# TP5: Modeling

Louis Martinez
louis.martinez@telecom-paris.fr

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

## Exercise A: RANSAC Shape detection in CloudCompare

**Question 1:**

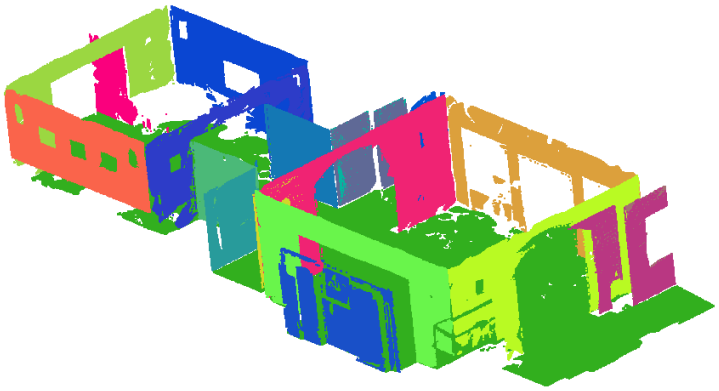

Figure 1: First 20 planes extracted with RANSAC in CloudCompare

Figure 1 was generated using the following set of parameters:

- Min support points per primitive: 1000
- Max distance to primitive: 0.101 (default)
- Sampling resolution: 0.201 (default)
- Max normal deviation: 3°
- Overlooking probability: 0.01 (default)
- No shape simplification

The procedure outputted 105 planes in total. For the sake of explainability though, we only keep the 20 first planes found by RANSAC. Most parameters don't have a a significant impact on the final result, unless we change their order of magnitude. However, the following parameters seem to have an important influence on the final number of planes:

- A higher number of points per primitive tends to encourages wider planes. Therefore the first extracted planes are the main ones of the cloud.
- A lower normal deviation enforces a better alignment between the normal of a candidate plane and the normal of a point.
- If the maximum allowed distance between a plane and a point, we may end up merging two parallel walls if they're too close to each other.

## Exercise B: RANSAC in Python

**Question 2:**

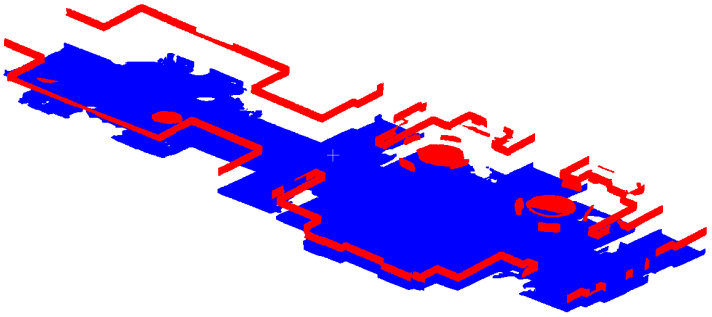

Figure 2: Two planes extracted consecutively by RANSAC

The second iteration of RANSAC didn't extract a surface. We would have expected to extract one of the walls of the scene. This results is linked to the fact that we count points within a margin centered on the candidate hyperplane. The extracted slice contains more points than any wall of the point cloud. To improve the procedure, we must enforce the inliers (wrt the candidate plane) to have their normal parallel with the one of the candidate plane.

**Question 3:**

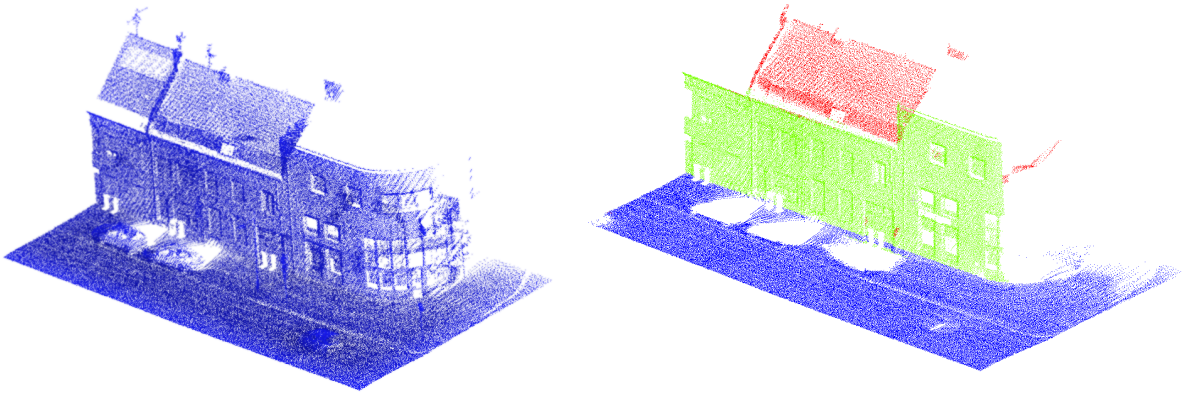

Figure 3: Source Lille Street point cloud (left) and three planes extracted consecutively by RANSAC (right)

We used `RANSAC_recursive()` with `nb_draws=300`, `threshold_in=0.20` and `nb_planes=3`. Results are satisfactory since the segmentation is neat and meaningful. Without needing additional information on the orientation of the normals we retrieved the main planar structure of the original point cloud.

**Question 4:**

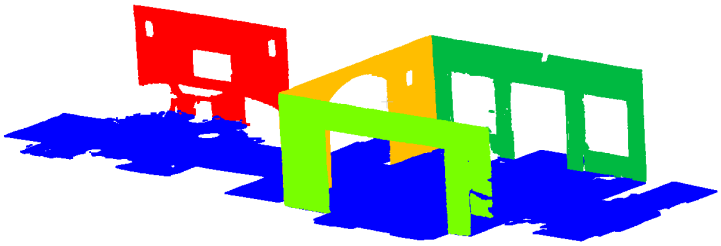

Figure 4: Five planes extracted consecutively by RANSAC with normals alignment

To ensure we actually extract hyperplanes, we constrain the normals of the points lying in the slice around a candidate hyperplane. To be considered as belonging to an hyperplane the normal of a given point must be aligned the normal of the hyperplane: $\left|\vec{n}_{\text{point}} \cdot \vec{n}_{\text{plane}}\right| < 1 - \varepsilon$ where $\varepsilon$ is a small margin, since the alignment may not be perfect. Figure 4 was obtained with `nb_draws=500`, `threshold_in=0.1`, `nb_planes=5` and $\varepsilon = 0.1$ (`normals_threshold` in the code). Normals were computed using local PCA with KNN ($k = 20$, `neigh` in the code). We notice a significant improvement compared to the initial version. The floor as well as some walls are well extracted.

## Exercise C: Going further (BONUS)

**Bonus Question:**

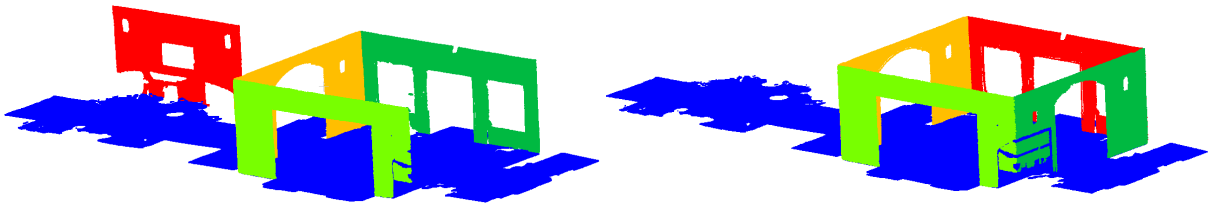

Figure 5: Five planes extracted consecutively by RANSAC (left : 291s) with 500 iterations and its efficient counterpart (right : $8_{\pm 0.4}$s) with draws

We propose two improvements of the vanilla RANSAC procedure:

- Starting from the observation that if a points belongs to a plane, so do most of its neighbors. Therefore, instead of sampling 3 points randomly across the whole point cloud, we randomly choose a first point and sample the two other ones among its 30 nearest neighbors. On average, this method requires less draws to determine an appropriate plane. This alternate version is further denoted by *N-RANSAC (Neighbor-RANSAC)*.
- To avoid running useless draws after determining an appropriate plane, we added early stopping based on the proportion of points lying in a candidate plane. In practice, we stop iterating draws when 5% of the points lye in this plane. This proportion may vary from one use case to another.

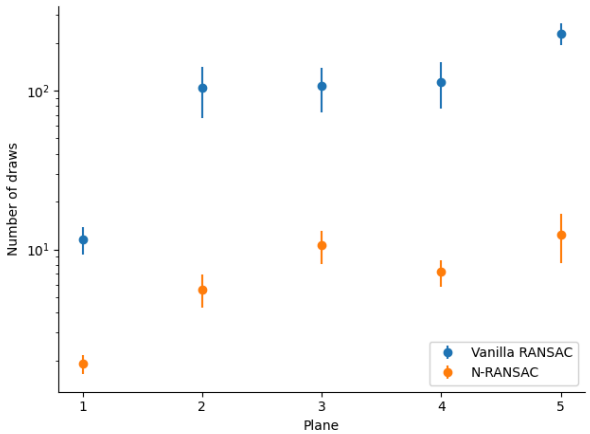


Figure 6: **Influence of early-stopping (semi log-scale).** With early stopping at 5% enabled, RANSAC runs more draws to find a plane satisfying the criterion than *N-RANSAC*. Additionally, the std error is overall more important on RANSAC than on *N_RANSAC*.

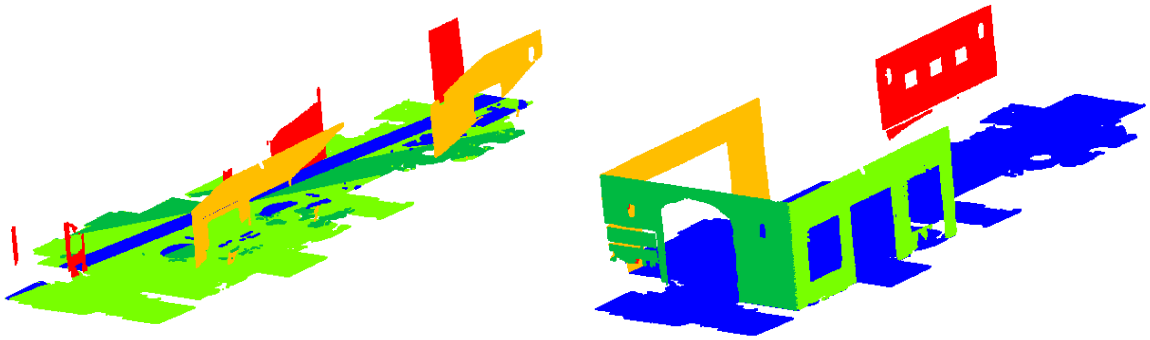

Figure 7: **Detected planes with RANSAC (left) and *N-RANSAC* (right).** For both methods, we extracted 5 planes and kept the early stopping criterion at 5%. RANSAC performs extremely bad with this criterion. Conversely, the neighboring condition brought by *N-RANSAC* allows an accurate segmentation while running 10 times faster on average (Figure 6).