TP 5: Modeling

Objectives

- Test the RANSAC Shape detection in CloudCompare
- Implement RANSAC in Python and analyse its behaviour

The report should be a pdf containing the answers to the **Questions** and named "TPX_LASTNAME.pdf". Your code should be in a zip file named "TPX_LASTNAME.zip". You can do the report as a pair, just state both your names inside the report and in the pdf and zip filenames, like "TPX_LASTNAME1_LASTNAME2.pdf"
Send your code along with the report to the email mva.npm3d@gmail.com. The object of the mail must be "[NPM3D] TPX LASTNAME" or "[NPM3D] TPX LASTNAME1 LASTNAME2" if you are a pair working on the report.

A. Test RANSAC Shape detection in CloudCompare

1) Open the point cloud "indoor_scan.ply"

2) Try to segment the point cloud into planes by using "Plugins -> RANSAC Shape Detection" and check only the "Plane" element.

3) Test different parameters to get the "best" segmentation

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.

B. Implement RANSAC in Python

You will implement a RANSAC plane detection in Python. The algorithm takes three random points in the points cloud, compute the plane passing through the 3 points and check the number of points belonging to that plane (inliers). After a fixed number of trials, RANSAC keeps the plane with the maximum number of inliers points.

In our implementation, a plane is defined by a point and a normal.

1) In ransac.ply write a function compute_plane(points) that computes the plane passing through three points represented by the three first lines of matrix points.
   Tip: You can compute the normal of the plane using the cross product of the two vectors p0p1 and p0p2 (where p0, p1, p2 are the three points) and use one of the points as point of the plane

2) Write a function in_plane(points, pt_plane, normal_plane, threshold_in=0.1) that returns an array of size points with 1 for inliers, points of points belonging to the plane plane at a distance smaller than threshold_in and 0 for others (outliers).

3) Write a function RANSAC(points, nb_draws=100, threshold_in=0.1) that computes the best plane fitting the cloud points by sampling randomly nb_draws triplets of points in points and counting the number of points in points at a distance smaller than threshold_in. The plane kept being the one with the most votes.

4) Write a function recursive_RANSAC(points, nb_draws =100, threshold_in=0.1, nb_planes=2) that apply RANSAC nb_planes times recursively (it means we apply a first RANSAC to detect the best plane in the point cloud and then remove the points belonging to that plane, then do a new RANSAC on the remaining points, detect the second best plane, extract the points...). Try with nb_planes=2 on indoor_scan.ply,

You should obtain something like the following screenshot (first extracted plane in blue, second in red):

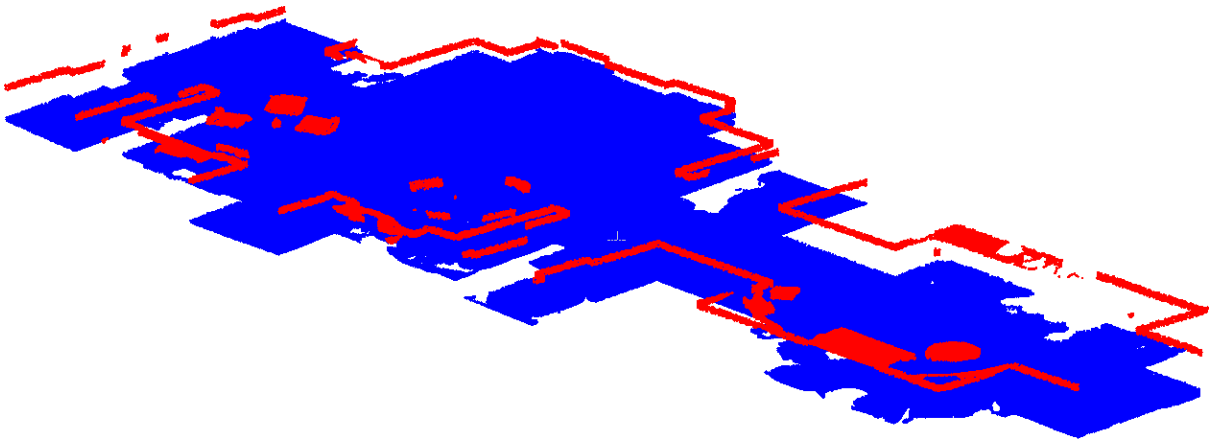

Figure 1 Two planes extracted consecutively by RANSAC

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

<u>Question 3:</u> 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.

Implement a variant of RANSAC, using normal on points (with scripts from TP3) to deal with the issue of the previous section.

<u>Question 4:</u> 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.

PS: For all experiments, you cannot use the label information present in the point cloud data.

C. Going further (BONUS)

RANSAC is a slow method because it requires testing many hypotheses to find a plane.

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.