

TP3 - Nuages de Points et Modélisation 3D

Chloé Habasque - Colin Coërchon

29 Janvier 2025

1 CloudCompare ICP

1.1 Question 1

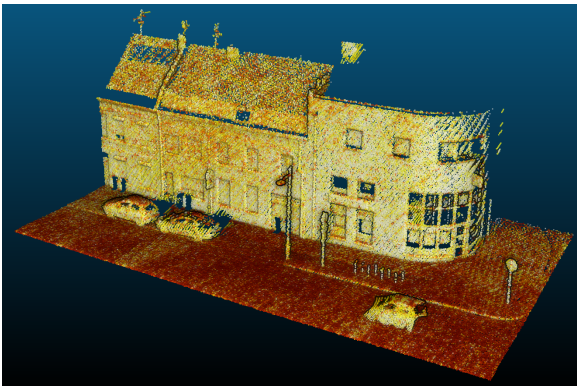


FIGURE 1 – radius = 0.01

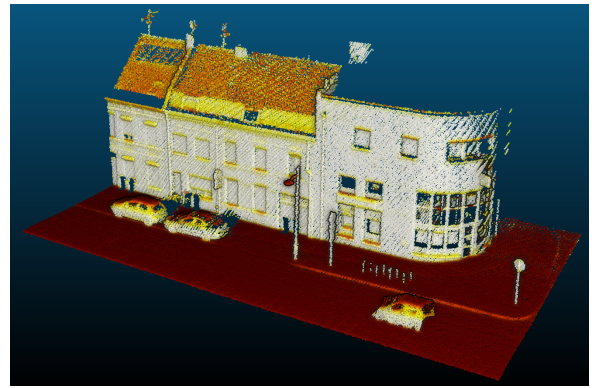


FIGURE 2 – radius = 0.2

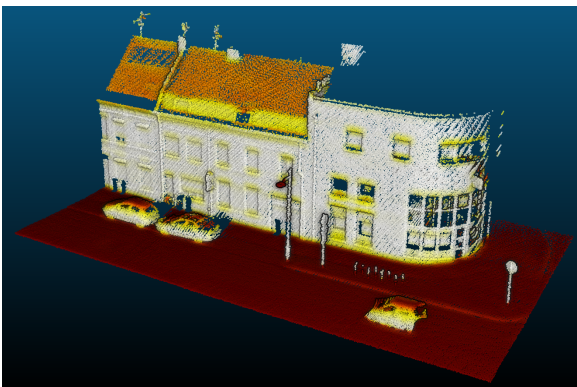


FIGURE 3 – radius = 0.5

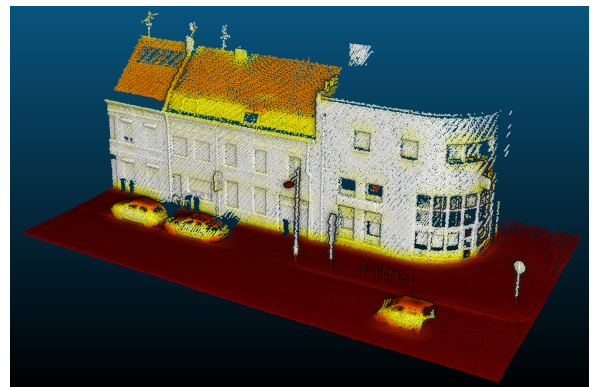


FIGURE 4 – radius = 1

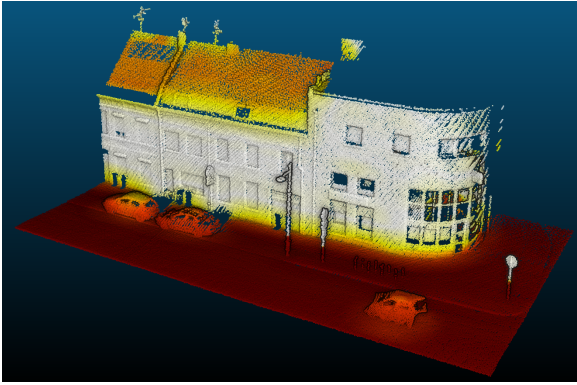


FIGURE 5 – radius = 2

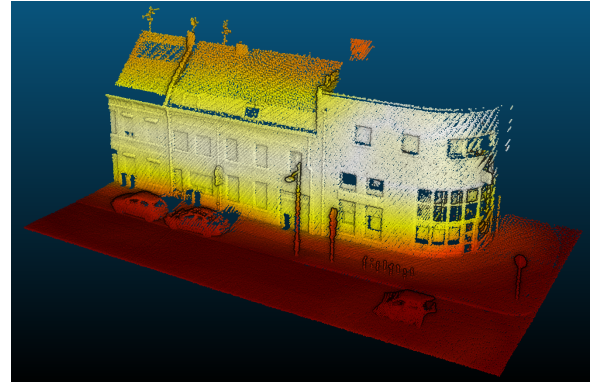


FIGURE 6 – radius = 5

Temps de calculs :

r (m)	0.01	0.2	0.5	1	2	5
Temps (s)	0.56	0.73	0.93	1.69	5.37	42.19

TABLE 1 – Temps de calcul en fonction du rayon r

- Lorsque le rayon est très petit, l'estimation des normales devient très locale et sensible au bruit. Cela entraîne des normales très variables, donnant un aspect irrégulier et bruité au nuage de points, voir [figure 1](#).
- Lorsque le rayon est trop grand, les normales sont lissées sur de grandes zones, ce qui efface les détails (bords, coins, petites structures), voir [figure 5](#) et [figure 6](#). Par exemple, on ne voit plus la structures des voitures.

1.2 Question 2

Un petit rayon donne des normales très détaillées mais bruitées. Un grand rayon lisse trop les normales et on y perd les détails.

On peut donc **choisir un rayon intermédiaire tel que 0.5**, qui permet d'éviter d'avoir trop de bruit tout en gardant les informations sur les détails géométriques de la structures de points, voir [figure 3](#).

2 Local PCA and normal computation in Python

ACP sur le nuage de points

En effectuant l'ACP sur le nuage de points `Lille_street_small.ply`, on retrouve bien les bonnes valeurs propres données dans l'énoncé :

$$[\lambda_1 \quad \lambda_2 \quad \lambda_3] = [89.58926189 \quad 21.78933003 \quad 5.25050514]$$

```

1 PS C:\Colin\Cours\MVA\S2\NPM3D\TP\TP3\code> python .\descriptors.py
2 [89.58926189 21.78933003 5.25050514]
```

Dans une ACP d'un nuage de points 3D décrivant une petite partie d'une surface, le vecteur propre associé à la valeur propre maximale (e_1) correspond à la direction où les points varient le plus. Et donc, en opposition, le vecteur propre associé à la plus petite valeur propre correspond à la direction où les points varient le moins.

C'est donc le **troisième vecteur propre** (e_3) qui représente la normale à la surface.

Changement de l'ordre des valeurs propres

Notre fonction PCA retournait les valeurs propres dans l'ordre décroissant. Il fallait donc choisir le **dernier vecteur propre** pour trouver le "vecteur normal" à chaque sous-ensemble de points.

Il faut donc modifier cette partie du code fournie :

```
1 | normals = all_eigenvectors[:, :, 2]
```

à la place de :

```
1 | normals = all_eigenvectors[:, :, 0]
```

2.1 Question 3

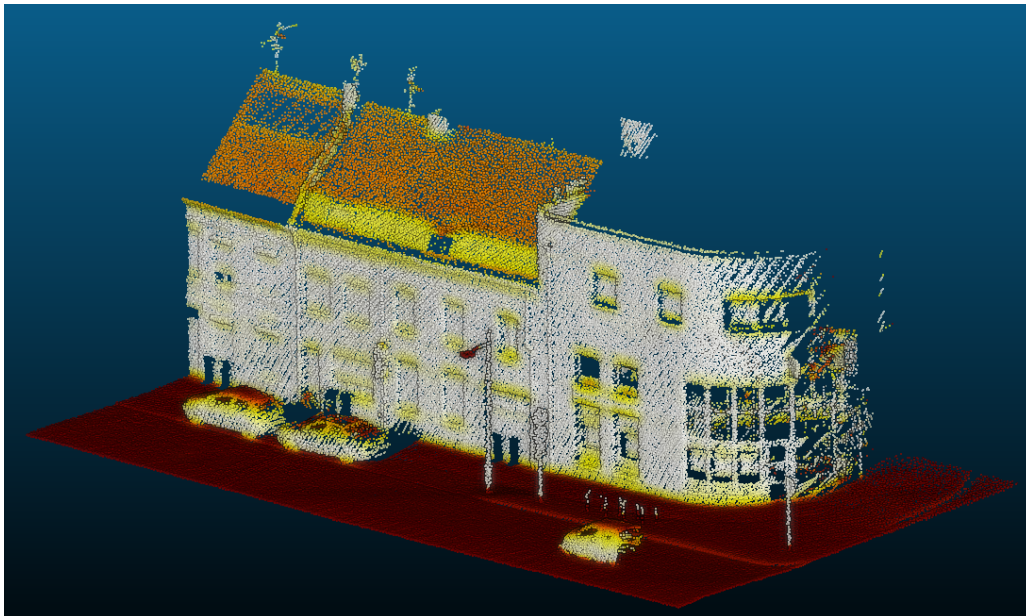


FIGURE 7 – Résultats de `compute_local_PCA` avec *KD Tree* et $\text{radius} = 0.5 \text{ m}$

Le résultat est visuellement **quasiment identique** à celui obtenu en calculant les normales directement par le logiciel *CloudCompare* (cf. [figure 3](#)).

2.2 Question 4

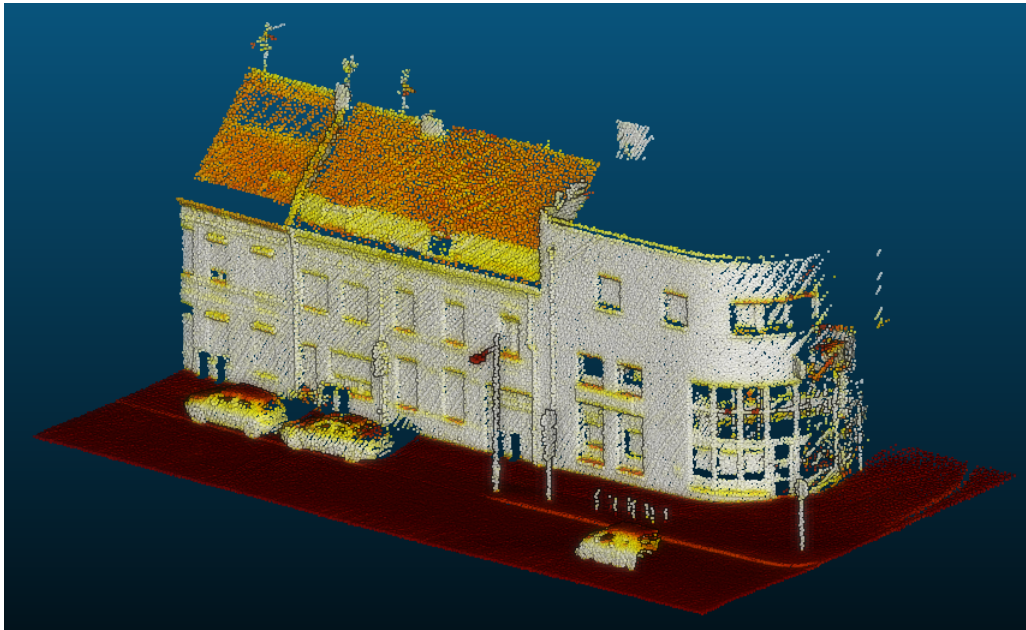


FIGURE 8 – Résultats de `compute_local_PCA` avec *KD Tree* et $k = 30$

Comparaison des deux méthodes

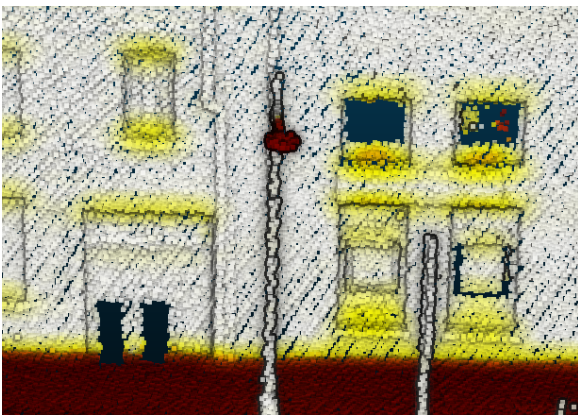


FIGURE 9 – Méthode radius - radius = 0.5

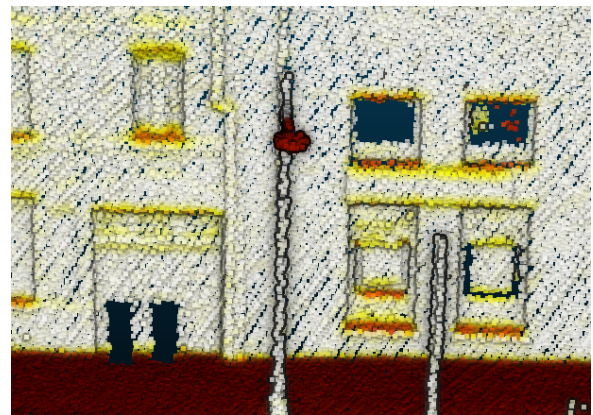


FIGURE 10 – Méthode K-NN - $k = 30$

FIGURE 11 – Comparaison des deux méthodes

La première méthode consiste à sélectionner les plus proches voisins selon un rayon donné, tandis que la deuxième prend les k plus proches voisins. Les résultats des deux méthodes sont très similaires, mais il y a quand même quelques petites différences.

En effet, comme le montre la [figure 10](#), **la méthode K-NN permet de mieux préserver les détails du nuage de points**. Cette différence est particulièrement visible au niveau des bords des fenêtres et des portes, comme le montrent les [figure 9](#) et [figure 10](#).

3 Going further (BONUS)

3.1 Question bonus

Verticalité

$$V = \frac{2 \arcsin |\langle \vec{n}, \vec{e}_z \rangle|}{\pi}$$

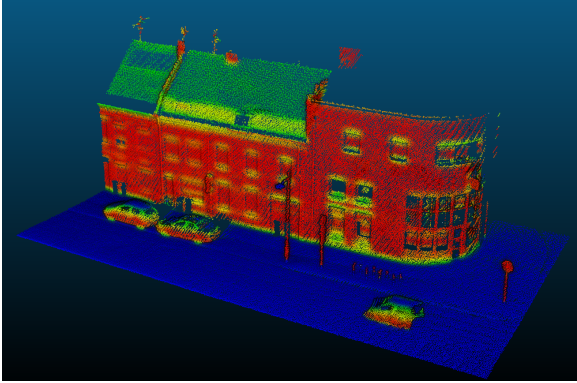


FIGURE 12 – CloudCompare

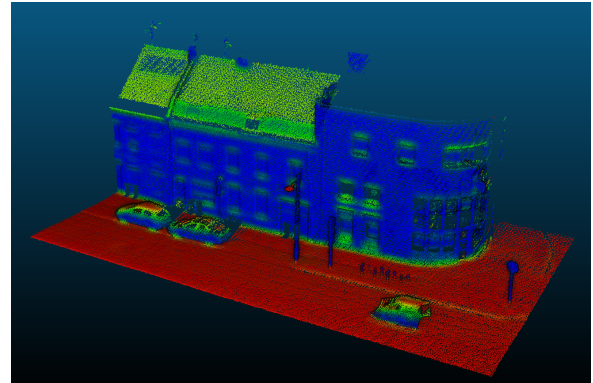


FIGURE 13 – Code

FIGURE 14 – Comparaison de la Verticalité avec rayon= 0.5

Notre formule semble donner des résultats cohérents, mais ne correspond pas aux couleurs données lors du calcul de la verticalité par le logiciel CloudCompare. Ce sont **les mêmes zones détectées**, mais avec des couleurs différentes.

Linéarité

$$L = \frac{\lambda_1 - \lambda_2}{\lambda_1}$$

Si $\lambda_1 \gg \lambda_2, \lambda_3$, alors les points sont alignés selon une seule direction dominante. $L \approx 1$ indique donc une forte linéarité.

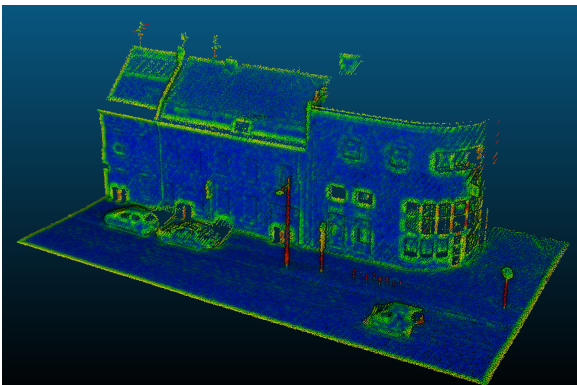


FIGURE 15 – CloudCompare

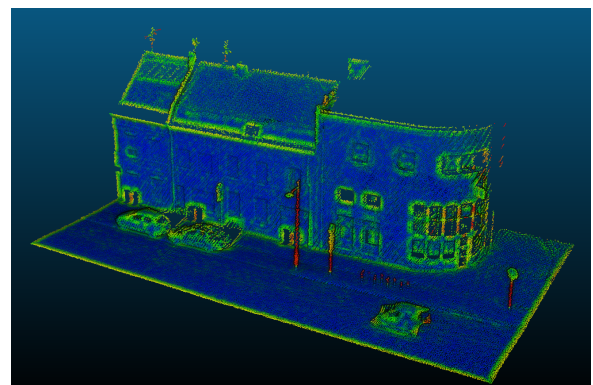


FIGURE 16 – Code

FIGURE 17 – Comparaison de la Linéarité avec rayon= 0.5

Planéité

$$P = \frac{\lambda_2 - \lambda_3}{\lambda_1}$$

Si $\lambda_1 \approx \lambda_2 \gg \lambda_3$, les points forment une surface plane. $P \approx 1$ indique donc une planéité élevée.

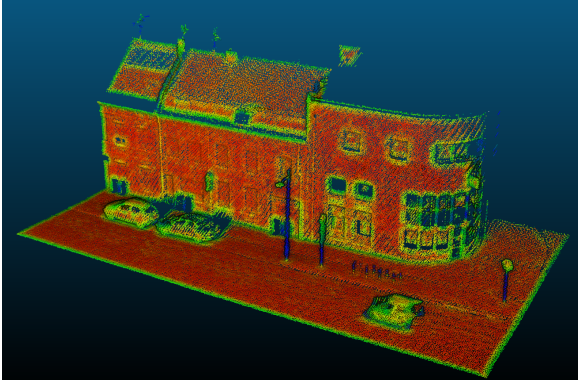


FIGURE 18 – CloudCompare

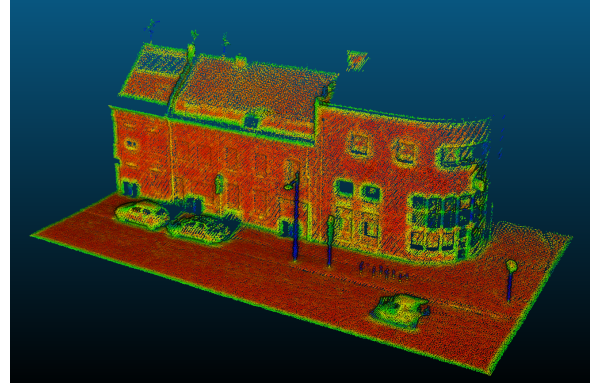


FIGURE 19 – Code

FIGURE 20 – Comparaison de la Planéité avec rayon= 0.5

Sphéricité

$$S = \frac{\lambda_3}{\lambda_1}$$

Si $\lambda_1 \approx \lambda_2 \approx \lambda_3$, la distribution des points est uniforme dans l'espace, formant donc une structure sphérique. $S \approx 1$ indique donc une sphéricité élevée.

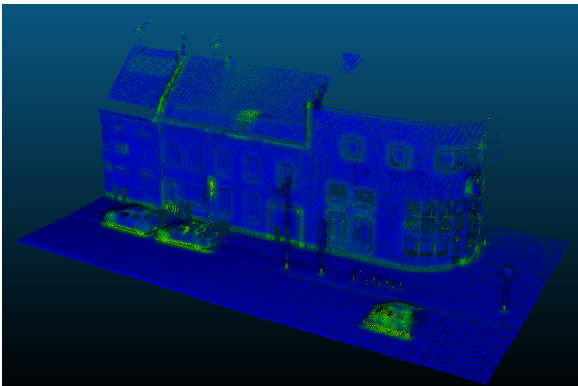


FIGURE 21 – CloudCompare

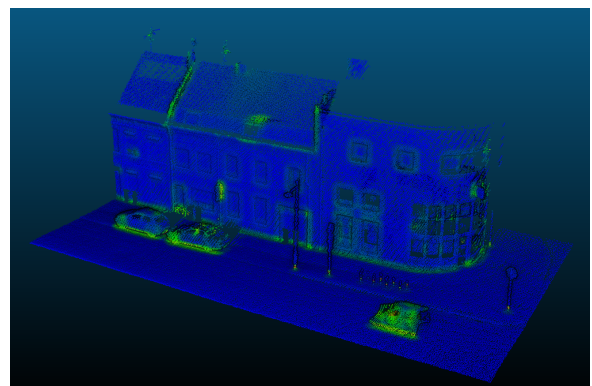


FIGURE 22 – Code

FIGURE 23 – Comparaison de la Sphéricité avec rayon = 0.5