

# TP5 - Nuages de Points et Modélisation 3D

Chloé Habasque - Colin Coërchon

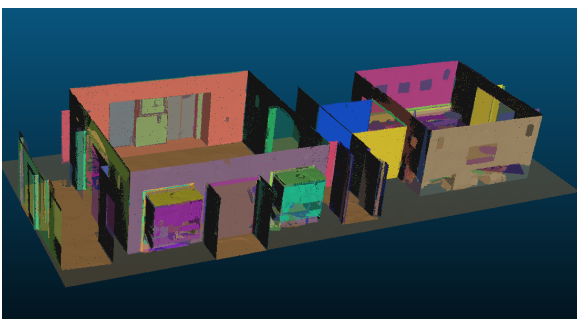
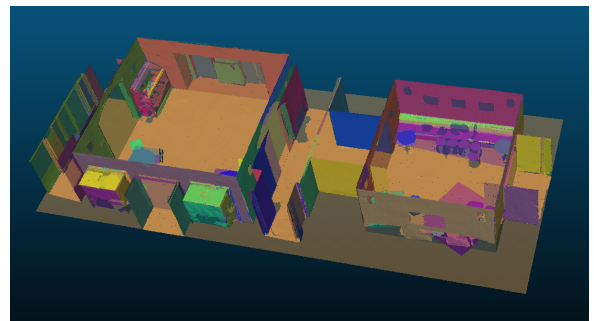
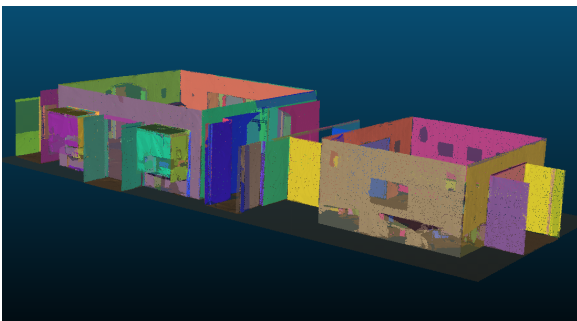
19 Février 2025

## 1 Test RANSAC Shape detection in CloudCompare

### Question 1

Paramètres sélectionnés pour la meilleur segmentation :

| Paramètre   | Valeur |
|-------------|--------|
| $e$         | 0.05   |
| $b$         | 0.1    |
| $a$         | 10%    |
| Probabilité | 0.01   |



Le temps de recherche est de 7.124s pour trouver 185 plans. On a cherché les paramètres qui permettent d'obtenir le plus de plans possible, correspondant aux surfaces des murs et du sol. On observe aussi que l'algorithme à trouver des surfaces planes pour les portes et les objets tels que les tables et les chaises.

## 2 Implement RANSAC in Python

### Question 2

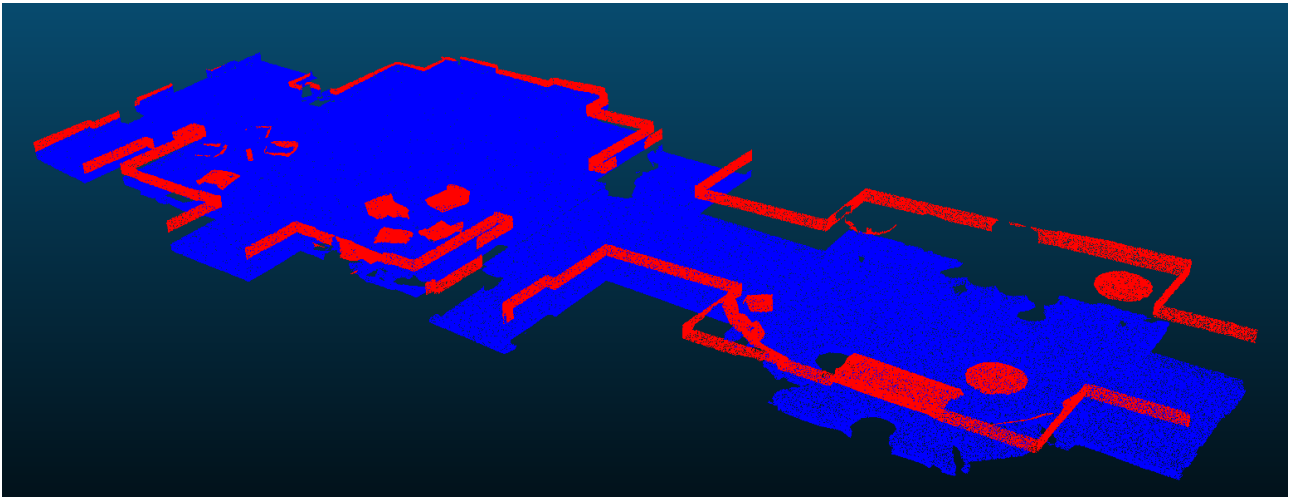


FIGURE 1 – 2 plans found with RANSAC

Nous avons obtenu deux plans avec notre algorithme RANSAC. Le premier (en **bleu**), correspond au sol, et le deuxième plan détecté (en **rouge**) est une surface horizontale globalement alignée avec les tables de l'appartement.

C'est le résultat attendu, mais ce n'est malheureusement pas le résultat souhaité. Le premier plan détecté correspond bien au plan le plus important de l'appartement : le sol. Mais pour le deuxième plan, **nous aurions préféré l'extraction d'un plan vertical, comme un mur**.

Cette sélection inattendue s'explique par la nature de l'algorithme, qui **privilégie les surfaces les plus étendues** et les mieux représentées dans le nuage de points. Ainsi, les tables, occupant une large zone horizontale (avec des parties de murs), ont été détectées avant un mur, dont les points sont moins nombreux.

### Question 3

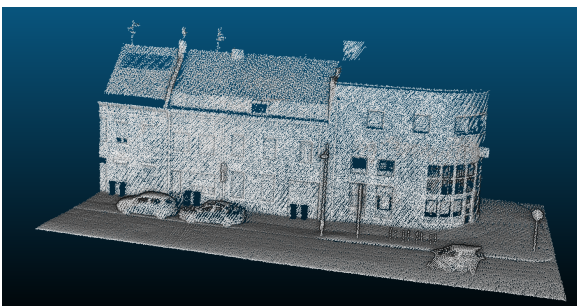


FIGURE 2 – Lille Street

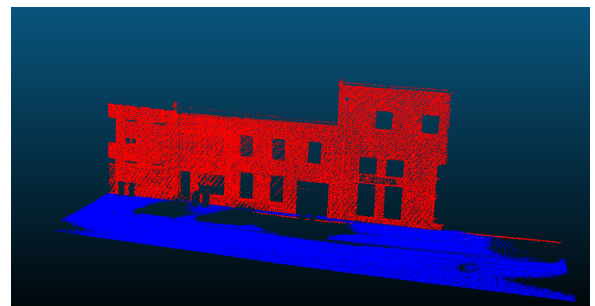


FIGURE 3 – Application de l'algorithme RANSAC sur Lille Street

Paramètres utilisés :

| Paramètre    | Valeur |
|--------------|--------|
| nb_draws     | 100    |
| threshold_in | 0.1    |
| nb_planes    | 2      |

Dans cet exemple, l'algorithme détecte bien les 2 plans principales du nuage de points.

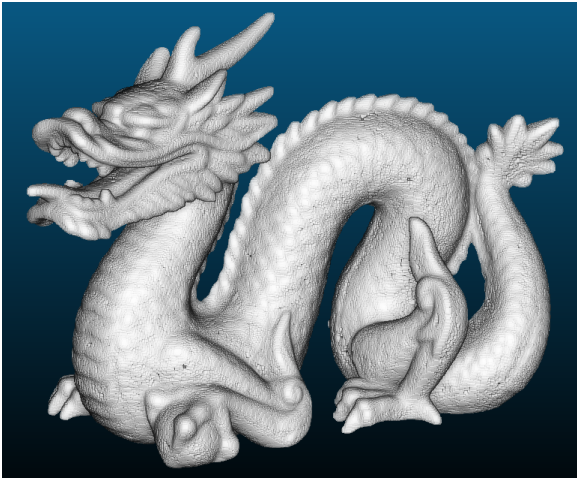


FIGURE 4 – Dragon de Stanford

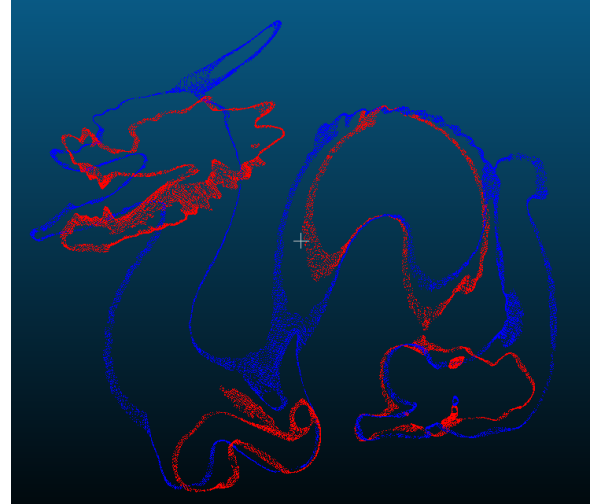


FIGURE 5 – Application de l'algorithme RANSAC sur le dragon

Paramètres utilisés :

| Paramètre    | Valeur |
|--------------|--------|
| nb_draws     | 100    |
| threshold_in | 0.001  |
| nb_planes    | 2      |

## Question 4

### Méthodologie :

Avec un RANSAC classique, de grandes surfaces horizontales (comme une table) risquent d'être détectées avant les murs verticaux, même si l'on souhaite en priorité segmenter ces murs. Pour éviter ce biais, nous avons introduit un **critère d'angle** : un point n'est considéré *inlier* que si

1. sa distance au plan est inférieure à un seuil (comme auparavant),
2. et l'angle entre la normale du point et la normale du plan détecté est suffisamment petit.

Cette approche permet de privilégier les parois verticales par rapport à d'autres surfaces parallèles non désirées.

Pour calculer la normale de chaque point, nous avons réutilisé deux fonctions du TP3 :

- `PCA`, qui calcule les valeurs et vecteurs propres de la matrice de covariance d'un ensemble de voisins,
- `compute_local_PCA`, qui applique PCA localement (sur un voisinage) afin d'estimer la normale de chaque point.

## Paramètres et résultats

| Paramètre           | Valeur |
|---------------------|--------|
| nb_draws            | 300    |
| dist_threshold      | 0.25   |
| angle_threshold_deg | 8.0    |
| nb_planes           | 5      |
| Temps d'exécution   | 220 s  |

TABLE 1 – Paramètres utilisés pour la RANSAC améliorée

Le code s'exécute en 220 secondes pour extraire les 5 premiers plans (figure 6). Les plans sont clairement segmentés et on constate que les murs verticaux sont mieux détectés qu'avec un RANSAC purement basé sur la distance.

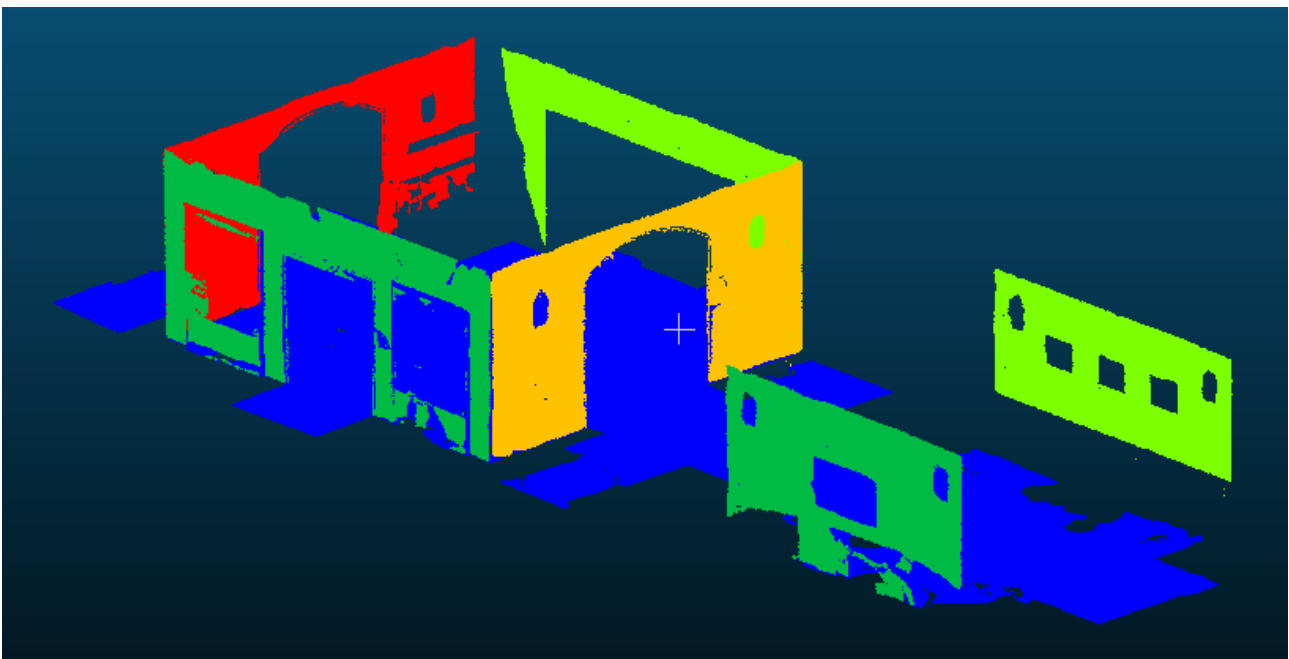


FIGURE 6 – RANSAC améliorée (les 5 premiers plans)

## 3 Going further (BONUS)

Pour accélérer l'algorithme RANSAC, une idée naturelle consiste à **réduire le nombre de points du nuage**. Nous avons donc mis en place un *sous-échantillonnage* aléatoire pour réduire le nombre de points. Une fois un plan détecté sur ce nuage réduit, nous effectuons un *raffinement* pour retrouver les inliers correspondants dans l'ensemble des points originaux.

## Paramètres et résultats

| Paramètre                | Valeur    |
|--------------------------|-----------|
| nb_draws                 | 300       |
| dist_threshold           | 0.25      |
| angle_threshold_deg      | 8.0       |
| nb_planes                | 5         |
| subsample_ratio          | 0.1 (10%) |
| <b>Temps d'exécution</b> | 35 s      |

Après de nombreux essais sur les paramètres de la fonction RANSAC, nous obtenons les résultats de la figure 7, avec la détection de 10 plans très cohérents dans le nuage de points et un temps d'exécution considérablement réduit ( $\approx 36$  s).

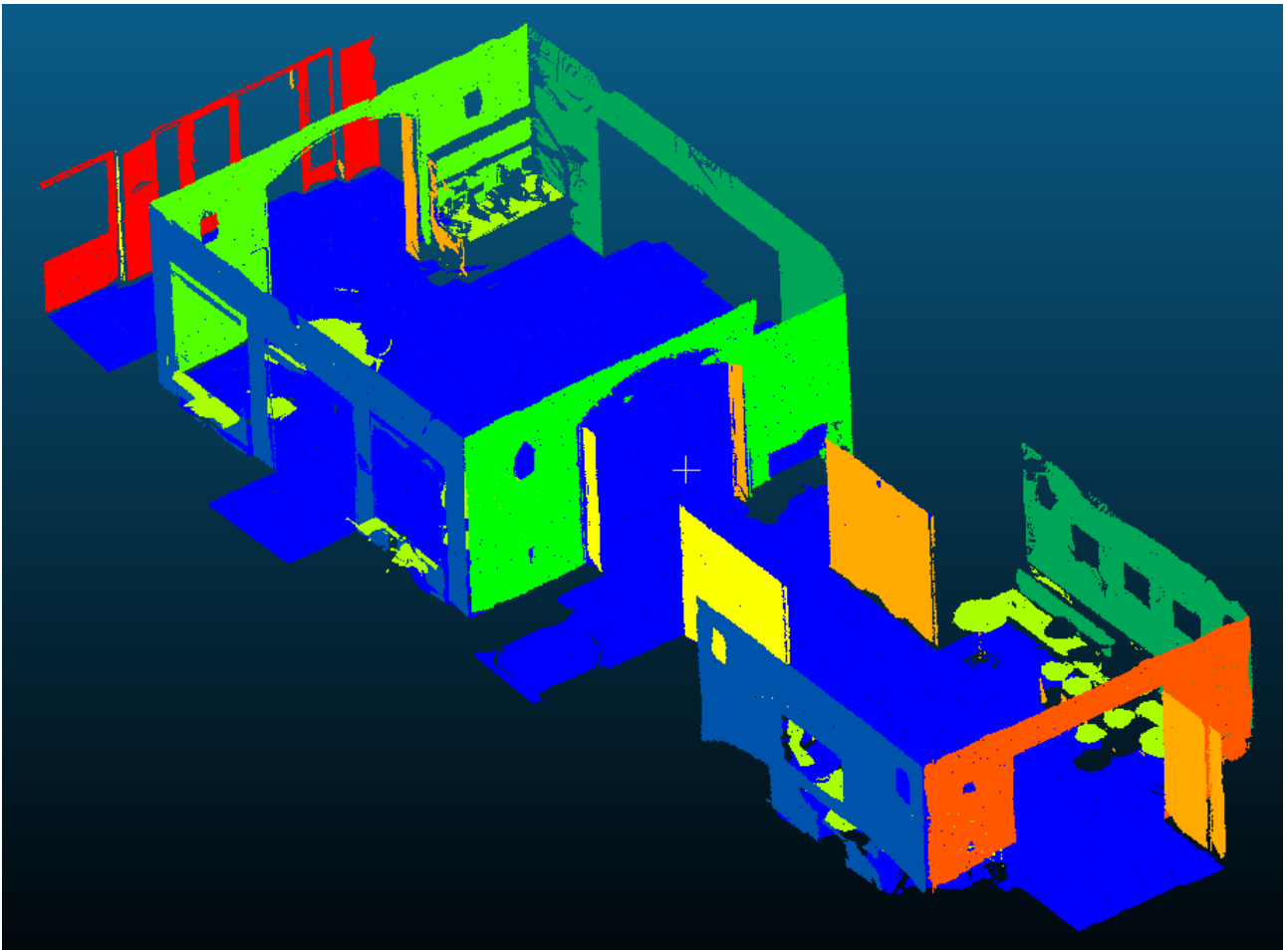


FIGURE 7 – Résultat de notre RANSAC "rapide" (10 plans détectés).

Et pour 15 plans :

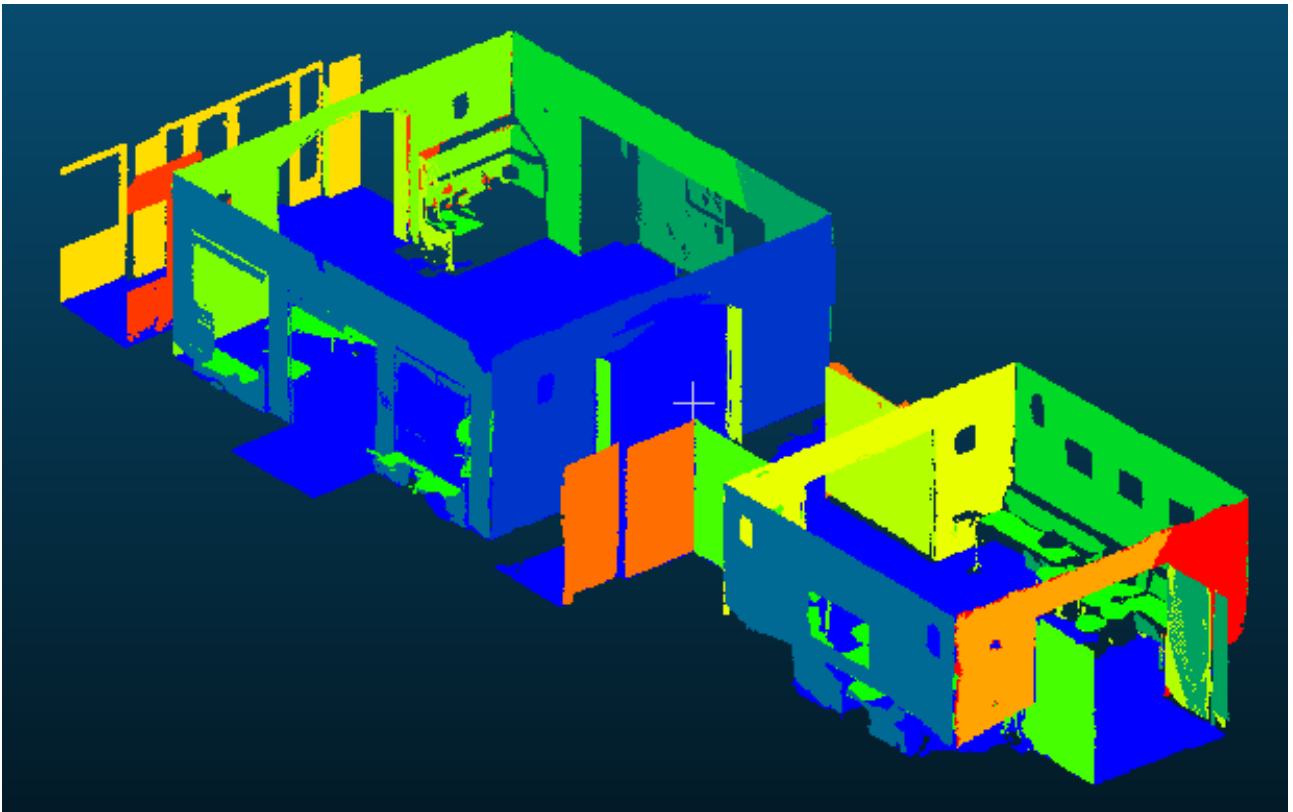


FIGURE 8 – Résultat de notre RANSAC "rapide" (15 plans détectés).