CS 4610/5335: Robotic Science and Systems (Spring 2024)	X. Zhi Tan
Northeastern University	version: 1.0.1

HW 3: Point Cloud Analysis

Please remember the following policies:

- Submissions should be made electronically via the Canvas. Please ensure that your solutions for both the written and/or programming parts are present and zipped into a single file.

- Solutions may be handwritten or typeset. For the former, please ensure handwriting is legible.

- You are welcome to discuss the programming questions with other students in the class. However, you must understand and write all code yourself. Also, you must list all students (if any) with whom you discussed your solutions to the programming questions.

- Any use of generative AI must follow the class' generative AI policy.

All of the code for this assignment is in Python, and located within 'hw3_code_YYYYMMDD.zip'. We will use the same conda environment installed in HW2 with one additional package (Scipy). Recommended Steps:

- Download `hw3.zip` from Canvas. Unzip the file to your preferred directory (e.g., `/Users/Zhi/RSS/HW3`).

- Activate the previous environment - `conda activate rss-hw2`

- Install scipy - `conda install scipy==1.11.4 -y`.

- Navigate to the directory and try running `python hw3.py -q q1_a`. You will see the figure below.

ChangeLog

- v1.0.1
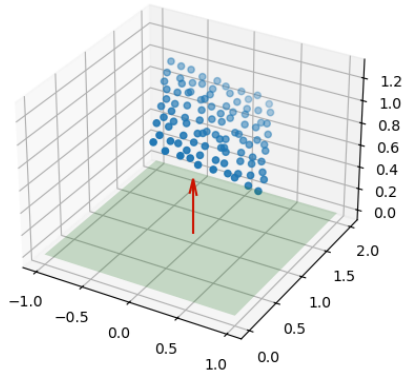 - [**Code**] Remove hint that was no longer relevant in Q2.

Figure 1: The result of running `python hw3.py -q q1_a`.

Q1. Plane fitting (**5 points**).
In this question, you are given a set P of 100 points and asked to fit a plane to these points. The output of these functions should be a point intercept of the plane (a vector called "center") and the plane surface normal (a vector called "normal").

(a) Implement the function `q1_a` in questions.py: fit a plane by calculating the sample mean and covariance matrix of the points. You will need to obtain the Eigen values and vectors of the covariance matrix in order to complete this question. You can test your implementation by running the command `$python -q q1_a` in your terminal.

(b) Test your plane fitting on an example with outliers by running the command `$python -q q1_b`. How is this different from the result in part (a) and why?

(c) Implement the function `q1_c` in questions.py: fit a plane using a ransac based method. You can test your implementation by running `$python hw3.py -q q1_c` in your terminal. What are the strengths and weaknesses of each approach?
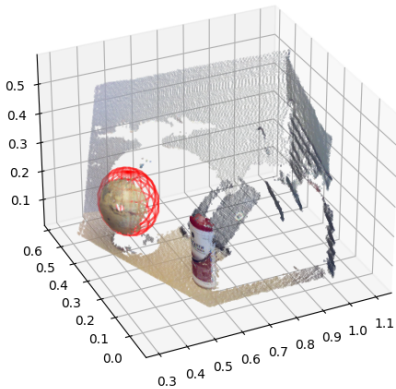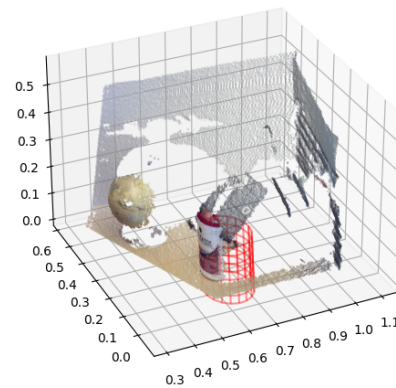

Figure 2: (a) the sphere localized in Q2.

Figure 3: (b) the cylinder localized in Q3.

Q2. Sphere fitting (**5 points**).
In this question, you must use RANSAC to localize a sphere given point cloud data. The point cloud contains a ball that is only partially visible to the sensor. The position of the center of the ball is unknown. The radius is unknown, but between 5cm (0.05m) and 11cm (0.11m). Write the function `q2` in questions.py that calculates these two quantities. Because the operation takes a while (1 min on M2 Max), you can test your code on a cropped point cloud by enabling test mode (`python hw3.py -q q2 -t`). In test mode, only the points around

the sphere are passed to the function. However, you should submit code that works on the entire point cloud (not test mode). Important: please do NOT use any external libraries for plane fitting. However, you MAY use the provided function `utils.estimate_normals` to generate normals from the point cloud.

Hint: one way to generate sphere hypotheses is as follows:

(a) sample a point from the cloud;

(b) sample a radius of the candidate sphere between 5 and 11cm;

(c) project a vector from the sampled point in the direction of the associated surface normal for a distance equal to the sampled radius. This point would be at the center of the candidate sphere.

Q3. Cylinder fitting (**5 points**).
Same as Q3 except for a cylinder. This question is harder because you need to calculate the center, the orientation, and the radius (between 0.05m and 0.1m). The orientation should be returned in the form of a unit vector pointing along the axis of the cylinder (either direction is fine). You only need to solve this problem for the segmented cloud, as implemented in the code. The function you should implement is `q3` in questions.py.

Hint: one way to generate cylinder hypotheses is as follows:

(a) Sample a radius for the candidate cylinder between 5 and 10 cm.

(b) Sample two points from the cloud.

(c) Set the cylinder axis direction equal to the direction of the cross product between the surface normals associated with the two sampled points.

(d) Pick one of the sampled points from the cloud and use it to estimate a candidate center, just as you did in Q1.

(e) Project the points in the cloud onto the plane orthogonal to the axis you just calculated. You can do this projection by multiplying the points in the cloud by this matrix: $I - \hat{a}\hat{a}^T$, where $\hat{a}$ is equal to the axis of the cylinder. Also project the candidate center into this plane in the same way.

(f) Evaluate number of inliers in the plane for a circle with the given projected center and the sampled radius.

Q4. Iterated closest point (ICP) (**5 points**).
Given a set of points M, and another a set of points D with a different pose, ICP algorithm could be used to find a 4-by-4 transformation matrix T so that $D = TM$.

(a) Implement the function `q4_a` to find the transformation matrix that aligns two point clouds given full correspondences between points in the two clouds. In other words, D and M are the same point cloud but in different poses. You can test your implementation by running: `$python hw3.py -q q4_a`

(b) Run `$python hw3.py -q q4_b` to test your implementation from part (a) on noisy data. Explain why the algorithm still works when gaussian noise is added to one of the point clouds, but does not work when the order of the points is shuffled.

(c) Implement the function `q4_c` to perform iterative closest point (ICP). Your implementation should get reasonably alignment on shuffled and noisy data: run `$python hw3.py -q q4_c` to test this.