16-720J: Homework 5
RANSAC and 3D Reconstruction

Luting Wang

November 30, 2015

1 Theory Questions

Question 1.1 (5pts)

The coordinate of projection point P_r is $[0,0,1]^T$, and P_l is $[0,0,1]^T$. According to ${P_r}^T F P_l = 0$,
$\begin{bmatrix}0 & 0 & 1\end{bmatrix}\begin{bmatrix}F_{11} & F_{12} & F_{13}\\ F_{21} & F_{22} & F_{23}\\ F_{31} & F_{32} & F_{33}\end{bmatrix}\begin{bmatrix}0\\0\\1\end{bmatrix} = \begin{bmatrix}F_{31} & F_{32} & F_{33}\end{bmatrix}\begin{bmatrix}0\\0\\1\end{bmatrix} = F_{33} = 0$

Question 1.2 (10 pts)

If the motion of the cameras is a pure translation parallel to x-axis, we can assume the two cameras are $P = K[I|0]$, $P' = K[I|t]$. And according to $F = [e']_\times K' R K^{-1}$, so F can be written as $F = [e^{'}]_\times K K^{-1} = [e^{'}]_\times$. If the camera translation is parallel to the x-axis, then $e' = [1,0,0]^T$. Therefore, $F = \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & -1\\ 0 & 1 & 0\end{bmatrix}$.
And from ${x^{'}}^T F x = 0$, we can obtain $y = y'$. And because the epipoles are at infinity, therefore the epopolar lines are parallel to the x-axis.

Question 1.3 (10 pts)

Suppose the coordinate of the point in the image and the point in the mirror image is $[a,b,1]^T$ and $-[a,b,1]^T$, according to ${P_r}^T F P_l = 0$, we have: $\begin{bmatrix}a & b & 1\end{bmatrix}\begin{bmatrix}F_{11} & F_{12} & F_{13}\\ F_{21} & F_{22} & F_{23}\\ F_{31} & F_{32} & F_{33}\end{bmatrix}\begin{bmatrix}-a\\-b\\-1\end{bmatrix} = -a^2F_{11} - b^2F_{22} - ab(F_{21}+F_{12}) - ac(F_{13}+F_{31}) - bc(F_{32}+F_{23}) - F_{33} = 0$ Then, we can get: $F_{11} = F_{22} = F_{33} = 0, F_{21} + F_{12} = F_{13} + F_{31} = F_{32} + F_{23} = 0$ Therefore, F is skew-symmetric.

Question 1.4 (10 pts)

1) Given the point p_2 the epipolar line l_2 passing through p_2 and the epipole e_2 can be written as $l_2 = [e_2]_\times p_2$. Since p_2 may be written as $p_2 = H p_1$, we have $l_2 = [e_2]_\times H p_1$.
2)No. Because if two cameras are only separated by a pure rotation, the optical centers of the cameras lenses will be identical. Therefore, there is no epipolar plane or epipolar points. So we can't explain epipolar line as before.

2 Implementation Questions

Fundamental matrix estimation

The Eight Point Algorithm

Figure 1 and Figure 2 are $F_{8,clean}$ and its corresponding visualization, while Figure 3 and Figure 4 are $F_{8,noise}$ and its corresponding visualization. The fundamental matrix of clean data is more accurate, as you can see from its epipolar line. Because it uses clean point correspondence to calculate, and the accuracy of fundamental matrix is sensitive to noise. The code is shown below.

-3.0021e-10	-3.0692e-08	2.7052e-05
6.0434e-08	-5.5517e-09	-3.6033e-04
-4.6576e-05	3.4853e-04	-0.0054

Figure 1: Fundamental Matrix of clean data with Eight Point Algorithm

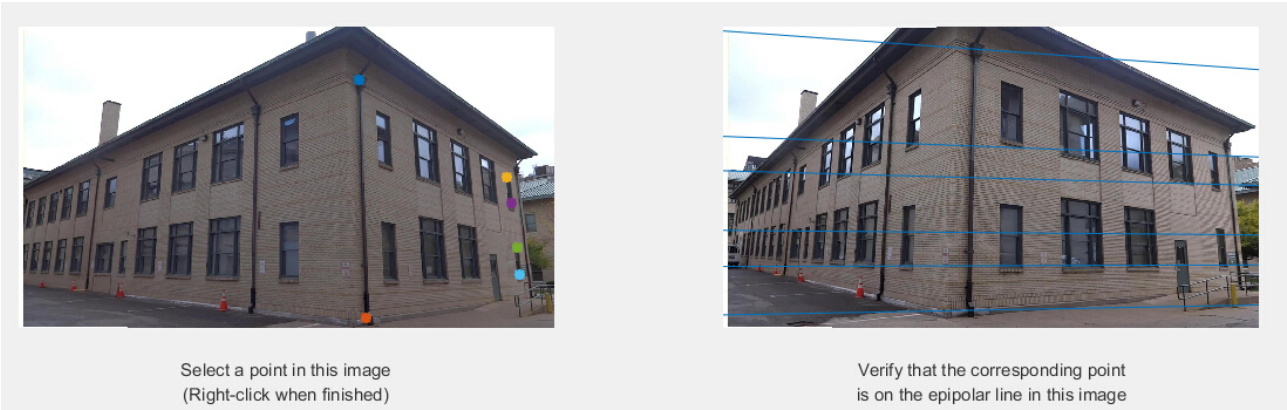

Figure 2: displayEpipolarF from clean data with Eight Point Algorithm

-1.5295e-08	6.1241e-09	1.4292e-06
5.7731e-10	-2.4491e-07	1.7252e-04
2.0963e-05	1.2775e-04	-0.0980

Figure 3: Fundamental Matrix of noisy data with Eight Point Algorithm

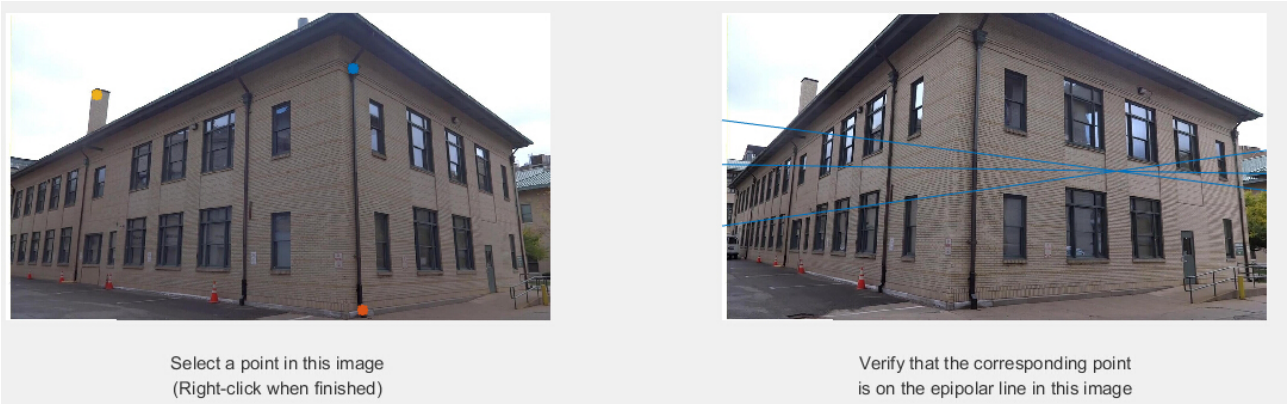

Figure 4: displayEpipolarF from noisy data with Eight Point Algorithm

```
function F = eightpoint_norm(pts1, pts2, normalization_constant)

N = size(pts1, 2);
T = eye(3)/normalization_constant;
T(3,3)=1;
pts1 = [pts1', ones(N, 1)]*T;
pts2 = [pts2', ones(N, 1)]*T;
x1 = pts1(:, 1);
y1 = pts1(:, 2);
x2 = pts2(:, 1);
y2 = pts2(:, 2);

A = [x1.*x2, y1.*x2, x2, x1.*y2, y1.*y2, y2, x1, y1, ones(N, 1)];
[¬,¬,V] = svd(A, 0);
F = reshape(V(:,9), 3, 3)';

[U, S, V,] = svd(F, 0);
F=U*diag([S(1,1) S(2,2) 0])*V';
F = T'*F*T;

end
```

The Seven Point Algorithm

Question 2.1 (15 pts)

There are three different fundamental matrix derived, as you can see in Figure 5, 6, 7. And from their corresponding visualization shown in Figure 8, 9, 10, we know the first one is correct. The code is shown below.

-3.9620e-09	-5.8941e-09	4.3314e-05
1.0013e-07	-1.0421e-08	-9.2763e-04
-9.8376e-05	8.9909e-04	-0.0203

Figure 5: Fundamental Matrix 1 of clean data with seven Point Algorithm

-2.0300e-08	1.8840e-07	-5.4223e-05
-1.2993e-07	-1.7406e-08	9.9422e-05
5.2403e-05	-8.3739e-05	-0.0070

Figure 6: Fundamental Matrix 2 of clean data with seven Point Algorithm

-2.1694e-08	2.0498e-07	-6.2546e-05
-1.4957e-07	-1.8002e-08	1.8707e-04
6.5269e-05	-1.6761e-04	-0.0058

Figure 7: Fundamental Matrix 3 of clean data with seven Point Algorithm

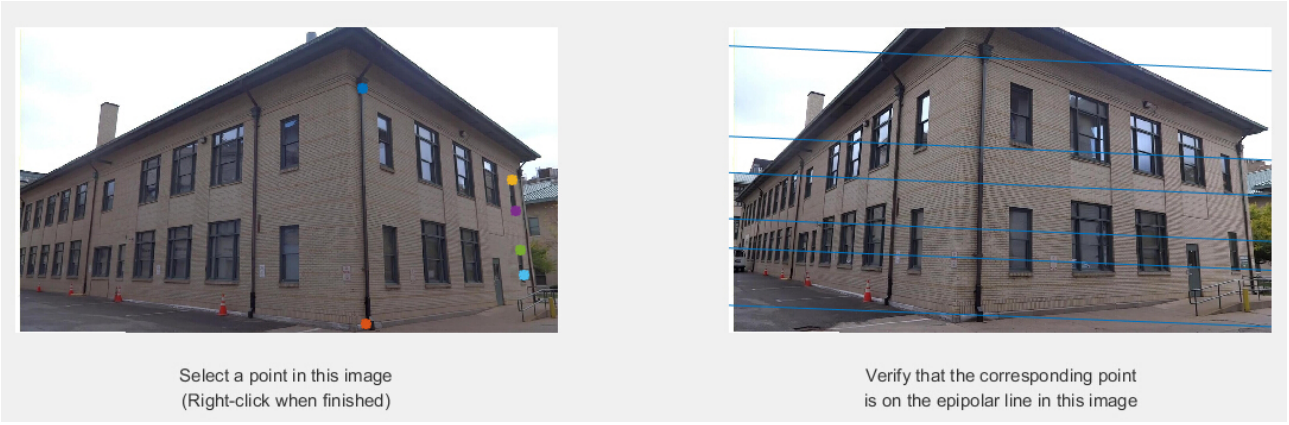

Figure 8: displayEpipolarF 1 from clean data with Seven Point Algorithm

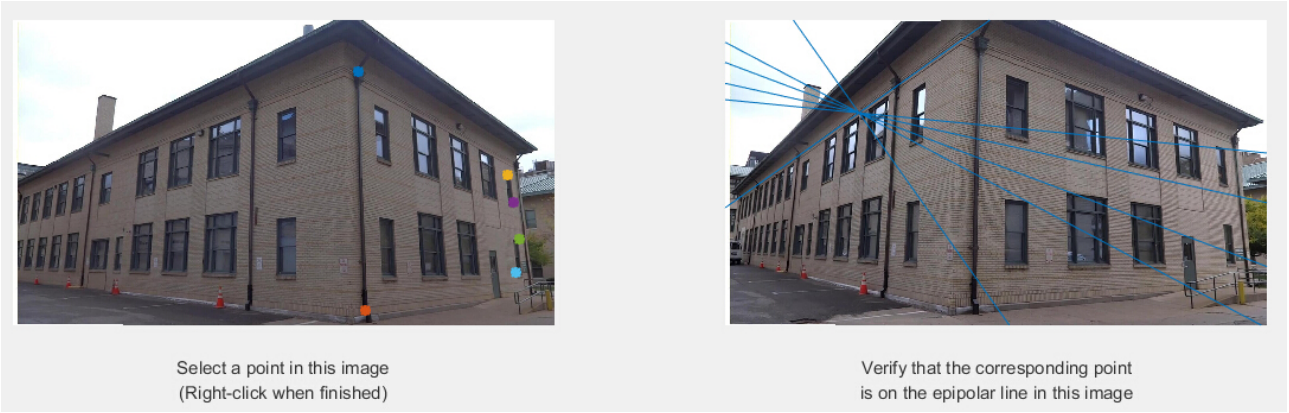

Figure 9: displayEpipolarF 2 from clean data with Seven Point Algorithm

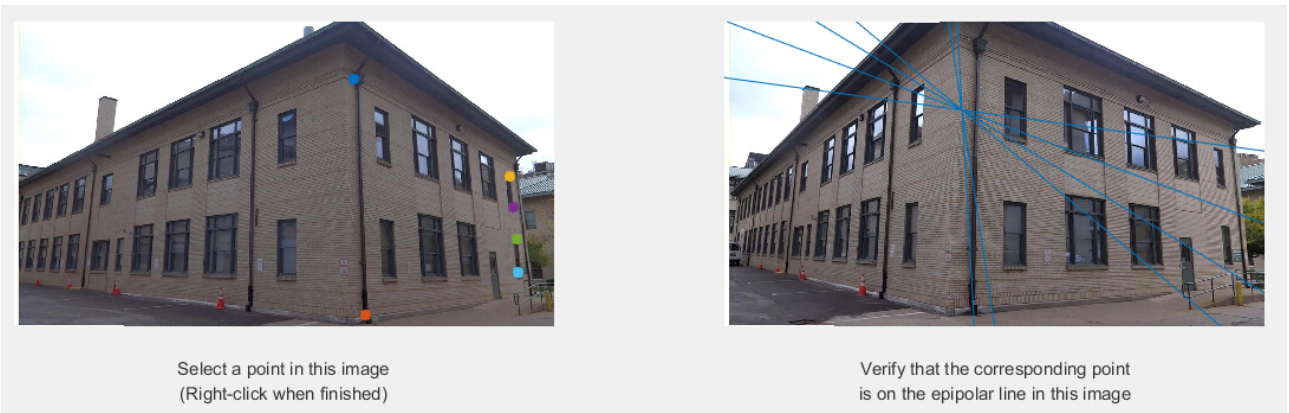

Figure 10: displayEpipolarF 3 from clean data with Seven Point Algorithm

```
function F = sevenpoint_norm(pts1, pts2, normalization_constant)

N = size(pts1, 2);
T = eye(3)/normalization_constant;
T(3,3)=1;
pts1 = [pts1', ones(N, 1)]*T;
pts2 = [pts2', ones(N, 1)]*T;
x1 = pts1(:, 1);
y1 = pts1(:, 2);
x2 = pts2(:, 1);
y2 = pts2(:, 2);
```

```
12
13 A = [x1.*x2, y1.*x2, x2, x1.*y2, y1.*y2, y2, x1, y1, ones(N, 1)];
14 [¬,¬,V] = svd(A);
15 f1 = reshape(V(:,8), 3, 3)';
16 f2 = reshape(V(:,9), 3, 3)';
17
18 syms lamda;
19 F0 = det(lamda*f1 + (1-lamda)*f2);
20 coef = sym2poly(F0);
21 lamda = roots(coef);
22
23 real_index = lamda == real(lamda);
24 lamda = lamda(real_index);
25 numOfF = length(lamda);
26 if(numOfF == 1)
27     F{1} = lamda*f1 + (1-lamda)*f2;
28     F{1} = T'*F{1}*T;
29 else
30     for j = 1:3
31         F{j} = lamda(j)*f1 + (1-lamda(j))*f2;
32         F{j} = T'*F{j}*T;
33     end
34 end
35
36 end
```

Computing F from Noisy Correspondences with RANSAC

Question 2.2 (10 pts)

RANSAC is helpful when there is some noise in the data set. As you can see from Figure 12, the epipolar line is reasonable after we apply RANSAC with seven point algorithm and obtain 41 cleaner point correspondence. Its fundamental matrix is shown in Figure 11.

-1.1462e-10	-1.3123e-08	1.1892e-05
4.3012e-08	-1.0322e-08	-3.3001e-04
-3.3798e-05	3.2753e-04	-0.0082

Figure 11: Fundamental Matrix of noisy data with RANSAC

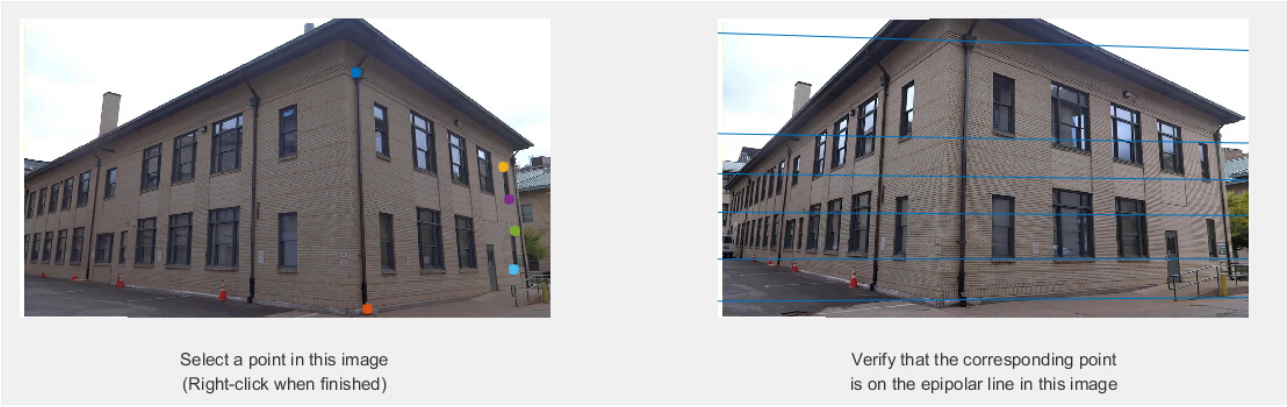

Figure 12: displayEpipolarF from noisy data with RANSAC

```
function [F, inliers] = ransacF(pts1, pts2, normalization_constant)

numOfConresponse = size(pts1,2);
th = 0.01;
m = 3000;
maxNumOfInliers = 0; %inilize

for j = 1:m
    temp = randperm(numOfConresponse);
    random_index = temp(1:7);
    selected_pts1 = pts1(:, random_index);
    selected_pts2 = pts2(:, random_index);

    temp_F = sevenpoint_norm(selected_pts1, selected_pts2, ...
       normalization_constant);

    numOfF = length(temp_F);
    for k = 1:numOfF
        temp_inliers = [];
        for i = 1 : numOfConresponse
            p1 = [pts1(:, i);1];
            p2 = [pts2(:, i);1];
            eline = temp_F{k}'*p2;
            dis = abs(p1'*temp_F{k}'*p2/(sqrt(sum(eline.^2))));
            if(dis < th)
                temp_inliers = [temp_inliers,i];
            end
        end
        numOfInliers = size(temp_inliers, 2);
        if(numOfInliers > maxNumOfInliers)
            maxNumOfInliers = numOfInliers;
            inliers = temp_inliers;
            F = temp_F{k};
        end
    end
end

end
```

Generating Novel Views of Smith Hall

Question 2.3 (30 pts)

To build 3D reconstruction from two views, first we should compute the fundamental matrix from point correspondences. Because they are selected manually, there is some noise inside. So first I use RANSAC to eliminate noise correspondence and obtain fundamental matrix with cleaner data. Then, with fundamental matrix and intrinsic camera parameters, we can compute camera matrices. After that, given two cameras matrices, I triangulate the set of inlier points and obtain 3D location of them. Since now I have a group of 3D points, and I know there are two obvious planes in the scene, we need to find these two main planes. To do so, I use RANSAC again. I randomly choose three points to obtain one plane, and calculate how many points are belong to this plane as inliers. This process is iterated thousands of times and finally the plane with maximum inliers are picked. Then with remaining points, I use RANSAC again and obtain the other plane, in order to eliminate some noise not belonging to both of these two main planes. After I get two planes, it's easy to create novel views from camera1's or camera2's

viewpoint, as shown in Figure 13, 14. To change the camera's viewpoint, I modify camera1' matrix (rotation and translation matrix) and get another novel view from a higher angle, as shown in Figure 15.

Figure 13: Novel view of Smith Hall from camera1's viewpoint

Figure 14: Novel view of Smith Hall from camera2's viewpoint

```
function genNovelView
    addpath(genpath('.'));
    load('data/K.mat'); %intrinsic parameters K
    i1 = imread('data/i1.jpg');
    i2 = imread('data/i2.jpg');
```

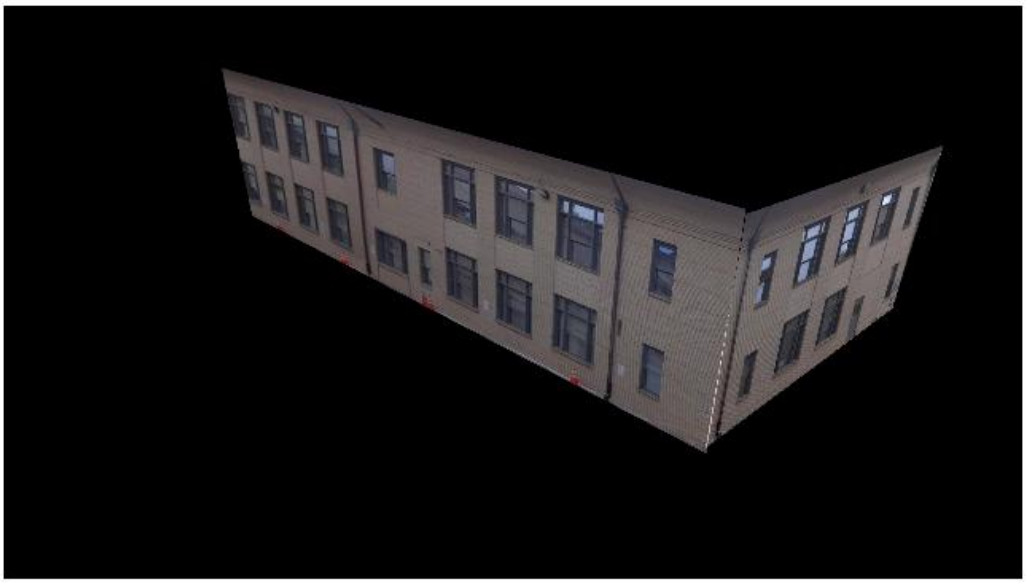

Figure 15: Novel view of Smith Hall from a higher angle

```
6       load('noisy_correspondences.mat');
7
8       normalization_constant = max(max(size(i1),size(i2)));
9       [F, inliers] = ransacF(pts1, pts2, normalization_constant);
10
11      pts1 = pts1(:, inliers);
12      pts2 = pts2(:, inliers);
13      M1 = K*eye(3, 4);
14      M2 = camera2(F, K, K, pts1, pts2);
15      P = triangulate(M1,pts1, M2, pts2);
16
17      th = 0.05;
18      [plane1,inliers1] = ransacF_plane(P, th);
19
20      P(:,inliers1) = [];
21      points_plane2 = P;
22      th = 0.03;
23      [plane2,inliers2] = ransacF_plane(points_plane2, th);
24
25      frame = drawNovelView(plane1', plane2', M1);      %novel view 1
26      imshow(frame);
27
28      theta = 40*pi/180;
29      R = [1, 0, 0; 0, cos(theta), -sin(theta); 0, sin(theta), cos(theta)];
30      t = [1;2;3];
31      M3 = K*[R, t];
32      frame = drawNovelView(plane1', plane2', M3);    %novel view 2
33      imshow(frame);
34
35
36
37  end
```

```
function [plane,new_inliers] = ransacF_plane(pts, th)

numOfPoints = size(pts,2);
m = 3000;
maxNumOfInliers = 0;
for j = 1:m
    temp = randperm(numOfPoints);
    random_index = temp(1:3);
    selected_pts = pts(:, random_index);

    temp_plane = genPlane(selected_pts(:,1)',  selected_pts(:,2)', ...
       selected_pts(:,3)');
    temp_inliers = [];

    for i = 1:numOfPoints
        point = pts(:, i)';
        dis =  abs(dot([point 1],temp_plane))/norm(temp_plane(1:3));
        if(dis < th)
             temp_inliers = [temp_inliers,i];
        end
    end

    numOfInliers = size(temp_inliers, 2);
    if(numOfInliers > maxNumOfInliers)
        maxNumOfInliers = numOfInliers;
        new_inliers = temp_inliers;
        plane = temp_plane;
    end
end

end
```

```
function plane = genPlane(P1, P2, P3)

    normal_line = cross(P2-P1, P3-P1);
    p = - dot(normal_line, P1);
    plane = [normal_line p];
    plane = plane/norm(plane);

end
```