Numérisation partielle de bâtiments et gestion des levés complémentaires

Clément Hamdi

HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD

le cnam École supérieure d'ingénieurs géomètres et topographes esgt

CONSERVATOIRE NATIONAL DES ARTS ET MÉTIERS

ECOLE SUPÉRIEURE DES GÉOMÈTRES ET TOPOGRAPHES

MEMOIRE

Présenté en vue d'obtenir

Le DIPLOME D'INGENIEUR CNAM

SPÉCIALITÉ : Géomètre et Topographe

Par

Clément HAMDI

Numérisation partielle de bâtiments et gestion des levés complémentaires

Soutenu le 4 septembre 2024

JURY

Monsieur Ousmane DIOUF	Président du jury
Monsieur Bertrand CANNELLE	Maitre de stage
Monsieur Pierre KERVELLA	Enseignant référent

Remerciements

Je tiens à remercier l'ensemble des personnes m'ayant permis de réaliser ce travail de fin d'études.

Tout d'abord, je remercie mon maitre de stage, monsieur Bertrand CANNELLE, pour sa disponibilité et son aide tout au long de ces 6 mois de stage. La mise en place de réunions hebdomadaires m'a été extrêmement bénéfique et a rendu le travail très agréable.

Je remercie également l'ensemble du département EC+G (Environnement Construit et Géoinformation) pour leur accueil au sein de leur bureau.

Je tiens aussi à remercier mon enseignant référent, monsieur Pierre KERVELLA, pour son suivi et ses conseils durant cette période.

Je remercie l'ensemble de mes amis de l'ESGT, et notamment Julia et Guillaume, pour leur accueil au cours de ces nombreux weekends.

Pour finir, je remercie ma famille et plus particulièrement mes parents pour leur soutien sans faille tout au long de mes années d'études.

Glossaire

Apache : Logiciel permettant la création d'un serveur web.

BCF (BIM Collaboration Format) : Format de fichier permettant de faire un commentaire sur des objets d'un modèle BIM.

BIM (Building Information Modeling) : Méthode de gestion des projets de construction, basée sur une maquette numérique 3D.

E57 : Format utilisé pour stocker des nuages de points.

EPSG : European Petroleum Survey Group

FOV (Field Of View) : Champ de vision d'un élément (œil, caméra…)

GPKG (GeoPackage) : Format de données géospatiales.

HTML (HyperText Markup Language) : Langage de balisage utilisé pour la création de pages web.

ICP (Iterative Closest Point) : Algorithme utilisé dans le but de mettre en correspondance deux jeux de données**.**

IGN : Institut national de l'information géographique et forestière.

Javascript : Langage de programmation utilisé par les développeurs pour concevoir des sites web interactifs.

LAS (LASer) : Format stockant des données de nuages de points LIDAR.

LAZ : Version compressée du format de fichier LAS.

LiDAR (Light Detection And Ranging) : Système de mesure de terrain par balayage laser.

OGC : Open Geospatial Consortium

PDAL : Point Data Abstraction Library

Python : Langage de programmation.

SQL (Structured Query Language) : Langage informatique normalisé servant à exploiter des bases de données relationnelles.

WebGL : Spécification d'interface de programmation de 3D dynamique.

Table des matières

Introduction

En Suisse, comme en France, le secteur de la construction fait face à de nombreux défis :

- Raréfaction des ressources foncières.
- Objectif de neutralité carbone.
- Contraintes normatives en constante évolution.

Ce travail s'inscrit dans le projet SwissRenov, lancé à l'initiative de la Confédération Suisse[1]. Le projet SwissRenov a pour objectif final la rénovation des bâtiments, en particulier sur des friches industrielles. Les friches étant des environnements complexes une évaluation minutieuse des meilleures options est donc nécessaire.

L'objectif est de créer un modèle reproductible de transformation circulaire de la chaîne de valeur de la construction, en mettant l'accent sur les friches industrielles comme source essentielle de ressources. Le modèle sera développé dans la région du canton du Jura (voir figure 1), qui dispose de friches industrielles bien répertoriées, et servira de démonstrateur pour une diffusion à l'échelle nationale. Cette approche globale permettra une prise de décision éclairée, favorisant la valorisation des ressources et la construction durable en Suisse. Le projet SwissRenov réunit un grand nombre d'entreprises et d'écoles réparties dans toute la Suisse, et notamment le département Environnement Construit et Géoinformation (EC+G) présent au sein de la HEIG-VD. Ce projet va permettre notamment de préserver l'environnement. En effet, « *la réhabilitation de friches industrielles permet la décontamination des sols, la réduction des émissions polluantes et la préservation des ressources naturelles, contribuant ainsi à la protection de l'environnement. En réhabilitant les friches industrielles, nous évitons la conversion de terres vierges ou de zones naturelles, préservant ainsi la biodiversité et les écosystèmes existants. L'innovation systémique incite à la récupération et la réutilisation des matériaux et des ressources présentes dans les friches industrielles, réduisant ainsi les déchets et minimisant l'empreinte écologique.* » [1]

Figure 1: Le canton du Jura

[1] Flagship soutenu par Innosuisse

Ce projet est divisé en différents sous-projets que voici :

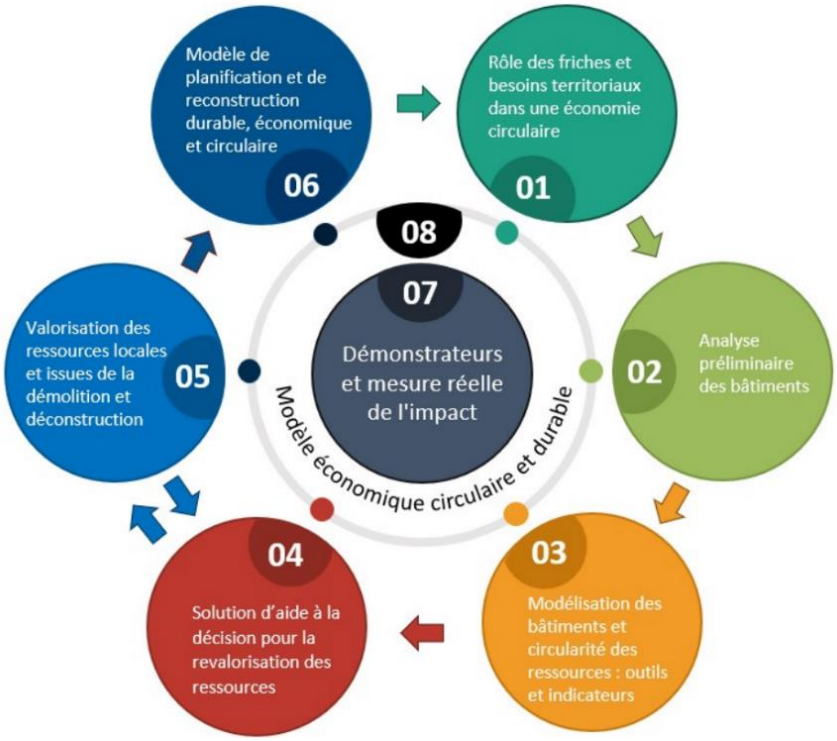


Figure 2: Les sous-projets de SwissRenov [1]

Le travail présenté dans ce rapport s'inscrit dans le cadre du sous projet SP3 « Modélisation des bâtiments et circularité des ressources : outils et indicateurs » visible dans la figure 2. Ce sous projet a pour objectif de « *développer un outil innovant et évolutif permettant de caractériser et de modéliser de manière simple et rapide un bâtiment existant sous l'angle de la circularité*. » [1]

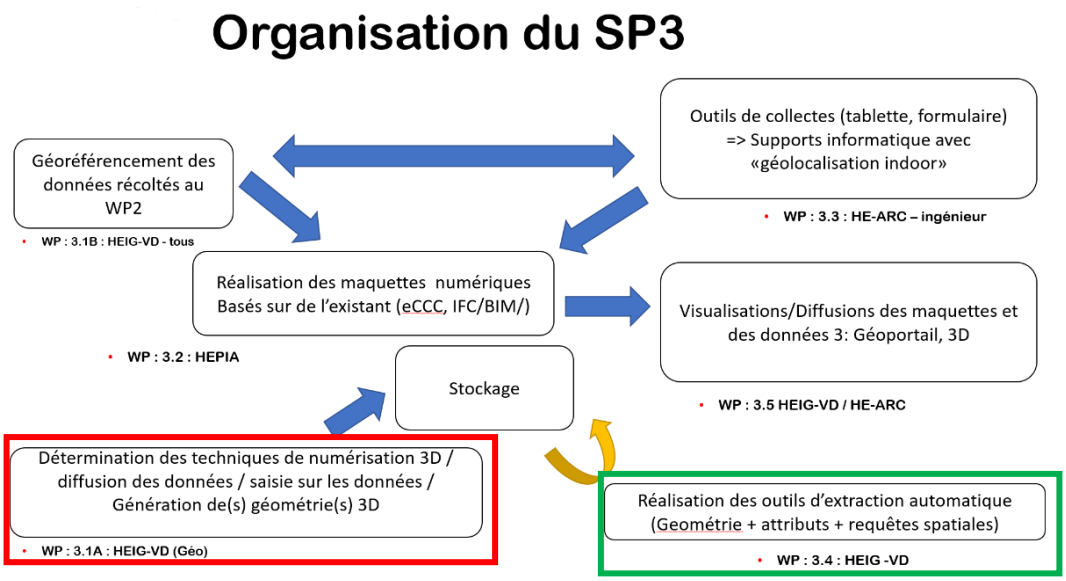


Figure 3: Les objectifs du SP3

Cette étude se situe à l'articulation entre les WP 3.1A (Rouge) et WP 3.4 (Vert) (voir figure 3).

Pour répondre aux attentes du projet SwissRenov, la question suivante vient à se poser :

Comment gérer, interagir et mettre à disposition des données 3D issues de sources hétérogènes ?

Ce travail de fin d'études a donc pour objectif de trouver une méthode pour gérer l'historique des données et de les diffuser. Une modélisation informatique sera mise en place (aussi bien dans le modèle de données que dans le stockage des fichiers) permettant :

- La gestion des différentes sources de levés.
- La gestion de l'historique des états de calcul.
- La gestion des levés complémentaires sur une même zone avec des caractéristiques différentes (résolution, échantillonnage…).

Ce mémoire fait la synthèse des recherches réalisées pour traiter cette question.

Je présenterai tout d'abord l'ensemble des données utilisées pour réaliser les expérimentations (données « tests »), mais également les différentes contraintes imposées par les données terrain, qui seront celles issues des relevés réalisés dans le cadre du projet SwissRenov.

Par la suite, la seconde partie aura pour but de trouver la plateforme de diffusion des données la plus adaptée à des personnes n'ayant pas, ou peu, de connaissances sur les nuages de points et leur manipulation. Ici, on recherchera principalement des outils (mesures, coupes, export) simples à utiliser.

Après avoir choisi la plateforme de visualisation, une troisième partie décrira la mise en place d'une base de données, avec le choix des métadonnées relatives aux nuages de points qu'il serait intéressant d'intégrer, puis la plateforme de création de la base de données, et enfin le choix du format des données spatiales. Nous verrons l'automatisation via Python de la création de la base de données. En raison de l'utilisation de codes Python, il est important de préciser que l'entièreté de ce travail a été réalisée sous Windows.

Nous verrons finalement les perspectives d'évolutions dans le cadre de ce travail de fin d'études.

I. Les données

Cette première partie va décrire l'ensemble des données utilisées pour réaliser les expérimentations (données « tests »). Dans un second temps, nous verrons les contraintes à prendre en compte dans le cadre des acquisitions du projet SwissRenov, notamment avec des données terrain, hétérogènes, ayant des caractéristiques différentes (sources, résolution, nature).

I.1 Les données « tests »

Pour confronter les différentes plateformes, des données tests ont été utilisées. Les données issues des friches n'étant pas encore disponibles, j'ai eu la possibilité d'utiliser un nuage de point test, issu du relevé du pont de Ballaigues, dans le canton de Vaud. Ces données ont été acquises en mars et octobre 2023, à l'aide d'un scanner RIEGL VZ 2000i. J'ai également pu utiliser les données mises à disposition par SwissTopo dans le cadre de leur mission LiDAR swissSURFACE3D, « *qui décrit tous les éléments naturels et construits de la surface de la Suisse sous forme d'un nuage de points classifiés. Acquises par LiDAR aéroporté, ces données bénéficient d'une densité spatiale et d'une précision élevée.* » [2] Afin de vérifier le fonctionnement de certains codes Python, j'ai dû utiliser des données fictives qui ne correspondent pas à des données réelles, mais qui m'ont permis de valider le fonctionnement de ces codes.

D'autres données plus conséquentes pourront être utilisées dans le cadre des expérimentations des codes Python, notamment des nuages acquis dans le cadre de travaux de recherche en cours au sein de la HEIG-VD. L'utilisation de ces données permettra de confronter les différents éléments, notamment les codes Python, à des données plus lourdes. Le format des nuages de points utilisé est le LAS, qui, « *étant modulable* », permet de *« rapidement et efficacement utiliser des nuages de points volumineux*. » [3]. Le format LAS est un format contenant un grand nombre d'informations et est très utilisé dans le domaine des données 3D.

L'ensemble de ces données va permettre d'évaluer d'un côté les plateformes de visualisation et leur capacité à afficher des données volumineuses, et de l'autre les codes Python. Elles vont également permettre de simuler une hétérogénéité des sources, qui est un aspect fondamental dans le projet SwissRenov.

I.2 Les données terrain

Les données acquises pendant les campagnes de relevé sur les friches seront proches des données test, mais l'échelle et la quantité de données sera plus grande. Cette grande quantité de données impose donc d'avoir un suivi rigoureux.

Dans certains cas, le même lieu peut être relevé plusieurs fois, à des dates différentes, avec des capteurs ou des résolutions différentes, peut-être même par des entreprises différentes.

Il serait en effet très avantageux de n'avoir qu'un seul nuage de toute la zone, mais ce n'est pas forcément adapté dans le cas des friches. Il a donc fallu réfléchir à la meilleure méthode pour recueillir et stocker les données.

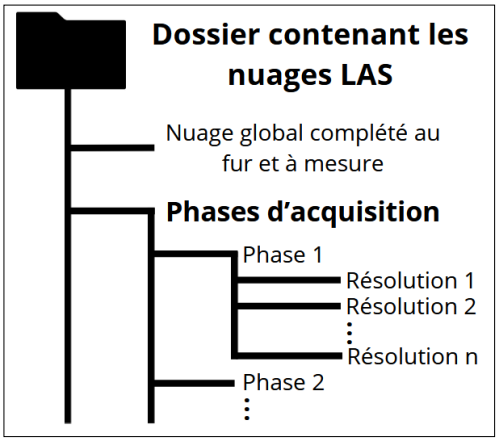


Figure 4: Arborescence pour le stockage des nuages

Lors des relevés, il est possible qu'une zone soit acquise avec une résolution différente du reste du relevé, ce qui a également été pris en compte dans le stockage des données présenté à la figure 4. De cette manière, le nuage global de la zone serait disponible et complété au fur et à mesure de l'avancée du chantier (voir figure 5), tandis que les zones acquises au cours du temps seront disponibles individuellement avec des résolutions différentes le cas échéant (voir figure 6). De cette façon, il sera possible de ne charger seulement des parties d'une friche pour la visualiser, ce qui permettra de ne pas charger l'ensemble de la friche et donc un trop grand nombre de points. En effet, si l'utilisateur souhaite effectuer des mesures dans un bâtiment en particulier, il aura la possibilité de ne charger que le nuage contenant ce bâtiment. La suite des expérimentations présentées dans ce mémoire, seront pour la majorité, réalisés avec les données « tests » du pont de Ballaigues.

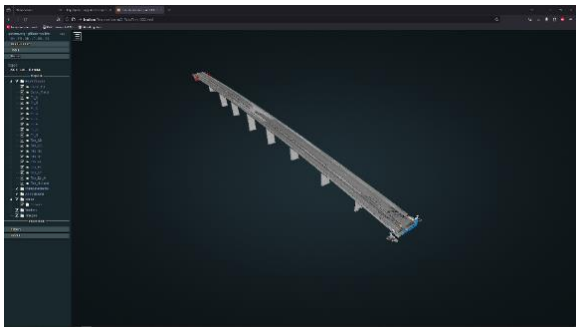


Figure 5: Nuage global du pont de Ballaigues

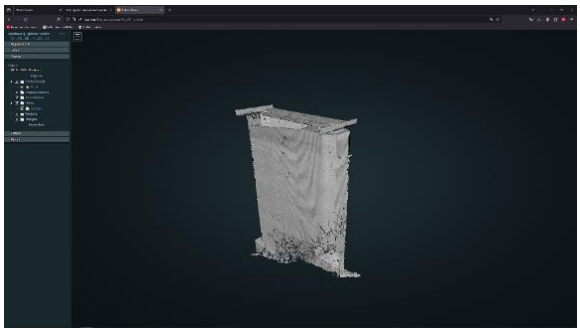


Figure 6: Nuage d'une pile du pont de Ballaigues

La partie suivante de ce rapport va concerner la recherche d'une plateforme de visualisation des données adaptée à des personnes peu familières avec la manipulation et l'interaction avec des nuages de points 3D.

II. Les plateformes de visualisation

L'objectif est de permettre à des acteurs n'ayant pas, ou peu, de connaissances dans les outils liés à la modélisation 3D d'interagir avec les nuages de points acquis, afin de pouvoir réaliser des mesures simplement, même sans aucune formation préalable. La première partie de cette étude s'est donc centrée autour de la recherche d'une plateforme permettant de visualiser et d'interagir avec les données.

II.1 Les logiciels

II.1.1 CloudCompare

CloudCompare [4] est un logiciel gratuit développé par Daniel Girardeau-Montaut, très utilisé dans le domaine de la gestion de nuages de points 3D.

Les premiers tests ont montré que cette plateforme met à disposition un grand nombre d'outils permettant d'interagir avec les données 3D, comme la possibilité de réaliser des coupes, des mesures, des alignements entre les nuages.

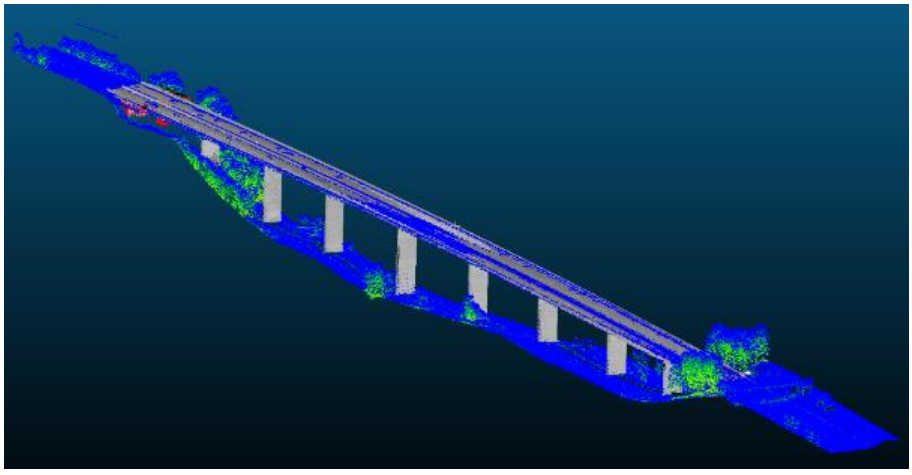


Figure 7: Alignement du pont issus de différentes sources

Cette solution s'est tout de même présentée comme étant assez compliquée à prendre en main pour des personnes non-initiées. En revanche elle pourrait s'avérer utile afin de retoucher certaines données avant leur diffusion, notamment lorsque des données de différentes sources doivent être utilisées.

Dans le cas ou des données doivent être rapidement visualisées et celles-ci ne peuvent pas être géoréférencées, CloudCompare permet d'aligner les nuages à l'aide d'une ICP ou peuvent être alignés manuellement dans l'attente d'un calcul précis des coordonnées. Sur la figure 7, le pont de Ballaigues acquis par la HEIG-VD (en gris) a été aligné sur le nuage issus des relevés LiDAR SwissTopo. CloudCompare est capable de charger la majorité des formats de fichier, mais aussi de les convertir ce qui peut être utile dans le cas ou des nuages sont issus de sources différentes.

CloudCompare ne sera pas la solution retenue pour diffuser les données mais ce logiciel pourra permettre dans certains cas d'apporter des modifications « de dernière minutes » ou de convertir les nuages d'un format à l'autre.

II.1.2 QGIS

QGIS [5] a également été testé, en effet ce logiciel gratuit permet de traiter des nuages de points 3D. Il est effectivement possible de charger des nuages de points grâce à la commande « Ajouter une couche nuage de points », puis de visualiser ce nuage avec l'extension qgis2threejs [6]. Cette extension a présenté de nombreux problèmes lors des tests, et n'inclue pas la possibilité de faire des mesures sur les nuages, ce qui ne sera pas compatible avec les attentes du projet. QGIS reste également assez compliqué à utiliser pour une personne n'ayant pas de connaissances dans la gestion des données 3D. QGIS sera cependant utilisé afin de visualiser les polygones représentant les emprises des nuages de points.

II.1.3 Comparaison

Les différents logiciels ont été comparés selon les critères suivant :

- Formats des données pris en charge
- Outils de mesures disponibles
- Possibilité d'insérer des annotations
- Possibilité de réaliser des coupes dans les nuages
- Possibilité d'exporter les coupes
- Possibilité d'exporter les données
- Possibilité de réaliser des ortho images

Le tableau 1 ci-dessous résume les différentes fonctionnalités testées afin de comparer les deux logiciels. Les deux logiciels précédents ont été testés avec les données du pont de Ballaigues. Il est ressorti de ces expérimentations plusieurs constats.

	QGIS [5]	CloudCompare [4]
Formats pris en charge	Large variété de formats vectoriels et raster	Large variété de formats de nuages de points et maillages
Outils de mesure	Angles, Points, Distances, Hauteurs, Diamètres, Surfaces, Volumes	Angles, Points, Distances, Hauteurs, Diamètres, Surfaces, Volumes
Annotations	Oui	Oui
Coupes	Oui	Oui
Export des coupes	Oui	Oui
Export des données	Oui	Oui
Ortho images	Oui	Non

Tableau 1: Tableau comparatif des logiciels

Tout d'abord, pour QGIS, il faut posséder un minimum de connaissances dans l'utilisation du logiciel et dans la gestion des bases de données. De plus, la manipulation des nuages de points est assez compliquée et ne fonctionne pas toujours en utilisant l'extension qgis2threejs, ce qui est compréhensible car ce logiciel n'est, à l'origine, pas fait pour interagir avec des nuages de points. L'ajout d'une couche nuage de points a été ajoutée avec la mise à jour 3.32 datant de juin 2023.[7]

Du côté de CloudCompare, l'interaction avec les nuages de points est assez facile à prendre en main. Nous pouvons voir cependant que les outils de mesures sont assez similaires entre les deux logiciels, mais CloudCompare est plus intuitif à utiliser.

Il est important de préciser que d'autres logiciels ont été testés comme Meshlab[2] et Cyclone3DR[3] mais ces derniers n'ont pas été retenus. Meshlab possède une interface assez complexe pour manipuler les nuages de points, tandis que Cyclone3DR nécessite un abonnement, et induit également une logistique supplémentaire (clés d'accès, stockage), donc des contraintes pour les utilisateurs.

[2] https://www.meshlab.net/
[3] https://leica-geosystems.com/fr-ch/products/laser-scanners/software/leica-cyclone/leica-cyclone-3dr

L’utilisation de logiciels induit des inconvénients notables pour l’utilisateur, notamment la nécessité de télécharger des éléments, même si certains logiciels peuvent être utilisés plus simplement grâce à des fichiers zip. L’installation nécessite également de l’espace de stockage, ce qui peut ajouter une contrainte supplémentaire. Les mises à jour et les compatibilités entre différentes versions peuvent aussi induire des problèmes.

La partie suivante traite des solutions disponibles en ligne et permettant également une interaction avec des nuages de points 3D.

II.2 Les solutions en ligne

Après une recherche au niveau des logiciels, j’ai décidé de me tourner vers des solutions disponibles en ligne. L’avantage de ces solutions est qu’aucune installation/téléchargement n’est nécessaire pour l’utilisateur final, et que les données sont accessibles de partout, avec seulement une connexion Internet. Les deux solutions en ligne qui ont été testées sont ATIS.Cloud et Potree.

II.2.1 ATIS.Cloud

ATIS.Cloud « *se définit comme une plateforme web collaborative qui permet de traiter, visualiser, mesurer et partager des nuages de points rapidement et facilement* » [8].

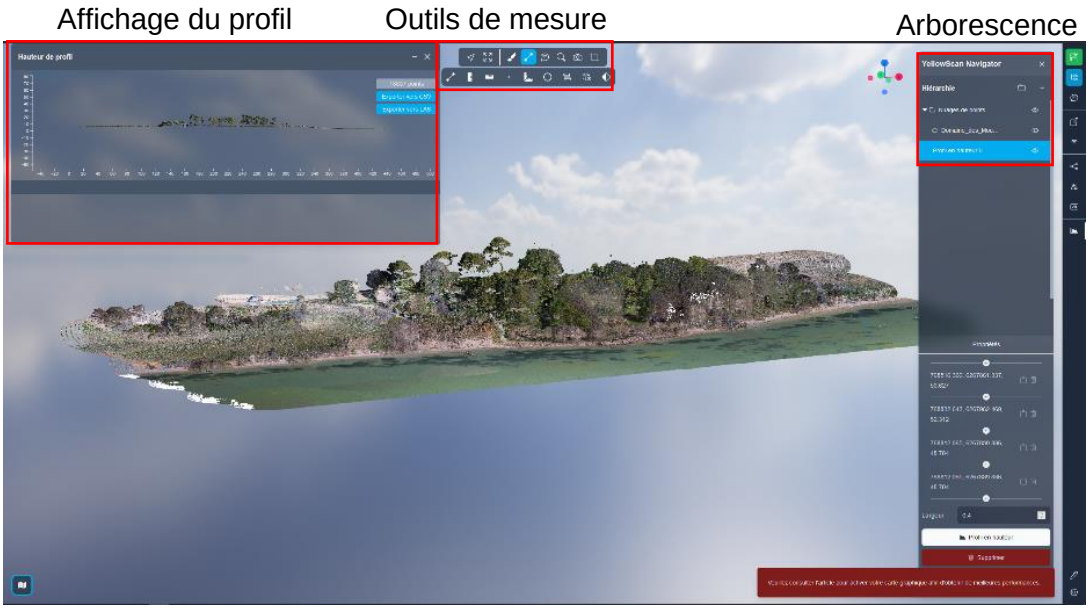


Figure 8: Interface d'ATIS.Cloud

ATIS.Cloud laisse la possibilité de réaliser des coupes, des mesures, d’importer différents formats, des modèles 3D, mais aussi les stations avec les photos (voir figure 8).

L’interface est simple et est accessible via des plateformes comme des smartphones, ce qui peut être intéressant pour un ouvrier sur le chantier qui n’a pas un ordinateur à disposition. Cette plateforme permet de diffuser des nuages de points mais aussi des données BIM et d’intégrer du BCF qui est un fichier qui stocke des annotations, ce qui pourrait être intéressant pour le futur du projet SwissRenov. ATIS.Cloud prend en charge un grand nombre de formats de nuages de points notamment le format LAS, qui est le format des données tests.

II.2.2 Potree

Potree [9] est un outil de visualisation web open source pour des nuages de points basé sur la technologie WebGL. Il permet de partager, d’afficher et d’analyser des jeux de données 3D volumineux. Un nuage de points peut être colorisé, affiché ou masqué dynamiquement selon des attributs choisis. Des mesures 3D (points, distances, polylignes, volumes) peuvent être effectuées directement dans le nuage de points. Des coupes peuvent également être réalisées et exportées aux formats CSV et LAS (voir figure 9). Potree est très utilisé afin de partager des nuages de points, comme par exemple dans le cadre de la mission LiDAR HD, conduite par l’IGN.

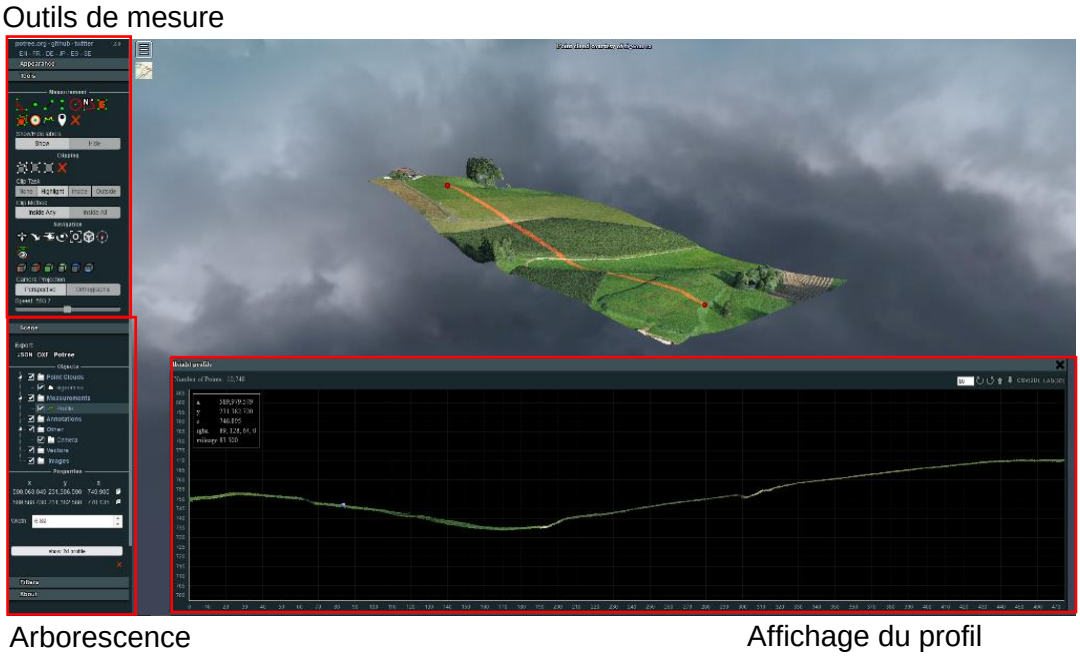


Figure 9: Interface de Potree

II.2.3 Comparaison

Les plateformes en ligne ayant rapidement été envisagées, j’ai pu procéder à une comparaison entre les deux principales solutions envisagées, ATIS.Cloud et Potree.

Les solutions en ligne ont été testées sur les mêmes critères que les logiciels, à la différence qu'il a été ajouté comme critère de sélection le coût, et l'accès aux codes HTML afin de générer des pages automatiquement.

Le principal avantage de Potree est la possibilité d'accéder facilement aux codes HTML des pages en ligne, qui sont générés à chaque conversion d'un nuage vers le format Potree, ce qui pourra permettre par la suite de générer ces fichiers HTML automatiquement afin de charger uniquement les nuages souhaités. ATIS.Cloud, malgré son interface assez ergonomique, ne permet pas autant de liberté que Potree en ce qui concerne les éléments à charger.

	ATIS.Cloud [10]	**Potree [9]**
Formats pris en charge	RCP, RCS (via ZIP) LAS LAZ E57 BPF	Formats classiques (LAS, E57, etc…) convertis au format Potree
Outils de mesure	Angles, Points, Distances, Hauteurs, Diamètres, Surfaces, Volumes	Angles, Points, Distances, Hauteurs, Diamètres, Surfaces, Volumes
Annotations	Oui	Oui
Coupes	Oui	Oui
Export des coupes	Oui	Oui
Export des données	Oui	Oui
Ortho images	Oui	Non
Accès aux codes HTML	Non	Oui
Coût	Abonnement	Gratuit

Tableau 2: Tableau comparatif des solutions en ligne

Le tableau 2 ci-dessus compare les différentes fonctionnalités des deux solutions en ligne. Le principal critère ayant permis de choisir une des solutions est l'accès aux codes sources des pages. En effet, dans un objectif d'automatisation et de sélection des nuages à afficher, les codes HTML de Potree étant facilement accessibles, il sera plus simplement possible d'afficher les données souhaitées. Sa gratuité est également un avantage notable.

La possibilité de pouvoir modifier les fichiers HTML va permettre de générer automatiquement des fichiers correspondant aux nuages choisis par l'utilisateur.

Il est important de noter que d'autres solutions ont été testées comme Open3D[4] et iTowns[5], mais elles n'ont pas été conservées. En effet, Open3D nécessite d'être manipulé avec Python, ce qui complexifie son utilisation. iTowns de son côté est une solution orientée vers une échelle plus grande que l'utilisation prévue pour le projet SwissRenov (échelle d'une ville par exemple). ITowns s'est également révélé être assez complexe, autant du côté de la mise en place que de l'utilisation.

La solution choisie pour diffuser les données sera donc Potree. Sa grande utilisation de la part de différents organismes permet de confirmer l'efficacité de cet outil.

II.2.4 PotreeConverter

Pour intégrer un nuage de points sur Potree, il est nécessaire de le convertir au bon format, à l'aide de PotreeConverter, qui est un logiciel open source permettant de convertir les nuages de points au format Potree. Pour convertir un nuage au format Potree, il suffit de lancer l'exécutable dans l'invite de commande Windows (voir figure 10) et de spécifier le chemin du nuage à convertir (input) ainsi que le chemin d'enregistrement du nuage converti (output).

```
C:\Users\clement.hamdi>PotreeConverter.exe <input> -o <outputDir>
```

Figure 10: Commande de conversion au format Potree

Le format Potree est construit comme suit :

- Le fichier HTML permettant d'ouvrir la page Web
- Un dossier contenant les librairies nécessaires
- Un dossier contenant (voir figure 11) :
    - Un fichier hierarchy.bin
    - Un fichier octree.bin
    - Un fichier JSON contenant les coordonnées des points du nuage

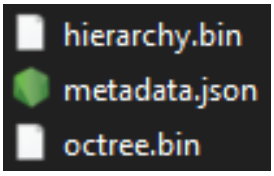


Figure 11: Dossier contenant les nuages

Nous allons par la suite nous intéresser à la structure du fichier HTML de Potree, visible à la figure 12 ci-dessous. Lors d'une conversion d'un nuage de points au format Potree, un fichier HTML va être créé.

[4] https://www.open3d.org/
[5] https://www.itowns-project.org/

Ce fichier va contenir différents éléments permettant d'afficher des éléments sur la page web, comme le nom de la page, la langue d'ouverture, ainsi que des éléments relatifs au style.

Le premier bloc du fichier HTML permet de générer le document de « type HTML » et de charger l'ensemble des librairies nécessaires à l'ouverture de Potree sur une page Web. Ces librairies sont stockées dans un dossier appelé « lib ». Le deuxième bloc concerne les éléments à charger à l'ouverture de la page, comme le FOV, le nombre de points maximum, les menus à ouvrir et la langue. Il est également possible de charger une description de la page qui s'affichera au-dessus du nuage.

Figure 12: Structure du fichier HTML

Le dernier bloc de code du fichier HTML permet de charger le nuage de points souhaité. Il faut spécifier le chemin du nuage (fichier « metadata.json » généré par la conversion). Il est possible de spécifier la taille et la forme des points du nuage, mais aussi la manière dont va s'afficher le nuage (RGB, élévation…). Il est possible de charger plusieurs nuages en dupliquant ces lignes de codes de chargement et en modifiant les chemins vers les nuages. Il est important de bien comprendre les différentes parties du code car l'objectif futur sera d'automatiser la génération de ce fichier HTML grâce à un code Python afin que l'utilisateur n'ai qu'à choisir les nuages à afficher en spécifiant leur identifiant. Le stockage des nuages de points en amont est très important afin qu'aucun problème n'apparaisse dans le chargement des nuages, car il faut y spécifier le chemin de stockage du/des nuage(s) à ouvrir.

II.2.5 Diffusion des données Potree

Durant cette étude, j'ai simulé la diffusion des données vers un serveur web local via Apache, en utilisant le logiciel XAMPP.

Les nuages, une fois convertis au format Potree, doivent être placés dans le bon dossier afin de pouvoir y accéder depuis la page web mise en place après l'activation du server local Apache. Dans ce cas les données ont été placées dans le dossier suivant : *C:\xampp\htdocs\Nomdedossier*.

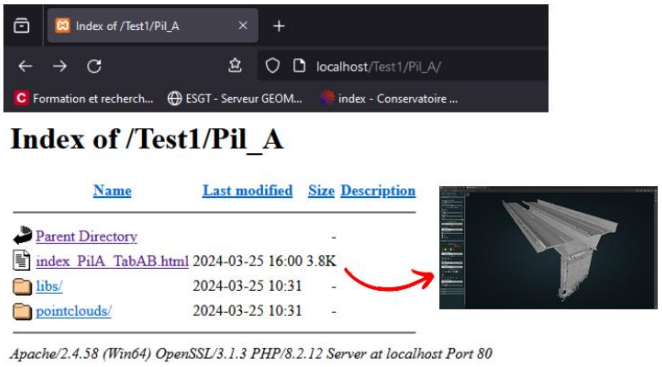


Figure 13: Page localhost

Une fois les nuages placés dans le dossier, il faut activer le serveur Apache à l'aide de la boite de dialogue XAMPP.

En accédant à l'adresse localhost/ Nomdedossier, il est possible d'ouvrir le fichier HTML de la page Potree. Sur la figure 13, nous pouvons voir que le dossier « Pil_A » est ouvert, contenu lui-même dans le dossier « Test1 », qui contient les librairies nécessaires au chargement de la page et le dossier contenant les nuages de points. En cliquant sur le fichier HTML, la page Potree va s'ouvrir en affichant les nuages.

La partie diffusion des données ou du moins choix d'une plateforme de visualisation simple pour les nuages de points est terminée. Potree permet de diffuser plusieurs types de données (nuages, shapefiles…) et de réaliser des mesures simples (coupes, distances, coordonnées) accessibles sans avoir de grandes connaissances dans la manipulation des nuages de points, ce qui est nécessaire pour les professionnels du bâtiment qui n'auront pas besoin de travailler en profondeur dans les nuages, mais devront tout de même accéder à un minimum d'informations. Potree s'est donc imposé comme étant la solution la plus appropriée pour cette utilisation. La manipulation des fichiers HTML/JavaScript est aussi assez simple et flexible.

Un modèle de base de données pouvant être utilisé dans le cadre du projet SwissRenov va être présenté dans la suite de cette étude. En interrogeant la base de données (requêtes SQL, zones tampons, buffers, sélection par clic…), l'utilisateur serait automatiquement renvoyé vers une page Potree contenant les nuages sélectionnés, lui permettant de réaliser des mesures sur cette zone.

III. La base de données

La troisième partie de ce rapport va concerner la base de données, en commençant tout d'abord par le choix des métadonnées appropriées afin de garantir une traçabilité et un historique des données. Plusieurs hypothèses de métadonnées ont été abordées durant cette étude et seront décrites. Nous reviendrons également sur le stockage des données du point de vue des formats et des conversions dans l'objectif de pouvoir afficher dans Potree des nuages déterminés, mais aussi dans le stockage des métadonnées. L'ajout dans la base de données des métadonnées et des emprises des nuages sera ensuite abordé, avant de traiter la question de la diffusion et de l'affichage de ces derniers.

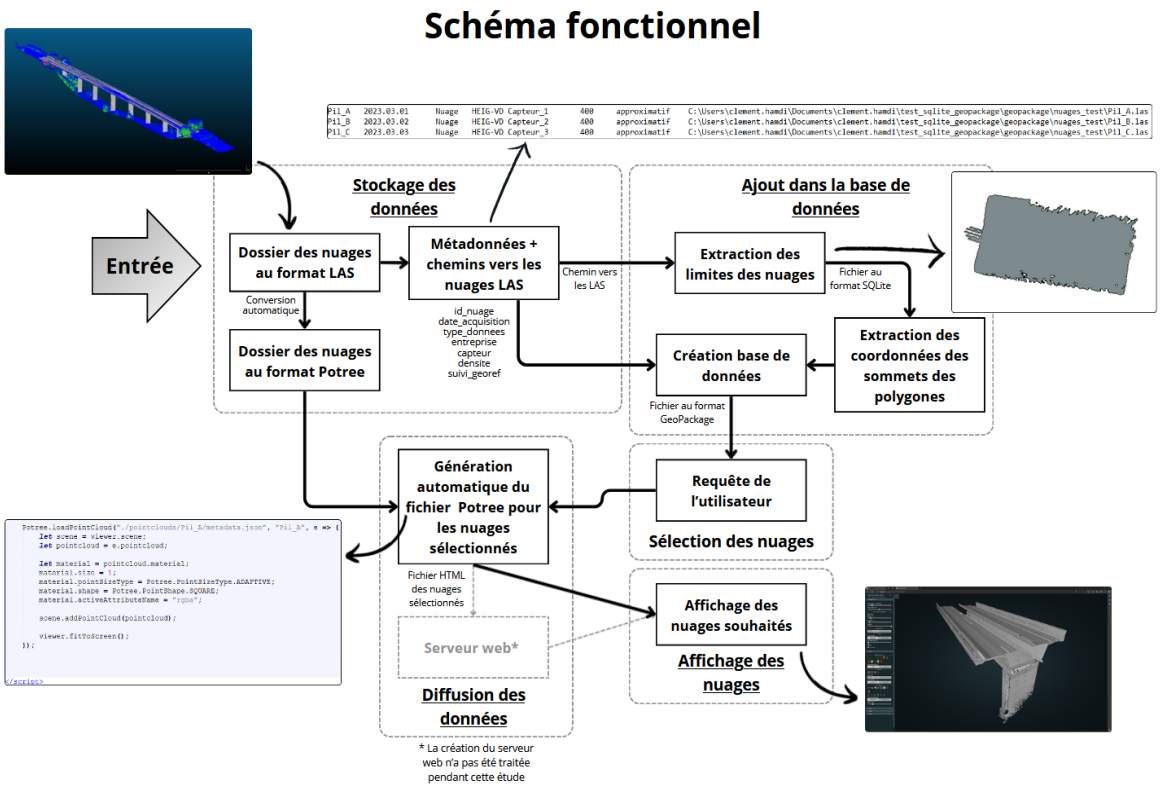


Figure 14: Schéma fonctionnel des étapes de création de la base de données

Le schéma fonctionnel présenté à la figure 14 ci-dessus, permet de visualiser le processus de traitement des données, tout d'abord dans leur conversion vers le format Potree, puis dans la création de la base de données. Une fois la base de données créée et les données converties, il sera possible de choisir les nuages à afficher dans la page Potree.

III.1 Les métadonnées

Il est important de rappeler ce que sont des métadonnées. Ce sont des éléments donnants des informations sur des données. Une définition concernant les métadonnées est donnée par le Commissariat à la protection de la vie privée du Canada comme étant « *une donnée qui fournit de l'information sur une autre donnée. Il s'agit en fait des renseignements qui sont générés lorsqu'on utilise la technologie et qui permettent de situer dans leur contexte (qui, quoi, où, quand et comment) diverses activités* » [11].

Avant de créer la base de données, il a fallu déterminer les métadonnées appropriées, à intégrer afin de donner des informations sur les données. Plusieurs hypothèses ont été envisagées.

III.1.1 Hypothèse « 1 table »

La première hypothèse était de créer une seule table (voir figure 15) contenant seulement les éléments relatifs aux données acquises. Dans ce cas-là, seules certaines informations liées aux nuages de points sont gardées.

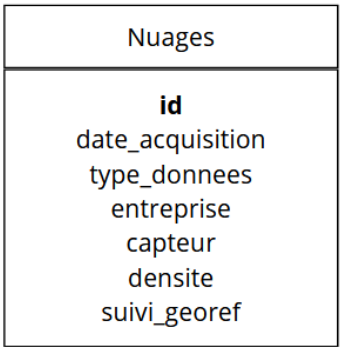


Figure 15: Hypothèse "1 table"

Les métadonnées choisies sont les suivantes :

- **id ou id_nuage** : Identifiant propre au nuage.
- **date_acquisition** : Date à laquelle les données ont été acquises (historique).
- **type_donnees** : Type de données acquises (cette étude se porte sur des nuages mais d'autres types de données pourraient être intégrées dans le futur).
- **entreprise** : Nom de l'entreprise ayant acquis les données (traçabilité/source).
- **capteur** : Type de capteur utilisé pour acquérir les données (scanner, appareil photo…).
- **densite** : Densité moyenne de points du nuage en nombre de points/cm².
- **suivi_georef ou georeferencement** : Suivi du géoréférencement. Dans la majorité des cas, les données sont directement géoréférencées lors des calculs, mais par précaution, si certaines ne le sont pas, il est envisagé de les aligner manuellement avec le reste des données, ce qui est peu précis mais permet tout de même une diffusion. Cette colonne pourra prendre deux valeurs, « précis » si le géoréférencement est réalisé par calcul ou « approximatif » s'il est réalisé manuellement, dans l'attente d'un calcul précis.

III.1.2 Hypothèse « 2 tables »

La deuxième hypothèse était de créer deux tables différentes (voir figure 16), dont une contenant les informations des nuages de points vus précédemment, et une autre contenant des informations sur les zones scannées comme les parcelles et les bâtiments.

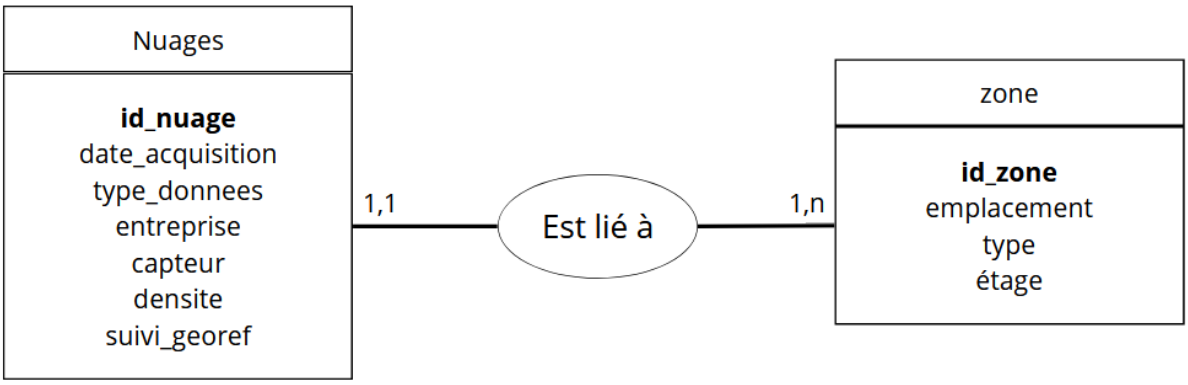


Figure 16: Hypothèse "2 tables"

Les éléments envisagés pour la table « zone » étaient les suivants :

- Types de bâtiments/zones (usine, étang, entrepôt, bureaux, forêt…)
- Commune
- Canton
- Parcelle cadastrale
- Activité avant la friche
- Nombre de bâtiments
- Date de construction des bâtiments
- Date de mise en friche
- Surface de la friche
- Surface des bâtiments
- Propriétaire
- Zone de règlement d'urbanisme (équivalent PLU)
- Coordonnées

Dans cette hypothèse, les éléments de la table « Nuages » ne peuvent être liés qu'à un, et un seul, élément de la table « Zone ». En effet, un nuage ne peut ici appartenir qu'à une seule friche. En revanche, les éléments de la table « Zone » peuvent être liés à plusieurs nuages.

La création de cette deuxième table a été inspirée par le projet Cartofriche, lancé en France par le CEREMA et qui recense les friches sur le territoire national dans le but « *d'aider au recensement les friches pour les qualifier et faciliter leur réutilisation.* » [12]. Après réflexion, il a été décidé de ne conserver que la première hypothèse lors de cette étude. La seconde table pouvant être envisagée par la suite lorsque la mise en place de la base de données des nuages

sera effectuée. Les tests réalisés dans la suite de ce rapport ne prendront donc en compte que la table « Nuages » présentée à la figure 15.

La partie suivante va traiter de la gestion de la base de données, et en premier lieu le choix de la plateforme permettant de gérer cette base de données. Plusieurs plateformes ont été confrontées afin de choisir la plus adaptée. Cette partie s'axera ensuite sur la création de cette base de données et notamment l'automatisation des différentes étapes.

III.2 La gestion des données

Nous allons parler ici de la méthode de stockage des métadonnées et des données 3D ainsi que du choix de la plateforme de gestion de la base de données. Plusieurs codes Python ont été créés afin d'automatiser les différentes étapes. Les différents codes seront présentés un à un, avant de voir un prototype final agrégeant l'ensemble des codes.

III.2.1 La gestion des métadonnées

En premier lieu, il a fallu déterminer comment alimenter de la façon la plus optimisée possible. Les recherches se sont très vites orientées vers deux méthodes couramment utilisées, le fichier texte (.txt) et le fichier Excel (.xlsx ou .csv). Il a été décidé d'utiliser, pour les phases de tests des différents codes, des fichiers .txt, qui se sont montrés plus simples à utiliser en raison du nombre réduit d'informations, mais la mise en place d'un fichier Excel contenant les métadonnées pourra être envisagée par la suite.

Les six premières colonnes correspondent aux métadonnées déterminées auparavant (id_nuage, date_acquisition, type, entreprise, capteur, densite, georeferencement), auxquelles nous ajoutons une colonne contenant les chemins physiques vers les nuages LAS, qui remplacent les coordonnées des sommets de limites des nuages.

De cette façon, l'ensemble des données utiles sont stockées dans un même fichier qui peut donc être chargé dans un code Python.

```
Pil_A   2023.03.01      Nuage   HEIG-VD Capteur_1       400     approximatif    C:\Users\clement.hamdi\Documents\clement.hamdi\test_sqlite_geopackage\geopackage\nuages_test\Pil_A.las
Pil_B   2023.03.02      Nuage   HEIG-VD Capteur_2       400     approximatif    C:\Users\clement.hamdi\Documents\clement.hamdi\test_sqlite_geopackage\geopackage\nuages_test\Pil_B.las
Pil_C   2023.03.03      Nuage   HEIG-VD Capteur_3       400     approximatif    C:\Users\clement.hamdi\Documents\clement.hamdi\test_sqlite_geopackage\geopackage\nuages_test\Pil_C.las
```

Figure 17: Fichier texte des métadonnées tests

Les données contenues dans ce fichier texte (voir figure 17) vont permettre de réaliser deux choses. Tout d'abord, les six premières colonnes vont être récupérées afin de compléter la base de données des informations concernant les nuages.

Puis la dernière colonne contenant les chemins physiques des nuages au format LAS va être utilisée dans le but de générer l'emprise de ces nuages. Ces étapes seront détaillées dans la suite de ce rapport.

III.2.2 La gestion des nuages de points

La gestion des nuages de points est également une partie importante en vue de générer la base de données. Lors de cette étude, les nuages utilisés sont uniquement au format LAS, afin de faciliter leur conversion vers le format Potree.

Chaque nuage acquis sera donc stocké dans un dossier contenant l'ensemble des nuages de la friche correspondante.

Les étapes se déroulent de la manière suivante :

- Les polygones des limites des nuages sont extraits des nuages LAS. Le code Python utilise les bibliothèques os [13] et subprocess [14], qui permettent respectivement d'utiliser des fonctions liées au système d'exploitation, et d'exécuter les commandes internes au code.
- Ensuite, une boucle est créée afin d'appliquer la commande de conversion vers le format Potree (Figure 10) sur chaque fichier contenant l'extension « .las » ou « .LAS ». Ces derniers sont utilisés dans un code Python (Annexe 1) de conversion vers le format Potree, et seront stockés dans un autre dossier.
- On retrouve donc enfin chaque nuage converti au bon format dans le dossier correspondant. Ils seront utilisés par la suite afin de générer le fichier HTML au format Potree permettant d'afficher les nuages souhaités.

Une fois la partie gestion, des nuages et des métadonnées, complétée, nous pouvons nous pencher sur la gestion de la base de données, avec tout d'abord le choix d'une plateforme adaptée.

III.2.3 La plateforme de gestion

Le choix de la plateforme de gestion de la base de données a été un élément important pour la suite de cette étude.

Il existe un grand nombre de plateformes permettant de gérer des données spatiales comme PostgreSQL[6] (avec l'extension PostGIS[7]) ou MapServer[8]. La plupart de ces plateformes nécessitent de mettre en place un serveur, mais comme expliqué précédemment, il a été décidé de réaliser les tests en local et de ne pas mettre en place de serveur, pour l'instant. Il a donc été décidé d'utiliser la plateforme SQLite Studio, qui va permettre de se dispenser de la mise en place d'un serveur, et qui est la plateforme utilisée dans le cadre des travaux dirigés au sein de la HEIG-VD. SQLite Studio est un logiciel open source gratuit de gestion de base de données, permettant notamment de lire des données spatiales. Lors du chargement de données spatiales dans SQLite Studio, ce dernier génère des tables incluant des informations sur les géométries, comme notamment les systèmes de références, les types de géométries, le codes EPSG. Les détails de ces tables seront décrits dans la partie suivante.

III.2.4 Le format des données spatiales

Il a été décidé d'utiliser le format GeoPackage, qui s'est avéré être le plus adapté pour gérer les données spatiales. Avant de présenter le code permettant de générer les données au format correspondant, il est important de savoir comment est structuré le format GeoPackage lorsqu'il est chargé dans SQLite Studio. Lorsqu'on charge un GeoPackage dans SQLite Studio, des tables, définies par une norme OGC, sont automatiquement créées afin de contenir la géométrie (voir figure 18). Le format GeoPackage pourra par la suitte être chargé sur d'autres plateformes comme PostgreSQL.

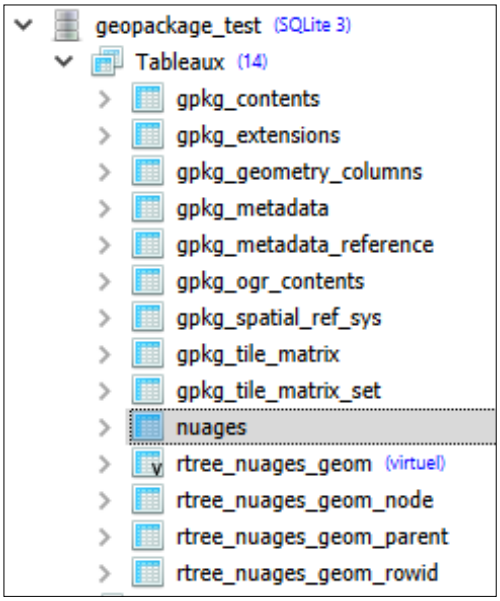


Figure 18: Tables générées dans SQLite Studio

[6] https://www.postgresql.org/

[7] https://postgis.net/

[8] https://mapserver.org/

Les tables générées sont les suivantes [15] :

- gpkg_contents : Contenu du GeoPackage.
    - table_name : Nom défini par l'utilisateur (clé primaire).
    - data_type : Type de données (tuiles…).
    - identifier : « Titre » lisible.
    - last_change : Date de la dernière modification.
    - min_x, min_y : Coordonnées max et min du contenu.
    - srs_id : Système de référence.

- gpkg_spatial_ref_sys : Contient les données des systèmes de référence.
    - srs_name : Description lisible du système de référence.
    - srs_id : Identifiant unique du système de référence (clé primaire).
    - organization : EPSG.
    - organization_coordsys_id : Identifiant du système de référence.
    - definition : Définition du système de référence.

- gpkg_geometry_column : Description de la géométrie.
    - table_name et column_name : Stockage des géométries.
    - geometry_type_name : Type de géométries.
    - srs_id : Système de référence.
    - z et m : les valeurs z sont pour la hauteur/l'élévation/la profondeur et les valeurs m sont réservées à d'autres types de mesures spécifiques au domaine).

- gpkg_tile_matrix_set : Décrit les noms d'un ensemble de matrices de tuiles (pyramide).
    - Table_name et srs_id : Références aux éléments de gpkg_content.
    - min_x, min_y : Coordonnées de la pyramide de tuiles.

- gpkg_tile_matrix :
    - table_name : Références aux éléments de gpkg_content.
    - zoom_level : Niveau de zoom.
    - matrix_width et matrix_height : Taille de la matrice de tuiles.
    - tile_width et tile_height : Taille de chaque tuile en pixels.
    - pixel_x_size et pixel_y_size : Taille de chaque pixel.

- gpkg_extension :
    - table_name : Nom de la table SQLite où l'extension s'applique.
    - column_name : Nom de la colonne SQLite où l'extension s'applique.
    - extension_name : Nom de l'extension requise.
    - definition : Liens, URL ou référence vers un document qui définit l'extension.

III.3 Agrégation de la base de données

III.3.1 Extraction des limites des nuages

La modélisation des zones relevées a été un point majeur dans la création de la base de données. On parle ici de la façon dont les utilisateurs vont voir les zones relevées et celles qui ne le sont pas.

Tout d'abord, il a fallu choisir dans quelle dimension (2D ou 3D) afficher les zones. Il a été décidé, pour cette étude, de modéliser les emprises des nuages en 2 dimensions, mais des hypothèses ont été émise afin d'intégrer une troisième dimension afin de gagner en clarté. Les expérimentations ont tout d'abord débuté sur QGIS, qui permet de charger des nuages de points. Les emprises ont tout d'abord été dessinées à la main, par la création d'un polygone de l'emprise du nuage. Par la suite, la commande QGIS « Limite » a été utilisée afin de générer automatiquement l'emprise du nuage. Sur la figure 19 ci-dessous, les limites sont extraites d'une partie du nuage du Pont de Ballaigues. Nous pouvons voir deux parties de tablier ainsi que trois piles.

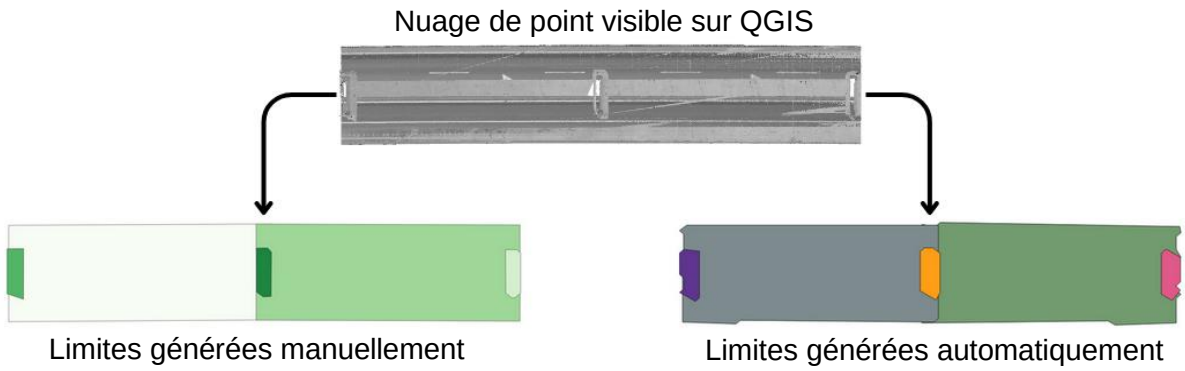


Figure 19: Génération des limites des nuages

Sur les limites générées manuellement, les nuages sont classés par date, du plus ancien (vert clair) au plus récent (vert foncé), afin de montrer comment il serait possible de visualiser et d'intégrer la composante historique dans l'affichage des données.

Au regard des résultats obtenus avec la génération automatique des limites, il a été décidé que cette méthode sera celle privilégiée en vue d'une automatisation sous Python, même si, dans certains cas, il sera tout de même possible de générer cette limite manuellement.

Pour générer la limite des nuages de points, QGIS utilise la librairie de gestion de nuages de points PDAL et notamment la commande boundary [16], visible sur la figure 20. Cette commande utilise le logiciel Hexer et la méthode de la tessellation hexagonale [17]. Cette méthode consiste à générer une grille d'hexagones sur le nuage de points. Le logiciel mesure ensuite le nombre de points du nuage dans chaque hexagone. Il est possible de donner un nombre limite de points dans les hexagones en dessous duquel les hexagones ne seront pas pris en compte.

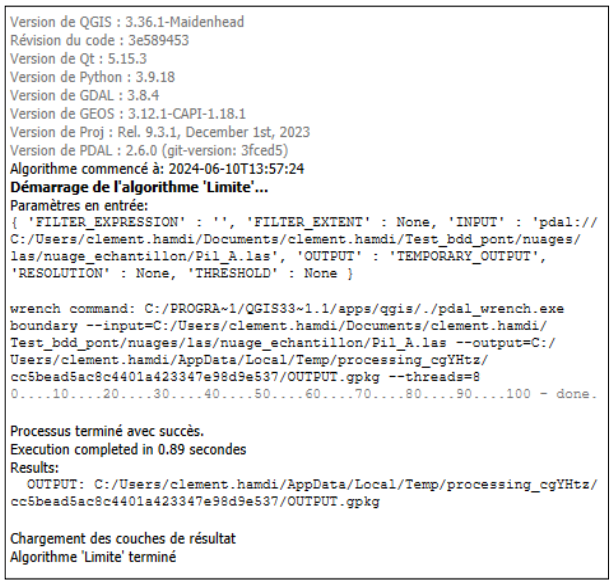


Figure 20: Fenêtre d'exécution de la commande "Limite"

Enfin, le logiciel connecte les centres des hexagones ayant un nombre de points supérieur au seuil et ainsi génère la limite du nuage de points.

Les paramètres qui vont permettre d'influer sur la limite sont les suivants :

- RESOLUTION : Correspond à la taille des cellules (hexagones) qui vont permettre de calculer la limite.
- THRESHOLD : Correspond au seuil minimal de points présents dans une cellule pour considérer la cellule comme occupée.

Une couche au format GeoPackage est créée en sortie et est donc exportable dans une base de données. La commande filter.hexbin [18] peut également être utilisée afin de générer la limite d'un nuage de points, et c'est cette commande qui sera utilisée lors de l'automatisation du processus d'extraction des limites. La commande utilise également tindex qui « *est une commande PDAL permettant de créer un jeu de données […] sous la forme d'un index de tuiles matricielles.* » [19]. La commande complète permettant de générer la limite d'un nuage est présenté à la figure 21.

```
command = (
    '"C:/Program Files/QGIS 3.36.1/bin/pdal.exe" tindex '
    '--filters.hexbin.threshold=2 create '
    '--tindex ./limitenuage.sqlite '
    '--filespec "C:/Users/clement.hamdi/Documents/clement.hamdi/test_sqlite_geopackage/'
    'test_limite_qgis/las_tests_elec/ligne_elec_test.las" -f SQLite'
```

Figure 21: Commande d'extraction de la limite d'un nuage

La commande va récupérer l'exécutable de PDAL présent dans le dossier d'installation de QGIS, afin de lancer la commande tindex. On applique ensuite filter.hexbin en indiquant la valeur de threshold. Le threshold va influer sur la correspondance entre le nuage vrai et le polygone généré, dans le sens ou plus la valeur de seuil sera petite, plus le polygone va suivre le contour du nuage. La troisième ligne indique le fichier de sortie qui sera généré, au format SQLite. La commande filespec permet d'indiquer le chemin du nuage sur lequel nous voulons extraire la limite.

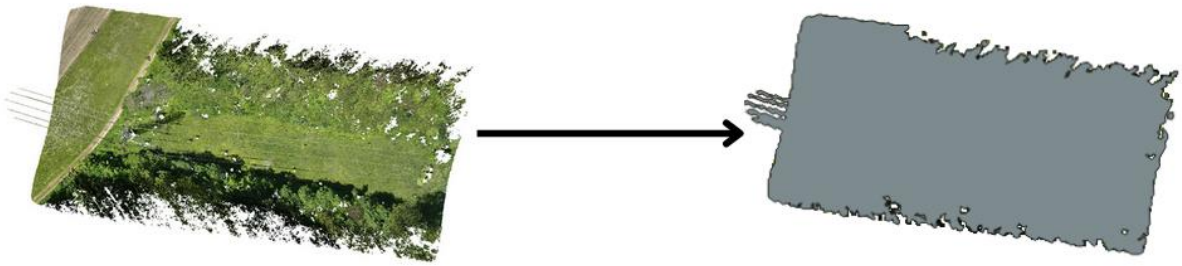


Figure 22: Résultat de la commande

La figure 22 ci-dessus présente le résultat de l'application de cette commande sur un nuage de point acquis au drone. Le résultat est très satisfaisant, tout d'abord du côté de la correspondance entre le nuage et son polygone limite mais également par la rapidité d'exécution de la commande, en effet, dans le cas de ce nuage contenant environ 14 Million de points, le résultat est obtenu en moins de 0,5 seconde. Plusieurs prototypes de codes ont été testés avant d'arriver à une solution convenable.

La première version contenait la commande dans une commande (Annexe 2a), dans le but de l'intégrer dans un fichier Batch pouvant être exécuté via l'invite de commande Windows, mais l'idée de générer un fichier batch pour générer les limites des nuages a été abandonnée.

La version retenue va être de générer la limite sans passer par la mise en place d'un fichier batch, et de la générer directement via Python (Annexe 2b). Les bibliothèques nécessaires sont os et subprocess, utilisées précédemment, mais également numpy [20], qui ici va permettre de charger le fichier texte contenant les métadonnées et les chemins vers les nuages LAS, avec l'utilisation de la commande genfromtxt [21]. Il faut ensuite indiquer l'emplacement du module PDAL, qui est le même utilisé pour la version précédente, et qui est contenu dans les dossiers de QGIS.

Une fois les éléments nécessaires mis en place, une boucle est créée afin de traiter chaque nuage de point LAS (Les chemins sont récupérés depuis le fichier texte). Un fichier SQLite sera donc généré pour chaque nuage. On construit ensuite la commande PDAL avec les différents éléments cités précédemment (voir figure 23). La boucle se termine par l'exécution de la commande PDAL. On retrouve donc ensuite les fichier SQLite des limites des nuages dans un dossier afin de vérifier les résultats. Une fois les polygones des limites des nuages générés, il faut en extraire les coordonnées des sommets en vue afin de pouvoir par la suite créer/compléter la base de données spatiale.

```
# Construire la commande PDAL
command = [
    pdal_executable,
    "tindex",
    "--filters.hexbin.threshold=2",
    "create",
    "--tindex",
    output_sqlite,
    "--filespec",
    las_file,
    "-f",
    "SQLite"
]
```

Figure 23: Construction de la commande PDAL dans le code

III.3.2 Extraction des coordonnées des limites.

La suite du processus concerne l'extraction des coordonnées des polygones, afin de pouvoir les inclure dans le fichier GeoPackage final. Pour se faire, un autre code Python a été créé afin d'extraire ces coordonnées (Annexe 3). La bibliothèque geopandas [22] a été utilisé pour réaliser ce code. C'est une bibliothèque open-source permettant de traiter des données spatiales via Python, et en effet, la commande read_file [23] va permettre de lire le fichier de limite SQLite généré avec le code précèdent.

Le code permet d'extraire les coordonnées des sommets des polygones générés après la commande PDAL. Il est important de noter que le premier sommet est répété à la fin de la liste, donc pour un polygones de 4 sommets, on aura en réalité 5 sommets dans la liste :

```
Les coordonnées du polygones sont:  [(-127.09145027, -74.20608526), (-127.09145027, -66.35648888),
(-124.74057986, -66.35648888), (-124.74057986, -74.20608526), (-127.09145027, -74.20608526)]
```

Figure 24: Extraction des coordonnées d'un polygone de 4 sommets

Polygone ABCD → [(xA, yA), (xB, yB) , (xC, yC) , (xD, yD) , (xA, yA)]

La figure 24 présente un exemple d'extraction des coordonnées d'un polygone généré à partir d'un nuage d'une pile du pont de Ballaigues. On observe bien que le premier sommet est répété à la fin de la liste. Les coordonnées seront ensuite stockées dans une liste afin de pouvoir les utiliser dans le code de génération de la base de donnée au format GeoPackage. Elles seront ainsi associées aux métadonnées des nuages stockées dans le fichier texte.

III.3.3 Création de la base de données

Une fois les polygones des limites générées et leurs coordonnées extraites, il est maintenant possible de créer/compléter la base de données au format SQLite. Pour créer/compléter la base de données, plusieurs codes ont été testés afin de parvenir à un résultat acceptable. Nous allons voir les différentes variantes de ce code. Ici, c'est la bibliothèque shapely.geometry [24] qui a été utilisée afin de pouvoir traiter les données polygonales issues des limites.

Dans la première version (Annexe 4a), les métadonnées étaient directement incluses dans le code Python en passant par la création d'un GeoDataframe [25] (issus de GeoPandas).

La figure 25 présente le stockage de ces métadonnées test directement dans le code Python, où l'on peut voir les différentes listes contenant les métadonnées. On peut voir sur cette figure 25 que l'on retrouve bien les métadonnées choisies auparavant, mais deux colonnes se sont ajoutées, celles des coordonnées X et Y des sommets des nuages, utilisées afin de générer des polygones (dans le cas de ces tests, ces coordonnées n'étaient pas issues de nuages). Les polygones et les métadonnées étaient ensuite stockées dans des listes afin de pouvoir ensuite lier les métadonnées avec les polygones correspondants et de générer un fichier GeoPackage. Le stockage des métadonnées et des coordonnées directement dans le code n'étant pas viable, une deuxième version (Annexe 4b) a été créée en stockant ces informations dans un fichier texte, contenant donc les mêmes colonnes que sur la figure 24.

```
df = gpd.GeoDataFrame(
    {
        "id_nuage": ["Nuage_1", "Nuage_2", "Nuage_3"],
        "date_acquisition": ["2023.03.01", "2023.03.02", "2023.03.03"],
        "type_donnees": ["Nuage", "Nuage", "Nuage"],
        "entreprise": ["Ent_1", "Ent_2", "Ent_2"],
        "capteur": ["Capteur_1", "Capteur_2", "Capteur_2"],
        "densite": ["100", "200", "300"],
        "georeferencement": ["approximatif", "precis", "precis"],
        "List_X": [
            [2521696.7, 2521699.0, 2521945.5, 2521948.5, 2521696.7],
            [2521796.7, 2521799.0, 2521745.5, 2521748.5, 2521796.7],
            [2531796.7, 2531799.0, 2531745.5, 2531748.5, 2531796.7],
        ],
        "List_Y": [
            [1175509.9, 1175491.8, 1175521.2, 1175536.9, 1175509.8],
            [1175900.9, 1175800.8, 1175700.2, 1175600.9, 1175500.8],
            [1185900.9, 1185800.8, 1185700.2, 1185600.9, 1185500.8],
        ],
    }
)
```

Figure 25: Métadonnées stockées dans un GeoDataframe

Il a fallu également, au début du code, préciser le type de colonnes présentes dans ce fichier texte, car certaines informations sont des textes et d'autres sont des valeurs numériques.

Cette version a également fonctionné et a notamment permis de pouvoir mettre à jour les métadonnées ou les coordonnées directement dans le fichier texte sans ouvrir le code Python, ce qui fut donc une avancée non négligeable.

L'inconvénient majeur de ces deux versions est le stockage des coordonnées des sommets des polygones. En effet, en cas de nombre conséquent de coordonnées, il n'est plus possible de tout stocker dans un fichier texte qui devient rapidement illisible.

C’est donc pour cette raison qu’il a été décidé de remplacer les deux colonnes contenant les coordonnées par une colonne unique contenant les chemins vers les nuages de points au format LAS, afin de récupérer les coordonnées directement dans le code. Comme on peut voir sur l’extrait de schéma fonctionnel (voir figure 26), le code de création vient donc récupérer les éléments depuis deux entrées différentes :

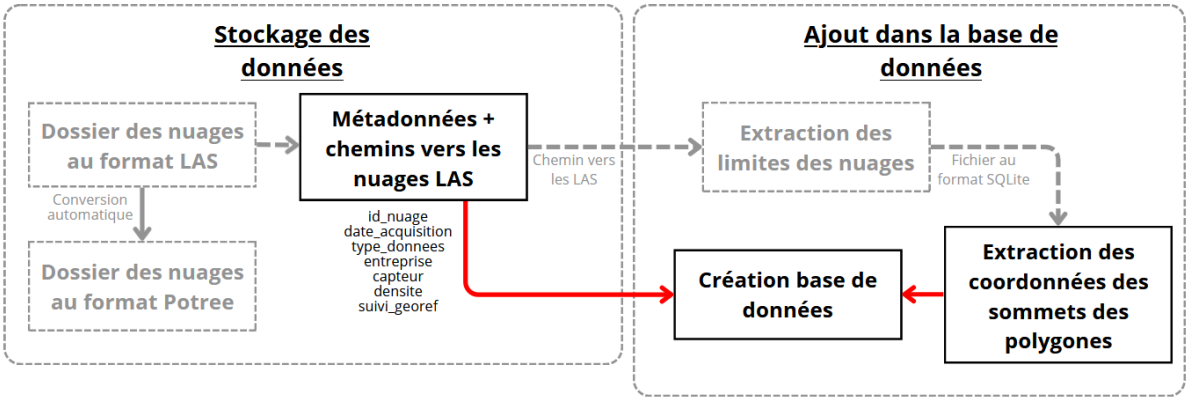


Figure 26: Entrées pour générer la base de données

La première entrée vient directement du fichier texte et contient les métadonnées, alors que la seconde entrée est issue du code d’extraction des coordonnées des sommets des limites, qui sont stockées dans une liste. En récupérant ces informations, il est donc possible, grâce à l’utilisation d’un GeoDataframe de générer un fichier GeoPackage pouvant être lu par SQLite Studio. Pour générer le fichier au format GeoPackage, il faut préciser dans la commande le paramètre « Driver » comme on peut le voir sur la figure 27.

```
# Écrire dans un fichier GeoPackage
polygon_df.to_file('test_bdd_polygon_excel.gpkg', driver='GPKG', layer='name')
```

Figure 27: Création du fichier GPKG

D’autres options d’écriture sont disponibles, comme le shapefile (.shp) et le GeoJSON (.GeoJSON). Une fois la base de données complétée des métadonnées, des données spatiales, et générée au bon format, elle devient utilisable afin de simuler des requêtes de choix des nuages.

La partie suivante va donc traiter la question de l’affichage des données souhaités.

III.4. Diffusion des données

III.4.1 Sélection des nuages par l'utilisateur

Pour l'instant, les nuages ne sont pas sélectionnés par requêtes, mais à terme, après la mise en place d'un serveur web, il sera possible d'utiliser des plateformes de gestions de bases de données comme QGIS server qui permet de « *mettre en œuvre des fonctionnalités cartographiques avancées pour la cartographie thématique. […] QGIS Desktop et QGIS Server utilisent les mêmes bibliothèques de visualisation* » [26]. Ces plateformes vont permettre à l'utilisateur de sélectionner les nuages selon des requêtes (date, densité…) à l'image de la figure 28 et ainsi, en récupérant les nuages sélectionnés, il sera possible de générer automatiquement un fichier HTML Potree pour visualiser les nuages sélectionnés.

L'utilisation d'une plateforme comme QGIS server reste une hypothèse d'évolution en vue de la diffusion au grand public des données de SwissRenov. De plus, la génération automatique du fichier HTML Potree est un aspect crucial dans l'automatisation du processus et permet de simplifier drastiquement la visualisation des nuages de points pour des utilisateurs peu familiers avec ce type de données. La mise en place de ce code Python a été traitée au cour de cette étude et va être présentée dans la partie suivante.

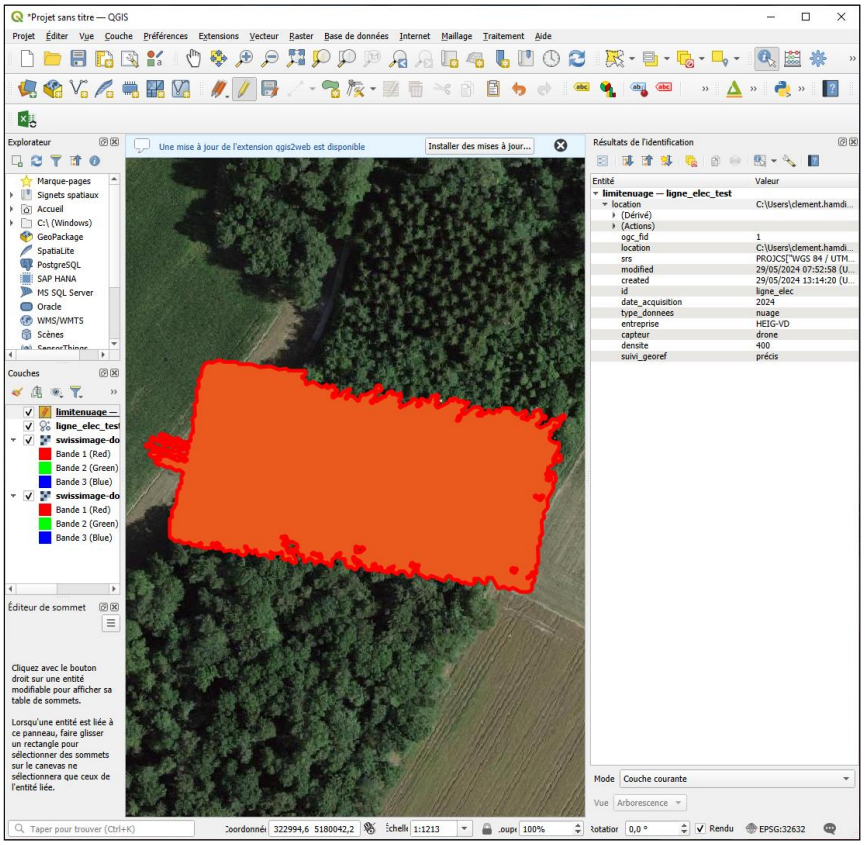


Figure 28: Sélection d'un nuage sur QGIS

III.4.2 Ecriture automatique du fichier HTML Potree

Après avoir généré la base de données, l'utilisateur peut effectuer des requêtes afin de sélectionner des nuages selon des critères précis (date, capteur…). L'objectif de cette partie est de pouvoir générer le fichier HTML de Potree en fonction des nuages sélectionnés. Le fichier HTML étant généré automatiquement lors de la conversion des nuages LAS, l'objectif va être de le générer automatiquement lors de chaque requête de l'utilisateur. Un code Python a donc été créé afin de générer automatiquement le bon fichier HTML en fonction des nuages sélectionnés. Ce code (Annexe 5) utilise la bibliothèque webbrowser [27] afin d'ouvrir directement une page web lorsque les nuages sont sélectionnés. Une liste contient tout d'abord les identifiant des nuages au format Potree, présents dans le dossier « Pointcloud ». Chaque nuage présent dans la liste est lié à un indice, qui correspond à sa place dans la liste, qu'il faudra spécifier par la suite. La suite du code permet d'écrire le fichier HTML correspondant, en utilisant les trois blocs du fichier vu précédemment dans la figure 12.

Dans un premier temps, on écrit le bloc des librairies, qui est commun à tous les fichier HTML, et qui ne dépend pas des nuages choisis. Dans un second temps, on écrit le bloc de chargement des nuages de points. Une boucle va donc dupliquer ce bloc pour chaque nuage choisi par l'utilisateur, en faisant correspondre les chemins vers les nuages.

Le code termine en écrivant les dernières balises nécessaires au bon fonctionnement du fichier HTML. En exécutant le code, l'utilisateur choisit donc les nuages qu'il souhaite en indiquant leurs indices (voir figure 29), puis le code génère automatiquement le fichier HTML et ouvre la page web correspondante (voir figure 30).

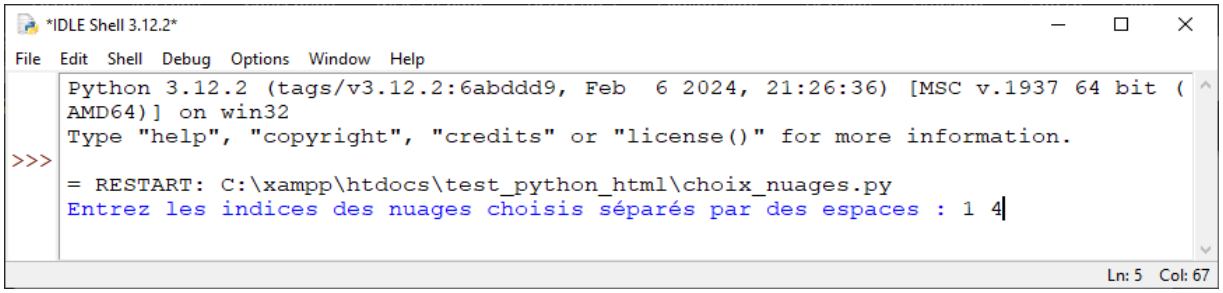


Figure 29: Choix des nuages à afficher

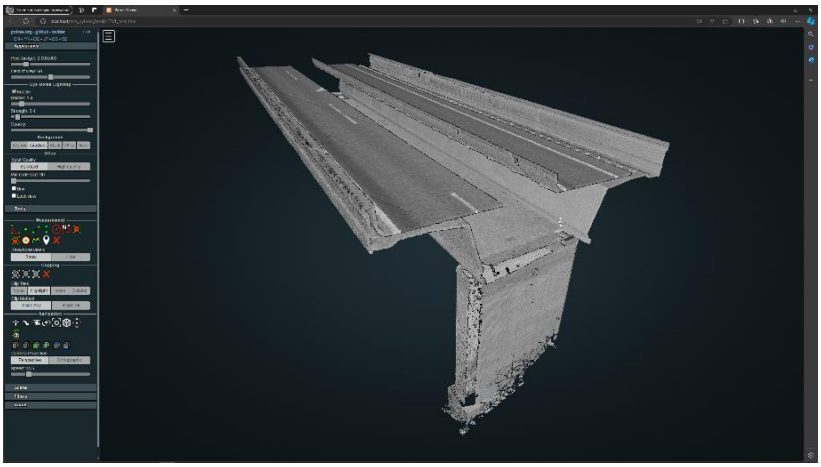


Figure 30: Affichage des nuages sélectionnés

De cette façon, l'utilisateur ne travaille qu'avec les nuages qu'il souhaite et non pas avec l'ensemble du nuage global de la zone, ce qui impliquerait de charger dans la page Potree un grand nombre de données pouvant nuire aux performances, ou rendant plus difficile la compréhension des nuages de points par un utilisateur peu habitué.

Comme nous pouvons voir dans le schéma fonctionnel (voir figure 14), la partie diffusion des données comporte une partie relative, un serveur web. La mise en place de ce serveur n'a pas été traitée pendant cette étude, qui était plus particulièrement liée à la base de données en elle-même. La diffusion des données tests par ce code Python a donc été réalisée par la mise en place d'un serveur XAMMP en local comme expliqué plus tôt dans ce mémoire.

Malgré le fait que certains aspects n'ai pu être traités, un prototype de code Python regroupant l'ensemble des aspect présentés dans cette dernière partie a été entrepris, et a permis d'avoir une idée plus claire de la chaine de traitement des données réalisée au cour de cette étude. Les codes précédent on également été améliorés dans le cadre de ce prototype. Ce prototype sera présenté dans la partie suivante.

III.5 Prototypage

Une fois les différents codes mis en place séparément, la création d'un prototype de chaine de traitement a été initié. Cela correspond donc à un nouveau code regroupant une partie des étapes vues précédemment, certaines restant réalisées à part, comme la conversion des nuages du format LAS vers le format Potree. Le prototype extrait tout d'abord les limites des nuages en utilisant la dernière colonne du fichier texte, ce qui correspond donc aux chemins vers les nuages LAS.

Cette étape est réalisée de la même manière que le code présenté précédemment, en construisant la commande PDAL directement dans le code Python et en l'appliquant pour chaque nuage présent dans le dossier de stockage. Ces limites sont stockées dans un dossier dédier regroupant l'ensemble des limites des nuages, séparément, au format SQLite.

Une fois les limites générées, une fonction va récupérer ces limites afin d'en extraire les coordonnées des sommets. Une boucle parcourt les fichiers SQLite présents dans le dossier, puis, chaque fichier est lu afin d'en extraire les coordonnées. Les nuages n'étant pas géoréférencés, les coordonnées sont donc locales. Après avoir extrait les coordonnées, un GeoDataframe est créé afin de fusionner les métadonnées avec les coordonnées des limites. Le GeoDataframe écrit enfin les données dans un fichier GeoPackage, contenant donc pour chaque nuage, la géométrie correspondante à sa limite ainsi que ses métadonnées. La figure 31 présente les colonnes du fichier GeoPackage. La colonne « geom » contient les géométries des limites, dont les données sont des polygones.

	Nom	Type de données	Clé primaire	Clé étrangère	Unique	Contrôle	Non NULL	Collecter	Généré	
1	fid	INTEGER								NULL
2	geom	POLYGON								NULL
3	id_nuage	TEXT								NULL
4	date	TEXT								NULL
5	type	TEXT								NULL
6	entreprise	TEXT								NULL
7	capteur	TEXT								NULL
8	densite	TEXT								NULL
9	georeferencement	TEXT								NULL

Figure 31: Structure du GeoPackage

IV. Possibilités d'évolution

A ce stade de la recherche, les différents blocs du schéma fonctionnel fonctionnent, et permettent de mieux comprendre l'objectif final de cette étude dans le contexte du projet SwissRenov. Certaines liaisons entre les différents éléments n'ont pas pu être développés mais des hypothèses ont été formulées.

La figure 32 présente les différentes étapes réalisées (en vert) et les étapes prévues dans la suite du développement (en rouge).

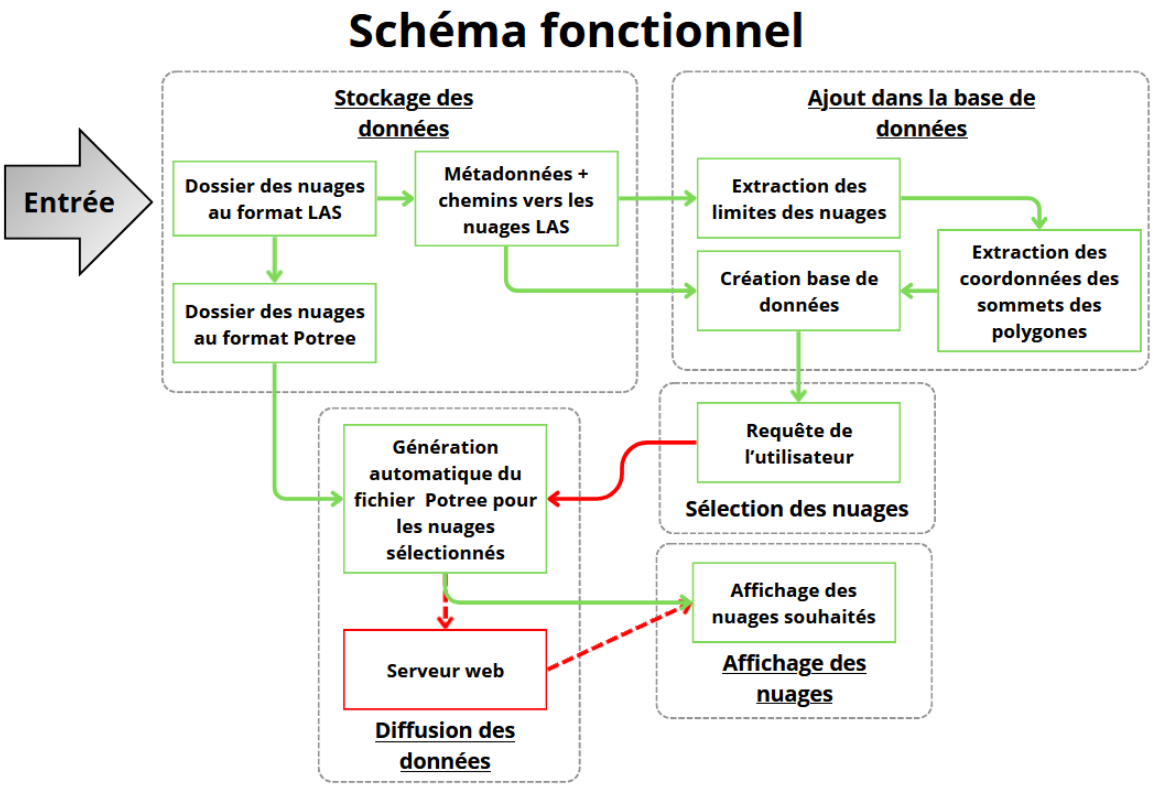


Figure 32: Résultats du développement

IV.1 Extraction des entités sélectionnées

Pour l'instant, le choix des nuages à afficher dans Potree est réalisé manuellement en rentrant l'indice des nuages dans l'interface Python et n'est pas issus des requêtes réalisées dans QGIS ou dans un autre logiciel de gestion de données spatiales.

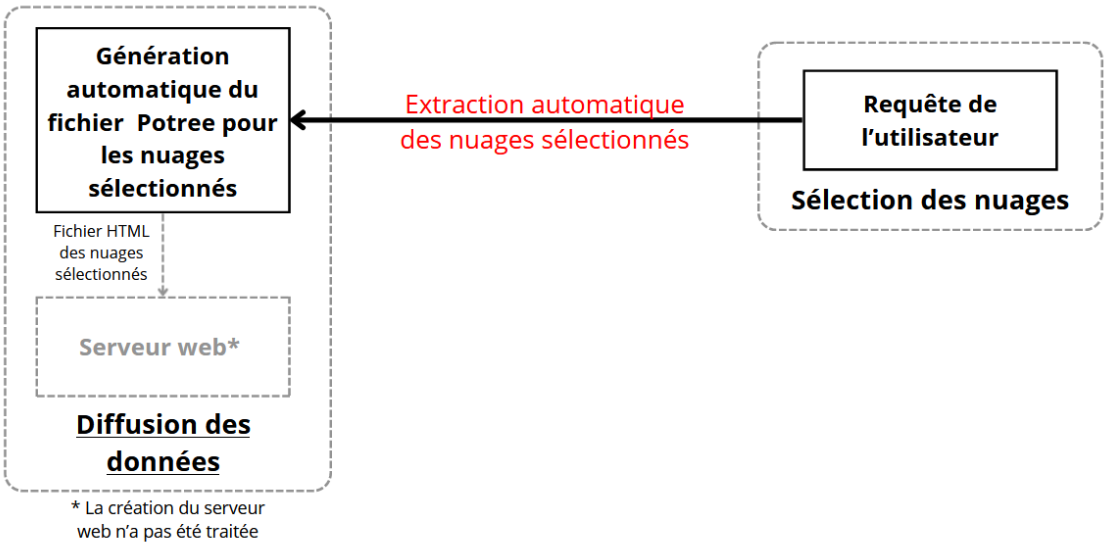


Figure 33: Extraction des entités sélectionnés

Il serait donc intéressant d'ajouter une liaison (plugin, code Python) automatique entre les blocs « Sélection des nuages » et « Diffusion des données » visibles à la figure 33. L'objectif de cette liaison serait donc de récupérer automatiquement les identifiant des nuages sélectionnés par requête, afin de pouvoir directement générer les fichier HTML. De cette façon l'utilisateur n'aurait plus qu'à effectuer ses requêtes afin d'avoir accès à la page Potree. QGIS possédant une console Python intégrée, il serait possible de stocker les indices des nuages afin de les réinjecter dans le code déjà créé. D'autre part, cela permettrait de comprendre le fonctionnement dans le cadre d'une utilisation de QGIS Server.

IV.2 Diffusion sur une plateforme en ligne

L'objectif principal étant de permettre à une grande variété d'utilisateurs de pouvoir accéder aux données, simplement, rapidement et depuis n'importe quelle plateforme, il serait très intéressant de créer une plateforme en ligne.

Cet aspect n'a pas été traité pendant cette étude mais pourrais par la suite être développé. En effet, le développement d'une telle plateforme nécessite la création interface utilisateur pour faciliter l'accès et l'interaction avec les données.

Cet aspect de diffusion des données rejoint la création d'un serveur web et nécessite donc un travail conséquent afin de réaliser une plateforme facile et agréable à utiliser pour des utilisateurs peu familiers avec des nuages de points 3D et les bases de données. Un autre aspect important à prendre en compte lors de la création d'une plateforme en ligne est la sécurité. La diffusion de ces données 3D doit être particulièrement sécurisé, c'est notamment une des raisons pour lesquelles la table contenant les métadonnées sur les nuages de points comporte une colonne « entreprise » comportant seulement le nom de l'entreprise, nécessaire pour remplir l'objectif de traçabilité. Il a été évoqué pendant la discussion sur ces métadonnées, la possibilité d'inclure le nom de l'opérateur ayant réalisé l'acquisition, mais cette proposition n'a pas été gardée pour des raisons de confidentialité. Il sera nécessaire de mettre en place un système d'authentification afin de pouvoir accéder aux données.

Ces différents aspects seront donc à approfondir dans le futur afin de pouvoir diffuser pleinement ces données au grand public.

IV.3 Ajout d’autres formats de données

Lors de cette étude, les données utilisées étaient des nuages de points 3D, car ils vont représenter la grande majorité des données acquises lors des campagnes d’acquisition sur les friches. Mais d’autres types de données pourront être ajoutés, comme des shapefiles, ou des orthophotographies, qui peuvent également être incluses dans l’affichage sur Potree. Il serait également intéressant de relier les informations contenues à la base de données à des maquettes BIM, qui sont traitées dans le WP3.2 du sous-projet 3.

Conclusion

Le projet SwissRenov est encore dans ses phases initiales de développement, et de nombreux aspects restent à explorer et à traiter avant d'aboutir à une solution viable pour la réutilisation des friches industrielles. Ce projet d'envergure nationale rassemble divers acteurs clés à travers la Suisse, dont la coopération est cruciale pour sa réussite. En effet, cette étude n'en représente qu'une petite fraction, mais elle est néanmoins essentielle pour la bonne compréhension des défis et opportunités liés à ce vaste projet.

La gestion des données, constitue un élément central de ce projet en raison de la grande diversité des métiers impliqués et des étendues géographiques considérables qu'il englobe. L'établissement d'une base de données pour les données 3D recueillies permettra à l'ensemble des participants de travailler efficacement sur les nuages de points, facilitant ainsi la collaboration et l'analyse des données, grâce à un choix des métadonnées adaptées.

Dans un premier temps, la recherche d'une plateforme de diffusion appropriée a impliqué une comparaison de nombreux logiciels et plateformes en ligne. Ce processus de sélection a abouti au choix de Potree, qui s'est avéré être la plateforme la plus adéquate. Cette solution permet aux utilisateurs d'interagir avec les nuages de points sans nécessiter de formation préalable sur l'utilisation des données 3D, ce qui est un atout considérable pour l'intégration rapide et efficace de différents utilisateurs. Le choix de Potree est également conforté par son utilisation dans divers projets menés par des institutions réputées comme l'IGN, garantissant ainsi une fiabilité et une reconnaissance de son efficacité.

Par la suite, la mise en place d'une chaîne de traitement pour les nuages de points a permis de définir précisément les données d'entrée ainsi que les résultats attendus en sortie. Les premiers résultats sont encourageants et satisfaisants, témoignant de l'efficacité de cette chaîne de traitement.

Les différents modules de traitement fonctionnent correctement, avec un ensemble de scripts Python dédiés qui assurent, d'une part, la conversion des données vers le format compatible avec Potree et, d'autre part, la création d'une base de données au format GeoPackage. Cette base de données inclut à la fois les métadonnées et les données spatiales, permettant ainsi une gestion exhaustive des informations collectées. L'automatisation des différentes étapes, comme la génération des limites, l'extraction de leurs coordonnées, ainsi que la création de la base de données, est un facteur clé de la rapidité des traitements des données 3D. Cela représente un avantage considérable dans le cadre du projet SwissRenov. Toutefois, certaines parties du processus nécessitent encore des améliorations pour parvenir à une automatisation complète de la chaîne de traitement. Le prototype réalisé à ce stade intègre les principales étapes de ce processus et offre une vision claire de ce à quoi pourrait ressembler le produit final.

Enfin, la formulation d'hypothèses d'évolution permet de déterminer les orientations futures du projet, notamment en envisageant une transition de SQLite vers PostgreSQL pour la gestion de la base de données. L'établissement d'un serveur web est également prévu, facilitant ainsi la diffusion des données auprès des différents utilisateurs et partenaires. De plus, l'intégration de nouveaux formats de données sera essentielle pour incorporer un maximum d'informations sur les friches industrielles, enrichissant ainsi la base de données et les possibilités d'analyse. Le développement d'une interface web, potentiellement similaire à GeoPortail, est également envisagé pour permettre à tous les acteurs d'accéder aisément aux données.

Les résultats finaux de cette étude sont globalement satisfaisants par rapport aux attentes initiales. En effet, les principaux axes de la problématique, à savoir la gestion, l'interaction et la mise à disposition des données 3D, ont été traités avec succès. Cela constitue une avancée significative dans le développement de la plateforme demandée dans le cadre du projet SwissRenov, et ouvre la voie à de nouvelles perspectives pour la suite du projet.

Bibliographie

[1] SWISSRENOV, Le projet, [En ligne]. Disponible sur : https://swissrenov.ch/, consulté le 11 juin 2024.

[2] SWISSTOPO, SwissSURFACE3D, [En ligne]. Disponible sur : https://www.swisstopo.admin.ch/fr/modele-altimetrique-swisssurface3d, consulté le 18 mars 2024.

[3] ARCGIS DESKTOP, ArcMap, [En ligne]. Disponible sur : https://desktop.arcgis.com/fr/arcmap/latest/manage-data/las-dataset/benefits-of-using-las-datasets.html, consulté le 25 juin 2024.

[4] DANIELGM, Documentation CloudCompare version 2.4, [En ligne]. Disponible sur : https://www.danielgm.net/cc/doc/qCC/Documentation_CloudCompare_version_2_4.pdf, consulté le 30 mai 2024.

[5] QGIS DOCUMENTATION, Working with Point Clouds, [En ligne]. Disponible sur : https://docs.qgis.org/3.34/en/docs/user_manual/working_with_point_clouds/point_clouds.html, consulté le 30 mai 2024.

[6] GITHUB, Qgis2threejs, [En ligne]. Disponible sur : https://github.com/minorua/Qgis2threejs, consulté le 25 juin 2024.

[7] QGIS, Changelog for QGIS 3.32, [En ligne]. Disponible sur : https://www.qgis.org/en/site/forusers/visualchangelog332/index.html, Consulté le 31 mai 2024.

[8] Simon COLLE, 2022, « Étude des solutions permettant de capitaliser, partager et publier des données en trois dimensions de natures différentes à partir de projets d'affleurements rocheux et d'infrastructures », [En ligne]. Mémoire ESGT, p40, Consulté le 31 mai 2024.

[9] WILD-LIGHTS, Potree, [En ligne]. Disponible sur : http://www.wildlights.com/archeodev/potree.pdf, consulté le 31 mai 2024.

[10] ATIS.CLOUD, Fonctionnalités, [En ligne]. Disponible sur : https://www.atis.cloud/fr/fonctionnalites, consulté le 31 mai 2024.

[11] COMMISSARIAT A LA PROTECTION DE LA VIE PRIVEE DU CANADA, Qu'est-ce qu'une « Métadonnée » ?, [En ligne]. Disponible sur : https://www.priv.gc.ca/media/2347/md_info_201410_f.pdf, consulté le 15 mai 2024.

[12] CEREMA, Cartofriche, , [En ligne]. Disponible sur : https://cartofriches.cerema.fr/cartofriches/, consulté le 03 juin 2024.

[13] DOC PYTHON, os – Miscellaneous operating system interfaces, [En ligne]. Disponible sur : https://docs.Python.org/fr/3/library/os.html, consulté le 12 juin 2024.

[14] DOC PYTHON, subprocess – Subprocess management, [En ligne]. Disponible sur : https://docs.Python.org/3/library/subprocess.html, consulté le 12 juin 2024.

[15] GEOPACKAGE, Getting started with GeoPackage, [En ligne]. Disponible sur : http://www.geopackage.org/guidance/getting-started.html, consulté le 24 avril 2024.

[16] PDAL, Finding the boundary, [En ligne]. Disponible sur : https://pdal.io/en/2.6.0/workshop/manipulation/boundary/boundary.html, consulté le 10 juin 2024.

[17] GITHUB, Hobuinc/hexer, [En ligne]. Disponible sur : https://github.com/hobuinc/hexer, consulté le 10 juin 2024.

[18] PDAL, filter.hexbin, [En ligne]. Disponible sur : https://pdal.io/en/2.6.0/stages/filters.hexbin.html, consulté le 20 juin 2024.

[19] GDAL, gdaltindex, [En ligne]. Disponible sur : https://gdal.org/programs/gdaltindex.html, consulté le 10 juin 2024.

[20] NUMPY, Numpy, [En ligne]. Disponible sur : https://numpy.org/, consulté le 12 juin 2024.

[21] NUMPY, Numpy.genfromtxt, [En ligne]. Disponible sur : https://numpy.org/doc/stable/reference/generated/numpy.genfromtxt.html, consulté le 20 juin 2024.

[22] GEOPANDAS, geopandas 0.14.4, [En ligne]. Disponible sur : https://geopandas.org/en/stable/, consulté le 12 juin 2024.

[23] GEOPANDAS, geopandas.read_files, [En ligne]. Disponible sur : https://geopandas.org/en/stable/docs/reference/api/geopandas.read_file.html, consulté le 20 juin 2024.

[24] SHAPELY, geometry, [En ligne]. Disponible sur : https://shapely.readthedocs.io/en/2.0.4/geometry.html, consulté le 13 juin 2024.

[25] GEOPANDAS, geopandas.geodataframe, [En ligne]. Disponible sur : https://geopandas.org/en/stable/docs/reference/api/geopandas.GeoDataFrame.html, consulté le 13 juin 2024.

[26] QGIS DOCUMENTATION, Introduction, [En ligne]. Disponible sur : https://docs.qgis.org/3.34/fr/docs/server_manual/introduction.html, consulté le 05 mai 2024.

[27] DOC PYTHON, webbrowser -Convenient web browser controller, [En ligne]. Disponible sur : https://docs.Python.org/3/library/webbrowser.html, consulté le 19 juin 2024.

Liste des figures

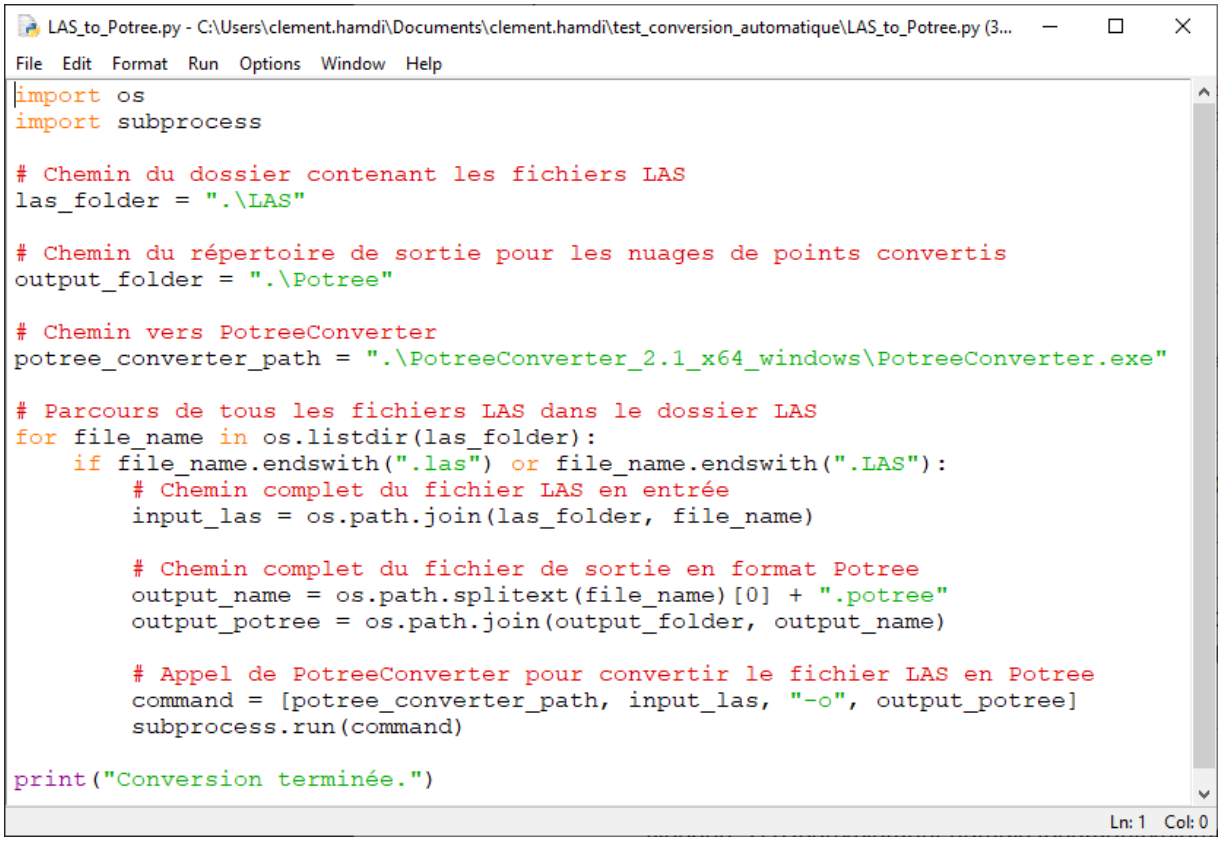
Liste des tableaux

Table des annexes

Annexe 1 : Conversion du format LAS vers le format Potree

Ce code permet de convertir automatiquement les nuages (LAS) vers le format Potree. Celui-ci utilise la version 2.1 de PotreeConverter. Ici, les éléments importants à modifier en cas d'utilisation sont:

- Le chemin vers les nuages LAS (las_folder).
- Le chemin vers le répertoire de sortie (output_folder) des nuages convertis.
- Le chemin vers PotreeConverter (potree_converter_path).



Annexe 2a et 2b : Extraction des limites des nuages de points

Ce code permet d'extraire les limites des nuages de points automatiquement en utilisant la commande PDAL « Boundary ». Il existe deux versions de ce code, la première générant un fichier batch contenant la commande PDAL, et la deuxième version exécutant la commande directement dans code sans passer par un fichier batch.

Annexe 2a (Version 1)

```
import subprocess
import os
import os.path

# Définir la commande à écrire dans le fichier batch
command = (
    '"C:/Program Files/QGIS 3.36.1/bin/pdal.exe" tindex '
    '--filters.hexbin.threshold=2 create '
    '--tindex ./limitenuage.sqlite '
    '--filespec "C:/Users/clement.hamdi/Documents/clement.hamdi/test_sqlite_geopackage/'
    'test_limite_qgis/las_tests_elec/ligne_elec_test.las" -f SQLite'

)
# Nom du fichier batch
batch_file = 'MonBat2.bat'

# Ouverture d'un fichier batch en mode écriture
with open(batch_file, 'w') as file:
    # Écriture de la commande dans le fichier batch
    file.write(command + '\n')  # Écriture de la commande spécifiée

print(f"Le fichier batch '{batch_file}' a été généré avec succès.")

# Demander à l'utilisateur s'il souhaite exécuter le fichier batch
user_input = input("Voulez-vous exécuter le fichier batch maintenant ? (oui/non) : ")

if user_input.lower() in ['o', 'oui', 'y', 'yes']:
    # Exécution du fichier batch
    try:
        subprocess.run(batch_file, check=True, shell=True)
        print(f"Le fichier batch '{batch_file}' a été exécuté avec succès.")
    except subprocess.CalledProcessError as e:
        print(f"Erreur lors de l'exécution du fichier batch '{batch_file}': {e}")
else:
    print("Le fichier batch n'a pas été exécuté.")
```

Annexe 2b (Version 2)

```
import subprocess
import numpy as np
import os
from shapely.geometry import Polygon
import geopandas as gpd

############################################################
################## Extraction des limites ##################
############################################################

# Chemin du fichier texte contenant les chemins des fichiers LAS
data = np.genfromtxt("C:/Users/clement.hamdi/Documents/clement.hamdi/Prototype_final/test_bdd.txt", dtype=str)

# Lire les chemins des fichiers LAS
las_files = data[:, 7]

# Emplacement de l'exécutable PDAL
pdal_executable = "C:/Program Files/QGIS 3.36.1/bin/pdal.exe"

# Boucle pour traiter chaque fichier LAS
for las_file in las_files:
    # Générer un nom de fichier SQLite unique basé sur le nom du fichier LAS
    base_name = os.path.basename(las_file)
    name_without_ext = os.path.splitext(base_name)[0]
    output_sqlite = f"C:/Users/clement.hamdi/Documents/clement.hamdi/Prototype_final/Codes/limites_nuages/limitenuage_{name_without_ext}.sqlite"

    # Construire la commande PDAL
    command = [
        pdal_executable,
        "tindex",
        "--filters.hexbin.threshold=2",
        "create",
        "--tindex",
        output_sqlite,
        "--filespec",
        las_file,
        "-f",
        "SQLite"
    ]

    # Exécuter la commande PDAL
    try:
        result = subprocess.run(command, check=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE)
        print(f"Le fichier {las_file} a été traité avec succès et enregistré sous {output_sqlite}.")
        print(result.stdout.decode())
    except subprocess.CalledProcessError as e:
        print(f"Erreur lors du traitement du fichier {las_file}: {e.stderr.decode()}")
```

Dans ce code, les éléments importants à modifier en cas d'utilisation sont:

- Le chemin vers le fichier texte (data).
- Le chemin vers l'exécutable PDAL (pdal_executable).
- La commande threshold dans la commande PDAL.

Annexe 3 : Extraction des coordonnées des limites

Ce code permet d'extraire les coordonnées des sommets des limites générées précédemment (fichier SQLite). Ici, les éléments importants à modifier en cas d'utilisation sont:

- Le chemin vers le fichier SQLite (limite).

```
import geopandas as gpd

# Lecture du fichier SQLite
limite = gpd.read_file("limitenuage.sqlite")

# Fonction d'extraction des coordonnées d'une géométrie
def extract_polygon_coords(geom):
    if geom is None:
        return []
    if geom.geom_type == 'Polygon':
        return list(geom.exterior.coords)
    elif geom.geom_type == 'MultiPolygon':
        coords = []
        for poly in geom.geoms:
            coords.extend(list(poly.exterior.coords))
        return coords
    else:
        return []

# Application de la fonction
coords = limite.geometry.apply(extract_polygon_coords)

# Affichage des coordonnées
for polygon_coords in coords:
    print("Les coordonnées des sommets du polygone sont: ", polygon_coords)
    print("Le polygone contient", len(polygon_coords), "sommets")
```

Annexe 4a et 4b : Génération de la base de données

Ce code permet de générer le fichier GeoPackage de la base de données. Deux versions ont été créées, la première contient les données directement dans le code Python, alors que la deuxième version va les récupérer dans un fichier texte.

Annexe 4a (Version 1)

```python
import geopandas as gpd
from shapely.geometry import Polygon

# Métadonnées
df = gpd.GeoDataFrame(
    {
        "id_nuage": ["Nuage_1", "Nuage_2", "Nuage_3"],
        "date_acquisition": ["2023.03.01", "2023.03.02", "2023.03.03"],
        "type_donnees": ["Nuage", "Nuage", "Nuage"],
        "entreprise": ["Ent_1", "Ent_2", "Ent_2"],
        "capteur": ["Capteur_1", "Capteur_2", "Capteur_2"],
        "densite": ["100", "200", "300"],
        "georeferencement": ["approximatif", "precis", "precis"],
        "List_X": [
            [2521696.7, 2521699.0, 2521945.5, 2521948.5, 2521696.7],
            [2521796.7, 2521799.0, 2521745.5, 2521748.5, 2521796.7],
            [2531796.7, 2531799.0, 2531745.5, 2531748.5, 2531796.7],
        ],
        "List_Y": [
            [1175509.9, 1175491.8, 1175521.2, 1175536.9, 1175509.8],
            [1175900.9, 1175800.8, 1175700.2, 1175600.9, 1175500.8],
            [1185900.9, 1185800.8, 1185700.2, 1185600.9, 1185500.8],
        ],
    }
)

# Création des listes stockant les polygones et les métadonnées des nuages
polygons = []
id_nuage = []
date_acquisition = []
type_donnees = []
entreprise = []
capteur = []
densite = []
georeferencement = []

# Création des polygones et des insertion des métadonnées dans les colonnes
for index, row in df.iterrows():
    polygon_geom = Polygon(zip(row['List_X'], row['List_Y']))
    polygons.append(polygon_geom)
    id_nuage.append(row['id_nuage'])
    date_acquisition.append(row['date_acquisition'])
    type_donnees.append(row['type_donnees'])
    entreprise.append(row['entreprise'])
    capteur.append(row['capteur'])
    densite.append(row['densite'])
    georeferencement.append(row['georeferencement'])

# Création du GéoDataFrame
polygon = gpd.GeoDataFrame({
    'id_nuage': id_nuage,
    'date_acquisition': date_acquisition,
    'type_donnees': type_donnees,
    'entreprise': entreprise,
    'capteur': capteur,
    'densite': densite,
    'georeferencement': georeferencement,
    'geometry': polygons
}, crs='epsg:2056')

polygon.to_file('test_bdd_polygon.gpkg', driver='GPKG', layer='name')

print("Base de données générée avec succès")
```

Annexe 4b (Version 2)

```python
import geopandas as gpd
from shapely.geometry import Polygon
import numpy as np

# Définir les types des colonnes
dtype = [('id_nuage', 'U10'), ('date', 'U10'), ('type_nuage', 'U10'),
         ('ent', 'U10'), ('capteur', 'U10'), ('valeur', 'i4'),
         ('qualite', 'U10'), ('coordonnees_X', 'U50'), ('coordonnees_Y', 'U50')]

# Lecture des données
data = np.genfromtxt("test_bdd.txt", dtype=dtype, encoding='utf-8', delimiter='\t')

# Affichage des données
print("Données brutes :\n", data)

# Conversion des colonnes des coordonnées en listes de flottants
coords_X = [list(map(float, coord.split(', '))) for coord in data['coordonnees_X']]
coords_Y = [list(map(float, coord.split(', '))) for coord in data['coordonnees_Y']]

# Création des polygones
polygons = [Polygon(zip(x, y)) for x, y in zip(coords_X, coords_Y)]

# Création du GeoDataFrame
polygon_df = gpd.GeoDataFrame({
    'id_nuage': data['id_nuage'],
    'date': data['date'],
    'type_nuage': data['type_nuage'],
    'ent': data['ent'],
    'capteur': data['capteur'],
    'valeur': data['valeur'],
    'qualite': data['qualite'],
    'geometry': polygons
}, crs='epsg:2056')

# Affichage des premières lignes du GeoDataFrame
print(polygon_df.head())

# Écriture dans un fichier GeoPackage
polygon_df.to_file('test_bdd_polygon_excel.gpkg', driver='GPKG', layer='name')

print("Base de données générée avec succès")
```

Ici, les éléments importants à modifier en cas d'utilisation sont:

- Les types de données (dtype).
- Le chemin vers le fichier texte (data).
- Le format de sortie des données (driver).

Annexe 5 : Génération automatique du fichier HTML

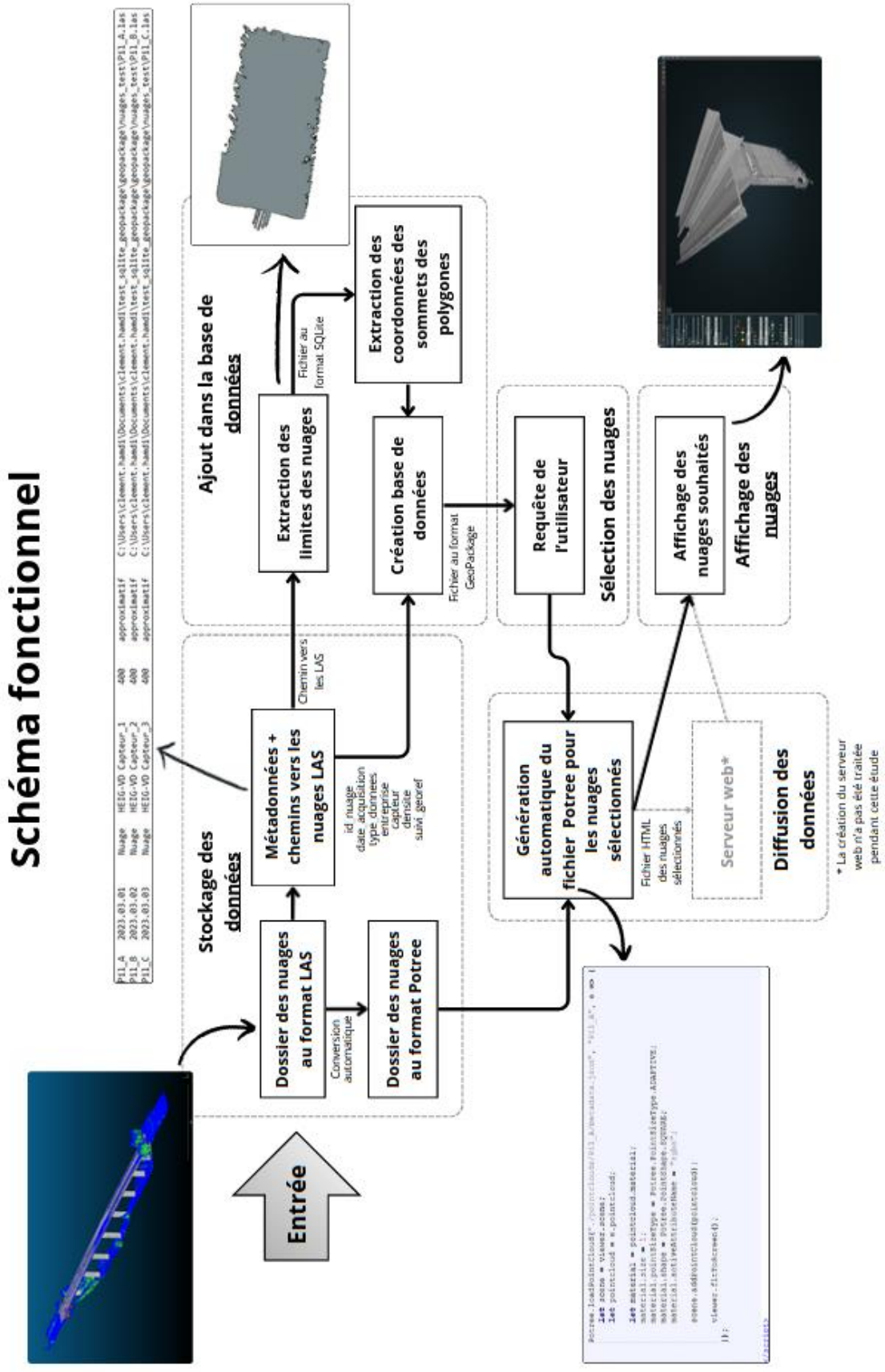
Ce code génère automatiquement le fichier HTML Potree en fonction des nuages sélectionnés. Ici, les éléments importants à modifier en cas d'utilisation sont:

- La liste des nuages (liste_nuages).
- Le chemins vers les nuages au format Potree (Potree.loadPointCloud)

```
import webbrowser

# Création de la liste des nuages
liste_nuages = ["Pil_A", "Pil_B", "Pil_C", "Tab_AB", "Tab_BC"]
indices = [int(i) for i in input("Entrez les indices des nuages choisis séparés par des espaces : ").split()]
choix_nuages = [liste_nuages[i-1] for i in indices]

# Ecriture du premier bloc du html (librairies + éléments de la page)
with open("HTML_test.html", "w") as fichier:
    fichier.write("""<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="description" content="">
    <meta name="author" content="">
    <meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no">
    <title>Potree Viewer</title>

    <link rel="stylesheet" type="text/css" href="./libs/potree/potree.css">
    <link rel="stylesheet" type="text/css" href="./libs/jquery-ui/jquery-ui.min.css">
    <link rel="stylesheet" type="text/css" href="./libs/openlayers3/ol.css">
    <link rel="stylesheet" type="text/css" href="./libs/spectrum/spectrum.css">
    <link rel="stylesheet" type="text/css" href="./libs/jstree/themes/mixed/style.css">
</head>

<body>
    <script src="./libs/jquery/jquery-3.1.1.min.js"></script>
    <script src="./libs/spectrum/spectrum.js"></script>
    <script src="./libs/jquery-ui/jquery-ui.min.js"></script>
    <script src="./libs/other/BinaryHeap.js"></script>
    <script src="./libs/tween/tween.min.js"></script>
    <script src="./libs/d3/d3.js"></script>
    <script src="./libs/proj4/proj4.js"></script>
    <script src="./libs/openlayers3/ol.js"></script>
    <script src="./libs/i18next/i18next.js"></script>
    <script src="./libs/jstree/jstree.js"></script>
    <script src="./libs/potree/potree.js"></script>
    <script src="./libs/plasio/js/laslaz.js"></script>

    <!-- INCLUDE ADDITIONAL DEPENDENCIES HERE -->
    <!-- INCLUDE SETTINGS HERE -->

    <div class="potree_container" style="position: absolute; width: 100%; height: 100%; left: 0px; top: 0px; ">
        <div id="potree_render_area" style="background-image: url('../build/potree/resources/images/background.jpg');"></div>
        <div id="potree_sidebar_container"> </div>
    </div>

    <script>

        window.viewer = new Potree.Viewer(document.getElementById("potree_render_area"));

        viewer.setEDLEnabled(true);
        viewer.setFOV(60);
        viewer.setPointBudget(2_000_000);
        <!-- INCLUDE SETTINGS HERE -->
        viewer.loadSettingsFromURL();

        viewer.setDescription("");

        viewer.loadGUI(() => {
            viewer.setLanguage('en');
            $("#menu_appearance").next().show();
            $("#menu_tools").next().show();
            $("#menu_clipping").next().show();
            viewer.toggleSidebar();
        });
""")
```

```
# Ecriture du second bloc du html (chargement des nuages)
with open("HTML_test.html", "a") as fichier:
    for nuage in choix_nuages:
        fichier.write(f"""

        Potree.loadPointCloud("./pointclouds/{nuage}/metadata.json", "{nuage}", e => {{
            let scene = viewer.scene;
            let pointcloud = e.pointcloud;

            let material = pointcloud.material;
            material.size = 1;
            material.pointSizeType = Potree.PointSizeType.ADAPTIVE;
            material.shape = Potree.PointShape.SQUARE;
            material.activeAttributeName = "rgba";

            scene.addPointCloud(pointcloud);

            viewer.fitToScreen();
        }});
""")

# Ecriture du dernier bloc du html (balises de fermeture)
with open("HTML_test.html", "a") as fichier:
    fichier.write("""


        </script>


  </body>
</html>
""")

webbrowser.open('localhost/test_python_html/HTML_test.html')
```

Annexe 6 : Schéma fonctionnel



Numérisation partielle de bâtiments et gestion des levés complémentaires

Mémoire d'Ingénieur C.N.A.M., Le Mans 2024

RÉSUMÉ

Ce mémoire présente les différentes étapes de recherche afin d'aboutir à une solution viable permettant de gérer, interagir et diffuser des données 3D. Ce travail s'inscrit dans le cadre du projet SwissRenov SwissRenov initié par la confédération Suisse, ayant pour objectif d'introduire un modèle de transformation circulaire de la chaine de valeur de la construction en mettant l'accent sur les friches industrielles comme source essentielle de ressources. Le modèle développé dans le canton du Jura, servira de démonstrateur pour une diffusion à l'échelle nationale. Dans cadre du sous-projet « Modélisation des bâtiments et circularité des ressources : outils et indicateurs », l'objectif est de créer une base de données spatiales, contenant les acquisitions réalisées sur le terrain afin de les diffuser aux acteurs participant au projet, n'ayant pas forcément de connaissances dans la manipulation des données 3D. Les axes de recherche sont le stockage des données issues de sources hétérogènes et possédant des caractéristiques différentes, la recherche d'une plateforme de diffusion permettant aux professionnels de réaliser des mesures simplement, dans des nuages de points 3D, et enfin, l'automatisation des différentes étapes de conversion, extraction d'information et création de la base de données au format GeoPackage.

Mots clés : SwissRenov, transformation circulaire, friches industrielles, base de données, données spatiales, nuages de points 3D, GeoPackage

---

SUMMARY

This dissertation presents the various stages of research in order to arrive at a viable solution for managing, interacting with and distributing 3D data. This work is part of the SwissRenov project SwissRenov, initiated by the Swiss Confederation, aims to introduce a circular transformation model for the construction value chain, focusing on brownfield sites as an essential source of resources. The model developed in the canton of Jura will serve as a demonstrator for nationwide dissemination. As part of the "Modelling buildings and the circularity of resources: tools and indicators" sub-project, the aim is to create a spatial database containing the data acquired in the field, so that it can be disseminated to project participants who do not necessarily have expertise in handling 3D data. The research focuses on the storage of data from heterogeneous sources with different characteristics, the search for a distribution platform enabling professionals to take simple measurements in 3D point clouds, and finally, the automation of the various stages of conversion, information extraction and creation of the database in GeoPackage format.

Keywords : SwissRenov, circular transformation, brownfield sites, database, spatial data, 3D point clouds, GeoPackage