RANSAC

In this exercise, similarly to exercise 4.1, we want to detect the cylinder in the given point cloud. To solve this exercise, we can use the function pcfitcylinder. This function implements a special version of RANSAC, which is M-estimator SAmple Consensus (MSAC), where the main idea is to evaluate the quality of the consensus set calculating its likelihood to compensate the undesiderable related to a too big or too small choice of the threshold.

CODE DESCRIPTION

First of all, the pointCloud.mat matrix is imported and the ptCloud to be used in this exercise is defined and displayed.

Then, the parameters to be used in the algorithm are defined. The threshold for the cylinder has a small value (0.005) in order to avoid wrong detection of points. Then, a region of interest is defined: unfortunately, the algorithm is not able to detect the cylinder (for the reasons explained in the “FINAL CONSIDERATION” section) if this range is not defined. The region defines the range along x, y and z direction where the cylinder is located, and the values can be assessted only by visual inspection of the original point cloud. The findPointsInROI function is used to select the points in this region (the points within the specified ROI are obtained using Kd-tree based search algorithm.). The additional axis of the cylinder (which in this case is the axis z = [0 0 1]) should also be defined.

At this point we can call the function pcfitcylinder, which receives in input the point cloud, the threshold, the cylinder axis and the samples where to look for the cylinder and return the model parameters (as a special class called “cylinderModel” where Parameters, Center, Orientation, Height and Radius of the defined cylinder are stored), the inliers and the outliers (both as a column vector containing the linear indices of the inlier and outlier points). The pcfitcylinder function is the core of the whole algorithm: in it, there are all the instruction to perform RANSAC for the segmentation of the cylinder.

After the calling of the pcfitcylinder function, the select function is used to return a pointCloud object containing only the passed indices. This point cloud contains all the cylinder inliers.

Finally, as requested by the exercise, the pose of the cylinder, together with other parameters, are displayed in the command window:

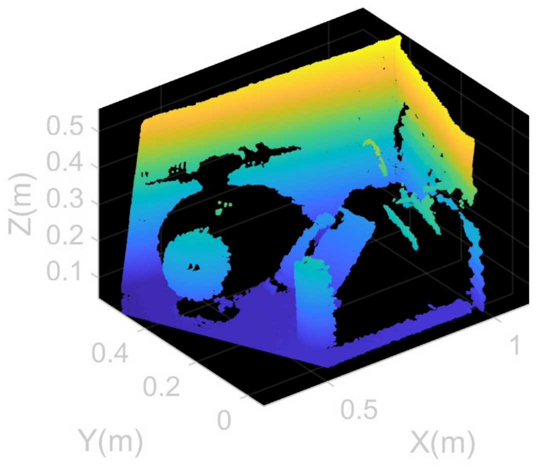
```
>> BUSETTO_RANSAC_cylinder
```

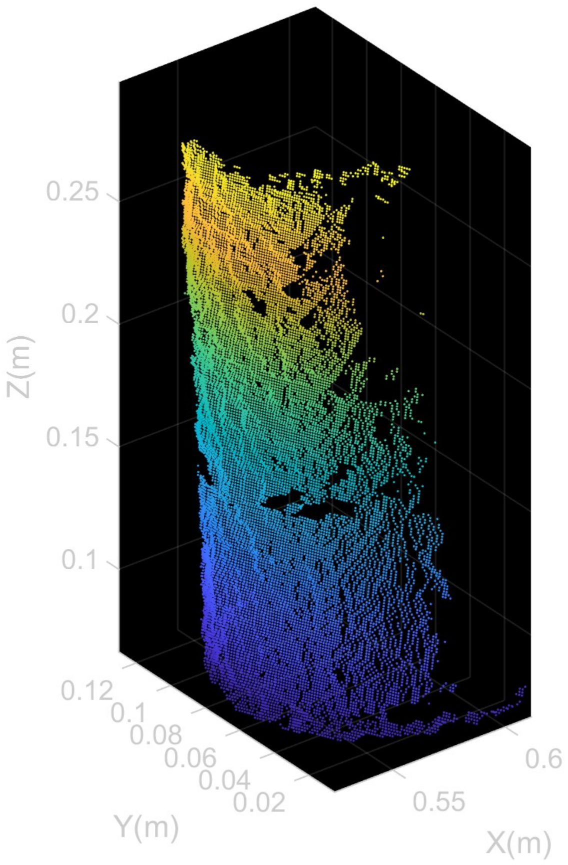
```
Position of the cylinder:
 5.876414e-01
7.333773e-02
1.818918e-01


Pose of the cylinder:
 -1.120579e-02
1.177479e-02
2.308788e-01


Radius of the cylinder: 5.815265e-02


Height of the cylinder: 2.314503e-01
```


0.5
0.4
0.3
0.2
0.1
Z(m)
0.4
0.2
0
Y(m)
0.5
1
X(m)


0.25
0.2
0.15
0.1
Z(m)
0.12
0.1
0.08
0.06
0.04
0.02
Y(m)
0.55
0.6
X(m)

FINAL CONSIDERATIONS AND ANALYSIS OF RESULTS

Before defining the region of interest, I run the code many times but there was no chance to get the right cylinder of the point cloud. Unfortunately, this region must be defined (even if in principle the algorithm should be able to detect the cylinder without the need to define any region). The reason is related to the number of points that are in the cylinder in the point cloud: actually, only half of the cylinder is shown in the point cloud and this makes the cylinder more difficult to find for the algorithm.

However, how you can see from code of EXERCISE 4.1, in the very last part I implemented the same function to see if it works without the region of interest definition after the removal of all the three planes, and results are positive: often, the algorithm correctly detects the cylinder without passing ROI as an input. In fact, removing the planes a lot of points are removed, reducing the probability for the pcfitcylinder function to wrongly select planes’ points as cylinder points!

The implemented code has proven to be robust for the cylinder detection, and the pcfitcylinder function is extremely fast in returning the result.