[SNAR2019] THE LAST HOMEWORK (hope so…)

Богословский А. М.
КРБО-01-16

Цель:

1. Написать программу для нахождения плоскости дороги, основываясь на входном облаке точек с лидара.
2. Написать программу-генератор тестовых облаков точек.

Ход работы:

1. *Отрицание.*

На первом этапе я пытался придумать алгоритм сортировки сам. Я прикинул, что для нахождения верной плоскости нужно перебрать все вариации из 3 точек входного облака точек, и для каждой найти свои коэффициенты уравения плоскости. Далее найти, сколько точек находятся на расстоянии погрешности лидара от плоскости, тем самым получив для каждой плоскости рейтинг. Затем выбрать коэффициенты с самым высоким рейтингом, и вуаля, вот она - искомая плоскость.

2. *Гнев.*

Однако в голову сразу пришли сложности:

- экспоненциальный рост вариаций. Для 10 точек вариаций будет 120, а для 50 - уже 19600, что не лучшим образом будет влиять на производительность и память.
- кропотливая работа с матрицами для нахождения коэффициентов (математика, фу!)
- вопросы к архиектуре хранения всех коэффициентов и к способу нахождения близлежащих точек

3. *Торг.*

Начал гуглить варианты, гуглил на русском, потом на английском. По словам “find plane point cloud” нашел алгоритм RANSAC, который в целом повторял мои мысли, но более изящно:

https://en.wikipedia.org/wiki/Random_sample_consensus

Начал искать готовые варинты алгоритма, нашел это:
https://github.com/AlexandraPapadaki/RANSAC-PlaneDetection/blob/master/Ransac_Plane_detection.m

Но как истинный ленивый кодер я решил, что и это длинновато как-то. В итоге моя лень была вознаграждена: в матлабе есть фунция *pcfitplane()*, которая находит плоскость в облаке точек по алгоритму MSAC (вариация RANSAC). Эта функция является основой моего матлабовского кода.

4. *Депрессия.*

Скучный этап написания кода матлаба, затупливание на преобразованиях типов и создании рандомных значений.

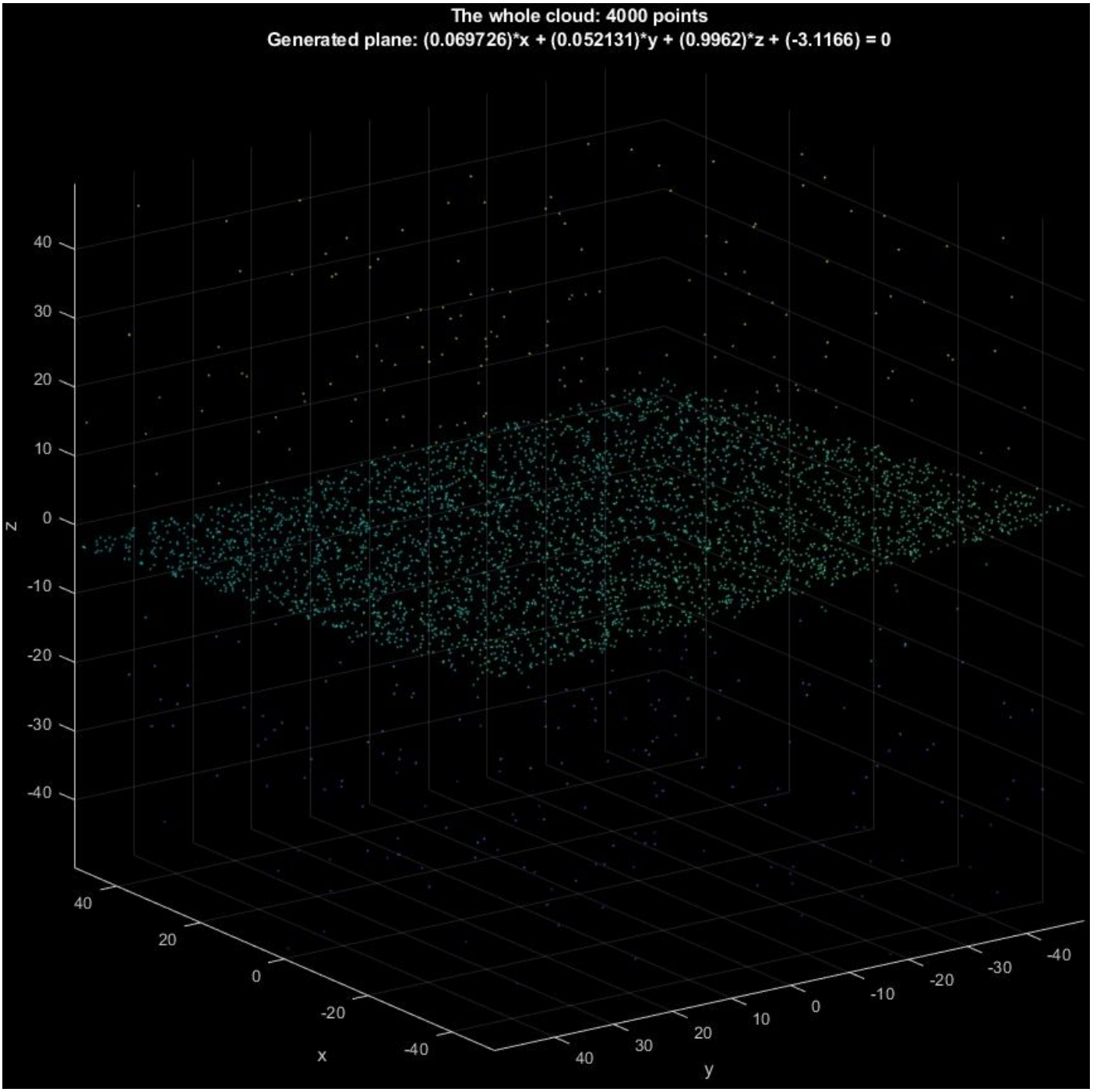
https://github.com/slaveareyou/lidar_plane_ransac

В проекте 2 файла: генератор и вычислитель плоскости, на вход вычислителя подается выход генератора, в котором хранятся коэффициенты правильной плоскости для тестов, и название файла (расширение *.txt* добавляется автоматически), в которое было записано облако точек.

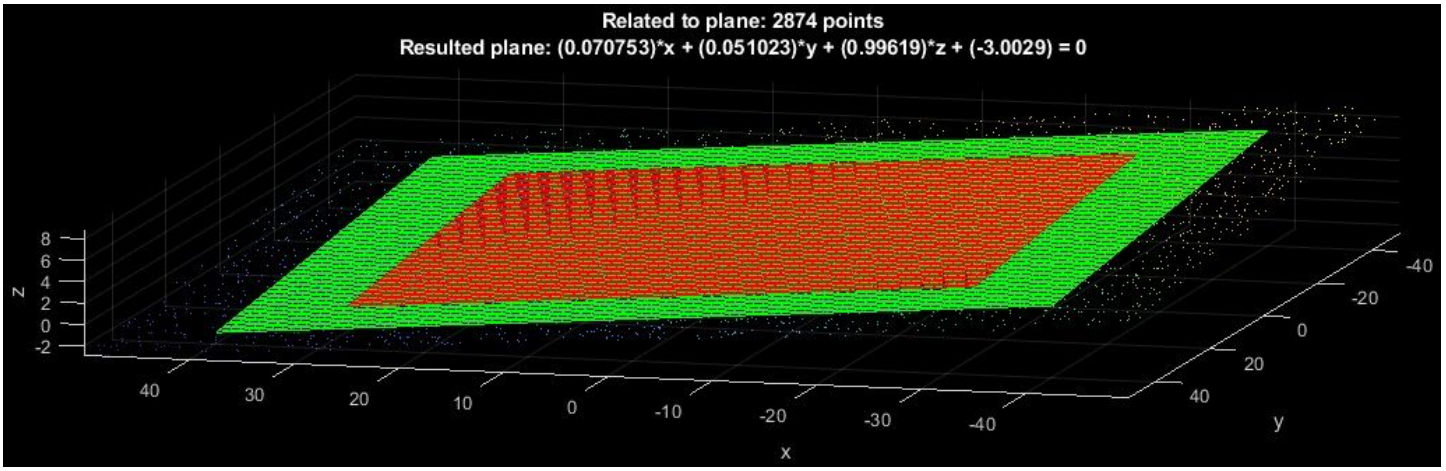
5. *Принятие.*
   Для использования программы, после компиляции кода, нужно ввести это:

```
road_lane(road_lane_generate('test', 4000))
```

4000 - суммарное количество точек в облаке. Как минимум половина будет принадлежать начальной плоскоти, остальные будут равномерно зарандомлены.



На скрине выше визуализирован входное облако точек.



А тут уже результат. Показываются точки, только принадлежащие плоскости. Зеленая плоскость - правильная исходная плоскость. Красная плоскость - практически вычечсленная. Во всех экспериментах все 3 слоя графика совпадают, ну или почти совпадают.

Пояснения кода, отвечающего за обработку точек:

```
referenceVector = [0,0,1];
maxAngularDistance = 20;
[model,inlierIndices] = pcfitplane(ptCloud,maxDistance,
referenceVector,maxAngularDistance);
```

1 строка: задание вектора нормали для искомой плоскости. То есть если будет несколько кандидатов, выбирается ближний к нормали.

2 строка: задание угла, на который искомая поверхность может отклоняться от нормали. Я не пожалел и поставил 20 градусов (знаю я ваши “ровные” дороги)

3 строка: в model записываю информацию о плоскости, в том числе коэффициенты; в inlierIndices пишем индексы точек, принадлежащие плоскости с учетом погрешности датчика, поданного в виде maxDistance; ptCloud - изначальное облако точек.

Ну вроде про все написал.

P.S. Надеюсь, Вы покрутите-посмотрите графики, и если что непонятно, напишите вопрос: с радостью отвечу. Я в чате телеги с ником ‘.’ (самоуверенный ник, я знаю).