TP 5: Modeling

Objectives

- Implement RANSAC and analyse its behaviour.
- Implement Region Growing algorithm and analyse its behaviour.

A. RANSAC

A plane can be defined either by (a point, a normal) or by (a distance to the origin, a normal). You can use your favorite convention.

1) In RANSAC.ply write a function compute_plane(cloud) that computes the plane passing through the three points represented by the three first lines of matrix cloud.

2) Write a function in_plane(cloud, plane, threshold_in) that computes all points of cloud belonging to the plane plane at a distance smaller than threshold_in.

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

4) Write a function recursive_RANSAC(cloud, n, threshold_in, m) that apply RANSAC m times recursively. Try with m=2 on indoor_scan.ply,

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

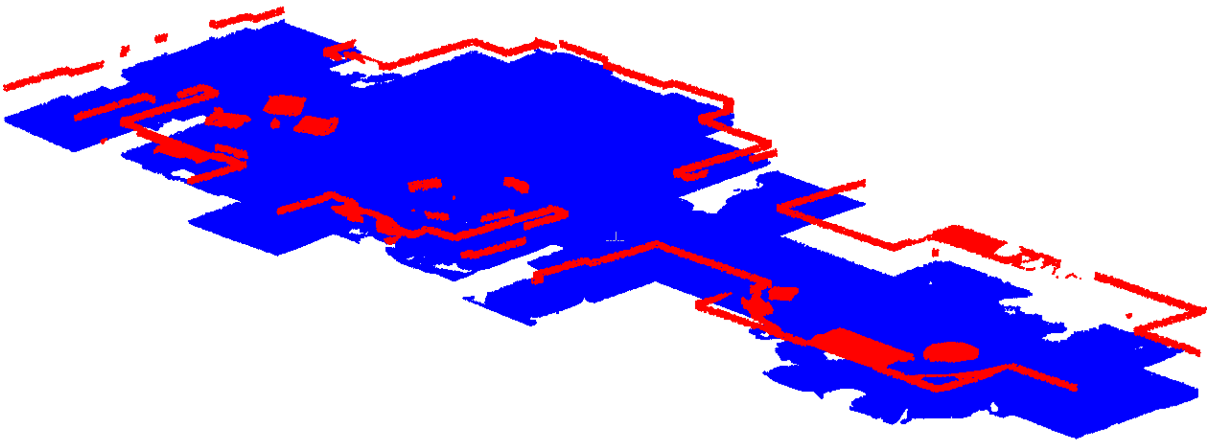

Figure 1: Two planes extracted consecutively by RANSAC

We can be satisfied with the extraction of the blue plane, but we would have preferred that the second plane extracted would be one of the walls, or at least would be in a single piece.

<u>Question 1:</u> Explain what produces this behaviour.

<u>Question 2:</u> Do you have any ideas to prevent this behavior from happening?

B. Region Growing

5) In RegionGrowing.ply write a function compute_curvatures_and_normals(cloud, search_tree, r) that computes for each point of cloud the curvature c (given by formula [1]) and the normal using the search_tree to find neighborhoods with the radius r. You can use your code from TP4.

$$c = \frac{\lambda_3}{\lambda_1+\lambda_2+\lambda_3} \quad [1]$$

where $\lambda_1 \geq \lambda_2 \geq \lambda_3 > 0$ are the 3 eigenvalues of the local covariance matrix.

6) Write a function region_criterion(p1, p2, n1, n2) that returns True if point p2 belongs to the plane defined by the point p1 and the normal n1, and if normals n1 and n2 form an angle smaller than a certain threshold.

7) Write a function queue_criterion(c) that returns True if curvature c is smaller than a certain threshold.

As a reminder, the steps of Region Growing algorithm are:

- Find a seed that we put in a queue Q
- Instantiate a region R, containing the seed
- While the queue Q is not empty:
    - extract one point q of the queue Q,
    - find all its neighbors,
    - For each neighbor p:
        - If p, q and their normals verify some criterion:
            - add p to the region R,
            - If p verify some criterion (for exemple on its curvature):
                - add p to Q.

8) Write a function RegionGrowing(cloud, normals, curvatures, search_tree, radius, region_criterion, queue_criterion) that applies this algorithm to find a plane in cloud. Apply this function to indoor_scan.ply. Test different values of radius and different thresholds in functions region_criterion and queue_criterion.

Question 3: How does the values of radius and thresholds affect the plane segmentation?

Question 4: Do you have any ideas to find a seed which increases the chances of finding a plane?

9) Write a function recursive_RegionGrowing(cloud, normals, curvatures, search_tree, radius, region_criterion, queue_criterion, m) that apply the Region Growing algorithm m times recursively. Try with m=2 on indoor_scan.ply,

Question 5: Show a screenshot of the extracted planes, with a different color for each plane. You should obtain something like in figure 2.

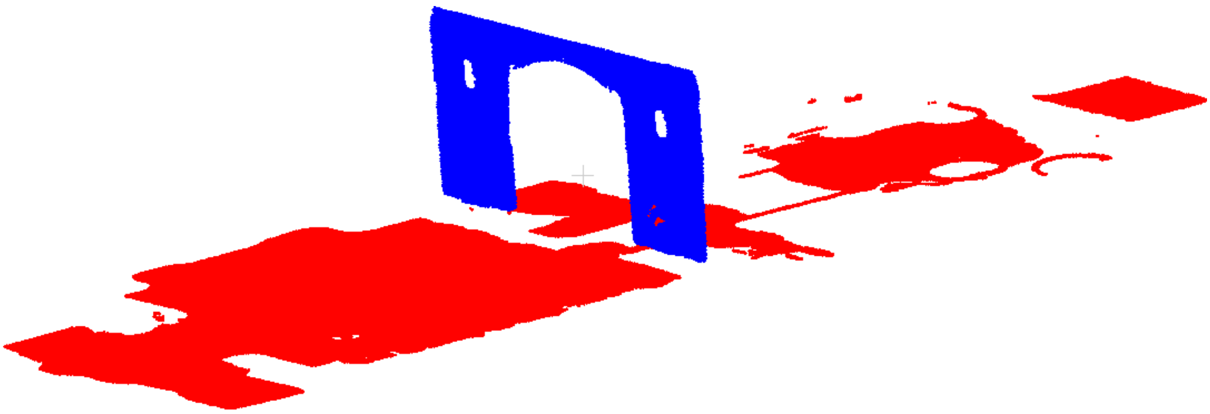

Figure 2: Two planes extracted by Region Growing