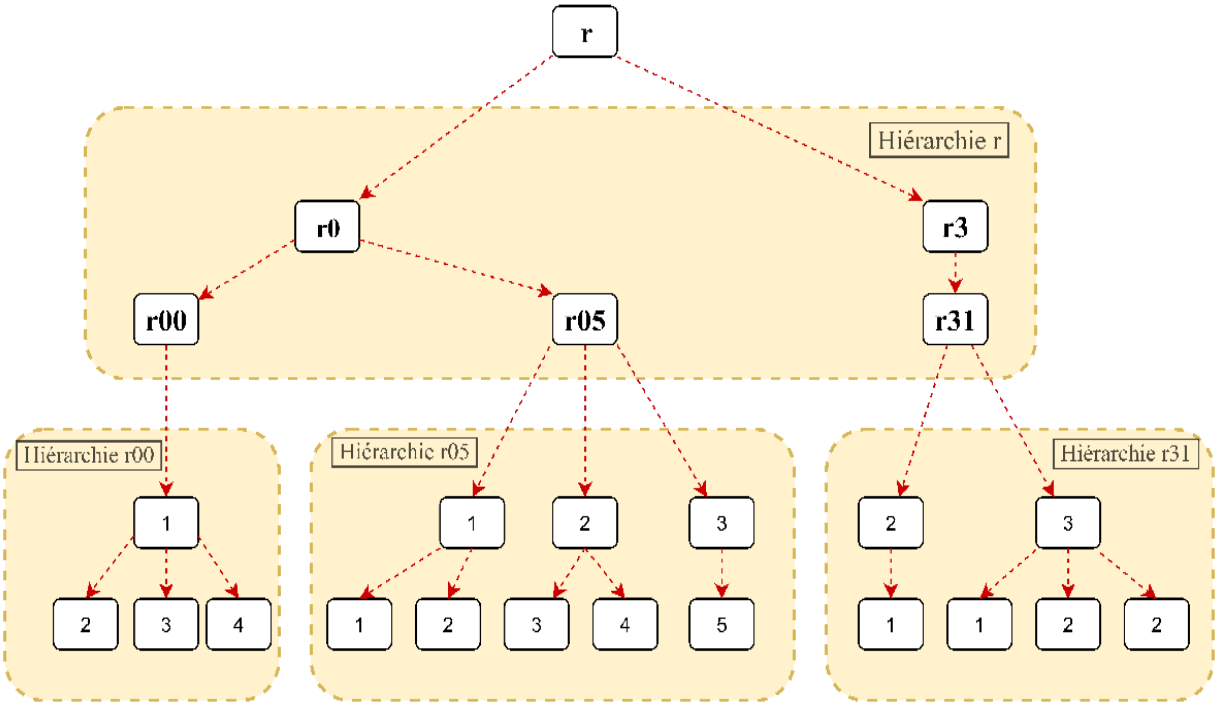


Figure 2. 13 *La hiérarchie d'octree est partitionnée en lots avec HierarchyStepSize égale à 2 dans cet exemple*

Le processus de création prend en entrée un ou plusieurs nuages de points, les partitionne en Octree et stocke le résultat un système de fichiers avec un fichier pour chaque nœud. Le processus de conversion suit ces règles :

1. Initialement, l'Octree est constitué d'un seul nœud racine, qui sert également de feuille à cet endroit.
2. Les points sont ajoutés, un par un, au nœud racine.
3. Les nœuds internes gardent un point si aucun autre point ne se trouve dans la distance minimale (espacement) et le transmettent à des « nœuds fils » autrement.
4. L'espacement est réduit de moitié à chaque niveau.
5. Les nœuds « feuilles » gardent tous les points en premier.
6. Si un certain seuil de points est atteint, un nœud « feuille » est développé. Il devient un nœud interne et ajoute tous les points stockés à lui-même, mais cette fois en suivant les règles du nœud interne. Les points avec une certaine distance minimale restent dans l'ancien nœud « feuille » et tous les autres points sont transmis aux nœuds « fils » nouvellement créés.
7. Les données sont régulièrement vidées sur le disque, par exemple chaque fois que 10 millions de points ont été traités.

8. Si aucun nœud n'a été touché depuis le vidage précédent, ses données seront supprimées de la mémoire lors du prochain vidage.
9. Si un point est sur le point d'être ajouté à un nœud qui a été supprimé de la mémoire, les données seront d'abord lues de nouveau sur le disque.

La conversion d'un nuage de point sous format d'Octree de Potree est faite à l'aide d'un convertisseur qui prend en entrée plusieurs types de format de nuage de points (xyz, las, laz, ply, ptx) appelé « PotreeConverter » développé par Markus Schütz.

Ce format permet une indexation spatiale des nuages de points et le stockage de plusieurs attributs. En plus des coordonnées et de la composante colorimétrique, il permet de stocker l'intensité, la classification, le numéro de retour et le numéro de la ligne de vol. Une description détaillée de cet outil est présentée dans l'annexe 2 de ce travail.

2.3 La réalité virtuelle : un nouveau mode de visualisation 3D

2.3.1 Définition

La Réalité Virtuelle (RV) représente une technique d'immersion dans un environnement en trois dimensions totalement numérique, généralement mais pas obligatoirement à l'aide d'un visiocasque ou dans un CAVE, qui consiste en un espace clos recouvert d'écrans où sont projetées des vues de l'environnement virtuel. Cet environnement peut s'inspirer d'éléments réels, en reproduisant des lieux, des œuvres d'arts, des paysages naturels via une modélisation en 3D, mais il peut également donner accès à un univers symbolique ou issu de l'imagination des concepteurs et graphistes 3D.

La réalité virtuelle est composée uniquement d'éléments virtuels, là où la réalité augmentée qui est un autre mode de visualisation, combine des images réelles avec des éléments synthétiques virtuels. Une difficulté singulière de la réalité augmentée est liée à la contrainte de devoir parfaitement positionner ces objets virtuels à l'intérieur des images réelles (on parle de technologies de « tracking » en anglais ou suivi de position).

Il existe par ailleurs la notion d'un continuum (*Figure 2. 14*) entre le virtuel et le réel et donc entre les représentations dans la réalité (environnement réel) et en réalité virtuelle (environnement virtuel) appelé le Continuum de Milgram et Kishino. L'appellation « Réalité

Mixte » englobe alors l'ensemble des représentations possibles mêlant éléments réels et virtuels de différentes manières.

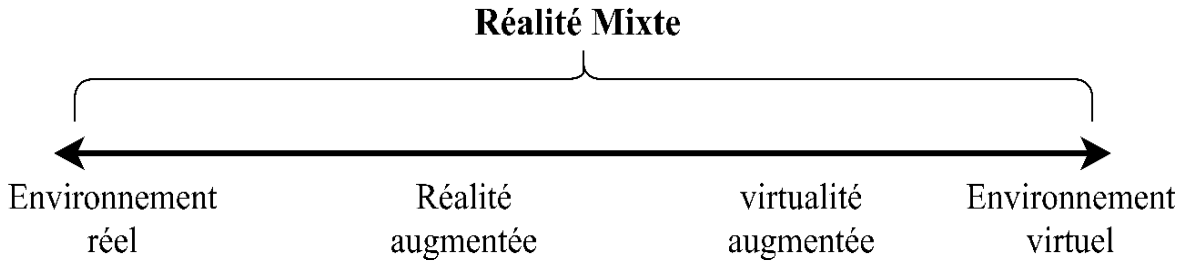


Figure 2. 14 *Le continuum réel-virtuel de Paul Milgram et Fumio Kishino (1994)*

Un dispositif de réalité virtuelle implique le plus souvent un environnement 3D synthétisé en temps-réel par un ordinateur avec lequel l'utilisateur peut interagir et dans lequel il peut s'immerger par le biais de retours sensoriels (le plus souvent visuel et stéréoscopique, mais également auditif et/ou haptique c'est-à-dire sur le canal du toucher).

2.3.2 Principe de fonctionnement

Au cours des dernières années, les dispositifs de réalité virtuelle (par exemple, Oculus Rift S[5] dans la figure ci-dessous, HTC Vive) sont apparus et permettent de nouvelles façons de présenter le contenu numérique 3D destiné aux différents usagers. Les éléments principaux d'un casque RV sont les suivants : un écran qui va projeter des images remplissant le champ de vision de l'utilisateur ainsi que des lentilles pour voir cet écran à distance raisonnable, même s'il se situe près de nos yeux. Pour relier ce casque à un ordinateur, 3 câbles sont nécessaires : un câble USB pour communiquer avec le PC, un câble pour acheminer l'image dans le casque (câble HDMI) et un câble d'alimentation. À noter que les casques RV mobiles n'ont pas besoin de ces câbles.

[5] https://www.oculus.com/

Figure 2. 15 *Casque de RV, à droite l'oculus rift, à gauche sa version mobile Oculus Quest*

Ces casques fonctionnent par stéréoscopie, technique grâce à laquelle le cerveau perçoit un relief. Ils reconstituent une image à partir des deux images planes et différentes perçues par chaque œil. Ces images stéréoscopiques, générées à l'aide d'un ordinateur, sont créées par deux lentilles situées en face de chaque œil. Elles sont ensuite envoyées dans le dispositif d'affichage du casque qui va les afficher en 3D. Le logiciel doit prendre en compte la position de l'utilisateur pour afficher les éléments au bon moment.

Le champ de vision devra également être assez proche de celui d'un humain (180°) pour ressembler à la réalité. Le champ de vision disponible sur la grande majorité des modèles récents de casque est d'environ 110°. Il peut descendre en dessous des 110 degrés sur les casques mobiles. De plus, pour assurer une immersion parfaite, le casque doit afficher au minimum 90 images par seconde. Il peut même monter à 120 FPS, Plus le nombre d'images est élevée, plus l'immersion sera qualitative, et nécessitera des machines puissantes.

Tous les mouvements sont captés à l'aide de détecteurs de mouvements (*Figure 2. 16*). Ces détecteurs vont ainsi suivre le positionnement du casque et des accessoires à l'aide de l'infrarouge pour mieux retranscrire la position de l'utilisateur dans l'environnement de RV. Parmi les outils utilisés, on peut citer le gyroscope qui permet de détecter les inclinaisons, l'accéléromètre pour détecter où est situé le sol. À noter que ces capteurs ne sont pas présents sur tous les casques, ce qui va influer sur la qualité de l'immersion.

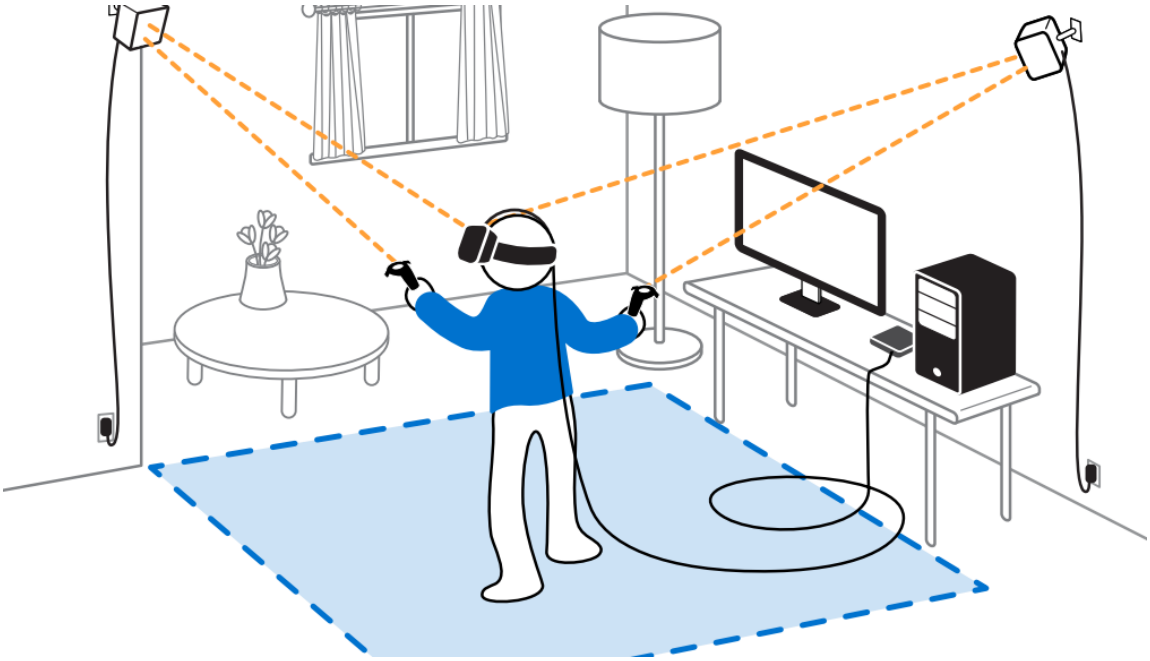


Figure 2. 16 *Casque de RV avec les capteurs de mouvement, les manettes et le casque, Source: octopusrift.com/setup-your-room-for-vr/*

C'est grâce à un suivi positionnel que les casques parviennent à adapter l'image en temps réel et donner une illusion d'immersion. Tandis que la plupart des casques se contentent de capter les mouvements de tête, les plus évolués captent le déplacement des pupilles, ou les changements d'altitude de la tête ou des mains afin de rapprocher un peu plus encore l'expérience de la réalité. À noter que les manettes sont toutes aussi importantes car elles retranscrivent les mains de l'utilisateur au sein de l'expérience RV.

En ce qui concerne l'ouïe, c'est un son stéréophonique qui s'en occupe. Ce son est obtenu grâce à deux haut-parleurs (gauche et droit) ce qui permet à l'utilisateur d'entendre les sons comme dans la nature. Par ailleurs, il va être plus immergé au sein de l'expérience RV. Les dispositifs de réalité virtuelle basiques se servent donc de la vue et de l'ouïe de son utilisateur. Enfin, le principal élément à prendre en compte est l'interaction. En effet, elle permet d'immerger complètement un utilisateur dans le monde de la RV.

La simulation d'environnements virtuels par un ordinateur est utile pour de nombreuses applications. Ces environnements ne sont pas limités par les contraintes physiques du monde réel, et peuvent prendre de nombreuses formes. Un environnement virtuel 3D est alors une représentation d'un ensemble d'objets virtuels avec lesquels des interactions sont possibles. Les interactions possibles dépendent de l'application (Benejean, 2013)

Il existe trois types d'interactions possibles dans un environnement 3D : la navigation pour le déplacement du point de vue de l'utilisateur, la sélection et manipulation pour la détermination et l'interaction avec la cible et enfin le contrôle du système pour le changement des propriétés de l'environnement 3D (Bowman et al., 2001)

Cependant, le rendu en réalité virtuelle pose des problèmes supplémentaires aux systèmes de rendu de nuages de points 3D qui sont généralement adaptés pour des applications non immersives. Premièrement, chaque scène doit être rendue simultanément sur deux écrans pour générer une image stéréoscopique. Deuxièmement, les artefacts visuels (encombrement visuel, trous entre les points voisins) ont tendance à être plus visibles lorsque vus dans le casque RV et peut facilement casser l'immersion. Troisièmement, pour prévenir les sensations de Cybersickness, les appareils de RV tels qu'Oculus Rift ou HTC Vive fonctionnent à 90 Hz. Cela nécessite que les images soient rendues à des taux nettement plus élevés par rapport aux applications non immersives (c'est-à-dire où 30 à 60 FPS sont typiques).

2.3.3 Applications en RV

Il ne fait aucun doute que la réalité virtuelle a suscité beaucoup d'intérêt ces dernières années. En tant que nouveau paradigme d'interface utilisateur, il offre de grands avantages dans de nombreux domaines d'application. Il fournit un moyen simple, puissant et intuitif d'interaction homme-machine. L'utilisateur peut regarder et manipuler l'environnement simulé de la même manière que nous agissons dans le monde réel, sans avoir besoin de savoir comment fonctionne l'interface utilisateur compliquée. Par conséquent, de nombreuses applications telles que les simulateurs de vol, les procédures d'architecture ou les systèmes de visualisation de données ont été développées rapidement. Plus tard, la réalité virtuelle a été appliquée en tant que support de télé opération et de collaboration, et bien sûr dans le domaine du divertissement(Mazuryk and Gervautz, 1996).

D'autres types d'applications sont résumés ci-dessous :

1. *Éducation*

Côté Géographie, grâce à l'application « Google Earth VR », les élèves ont désormais la possibilité de visiter la planète entière. Rabat, Les atlas, El Maamoura ou encore Merzouga pour des visites insolites qui en émerveilleront plus d'un. Cet outil permet d'apporter le

support visuel nécessaire à la compréhension d'un cours de géographie. Mais il permet également aux élèves de réaliser une « sortie scolaire » virtuelle. En effet, toutes les écoles n'ont pas les moyens financiers pour réaliser des voyages scolaires. La RV semble se présenter comme une solution alternative abordable.

2. *Histoire et archéologie*

Pour les applications d'Histoire, il est possible de s'immerger au sein de sites historiques endommagés ou disparus grâce à des reconstitutions virtuelles comme la Rome d'il y a 2000 ans. Mais également de se plonger au cœur d'évènements historiques comme le débarquement allié en Normandie. « Unimersiv », créateur d'applications RV éducatives a imaginé la visite en RV de Stonehenge, le Titanic, l'Acropolis d'Athènes.

3. *Test de réaction lors d'entretiens d'embauche*

Le recrutement est une étape majeure pour l'entreprise. En effet, une erreur de candidat peut rapidement avoir d'importantes répercussions financières et organisationnelles. Pour remédier à cela, les recruteurs ont pour habitude de tester les candidats lors des entretiens (tests de personnalité, tests techniques…). La réalité virtuelle intervient désormais en complément de tous les autres tests.

4. *En médecine*

Les formations RV se démocratisent de plus en plus. En voici quelques exemples :
Une start-up bordelaise « SimforHealth » a imaginé une formation médicale en RV. À destination des personnels hospitaliers, elle a pour but d'entraîner à la pratique chirurgicale afin qu'ils apprennent les bons gestes.

Conclusion

Plusieurs solutions et structures de données 3D pour la visualisation des nuages de points massifs en réalité virtuelle ont été développé mais le manque d'information sémantique et l'optimisation du rendu reste sujette d'innovation et d'amélioration. Le système de rendu proposé dans le présent travail vise une optimisation en ce qui concerne ces deux objectifs contradictoires : une qualité visuelle améliorée des artefacts de rendu et des performances de rendu accrue.

CHAPITRE III
SEGMENTATION ET CLASSIFICATION DES NUAGES DE POINTS

Introduction

Pour atteindre les objectifs fixés par ce travail, une méthodologie propre à ce mémoire a été suivie. Dans cette section, nous présentons les étapes de cette méthodologie, en commençant par la segmentation des différents nuages de points, leurs classifications sur Cloud Compare et leur structuration sur PotreeConverter. Dans chaque étape, les résultats intermédiaires sont présentés.

3.1 Organigramme méthodologique

Dans le but d'atteindre les objectifs fixés par ce travail dans la classification et l'intégration des nuages de points dans un environnement de réalité virtuelle nous avons suivi une méthodologie de quatre étapes. La première est liée à l'acquisition des données par lasergrammétrie. La deuxième étape consiste dans la segmentation et la classification de plusieurs nuages de points pour leur ajouter de l'information sémantique (*Figure 3. 1*). Une troisième étape porte sur la structuration des nuages de points selon le format Octree de Potree à l'aide de PotreeConverter. La quatrième étape s'agit de l'implémentation des scripts pour la lecture et la visualisation du nuage de points en structure MNO dans un dispositif de réalité virtuelle.

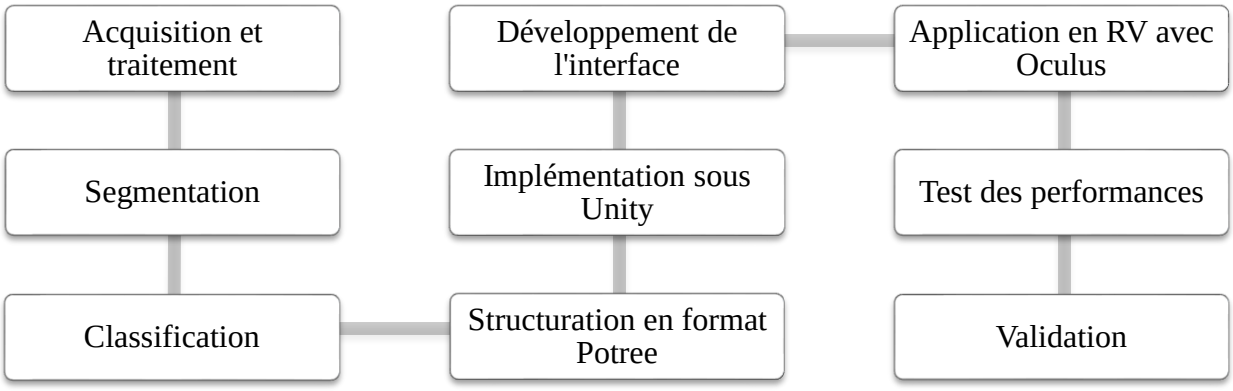


Figure 3. 1 *Méthodologie générale suivie dans ce travail*

La majore partie de cette méthodologie consiste au développement des scripts en C# pour la gestion de nuages de points structurés en format dans sa version améliorée pour intégrer de nouveaux attributs.

3.2 Acquisition et traitement

Pour avoir les données de base qui forment le nuage de points de départ de ce travail, nous avons effectué une mission d'acquisition par lasergrammétrie. La mission est effectuée pour lever une partie du château de Jehay avec un scanner Leica P30 (*Figure 3. 2*). Nous avons effectué une dizaine de scans en parallèle avec le levé des cibles standards rondes par station totale (*Figure 3. 2*). Ces cibles ont constitué les points essentiels pour la consolidation des différents scans dans le logiciel « Cyclone » de Leica.

Figure 3. 2 *Scanner laser Leica P30 à gauche, et station totale à droite pour le levé*

Après le suivi des étapes de prétraitement cités dans la section « 2.1 Acquisition et traitement des nuages de points », nous avons obtenu un nuage de points prêt à être exploité dans la suite du travail.

La mission n'est faite que sur une partie du château, nous avons intégré le résultat de notre mission avec les nuages de points des missions de scans déjà réalisés dans les années passées. Le résultat été un nuage de points du château entier que nous avons exploité dans la suite du travail. En plus de cela d'autres nuages de points déjà acquis sont exploités.

3.3 Segmentation

Dans cette étape, nous avons procédé à une segmentation de plusieurs nuages de points (*Tableau 3. 1*) avec des nombres de points différents partant d'une centaine de millions à une dizaine de milliards, acquises par les différents capteurs dont le scanner laser et le Matterport[6], et avec des attributs différents (avec composante RGB ou non, et avec ou sans intensité). Le but de cette diversité consiste à tester l'influence de l'ajout ou du manque des informations (RGB, intensité) sur la difficulté de la segmentation surtout lorsque cette opération est assistée par un opérateur. Le deuxième but est de tester la solution développée sur plusieurs nuages de points avec des tailles variables allant du petit jusqu'au grand nombre de points par nuage. Le troisième but est de créer des « Data sets » qui peuvent être exploités comme des données d'entrainement des algorithmes d'apprentissage approfondi pour la segmentation sémantique développé au sein de l'unité de géomatique par le chercheur Florent Poux[7].

En général, les résultats de cette segmentation seront très utiles pour toute la communauté des chercheurs vu le manque énorme dans les données d'entrainement et aussi la complexité de cette tâche qui reste chronophage et lourde et qui doit être exacte par ce qu'elle sera une donnée de référence afin de constituer les données d'entrainement des modèles d'apprentissage automatique.

Tableau 3. 1 *Caractéristiques des nuages de points segmentés*

Nom	**Nombre de points**	**Attribut**	**Capteur**	**Taille Go**
CHÂTEAU_JEHAY	2.300.247.428	RGB, intensité	scanner	69.636
PCID10_RTWH_Exterior	312.710.687	RGB, intensité	scanner	4,907
PCID11_RTWH_CHAIR	259.101.028	RGB, intensité	scanner	4,807
PCID2_ULG_B5a	115.190.236	RGB, intensité	Scanner	3,824
PCID8_NAAVIS_1	44.847.540	RGB	scanner	0,657
PCID6_REVO	53.800.194	Nul	Matterport	0,630
PCID9_NAAVIS_2	4.244.416	RGB	scanner	0,062

[6] https://matterport.com/
[7] http://pointcloudproject.com/

La segmentation est faite sous l'outil open source CloudCompare dont un récapitulatif détaillé sur les fonctionnalités utilisés est associé au travail dans l'annexe 1. Afin de maîtriser cette étape, nous avons commencé par la segmentation du nuage de point en pièces (Pour les cas de bâtiments), puis la segmentation de chaque pièce en segments significatifs plus petits (les murs, les portes, les fenêtres, les chaises, les bureaux, les placards, etc.) (*Figure 3. 3*)

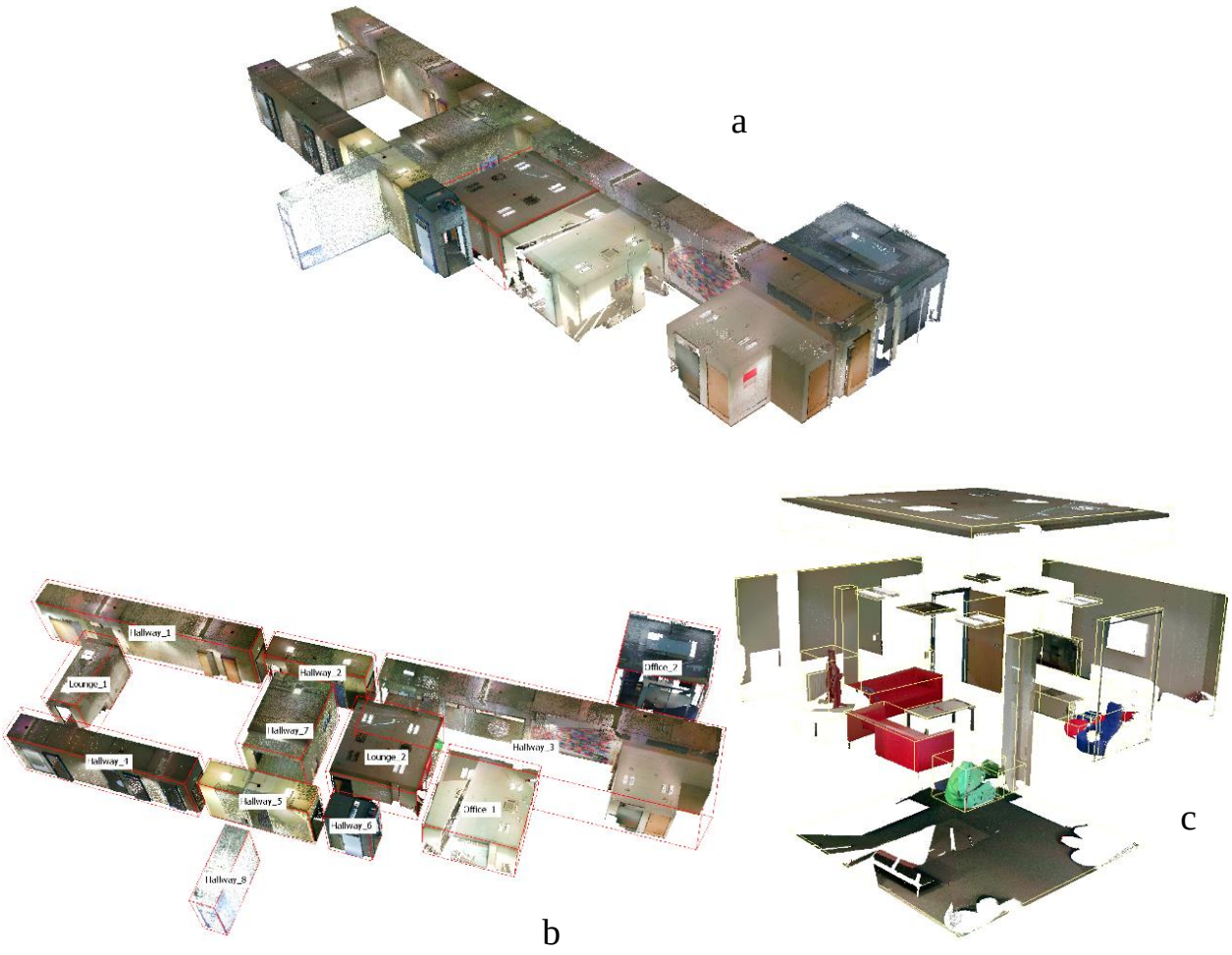


Figure 3. 3 *Nuage de point brut (a) ; Le nuage de point segmenté en pièce (b) ; Pièce segmentée en plusieurs éléments plus petits (c)*

La majorité du travail de segmentation est effectué manuellement et semi automatiquement avec les outils de sélection disponibles sur le logiciel, assisté par les plugins automatiques suivants :

- **RANSAC Shape Detection** : basé sur une interface simple avec l'algorithme de détection automatique des formes comme proposé par (Wahl and Klein, 2007) de l'université de

Bonn. Il permet de détecter les formes géométriques, des plans, des sphères, des cylindres, des cônes, et des tores. Dans notre cas des nuages de points, cet algorithme n'est utilisé que dans les cas de bâtiments afin d'extraire les murs, le toit et les formes géométriques bien précises.

- **CANUPO**[8] : est un moyen simple mais efficace pour classer automatiquement un nuage de points. Il permet de créer des classificateurs (en les formant sur de petits échantillons) et / ou d'appliquer un classificateur à la fois sur un nuage de points afin de le séparer en deux sous-ensembles. Il génère également une valeur de confiance de classification pour chaque point afin d'identifier rapidement les cas qui forment des problèmes de classification ou une fausse classification (généralement aux limites des classes). Il est trop utile pour les zones extérieures, où il y a présence du terrain naturel et des bâtiments et objets qui s'élèvent en dessus.
- **CSF** : (en anglais **C**loth **S**imulation **F**ilter) est un outil qui sert à séparer les points au sol du reste du nuages de points, basé sur le filtre de simulation de tissu développé par (Zhang et al., 2016)
- **Label Connected Components** : Cet outil segmente le ou les nuages sélectionnés en parties plus petites séparées par une distance minimale. En effet, ce plugin s'avère utile pour isoler des objets regroupés après séparation des murs, du toit, des sols des bâtiments comme par exemple les chaises, les bureaux, les lampes, etc.

À la fin de cette étape de segmentation, une dizaine à une centaine de segments est séparée dans chaque « Data set », mais ne contient pas encore l'information sémantique sur ces segments. La sémantisation fera l'objet de l'étape à venir qui consiste à faire une classification de tous les segments. Dans cette étape une approche descendante a été suivie, en commençant d'une petite échelle vers une grande échelle avec les plus petits détails du monde physique. Dans la prochaine étape une approche inverse sera faite afin d'avoir à la fin un nuage de points classé.

3.4 Classification

Elle s'agit de l'affectation de chaque segment créé à une classe précise. En premier, une liste de classification est établie avec des classes de l'intérieur (Indoor) et des classes de l'extérieur (Outdoor) (*Tableau 3. 2*). Ensuite, un nouveau champ scalaire avec une valeur

[8] https://nicolas.brodu.net/fr/recherche/canupo

constante est créé (en anglais Constant Scalar Field) nommé « Classification » et cela dans le but de respecter les spécifications du format d'export qui sera le format LAS[9]. Ce format de nuage de points support par défaut cet attribut. Un numéro de la classe est affecté à ce champ de classification crée.

Tableau 3. 2 *Extrait la table des classes, une liste complète des classes est en annexe 6.4*

Indoor		Outdoor	
Classe	Numéro	Classe	Numéro
0	Floor	40	LowVegetation
1	Ceiling	41	Humain
2	Wall	42	Pole
3	Beam	43	LightingPole
…	…	…	…
24	Poster	50	Roof

Enfin, tout objet du nuage de points possède une information de la classe qui lui correspond. Alors que pour un même nuage de point, il peut y avoir plusieurs segments qui appartiennent à la même classe (par exemple, la classe arbre contient une dizaine d'arbres), l'ajout d'un identifiant unique pour chaque segment de chaque classe s'avère nécessaire pour le caractériser (cet identifiant commence de 1 jusqu'à le nombre de segment qui existent dans la classe correspondante).

Une fois tous les segments (objets) classés et disposant de leurs identifiants uniques, ces objets sont assemblés pour former un seul nuage de points comme celui d'origine mais qui contient une information sémantique supplémentaire. Selon les besoins, d'autres champs comme l'intensité « Intensity » peuvent être ajoutés (comme notre cas), puis le nuage de points est exporté en format (. *LAS*) ou son format compressé (. *LAZ*).

[9] LAS est un format de fichier pour l'échange de données de nuages de points en 3D, les spécifications de ce format sont été développé et est maintenu comme spécification publique par l'American Society for Photogrammetry and Remote Sensing (ASPRS)

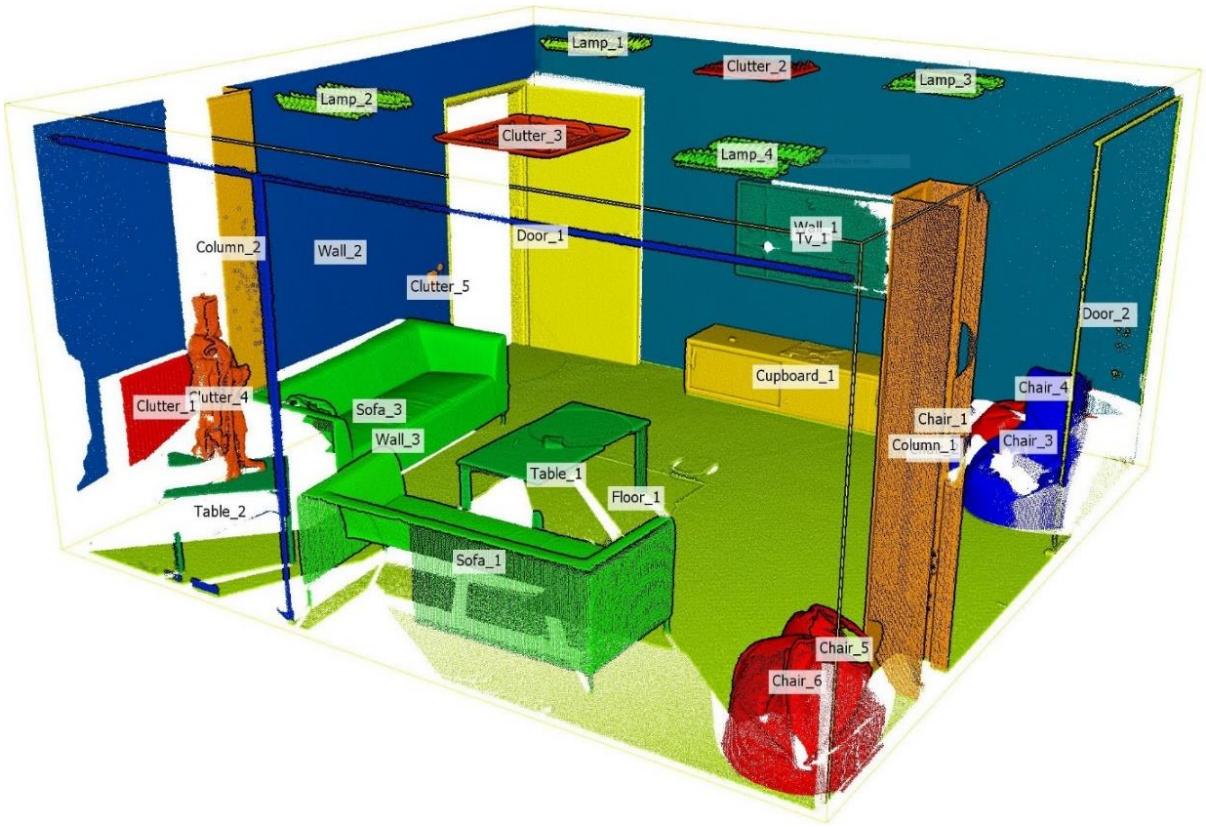
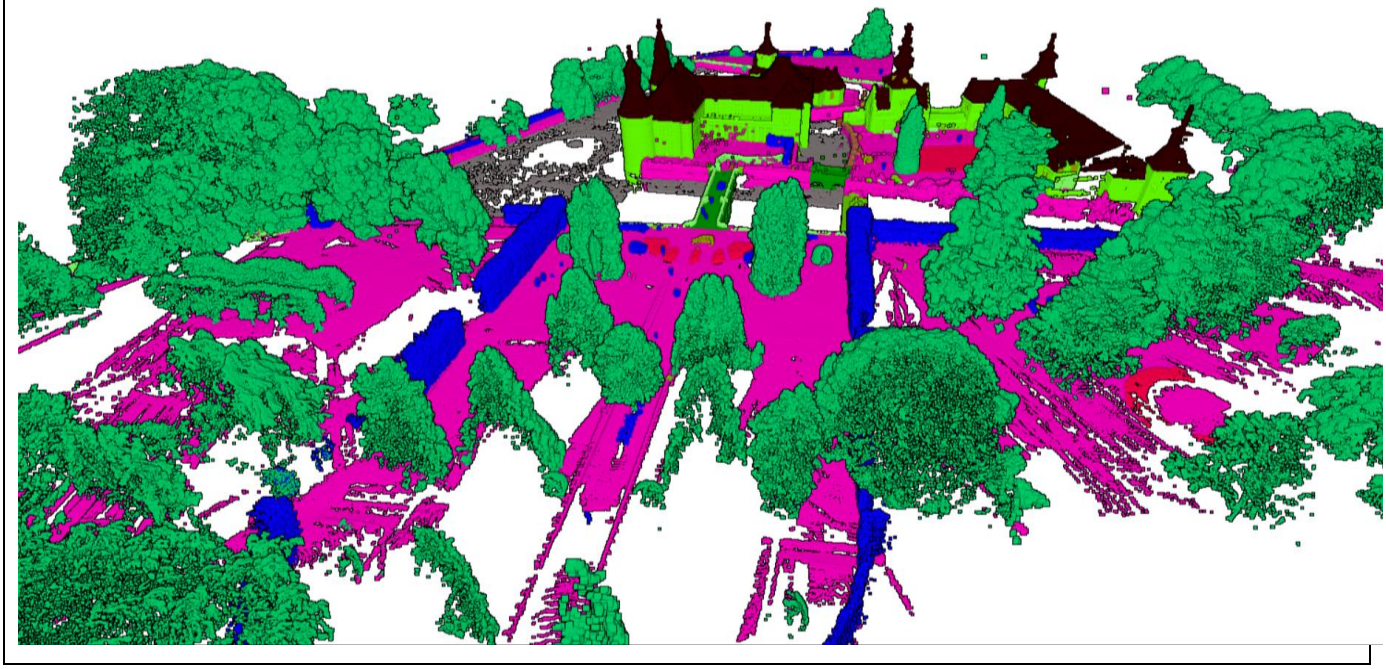


Figure 3. 4 *Exemple de pièce classée, chaque classe est affichée avec une couleur.*

La *Figure 3. 5* en dessous présente les différents « Data sets » après classification où chaque classe est attribuée une couleur choisie par l'utilisateur pour l'affichage.

a. Château de Jehay



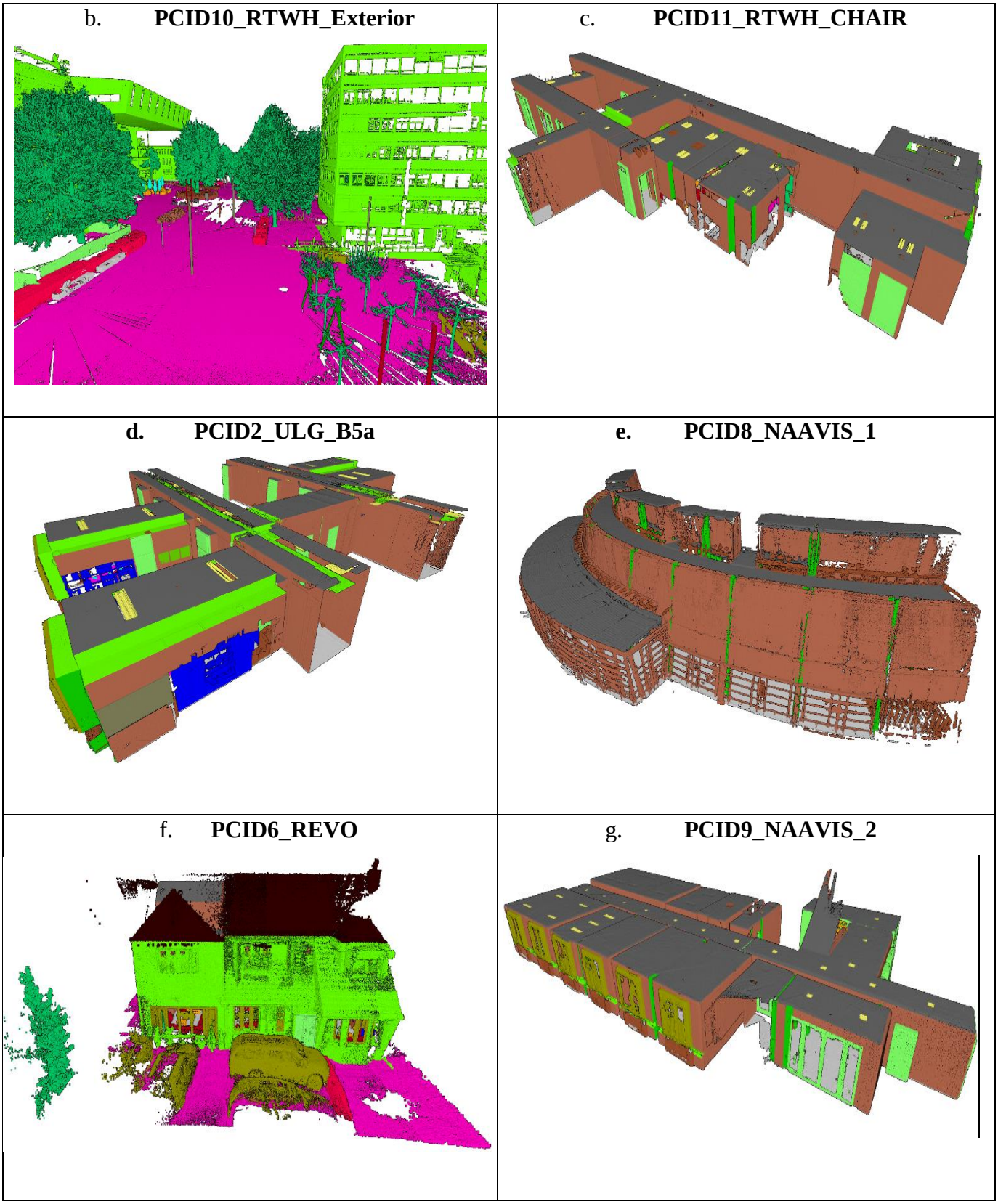


Figure 3. 5 *Résultats post-classification*

3.5 Structuration en Octree de Potree

La structuration a pour but de faire une indexation spatiale du nuage de points. En effet, ce dernier sera subdivisé selon plusieurs portions de l'espace (cubes) qui constituent les nœuds de l'arbre. Le nuage de points est subdivisé en huit parties de l'espace selon le « Bounding box » initial, avec un grand espacement et un niveau de détail faible. Ensuite, chaque partie est à son tour subdivisée de la même façon et l'espacement est réduit à moitié afin d'augmenter la densité. Ainsi, plus la profondeur de l'arbre augmente plus l'espacement diminue et le niveau de détail augmente.

Cette étape est faite avec l'outil « PotreeConverter » version 1.6 développé par Markus Schütz de l'université technique de Vienne. Il supporte plusieurs formats de nuage de points comme entrée (xyz, las, laz, ply, ptx) format LAS dans notre cas et plusieurs formats en sortie (.bin[10], .las, .laz) format binaire dans notre cas. L'utilisation de cet outil est faite par ligne commande avec plusieurs instructions spécifiques, plus de détails en annexe 2.

Pour avoir un nuage de points structuré en format Potree, et qui prend en considération les attributs voulus (classification, intensité) la commande suivante a été utilisée :

```
.\PotreeConverter.exe -i fichierEntree -o dossierSortie -p pageName
-a RGB INTENSITY CLASSIFICATION
```

À noter que les deux attributs liés à la position xyz et à la composante colorimétrique RGB sont pris en compte par défaut. La signification des instructions utilisées est comme suit:

```
-i : le fichier du nuage de points .las entrée
-o : le chemin de sortie du résultat
-p : le nom de page HTML associé aux nuages de points pour l'affichage.
-a : les attributs à spécifier (rgb, classification, intensité, source id)
--output-format : le format de sortie, le format par défaut est le .bin de Potree,
mais il peut être spécifie le .las et le .laz
```

En plus de ces instructions, il y en a d'autres plus avancées permettant de personnaliser la sortie, par exemple le choix de l'espacement (`-s`), le titre de la page web (`--title`), la projection utilisée en format proj4 (`--projection`), Bounding cube du nuage de points initial (`--aabb`), ainsi que d'autres détails attachés en annexe 2.

[10] Ce format binaire est spécifique à Potree.

Le dossier du nuage de points résultat (*Figure 3. 6*) contient 3 sous-dossiers dont :

- **Libs** : qui regroupe l'ensemble des librairies de JavaScript (Cesium, d3, jstree, openlayers3, plasio, Potree, proj4, three.js, etc.) utiles pour la visualisation du nuage de points sur le navigateur web.
- **Pointclouds** : ce dossier est le plus important pour notre travail, car il contient les nuages de points structurés en format Potree avec la hiérarchie liée dans des fichiers en format spécifique à Potree *(.hrc)*. il contient aussi un fichier *cloud.js* qui spécifie les métadonnées sur les nuages de points (version, projection, Bounding box, les attributs crées, l'espacement initial, l'échelle et le *HierarchyStepSize* comme spécifié dans la partie 2.2.3. Un autre fichier *sources.json* spécifie des informations sur le nuage de points original (son Bounding box, sa projection, son nom, et le nombre de points)
- **La page HTML** : pour affichage sur navigateur ou sur un serveur (spécifié dans la ligne de commande)

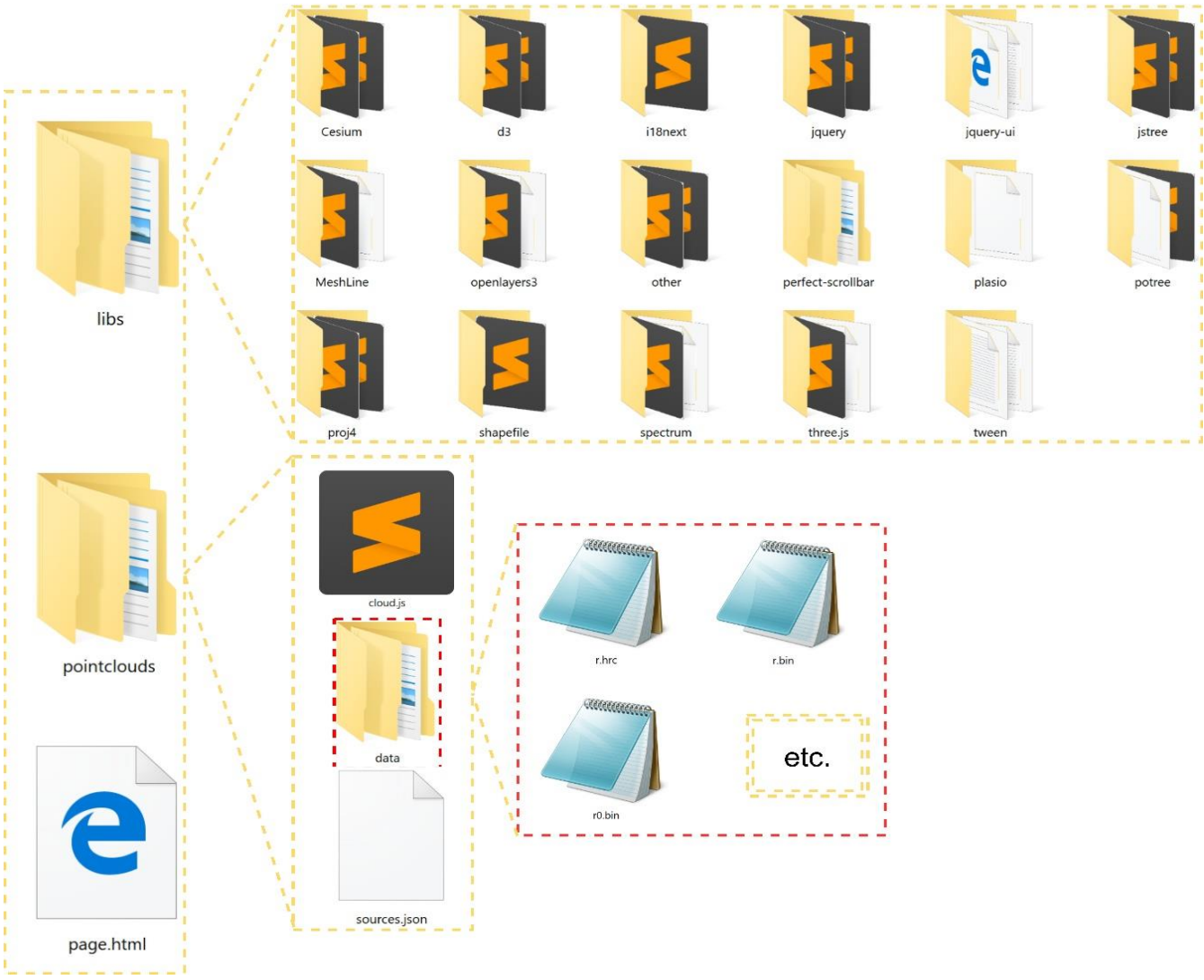


Figure 3. 6 *Structure des fichiers après conversion par PotreeConverter*

Conclusion

Pour atteindre le premier objectif de ce travail, qui consiste à un enrichissement sémantique du nuage de points, nous avons appliqué les différentes méthodes de segmentation manuelle, semi-automatique ou complétement automatique. Puis nous avons procédés à une classification basée segment. Les résultats de cette étape sont les différents nuages de points classifiés. Puis, nous avons procédé à une indexation spatiale des nuages de points en format octree de Potree pour préparer les données à l'étape prochaine de la méthodologie qui consiste à implémenter la solution sous Unity pour lire, et visualiser ces données structurées.

CHAPITRE IV
APPLICATION DE LA RALITE VIRTUELLE

Introduction

Une fois le nuage de points classé et structuré en format Octree de Potree, il sera utilisé pour visualisation dynamique et en temps réel, à travers l'implémentation de plusieurs scripts en C#. En premier, le résultat de la classification sera adapté et visualisé sous le Viewer WebGL de Potree. Puis nous procédons à l'implémentation de la solution sous le Game Engine unity. Ensuite, l'application est déployée pour la visualisation avec le casque RV « Oculus rift ». À la fin, des tests de performances sont réalisés sur base des critères objectifs comme la consommation en mémoire, et le nombre d'images par seconde. L'analyse des résultats s'avère nécessaire afin de savoir l'innovation et la qualité de la solution développée.

4.1 Visualisation avec Potree

Potree est un outil open source basé WebGL pour les nuages de points massifs. La partie développée en JavaScript sur Potree consiste en l'ajout de la classification selon les classes créées lors de la classification du nuage de points, ainsi que l'affectation des couleurs correspondantes à chaque classe, comme montré dans la *Figure 4. 1*.

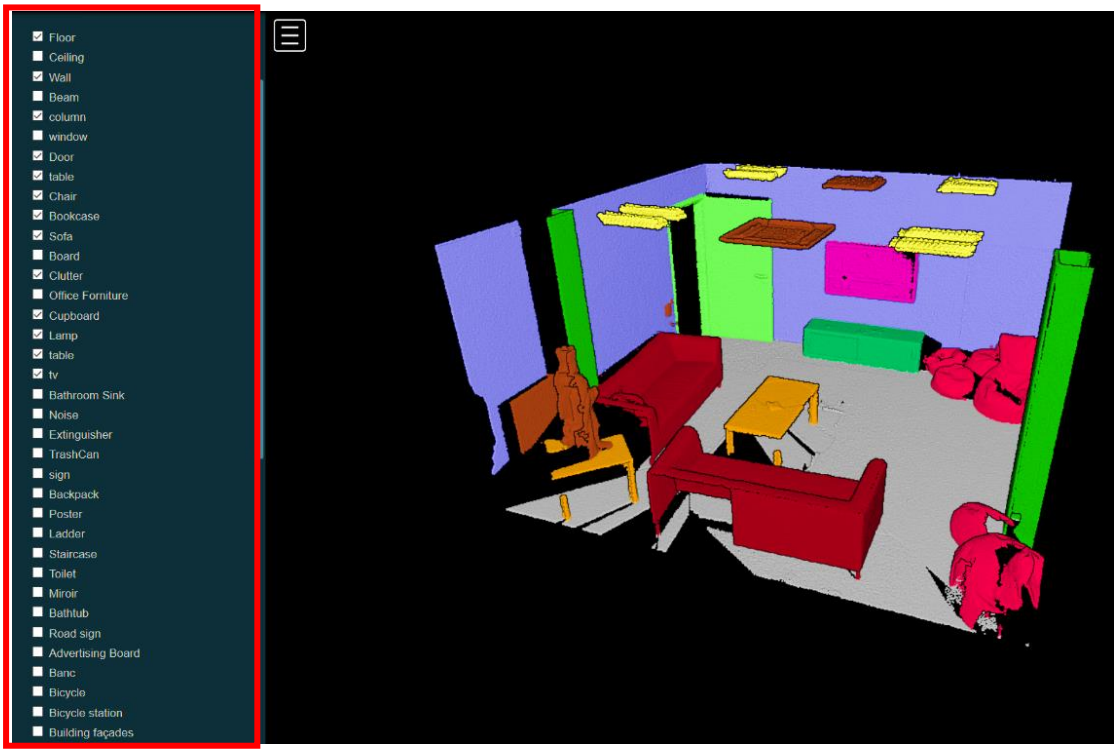


Figure 4. 1 *Visualisation d'un extrait de nuage de points sur Potree*

Pour ce faire, à chaque numéro de classe stocké dans le champ classification on affecte le nom de la classe à afficher, ainsi que son état de visibilité. Ci-dessous un extrait du code ajouté dans le code du *Potree.js*

```
This. Classifications = {
                0: {visible: true, name: ‘Floor’},
                1: {visible: true, name: ‘Ceiling’},
                2: {visible: true, name: ‘Wall’},
                .. : .. .. .. .. .. .. .. .. .. .. ..
                49: {visible: true, name: ‘Ground’},
                50: {visible: true, name: ‘Roof’}};
```

Par après, à chaque classe est affectée une couleur pour sa visualisation comme suit ;

```
Potree. Classification = {'DEFAULT': {
        0: new THREE.Vector4 (0.5, 0.5, 0.5, 1.0),
        1: new THREE.Vector4 (0.5, 0.5, 0.5, 1.0),
        2: new THREE.Vector4 (0.63, 0.32, 0.18, 1.0),
        .. : .. .. .. .. .. .. .. .. .. .. .. ..
        50: new THREE.Vector4 (0.60, 0.16, 0.0, 1.0),
        ‘DEFAULT’ : new THREE.Vector4 (0.3, 0.6, 0.6, 0.5)}} ;
```

Enfin, pour afficher les classes dans la barre de menu à gauche (*Figure 4. 1*) on ajoute des check box à la liste de classification avec le numéro et le nom de la classe, qui sont liés avec la première étape qui retourne l’état de la classe, si elle visible ou non.

```
Let elClassificationList = $('#classificationList');
                AddClassificationItem (0, 'Floor');
                AddClassificationItem (1, 'Ceiling');
                AddClassificationItem (2, 'Wall');
                .. : .. .. .. .. .. .. .. .. ..
                AddClassificationItem (49, 'Ground');
                AddClassificationItem (50, 'roof') ;
```

4.2 Chargement du nuage de points

Dans le but d'atteindre les objectifs fixés par ce travail pour la visualisation des nuages de points dans un environnement de réalité virtuel, nous avons entamé la deuxième étape d'implémentation de la solution. Ce système de rendu a été développé et implémenté en C# sous le Game Engine Unity version ***2018.4.1f1***, sur base des scripts développés dans le cadre de la thèse bachelor [11]de Simon Maximilian Fraiss (2017), de l'université technique de Vienne dont le sujet est intitulé « Rendering Large Points Cloud in Unity ».

Notre apport dans le cadre de cette recherche consiste à intégrer l'information sémantique sur le nuage de points, ce qui permettra de visualiser la classification dans un environnement virtuel, ainsi que l'ajout d'autres attributs comme l'intensité. Aussi, une interface utilisateur a été intégrée dans ce travail pour permettre à l'utilisateur une interaction avec le nuage de points, via laquelle il peut changer les paramètres d'affichage liés à la taille des points, leurs formes géométriques, et l'interpolation utilisée pour la visualisation. L'utilisateur peut facilement au travers cette interface visualiser ou cacher des classes spécifiques par son pointeur laser. Une contribution majeure de ce travail porte sur l'aspect visuel et consiste dans la création d'un Shader (en cg[12]) pour simuler l'effet de la lumière sur le nuage de points sans avoir besoin à un calcul des normales. Enfin, l'intégration du dispositif de réalité virtuelle (Oculus Rift) est faite, ainsi que le développement d'un mode de vol permettant à l'utilisateur de voler et se déplacer librement dans le nuage de points.

4.2.1 Plateforme d'implémentation : Unity

Unity[13] est un moteur de jeu Multiplateforme. Il support l'intégration de l'oculus Rift qui est un dispositif de RV développé par Facebook. La version utilisée est une version personnelle gratuite, ce qui a justifié le choix de cet outil.

Pour faciliter l'implémentation en cas de reprise du travail, l'approche suivie et les concepts généraux du travail seront présentés et à la fin le code développé en C# sera mis en open Access sur Git hub (lien en annexe 5).

[11] Disponible sur Git hub : https://github.com/SFraissTU/BA_PointCloud

[12] C for Graphics est un langage de haut niveau créé par NVIDIA pour programmer les vertex et les pixels shaders. Il est très similaire au langage HLSL de Microsoft.

[13] https://unity.com/

L'élément le plus important dans Unity est le graphe de la scène (*scene graph*), qui est une arborescence contenant ce que l'on appelle des objets de jeu (*Game Objects*). Un objet de jeu peut comporter plusieurs composants (*components*). Chaque objet de jeu a un composant de transformation (*Transform*) qui lui donne une position dans l'espace 3D. De plus, d'autres composants peuvent être ajoutés tels que des éclairages(*Light*), des caméras (*cameras*), des corps rigides (*rigid bodies*) pour simulations physiques ou des maillages. Les scripts peuvent également être attachés en tant que composant. Ils sont écrits en *C #* ou en *JavaScript* auparavant.

Pour ce travail, *C #* a été utilisé avec l'éditeur Visual studio 2017 lié à unity. Lors de la création d'un nouveau script, une nouvelle *classe C #* est créée, laquelle étend la classe *MonoBehaviour*. Une méthode de démarrage peut être implémentée (*void Start ()*), elle est appelée au démarrage du script (cela se produit dès que l'objet de jeu auquel le script est attaché apparaît dans la scène), ainsi qu'une méthode de mise à jour *(void update (), et void FixedUpdate()),* cette fonction est appelée lors du rendu de chaque image. Les objets de jeu peuvent être désactivés pour qu'ils existent toujours dans le graphe de la scène, mais ils ne sont pas mis à jour et ils n'ont aucun effet sur la scène (par exemple, les mailles attachées ne sont pas affichées).

Un élément important est le filtre à mailles (*Mesh Filter*). Avec un filtre de maillage, un maillage de polygones ou de points peut être défini. Un maillage est constitué de sommets avec leurs couleurs, leurs coordonnées de texture et leurs normales correspondantes. Ceux-ci peuvent être assemblés en polygones en fournissant des index et une topologie en maillage (Unity fournit les topologies *Triangles*, *Quads*, *Lines*, *LineStrip* et *Points*). Un maillage peut comporter jusqu'à ***65 000*** sommets(*Vertices*), ce nombre présentera une limitation lors du chargement des points.

Un filtre de maillage seul n'affiche pas encore de maillage. Un *Mesh Renderer* est également nécessaire pour cela. Dans un *Mesh Renderer*, les options de rendu peuvent être modifiées, telles que les paramètres d'ombre ou le matériau utilisé pour le rendu. Les matériaux sont dérivés de *shaders*, qui sont écrits dans une variante de *HLSL* [14] appelée *Cg*.

[14] HLSL (*High-Level Shader Language*) est le langage de programmation des pipelines des cartes graphiques 3D intégré à l'API Direct3D de Microsoft.

4.2.2 Chargement du nuage de points sous Unity

Les nuages de points sont stockés au format de fichier Potree (voir section 3.4) et ont été créés avec le convertisseur de Potree. Par conséquent, les points sont stockés dans une structure de données Octree, où chaque nœud de l'Octree contient un sous-ensemble de points. La hiérarchie est stockée dans des fichiers ***(.hrc)*** des différents des points eux-mêmes.

Dans le présent travail, nous avons intégré les attributs : classification, intensité, et source id. Vu que les points sont structurés en format binaire de Potree, la taille en byte de chaque point est variable selon les attributs pris en considération, pour une gestion dynamique de cette taille selon les différents attributs *Tableau 4. 1*. Une fonction de calcul ***ByteSize ()*** est créée sur base des attributs qui figurent dans le fichier ***cloud.js*** et cela selon les spécifications du format binaire de Potree qui permet de coder chaque attribut sur un nombre précis de bytes comme suivant :

Tableau 4. 1 *la taille en byte de chaque attribut codé sur le format binaire de Potree*

Attribut	**X**	**Y**	**Z**	**R**	**G**	**B**	Alpha	Intensité	Classification	Source id
Taille en byte	4	4	4	1	1	1	1	2	1	2

Cette fonction permette de calculer la taille en byte de chaque point (par exemple : pour un nuage de points contenant la position (12 bytes), RGB et alpha (4 bytes), intensité (2 bytes), et classification (1 bytes). Dans ce cas, chaque point est codé sur 19 bytes). Le but est de savoir le nombre des points par fichier sur base de la taille d'un point et la taille du fichier complet en bytes.

Un élément important à noter est qu'Unity stocke les composants de vecteurs sous forme de valeurs flottantes. Comme les points peuvent avoir des valeurs très élevées en tant que positions, il peut y avoir une perte de précision. Dans la mesure du possible, les valeurs sont stockées sous forme de valeurs doubles aussi longtemps que possible, mais toutes les valeurs doivent être converties en flottants avant de les transmettre au GPU. Le système implémenté offre la possibilité de déplacer le nuage de points vers l'origine à travers une option « move to centre » qui doit être coché avant le lancement de l'application, ce qui entraîne une perte de précision moins importante.

Au début du programme, la hiérarchie est chargée à partir des fichiers de hiérarchie avec leur extension ***.hrc***. Chaque nœud de l'Octree est représenté en tant qu'objet d'une classe ***Node*** dans notre application.

Pour le même nuage de points, plusieurs fichiers d'hiérarchie peuvent être créés une fois un certain niveau de nœuds est dépassé (selon le *HierarchyStepSize*), et cela dans le but de ne pas avoir un fichier très volumineux, qui prend du temps lors du chargement de la hiérarchie. Les fichiers d'hiérarchie contiennent également le nombre de points pour chaque nœud, mais peuvent être aussi calculés lors de l'exécution du programme afin de contourner tout bug lors de la conversion.

Le nombre réel de points peut être déduit de la taille des fichiers de nœud et du nombre d'octets pour chaque point calculé par la fonction***(ByteSize)***. Le nombre de points correct pour chaque nœud est déterminé la première fois que les points du nœud sont chargés en mémoire. Le chargement de la hiérarchie s'effectue dans un thread parallèle, appelé dans l'une des fonctions de démarrage d'Unity ***Start ()***. Ceci est fait pour que d'autres objets de la scène puissent déjà être rendus pendant le chargement de la hiérarchie de nuages de points. Dès que la hiérarchie est chargée, le processus de rendu du nuage de points commence.

4.2.3 Concept de chargement de base « multithread »

Le rendu des objets dans Unity est réalisé en créant des objets de jeu qu'on appelle des « *Game Objects*» pour chaque objet à rendre. Pour chaque nœud qui doit être visible, un ou plusieurs *Game Objects* doit être créé avec un filtre de maillage (*Mesh Filter*) approprié et un outil de rendu de maillage (*Renderer Filter*). Pour chaque nœud de l'Octree que nous voulons afficher, nous utilisons un « *Game Object* ». Si le nœud comprend plus de 65 000 sommets comme spécifie par Unity, plusieurs *Game Objects* sont nécessaires.

Le processus de chargement (comme sur la figure 4.2) utilise trois threads ou fils : le thread principal d'Unity, un thread de traversée et un thread de chargement. Dans le thread principal, les objets de jeu visibles sont mis à jour une fois par image si des modifications nécessaires ont été détectées dans le thread de traversée. Les objets de jeu sont créés pour les nœuds Octree qui devraient être visibles et ne possèdent pas encore d'objets de jeu. Les nœuds qui ne devraient plus être visibles voient leurs objets de jeu supprimés. Déterminer quels nœuds les objets de jeu doivent être créés ou supprimés est le travail du thread de traversée. Le fil de chargement est utilisé pour charger les données de points à partir des fichiers. Chaque fil est décrit plus en détail dans les prochaines sections thread principal, de traversée, et de chargement.

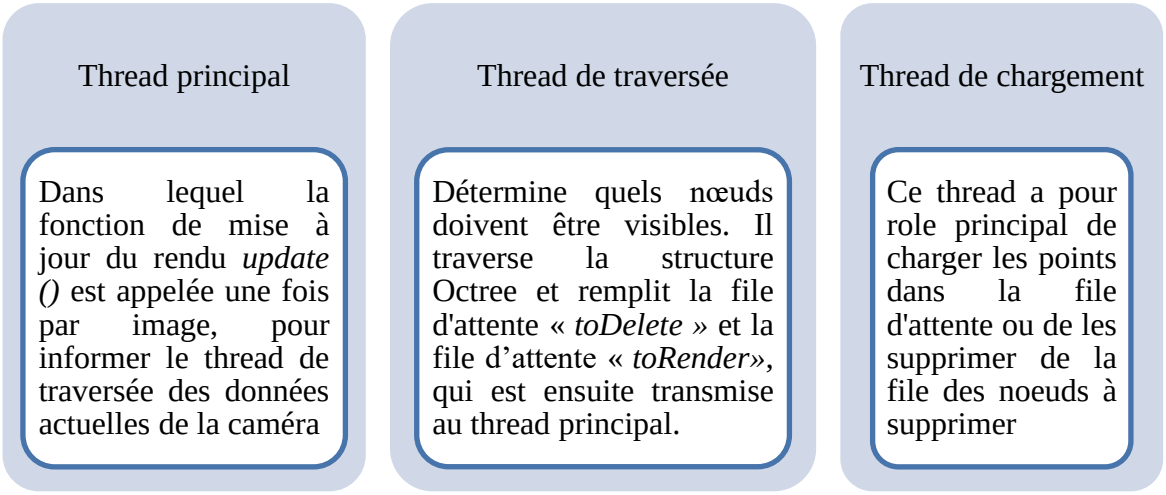


Figure 4. 2 *Organigramme des différentes thread et leur rôle*

4.2.4 Thread principal

Dans le fil principal d'Unity, la fonction de mise à jour du rendu est appelée une fois par image, qui est la fonction *update ()*. Dans cette fonction, le moteur de rendu informe le fil de parcours *thread de traversée* des données actuelles de la caméra (position, vue de dessus, etc.), car ces données ne peuvent être acquises que dans le fil principal d'Unity. Ensuite, il vérifie sa file d'attente *toDelete*. Cette file d'attente a été remplie par le fil de traversée et contient tous les nœuds dont les objets de jeu doivent être supprimés. Le thread principal supprime ces objets Unity mais conserve les points en mémoire (LRU) au cas où le nœud redeviendrait visible. Ensuite, le thread principal vérifie sa file d'attente *toRender*, qui a également été remplie par le thread de traversée, et crée des objets de jeu pour chaque nœud qui s'y trouve. À la fin, le thread de traversée est averti que la mise à jour des objets est terminée, afin qu'il puisse continuer avec une nouvelle traversée d'Octree.

4.2.5 Thread de traversée

Le thread de traversée est chargé de déterminer quels nœuds doivent être visibles. Il traverse la structure Octree et remplit la file d'attente « *toDelete* » et la file d'attente « *toRender*», qui est ensuite transmise au thread principal.

Tout d'abord, de nouvelles files d'attente vides sont créées. Ensuite, la traversée est terminée. Pour chaque nœud racine (il peut y avoir plusieurs nœuds racine si plus d'un nuage de points se trouve dans la scène), il est vérifié si sa taille projetée est supérieure ou égale à la taille projetée minimale. Si c'est le cas, le nœud est inséré dans une file d'attente de priorités, où la priorité est une combinaison de la taille projetée et de la centralité à l'écran (voir la section

priorité de traversée). Si ce n'est pas le cas, le nœud ne doit pas être visible. Par conséquent, s'il contient déjà des objets de jeu, le nœud et ses fils visibles sont insérés dans la file d'attente de suppression.

Ensuite, une variable qui contient le nombre de points de rendu, qui sera utilisée pour compter le nombre de points restitués, est initialisée à zéro et la file d'attente prioritaire est itérée dans une boucle. Lors de chaque passe en boucle, le nœud avec la priorité la plus élevée est supprimé de la file d'attente et les possibilités suivantes sont vérifiées :

1. Le nœud est en dehors champ de vision : Si le nœud a des objets de jeu, le nœud et ses fils visibles sont insérés dans la file d'attente *toDelete*.

2. Le nombre de points du nœud n'est pas encore connu : Planifiez le chargement du nœud par le thread de chargement. Il ne sera pas rendu cette image.

3. Le nombre de points est connu et son ajout au nombre de points de rendu ne dépasse pas le budget de points :

- Le nœud à des objets de jeu : Augmentez le nombre de points de rendu par le nombre de points de ce nœud. Le nœud n'a pas besoin d'être placé dans la file d'attente « *toRender* », car il est déjà visible.
- Le nœud n'a pas d'objets de jeu, mais ses points sont dans la mémoire : augmentez le nombre de points de rendu et placez le nœud dans la file d'attente *toRender*.
- Le nœud ne contient ni objet de jeu ni donnée de point stockée : planifiez son chargement par le fil de chargement. Le nœud ne sera pas rendu cette image.

4. Le nombre de points est connu et son ajout au nombre de points de rendu dépasserait le budget de points : s'ils possèdent des objets de jeu, ce nœud et ses fils sont placés dans la file d'attente *toDelete*. L'itération de la file d'attente s'arrête.

Après cette vérification, une boucle parcourt les « fils » du nœud. Si la taille projetée d'un fils est supérieure ou égale à la taille projetée minimale, elle est également insérée dans la file d'attente prioritaire. Si elle est inférieure à la taille minimale projetée et si elle contient des objets de jeu, elle et ses fils visibles sont insérés dans la file d'attente *toDelete*.

Le nombre de nœuds devant être programmés pour le chargement par parcours est limité, de même que le nombre de créations d'objet de jeu planifiées. De cette manière, le chargement des nœuds qui pourraient ne pas être utilisés dans la trame suivante est réduit. Le nombre

d'objets de jeu en cours de création est limité, car la création d'objets de jeu peut être une opération coûteuse.

L'itération de la file d'attente prioritaire s'arrête, lorsque le budget de points est atteint, le nombre maximum de chargements planifiés est atteint, le nombre maximum de créations d'objet de jeu planifiées est atteint ou la file d'attente est vide.

Après cette itération, il est vérifié s'il reste des nœuds contenant des objets de jeu, même s'ils ne doivent pas être visibles. Ces nœuds n'ont pas été vérifiés lors de la boucle d'itération, car le dépassement du budget en points a arrêté la boucle avant de les vérifier. Ces nœuds sont également placés dans la file d'attente « *toDelete* ». Ceci est réalisé en ayant des listes des nœuds visibles pour dernier parcours. S'il est déterminé qu'un nœud est visible, il est inséré dans la nouvelle liste et supprimé de l'ancienne. Ensuite, seuls les nœuds restants de l'ancienne liste doivent être placés dans la file d'attente à supprimer. À l'origine, la traversée se faisait dans le thread principal. Cependant, cela entraînait parfois un faible nombre d'images rendues par seconde lors de l'utilisation de gros nuages de points. Donc, il a été déplacé dans un fil supplémentaire pour améliorer l'expérience utilisateur. De cette façon, plusieurs images peuvent être rendues à un niveau de détail inférieur jusqu'à la fin du parcours.

4.2.6 Thread de chargement

Le thread de chargement possède une file d'attente de chargement. Chaque fois que le fil de traversée trouve un nœud à charger, il l'insère dans cette file d'attente. Le thread de chargement supprime en permanence le premier nœud de la file et charge les points de ce nœud (s’ils n’ont pas déjà été chargés entre-temps depuis leur insertion dans la file).

4.2.7 La priorité de traversée

La priorité utilisée pour la file d'attente de priorités dans le fil de traversée doit refléter la perception de l'utilisateur. Les nœuds avec une taille projetée plus grande auront généralement un impact plus important sur l'image rendue que les nœuds avec une taille projetée plus petite. De plus, comme la conscience de l'utilisateur est principalement centrée sur le centre de l'écran, les nœuds du centre doivent avoir une priorité supérieure à celle des nœuds de même taille situés au bord de l'écran. La valeur de priorité est déterminée comme indiqué dans les équations suivantes :

$$slope = \tan(\frac{fov}{2}) \quad (1)$$

$$projectedSize = \frac{screenHeight}{2} * \frac{radius}{slope * distance} \quad (2)$$

$$angle = \cos^{-1}(camToscreenCenter.camToNodeCenter) \quad (3)$$

$$priority = \frac{projectedSize}{|angle| + 1} \quad (4)$$

L'équation (2) montre comment calculer la taille projetée d'un nœud. Le rayon fait référence au rayon de la sphère englobante du nœud. La distance est la distance entre la caméra et le centre de la boîte englobant des nœuds. L'équation (3) montre comment calculer l'angle. Les deux vecteurs sont le vecteur avant de la caméra et la direction normalisée de la caméra au centre du nœud. L'équation (4) montre comment la priorité globale est calculée. Un est ajouté à la valeur absolue de l'angle, pour éviter des valeurs de priorité infiniment grandes ainsi que des divisions par zéro. La file d'attente prioritaire est implémentée à l'aide d'un segment de mémoire maximal-max heap).

Un exemple de l'affichage des nœuds par priorité c'est-à-dire avec des niveaux de détails différents est illustré avec la figure 4.3, et la figure 4.4 ci-dessous.

Figure 4. 3 *les nœuds en face de la caméra (en rouge) sont rendus plus en détails que ceux partiellement dans la vue, château de Jehay*

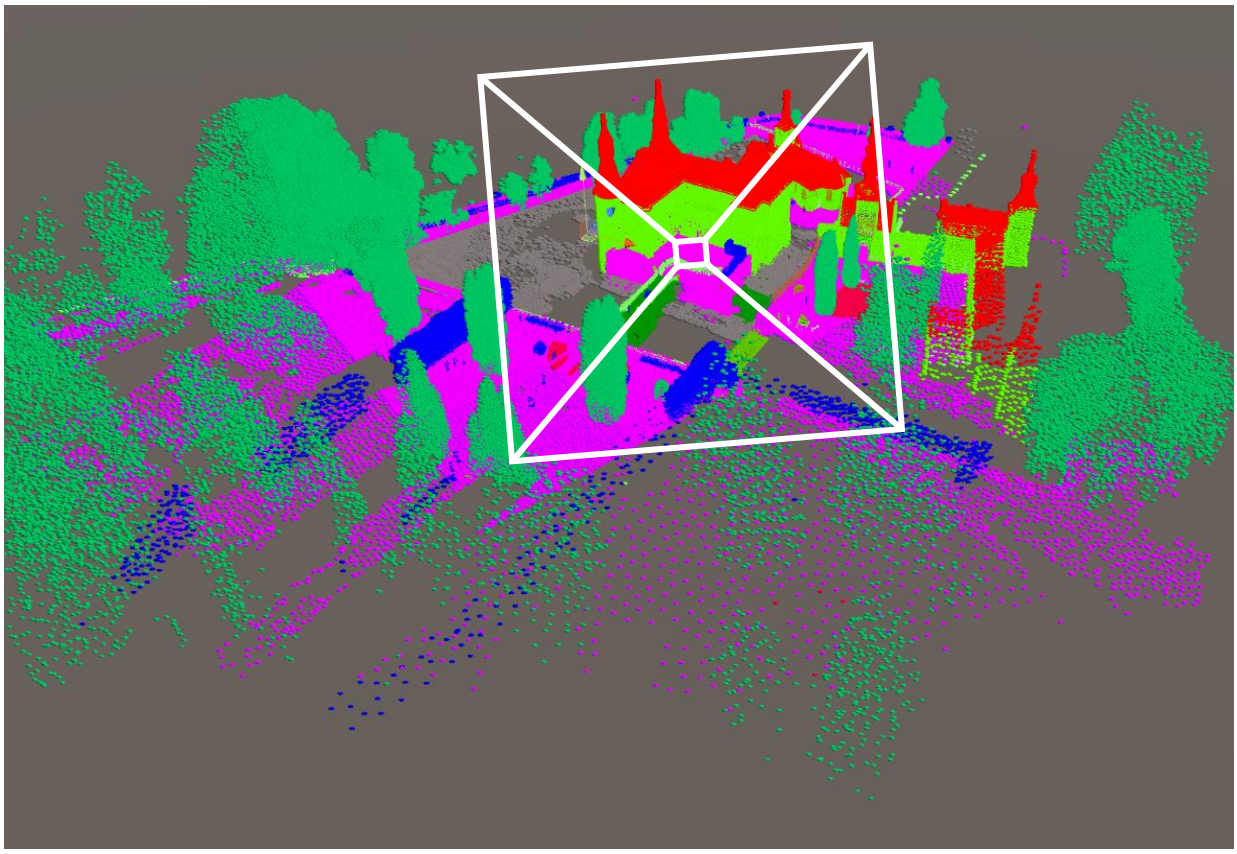


Figure 4. 4 *Un zoom sur le nuage de points classés montrant l'affichage par niveau de détails selon l'importance, château de Jehay*

4.2.8 Rendu des points

Pour donner un aspect plus réalistique au nuage de points, différentes techniques de rendu des points ont été mises en œuvre sur base du travail de (Fraiss, 2017). Aucun modèle d'éclairage n'a été mis en œuvre car les jeux de données de test ne contiennent pas de normales et ont été colorés par des photos qui donnent au modèle une forme d'éclairage statique. Mais avec l'ajout de la classification, le nuage de points n'a plus un aspect réalistique de 3D et de profondeur. Ainsi, la création d'un Shader qui permettra de simuler une source lumineuse s'avère nécessaire. Cette amélioration dans l'aspect visuel est montrée dans les figures 4.6 et 4.7 ci-dessous.

Le principe de ce Shader est simple, il consisté à créer un pseudo- Eye-Dome Lighting par la représentation de chaque point sous forme de carré ou de cercle, puis l'affectation d'une couleur noire à un coin (en bas à gauche dans notre cas) du carré ou cercle comme sur la figure suivante :

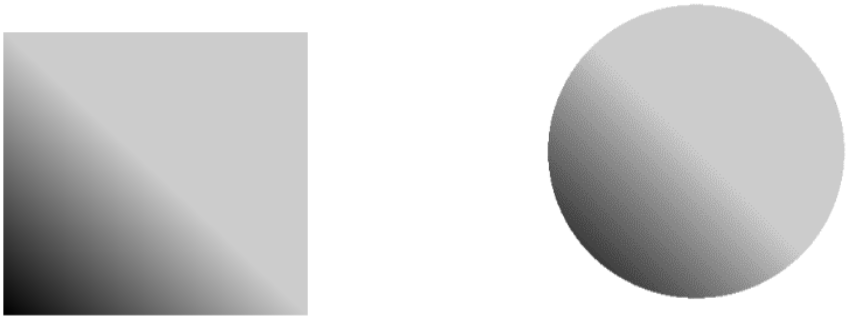


Figure 4. 5 *Principe du Shader implémenté, montré avec la forme carré (à gauche), et un cercle (à droite)*

Pour gérer les situations des occlusions entre les points, Schütz et Wimmer (2016) ont développé une méthode qui crée une interpolation des points de type le plus proche voisin. Pour ce faire, les points sont représentés sous la forme de formes 3D, telles que des cônes, des sphères ou des paraboloïdes, faisant face à l'écran, au lieu de simples carrés. Dans ce travail, des interpolations avec des cônes et paraboloïdes ont été mis en œuvre. Ces shaders font partie du travail de Fraiss, (2017) comme cité auparavant, et ont amélioré la visualisation de la classification comme présenté ci-dessous.

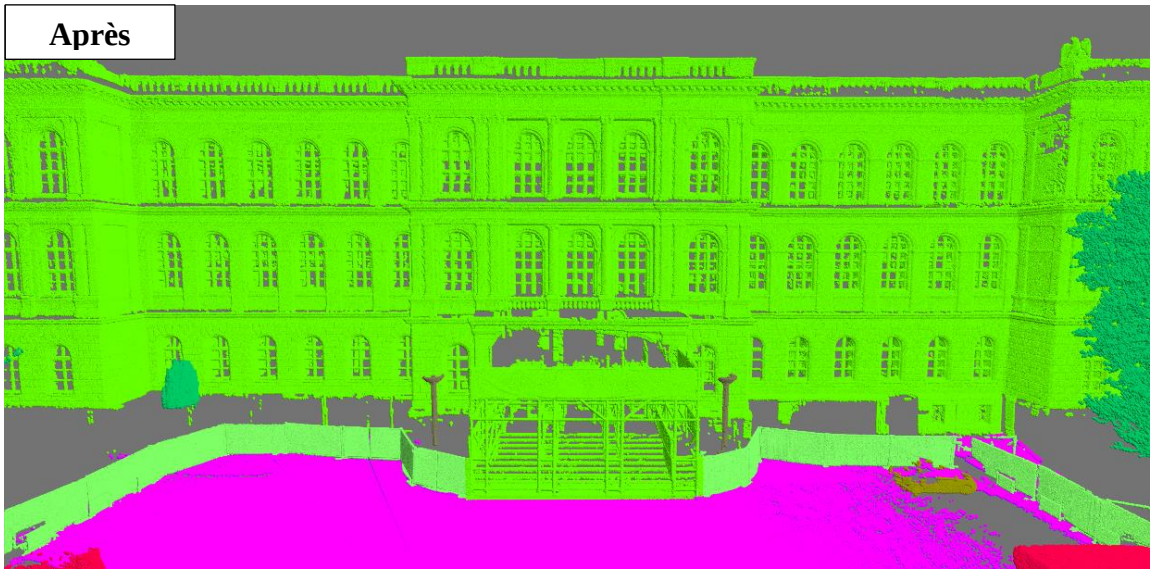


Figure 4. 6 *Nuage de point classé rendu avant et après application du Shader implémenté*

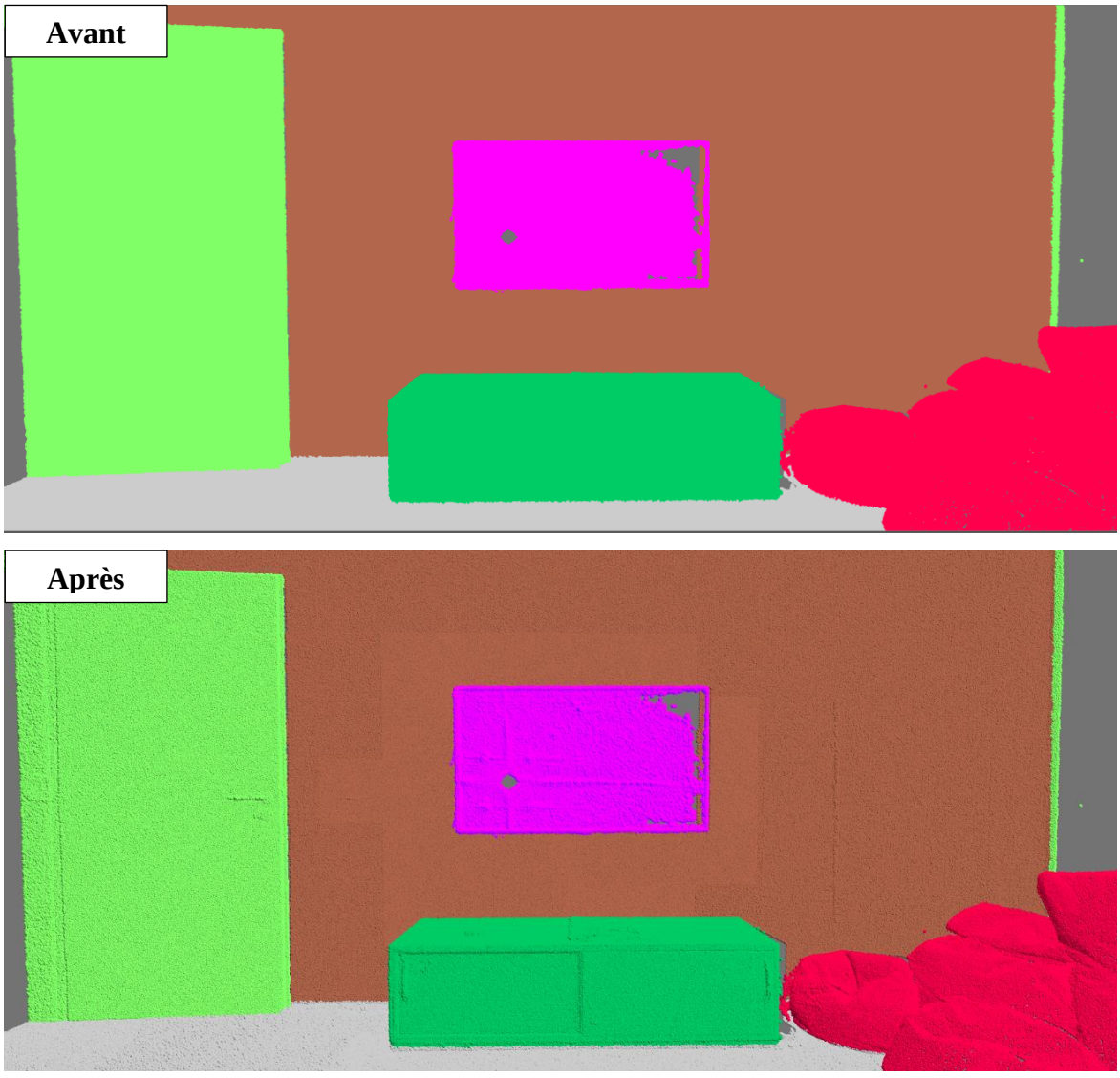


Figure 4. 7 *Nuage de point classé rendu avant et après application du Shader implémenté*

En effet, les points peuvent être rendus de plusieurs manières, dont la plus simple est de représenter chaque point avec un pixel sur l'écran. Le Shader des sommets affecte à chaque point une coordonnée sur l'espace écran et le fragment Shader lui donne la couleur appropriée. Mais le fait de représenter un point par pixel laisse apparaitre des trous dans les nuages de point surtout dans les zones à densité faible. Pour cela, un Shader de géométrie est appliqué afin de représenter chaque point par un espace carré ou cercle sur l'écran soit par un nombre de pixels soit par nombre d'unité dans le monde réel. Dans ce dernier, les points proches de la caméra auront une dimension plus grande que ceux qui sont plus loin. Afin de contourner l'occlusion des points les uns par les autres, plusieurs interpolations sont utilisées, qui consistent à rendre chaque point sous forme d'une sphère, un cône, ou un paraboloïde au lieu d'une géométrie 2D comme un carré ou un cercle.
Les scripts et les Shaders sont présentés en annexe 5 de ce document.

4.3 Création de l'interface Utilisateur

Pour faciliter l'interaction et rendre l'expérience en réalité virtuelle plus immersive, une interface utilisateur est créée par l'ajout d'un *canvas* qui est un « GameObject » qui englobe les éléments de l'interface utilisateur (Figure 4.8). Une fois affichée dans le monde virtuel directement face à l'utilisateur, elle garde la même position figée pour faciliter la sélection et le pointage par un pointeur laser (élément 5) créé par ajout du *Graphic Raycaster*[15] à notre canevas (Menu qui englobe les différents éléments de type : Bouton, slider, check box, et texte). Après modification des paramètres, l'interface peut être cachée et réaffichée en cas de besoin. Cette fenêtre permet de changer la taille des points (élément 1), de changer leurs formes (élément 2) (carré ou cercle), de choisir l'interpolation à utiliser entre les types de paraboloïde, cône, ou sans interpolation (élément 3), de sélectionner les attributs (élément 4) (couleur rgb, classification, intensité) pour l'affichage, de cocher et décocher les classes à visualiser (élément 6).

[15] Le Graphic Raycaster est utilisé pour diffuser des images sur un canevas. Le Raycast examine tous les graphiques sur le canvas et détermine si l'un d'entre eux a été touché par le poiteur laser dans notre cas.

Figure 4. 8 *L'interface utilisateur IU développé sous unity et visible en mode RV*

4.4 Gestion des attributs

Dans le cas du présent travail qu'a été développé sous unity, on a eu recours à créer une structure de données en parallèle pour stocker les informations des points affiché. En effet, unity un GameObjects ne prend par défaut que des informations sur le nom, les vertex et la couleur de chaque vertex, la raison pour laquelle on a créé des classes en parallèle à chaque fois qu'un nœud est affiché et qui permette de stocker toutes ses informations originales. Dans cette classe (extrait du code en dessous) chaque point prends les coordonnées du vertex, son identifiant, sa couleur RGB, sa classes, et son intensité.

```
    Public class MyObject
    {   Public string name;
        Public Vector3[] vertice;
        Public Color[] rgb;
        Public UInt16[] intensity;
        Public UInt16[] classif;
        Public UInt16[] id;
        Public BoundingBox bounding;
        Public MyObject(string name, Vector3[] vertexData, Color[] colorData,
UInt16[] intensit, UInt16[] classes, UInt16[] ident, BoundingBox bBox)
        {   this.name = name;
            this.vertice = vertexData;
            this.rgb = colorData;
            this.intensity = intensit;
            this.classif = classes;
            this.id = ident;
            this.bounding = bBox; }}
```

Pour le changement entre les différents attributs (rgb, classification, intensité), une condition est créée qui permette de visualiser les nouveaux nœuds directement par l'attribut choisie (couleur des classes), cette fonction permette aussi de changer selon ce nouvel attribut la couleur de tous les nœuds(*GameObjects*) déjà affiché par l'option *Mesh.colors* du GameObject. Pour visualiser ou cacher une classe(le sol par exemple), une fonction similaire est créée, qui modifier les vertex du Mesh via *Mesh.vertex* de chaque Gameobject. Cette fonction permette de visualiser les points des nouveaux nœuds directement selon les classes cochés. Elle permette aussi de modifier les vertex de tous les GameObjects affichés dans la scène via une condition qui parcourt tous les points affiché et supprime ceux qui seront plus visibles. En effet, pour contourner l'erreur liée à la correspondance entre le nombre des triangles et des vertex du nouveau Mesh. Il faut supprimer l'ancien Mesh par *Mesh.Clear*.

4.5 Application en RV

Après implémentation du code sur unity pour la lecture et visualisation de la structure Octree de Potree et la création de l'interface utilisateur, nous avons ajouté le package de l'*oculus intégration* au projet afin de permettre un rendu sur le casque oculus rift de Facebook, la version de l'application Oculus utilisée est ***Oculus 1.39.0.272909***.

« OVRPlayerController » est le moyen utilisé dans ce travail et qui sert à naviguer dans un environnement virtuel. Il s'agit essentiellement d'un préfabriqué *OVRCameraRig* associé à un simple contrôleur de caractères. Il comprend une capsule physique, un système de mouvement, un système de menus simple avec rendu stéréo des champs de texte et un composant réticulé.

Pour implémenter un mode de vol et de déplacement libre, on a modifié les scripts de *OVRPlayerController* afin d'avoir une force de gravité nulle (pour que Player ne soit pas toujours lié au sol, et le libérer de la contrainte de gravité). On a désactivé les mouvements linéaires (normalement le Player ne peut qu'avancer dans un plan précis selon des directions planaires). Puis, on a développé un script qui permettra à l'utilisateur d'avancer vers l'avant ou de reculer en arrière et cela dans sa direction de vue, c'est-à-dire qu'il bouge vers le point où il est en train de regarder.

Pour la lecture d'un nuage de points en format Potree, l'utilisateur n'a qu'entrer le chemin du dossier contenant le fichier cloud.js qui contient les métadonnées sur tous les nœuds et la hiérarchie. Une fois le chemin spécifié, l'utilisateur doit activer l'option « Virtual reality SDKs » sur Unity pour activer l'affichage sur casque (Figure 4.9).

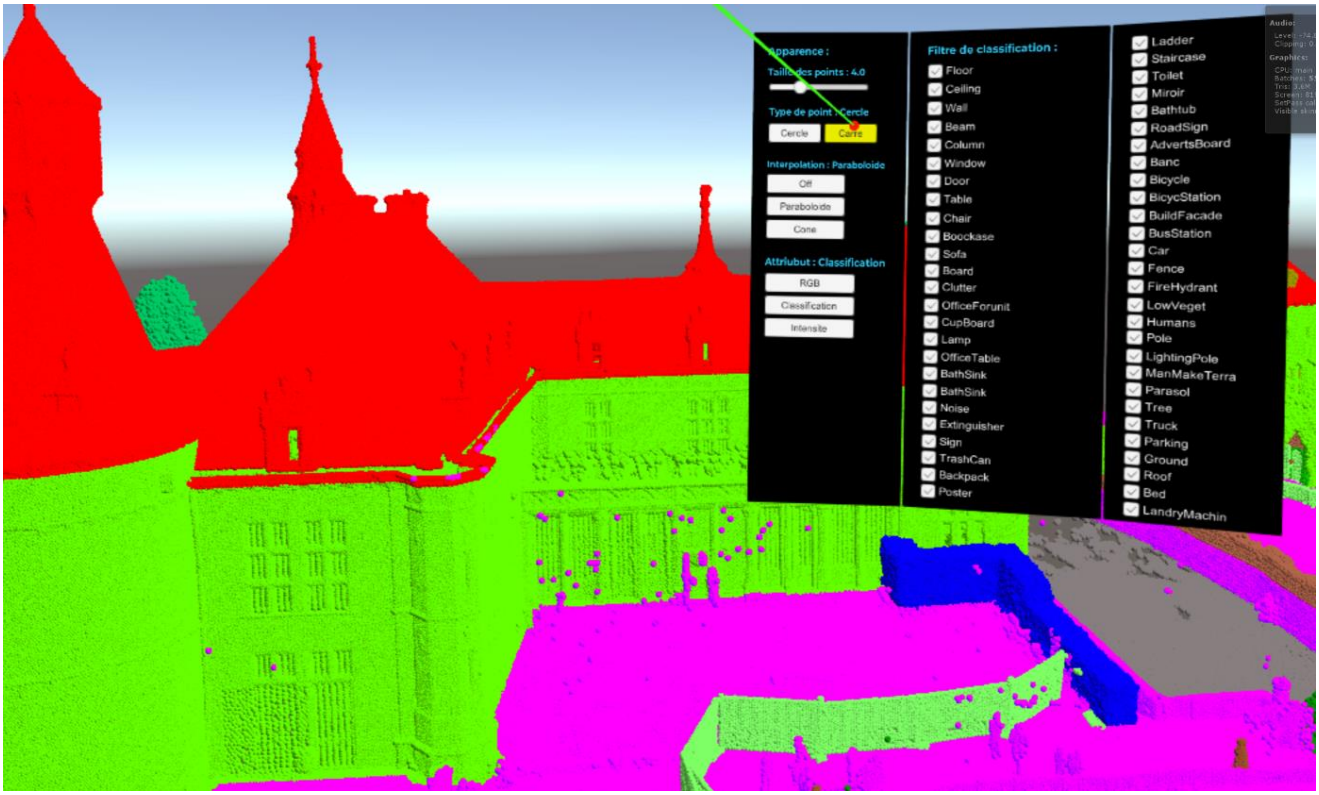


Figure 4. 9 *Casque oculus Rift S avec les deux manettes et les capteurs (à gauche), et la scène comme vue dans le casque par l'utilisateur*

Les boutons de navigation sont utilisés comme suit et comme illustré par la figure 4.10 :

- Pour affichage d'interface utilisateur : cliquez Grib droite et maintenu le clic (élément 1)
- Pour pointer et interagir avec IU : cliquez Trigger droit (élément 2)
- Pour le déplacement vers avant/arrière : ThumbStick de gauche (élément 3)
- Pour une rotation de 90° à gauche ou à droite : utiliser ThumbStick droit (élément 4)

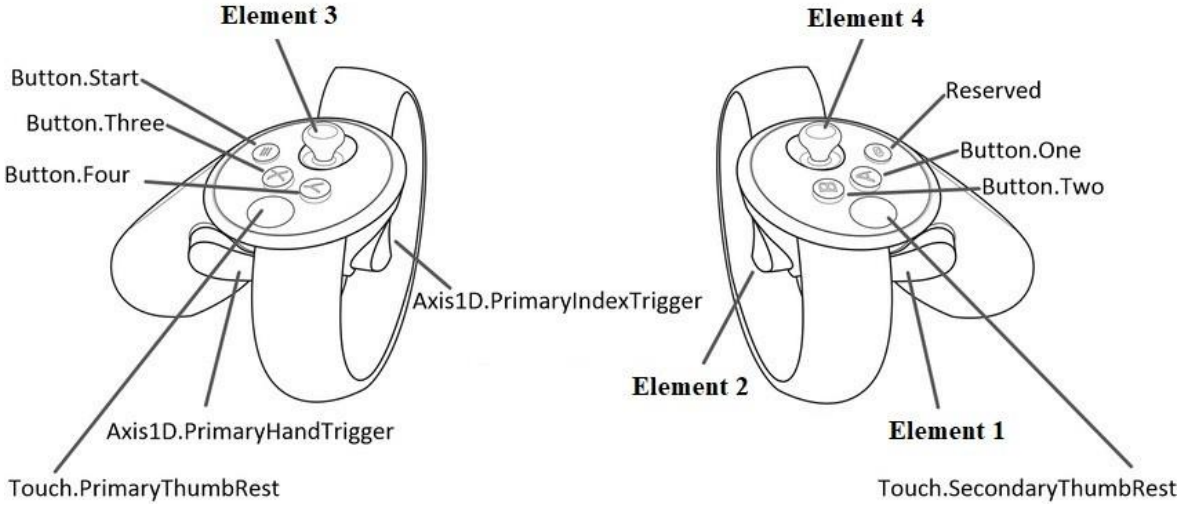


Figure 4. 10 *Les contrôleurs des manettes de l'oculus rift dans cette application, source: https://docs.unity3d.com/Manual/OculusControllers.html*

Conclusion

Nous avons présenté l'implémentation sous le Viewer WebGL des nuages de points massifs « Potree » afin d'intégrer les classes crées lors de la classification. Puis l'implémentation d'un système de rendu Out-Of-Core sous Unity permettant de lire le nuage de points dans son format binaire de Potree, en cherchant pour chaque image à temps réel les nœuds visibles, et en n'affichant que ces les points appartenant à ces nœuds. Enfin, pour permettre à l'opérateur plus d'interaction dans l'environnement virtuel, une interface utilisateur est développée permettant ainsi de changer la taille des points, leurs formes géométriques, l'interpolation utilisée pour l'affichage, les attributs à afficher (classification, RGB). Et enfin un mode de navigation en vol est implémenté afin de faciliter le déplacement de l'utilisateur partout dans le nuage de points.

CHAPITRE V
TETS DE PERFORMANCE ET DISCUSSION DE RESULTATS

Introduction

Dans ce chapitre, nous allons présenter les différents tests avec plusieurs nuages de points, et plusieurs utilisateurs pour valider les performances de la solution implémentée, et nous allons analyser et interpréter les résultats. Puis, l'apport de la solution dans l'intégration des nuages de points dans un environnement de réalité virtuelle est discuté, et les perspectives de recherche pour la continuité de ce travail sont détaillées.

5.1 Tests de performances

La méthodologie proposée précédemment est examinée sur plusieurs nuages de nuages de points avec des attributs différents et des tailles en termes de points variables allant d'une centaine de millions de points à plusieurs milliards de points. Aussi, des tests de confort avec le dispositif de réalité virtuelle sont faits afin de savoir l'avis des utilisateurs et leurs expériences en RV. En parallèle de ces deux tests, les performances logicielles en termes de rendu, de consommation de mémoire, et du nombre d'images rendues par seconde sont analysées et évaluées. Une autre analyse et évaluation a porté sur les résultats de la visualisation et d'optimisation

Afin de valider la méthodologie proposée et la solution implémentée dans précédent, plusieurs tests sont faits in Runtime. Ces derniers consistent à des tests sur les performances des méthodes de rendu utilisées, sur la capacité de l'application à visualiser des nuages de points massifs classifiés, ainsi que des tests de comparaison sur l'influence du nombre de points chargés sur ses performances en terme de consommation de mémoire et du nombre d'images par seconde(FPS en anglais frame per seconde).

Nous avons aussi évalué les performances de l'application en RV, par plusieurs personnes afin de savoir leur sensation, leur confort, et leur degré d'immersion et de déplacement en RV dans un nuage de points.

5.1.1 Tests sur plusieurs nuages de points

Afin d'analyser les performances de rendu de l'application en réalité virtuelle, le casque oculus rift (*Figure 5. 1*) est utilisé avec une résolution de 1080*1200 pixel pour chaque œil. Cet écran porte ainsi une résolution finale de 2160*1200 pixels, avec une fréquence de rafraîchissement de 90HZ, ce qui exige de vérifier cette condition par notre application afin de garantir une fréquence suffisamment élevée pour prévenir le mal de déplacement et offrir une expérience fluide.

Figure 5. 1 *Le casque de RV « Oculus rift » de Facebook*

Dans le but de comparer l'influence de la taille de nuage de points chargé sur les performances du rendu, plusieurs nuages de points sont utilisés (*Tableau 3. 1*). Le test est effectué en deux étapes : La première est réalisée pour une position et orientation fixe de la caméra, et la deuxième lors du déplacement de l'utilisateur.

Pour se faire, la fenêtre des statistiques du rendu en temps réel sur Unity est exploitée, elle permet d'afficher plusieurs paramètres utiles pour l'optimisation dont le plus importants dans notre cas est le FPS. Ainsi que pour l'analyse de l'utilisation de la mémoire, nous utilisons le profileur de mémoire (Memory Profiler) sur unity.

Alors que l'utilisation du casque oculus exige d'avoir des caractéristiques minimales sur ordinateur surtout en terme de carte de graphique (Nvidia GeForce GTX 1060 ou mieux ; AMD Radeon RX 480 ou mieux), le test de l'application est fait sur un ordinateur (0) avec les caractéristiques suivantes :

Tableau 5. 1 *Caractéristiques principales d' l'ordinateur utilisé avec le casque Oculus*

Processeur	Intel® Core™ i6-6800K CPU @ 3.40GHz 3.40 GHz
Carte graphique	NVIDIA GeForce GTX 1080
RAM	48.0 Go
Système d'exploitation	Windows 10 Pro, 64 bits

Tous les tests suivant sont faits avec un rendu basé sur le Shader qui permet une interpolation de type « paraboloïde », une taille de point de 4, une cache de 2 million de points et une forme de points de type « cercle » rendus avec l'effet pseudo-EDL développé dans ce travail. Les résultats sont présentés ci-dessous.

✓ **Influence du nombre de points chargé sur le FPS :**

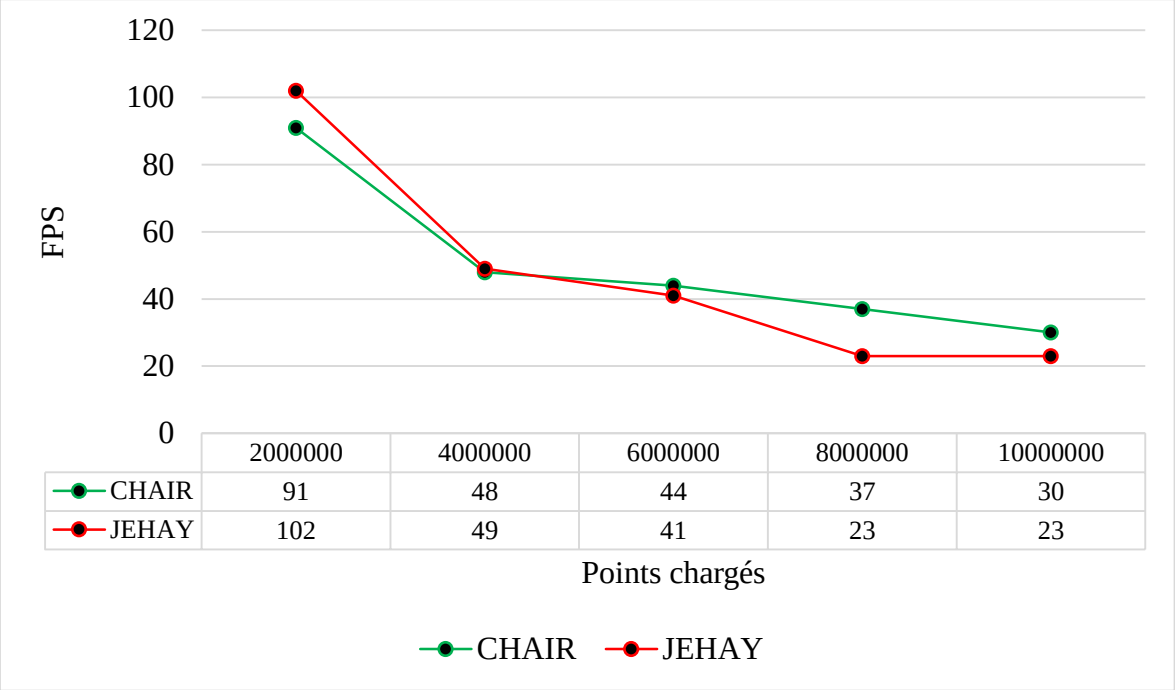


Figure 5. 2 *La variation du nombre FPS en fonction du nombre de points chargé en position fixe, le CHAIR avec 260 millions de point et JEHAY avec 2.3 Milliards.*

Le résultat de ce test montre que l'influence de la taille du nuage de points origine ne porte pas trop d'importance vue que les deux nuages de points choisis (JEHAY, et CHAIR) ont un espacement de (2.28m pour JEHAY, et 0.33 pour CHAIR), et que l'étendue de JEHAY et plus grande que CHAIR. C'est-à-dire que la structuration de ces deux nuages de points est presque similaire en termes de proportionnalité étendue/espacement, et n'entre pas trop en jeu mais plutôt les paramètres du rendu qui influencent.

Nous pouvons ainsi déduire l'influence du nombre de points autorisé à être chargé dans la scène (points budget) sur le nombre d'images rendues par seconde. Il est clairement visible que plus ce nombre augmente plus le FPS diminue, et cela pour la raison qu'il y a plus de points à afficher et à rendre. Avec un nombre de 2 Millions qui permet d'assurer un bon aspect visuel de la scène visible, on voit que le FPS est de plus de 90 qui largement suffisant pour garantir une visualisation fluide et confortable pour l'utilisateur.

Dans la suite, on va effectuer la suite des tests avec le nuage de points de JEHAY comme référence des tests. Alors que e nuage de point de CHAIR sera adopté comme un nuage de point témoin.

✓ **Influence du cache sur la consommation en mémoire**

La taille du nuage de points choisi est respectivement de 69.64Go pour JEHAY et de 4.81 Go pour CHAIR. Les deux paramètres principaux qui influencent sur la consommation en mémoire sont le nombre de points chargés et la taille de cache en termes de points c'est-à-dire les points qui restent stockés dans la mémoire au cas où ils deviennent de nouveau apparents.

Les résultats des tests que nous avons réalisés sont présentés dans les figures 5.3 et 5.4. Ils montrent que la consommation en mémoire augmente proportionnellement avec l'augmentation de la cache et avec l'augmentation du nombre de points chargés. Par exemple, pour une cache de 100 Millions, on voit clairement que la mémoire consommée passe de 1.475 Giga Octet pour 2 Millions de points à charger vers 1.985 Go pour 10 Millions de points à charger.

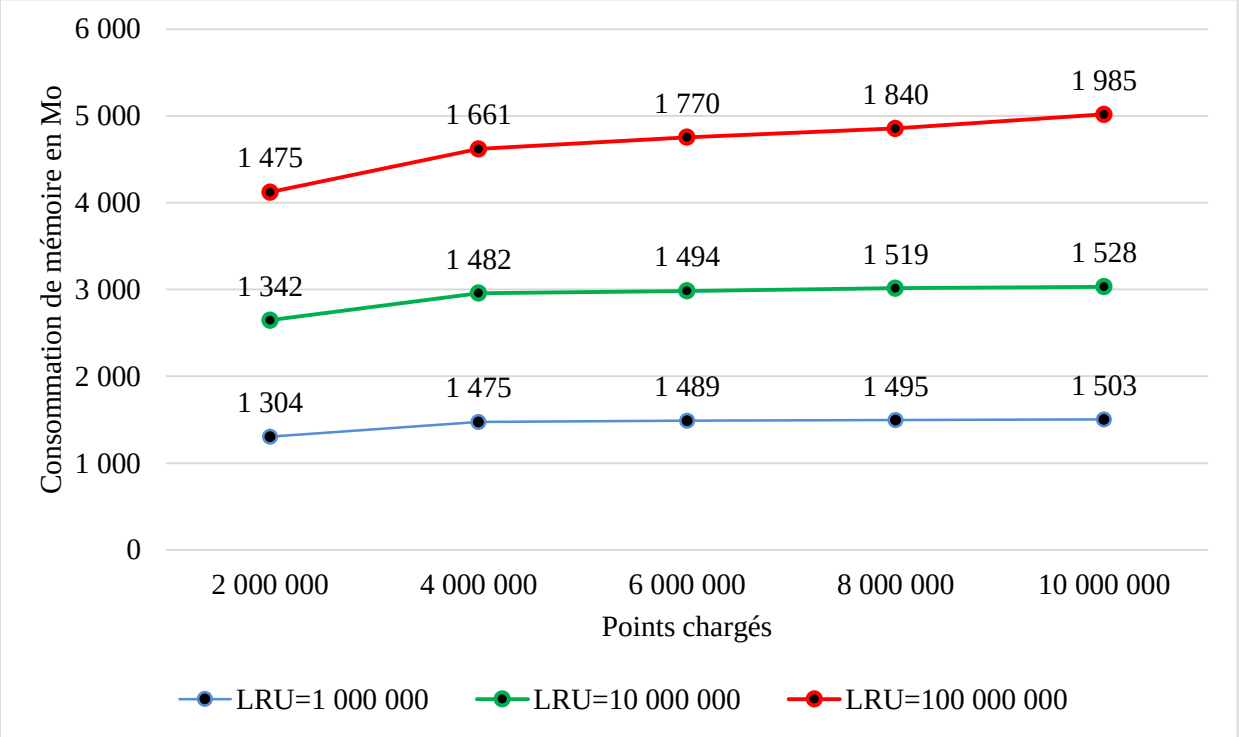


Figure 5. 3 *Variation de la consommation de la mémoire(en Mo) en fonction de la taille du nuage de point chargé et la cache LRU, en position fixe(JEHAY)*

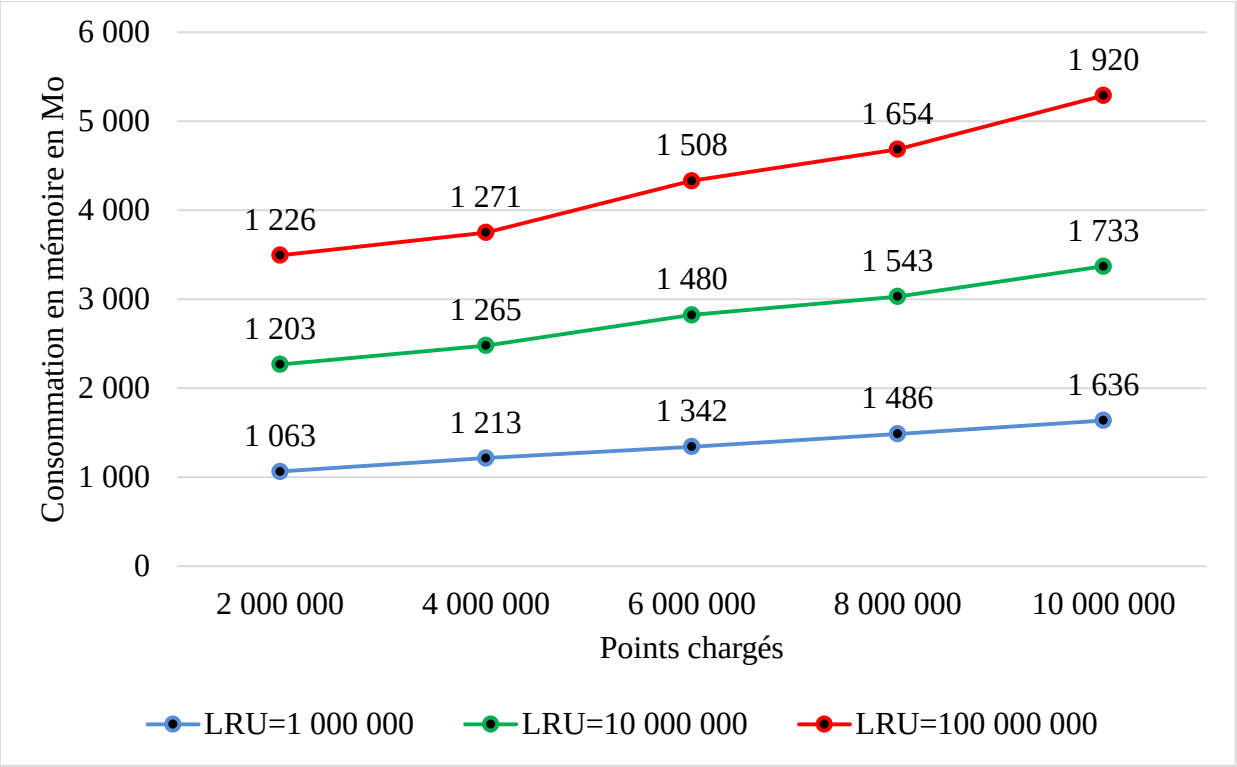
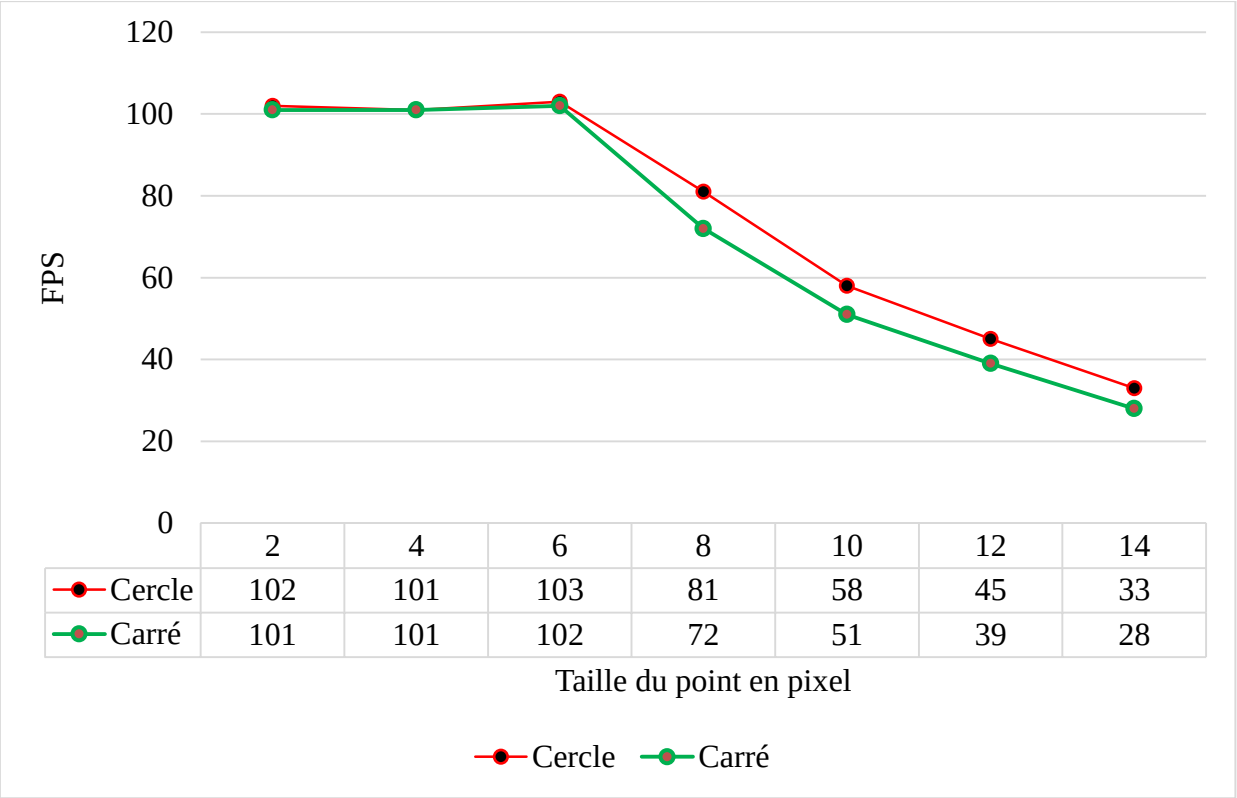


Figure 5. 4 *Variation de la consommation de la mémoire en fonction de la taille du nuage de point chargé et la cache LRU, en position fixe (CHAIR)*

✓ **Influence de la taille de point sur le FPS :**

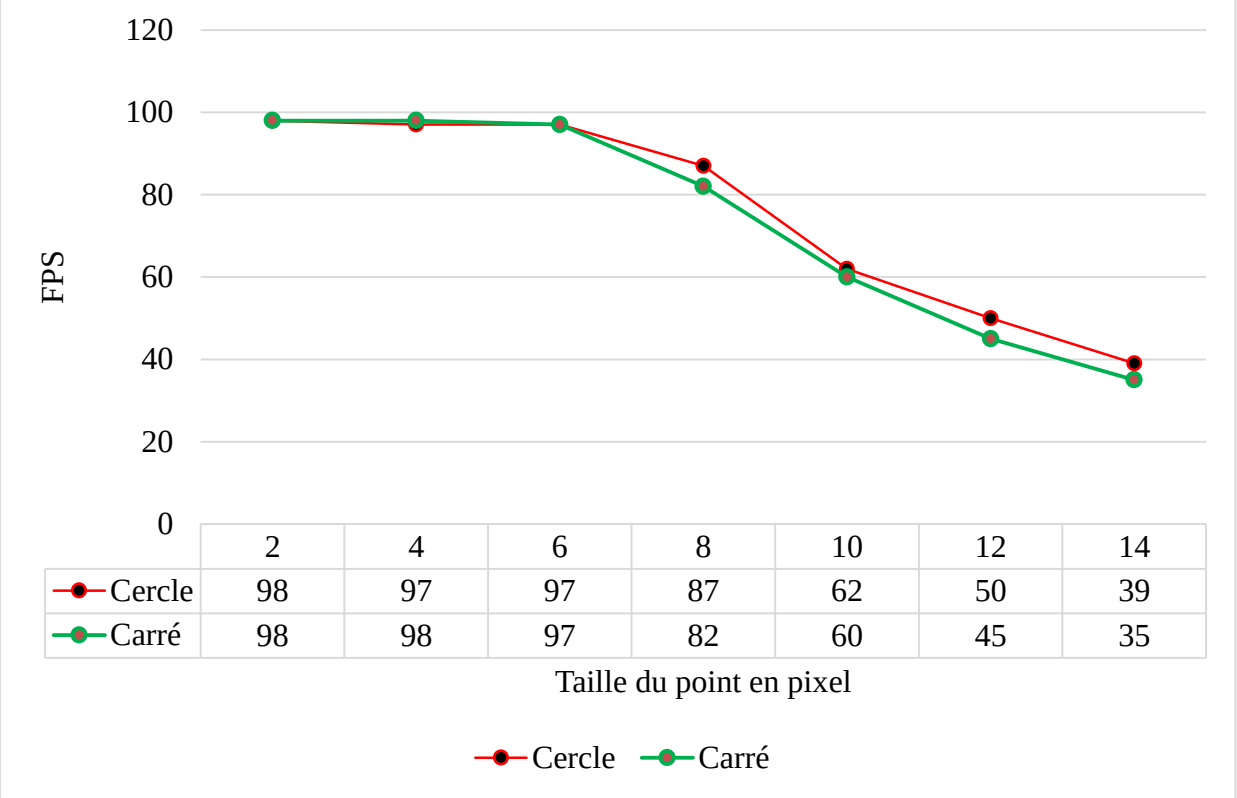
Parmi les méthodes de rendu utilisées dans ce travail, il y a le Shader qui permet de visualiser chaque point sous forme de cercle de carré sur l'écran au lieu de le visualiser sur un seul pixel. Ce type de visualisation permet de contourner les effets de la densité des nuages de points qui forment des vides entre les points lorsque cette densité est faible. Mais en plus de leurs avantages, ces méthodes de rendu sont coûteuse en termes de rendu et diminuent fortement le FPS.

Nous avons effectué un autre test de variation de la taille de points en nombre de pixels sur l'écran pour voir son effet sur le nombre de FPS. Il est clairement visible que le nombre de FPS diminue d'une manière inversement proportionnelle avec la taille du point. Plus la taille de points rendus soit en cercle ou en carré, plus le nombre de FPS diminue.



	2	4	6	8	10	12	14
Cercle	102	101	103	81	58	45	33
Carré	101	101	102	72	51	39	28

Figure 5. 5 *Influence de la variation de la taille du point sur le FPS, JEHAY avec un budget de 2Million, interpolation=paraboloïde, LRU=2M*



	2	4	6	8	10	12	14
Cercle	98	97	97	87	62	50	39
Carré	98	98	97	82	60	45	35

Figure 5. 6 *Influence de la variation de la taille du point sur le FPS, CHAIR avec un budget de 2Million, interpolation=paraboloïde, LRU=2M*

5.1.2 Tests sur l'expérience utilisateurs

Pour évaluer le confort des utilisateurs, nous avons préparé un questionnaire pour avoir un retour d'expérience et l'impression des utilisateurs de l'application de RV. Les tests sont faits auprès des personnes du laboratoire. D'après les résultats, nous avons constaté que les utilisateurs qui utilisent pour la première fois le casque de réalité virtuelle souffrent du vertige et de certains malaises, ce qui rend ce type de test un moyen insuffisant pour l'évaluation de ce type application. Après un certain temps d'usage du casque, les utilisateurs donnent de bonnes impressions et deviennent habitués au déplacement en RV. On a pu alors conclue, que ce type d'étude de performance à partir des expériences des utilisateurs débutants est un moyen non objectif, et il faut se baser sur des aspects scientifiques, dont notre application montre une très grande performance et efficacité.

Pour la suite de ce travail, nous prévoyons faire adopter le questionnaire suivant inspiré des travaux de (Trinon, 2019), pour une évaluation scientifique basée sur le retour d'expérience utilisateurs.

Les questions sont :

1. Vous arrive-t-il d'être tellement absorbé(e) dans l'environnement virtuel que les gens autour de vous ont de la difficulté à vous en tirer ?
2. Jusqu'à quel point vous sentez-vous mentalement éveillé(e) ou vif (ve) d'esprit en ce moment même ?
3. Dans quelle mesure les mécanismes permettant votre mouvement dans l'environnement vous semblaient-ils naturels ?
4. Dans quelle mesure vos sens étaient-ils trompés par le réalisme du mouvement des objets à travers l'espace ?
5. Dans quelle mesure les expériences que vous avez vécues dans l'environnement virtuel ressemblent-elles à celles de l'environnement réel ?
6. Jusqu'à quel point la sensation de déplacement à l'intérieur de l'environnement virtuel était-elle confondante (réaliste) ?
7. À quel rythme vous êtes-vous adapté(e) à l'expérience vécue dans l'environnement virtuel ?
8. En termes d'interactions et de déplacements dans l'environnement virtuel, jusqu'à quel point vous sentiez-vous compétent(e) à la fin de l'expérience ?

9. Jusqu'à quel point la qualité visuelle de l'appareillage graphique vous a-t-elle incommodé(e) dans l'exécution des tâches requises ?
10. Jusqu'à quel point êtes-vous parvenu(e) à vous concentrer sur les tâches requises plutôt que sur les mécanismes utilisés pour effectuer lesdites tâches ?

Ces réponses sur ces questions sont de 1 (pas du tout), 2 (moyenne) et 3 (totalement)

5.2 Discussion et perspectives

En terme de visualisation du nuage de points pour des applications de réalité virtuelle, plusieurs solutions soit open source ou commerciales existent. Elles permettent la visualisation d'un nombre qui reste limité de points (des petits nuages de points d'une dizaine de millions) et une fois ce nombre dépassé, la visualisation devient lourde. Pour contourner ce problème nous avons exploité la structure octree de potree, qui est structure modifié du format modifiable Nested octree. Cette structure qui permet de faire une indexation spatiale du nuage de points en octree a montré une grande efficacité en termes de chargement de grands nuages de points qui peuvent atteindre les 3 Milliards dans notre cas de test.

La structure utilisée dans le cas du présent travail a donné de bons résultats en termes de visualisation en temps réel et en continu des nœuds de l'octree qui figurent dans le champ de vision de l'opérateur. En effet la recherche et l'accès aux points à afficher ne se fait plus d'une manière séquentielle mais plutôt directe par la recherche des « Bounding Box ». Ces performances de visualisation sont dues au stockage de la hiérarchie dans plusieurs fichiers qui sont faciles à charger et lire au lieu d'un seul fichier lourd.

Dans notre cas, nous avons profité de ces avantages pour la visualisation à travers une application qui donne de bons résultats en termes de FPS et de consommation de mémoire. D'après les tests faits, nous avons obtenu un nombre d'images par seconde qui dépasse la valeur nécessaire pour l'œil humain qui est de 90 FPS. En plus, à travers la structure de données utilisés la consommation en mémoire est optimisée pour que l'application peut être utilisé sur des PC de moyennes capacités en RAM (vers 8Go) à condition de bien paramétrer le nombre de points chargé sur scène(2 Millions par exemple) et le nombre de points qui reste en cache(2 Millions par exemple).

En plus de la partie visualisation, notre application permet une interaction via l'interface utilisateur sur laquelle il peut sélectionner les classes à visualiser ou désélectionner les

classes à cacher. Mais, pour l'interaction directe avec le nuage de points, il s'est avéré que pour visualiser les éléments d'une seule classe(les arbres par exemple qui sont dispersé sur tout l'espace), il faut faire un parcours séquentiel de tout le nuage de points ce qu'est une solution non optimale. Pour cela nous proposons la solution : qui consiste à faire une indexation/structuration par rapport à la classification (selon les numéros des classes) dont le but de faciliter la recherche et l'accès aux classes.

Pour la partie qualité du rendu, nous avons implémenté un nouveau Shader similaire à l'effet donné par Eye-dôme-Lightning et qui permet de simuler l'effet d'une source lumineuse sur le nuage de points sans calcul des normales. Ce nouveau Shader donne plus de réalisme aux points rendus surtout lors de l'utilisation du casque de réalité virtuelle, et précisément avec la classification qui ne présente pas un effet de profondeur.

Une contribution non négligeable de ce travail consiste à améliorer les paramètres de visualisation et de rendu en RV afin de donner une expérience confortable à l'utilisateur tout en évitant de lui créer des effets de Cybersickness (la cinétose[16], simulator sickness or RV sickness) qui sont dus principalement au déplacement

Les perspectives et les futurs travaux peuvent être résumés principalement dans les 4 points suivants :

- ✓ Création d'une double indexation spatiale et par classification pour l'interaction directe avec le nuage de points dans l'environnement de RV.
- ✓ implémentation d'une méthode de rendu par niveau de détails continu CLOD (Continuous Level of Detail), pour éviter l'effet des nœuds qui apparaissent et disparaissent donnants un effet flottant.
- ✓ Amélioration de la structure pour aller vers des interfaces qui permet de modifier ou introduire des nouveaux.
- ✓ Test de performances sur base des méthodes proposé dans la partie 5.1.2 Tests sur l'expérience utilisateurs.

[16] La cinétose, survient lorsque le sujet est immobile mais a la sensation de bouger à cause d'une exposition à des images visuelles changeantes (Norman and Vinson, 2012)

CONCLUSION GENERALE

La popularité et l'utilisation des données en nuage de points augmentent de manière exponentielle. Dans ce travail, nous avons proposé une approche complète pour la classification et la visualisation des nuages de points avec plusieurs milliards de points en temps réel et en continu dans un environnement de réalité virtuelle. Notre approche est basée sur une version modifiée de Potree issue de la structure de donnée Modifiable Nested Octree.

Récemment, plusieurs avancées dans les méthodes de classification automatique sont faites surtout avec l'arrivée des algorithmes d'apprentissage automatique et d'apprentissage approfondi. Ainsi que la gestion des données spatiales de types nuages de points massifs s'orientent vers l'utilisation des structures de données de type Octree. Ce travail se base sur ces deux orientations de recherche dans le domaine de la lasergrammétrie, afin de les exploiter pour une application de réalité virtuelle.

Nous avons suivi une méthodologie basée sur une partie pratique, et une partie de développement. En effet, après acquisition des données sur terrain nous avons procédé à une classification de plusieurs nuages de points de sources différentes et de tailles variables (plusieurs millions et plusieurs milliards de points). Cette classification est faite à l'aide des algorithmes automatiques, semi-automatiques et manuels. Puis, nous avons effectué une structuration des nuages de points classifiés sous un format libre de Potree qui utilise une version modifié du format MNO est faite afin d'optimiser l'affichage. Cette structure d'octree qui contient un sous-échantillon basse résolution dans son nœud racine et des sous-échantillons progressivement à plus haute résolution dans des nœuds de niveau supérieur, permet à Potree de restituer les régions distantes à un niveau de détail inférieur, réduisant ainsi le nombre de points ayant être chargé et rendu. Les régions qui sont en dehors du point de vue sont totalement ignorées. La structure de données de Potree utilisée est une extension de la structure MNO d'origine de telle sorte que la hiérarchie d'octree elle-même soit stockée dans un autre octree « hiérarchie-octree » qui permet également de charger la hiérarchie à la demande. Cela est devenu nécessaire car des millions de nœuds sont nécessaires pour stocker des nuages de points plus grands et le stockage de toute la hiérarchie dans un seul fichier entraîne une augmentation du temps de chargement initial.

Cette structuration en format Potree permet d'optimiser considérablement la visualisation en temps réel des grands nuages de points mais reste un format qui ne permet pas la

modification du nuage de points brut, c'est-à-dire on ne pas faire des traitements sur le nuage de points initial. En plus de cela, la structuration prend un temps considérable qui peut aller de quelques minutes à une dizaine d'heures lorsque le nuage de points dépasse des milliards de points.

Après la classification et la structuration du nuage de points, nous avons développé des algorithmes « Out-Of-Core » principalement sur la base des travaux de Fraiss, 2017. Ces algorithmes permettent de vérifier en continu les points qui existent dans le champ de vision et de charger en temps réel et en continu les nœuds de l'octree qui existent dans les champs de vision de l'utilisateur.

Pour obtenir une meilleure qualité visuelle, nous avons mis en œuvre des méthodes de pointe telles que le Shader développé (*Figure 4. 7*) pour fournir un éclairage pour les nuages de points sans normales et une projection de surface de haute qualité avec des cercles/carrés alignés sur l'écran. En plus de cela, nous avons également ajouté plusieurs Shaders avec des interpolations de type paraboloïde et cône, semblable au voisin le plus proche qui permettent de dessiner les points comme des paraboloïdes/cônes alignés sur l'écran plutôt que des cercles.

Nous avons développé une interface utilisateur contenant toute les classes élaborées lors de la classification et les paramètres de rendu. En effet, cette interface permet de changer les paramètres et méthodes de rendu (taille de points, type, type d'interpolation utilisé) en temps réel, ainsi que de visualiser les classes selon le choix de l'utilisateur. Cette solution a permis de visualiser des nuages de points qui dépassent 3 Milliards de points et elle a montré de grandes performances en termes de FPS qui dépasse le 93 et une bonne efficacité en termes de consommation du mémoire.

REFERENCES BILBIOGRAPHIQUES

Ait El Kadi, K., 2016. LiDAR terrestre (Light Détection And Ranging), note de cours

Benejean, M., 2013. Selection et amélioration de nuages de points 3D, thése de doctorat à l'université de TOULOUSE.

Bowman, D.A., Kruijff, E., LaViola, J.J., Poupyrev, I., 2001. An introduction to 3-D user interface design. Presence Teleoperators Virtual Environ. 10, 96–108.

Dewez, T.J.B., Girardeau-Montaut, D., Allanic, C., Rohmer, J., 2016. Facets : A cloudcompare plugin to extract geological planes from unstructured 3d point clouds. ISPRS Arch. 41, 799–804.

Fischler, M. a, Bolles, R.C., 1981. Paradigm for Model. Commun. ACM 24, 381–395.

Fraiss, S.M., 2017. Rendering Large Point Clouds in Unity.

G. Vosselman, B.G.H. Gorte, G. Sithole, T.R., 2002. recongnising structure in laser point clouds, article scientifique 1–6.

Gobbetti, E., Marton, F., 2004. Layered Point Clouds. Comput. Graph. 28, 815–826.

Grilli, E., Menna, F., Remondino, F., 2017. A review of point clouds segmentation and classification algorithms. ISPRS

High, A., Topography, R., 2016. Structure from Motion (SfM) Photogrammetry Field Methods Manual for Students 1–10.

Karami, E., Prasad, S., Shehata, M., 2017. Image Matching Using SIFT , SURF , BRIEF and ORB : Performance Comparison for Distorted Images.

Landes, T., Grussenmeyer, P., Boulaassal, H., 2011. Les principes fondamentaux de la lasergrammétrie terrestre : acquisition , traitement des données et applications, article XYZ (partie 2 / 2) 25–38.

Mazuryk, T., Gervautz, M., 1996. Virtual Reality - History, Applications, Technology anda Future 1–72.

Mures, O.A., Jaspe, A., Padrón, E.J., Rabuñal, J.R., 2016. Virtual Reality and Point-Based Rendering in Architecture and Heritage.

Norman, G., Vinson, N.G., 2012. NRC Publications Archive Archives des publications du CNRC Cybersickness induced by desktop virtual reality Cybersickness Induced by Desktop Virtual Reality.

Novak, D., 2014. Virtual Reality Technology and Applications.

Poux, F., 2019. THE SMART POINT CLOUD Structuring 3D intelligent point data.

Poux, F., 2014. Vers de nouvelles perspectives lasergrammétriques : optimisation et automatisation de la chaîne de production de modèles 3D Florent Poux, mémoire de PFE

Poux, F., Billen, R., 2019. Voxel-based 3D Point Cloud Semantic Segmentation: Unsupervised Geometric and Relationship Featuring vs Deep Learning Methods. ISPRS Int. J. Geo-Information 8, 213.

Rakotosaona, M., Polytechnique, É., 2019. P OINT C LEAN N ET : Learning to Denoise and Remove Outliers from Dense Point Clouds, article scientifique 1–17.

Rusinkiewicz, S., Levoy, M., 2005. QSplat, article scientifique

Scheiblauer, C., der Arbeit, V., Wimmer, M., Gervautz, M., gang Knecht, W.-, Marek, S., Hebart, G., Gschwantner, F.-M., Zimmer, N., Mayer, I., Fugger, V., Barsukov, Y., Preiner, R., Mayer, J., Pregesbauer, M., Tragust, M., Arikan, M., 2014. Interactions with Gigantic Point Clouds, article scientifique

Scheiblauer, C., Wimmer, M., 2011. Out-of-core selection and editing of huge point clouds. Comput. Graph, article scientifique

Schuetz, M., 2016. Potree: Rendering Large Point Clouds in Web Browsers, thése de master

Trinon, H., 2019. immersive technologies for virtual reality case study : flight simulator for pilot, mémoire de Mater

Wahl, R., Klein, R., 2007. Efficient RANSAC for Point-Cloud Shape Detection, article scientifique

Wimmer, M., Scheiblauer, C., 2006. [WS06] Instant Points. SPBG'06 Proc

Zhang, W., Qi, J., Wan, P., Wang, H., Xie, D., Wang, X., Yan, G., 2016. An easy-to-use airborne LiDAR data filtering method based on cloth simulation. Remote Sens. 8, 1–22.

WEBOGRAPHIQUE

Git Hub de ce mémoire : https://github.com/akharroubi/Point-clouds-classification-and-integration-in-VR-environment.

GitHub Large point clouds in Unity: https://github.com/SFraissTU/BA_PointCloud

Potree: http://www.potree.org/

GitHub Potree: https://github.com/potree/potree

Unity Game Engine: https://unity.com/fr

Oculus : https://www.oculus.com/,(consulté)

Git Hub PotreeConverter : https://github.com/potree/PotreeConverter

Thèse de Simon Maximilian Fraiss : https://www.cg.tuwien.ac.at/research/publications/2017/FRAISS-2017-PCU/

Note : Ces sites sont consultés entre juin et juillet 2019.

ANNEXES

ANNEXE 1 : Outils de segmentation et classification sur CloudCompare

Introduction

CloudCompare est un logiciel de gestion et de comparaison de nuages de points 3D (Ainsi que de maillages 3D dans une certaine mesure). Il a été développé en grande partie par Daniel Girardeau-Montaut.

Le menu en dessous englobe les outils et plugins de segmentation intégrée sur CloudCompare *Figure 6. 1*, dont la segmentation interactive ①, composantes connectés ②, le filtrage sur base des valeurs ③, CANUPO ④, CSF filtre ⑤, Facets pour la détection des surfaces planes ⑥ et RANSAC pour des détections des formes primitives ⑦.

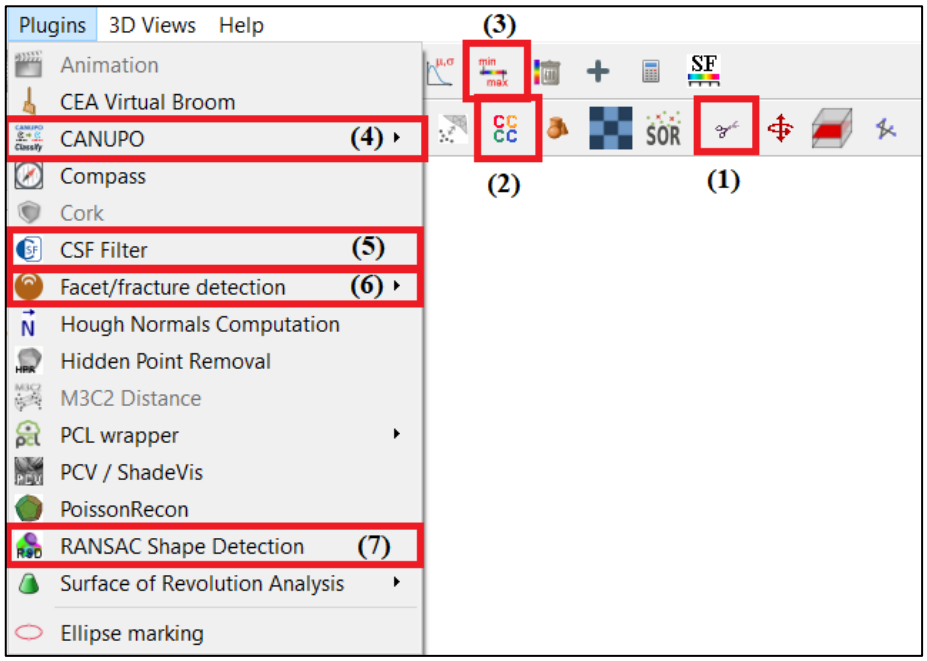


Figure 6. 1 *Outils et plugins de segmentation/classification intégrés sur CloudCompare*

a. Segmentation interactive

L'outil segmentation manuelle (ou segmentation graphique/interactive) est accessible via l'icône de la barre d'outils supérieure (Main Toolbar). Il permet de "découper" manuellement à l'écran la ou les entités sélectionnées.

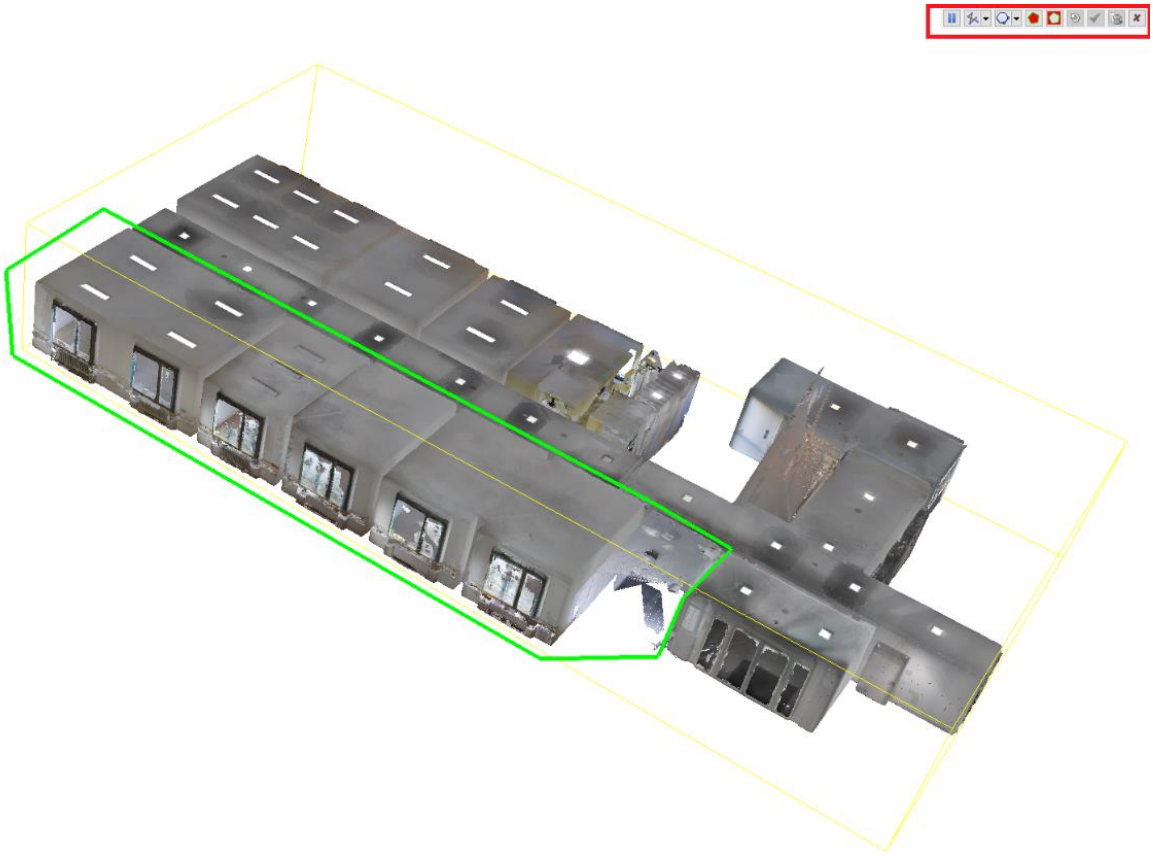


Figure 6. 2 *Outil de segmentation manuelle sur cloudcompare, en vert la partie à découper et en rouge la fenêtre qui englobe les outils de segmentation*

Cet outil permet en effet de définir un contour à l'écran (*Figure 6. 2*), puis de choisir si l'on veut "garder" les points ou les triangles présents à l'intérieur ou à l'extérieur de ce contour. Le processus est répétable à volonté, et les points/triangles rejetés sont cachés au fur et à mesure. Si l'utilisateur valide sa segmentation, les entités sélectionnées sont chacune divisées en deux : une partie correspondante aux points/triangles sélectionnés et l'autre avec les points/triangles restants.

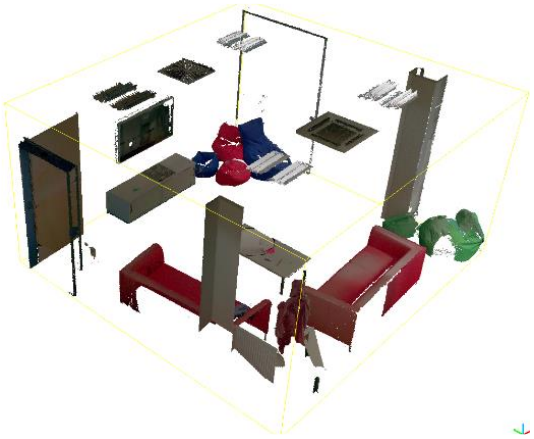
b. Label Connected Components

Cet outil est accessible via l'icône dans la barre d'outils principale supérieure ou avec la commande Outils → Segmentation → Label Connected Components.

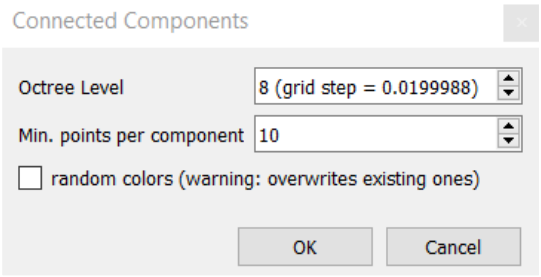
Cet outil segmente le ou les nuages sélectionnés en parties plus petites séparées par une distance minimale. Chaque partie est un composant connecté (c'est-à-dire un ensemble de points « connectés »).

Les paramètres à entrer sont (*Figure 6. 3*) :

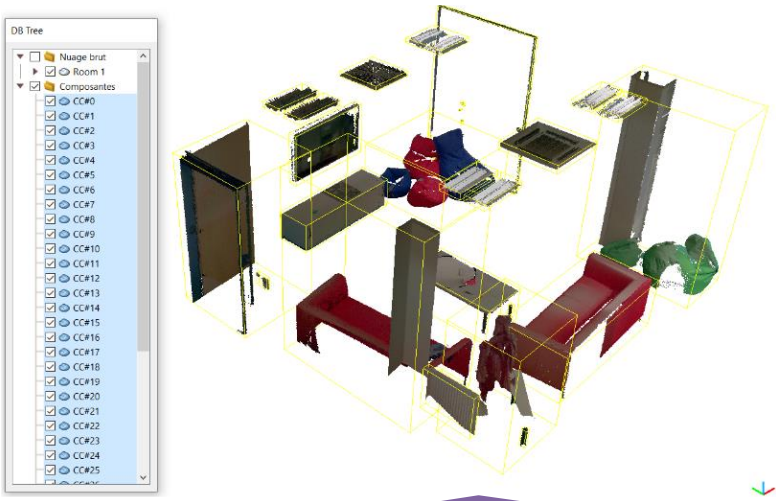
- *Niveau Octree* : CloudCompare utilise une grille 3D pour extraire les composants connectés. Cette grille est déduite de la structure Octree. En sélectionnant le niveau d'octet, vous définissez la taille minimale de l'écart minimum entre deux composants (la taille de cellule correspondante est affichée à côté du niveau). Plus le niveau est élevé et plus l'écart est petit (plus vous aurez de composants).
- *Min. points par composant* : les composants dont le nombre de points est inférieur au nombre spécifié seront ignorés (utile pour supprimer les plus petits composants)
- *Couleurs aléatoires* : si cette case est cochée, une couleur aléatoire sera attribuée à chaque composant (avertissement : toutes les couleurs existantes seront écrasées !)



1. Nuage de point brut



2. Paramétres entrée



3. Résultats

Figure 6. 3 *Composantes connectées crées avec les paramètres spécifiés*

c. Filtrage sur base de valeurs

Cet outil est accessible via le menu "Edition → Champs scalaires → Filtrer par valeur" ou l'icône dans la barre d'outils principale supérieure.

Filtre les points du nuage sélectionné en fonction de leur valeur scalaire associée. Un nouveau nuage (ou maillage) est créé avec les points situés dans la plage spécifiée (*Figure 6. 4*).

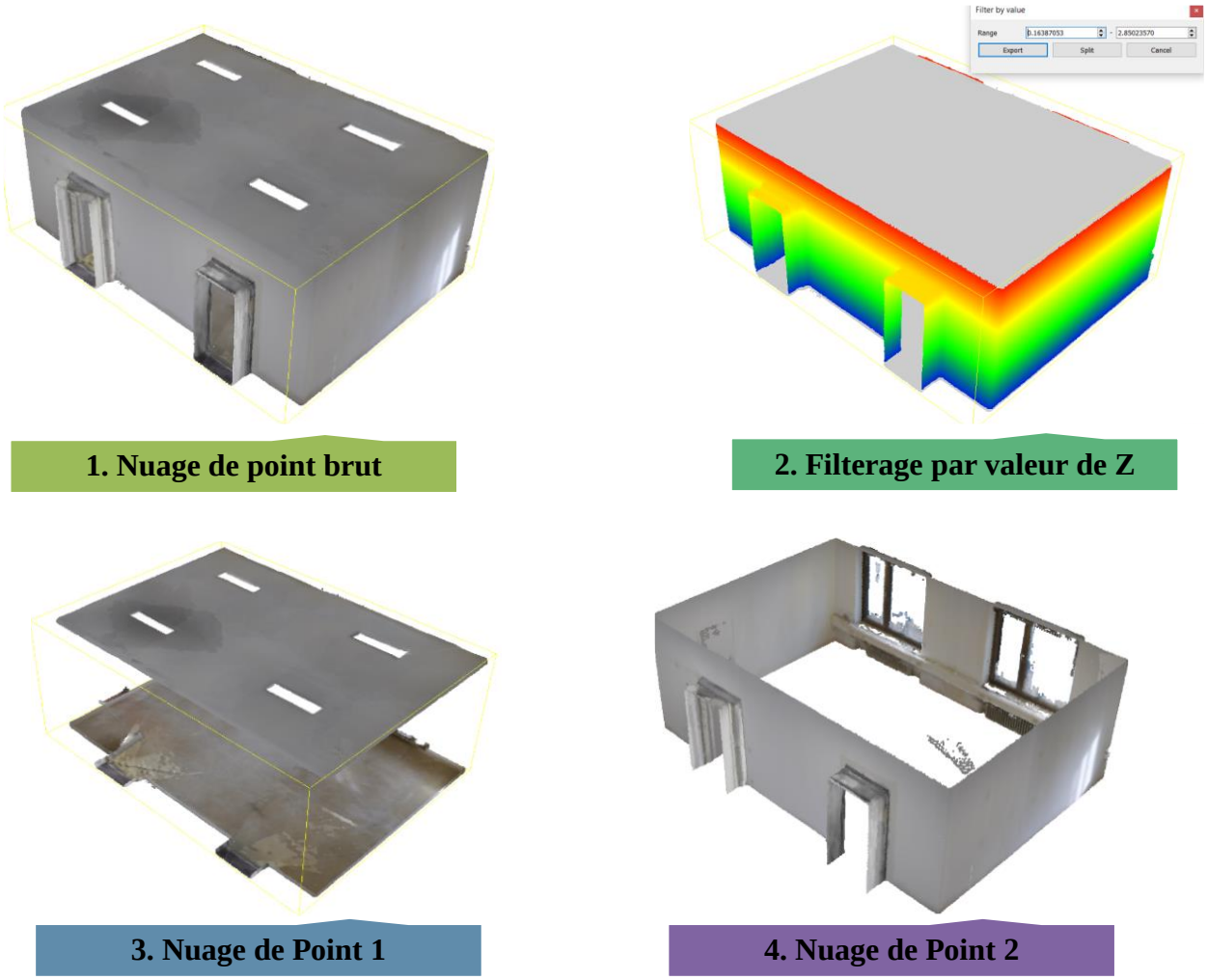


Figure 6. 4 *Filtrage sur base de la valeur de la hauteur Z*

d. CANUPO

Le plugin CANUPO est un moyen simple mais efficace de classer automatiquement un nuage de points. Il permet de créer des classificateurs (en les formant sur de petits échantillons) et / ou d'appliquer un classificateur à la fois sur un nuage de points afin de le séparer en deux sous-ensembles (*Figure 6. 5*). Il génère également une valeur de confiance

de classification pour chaque point afin de pouvoir identifier rapidement les cas problématiques (généralement aux limites des classes).

Les classificateurs sont stockés dans des fichiers «. prm » autonomes. Ils peuvent être appliqués à n'importe quel nuage de points, à condition que ses unités soient en concordance avec le classificateur. La classification avec cet outil passe par les étapes suivantes :

- Préparation des données qui représentent chaque classe à créer. Pour chaque classe, plusieurs sous-ensembles de points typiques sont pris et regrouper dans un seul nuage.
- Entrainement du classificateur CANUPO Create, sur base des données d'entrée, cet outil est accessible via le chemin « Plugins → qCANUPO → Train classifier »
- Application sur le nuage de point à classifier CANUPO Classify via le chemin « Plugins → qCANUPO → Classify »

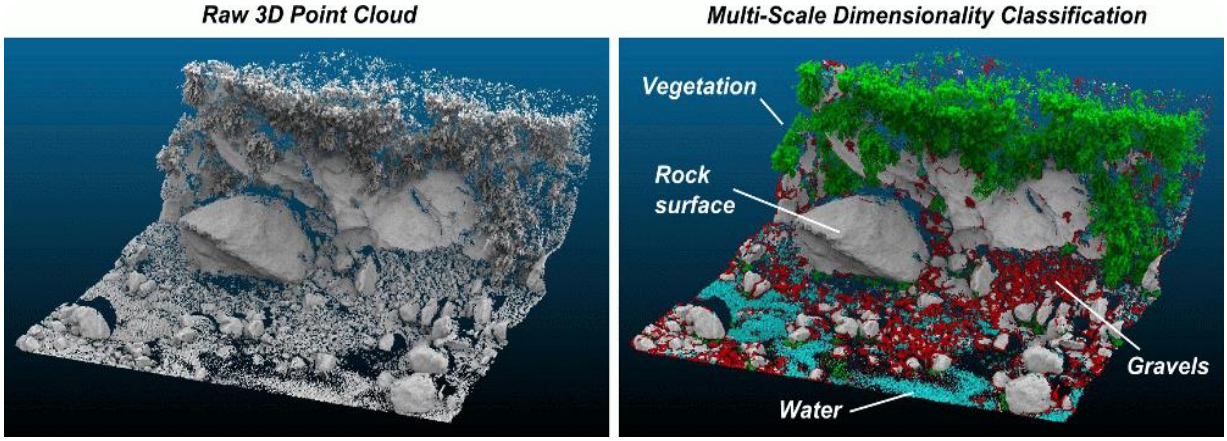


Figure 6. 5 *Nuage de points classifié par un classificateur entrainé et lancé avec CANUPO, extrait du site de CANUPO*[17].

- **CSF**

Cette méthode est basée sur une simulation de tissu qui est un algorithme graphique 3D et est utilisée pour simuler un tissu dans un programme informatique. Dans cette approche proposée, un nuage de points Lidar est inversé, puis un tissu rigide est utilisé pour recouvrir la surface inversée (*Figure 6. 6*). En analysant les interactions entre les nœuds de tissu et les points Lidar correspondants, il est possible de déterminer l'emplacement des nœuds de tissu pour générer une approximation de la surface du sol. Enfin, les points au sol peuvent être extraits du nuage de points Lidar en comparant les points Lidar d'origine et la surface générée.

[17] https://www.cloudcompare.org/doc/wiki/index.php?title=CANUPO_(plugin)

Cet algorithme de filtrage pourrait être appelé filtrage de simulation de tissu, CSF (*Figure 6. 7*).

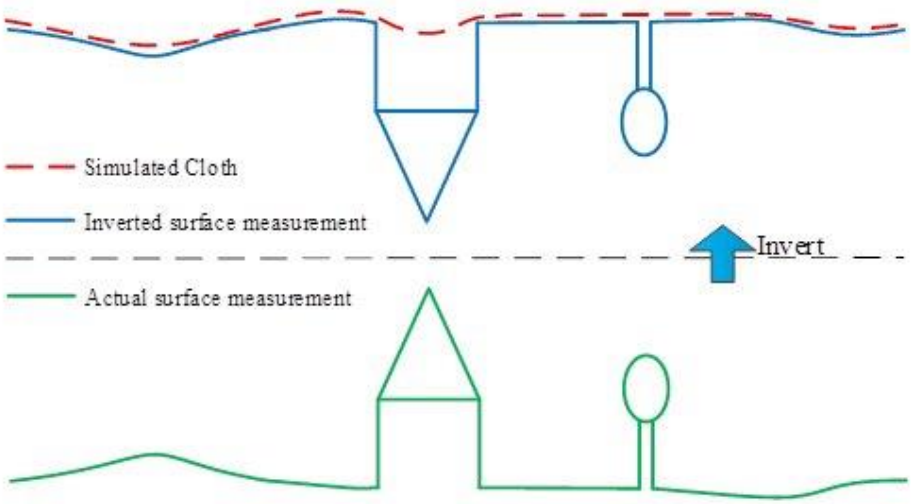


Figure 6. 6 *Principe de fonctionnement du CSF filter*

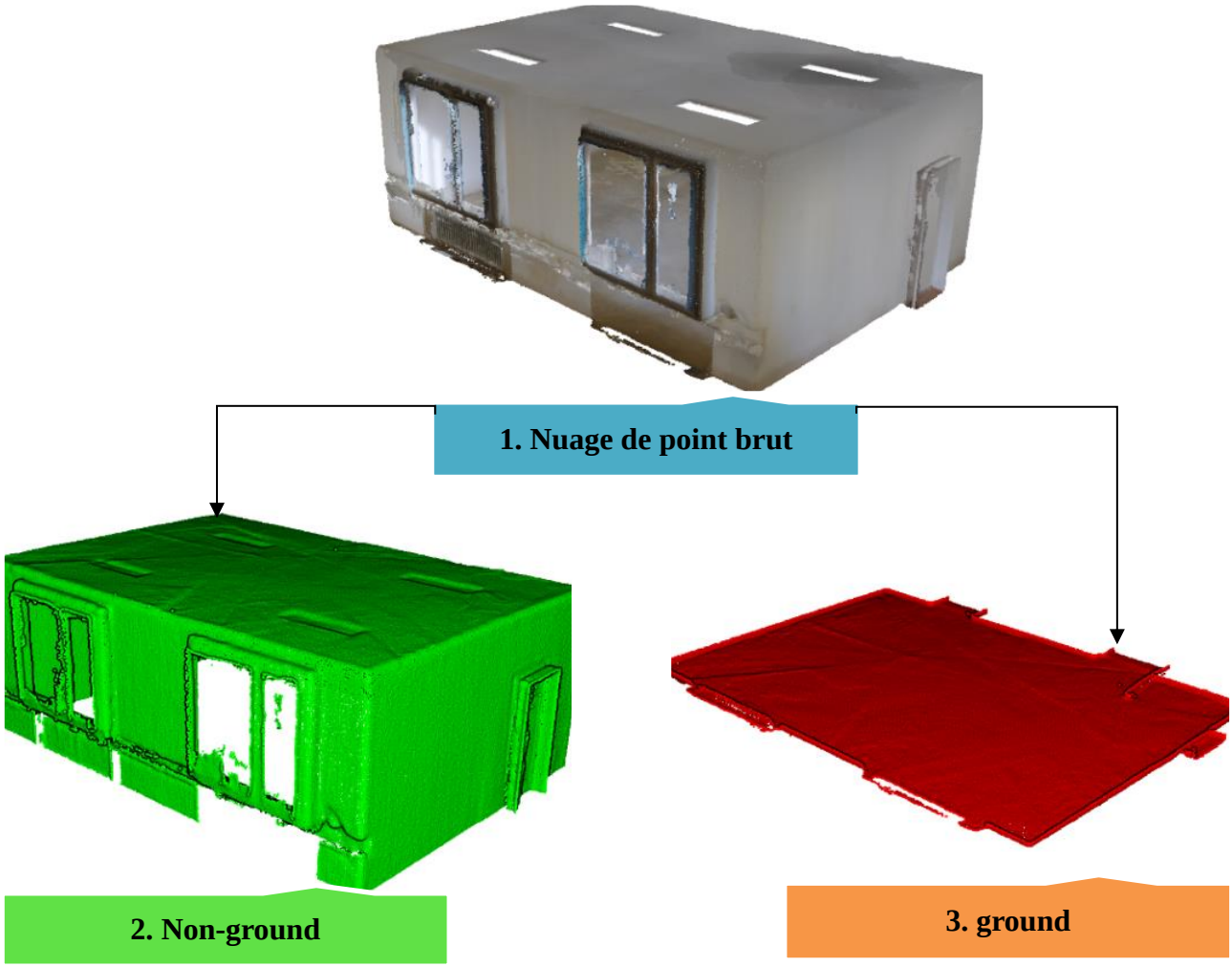


Figure 6. 7 *Nuage de point classé par CSF plugin, en partie sol et partie non sol*

e. Facets

Le plugin Facets(Dewez et al., 2016) permet les fonctionnalités suivantes :

- Extraire automatiquement les facettes planaires (par exemple, les plans de fracture) des nuages de points.
- Exporter les facettes dans des fichiers SHP
- Classer les facettes en fonction de leur orientation et de leur distance (orthogonale)
- Afficher les orientations sur un stéréogramme / stéréographie
- Filtrer les facettes (ou les points s'ils ont des normales) en fonction de leur orientation

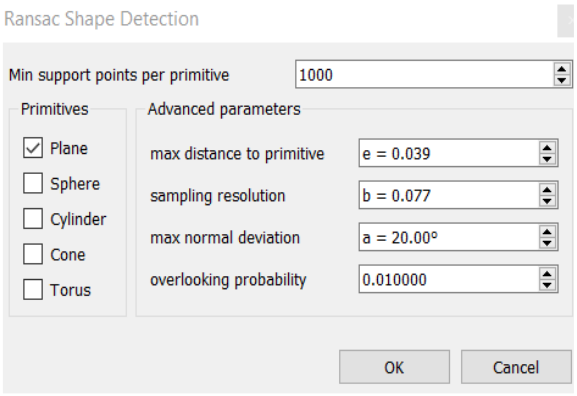
Deux algorithmes effectuent la segmentation : Kd-Tree et Fast Marching. Les deux divisent le nuage de points en sous-cellules, puis calculent des objets planaires élémentaires et les agrègent progressivement. Selon un seuil de planéité en polygones.

Ce plugin est accessible via la barre de menu → Plugins → Facets, il demande plusieurs paramètres en entrée selon l'algorithme choisie entre Kd-Tree et Fast Marching.

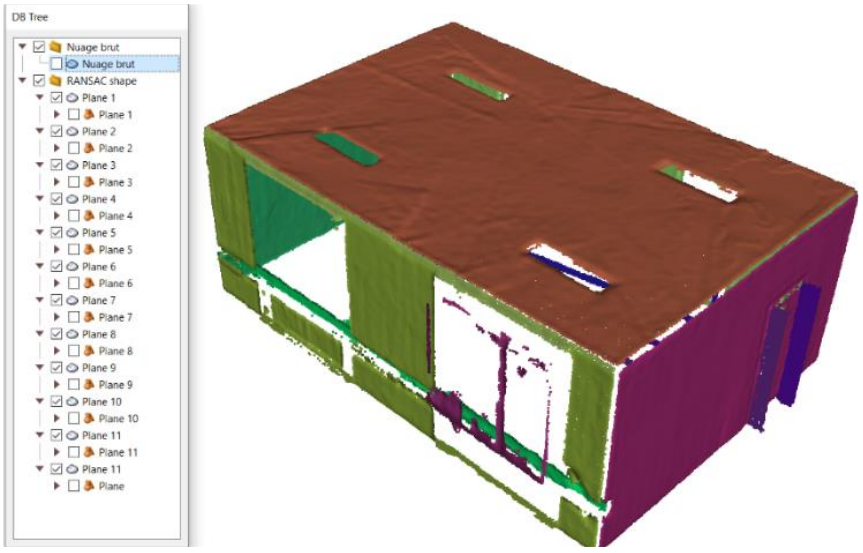
f. RANSAC Shape Detection

Dans ce plugin, il y a plusieurs paramètres a spécifie dont le plus important est le nombre d'échantillons (points de support) par primitive. Tout dépend de la densité du nuage et de la taille des formes à détecter. Uniquement le type de primitives à détecter est cocher (voir les cases à cocher dans les paramètres d'entrée) afin d'éviter facilement les détections fausses / inutiles. À la fin, le plugin créera un ensemble de nouvelles entités. Pour chaque forme détectée, est créé : un nuage de points correspondant au sous-ensemble de points prenant en charge la primitive détectée. Le nom du nuage intègre les paramètres de primitives estimés l'entité correspondante en tant que fils de ce nuage (*Figure 6. 8*)

1. Nuage de point brut



2. Paramétres entrée



3. Résultats

Figure 6. 8 *Nuage de point segmenté par l'algorithme de RANSAC*

ANNEXE 2 : Outil PotreeConverter

Ce convertisseur permet de Construire une structure Octree de Potree à partir de fichiers las, laz, binaires, xyz ou ptx, les options disponibles sur PotreeConverter sont les suivantes :

Tableau 6. 1 *Les lignes commandes propres à PotreeConverter et leurs significations*

-i [--source]	Les fichiers entrés
-h [--help]	Affichage Help
-p [--generate-page]	Génère une page Web prête à utiliser avec le nom donné.
-o [--outdir]	répertoire de sortie
-s [--spacing]	Distance entre les points au niveau de la racine. Distance réduit de moitié chaque niveau.
-d [--spacing-by-diagonal-fraction]	Nombre maximum de points sur la diagonale du premier niveau (définit l'espacement). Espacement = valeur diagonale
-l [--levels]	Nombre de niveaux qui seront générés. 0 : uniquement root, 1 : root et ses fils,
-f [--input-format]	Format d'entrée. xyz : coordonnées cartésiennes sous forme de flotteurs, rgb : couleurs sous forme de nombres, i : intensité sous forme de nombre
--color-range	...
--intensity-range	...
--output-format	Le format de sortie peut être BINARY, LAS ou LAZ. La valeur par défaut est BINARY
-a [--output-attributes]	Peut être n'importe quelle combinaison de RVB, INTENSITY et CLASSIFICATION. La valeur par défaut est RVB.
--scale	Échelle des coordonnées X, Y, Z dans les fichiers LAS et LAZ.
--aabb	Cube de sélection sous la forme « minX minY minZ maxX maxY maxZ ». Si non fourni, il est automatiquement calculé
--incremental	Ajouter de nouveaux points à la conversion existante
--overwrite	Remplacer la conversion existante dans le répertoire cible

--source-listing-only	Créez un fichier sources.json mais pas d'octree.
--projection	Spécifiez la projection au format proj4.
--list-of-files	Un fichier texte contenant une liste de fichiers à convertir.
--source	Fichier source. Peut-être LAS, LAZ, PTX ou PLY
--title	Titre de la page
--description	Description à afficher dans la page.
--edl-enabled	Activer Eye-Dome-Lighting.
--material	RGB, ELEVATION, INTENSITY, INTENSITY_GRADIENT, RETURN_NUMBER, SOURCE, LEVEL_OF_DETAIL

ANNEXE 3 : La RV sur Unity avec oculus Rift

L'objectif de cet annexe est de donner un guide simple pour commencer avec unity 3D, et l'intégration d'oculus.

a. Créer un nouveau projet :

Étape 1 : Cliquez nouveau « New »

Étape 2 : Tapez un nom de projet et le dossier dans lequel est créé, choisissez l'option 3D.

Étape 3 : Finalement, cliquez sur créer projet « Create Project » (*Figure 6. 9*)

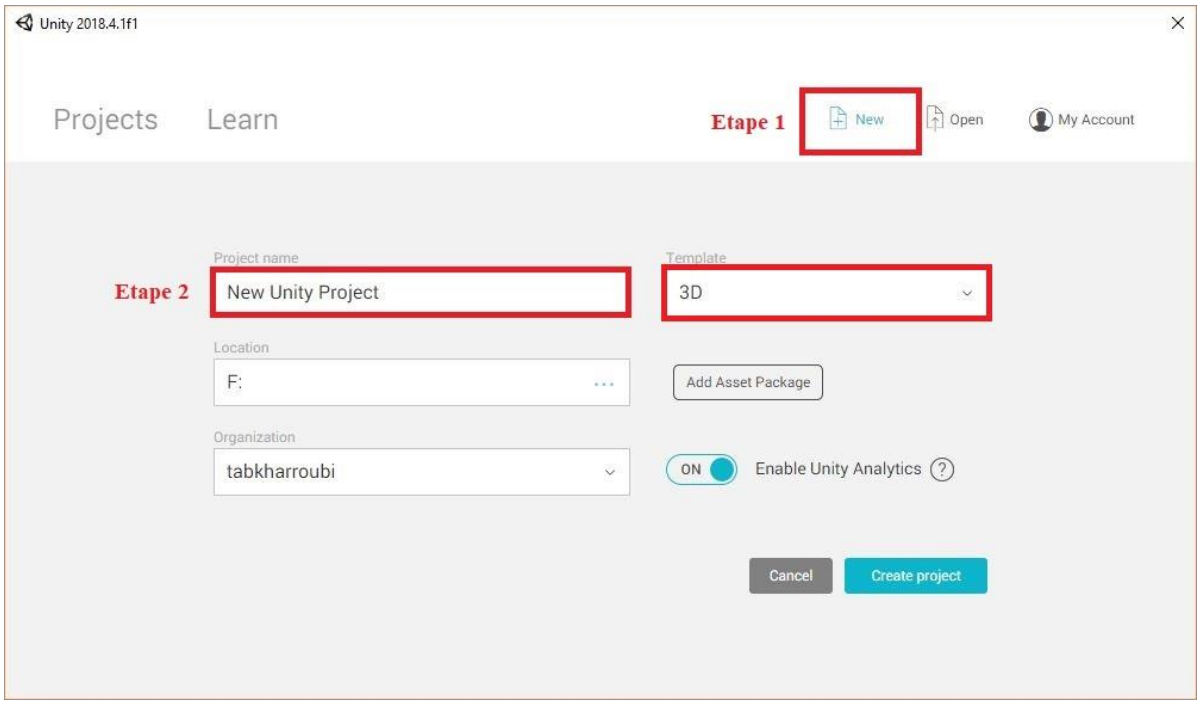


Figure 6. 9 *Création d'un nouveau projet sur unity*

b. Configuration RV

Le kit de développement logiciel peut être installé via le Centre de développement Oculus ou directement via l'asset store d'unity. Avant de télécharger le SDK, il est conseillé d'Installez Oculus Desktop App et Oculus Home. Ces applications donneront également l'occasion de confirmer si le système utilisateur est compatible et répond aux spécifications requises. Pour éviter les complications, il est recommandé d'activer des sources inconnues dans l'application d'exécution.

Un système rapide et fiable avec un fort et puissant processeur graphique (GPU) est un Nécessité de base pour que le Rift donne les résultats souhaités et fonctionne correctement. Les recommandations officielles sont :

• carte graphique GeForce GTX 970 or AMD Radeon R9 290 ou mieux

• CPU : Intel Core i5 4590 ou plus, RAM : 8GB ou plus

• Port Vidéo : HDMI 1.3, Port USB : 2 USB 3.0 ports, et Windows 7 SP1 ou plus récent

Après s'être assuré que le système remplit les conditions, l'Oculus Rift peut être configuré simplement en suivant les instructions simples sur l'application oculus (*Figure 6. 10*) qui apparaissent à l'écran une fois qu'il est branché et connecté.

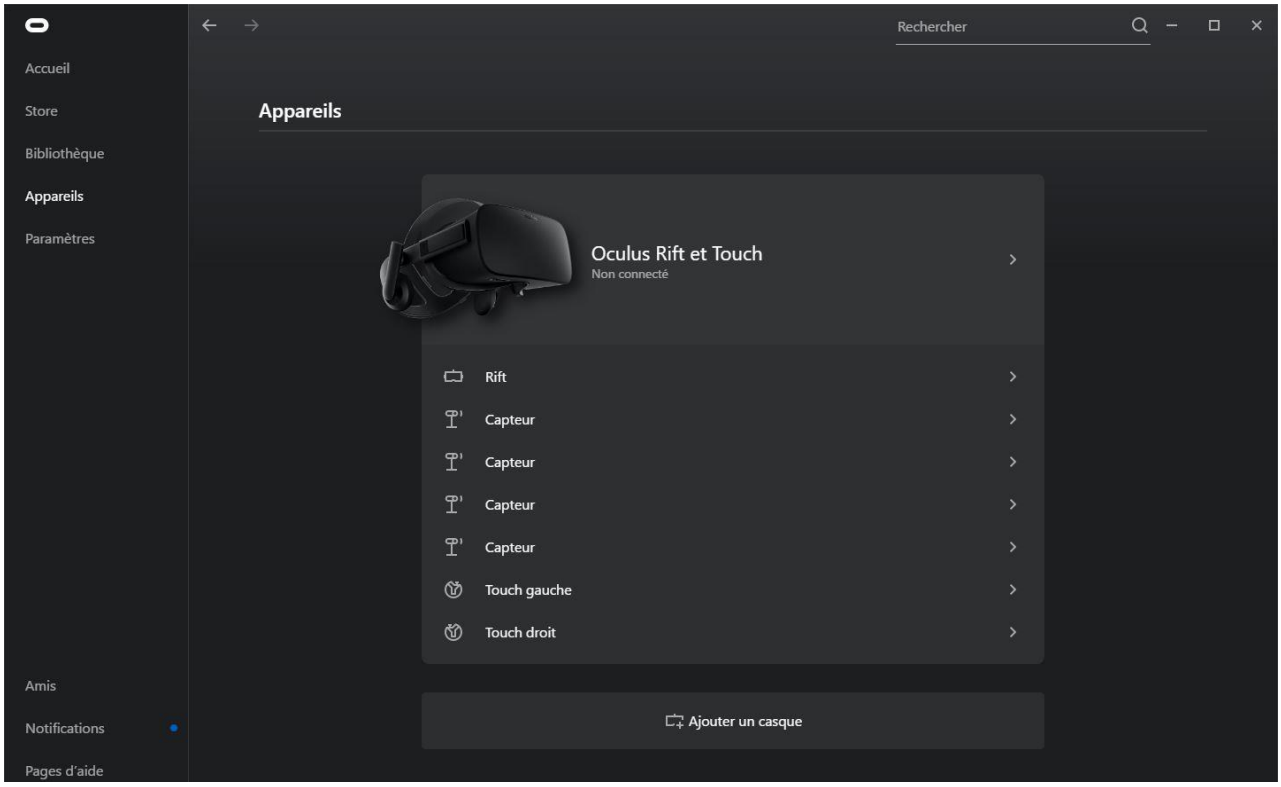


Figure 6. 10 *L'application Oculus pour la configuration du casque Oculus Rift*

Alors pour l'intégration de l'option de visualisation avec le casque oculus depuis Unity il faut suivre les étapes suivantes (*Figure 6. 11*) :

Étape 1 : Intégrez les utilités d'oculus via l'asset store, tapez « Oculus integration », puis téléchargez, et importez dans le projet.

Étape 2 : Dans la barre de Menu, cliquez sur File → Build settings → Player Settings → XR settings puis cochez l'option « Virtual Reality Supported » et choisissez Oculus.

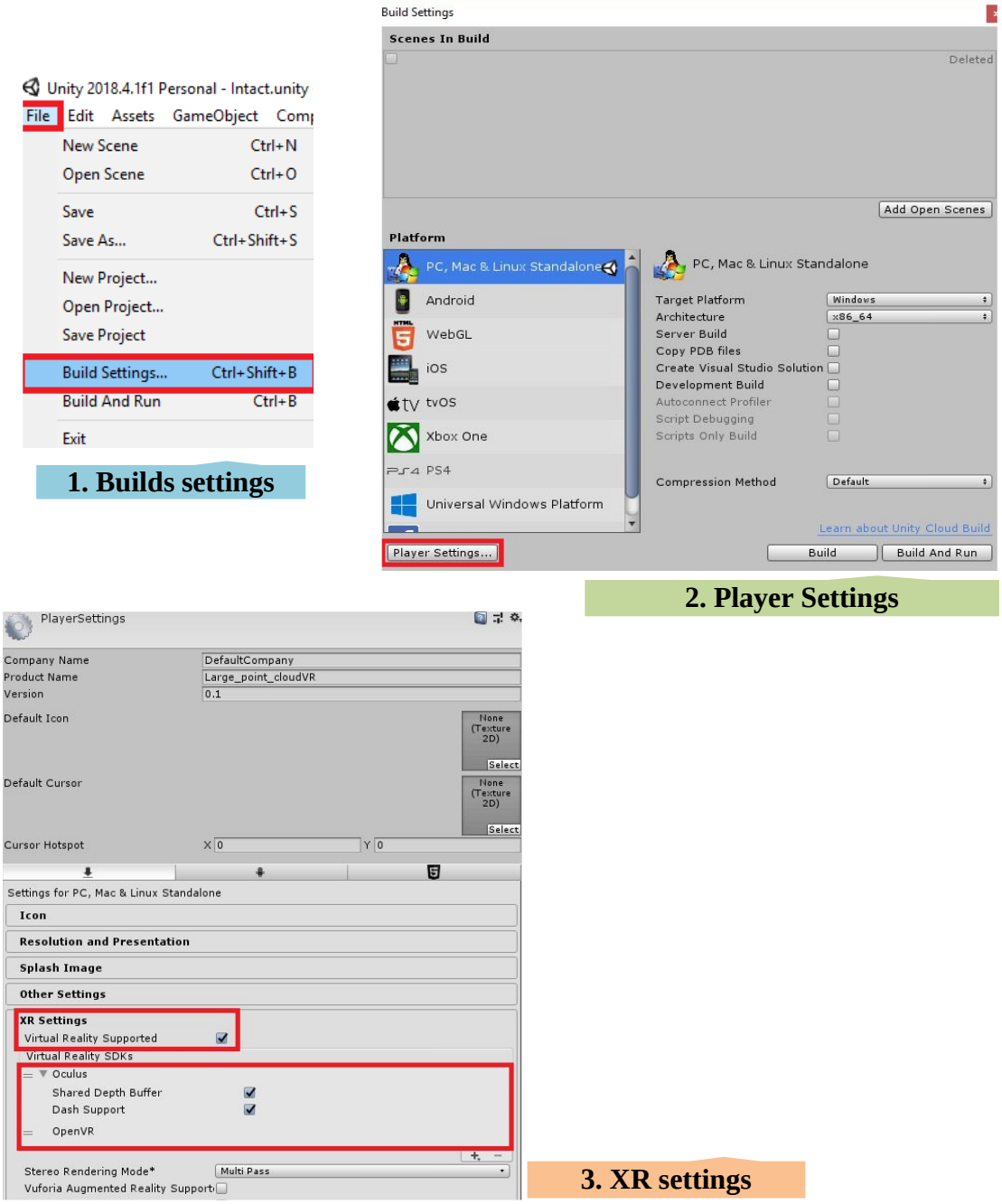


Figure 6. 11 *Étapes d'intégration de l'oculus*

c. Utilitaires Oculus pour Unity

Cette section fournit une vue d'ensemble du package Utilitaires, y compris de sa structure de répertoires, du prefabs et plusieurs scripts clés C #. Le contenu du dossier OVR dans OculusUtilities.unitypackage porte un nom unique et doit pouvoir être importé en toute sécurité dans un projet existant.

Le répertoire OVR contient les sous-répertoires suivants :

Tableau 6. 2 *Les sous-répertoires contenus dans le répertoire RV d'oculus*

Editor	Scripts qui ajoutent des fonctionnalités à Unity Editor et améliorent plusieurs scripts de composants C #
Materials	Les matériaux utilisés pour les composants graphiques dans le package Utilities, tels que l'Affichage graphique.
Meshes	Maillages requis par certains scripts *OVR*, tels que TrackerBounds.
Prefabs	Les principaux préfabriqués Unity utilisés pour fournir la prise en charge de la réalité virtuelle à une scène Unity : *OVRCameraRig* et *OVRPlayerController*.
Scenes	Exemples de scènes illustrant des concepts communs (comme les modes de déplacements)
Scripts	Fichiers C # utilisés pour lier l'ensemble de la structure VR et des composants Unity. Beaucoup de ces scripts fonctionnent ensemble dans les différents préfabriqués.
Textures	Images asset requis par certains composants de script.

Les prefabs utilisé dans le présent travail sont *OVRCameraRig* et *OVRPlayerController*. Pour les utiliser, il suffit de glisser-déposer l'un des préfabriqués dans la scène.

OVRCameraRig est une caméra RV personnalisée qui peut être utilisée pour remplacer la caméra Unity standard dans une scène. Glisser un *OVRCameraRig* dans la scène pour pouvoir commencer à regarder la scène avec Rift.

Alors que l'OVRPlayerController s'agit essentiellement d'un préfabriqué OVRCameraRig associé à un simple contrôleur de caractères. Il comprend une capsule physique, un système de mouvement, un système de menus simple avec rendu stéréo des champs de texte et un composant réticulé.

✓ Configuration de la caméra (OVRCameraRig et caméra principale)

Pour l'intégration de la caméra permettant l'affichage sur le casque oculus, OVRCameraRig est accessible via : Dans votre onglet Actifs, accédez à OVR → Prefabs (ou Oculus → VR → Prefabs), dans le dossier Prefabs, cliquez et faites glisser OVRCameraRig dans votre scène.

Maintenant, votre projet fait quelque chose de drôle. Vous venez de configurer un ensemble complet de caméras dans votre scène, mais la caméra principale faisait déjà des choses similaires. Nous devons donc supprimer la caméra principale d'origine pour l'empêcher d'interférer avec votre OVRCameraRig nouvellement importé.

Recherchez votre onglet hiérarchie, recherchez les mots «Caméra principale», sélectionnez la caméra principale et supprimez-la.

✓ Configuration du contrôleur tactile Oculus

Nous avons mis en place les caméras, qui représentent les yeux du joueur. Essentiellement, le casque Oculus Rift fait maintenant partie de ce projet. Maintenant, nous devons configurer les contrôleurs, qui représentent les mains du joueur.

Pour permettre au joueur d'utiliser ses mains d'Avatar Oculus (*Figure 6. 12*), recherchez l'onglet Ressources et accédez à OVRAvatar → Contenu → Préfabriqués (ou Oculus → Avatar → Contenu → Préfabriqués).

Figure 6. 12 *Les contrôleurs du local avatar*

Maintenant que nous avons les contrôleurs dans la scène, votre lecteur pourra interagir avec les objets de l'application que vous créez. Mais nous devons encore rendre les contrôleurs visibles en tant que mains d'avatar.

Commencez par cliquer sur l'objet de jeu Local Avatar dans votre onglet Hiérarchie. Ensuite, jetez un coup d'œil à votre onglet Inspecteur. Sous le composant OVR Avatar (Script) de l'onglet Inspecteur, recherchez la propriété "Démarrer avec les contrôleurs" et assurez-vous qu'elle est sélectionnée.

✓ Cliquez sur le bouton « Play » et profitez

Récapitulatif des étapes techniques :

- Aller au menu → Build settings → Player Settings → XR Settings et vérifier que l'option Virtual Reality Supported est cochée. Vérifier que « Oculus » apparaît dans la liste sinon cliquer sur le symbole plus ci-dessous et sélectionner-le dans le menu déroulant qui apparaît.
- Aller dans la barre the Unity Asset Store (Cliquer Window → Asset Store) sur Unity et chercher Oculus Integration dans la barre de recherche en haut. télécharger et importer Oculus Integration dans le projet.
- À la fin de l'importation, une invite indiquant que la mise à jour de l'API est requise apparaît. Cliquer sur « I made a backup, go ahead ! » puis Accéder à OVR → Prefabs dans l'assets tab.
- Cliquer et glisser le OVRCameraRig prefab dans la scène.
- Supprimer la caméra principale de la scène dans l'onglet hiérarchie car elle interférerait avec la OVRCameraRig.
- Pour activer l'Oculus Avatar avec les mains et utiliser les contrôleurs, accéder à OvrAvatar → Content → Prefabs dans l'assets tab.
- Cliquer et glisser le LocalAvatar prefab dans la scène.
- Sélectionner LocalAvatar à partir de la table d'hiérarchie « Hierarchy tab », puis chercher dans l'inspector tab. Chercher l'option « Start With Control » dans le composant « OVR Avatar(Script) », assurez-vous que la case à côté de celle-ci est cochée. Enregistrez la scène !
- Vous avez maintenant un projet prêt à développer pour Oculus Rift !

ANNEXE 4 : La table de classification

Les classes sont choisies d'une manière à être générales et englobantes de toutes les possibilités qui peuvent se présenter dans un nuage de points soit de l'intérieur des bâtiments ou bien de l'extérieur. Dans les scripts développés lors de ce travail, il y a une grande flexibilité d'ajout, de modification ou de suppression des classes.

Tableau 6. 3 *Les classes utilisées dans les différents nuages de points avec leurs numéros*

Indoor		Outdoor	
0	Floor	30	RoadSign
1	Ceiling	31	AdvertisingBoard
2	Wall	32	Banc
3	Beam	33	Bicycle
4	Column	34	BicycleStation
5	Window	35	BuildingFacades
6	Door	36	BusStation
7	Table	37	Car
8	Chair	38	Fence
9	Bookcase	39	FireHydrant
10	Sofa	40	LowVegetation
11	Board	41	Human
12	Clutter	42	Pole
13	OfficeForuniture	43	LightingPole
14	Cupboard	44	ManMakeTerrain
15	Lamp	45	Parasol
16	Table	46	Tree
17	TV	47	Truck
18	BathroomSink	48	Parking
19	Noise	49	Ground
20	Extinguisher	50	Roof
21	TrashCan	51	Hedge
22	Sign	52	Flag
23	Backpack	53	Water
24	Poster	54	Bed
25	Ladder	55	LandryMachine
26	Staircase	56	Sculpture
27	Toilet	57	Bredge
28	Miroir	58	

ANNEXE 5 : Scripts en C# et Shader

Les différents scripts développés dans ce travail ainsi que le projet entier est disponible sur Github dans le lien suivant : https://github.com/akharroubi/Point-clouds-classification-and-integration-in-VR-environment

ملخص

تقدم هذه الأطروحة في الهندسة الطوبوغرافية والعلوم الجيوماتيكية تطبيق في الواقع الافتراضي من أجل إظهار والتفاعل مع اسحب النقاط ضخمة في بيئة افتراضية. الحل مقترح يأتي في إطار ما بعد التصنيف ويستند إلى هياكل بيانات تتكيف مع إدارة البيانات الضخمة في الوقت الفعلي (عدة مليارات من النقاط). بشكل عام، تمثل السحب النقطية بيانات ثلاثية الأبعاد أساسية لتطوير عدد كبير من التطبيقات الثلاثية الأبعاد، خاصة تلك المتعلقة بالواقع الافتراضي. ومع ذلك، هاته البيانات لا تتضمن أي معلومات أو دلالات حول الأشياء التي تمثلها والتي تشكل المشهد الواقعي. وبالتالي، تتطلب السحب النقطية المعالجة والتصنيف من اجل لتصور الدلالي. من خلال دراستنا، نقدم عملية تصنيف لدمج معلومات على سحابة النقط. بعد ذلك، نقوم بتطوير نظام عرض سحابي واسع النطاق في الواقع الافتراضي ضمن البرنامج "Unity ". في هذا العمل، ينصب التركيز الرئيسي على دمج التصنيف في بيئة افتراضية وعلى إدارة هذه السحب الضخمة للنقاط، والتي تتجاوز عدة مليارات من النقاط، والتي لا يمكن تحميلها مباشرة في ذاكرة الوصول العشوائي"RAM". وبالتالي، يتم استخدام بنية بيانات من نوع "Octree" وتستخدم وخوارزميات خارج النواة "Out-Of-Core" للتحميل في الوقت الفعلي وبشكل مستمر، فقط النقاط التي تظهر في مجال رؤية المستخدم. نحن نقدم حلًا لتصور الواقع الافتراضي من خلال واجهة مستخدم تتيح التفاعل مع السحابة النقطية وتغيير المعلمات وطرق العرض. يوفر هذا الحل المتكامل أيضًا وضعًا للحركة في سحابة النقاط لضمان تجربة غامرة للمستخدم.

الكلمات المفتاح: السحب النقطية، التقسيم، التصنيف، Potree، Unity، Octree

تأطير	المترشح	تأطير
أ. حجي رفيقة (1)	خروبي عبد الرزاق	**أ.** رولون بيلان (2)
		د. فلورون بوكس (2)

(1) قسم الجيوذيزيا والطوبوغرافية، معهد الحسن للزراعة والبيطرة، الرباط

(2) وحدة الجيوماتيك، جامعة لييج، لييج، بلجيكا

المملكة المغربية

ROYAUME DU MAROC

معهد الحسن الثاني
للزراعة والبيطرة

INSTITUT AGRONOMIQUE
ET VÉTÉRINAIRE HASSAN II

مشروع نهاية الدراسات لنيل دبلوم مهندس دولة في الطبوغرافية

تصنيف وإدماج سحب النقاط ثلاثية الأبعاد في بيئة الواقع الافتراضي

قدم للعموم ونوقش من طرف:

خروبي عبد الرزاق

أمام اللجنة المكونة من:

الأستاذة كنزة آيت القاضي	رئيسة	معهد الحسن الثاني للزراعة والبيطرة
الأستاذة رفيقة حجي	مقررة	معهد الحسن الثاني للزراعة والبيطرة
الأستاذ رولون بيلان	مقرر	وحدة الجيوماتيك، جامعة لييج
الأستاذ طيب تشلايت	ممتحن	معهد الحسن الثاني للزراعة والبيطرة
الأستاذ أحمد طمطاوي	ممتحن	معهد الوطني للبريد والمواصلات

شتنبر 2019

العنوان: ص. ب 6202 الرباط المعاهد الرباط – المغرب
الهاتف: 59 / 58 17 77 0537 (00 212)
الفاكس: 45 58 77 0537 (00 212)
موقع الأنترنت:http://www.iav.ac.ma

Adresse : Madinat Al Irfane, B.P. 6202. Rabat – Maroc
Tél : (00 212) 0537 77 17 58/59
Fax : (00 212) 0537 77 58 45
Site web : http://www.iav.ac.ma