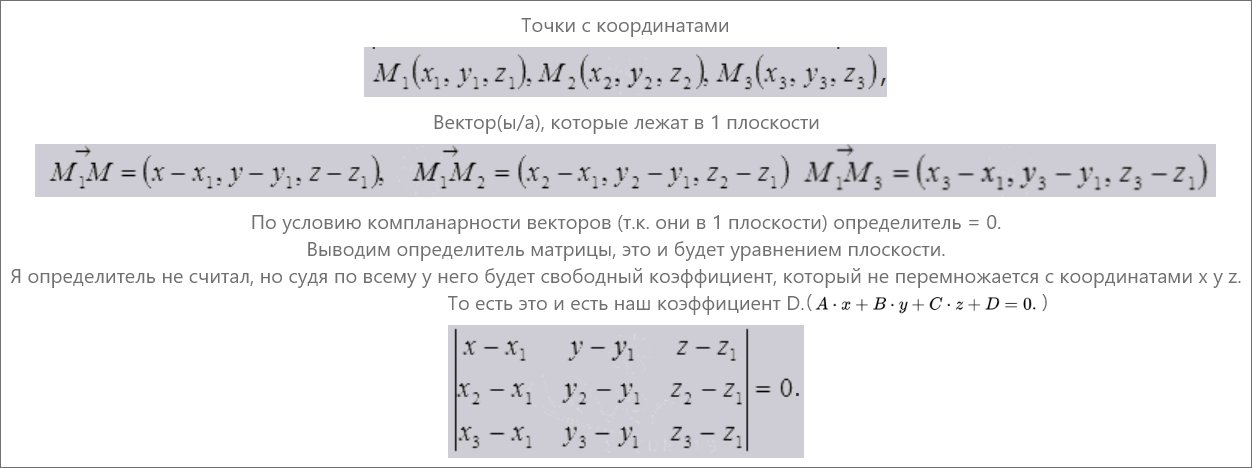
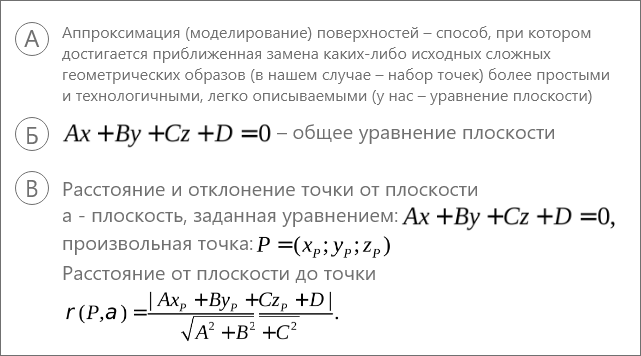
**Введение и немного базовой математической теории**

На повестке дня отчет по работе программы. Отчет в понимании его автора – это небольшая история успеха команды. Поэтому начнем описание проделанной работы издалека.

1-й курс был настолько далек в момент прочтения задания, что за радость было вспомнить факт – плоскость строится минимум по 3-м точкам, не лежащим на одной прямой (Рис. 1).

  
Рис. 1 – Вывод уравнения плоскости по 3-м точкам

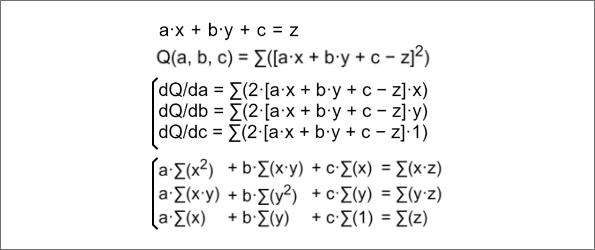
Работа в команде означает неизбежное столкновение разных мнений, как до полного понимания поставленной задачи, так и после. Наш случай не уникальный, поэтому без жаркого обсуждения конечной цели дела, естественно, не обошлись. Но в споре всплыло несколько важных понятий вроде аппроксимации (Рис. 2 - А), основного уравнения плоскости (Рис. 2 - Б) и формулы расстояния от точки до плоскости (В).

  
Рис. 2 – Аппроксимация, общее уравнение прямой, расстояние от точки до плоскости

Первые наработки дали понять, что к работе необходимо подойти намного серьезнее и нужно очень хорошо понимать, как отличать точки пола от точек других посторонних объектов, которые заметил лидар, а так же, как повысить точность модели. Первая идея, пришедшая в голову, подразумевала работу с “методом наименьших квадратов”.

**Описание работы алгоритма**

Идея алгоритма заключается в следующем. Метод наименьших квадратов подбирает такой набор параметров точек, при которых все точки расположены максимально близко к плоскости. Получается это за счет минимизации уравнения плоскости по трем неизвестным, то есть задача сводится к решению системы из трех уравнений, в которой производные функции по каждой неизвестной приравнены к нулю (Рис. 4).

  
Рис. 4 – Минимизация уравнения плоскости

После ряда не хитрых математических преобразований получилось найти прямую зависимость значений параметров плоскости от координат точек, по которым строится аппроксимированная плоскость:

*a1·x1 + b1·x2 + c1·x3 = d1*

*a2·x1 + b2·x2 + c2·x3 = d2*

*a3·x1 + b3·x2 + c3·x3 = d3*

Избавимся от х1 во втором и третьем уравнениях системы, воспользовавшись третьим эквивалентным преобразованием. Определим такие коэффициенты:

*k12 = −a2/a1* – коэффициент для преобразования второго уравнения,

*k13 = −a3/a1* – коэффициент для преобразования третьего.

Теперь если умножить каждый элемент первого уравнения на k12 и получившееся уравнение прибавить ко второму, множитель при х1 второго уравнения обнуляется:

*a1·k12·x1 + a2·x1 = (a1·(−a2/a1)·x1) + a2·x1 = (−a2·a1/a1 + a2)·x1 = 0·x1 = 0*

*b1·k12·x2 + b2·x2 = (b1·k12 + b2)·x2*

Аналогично с k13 – умножим на него каждый член первого уравнения и прибавим первое уравнение к третьему. Система становится такой:

*a1·x1 + b1·x2 + c1·x3 = d1*

*(b2 + k12·b1)·x2 + (c2 + k12·c1)·x3 = d2 + k12·d1*

*(b3 + k13·b1)·x2 + (c3 + k13·c1)·x3 = d3 + k13·d1*

Осталось избавиться от х2 в третьем уравнении, и система станет треугольной. Найдем коэффициент для умножения на каждый член второго уравнения, такой, чтобы х2 в третьем уравнении после сложения его со вторым пропал:

*k23 = −(b3 + k13·b1) / (b2 + k12·b1)*

Получаем систему, готовую к решению:

*a1·x1 + b1·x2 + c1·x3 = d1*

*(b2 + k12·b1)·x2 + (c2 + k12·c1)·x3 = d2 + k12·d1*

*((c3 + k13·c1) + k23·(c2 + k12·c1))·x3 = (d3 + k13·d1) + k23·(d2 + k12·d1)*

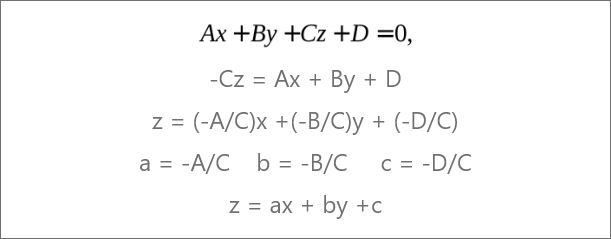
Как итог получим:

***x3 = ((d3 + k13·d1) + k23·(d2 + k12·d1)) / ((c3 + k13·c1) + k23·(c2 + k12·c1))***

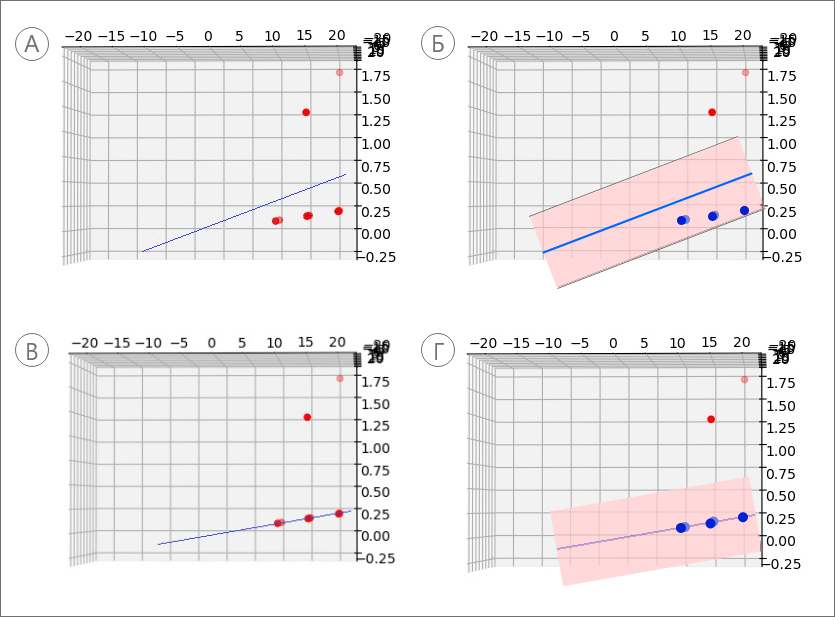
***x2 = (d2 + k12·d1 − (c2 + k12·c1)·x3) / (b2 + k12·b1)***

***x1 = (d1 − b1·x2 − c1·x3) / a1***

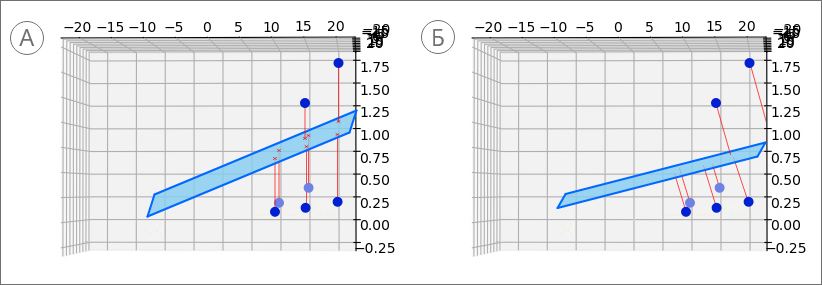
Подставив найденные х1, х2 и х3 вместо а, b и с в формулу a·x + b·y + c = z, получаем уравнение плоскости, аппроксимирующей нашу поверхность оптимально с точки зрения МНК. Для получения общего уравнения плоскости проведем преобразования, представленные на рисунке 5, только в обратном порядке.

  
Рис. 5 – Переход от общего уравнения плоскости

После того как уравнение было выведено, а плоскость построена (Рис. 6 – А), начинается работа основного алгоритма. Его основной задачей является “проверка качества пола”. Главные критерии для нас — это как много точек укладывается в существующую плоскость и насколько сильно удалены соседи. Для того что бы определить расстояние проведем вектор нормали и сравним его с параметром отклонения. После этой оценки найдем точки, входящие в установленный промежуток (Рис. 6 – Б) и проведем через них новую плоскость (Рис. 6 – В). Далее по тем же критерием оценим новые параметры уравнения (Рис. 6 – Г) и будем повторять алгоритм до тех пор, пока большинство точек не будет принадлежать полу. Графически это можно представить в таком виде:

  
Рис. 6 – Работа 1го составленного алгоритма по Методу Наименьших Квадратов

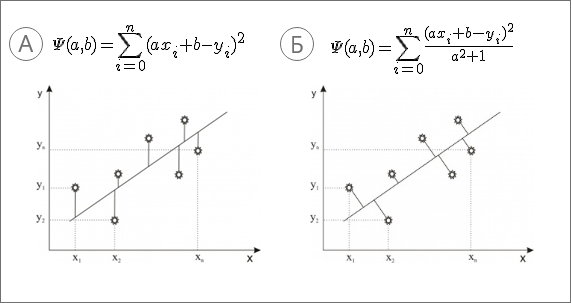
Такой алгоритм позволяет быстро найти необходимую поверхность, но как оказалось на практике он не слишком надежный и очень восприимчив к входным данным. Так, вместо поиска искомого расстояния от плоскости до точки (Рис. 7 - Б) уравнение ориентировалось на расстояние от плоскости до точки по оси z (Рис. 7 - А), что сильно ухудшало работу алгоритма.

Рис. 7 – Сравнение Метода Наименьших Квадратов и Метода Наименьших расстояний

Для новой проблемы конечно было найдено новое решение, но подвело незнание мат части. Необходимо было минимизировать сумму оригинального уравнения плоскости, при условии, показанном на картинке (Рис. 8).

  
Рис. 8 – Условия выполнения МНР

Можно было оставить часть уравнения прежней, а знаменатель пополнить новым условием. Удалось найти подобный трюк в сравнении для расстояния от прямой до точки по оси y (Рис. 9 - А) и искомому расстоянию (Рис. 9 - Б), но с плоскостью аналогию провести не вышло.

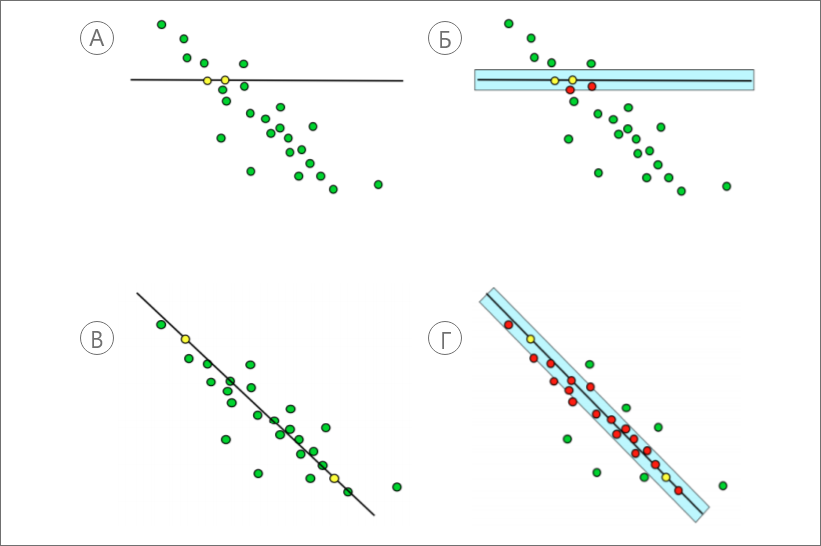
  
Рис. 9 – Сравнение работы МНР и МНК для прямой

Ядро программы было написано, но после обсуждения всех минусов алгоритма был выбран более гибкий способ нахождения оптимального уравнения плоскости.

**Более оптимальный и крутой алгоритм**

Настала пора плавно перейти к методу RANSAC, аббревиатура которого расшифровывается как Random Sample Consensus. От других алгоритмов он отличается тем, что позволяет отбросить шум путём перебора плоскостей, построенных по случайным 3-м точкам, проверяя количество входящих точек в заданную погрешность. Если количество точек, входящих в погрешность по выведенному уравнению плоскости, переваливает за 50% от общей суммы всех точек – алгоритм останавливается и выдает значение коэффициентов уравнения. Алгоритм можно модифицировать методом наименьших квадратов аппроксимируя плоскость по вошедшим точкам, но оставим этот вопрос до следующего раза.

Для полного понимания работы RANSAC рассмотрим пример с линией в двумерном пространстве. Алгоритм выбирает 2 точки и строит по ним прямую (Рис. 10 – А), затем проверяет, сколько точек входит в заданную погрешность по полученному уравнению (Рис. 10 – Б). Если количество точек меньше 50%, алгоритм выбирает новую пару точек (Рис. 10 – В) и проверяет количество входящих точек в установленную погрешность (Рис. 10 – Г). В этот раз условие 50%+ выполнено. Уравнение данной линии выводим как решение.

  
Рис. 10 – Работа алгоритма RANSAC в двумерном пространстве

Основная идея метода идеально подходит для полученного задания, потому что нам удается учесть погрешность, которую мы, ко всему прочему, ещё и сами задаем.

**Перенос алгоритма на компьютер**

В процессе обдумывания алгоритма была предложена блестящая мысль – для большей наглядности мы решили визуализировать входные данные (точки), и результат (плоскость). Благодаря обилию графических модулей языка python сухие цифры превратились в красивые картинки в трехмерном пространстве, которые еще и крутить можно.

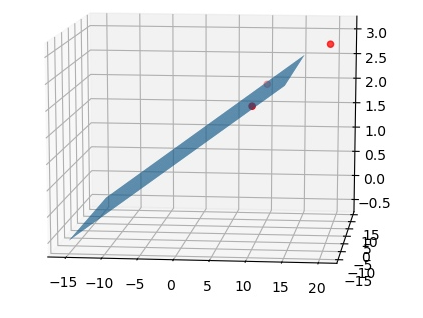
Работу программы можно разделить на 3 части.

Первая и основана часть просто принимает в себя три точки и по ним строит плоскость.

Вторая следит за тем сколько точек (с учётом погрешности) находится на поверхности и сохраняет лучшие варианты. В том случае, если программе удастся найти такое уравнение, при котором все точки будут находиться на “полу” (все точки входят в установленную погрешность отклонения от плоскости), то она экстренно завершит свою работу и запустит следующий шаг.

На третьем этапе программа упрощает и приводит уравнение в нормальный вид и так же создает отдельное окно и наносит график с точками.

Оцените качество работы алгоритма без бумаги и карандаша!

  
Рис. 11 – Результат работы программы



**Вывод**

Кому-то может показаться, что новый метод – это шаг назад. Ведь уравнение плоскости по 3-м точкам – пройденный этап, а кучу потраченного времени на реализацию так и не законченного метода наименьших квадратов, которая могла стоить денег, уже не вернуть. Но не тут-то было. Мы вовремя отказались от сложной затеи, использовали наши наработки, модифицировали их во что-то более приятное, чем просто слова и решили поставленную задачу. Если это не успех, то что тогда.

**Код**

<https://pastebin.com/emdphSnf> - RANSAC

<https://pastebin.com/FgVzjJJX> - Чуть менее удачный метод наименьших квадратов

**Команда #13**

Царьков Даниил  
Эсаулов Илья   
Строков Роман  
Федорович Кузьма