Mines Paris — PSL
Intelligence Artificielle, Systèmes et Données (IASD)
Nuages de Points et Modélisation 3D
Students: Alexandros Kouvatseas & Luka Lafaye de Micheaux

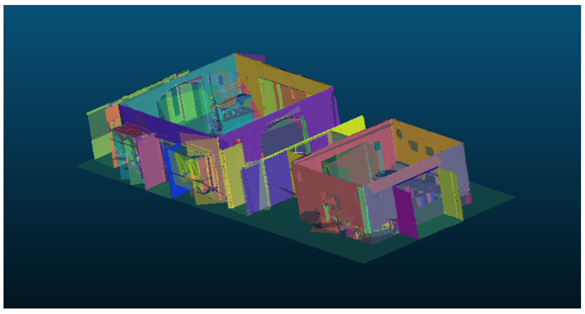
Question 1: Show a screenshot of your "best" segmentation. How many planes did you get? Give the parameters you used in CloudCompare. Comment the result.

	Basic	**Tolerant**	**Strict**
Min Support points	500	300	700
Max Distance	0.101	0.101	0.101
Sampling Resolution	0.201	0.201	0.201
Max Normal Deviation	25.00°	30.00°	20.00°
Overlooking Probability	0.01	0.05	0.005
Planes	193	224	163

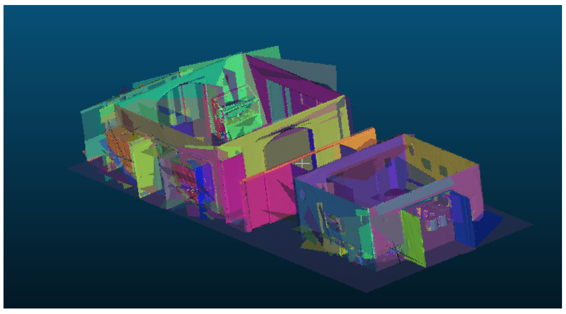
Table 1: Parameters of the different approaches.

In the beginning, we followed the balanced approach, as shown in *Table 1* and *Figure 1a*, where we observed 193 planes. From the figure, we can see that the segmentation closely matches the true boundaries of the objects and the walls; however, some inconsistencies remain. By implementing a more tolerant approach with larger sampling, using the values from the second column, we observe in *Figure 1b* that many errors arise, and the segmentation fails to correctly delineate the boundaries of the plot.

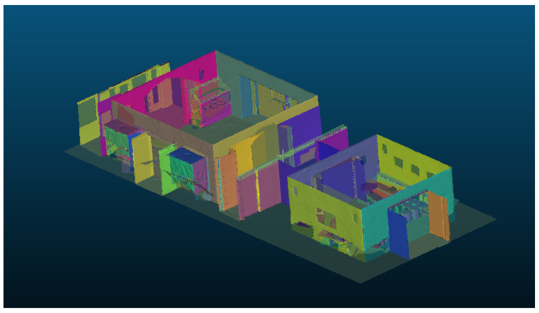
Finally, when applying a stricter detection method with finer sampling, using the values from the third column, we achieve the best performance, as seen in *Figure 1c*. This approach results in 163 planes, which is fewer than both the balanced and the more tolerant approaches, yet it better adapts to the characteristics of the plot. This makes it the most effective approach.



(a)



(b)



(c)

Figure 1: Comparison of the result of each approach on the Segmentation on the plot. (a): The balanced approach, (b): the tolerant one and (c): the strict one.

Question 2: Show a screenshot with the 2 planes you extracted. Explain why this is not the results you would like. What produces this behaviour?

By observing the screenshot in *Figure 2*, we can see the two planes extracted using the RANSAC implementation. While it's the result we expected, it's not the desired one. Firstly, the extracted planes do not completely cover the expected planar surfaces of the point cloud, as there are also various missing segments or fragmented planes, which suggests that the detection is incomplete.

This behavior could be produced by the sampling randomness. RANSAC randomly selects points to define a plane, and if an unlucky sample is chosen, the detected plane may not be representative. Also, some noise of the point cloud could lead some points close to the planes to not be classified as inliers. Finally, some of the planes of hte point cloud are not perfectly flat, leading RANSAC to struggle to fit a single plane to them.

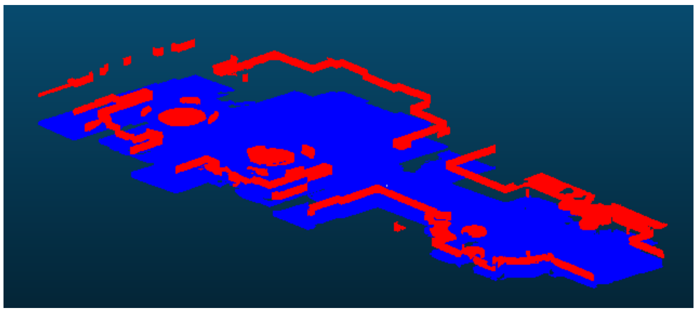


Figure 2: The planes that were extracted consecutively by RANSAC

Question 3: Apply the RANSAC plane detection on another point cloud (you can use one from previous TP or on Internet) and show a screenshot of the results. Give the parameters you used. Comment the result.

This time, we applied RANSAC with the same parameters (a threshold of 0.3 and a number of draws of 100) to the point cloud from TP3, `Lille_street_small.ply`, which represents a street in Lille. By examining *Figure 3*, we observe that the performance of RANSAC is significantly better. The planes are continuous, covering the full planar surfaces of the point cloud. This improvement can be attributed to the fact that the scene is simpler, with well-defined planar regions and less noise, making segmentation more effective.

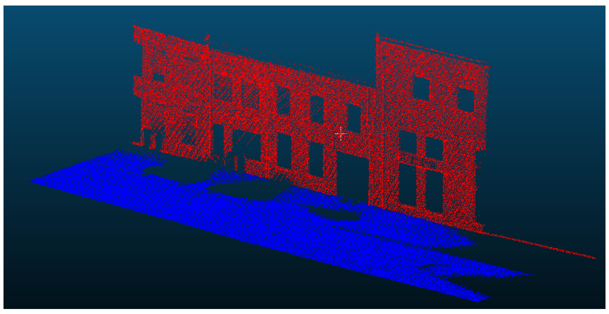


Figure 3: The planes that were extracted by RANSAC using the `Lille_street_small.ply` plot

Question 4: Explain your method. Show a screenshot with your "best" segmentation of 5 planes of the point cloud "indoor_scan.ply". Give the parameters you used. Comment the result.

Firstly, using the PCA function that we implemented in TP3 , we computed the local point normals. These normals helped to ensure that points belonged to a plane share a similar orientation. We, then, verified the alignment of the point normals with the detected plane normal, using a threshold to control the deviation.

We then begin the architecture of RANSAC, where the plane selection process now prioritizes planes where inlier points also have aligned normal vectors. Finally, with the implementation of Recursive RANSAC, the best plane is iteratively extracted and removed from the point cloud. The process is repeated until all 5 planes are detected or there are not enough remaining points.

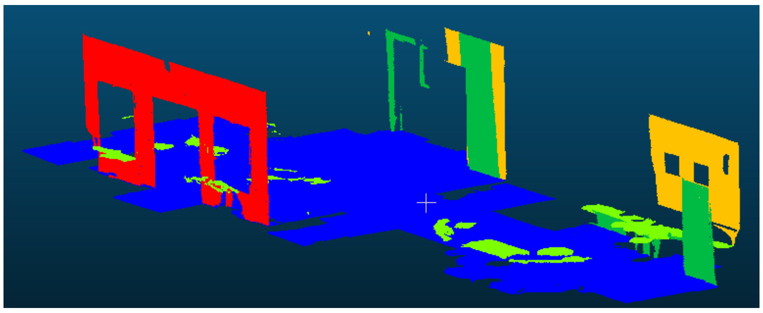


Figure 4: The 5 planes that were extracted consecutively by RANSAC using normals

By observing *Figure 4* we can see the segmentation, portraying the five detected planes. The detected planes (colored regions) roughly correspond to walls, floors, and ceilings, confirming that the segmentation works. However, some planes are incomplete. For example, some sections of walls and floors appear fragmented. In addition, certain regions, particularly green and red, seem misclassified or contain artifacts where surfaces should be continuous. This might be due to strict normal constraints, causing the algorithm to reject points that should belong to the same plane.

In other words, the algorithm successfully identifies multiple planes, capturing key structural elements, and normal-based filtering improves the accuracy by avoiding misclassifiation of non-planar regions, but it leaves many points unclassified, and the fragmentation of planes suggests that the algorithm may need some optimization yet.

Parameter	Value
Number of RANSAC draws (*nb_draws*)	100
Inlier distance threshold (*threshold_in*)	0.10
Normal similarity threshold (*normal_thresh*)	0.9
Number of planes (*nb_planes*)	5
Normal estimation radius	0.5

Table 2: RANSAC with normals Parameters

Question Bonus: Imagine a method to speed up RANSAC. Explain your algorithm. Show modeling results with and without acceleration. Display processing times without and with acceleration.

We speed up RANSAC by batching all random draws for a plane in a single parallel block. First, we pregenerate the random triplets of point indices rather than sampling each time inside the parallel tasks. Then, for each triplet, we fit a plane and count inliers in parallel using all CPU cores. This reduces the overhead of spawning many small parallel jobs and transferring data. After identifying the best plane, we remove its inliers and repeat the process on the remaining points to detect multiple planes. Because the core plane-fitting and inlier-counting logic is unchanged, we get the same results while significantly reducing processing time.

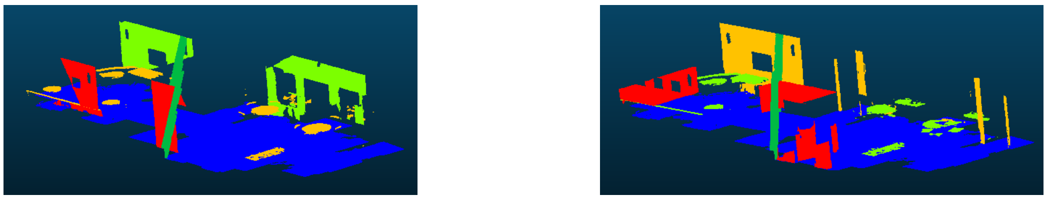
Parameter	Value
Number of RANSAC draws (*nb_draws*)	100
Inlier distance threshold (*threshold_in*)	0.15
Normal similarity threshold (*normal_thresh*)	0.6
Number of planes (*nb_planes*)	5

Table 3: Algorithm parameters for accelerated RANSAC

Method	Computation Time (seconds)
Standard RANSAC	17.665
Accelerated RANSAC	10.776
Acceleration Speedup	**1.64x**

Table 4: Comparison of computation time between standard and accelerated RANSAC

As we observe the result of the segmentation in *Figure 5*, we can see that the original algorithm produces planes that appear more continuous and well-defined, as larger surfaces are extracted more cleanly, though not in every case. On the other hand, the accelerated version yields more fragmented results, with thinner structures being segmented separately. There is also a higher presence of vertical structures that might not correspond to actual planes, and some surfaces appear noisier or less coherent. In conclusion, the normal RANSAC provides a cleaner segmentation, but the accelerated RANSAC is significantly faster.



(a) (b)

Figure 5: Comparison of the segmentation using RANSAC with normals. (a): Represents the original algorithm, while (b): the accelerated one.