Balthazar Neveu — Jamy Lafenetre
balthazarneveu@gmail.com —
jamy.lafenetre@ens-paris-saclay.fr

Question 1

The best segmentation we got was using a minimum number of points of 2000 per plane, a minimum distance of 0.1, a sampling resolution of 0.2, a maximum normal deviation of 10 degrees, and an overlooking probability of 0.01. The resulting segmentation was composed of 56 planes, as can be seen in Figure 1.

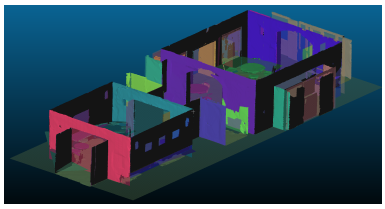

Figure 1: Result of the CloudCompare segmentation.

Question 2

This result is undesirable because the second plane is not a good representaion of the point cloud. It does not represent any underlying plane structure, but still managed to get the maximum ammount of votes by simply cutting the vertical axis. This highlights that a vote based on the distance criterion alone is not sufficient, especially when the samples are distributed radially. This can be seen on Figure 2.

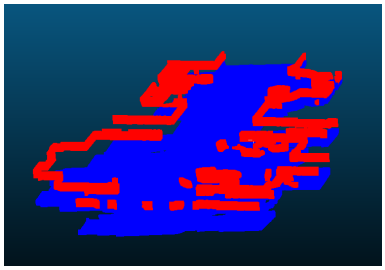

Figure 2: The distance criterion alone is not sufficient for the indoor scene.

Question 3

We evaluate the same algorithm on ”Lille_street” from previous assignments. We use 2 planes, a distance threshold of 0.1 and 200 draws. The segmentation gives better results on this scene because the point distribution is not radially distributed as in the indoor point cloud. This can be seen in Figure 3.

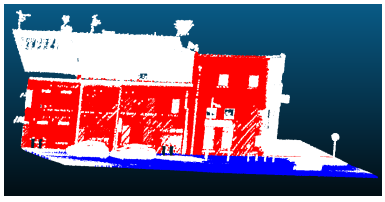

Figure 3: For Lille_street, the distance criterion is more satisfactory (for 2 planes at least).

Question 4

To avoid the previous pitfalls, we can enrich the voting criterion by estimating the normals of the point cloud using local PCA. Instead of proposing a plane by picking 3 random samples, we can instead pick a single sample and its associated normal. Then, candidate points will only vote if they satisfy the distance criterion, as well as a criterion measuring the angle difference between their normal and the proposed plane normal. There are two major improvements with this strategy:

- First, the candidate planes are far superior to what we had before because they are locally optimal. RANSAC should therefore converge in fewer iterations.
- Second, points can only be considered as inliers of a plane if the local geometry resembles the plane. This will strongly reject planes that intersect every structure.

By doing so, the segmentation is considerably better, as can be seen in Figure 4.

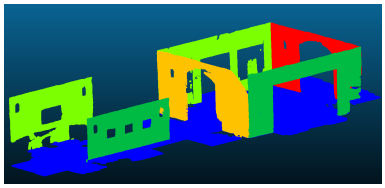

Figure 4: Segmentation with 5 planes, using a distance threshold of 0.2, an angle threshold of 10 degrees, and 500 draws. Adding a criterion on the angle between the normals and the plane significantly improves the segmentation.

Bonus

Our RANSAC implementation is already quite fast and takes around 10 seconds to run on GPU. This is because we treat every candidate plane in parallel instead of treating them one by one. Let's pretend in what follows that we do it in series (in a "for" loop).
A first source of optimization is to drop the fixed number of iterations N_{iter} and replace it with a fixed probability $1-\beta$ of convergence. We will instead dynamically update N_{iter} based on the best vote found so far. We call "inliers" points that vote positively for a candidate plane. By noting m the (unknown) exact total amount of inliers, the objective is to iterate until the total probability of picking the right plane among the N is more than $1-\beta$. In other words:

$$(1-\frac{m}{N})^{N_{iter}} \leq \beta$$

Which can be written as

$$N_{iter} \geq \frac{log(\beta)}{log(1-m/N)}$$

The trick is to notice that, although m is unknown, the best estimated number of inliers so far is a lower bound of it. We can therefore dynamically reduce N_{iter} as we iterate. A second (smaller) source of optimization is to only check the angle threshold condition if the distance threshold condition is satisfied.
These tricks are, however, mainly designed for CPU computing as they cannot use the SIMD computation power brought by GPUs. To further use the available hardware, we compute B regular RANSAC iterations at the same time, therefore simultaneously evaluating multiple planes. We empirically found that, for this specific point cloud and for the specific case of segmentation by planes, $B=5$ was optimal to minimize the overhead. With $\beta=0.01$ and $B=5$, we managed to make a 5-plane segmentation in 0.92 seconds. The quality result with and without the optimization trick is nearly identical, as can be seen in Figure 6, but the speedup is considerable, as can be seen in Figure 5.

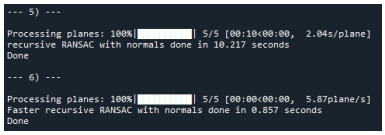

Figure 5: Execution time of the original and improved Ransac.

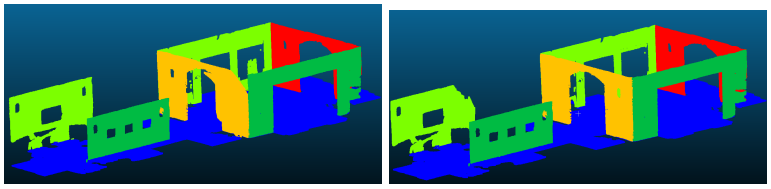

Figure 6: Segmentation with 5 planes, using a distance threshold 0.2, an angle threshold of 10 degrees and 500 draws. Left: using the original method. Right: using the improved method.

Since one of the inlier criterions involves the distance to the plane, a strategy could be to perform a KDTree radius query at regular positions on the candidate plane. This would enable to consider only points which already verify the distance condition, and may significantly speed up computation and save memory. However, our available implementation of the KDTree is CPU limited and too slow to speed-up anything. Better data structure may be preferable.