TP3: Neighborhood descriptors

Louis Martinez
louis.martinez@telecom-paris.fr

Mohamed Ali Srir
mohamed.srir@telecom-paris.fr

Exercise A: Normals in CloudCompare

Question 1:

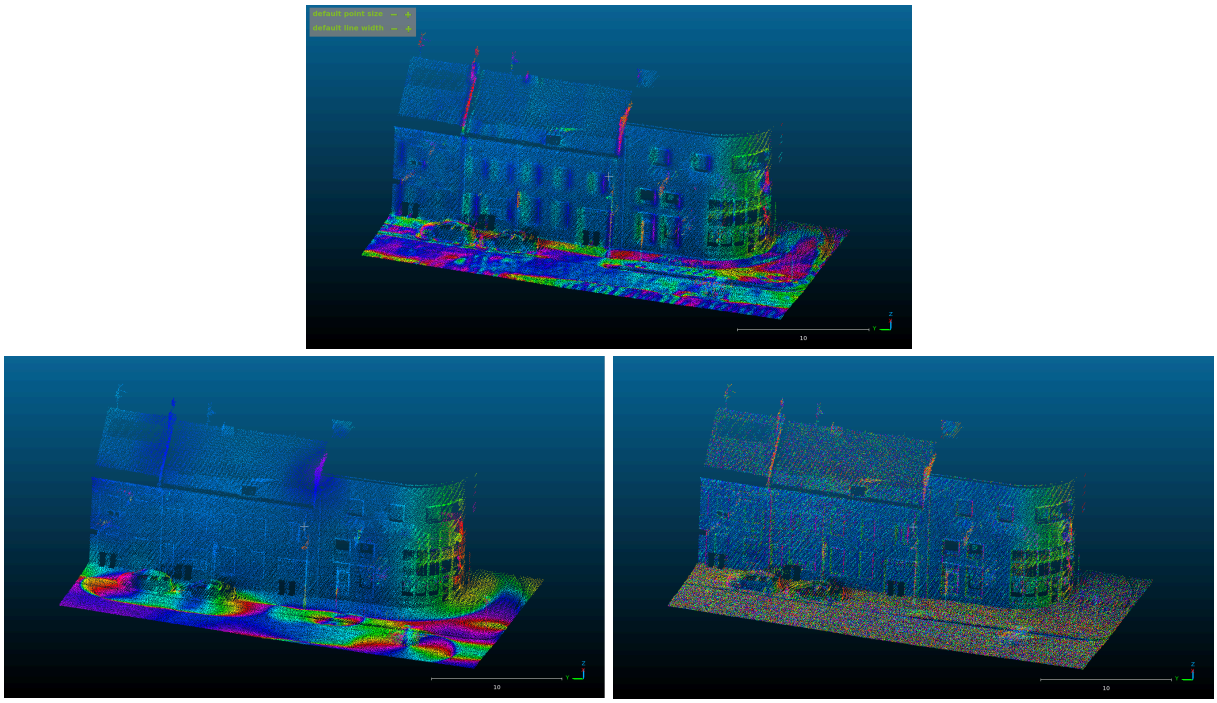

Figure 1: **Point clouds with DIP scalar field.** Normals were computed with a radius of 0.5m for the top image, 3m for the bottom left one, and 0.1m on the bottom right.

If the radius is too large (bottom left), normals are averaged over a region with many small details (small with respect to the overall size of the point cloud). In contrast, a too small radius (bottom right) provides a noisy map. This effect is due to the fact that points belonging to the same surface are not perfectly coplanar. The default comes from the precision limits of the scanner used to acquire the data.

Question 2:

Provided the aforementioned remarks, the neighborhood must be large enough to annihilate noise due to the acquisition, but narrow enough to reflect some small meaningful structures of the point cloud. A good heuristic may be for the radius to be half (because it's a radius) of the characteristic distance of the smallest structure of interest in the point-cloud.

Exercise B: Local PCA and normal computation in Python

Question 3:

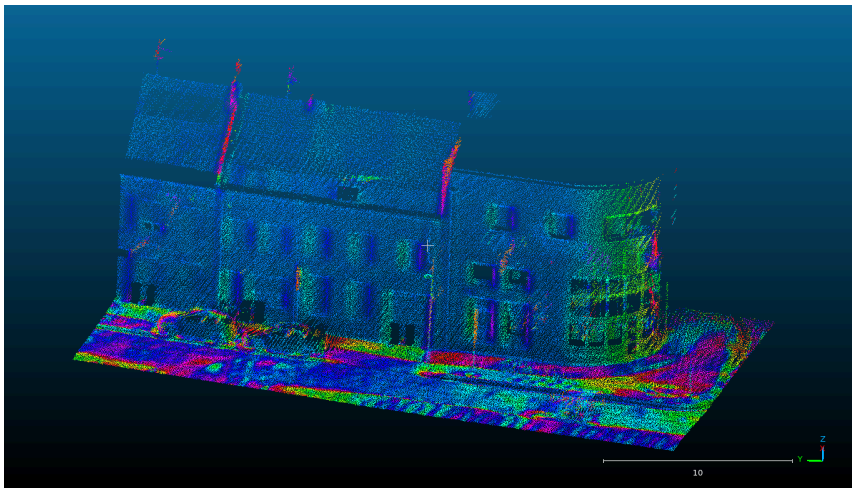

Figure 2: **DIP field computed with a radius of 0.5m.**

Question 4:

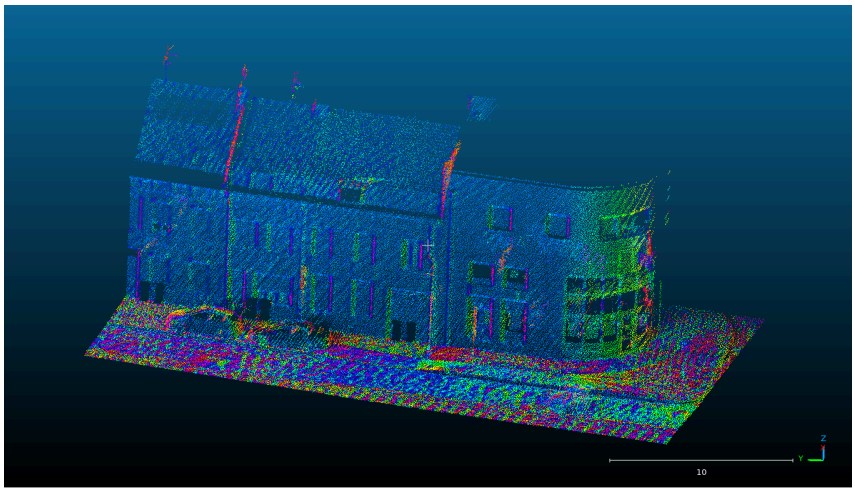

Figure 3: **DIP field computed with KNN and 30 neighbors.**

The scalar field obtained with KNN is less consistent than what we obtained with a fixed radius. Flat areas have varying colors. This highlights variations of density within the point cloud. Given one point, regarding how far its 30 nearest neighbors are, the resulting normal will be more or less sensitive to local noise (cf. Question 2). However, using KNN is computationally more efficient since using a fixed number of points allows easy parallelization of the process. Although there are some workarounds to parallelize the radius approach, it's harder to setup.

Exercise C: Going further (BONUS)

Bonus Question:

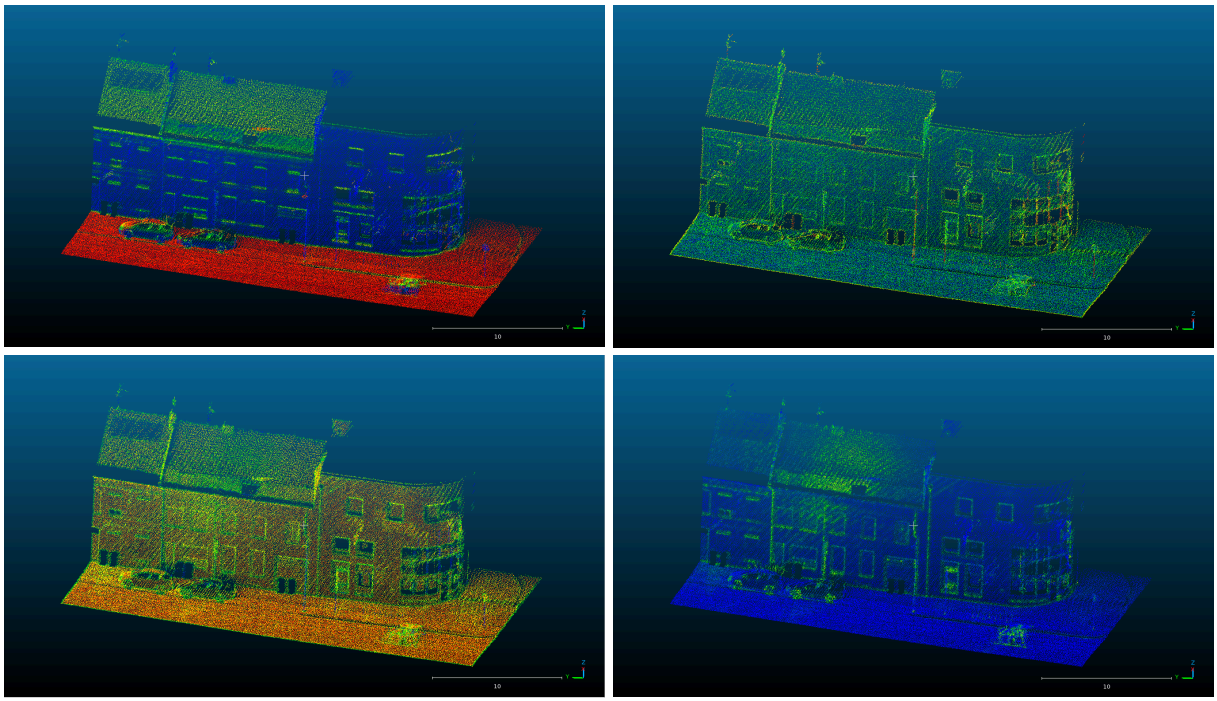

Figure 4: Verticality (top-left), linearity (top-right), planarity (bottom-left), sphericity (bottom-right) based on PCA computed by using the 30 nearest neighbors

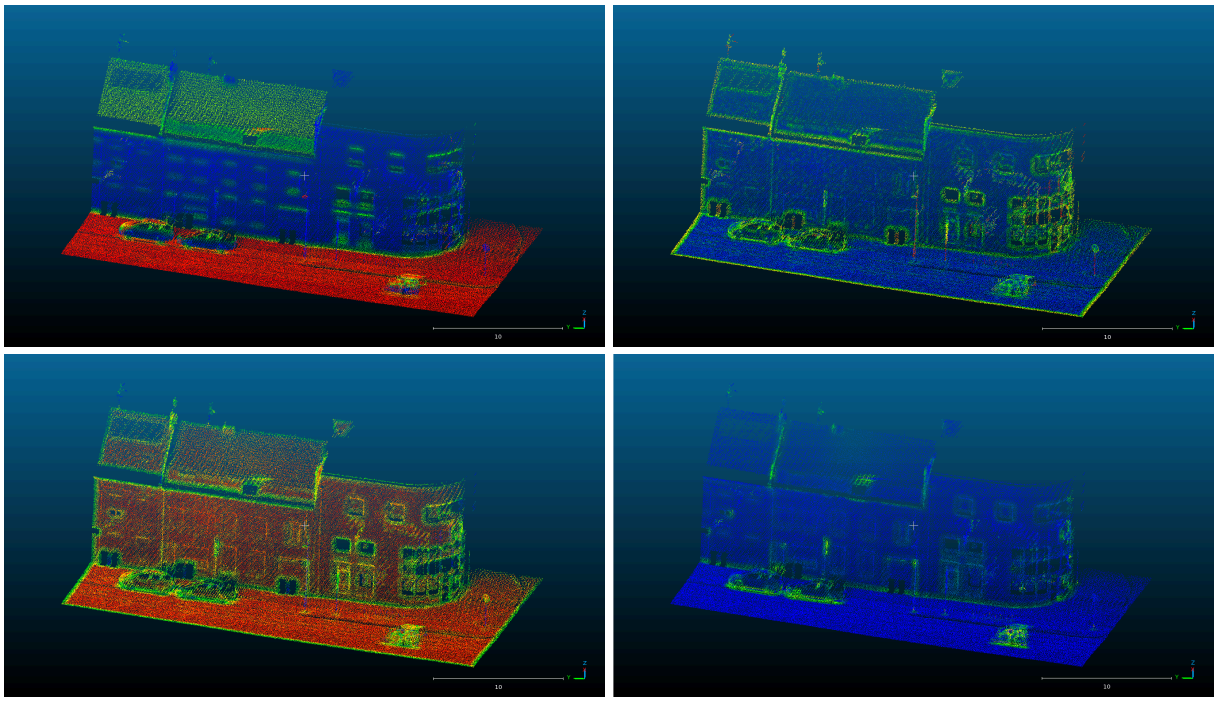

Figure 5: Verticality (top-left), linearity (top-right), planarity (bottom-left), sphericity (bottom-right) based on PCA computed by using a spherical neighborhood of radius 0.5

- Verticality : Verticality quantifies how well a point's local neighborhood aligns with the vertical direction, typically defined relative to the z-axis. Vertical walls exhibit values near 0 (blue), indicating their normals are perpendicular to z. In contrast, streets have values close to 1 (red), showing their normals are parallel to z, while roofs have intermediate values reflecting a mixed alignment.

- Linearity : Linearity measures how closely the local point distribution resembles a line, emphasizing linear features such as edges or borders (e.g., window frames). This is illustrated more clearly in Fig. 7.
- Planarity: Planarity measures how closely the local point distribution resembles a plane. Streets, roofs, and walls exhibit high planarity values (red, close to 1), which increase near edges.
- Sphericity: Sphericity indicates how much the local point distribution resembles a sphere. It is prominent around the car's wheels, where the point distribution is more spherical.