Computer Vision: Exp 2 Report
RANSAC Plane Fitting

Zhuoyang Jiang, 58119125

***Abstract*—n this experiment, I choose MATLAB to solve the question. I randomly generate the 3D data points to be fitted. Then I complete the RANSAC Algorithm to acquisite the inlier set of the best fitted result and use the least square method to obtain the best fitted the plane.n this experiment, I choose MATLAB to solve the question. I randomly generate the 3D data points to be fitted. Then I complete the RANSAC Algorithm to acquisite the inlier set of the best fitted result and use the least square method to obtain the best fitted the plane.I**

I. Experimental Objectives

In this lab we want to implement the RANSAC Algorithm for plane fitting, and use it on a randomly generated point set with inliers and outliers of a plane we defined. The specific work is as follows:

A. Random 3D-points Set Generate

Here we need to randomly generate the set of data points to be fitted. According to the experimental objective, we need to put forward requirements for our point set. The set should reflect the function of RANSAC algorithm and its effect.

B. Inlier Set Acquisition with RANSAC

Here we implement RANSAC algorithm to obttain the inlier set. The algorithm need to be set as an adaptive procedure with the setting of hyper-parameters.

C. Fitted Plane Generation with LS

We then need to fit the plane-model on the inlier set we obtained through RANSAC with the way of least square method.

II. Process Analysis

A. Random 3D-points Set Generate

We must firstly construct the data set in order to build and apply our target model.

1) Generate Plane and Points on It: **We firstly Generate 900 points on a specific plane defined by ourselves. We can define the plane with the general formula of plane equation.**

$$Ax + By + Cz + D = 0 \tag{1}$$

So we set the parameter of a plane and then we can andomly define the 2-D coordinate of 900 points within [0,30] in both direction.

Generate their function value with the model parameter according to the definition of the plane. The effect is showed in figure 1.

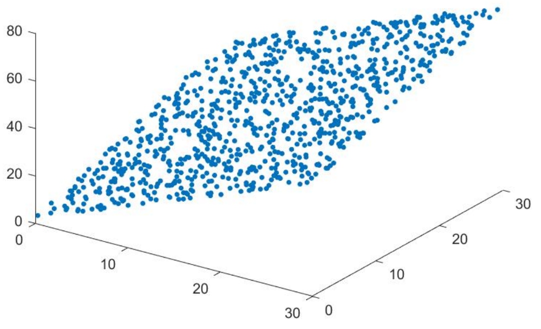


Fig. 1. Random Points Constrained by Plane

2) Generate Inlier Set: **Give all 900 points on the plane a white Gaussian noise value to get inliers. The Gaussian noise value is defined as:**

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} + \epsilon \begin{pmatrix} A \\ B \\ C \end{pmatrix} \tag{2}$$

ϵ is defined with gaussian white noise functio with a parameter represent the strength of noise which can be set with a wgn function in MATLAB.

Then the points will be shown in figure 2.

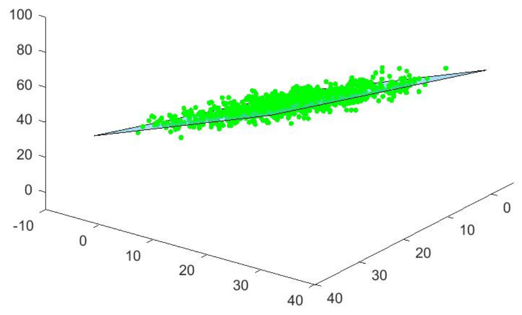


Fig. 2. Inliers

3) Generate Outlier Set: **Randomly generate 100 points within the domain space regarded as outliers.**

Then the points will be shown in figure 3.

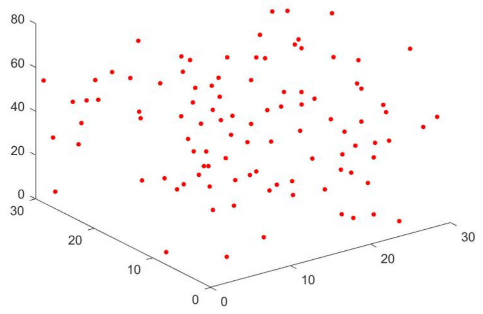


Fig. 3. Outliers

4) Generate Holistic point: **Obtain the point set combining the inlier set and outlier set.**

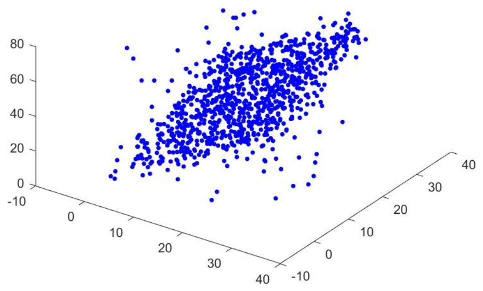


Fig. 4. Point Set

And then, we have obtained an application scenario of the RANSAC Algorithm.

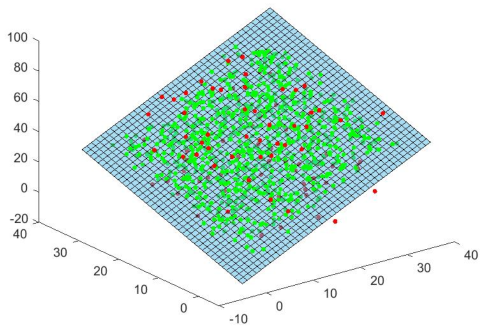


Fig. 5. Application Scenario Structure

B. RANSAC Algorithm

We need to designed a procedure using idea of RANSAC Algorithm to obtain the optimal inlier set fitting with the target plane.

1) Hyper Parameters of RANSAC: **Firstly, we have to define the hyper-parameters of RANSAC. And specificly, the hyper-parameters need to served for an adaptive procedure.**

- **s(Min-model-fit-set-size):**
  It represent the minimum nuber needed to fit the model.
  We can set it according to the model's complexity. obviously, in this question s = 3.
- **p(Not-all-lose-probability):**
  It represent the probability that in N iterative processes, we obtain a Min-model-fit-set whose points is all inliers in at least one iterative processes.
  So that, we know that '1-p' is the probability that in all of processes in N iterative , the point set we obtained has outliers.
  We can set it according to our own demand.
- **e(Outlier-ratio):**
  It represent probability that we get an outlier in one random selection.

$$e = \frac{numberOfOutliers}{numberOfPoints} = 1 - \frac{numberOfInliers}{numberOfPoints} \quad (3)$$

  Here we assume we do not know it previously, and will calculate it in iteration with the number of outliers of models we obtained every time.
- **N(Model-obtain-times/iterative times):**
  It represent the times of iterative processes(In one process, we randomly obtaind a model to judge if we accept it), also the times of iteration in the algorithm.
  So that we have the equation of N, e and p which can be used to calculate N in iteration:

$$(1-(1-e)^s)^N = 1-p \quad (4)$$

  We initial it with Infinity and it will be renewed adaptively by the equation above.
- **t(Inlier-threshold):**
  It represent the furthest distance weregard a point as an inlier.
  We can set it according to our own demand which is bounded by the gaussian noise strength.
- **d(Consensus-set-size-threshold):**
  It represent the least num of inliers for the judement wether we accept the model we randomly obtain.
  We can set it according to our own demand matched with e.

2) Store Structure: **It has the form which is an iteration of the model selection.**

Firstly we need to initial the structure we would use to judge in iteration.

- **Number of randomly selected models.**
- **Absolute error of the best model.**

3) RANSAC Algorithm: **We should then analyse the algorithm which is adaptive to the procedure.**

We need first know what the general RANSAC Algorithm is doing the fitting work.

So we can try to analyse its algorithm and write the pseudo code in our own opinion.

Let's design the Pseudo Code:

Algorithm 1 Adaptive RANSAC

```
1:  Initialization:
    $S \leftarrow 3$
    $P, t, d \leftarrow custom$
    $N \leftarrow \infty$
2:  Initialization:
    $sampleCount \leftarrow 0$
    $bestError \leftarrow \infty$
3:  while $N > sampleCount$ do
4:      $maybeInliers \leftarrow$ Randomly Choosed Smallest Set
5:      Obatain $maybeModel$ with $maybeInliers$
6:      for Every Point in Set do
7:          Calculate distance($point$, $maybeModel$)
8:          if distance($point$, $maybeModel$) $< t$ then
9:              Add $Point$ into $consensusSet$
10:         end if
11:     end for
12:     if $|consensusSet| > d$ then
13:         A beter model can be generate:
            Fit $betterModel$ with $consensusSet$
14:         $betterError \leftarrow$ Sum of Inlier-Model-Distance
15:         if $betterError < bestError$ then
16:             The best consensus set by now can be generate:
                $bestConsesusSet \leftarrow consensusSet$
17:             $bestError \leftarrow btterError$
18:             $bestInlierNum \leftarrow |consensusSet|$
19:             $e \leftarrow 1 - \frac{bestInlierNum}{pointNum}$
20:             The N can be renewed by e:
                $N \leftarrow \frac{\log(1-p)}{\log(1-(e)^s)}$
21:         end if
22:     end if
23:     $sampleCount \leftarrow sampleCount + 1$
24: end while
Output: $bestConsesusSet$ $bestError$ $bestInlierNum$
```

C. RANSAC Procedure

Then we can design procedure for the specific question.

1) Store Structure Initial: **Initial the structure we would use to judge in iteration:**

- **Number of randomly selected model**
- **Absolute error of the best model**

2) Enter Iteration: **With the condition of the while loop:**

$$N > sampleCount$$

3) Fit a maybe model: **Randomly choose 3 points in the point set.**

Fit the plane model with the 3 point using the analytic geometry method:
Calculate the vector oin two direction on the plane:

$$\begin{cases} \overrightarrow{v_{12}} = P_2 - P_1 \\ \overrightarrow{v_{13}} = P_3 - P_1 \end{cases} \tag{5}$$

So that, we can calculate the normal vector of the plane:

$$\overrightarrow{n} = \overrightarrow{v_{12}} \times \overrightarrow{v_{13}} \tag{6}$$

Then we can get the plane parameter:

$$\begin{cases} \overrightarrow{n} = (a, b, c) \\ d = -\overrightarrow{n} * P_1 \end{cases} \tag{7}$$

4) Distance to Maybe Model Plane: **Calculate the distance to judge the inliers with the analytic geometry method.**

We know that the point to line distance formula:

$$Distance(P_0, Plane) = \frac{|ax_0 + by_0 + cz_0 + d|}{\sqrt{a^2 + b^2 + c^2}} \tag{8}$$

Then we can calculate all the distance between from points and the maybe model plane

5) Count the inlier and outlier: **Count inlier number of the maybe model with the judgement of distance and the Inlier-threshold t.**

If the distance is less than the threshold t, it is the inlier of the maybe model.

6) Judgement of the Acceptment: **Judge whether we accept the model with inlier-number and Consensus-set-size-threshold.**

If the number of inlier is more than the threshold d, we accept the maybe model as a good model. And now, we need to judge if the model is better than the best model in the previous iteration.

7) Constract the Consensus Set: **We firstly construct the consensus set.**

8) Judgement of the Optimality: **We need to judge if it is a better model with a smaller error.**

To reach this target, we firstly calculate the absolute error with the sum of distance between inliers and the maybe model.

Then we judge if it is a better model than the previously best model. If it has a smaller error, it will be a better model.

9) Obtain the Iteration Result: **If the error is smaller, we regard the maybe model as the best model by now. So that we can obtain the result of this iteration: the items about the best model by now and rebew the store structure used in iteration.**

We have two result which might me the output:

- **The current consensus set which is the best by now**
- **The current inlier number which is the size of consensus set.**

And we also have the smaller error to renew the best error.

10) Renew the Iteration Time: **Define Outlier-ratio to renew N when N is larger than the iterations.**

$$e = 1 - \frac{bestInlierNumber}{pointNumber} \tag{9}$$

Calculate N with Rounding Function.

$$N = [\frac{\log(1-p)}{\log(1-(e)^s)}] \tag{10}$$

11) Increment SampleCount: **Increment samplecount by 1 to continue the loop.**

D. Fit the Best Model

Terminally, we can fit the best plane model with the result consensus set of RANSAC Algorithm using the least square method.

We assume the plane equation:

$$z = ax + by + cz \tag{11}$$

What we need do is to find (a,b,c) which can minimize the square:

$$E = \sum_{k=i}^{n=N_{inlier}} (ax + by + c - z)^2 \tag{12}$$

With the matrix form:

$$Y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \tag{13}$$

$$X = \begin{bmatrix} X_1 & 1 \\ \vdots & \vdots \\ X_n & 1 \end{bmatrix} \tag{14}$$

$$B = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{15}$$

So that, we have the represent the optimal function:

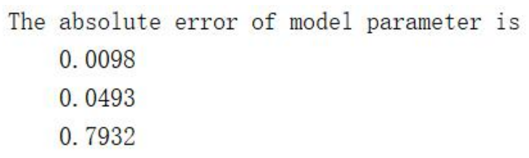
$$E = |Y - XB|^2 \tag{16}$$

Use the optimization method, we get the optimal parameter of the best model.

III. Results Description and Summary

A. Analysis of the Absolute Error

We use the absolute value of difference parameter between the best model and the actual model to describe error.



```
The absolute error of model parameter is
    0.0098
    0.0493
    0.7932
```

Fig. 6. Error Result

B. Result Visualization

With the result visualization, we can see the effect of fitting.

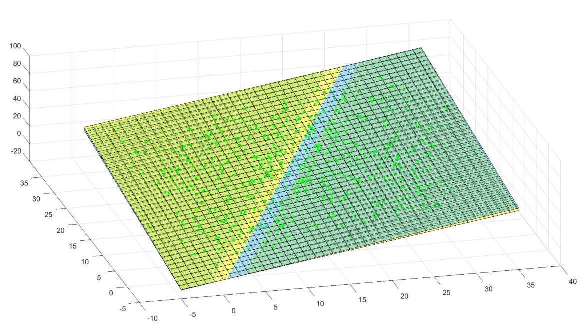


Fig. 7. Visual Result

C. Analysis of the Hyper-parameters

1) Relation between Hyper-parameters:

- **N is ralated to e and p as we analyse above.**
- **Consensus-set-size-threshold d and Inlier-threshold t offers a complementary function. I we have a larger d, we can set a smaller d, vice versa.**
- **Inlier-threshold t is not only related to Gaussian noise but also effect the probability for inlier.**
- **Consensus-set-size-threshold d should match expected inlier ratio 1-e.**

2) Effect of Hyper-parameters: **If we decrease the Consensus-set-size-threshold from 900 to 200 with the response decrease of the demand on Not-all-lose-probability p. The Frequency of bad result occurrence will increase sharply.**

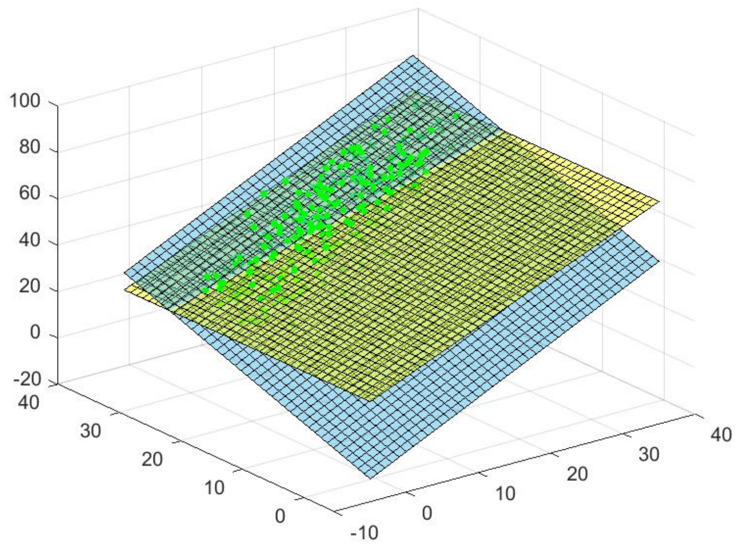


Fig. 8. An Example of the Bad Result

We will find that the stability of the result will be drastically reduced.

D. Summary

All in all, we have got a good good fitting performance using the RANSAC Algorithm with a good Hyper Parameter setting.

We can conclude that the merits and demerits of RANSAC.

Merits:

- **Simple to deneral**
- **Works well if the parameter setting is ok.**

demerits:

- **Lots of parameters to set.**
- **Can't always get good initialization because it need prior knowledge and later adjustment.**

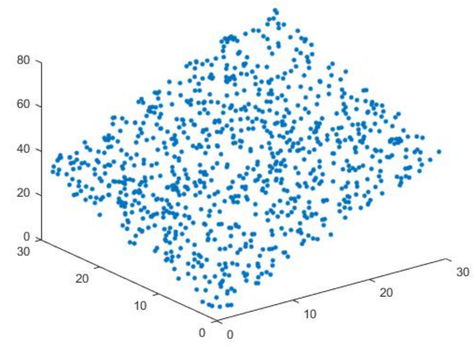
Though the experiment, I have a deeper understanding of RANSAC algorithm.

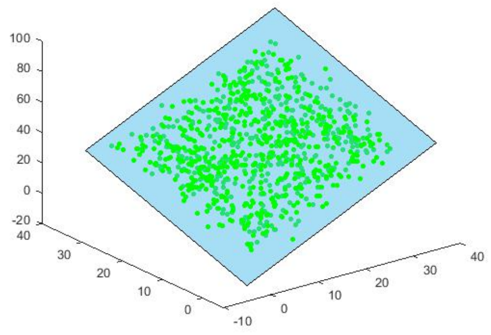
Ⅳ. SUPPORTING MARERIAL: MY CODE

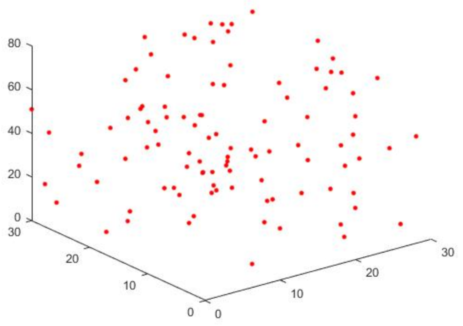
1.Randomly generate the set of data points to be fitted:

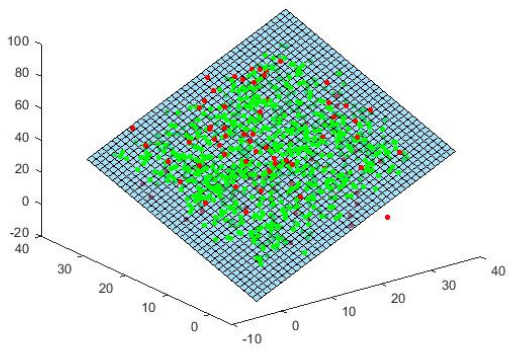
```
% 1.1.Generate 900 points on a specific plane defined by ourselves.

% 1) Define the parameter of the plane.

    actual_a = -3; actual_b = -2; actual_c = 2; actual_d = 5;

% 2) Randomly define the 2-D coordinate of 900 points within [0,30] in both direction .

    x = 0+(30-0)*rand(1,900);

    y = 0+(30-0)*rand(1,900);

% 3) Generate their function the model parameter according to the definition of the plane.

    z = -(actual_a/actual_c)*x - (actual_b/actual_c)*y + (actual_d/actual_c);

    actual_model = [-(actual_a/actual_c); -(actual_b/actual_c); actual_d/actual_c];

% 4) Show the points

    figure

    p0 = plot3(x,y,z,'.');

    p0.MarkerSize = 12;

% 1.2.Generate linear random points (inlier):

% 1) Give all 900 points a white Gaussian noise walue to get inliers matrix

    inliers = [x;y;z] + wgn(3,900,5); %sigma = 5

    % Another definition: inliers = [x;y;z] + [a;b;c]*wgn(1,900,0.1);

% 2) Show the inliers.

    figure

    p1_inlier = plot3(inliers(1,:),inliers(2,:),inliers(3,:),'.','Color','g');

    p1_inlier.MarkerSize = 12;

    hold on

    p1_x = -5:40:35;

    p1_y = p1_x;

    [p1_X,p1_Y] = meshgrid(p1_x);
    p1_Z = -(actual_a/actual_c)*p1_X - (actual_b/actual_c)*p1_Y + (actual_d/actual_c);

    p1_plane = surf(p1_X,p1_Y,p1_Z,'FaceAlpha',0.5);

    p1_plane.FaceColor = '#4DBEEE';

% 1.3.Generaye pure random points (outlier):

% 1) Randomly generate 100 points with in [-5,35] regarded as outliers.

    o_x = randi([0,30],1,100);

    o_y = randi([0,30],1,100);

    o_z = randi([3,77],1,100);

    outliers = [o_x;o_y;o_z];

% 2) Show the outliers:

    figure

    p1_outliers = plot3(outliers(1,:),outliers(2,:),outliers(3,:),'.','Color','r');

    p1_outliers.MarkerSize = 12;

% 3) Show the whole set model.

    figure

    p1_inlier = plot3(inliers(1,:),inliers(2,:),inliers(3,:),'.','Color','g');

    p1_inlier.MarkerSize = 12;


    hold on

    p1_x = -5:1:35;

    p1_y = p1_x;

    [p1_X,p1_Y] = meshgrid(p1_x);

    p1_Z = -(actual_a/actual_c)*p1_X - (actual_b/actual_c)*p1_Y + (actual_d/actual_c);

    p1_plane = surf(p1_X,p1_Y,p1_Z,'FaceAlpha',0.5);

    p1_plane.FaceColor = '#4DBEEE';


    hold on

    p1_outliers = plot3(outliers(1,:),outliers(2,:),outliers(3,:),'.','Color','r');

    p1_outliers.MarkerSize = 12;

% 1.4.Obtain the whole point set:
```

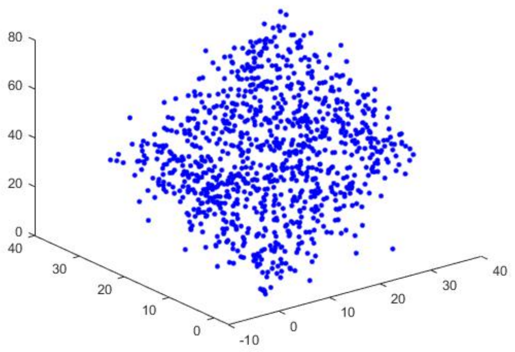
```
% 1) Merge the matrix of inliers and outliers.

   points = [inliers outliers];

% 2) Show the point set:

   figure

   p2 = plot3(points(1,:),points(2,:),points(3,:),'.','Color','b');

   p2.MarkerSize = 12;
```











2.Using RANSAC algorithm to obtain the inlier set:

```
% 2.1 Define the hyper-parameters of RANSAC(Adaptive procedure):
% 1)Min-model-fit-set-size: s
    % Min-model-fit-set-size: The minimum nuber needed to fit the model.
    s = 3; % According to the model's complexity.
% 2)Not-all-lose-probability: p
    % Not-all-lose-probability: The probability that in N iterative processes, we obtain
a Min-model-fit-set whose points is all inliers in at least one iterative processes.
    % 1-p: The probability that in all of processes in N iterative , the point set we obtained
has outliers.
    p = 0.99; % According to our own demand.
% 3)Outlier-ratio: e
    % Outlier-ratio: Probability that we get an outlier in one random selection.
    % e = number_of_outliers/number_of_points
    % Here we assume we do not know it previously, and will calculate it in iteration with
the number of outliers of models we obtained every time.
% 4)Model-obtain-times(iterative times): N
    % Model-sample-number(iterative times): The times of iterative processes(In one process,
we randomly obtaind a model to judge if we accept it), also the times of iteration in the
algorithm.
    % So that we have the equation of N, e and p which can be used to calculate N in iteration:
(1-(1-e)^s)^N = 1-p.
    N = Inf; % We initial it with Inf and it will be renewed adaptively .
% 5)Inlier-threshold: t:
    % Inlier-threshold: The furthest distance weregard a point as an inlier.
    t = 5; % According to our own demand.(According to the gaussian noise strength)
% 6)Consensus-set-size-threshold: d
    % Consensus-set-size-threshold: The least num of inliers for the judement wether we
```

```
accept the model we randomly obtain.

   d = 900; % According to our own demand.

%2.2 Iteration of the model selection:

% 1) Initial the structure we would use to judge in iteration.

% A.Number of randomly selected models:

   sample_count = 0;

% B.Absolute error of the best model:

   best_error = Inf;

   while(N > sample_count)

% 2) Randomly choose 3 points to fit a model

% A.Choose points:

      maybe_min_set = points(:,randi(1000,1,3));

% B.Fit model:

      v_p1_p2 = maybe_min_set(:,2) - maybe_min_set(:,1);

      v_p1_p3 = maybe_min_set(:,3) - maybe_min_set(:,1);

      v_n = cross(v_p1_p2, v_p1_p3);

      m_a = v_n(1);

      m_b = v_n(2);

      m_c = v_n(3);

      m_d = -m_a*maybe_min_set(1,1) - m_b*maybe_min_set(2,1) - m_c*maybe_min_set(3,1);

      maybe_model = [m_a, m_b, m_c, m_d];

% 3) Calculate the distance to judge the inliers

      distance_matrix = ( abs(maybe_model*[points;ones(1,1000)])/ sqrt(m_a*m_a + m_b*m_b

+m_c*m_c) );

% 4) Count inlier number of the maybe model with the judgement of distance and

Inlier-threshold.

      outlier_num = sum(distance_matrix > t);

      inlier_num = 1000 - outlier_num;

% 5) Judge whether we accept the model with inlier-number and Consensus-set-size-threshold.

         if(inlier_num >= d)% it is a model we can accept
```

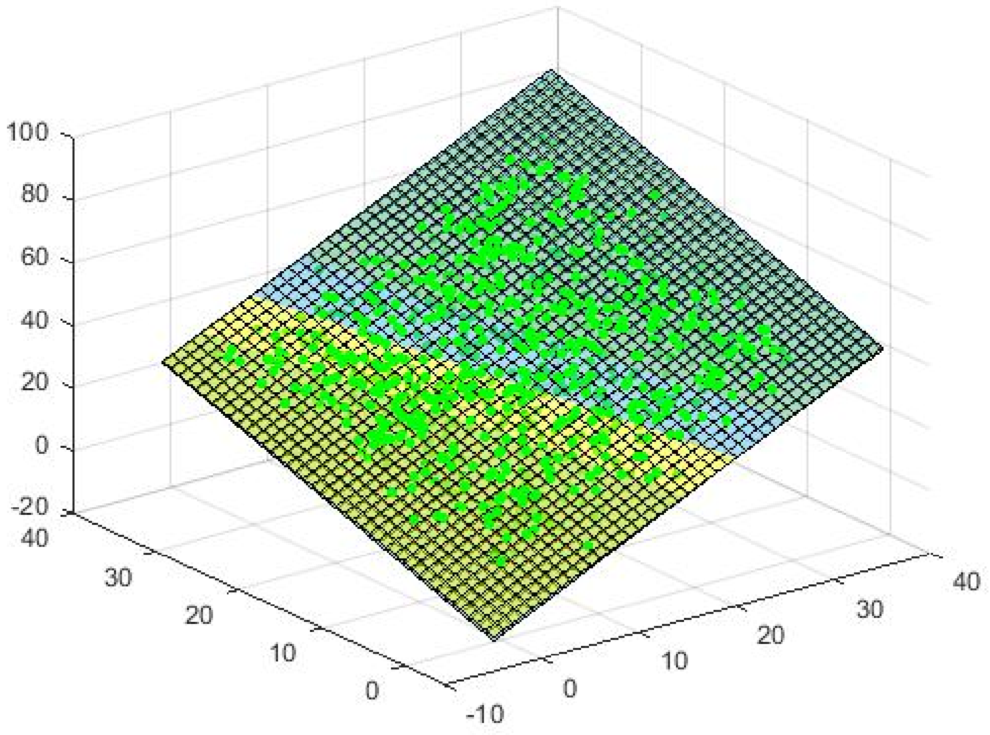
```
% 6) Constract the consensus set:

             consensus_set = repmat(((distance_matrix <= t)),3,1).*points;

             consensus_set(:,all(consensus_set==0,1)) = [];

% 7) To judge if it is a better model with a smaller error

% A. Calculate the absolute error with distance of inliers

             consensus_distance_matrix                                       =

( abs(maybe_model*[consensus_set;ones(1,inlier_num)])/ sqrt(m_a*m_a + m_b*m_b +m_c*m_c) );

             better_error = sum(abs(consensus_distance_matrix));

% C. Judge if it is the best model by now with error-judement-conditions.

             if(better_error < best_error)

% 8) Obtain the result of this iteration: the items about the best model by now.

% A.Its consensus set and inlier number which might be the output

                 best_consensus_set = consensus_set;

                 best_inlier_num = inlier_num;

% B.Its renewed best error which will be used in next iteration to do judgement.

                 best_error = better_error;

% 9) Define Outlier-ratio to renew N when N is larger than the iterations

                 e = 1 - best_inlier_num/1000;

                 N = ceil(log(1-p)/log(1-(1-e)^s));

                 disp('N is:');

                 disp(N);

             end

         end

%10) Increment sample_count by 1

         sample_count = sample_count+1;

         disp('Sample obatining time is:');

         disp(sample_count);

  end
```

```
N is:
    4

Sample obatining time is:
    1

Sample obatining time is:
    2

Sample obatining time is:
    3

Sample obatining time is:
    4
```

3.Fit the plane-model on the inlier set with the way of least square:

```
% 1) Use least square method to Fit the best-model with the best consensus set we obtaind through RANSAC.
   best_C = [best_consensus_set(1:2,:); ones(1,best_inlier_num)]';
   best_D = best_consensus_set(3,:)';
   best_model = best_C\best_D;
% 2) Calculate the absolute error and display it.
   para_absolute_error = abs(best_model - actual_model);
   fprintf('The absolute error of model parameter is\n');
   disp(para_absolute_error);
% 3) Result visualization
% A.Show the actual plane we want to fit.

   figure

   p4_x = -5:1:35;

   p4_y = p4_x;

   [p4_X,p4_Y] = meshgrid(p4_x);

   p4_Z = actual_model(1)*p4_X + actual_model(2)*p4_Y + actual_model(3);

   p4_plane = surf(p4_X,p4_Y,p4_Z,'FaceAlpha',0.5);

   p4_plane.FaceColor = '#4DBEEE';

% B.Show the best consensus points we used to fit the best model.

   hold on

   p3                                                                        =

plot3(best_consensus_set(1,:),best_consensus_set(2,:),best_consensus_set(3,:),'.','Col

or','g');

   p3.MarkerSize = 12;

% C.Show the best plane we fitted through RANSAC.

   hold on

   p6_x = -5:1:35;

   p6_y = p6_x;

   [p6_X,p6_Y] = meshgrid(p6_x);

   p6_Z = best_model(1)*p6_X + best_model(2)*p6_Y + best_model(3);

   p6_plane = surf(p6_X,p6_Y,p6_Z,'FaceAlpha',0.5);

   p6_plane.FaceColor = 'y';
```

```
The absolute error of model parameter is

   0.0500

   0.0197

   0.9057
```



58119125 蒋卓洋

Published with MATLAB® R2020b