CSI 2120 Programming Paradigms - Project 4

27 Mar, 2023

Meet Mehta
Student ID: 300261159

Overview

We define the given predicates in the questions and read the points from the provided files and store it in a point cloud called Points. The Points is then used to generate random points and run a partial RANSAC algorithm to get the most dominant plane based on the defined values of confidence, percentage and the eps distance.

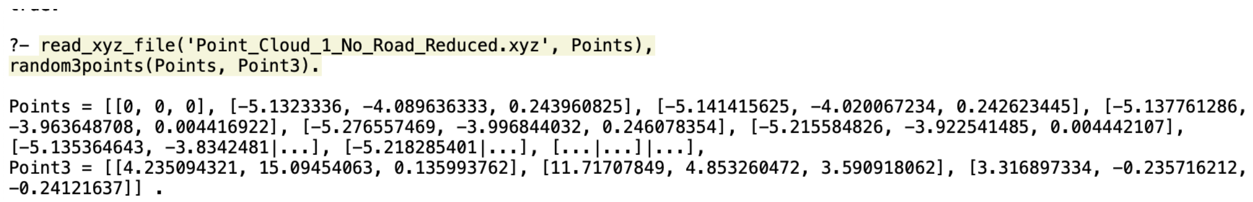
Below are the test cases of each of the predicates as asked in the question. The actual code can be referred to in the prolog file.

Testing the predicates

I. random3points(Points, Point3) :-

We can run the following commands to test and see if the predicate outputs the correct information

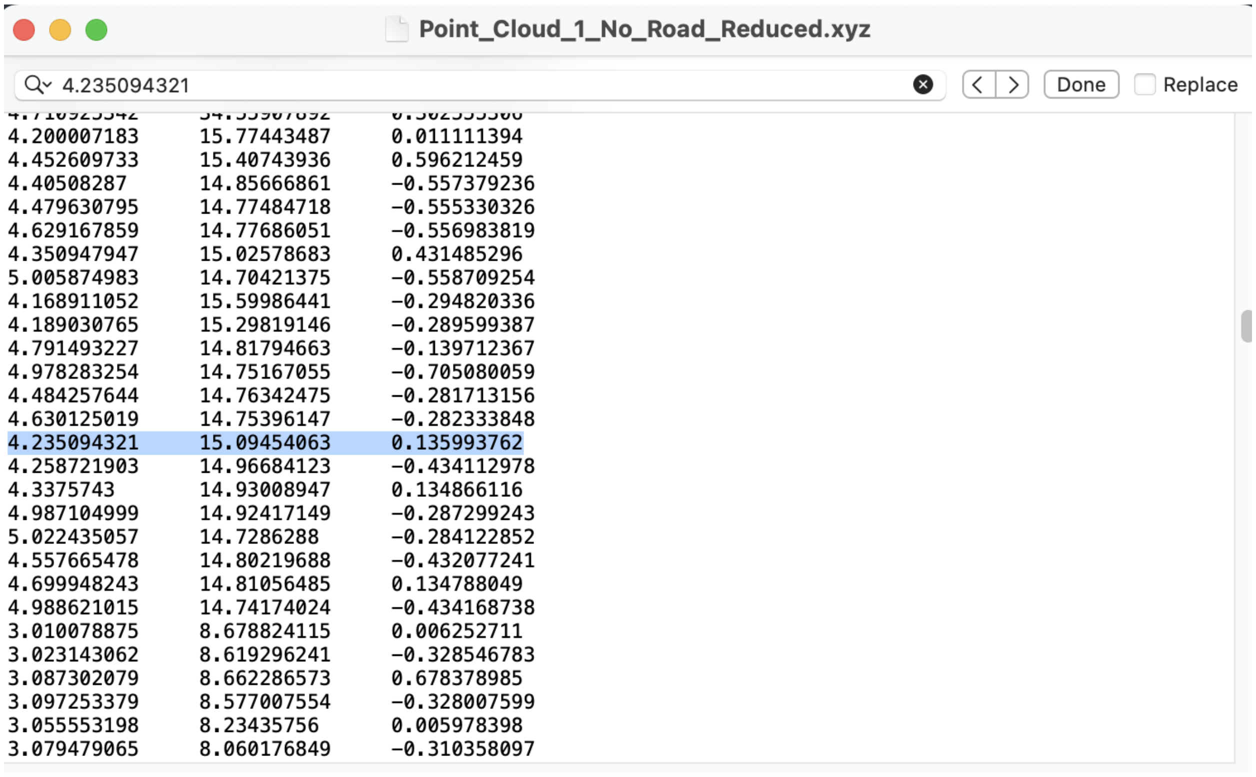
- read_xyz_file('Point_Cloud_1_No_Road_Reduced.xyz', Points), random3points(Points, Point3).



```
?- read_xyz_file('Point_Cloud_1_No_Road_Reduced.xyz', Points),
random3points(Points, Point3).

Points = [[0, 0, 0], [-5.1323336, -4.089636333, 0.243960825], [-5.141415625, -4.020067234, 0.242623445], [-5.137761286,
-3.963648708, 0.004416922], [-5.276557469, -3.996844032, 0.246078354], [-5.215584826, -3.922541485, 0.004442107],
[-5.135364643, -3.8342481|...], [-5.218285401|...], [...|...]|...],
Point3 = [[4.235094321, 15.09454063, 0.135993762], [11.71707849, 4.853260472, 3.590918062], [3.316897334, -0.235716212,
-0.24121637]] .
```

Figure: Output in the SWI-Prolog terminal as we run the above mentioned command to test the given random3points(Points,Point3) predicate

To make sure that the three random points exist in the file called 'Point_Cloud_1_No_Road_Reduced.xyz', we check the file itself for the points.



```
4.200007183     15.77443487     0.011111394
4.452609733     15.40743936     0.596212459
4.40508287      14.85666861     -0.557379236
4.479630795     14.77484718     -0.555330326
4.629167859     14.77686051     -0.556983819
4.350947947     15.02578683     0.431485296
5.005874983     14.70421375     -0.558709254
4.168911052     15.59986441     -0.294820336
4.189030765     15.29819146     -0.289599387
4.791493227     14.81794663     -0.139712367
4.978283254     14.75167055     -0.705080059
4.484257644     14.76342475     -0.281713156
4.630125019     14.75396147     -0.282333848
4.235094321     15.09454063     0.135993762
4.258721903     14.96684123     -0.434112978
4.3375743       14.93008947     0.134866116
4.987104999     14.92417149     -0.287299243
5.022435057     14.7286288      -0.284122852
4.557665478     14.80219688     -0.432077241
4.699948243     14.81056485     0.134788049
4.988621015     14.74174024     -0.434168738
3.010078875     8.678824115     0.006252711
3.023143062     8.619296241     -0.328546783
3.087302079     8.662286573     0.678378985
3.097253379     8.577007554     -0.328007599
3.055553198     8.23435756      0.005978398
3.079479065     8.060176849     -0.310358097
```

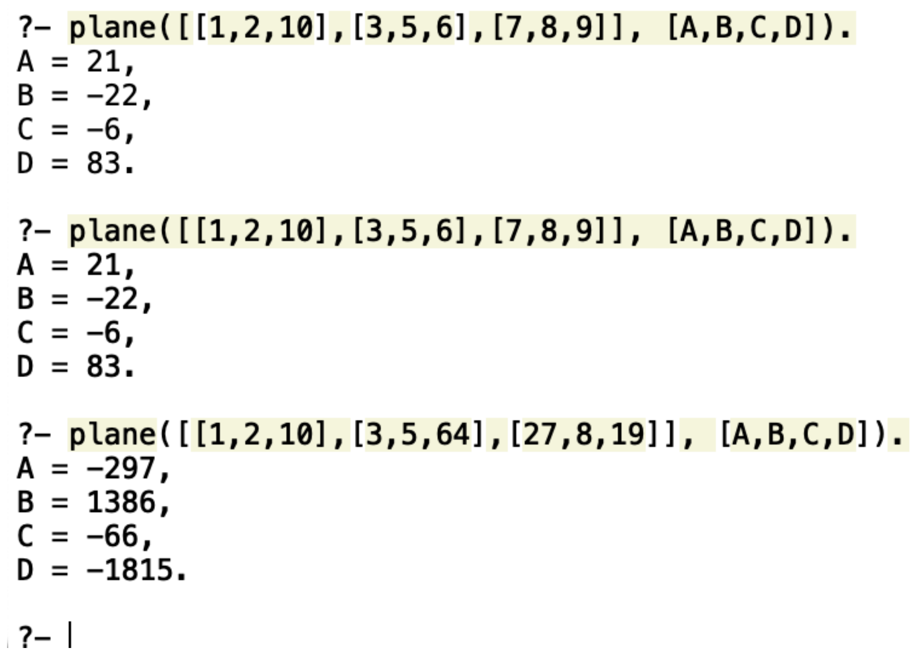
Figure: Point_Coud_1_No_Read_Reduced.xyz file highlighting the points that are a part of three random points generated by the predicate random3points(Points,Point3)

As seen in the screenshot above, the exact three points are a part of the pointcloud provided in the files.

II. plane([[X1, Y1, Z1], [X2, Y2, Z2], [X3, Y3, Z3]], [A, B, C, D]) :-

We can run the following commands to test and see if the predicate outputs the correct information. The command will run the predicate and show the result of A,B,C,D values.

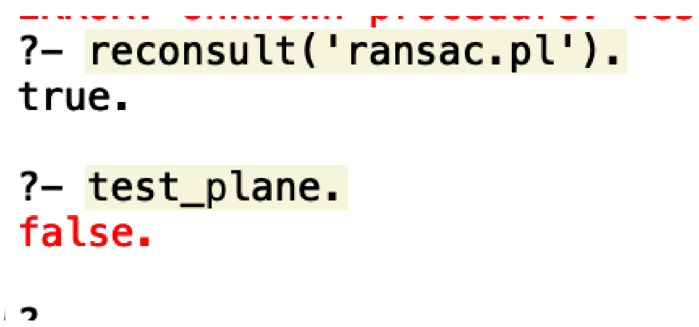
- plane([[1,2,10],[3,5,64],[27,8,19]], [A,B,C,D]).



```
?- plane([[1,2,10],[3,5,6],[7,8,9]], [A,B,C,D]).
A = 21,
B = -22,
C = -6,
D = 83.

?- plane([[1,2,10],[3,5,6],[7,8,9]], [A,B,C,D]).
A = 21,
B = -22,
C = -6,
D = 83.

?- plane([[1,2,10],[3,5,64],[27,8,19]], [A,B,C,D]).
A = -297,
B = 1386,
C = -66,
D = -1815.

?- |
```

Figure: Plane predicate running to output the values of A,B,C,D of the given plane

We have a test_plane function that will return true if the points in Point3 make the plane that make the plane 'Plane'.

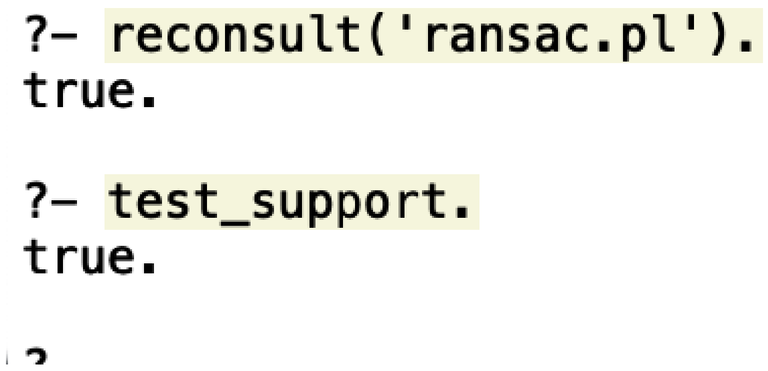


```
?- reconsult('ransac.pl').
true.

?- test_plane.
false.
```

Figure: test_plane predicate called to check the correctness of the predicate Plane

III. support([A, B, C, D], Points, Eps, N) :-

In the test case, we are passing the values of the coefficients of the plane i.e, A,B,C,D and the values of five points which satisfy the equation of the plane. We define the distance to be 0.1 from the plane and to return at least 3 points. As we can see in the image below, the predicate test_support returns true.

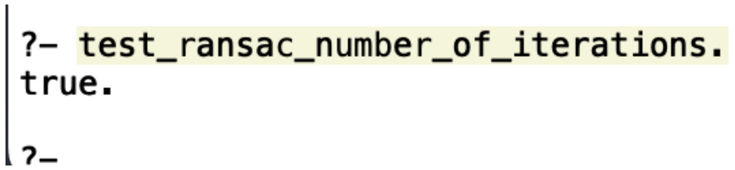


```
?- reconsult('ransac.pl').
true.

?- test_support.
true.

?
```

Figure: test_support predicate called to check the correctness of the predicate Support

IV. support([A, B, C, D], Points, Eps, N) :-

In the test below, we are passing 0.95, 0.3 for confidence and percentage respectively. Then we calculate the expected value using the given formula. If the function ransac_number_of_iterations return the same value (which it should), the predicate is deemed to be true.



```
?- test_ransac_number_of_iterations.
true.

?-
```

Figure: test_support predicate called to check the correctness of the predicate Support

Conclusion

In conclusion, we have successfully defined test cases for each of the predicates that were given in the question.