TP : 3D Modelling

MAREVA - October 2nd, 2019
hugues.thomas@mines-paristech.fr

Objectives

- Implement RANSAC algorithm and analyse its behaviour.

A. RANdom SAmple Consensus

There are several ways to define a plane, here we will use a point and a normal for convenience.

1) In RANSAC.py write the function compute_plane that computes the plane passing through the three points. You just have to choose one of the three points and compute the normal.
   Hint: use np.cross for cross product. Remember to normalize your normal vector.

2) Implement the function in_plane that takes as input a list of points, a plane, and a threshold value and returns the indices of the points whose distance to the plane are smaller than *threshold_in*.
   Hint: use np.dot for scalar product.

The RANSAC algorithm follows a very simple concept of trial and errors. In our case, we want to use it to find the prominent plane in a point cloud. Each iteration consists of two simple steps:

- Randomly sample 3 points from the cloud. Compute the plane they define.
- Count how many points from the cloud are in range of this plane as votes.

The plane that has the most votes is kept as the prominent plane.

3) Write the function RANSAC returning the prominent plane in a point cloud, given a number of tries and a threshold distance.
   Hint: use np.dot for scalar product.

<u>Question 1:</u> What does the prominent plane represent in the cloud? How many points does it count?

<u>Question 2:</u> How many tries do you need to get 99% chance of finding this plane.

We may want to extract more than one plane from a cloud. This can be achieved quite easily by applying RANSAC multiple times on the cloud. The points from the found planes just need to be removed between each new RANSAC call.

4) Write a function multi_RANSAC that apply RANSAC m times on a point cloud. Try with m=5 on indoor_scan.ply,

<u>Question 3:</u> Show a screenshot of the extracted planes. Are you satisfied with the extraction? Explain what produces this behaviour.