Université Gustave Eiffel — LaSTIG — Géodata Paris, l'école des ingénieur·es géographes — IGN, Institut National de l'Information Géographique et Forestière

Géodata Paris

Cycle Ingénieur - 2ème année

LiDAR 2 LEGO

Rapport de Projet d'Initiation à la Recherche

Auteurs :
Dan Quê NGUYEN
Romain DE BLOTEAU VALCHUK

Commanditaires :
Corentin LE BIHAN GAUTIER (LASTIG)
Mathieu BRÉDIF (LASTIG)
Théo SZANTO (LASTIG)

Janvier 2026

Remerciements

Nous tenons à remercier chaleureusement nos encadrants, **Corentin Le Bihan Gautier**, **Mathieu Brédif** et **Théo Szanto** du laboratoire LASTIG, pour leur disponibilité, leurs conseils techniques avisés et leur bienveillance tout au long de ce projet.

Nous remercions également l'équipe pédagogique de **Géodata Paris** pour l'organisation de ce projet d'initiation à la recherche.

Enfin, nous adressons nos remerciements à l'**IGN** pour la mise à disposition des données LiDAR HD, ainsi qu'à la communauté **LDraw.org** pour sa bibliothèque de pièces open source et ses outils de modélisation, indispensables à la réalisation de ce projet.

Résumé

Dans un contexte où la médiation urbaine requiert des supports tangibles accessibles, ce projet explore la valorisation des données LiDAR HD via leur transformation automatisée en maquettes LEGO. Nous proposons une chaîne de traitement complète (*pipeline*), allant de la voxelisation sémantique d'un nuage de points à la génération de fichiers modèles 3D standardisés (LDraw). La méthodologie repose sur une discrétisation spatiale adaptative, couplée à des algorithmes de nettoyage topologique (graphes de connectivité) et de fusion de briques par approche gloutonne (*Greedy Stripe*). Les résultats démontrent une réduction significative du nombre de pièces par rapport à une approche naïve, tout en garantissant la stabilité structurelle de l'objet grâce à une gestion intelligente des fondations. Ce travail aboutit à un outil Open Source reproductible, permettant aux acteurs du territoire de matérialiser physiquement la donnée 3D.

Abstract

In a context where urban mediation requires accessible tangible tools, this project explores the valorization of LiDAR HD data through its automated transformation into physical LEGO models. We propose a complete processing pipeline, ranging from semantic voxelization of point clouds to the generation of standardized 3D model files (LDraw). The methodology relies on adaptive spatial discretization, coupled with topological cleaning algorithms (connectivity graphs) and a greedy brick merging strategy (*Greedy Stripe*). Results demonstrate a significant reduction in the number of parts compared to a naive approach, while ensuring the structural stability of the object through intelligent foundation management. This work results in a reproducible Open Source tool, enabling territorial stakeholders to physically materialize 3D data.

Table des matières

1 Introduction

Contexte et approche. Dans les processus de concertation citoyenne, la compréhension partagée du territoire est un prérequis souvent entravé par l'abstraction des outils numériques. Pour devenir de véritables outils de médiation, les supports doivent être intelligibles par tous. Bien que les maquettes tangibles favorisent ce dialogue par leur présence physique, comme le souligne l'un de nos commanditaires : Corentin Gautier [1], leur production artisanale demeure complexe et coûteuse. De plus, la création de maquettes classiques (artisanales ou impression 3D) se heurte souvent à l'absence de modèles numériques 3D préexistants, nécessitant une lourde préparation en amont. Ce projet propose de lever ce verrou en automatisant leur création via un support universel : la brique LEGO. Contrairement à l'impression 3D figée, le LEGO offre une modularité permettant à l'usager de corriger ou mettre à jour la maquette, favorisant une appropriation active. Notre approche convertit des nuages de points LiDAR HD en structures constructibles via un pipeline Open Source [2]. L'algorithme se veut paramétrable et reproductible, permettant à tout acteur territorial de générer ses propres supports sans modélisation préalable.

2 État de l'art et objectifs

Positionnement scientifique. L'usage de voxels pour structurer des données 3D est éprouvé [3] et s'impose ici par l'analogie géométrique avec la brique 1×1. Cependant, la simple conversion ne garantit pas la stabilité physique. Notre méthodologie s'appuie sur les travaux de Testuz et al. [4] concernant les graphes de connectivité pour la constructibilité, tout en adaptant les stratégies d'optimisation d'agencement (*merging*) explorées par Lee et al. [5] et Yun [6] aux contraintes massives de la donnée géospatiale.

Objectifs techniques. Le pipeline Python développé assure le traitement complet du nuage de points vers la maquette :

— Voxelisation paramétrable (échelle, densité) avec conservation de la sémantique (vote majoritaire).
— Traitements structurels pour le nettoyage du bruit et la consolidation statique.
— Fusion des briques via un solveur glouton (*Greedy*) optimisant la solidité.
— Exportation au format standardisé LDraw (*.ldr*) pour l'exploitation de la maquette.

3 Méthodologie

La chaîne de traitement, intégralement documentée sur le dépôt GitHub du projet [2], suit un workflow illustré ci-dessous.

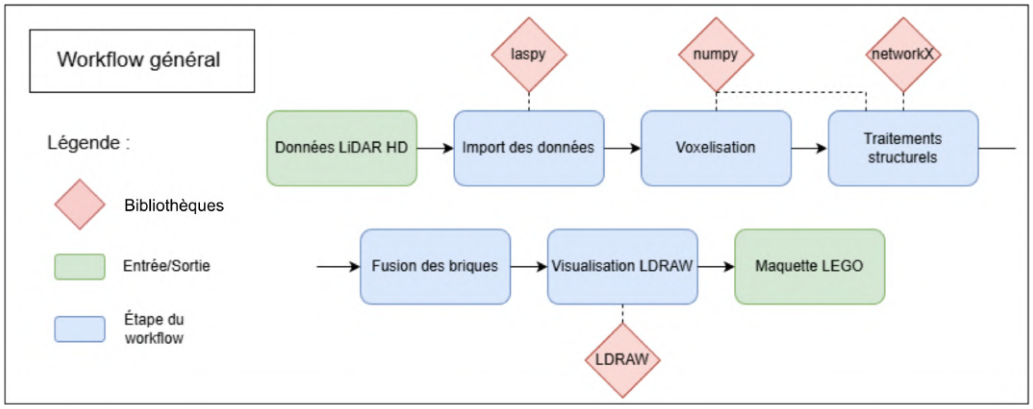

FIGURE 1 – Workflow général de la chaîne de traitement.

3.1 Acquisition et gestion des données

Source et ingestion. Pour répondre au besoin de données 3D Open Source et disponibles sur tout le territoire, ce projet exploite le LiDAR HD de l'IGN [7]. Si cette source allie haute densité (10 points/m^2) et richesse sémantique, sa massivité impose des contraintes fortes : les traitements haute fréquence nécessaires à la précision géométrique doivent être adaptés à l'échelle de représentation visée pour maîtriser l'explosion des temps de calcul. L'importation via `laspy` et le filtrage (script `LIDAR_numpy.py`) privilégient donc des tableaux `NumPy` optimisés pour la performance.

Stratégie d'échantillonnage. Afin d'adapter le ratio temps de calcul/surface aux besoins de médiation (quartier vs territoire), trois modes d'importation sont implémentés via le script `donnees_echantillonnees_LIDAR.py` [2] : la Dalle Complète, l'Échantillonnage Aléatoire (pour validation techniques) et l'**Échantillonnage Rectangulaire**. Ce dernier mode est privilégié car il permet de cibler précisément une emprise (ex : îlot urbain) via ses coordonnées.

3.2 Voxelisation et discrétisation spatiale

La voxelisation transforme le nuage continu en structure discrète via une grille régulière (*Regular Voxel Grid*), implémentée dans le script `LIDAR_couches.py` [2].

Résolution et filtrage. Pilotée par le paramètre `taille_xy`, la résolution impose un arbitrage critique entre fidélité urbaine et complexité d'assemblage (cf. Annexe A, Figure 3). Le peuplement s'opère par calcul vectoriel ; la validation d'un voxel est conditionnée par un seuil de densité minimal filtrant le bruit de mesure, conformément aux travaux de Cohen-Or et Kaufman [8].

Anisotropie (Ratio Z). Contrairement à l'approche scientifique isotrope, la modélisation tangible intègre les contraintes du support [1]. Le paramètre `lego_ratio` ($\Delta z = \Delta x \times$ `lego_ratio`) découple la résolution verticale. Fixé à 1.2 pour le respect métrique (cf. Figure 4), il peut être réduit (< 1.0) pour accentuer le relief (*z-scale*) à des fins de médiation (cf. Annexe A, Tableau 1).

Gestion sémantique. L'attribution (Bâtiment, Végétation, Sol) repose sur un vote majoritaire pondéré (cf. Annexe A, Figure 5). Une règle conditionnelle écarte la classe « Non classé » dès qu'une classe sémantique forte (ex : Végétation) est significative, densifiant ainsi l'information utile (cf. Annexe A, Figure 6).

3.3 Traitements structurels et analyse topologique

La donnée voxélisée brute, sujette aux artefacts (bruit, occultations), ne répond pas encore aux exigences de constructibilité physique. Le post-traitement est assuré par le script `LIDAR_traitement.py` [2], qui convertit la grille en un graphe 6-connexe via la librairie `NetworkX`, simplifiant nos calculs. Ce choix de modélisation, préconisé par Cohen-Or et Kaufman [8], est impératif pour garantir la stabilité de l'assemblage LEGO en excluant les voxels connectés uniquement par une arête.

Nettoyage et continuité (Gravité et Tobler). La stabilité gravitaire est assurée par un filtrage des composantes connexes (fonction `graphe_filtre_sol`) qui supprime tout sous-graphe non relié au « Sol » (Code 2), éliminant ainsi les artefacts aériens (cf. Annexe A, Figure 7). Parallèlement, la gestion des voxels « Non-Classés » (Code 1) exploite l'hypothèse de continuité spatiale de Tobler. Plutôt que la suppression, la fonction `corriger_voxels_non_classes_iteratif` propage les classes fortes (Bâtiment, Végétation) aux nœuds adjacents. Cette « contamination » locale comble les micro-lacunes sans miter le modèle (cf. Annexe B, Figure 8).

Correction morphologique. Pour pallier les occultations verticales du LiDAR, l'algorithme `remplir_trous_verticaux` détecte et interpole les ruptures d'axe Z dans les colonnes de « Bâtiment » (Code 6). Cette opération restaure l'intégrité des murs porteurs et ferme les volumes (cf. Annexe B, Figure 9).

Optimisation matérielle (Coque et Piliers). Inspirée de Testuz et al. [4], la stratégie d'évidement (`ajouter_sol_coque_pillier`) concilie économie de matière et stabilité via deux opérations :

— **Génération de la coque :** Les voxels « intérieurs » (entourés de 6 voisins) sont supprimés par érosion pour ne garder que l'enveloppe.

— **Piliers de soutènement :** Un masque périodique (Pattern Modulo) protège certains voxels de l'érosion, créant des colonnes porteuses internes pour soutenir les toitures (cf. Annexe B, Figure 10).

Cette configuration assure la transmission verticale des charges tout en économisant des briques.

3.4 Algorithme de fusion

L'algorithme vise à réduire la complexité du modèle (briques 1×1) tout en maximisant sa solidité structurelle.

3.4.1 Initialisation gloutonne

Les travaux de Lee et al. [5] reposent sur un algorithme génétique dont la convergence dépend de la qualité de la population initiale (« Génération 0 ») et d'une fonction de coût définissant la robustesse structurelle. Nous nous sommes concentrés sur la constitution optimale de cette « Génération 0 » via un algorithme glouton déterministe. Ce choix garantit un modèle valide dès la première itération et réduit significativement le nombre de pièces par rapport au modèle brut.

3.4.2 Définition du voisinage

Conformément à Testuz et al. [4], l'assemblage est modélisé par un graphe d'adjacence où chaque brique est un sommet (cf. Annexe C, Figure 11). Deux briques sont définies comme « fusionnables » dans le script `merge.py` [2] si elles valident trois critères cumulatifs.

Premièrement, l'**adjacence spatiale** (fonction `are_neighbors`) impose le partage d'une face plane pour assurer la continuité physique de la matière. Deuxièmement, l'**homogénéité des attributs** exige une classification spectrale identique entre voisins, préservant ainsi la sémantique du nuage de points. Enfin, la **légalité du successeur** vérifie que la pièce résultante appartient au catalogue manufacturé (`VALID_SIZES`), une contrainte de constructibilité impérative selon Yun et al. [6] pour garantir la réalisabilité du modèle.

3.4.3 Fusion

Contrairement à l'approche aléatoire de Testuz et al. [4], le script `solver.py` [2] implémente une stratégie déterministe par balayage de couches (rasterisation) pour maximiser la régularité (cf. Annexe C, Figure 12).

Partitionnement 1D (Smart Stripe). La fonction `get_best_partition` identifie les segments continus de voxels (briques 1×1 de même couleur) et calcule leur découpe mathématique optimale pour minimiser le nombre de pièces.

Expansion latérale (Fusion 2D). Une passe secondaire fusionne les briques $1 \times n$ adjacentes latéralement en modules $2 \times n$. Cette géométrie maximise la surface de tenon, renforçant la friction et la résistance au cisaillement.

Alternance orthogonale. Selon les règles de Gower et al. [9], l'axe de balayage s'inverse à chaque niveau. La couche N traitée selon l'axe X (puis Y), suivie d'une couche $N+1$ traitée selon l'axe Y (puis X). Ce croisement mécanique des joints prévient les « coups de sabre » (failles verticales) et verrouille la structure.

3.4.4 Évaluation par fonction de coût

La validité structurelle du modèle généré est quantifiée par une fonction de coût implémentée dans le script `cost_function.py` [2], inspirée de Gower, Heydtmann et Petersen [9]. Cette métrique agrège trois pénalités que l'algorithme cherche à minimiser :

— **Pénalité de croisement (P1) :** Mesure si les briques d'une couche L sont perpendiculaires à celles de la couche L - 1.
— **Pénalité de jointure (P2) :** Sanctionne les « coups de sabre », où les limites de briques sont alignées verticalement sur deux couches.
— **Pénalité de connectivité (P3) :** Pénalise les « jonctions en T » qui ne sont pas au centre, sur une même couche.

Dans notre implémentation, cette fonction sert d'outil de mesure de performance pour comparer l'instabilité du modèle 1×1 initial face à la robustesse du modèle fusionné final.

3.5 Visualisation : Le standard LDraw

L'étape finale sérialise le modèle optimisé au format standard ouvert LDraw (`.ldr`). Faute de bibilothèques adaptées au traitement de matrices massives, des modules d'export spécifique ont été développés (scripts `LIDAR_LDRAW.py` et fonction `export_to_ldr` dans le script `solver.py`[2]) afin de gérer la géométrie complexe des briques fusionnées.

La conversion de l'espace voxel vers l'unité LDraw (LDU) nécessite une transformation affine combinant mise à l'échelle (20×24 LDU) et inversion de l'axe vertical pour la conformité CAO : $X_{ldr} = 20 \cdot i, \quad Z_{ldr} = 20 \cdot j, \quad Y_{ldr} = -24 \cdot k$ (cf. Annexe A, Figure 4)

Enfin, la sémantique est préservée via un mapping colorimétrique (Codes ASPRS → Palette LEGO). Le fichier ASCII généré au format *.ldr* respecte la syntaxe standard LDRAW : 1 <Couleur> <x> <y> <z> <Matrice_Rotation> <ID_Pièce.dat> (cf. Annexe C, Figure 13), garantissant une interopérabilité avec l'écosystème logiciel existant (LDView, Stud.io).

4 Résultats et discussion

Les performances de la chaîne de traitement ont été évaluées principalement sur le site de Géodata Paris, choisi pour sa géométrie connue et ses typologies variées (bâti complexe avec vitrage, végétation). L'analyse se décompose en trois axes : la fidélité de la reconstruction face aux biais du capteur, l'efficience structurelle des algorithmes et la pertinence pour la médiation.

Impact critique de la résolution spatiale. L'absence de « résolution universelle » impose d'ajuster la taille du voxel (`taille_xy`) selon le besoin client (urbaniste, grand public) et l'échelle de la zone. Nos tests mettent en évidence deux régimes distincts :

— **Cas A : Échelle quartier ($<$ 200m).** La résolution de 1 mètre (brique 1×1) est le seuil critique. En dessous (0.5m), le modèle souffre de « trous » interstitiels dus au bruit de mesure. Entre 1m et 2m, la fidélité architecturale est optimale grâce à la conservation des volumes principaux tout en lissant le bruit de surface. Au-dessus de 2m, l'abstraction devient forte mais le coût en briques est moins élevé.
— **Cas B : Échelle territoriale (1 km^2).** Une résolution de 4 mètres offre le meilleur compromis pour une dalle complète, évitant l'explosion combinatoire des temps de calcul.

4.1 Évaluation de la performance structurelle.

L'apport majeur de ce travail réside dans la transformation d'un nuage de points instable en un objet physique autoportant. **Comparaison des Stratégies de Fondation.** L'algorithme de consolidation (fonction `ajouter_sol_coque_pillier`) a été confronté aux approches classiques.

— **Gain matériel.** Sur l'échantillon test, la méthode « Coque + Piliers » permet d'économiser 64% de briques par rapport au remplissage massif, rendant le coût de production acceptable pour une collectivité.

— **Stabilité physique.** Contrairement à la « Coque vide » qui présente des flèches de déformation critiques sur les grandes surfaces planes (toitures, places), l'ajout de piliers (taille et espacement paramétrable) assure une transmission des charges verticale sans compromettre l'économie de matière (cf. Annexe B, Figure 10.

Cette optimisation matérielle se double d'une efficacité computationnelle : le temps de génération varie de quelques secondes pour un bâtiment isolé à quelques dizaines de secondes pour un îlot urbain (selon la résolution), rendant l'outil compatible avec des itérations rapides.

Métriques de solidité (Cost Function). L'algorithme d'assemblage (fonction `solve_greedy_stripe`) ne se contente pas de réduire le nombre de pièces ; il améliore la cohésion mécanique. L'évaluation via la fonction de coût (script `cost_function.py`[2]) montre une amélioration nette des critères de solidité :

— **Croisement et connectivité ($P1$ & $P3$).** L'alternance stricte des couches (Horizontale/Verticale) impose un croisement des joints (« verrouillage »). Cela garantit mécaniquement qu'une brique repose sur plusieurs briques inférieures (connectivité), éliminant le risque d'effondrement par colonnes isolées.

— **Pénalité de « coup de sabre » ($P2$).** L'algorithme minimise, bien que sans l'annuler totalement, l'alignement vertical des joints, critique pour la résistance au cisaillement.

Performance de fusion. Le passage de la grille de voxels au modèle optimisé permet une réduction drastique du nombre d'éléments. Pour le modèle complet de l'école à 1m de résolution, on passe de $\approx$ 90 000 briques unitaires à $\approx$ 23 000 briques fusionnées, soit une réduction de 74% (cf. Annexe C, Figure 14). Cette optimisation rend le montage manuel envisageable.

4.2 Discussion : Limites et modularité

Analyse critique de la connectivité. Une validation externe via le logiciel *BrickLink Studio* a mis en évidence des limites structurelles invisibles à l'œil nu. L'analyse de connectivité révèle des sous-ensembles disjoints (en rose, voir Annexe D, Figure 16, 17, 18). La cause identifiée est la régularité du motif de génération du sol (briques 2×4 alignées) et de l'algorithme glouton. Contrairement à une approche génétique qui introduirait de l'aléatoire pour maximiser l'enchevêtrement, notre approche déterministe tend à créer des lignes de faille verticales (« plans de clivage ») isolant certaines sections (arbres, toitures) du noyau central.

L'Anisotropie comme levier de médiation. Au-delà de la fidélité métrique stricte, ce projet a exploré l'impact du ratio vertical sur la compréhension du territoire. Le paramètre `lego_ratio` n'est pas une simple constante technique, mais une variable d'ajustement sémantique :

— **Ratio métrique (1.2) :** Garantit la fidélité géométrique stricte. Ce mode est impératif pour les usages techniques (urbanisme) où le respect exact des pentes et des hauteurs relatives est la priorité.

— **Ratio exagéré (< 1.2) :** Agit comme un amplificateur cognitif. Cette distorsion volontaire sert à « forcer le trait » pour révéler un relief ou une densité peu perceptible à l'échelle 1 : 1, rendant le territoire immédiatement lisible pour le grand public (cf. Annexe C, Figure 15).

Compromis algorithmique : Greedy vs Génétique. Le choix de l'algorithme glouton privilégie la rapidité (complexité $O(N)$) face aux algorithmes génétiques, trop lents pour des données massives.

Adaptabilité et paramétrage. La structure modulaire du code (`main.py`[2]) transforme l'algorithme en un outil flexible capable de répondre à des contraintes variées :

— **Sémiologie graphique :** Choix entre une palette « Standard » (attractivité grand public) ou monochrome (support de projection pour l'urbanisme prospectif [1]).

— **Contraintes logistiques (Catalogue LDraw) :** Le paramètre `INVENTAIRE_BRIQUES` permet de restreindre le solveur à un catalogue défini de briques LDraw. L'utilisateur peut ainsi adapter la construction à son stock réel ou limiter la complexité d'approvisionnement.

— **Arbitrage coût/solidité :** La sélection du mode de consolidation (`TYPE_CONSOLIDATION`) offre un levier de contrôle sur le budget matériel, permettant de privilégier l'économie (Mode Coque) ou la durabilité maximale (Mode Piliers) selon la finalité de la maquette (exposition temporaire ou permanente).

4.3 Validation tangible

La finalité du projet étant la production d'un objet physique, la validation ultime repose sur l'assemblage réel et sa comparaison avec la donnée source.

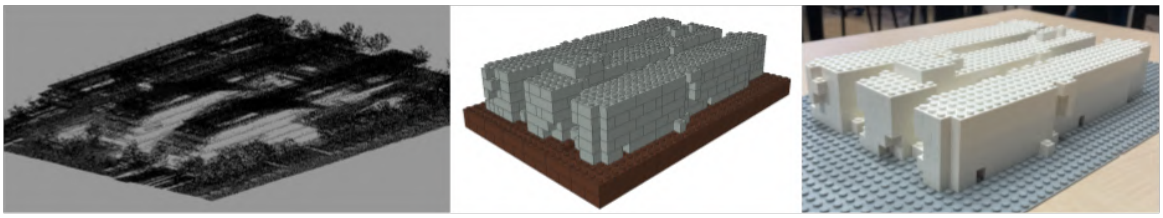

FIGURE 2 – Entrées et sorties de notre pipeline : passage de la donnée LiDAR aux visuels.

5 Conclusion

Ce Projet d'Initiation à la Recherche a permis d'explorer la conception de maquettes tangibles de territoire, en évaluant l'impact de choix critiques comme l'échelle ou la stabilité. En intégrant ces contraintes d'ingénierie au traitement de la donnée LiDAR HD, cette démarche dépasse la simple discrétisation spatiale pour transformer une information technique brute en un objet social accessible.

Au bilan, ce travail valide la faisabilité technique d'un workflow automatisé complet, allant de la donnée LiDAR HD brute à la maquette physique. L'algorithme développé démontre une efficacité réelle, offrant une optimisation matérielle majeure (réduction de plus de 70% du nombre de pièces par fusion) et garantissant la reproductibilité des résultats via un code Open Source disponible sur GitHub [2].

Enfin, ce pipeline constitue un socle évolutif pour de futurs travaux de recherche. L'outil ouvre notamment la voie à la génération automatique de notices de montage pas-à-pas (via LPub3D), à des approches d'optimisation hybrides (combinaison Glouton / Génétique ciblé), et au développement d'une interface QGIS pour démocratiser son usage auprès des services techniques.

Références

[1] Corentin Le-Bihan Gautier. « Représentations Dynamiques, Virtuelles et Tangibles de La Ville ». Thèse de doct. Université Lumière - Lyon II, 2025.

[2] Dan Quê Nguyen et Romain de Bloteau Valchuk. *LIDAR_2_LEGO*. Code source et documentation du projet (GitHub). 2026. URL : `https://github.com/DanQue69/LiDAR_2_LEGO/tree/main` (visité le 10/01/2026).

[3] Mitko Aleksandrov, Sisi Zlatanova et David J. Heslop. « Voxelisation Algorithms and Data Structures : A Review ». In : *Sensors* 21.24 (2021).

[4] Romain Testuz, Yuliy Schwartzburg et Mark Pauly. « Automatic Generation of Lego Sculptures ». In : *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2013)* 32.4 (2013).

[5] Sangyeop Lee et al. « Finding an Optimal LEGO® Brick Layout of Voxelized 3D Object Using a Genetic Algorithm ». In : (2015).

[6] Grim Yun et al. « Legorization with Multi-Height Bricks from Silhouette-Fitted Voxelization ». In : *Proceedings of the Computer Graphics International Conference*. ACM, 2017.

[7] IGN. *Descriptif de contenu du produit LiDAR HD*. Institut National de l'Information Géographique et Forestière. 2024. URL : `https://geoservices.ign.fr/lidarhd` (visité le 10/01/2026).

[8] Daniel Cohen-Or et Arie Kaufman. « Fundamentals of Surface Voxelization ». In : *Graphical Models and Image Processing* 57.6 (1995).

[9] Rebecca Gower, Agnes Heydtmann et Henrik Petersen. « Lego : Automated Model Construction ». In : (1998). University of Oxford.

[10] Harith Aljumaily et al. « Point Cloud Voxel Classification of Aerial Urban LiDAR Using Voxel Attributes ». In : *International Journal of Applied Earth Observation and Geoinformation* 118 (2023).

A Annexe A : Voxelisation et filtrage

Paramètre	Justification Scientifique et Technique
`taille_xy`	**Résolution Spatiale.** Compromis optimal entre la fidélité du détail urbain et la limitation du nombre de pièces (complexité d'assemblage).
`densite_min`	**Seuil d'Existence.** Valeur minimale pour conserver la connectivité des structures fines (câbles, murets) décrites par Aljumaily [10], au prix d'un bruit résiduel accepté.
`lego_ratio`	**Contrainte du Support.** Définit l'anisotropie du voxel nécessaire pour respecter la géométrie non-cubique des briques (propriétés LDraw).
`hauteur_couche`	**Slicing.** Calculé dynamiquement ($\Delta X \times lego\_ratio$) pour garantir l'assemblage physique couche par couche.

TABLE 1 – Paramètres critiques de l'algorithme de voxelisation.

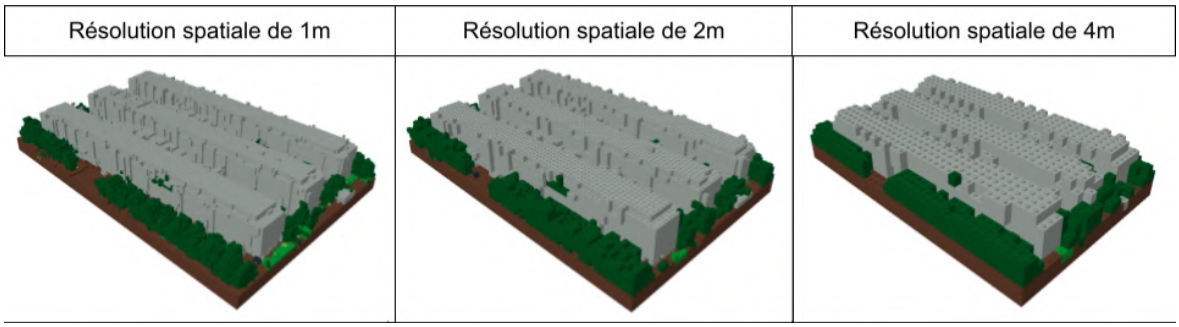

FIGURE 3 – Impact de la résolution spatiale sur la morphologie du bâti.

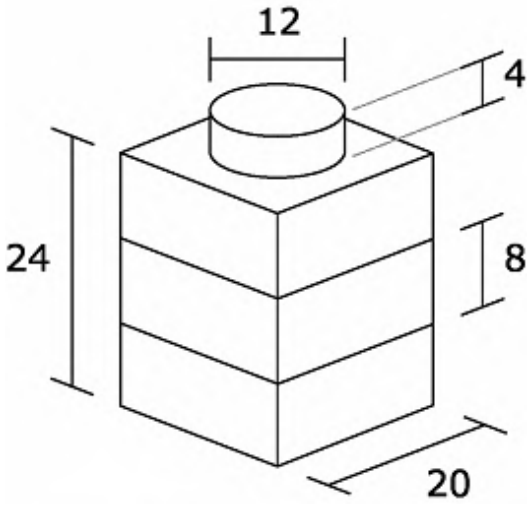

FIGURE 4 – Géométrie et système de coordonnées d'une brique LDraw. Source : LDraw.org.

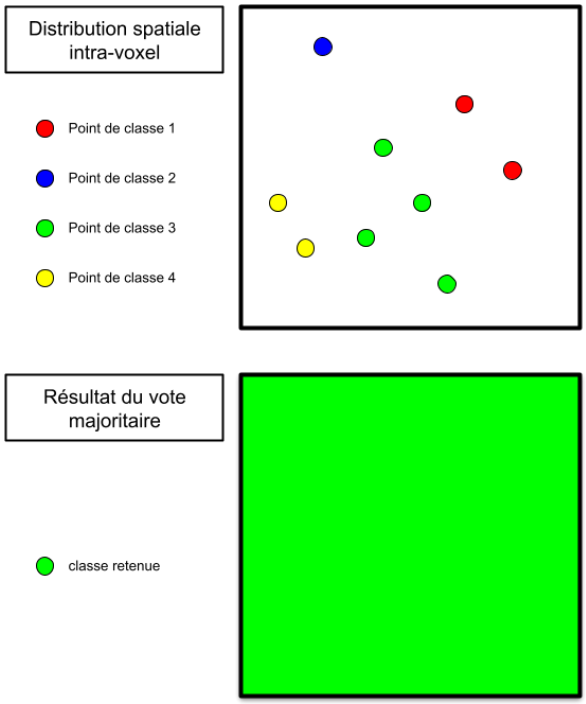

FIGURE 5 – Cas A : Vote majoritaire simple.

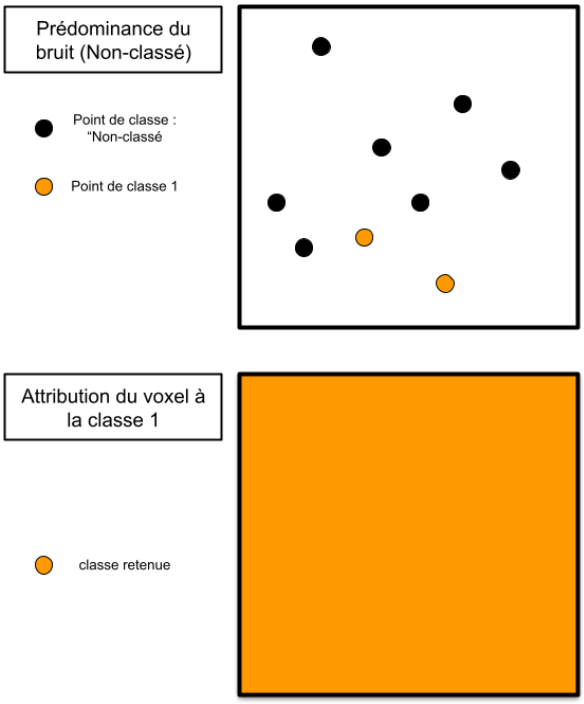

FIGURE 6 – Cas B : Vote conditionnel.

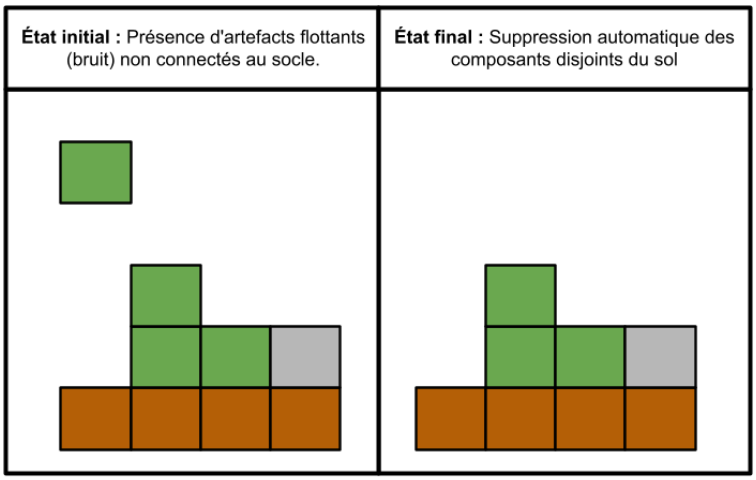

FIGURE 7 – Principe du nettoyage topologique (suppression des artefacts).

B Annexe B : Traitements structurels

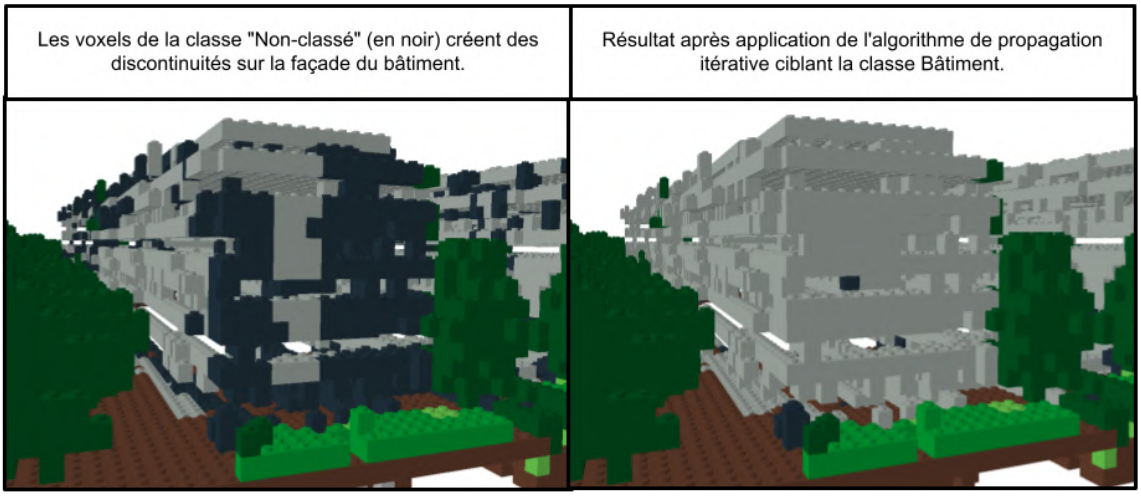

FIGURE 8 – Évaluation de l'algorithme de correction sémantique.

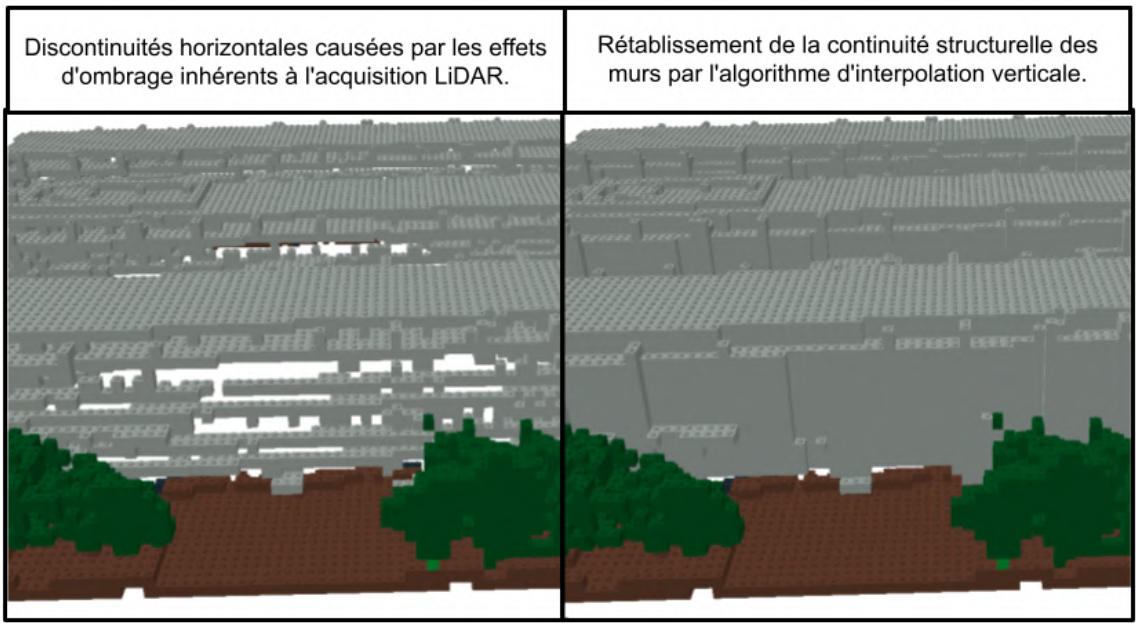

FIGURE 9 – Correction morphologique des façades par comblement vertical.

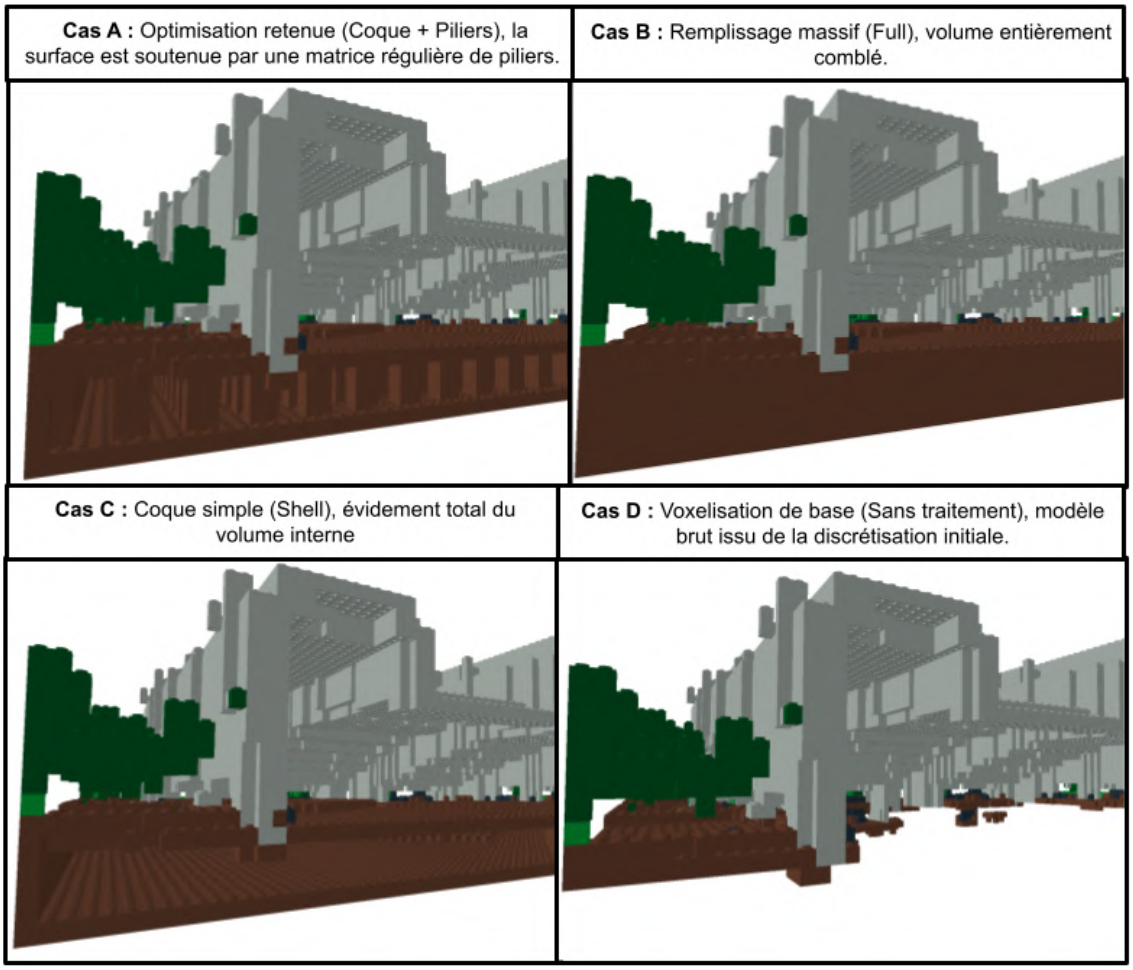

FIGURE 10 – Analyse comparative des stratégies de consolidation.

C Annexe C : Algorithme de Fusion et Export

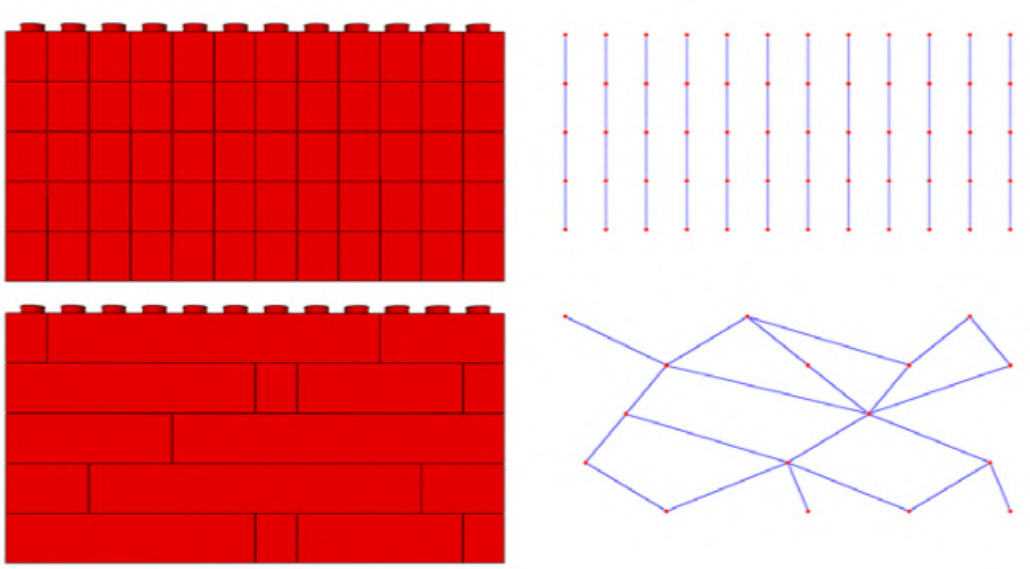

FIGURE 11 – Modélisation par graphe d'adjacence. Source : Testuz et al. [4].

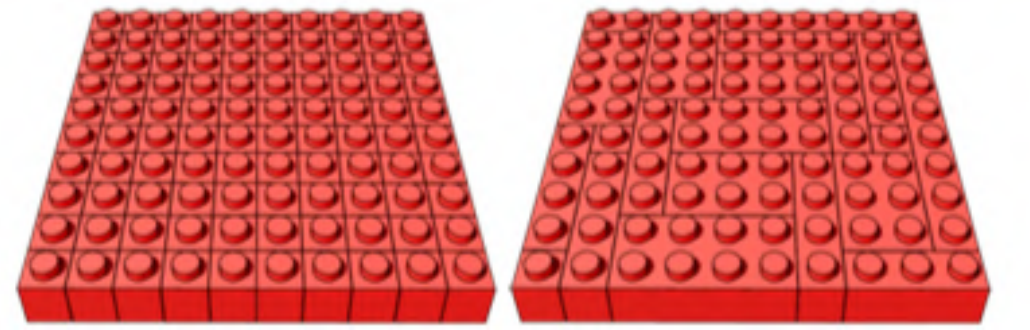

FIGURE 12 – Résultat de l'algorithme de fusion (Vue de dessus). Gauche : Couche brute (1×1). Droite : Couche optimisée ($1 \times n$ et $2 \times n$). Source : Testuz et al. [4].

03_FINAL_exemple.laz_ECHANTILLON-RECTANGLE_COULEUR.ldr

```
0 Optimized LEGO Model
0 Name: C:\Users\roro7\Desktop\Romain\ENSG_cours\
0 Author: Solver Smart V4
1 6 40.00 0.00 20.00 1 0 0 0 1 0 0 0 1 3001.dat
1 6 40.00 0.00 60.00 1 0 0 0 1 0 0 0 1 3001.dat
1 6 40.00 0.00 100.00 1 0 0 0 1 0 0 0 1 3001.dat
1 6 40.00 0.00 140.00 1 0 0 0 1 0 0 0 1 3001.dat
1 6 40.00 0.00 180.00 1 0 0 0 1 0 0 0 1 3001.dat
1 6 40.00 0.00 220.00 1 0 0 0 1 0 0 0 1 3001.dat
1 6 40.00 0.00 260.00 1 0 0 0 1 0 0 0 1 3001.dat
1 6 40.00 0.00 300.00 1 0 0 0 1 0 0 0 1 3001.dat
1 6 40.00 0.00 340.00 1 0 0 0 1 0 0 0 1 3001.dat
1 6 40.00 0.00 380.00 1 0 0 0 1 0 0 0 1 3001.dat
1 6 40.00 0.00 420.00 1 0 0 0 1 0 0 0 1 3001.dat
1 6 40.00 0.00 460.00 1 0 0 0 1 0 0 0 1 3001.dat
1 6 40.00 0.00 500.00 1 0 0 0 1 0 0 0 1 3001.dat
1 6 40.00 0.00 540.00 1 0 0 0 1 0 0 0 1 3001.dat
1 6 40.00 0.00 580.00 1 0 0 0 1 0 0 0 1 3001.dat
1 6 40.00 0.00 620.00 1 0 0 0 1 0 0 0 1 3001.dat
1 6 40.00 0.00 660.00 1 0 0 0 1 0 0 0 1 3001.dat
1 6 40.00 0.00 700.00 1 0 0 0 1 0 0 0 1 3001.dat
1 6 40.00 0.00 740.00 1 0 0 0 1 0 0 0 1 3001.dat
1 6 40.00 0.00 780.00 1 0 0 0 1 0 0 0 1 3001.dat
1 6 40.00 0.00 820.00 1 0 0 0 1 0 0 0 1 3001.dat
1 6 40.00 0.00 860.00 1 0 0 0 1 0 0 0 1 3001.dat
1 6 40.00 0.00 900.00 1 0 0 0 1 0 0 0 1 3001.dat
1 6 40.00 0.00 940.00 1 0 0 0 1 0 0 0 1 3001.dat
1 6 40.00 0.00 980.00 1 0 0 0 1 0 0 0 1 3001.dat
1 6 40.00 0.00 1020.00 1 0 0 0 1 0 0 0 1 3001.dat
1 6 40.00 0.00 1060.00 1 0 0 0 1 0 0 0 1 3001.dat
1 6 40.00 0.00 1100.00 1 0 0 0 1 0 0 0 1 3001.dat
```

FIGURE 13 – Extrait du fichier source .ldr généré. Chaque ligne est une instruction de brique standardisée.

D Annexe D : Analyse des résultats et validation

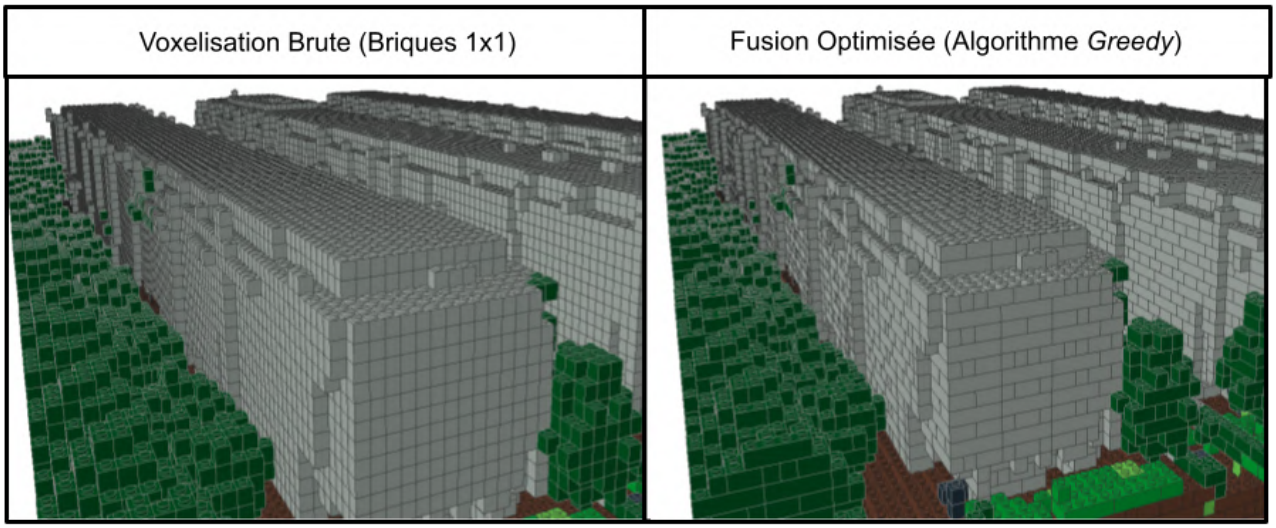

FIGURE 14 – Visualisation avant/après algorithme de fusion.

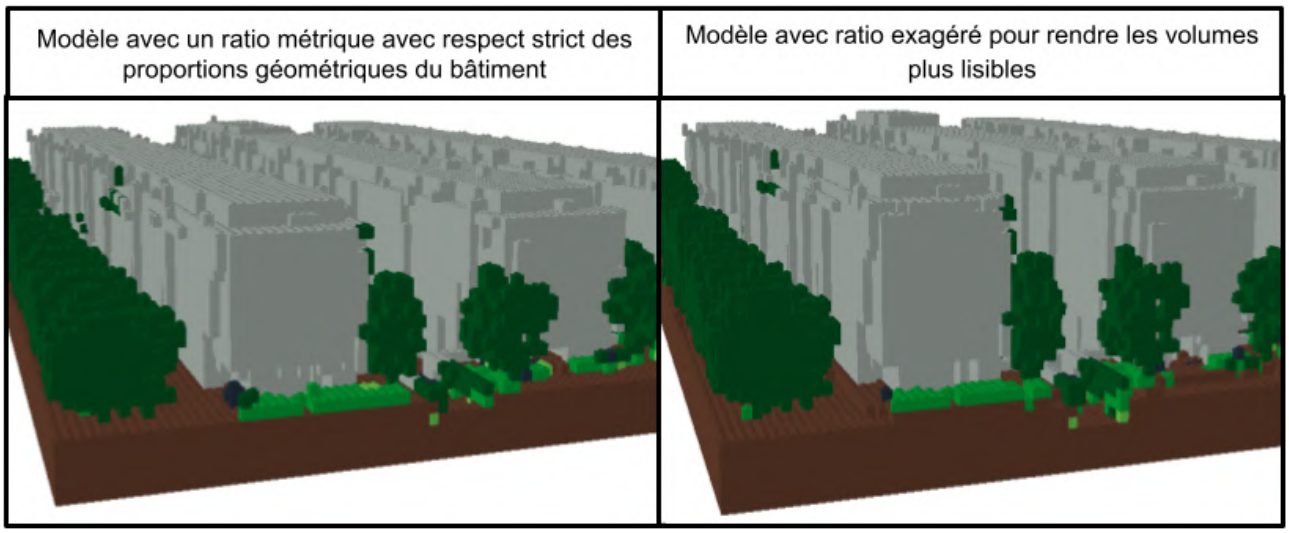

FIGURE 15 – Impact de l'Anisotropie (Z-Scale).

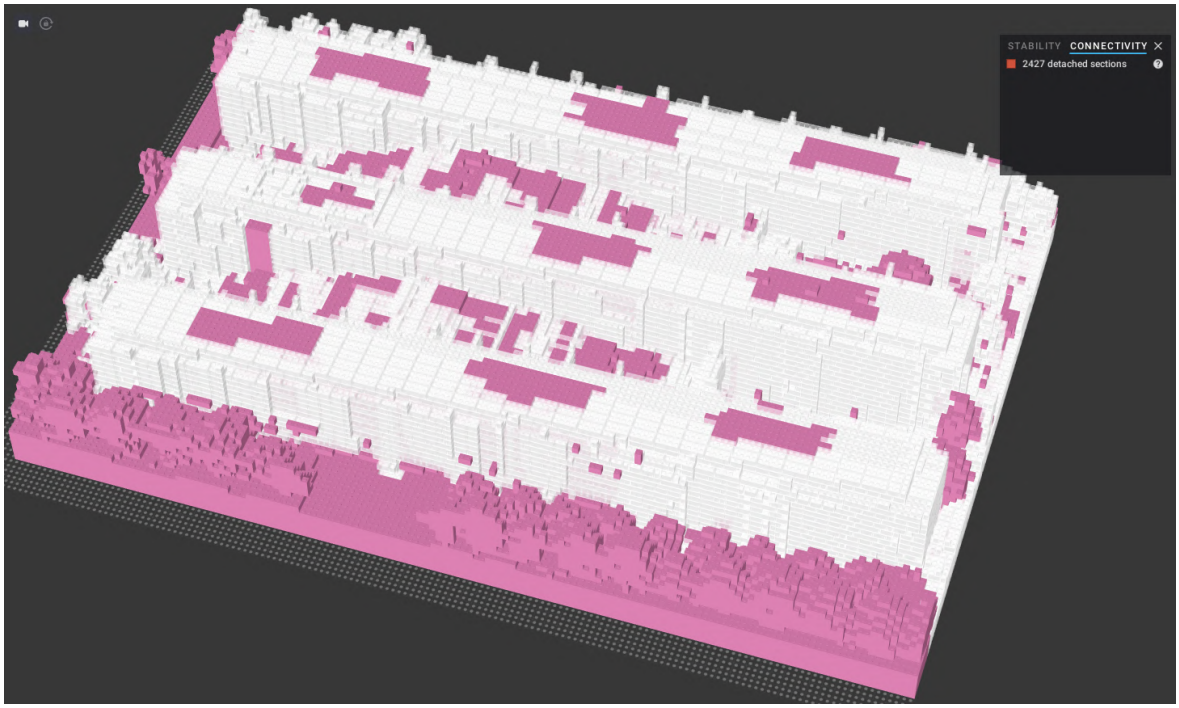

FIGURE 16 – Analyse de connectivité. En rose : sections disjointes du cœur du modèle.

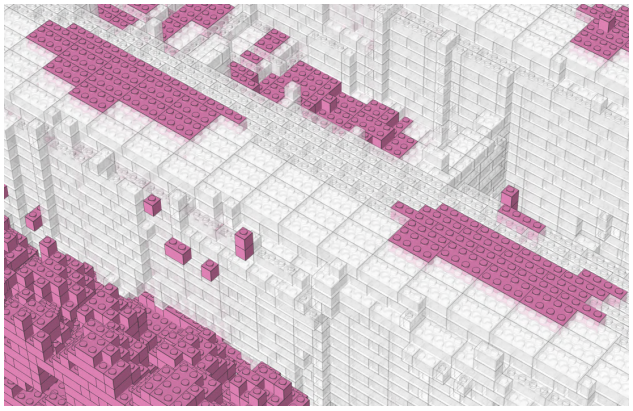

FIGURE 17 – Zoom Toiture : Défaut d'enchevêtrement vertical.

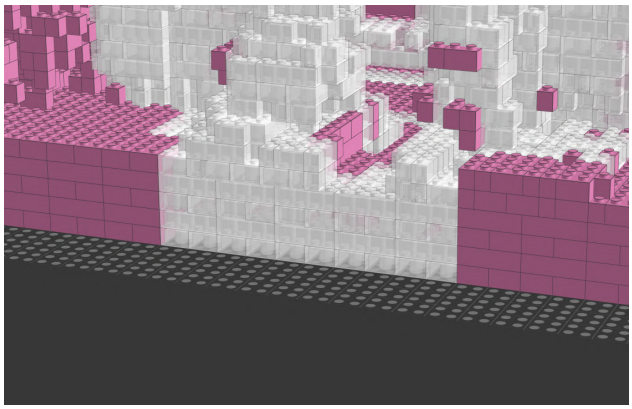

FIGURE 18 – Zoom Sol : Le motif régulier crée des ruptures.