Multi-Sensor Fusion & Tracking

Practical - 2:
4 Points Vs 2+1 Points

Masters in Computer Vision

UNIVERSITE DE BOURGOGNE

Centre Universitaire Condorcet - UB, Le Creusot

19 January 2018

Submitted By:
Mohit Kumar Ahuja

Submitted To:
Dr. Cedric Demonceaux

Index

1. Introduction

This Practical is the extension of practical-1 in which we performed the 8-Points Vs 5-Points Algorithm. In this practical we checked the different behaviors of 4-Points algorithm and 2+1 Point algorithm. We will start with a particular defined data and will analyze the computed homography by both the algorithms (4-Point and 2-Point). Then we will verify the resulted Homography matrix.

After that we will change the data and see the impact on the output of both algorithms. Then we will add some white noise to the data captured by camera-2 and analyse the effect on the output of both algorithms. Then we will add some white noise to data captured by camera-2 as well as we will add some noise to the IMU and analyse the effect.

As explained above, we have been assigned 4 tasks and according to each task, I made a specified file in matlab:

1. Practical2_TEST_1.m -: For a particular data.
2. Practical2_TEST_2.m -: Change of data.
3. Practical2_TEST_3.m -: Adding Noise to Camera-2 data.
4. Practical2_TEST_4.m -: Adding Noise to Camera-2 data and IMU data.

Preliminary Question: Read, Run and comment the file.

Solution: All the files have been fully commented and fully explained in the code.

2. Problem Statement

The goal of this practical is now to compare the classical linear 4 points algorithm and the linear 2 points knowing the vertical direction of the camera.

Let's consider 50 points randomly distributed in a plane of equation $N_T X_w + d = 0$ in the world frame (O_w, X_w, Y_w, Z_w).
Let's note respectively, ($O_{c1}, X_{c1}, Y_{c1}, Z_{c1}$) and ($O_{c2}, X_{c2}, Y_{c2}, Z_{c2}$) the camera positions.
We suppose a calibrated camera posed at a rotation R_i and T_i of the world coordinate ($X_w = R_i X_{ci} + T_i$).
The image points are noted P_i.

Preliminary questions

This practical is in relation with our first exercise "Homography estimation using IMU: 2+1 method". The algorithm is given in the _le Practical2.m.
Read, run and comment this file.

Comparison

1. Test 1: example with different data's, propose a test with different positions of the second camera ($R_1 = I$; $T_1 = 0$) with angles of rotation between 0_ and 45_ and translation of 0 to 100.

2. Test 2: example with noise, propose a test with different camera positions ($R_1 = I$, $T_1 = 0$) with angles of rotation between 0_ and 45_ and translation of 0 to 100 AND white noise in image points of camera 2 between 0 to 1 pixel std (use RANSAC functions).

3. Test 3 : example with noise on IMU information's, propose a test with different camera positions ($R_1 = I$; $T_1 = 0$) with angles of rotation between 0 and 45 and translation of 0 to 100, white noise in image points of camera 2 between 0 to 1 pixel std AND white noise in IMU between 0 to 2 (use RANSAC functions).

4. Conclusion...

{My Code Is well commented for better understanding of Viewer}

3. 4-Point Algorithm

As it has been mentioned in the problem statement that all the points lie on a same plane. So, we can compute the Homography using the following formula:

$$H = R - \frac{t * N^T}{d}$$

Where H = Homography, R = Rotation, t = Translation, N = Norm of the plane, d = Displacement of the ccamera from Norm of the plane.

$$P' = H * P$$

Where H = Homography, P' = Point in the second image and P = point in the first image.

This second equation can also be used for cross-verification of the computed Homography. If the result of the multiplication of H and P is equal to the ground truth of P', that means the computed homography is correct. The same method will be used for verification in the code.

This algorithm is being compared with another algorithm: 2+1 point and we had to compare the different behaviour of the output of same data when provided to this algorithm and when the same has been applied to the other algorithm.

4. 2-Point Algorithm

For 2+1 Point Algorithm, there is a Homography if the points belong on a vertical plane. And in our case, the points are on a vertical plane.

Homography for 2+1 Point algorithm is:

$$H = R_y + [t_x, t_y, t_z]^T [n_x, 0, n_z]$$

The 2-Point correspondence will give us 4-equations. Which makes the system Underdetermined. So, we will use SVD in this case to solve the problem.

$$Ah = b$$

$$A = UDV^T$$

$$h = Vy + wv \qquad v \quad \text{is the last column vector of } V$$

$$y = U^T b / D$$

$$det(H^T H - I) = 0 \Rightarrow w \quad \text{is solution of a 4th order polynomial}$$

This algorithm is being compared with another algorithm: 4 point and we had to compare the different behaviour of the output of same data when provided to this algorithm and when the same has been applied to the other algorithm.

5. Tests

We have been Given 4 Tests to perform and to compare the results of each test in accordance to 4-point algorithm and 2+1 point algorithm.

5.1 Test-1

In test-1 we were supposed to test the 4-Point algorithm with 2+1-Point algorithm with a particular defined data. The data is shown below:

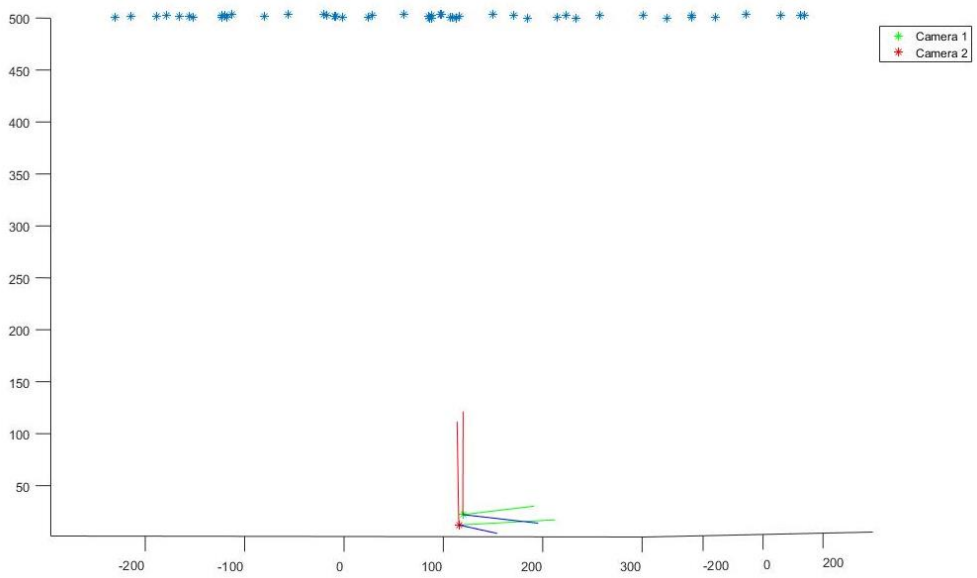

Figure 1: Data on a same Plane.

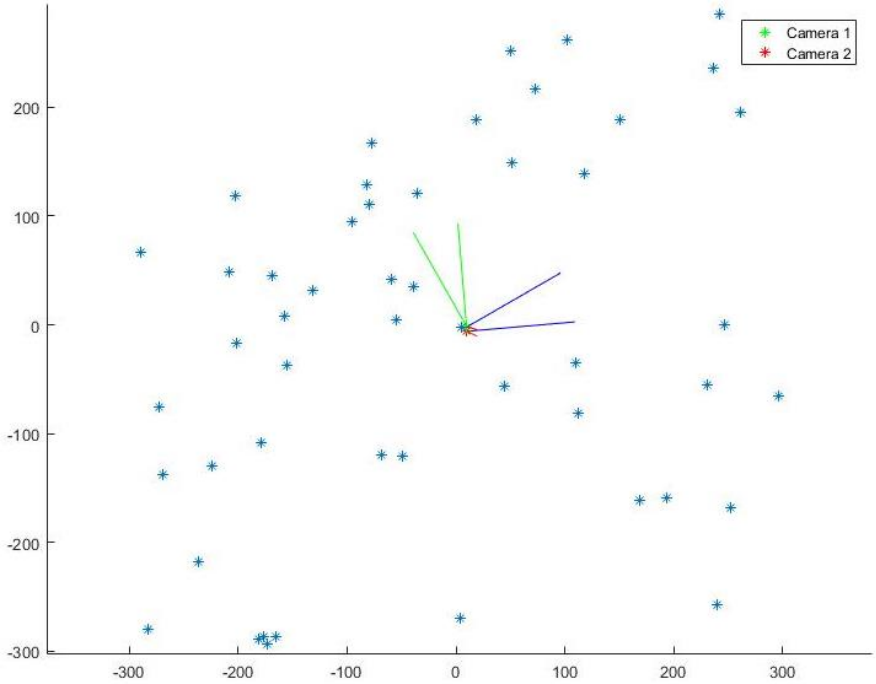

Figure 2: Two Camera's with Data

In Test-1, after defining the camera parameters, rotations and translations,

5.1.1 Using 4-Point Algorithm – Using Translation and Rotation matrix

We computed the Homography using formula:

```
H = Rt-Tt*(R1'*N)'/d;
```

And we got result:

$$H = \begin{matrix} 0.8896 & -0.4121 & 0.0270 \\ 0.4111 & 0.8907 & 0.0190 \\ -0.0313 & 0.0046 & 1.0000 \end{matrix}$$

Then we cross-verified the homography matrix using formula P' = H*P we got the same output:

```
HP1 =

    0.2553
   -0.1600
    1.0000

%% Ground Truth Should be = HP1 to validate the result %%
 Which Means, it is satisfying the equation P2(1,:) = H*P1(1,:)

GroundTruth =

    0.2553
   -0.1600
    1.0000
```

5.1.2 Using 4-Point Algorithm – Using "`homography2d`" Function

Now in this function we will compute the Homography using both Image Points (Image 1 and Image 2 points) using the SVD function to compute the homography using equation P' = H*P. And we got the result:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Homography estimation using Points in Two Images
%%%%% Using SVD,     P2(1,:) = H*P1(1,:)   %%%%%

H4pt =

    0.8896   -0.4121    0.0270
    0.4111    0.8907    0.0190
   -0.0313    0.0046    1.0000

You can see, the H and H4pt is exactly the same
```

5.1.3 Using 2-Point Algorithm – Using Translation and Rotation matrix

We computed homography using the translation and rotation matrix using this formula:

$$H = R_y + [t_x, t_y, t_z]^T [n_x, 0, n_z]$$

And we got the Homography Matrix as:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%% 2 Point Algorithm: %%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Homography estimation using the Translation and Rotation
%%%%%    Homography H in 2-Point Algo:  %%%%%

HV2 =

    0.9063   -0.4226    0.0007
    0.4226    0.9063    0.0083
         0         0    1.0208
```

For Validation, We got the output as;

```
 Validation of the Computed Homography:

H_Valid =

   -0.2193
   -0.3782
    0.8994

%% Ground Truth Should be = H_Valid to validate the result %%

GroundTruth =

   -0.2193
   -0.3782
    0.8994
```

5.1.4 Using 2-Point Algorithm – Using "`homography2d`" Function

Now in this function we will compute the Homography using both Image Points (Image 1 and Image 2 points) using the SVD function to compute the homography using equation P' = H*P. And we got the result:

```
%% True Homography : %%

TrueHomography =

    0.8878   -0.4140    0.0007
    0.4140    0.8878    0.0081
         0         0    1.0000

%% Here, The Estimated H should be = True Homography to validate our result

Estimated_H =

    0.8878   -0.4140    0.0007
    0.4140    0.8878    0.0081
         0         0    1.0000
```

Which is exactly the same.

Then we computed the Yaw Angle using formula shown below and validated it with the ground truth.

```
Yaw = atan2(H(2,1),H(1,1))
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Yaw Angle Computation:

Yaw =

    0.4363


YawGroundTruth =

    0.4363

 Both Yaw and YawGroundTruth should be Equal
```

These were the results of both the algorithms.

5.2 Test-2

In test-2 we were supposed to test the 4-Point algorithm with 2+1-Point algorithm with a different data. The data is shown below:

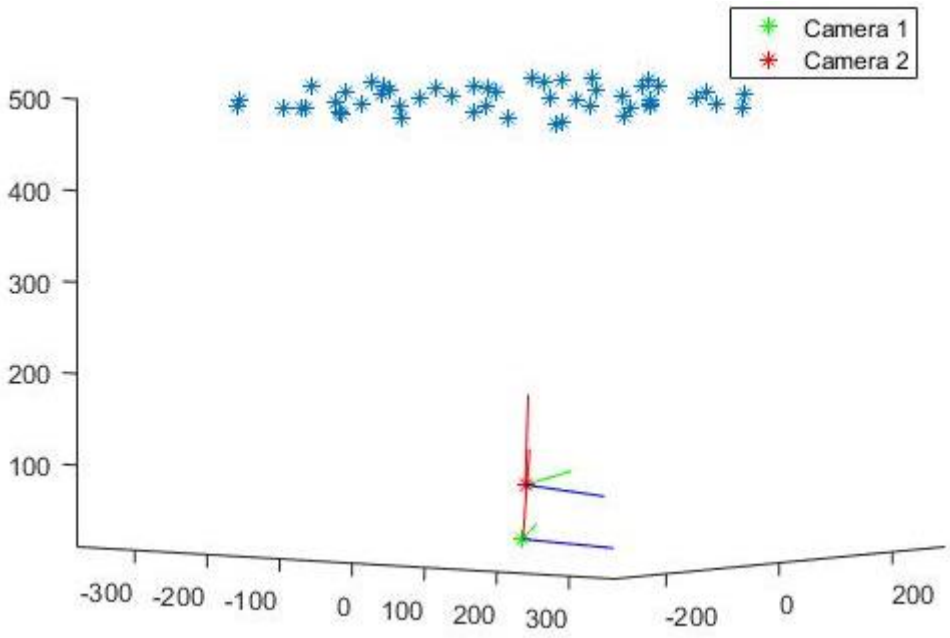

Figure 3: Data on a same Plane.

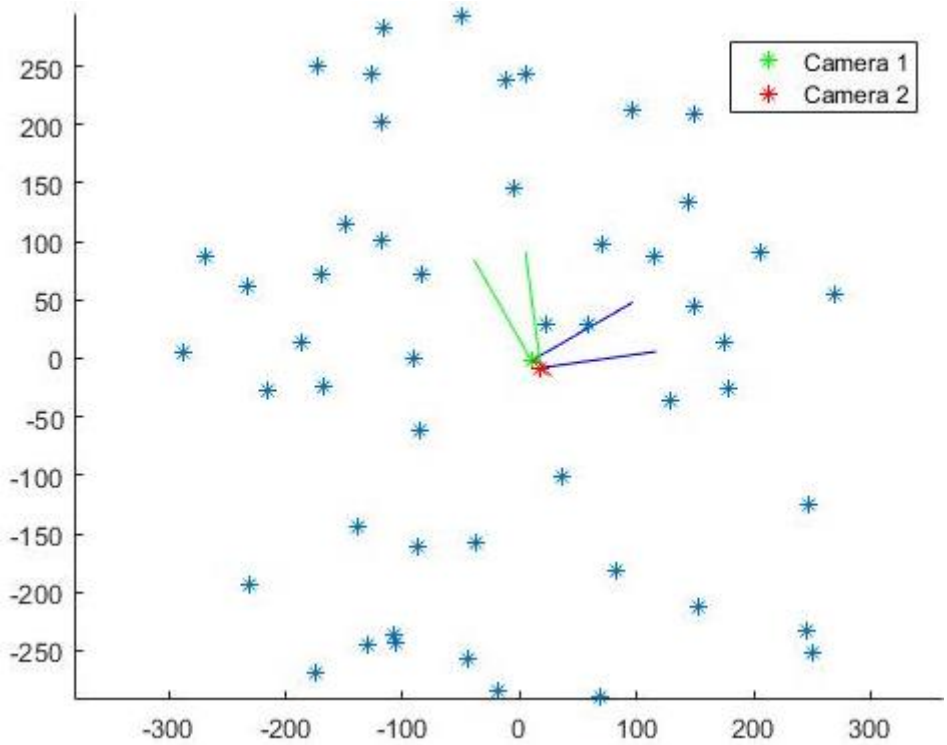

Figure 4: Two Camera's with Data

In Test-2, after defining the camera parameters, rotations and translations,

5.2.1 Using 4-Point Algorithm – Using Translation and Rotation matrix

We computed the Homography using formula:

```
H = Rt-Tt*(R1'*N)'/d;
```

And we got result:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Homography estimation using the Translation and Rotation
%%%%%    Homography H = Rt-Tt*(Rl*N)/d   :   %%%%%

H =

    1.0628   -0.4267    0.0257
    0.4247    1.0628    0.0488
   -0.0318   -0.0428    1.0000
```

Then we cross-verified the homography matrix using formula P' = H*P we got the same output:

```
HPl =

   -0.3104
   -0.4694
    1.0000

%% Ground Truth Should be = HPl to validate the result %%
 Which Means, it is satisfying the equation P2(1,:) = H*Pl(1,:)

GroundTruth =

   -0.3104
   -0.4694
    1.0000
```

5.2.2 Using 4-Point Algorithm – Using "`homography2d`" Function

Now in this function we will compute the Homography using both Image Points (Image 1 and Image 2 points) using the SVD function to compute the homography using equation P' = H*P. And we got the result:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Homography estimation using Points in Two Images
%%%%% Using SVD,     P2(1,:) = H*Pl(1,:)   %%%%%

H4pt =

    1.0628   -0.4267    0.0257
    0.4247    1.0628    0.0488
   -0.0318   -0.0428    1.0000

You can see, the H and H4pt is exactly the same
```

5.2.3 Using 2-Point Algorithm – Using Translation and Rotation matrix

We computed homography using the translation and rotation matrix using this formula:

$$H = R_y + [t_x, t_y, t_z]^T [n_x, 0, n_z]$$

And we got the Homography Matrix as:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%% 2 Point Algorithm: %%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Homography estimation using the Translation and Rotation
%%%%%    Homography H in 2-Point Algo:  %%%%%

HV2 =

    0.9272   -0.3746   -0.0148
    0.3746    0.9272    0.0147
         0         0    0.8750
```

For Validation, We got the output as;

```
 Validation of the Computed Homography:

H_Valid =

    0.2263
    0.4322
    0.8729

%% Ground Truth Should be = H_Valid to validate the result %%

GroundTruth =

    0.2263
    0.4322
    0.8729
```

5.2.4 Using 2-Point Algorithm – Using "`homography2d`" Function

Now in this function we will compute the Homography using both Image Points (Image 1 and Image 2 points) using the SVD function to compute the homography using equation P' = H*P. And we got the result:

```
%% True Homography : %%

TrueHomography =

    1.0596   -0.4281   -0.0169
    0.4281    1.0596    0.0168
         0         0    1.0000

%% Here, The Estimated H should be = True Homography to validate our resul

Estimated_H =

    1.0596   -0.4281   -0.0169
    0.4281    1.0596    0.0168
         0         0    1.0000
```

Which is exactly the same.

Then we computed the Yaw Angle using formula shown below and validated it with the ground truth.

```
Yaw = atan2(H(2,1),H(1,1))
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Yaw Angle Computation:

Yaw =

    0.3840

YawGroundTruth =

    0.3840

 Both Yaw and YawGroundTruth should be Equal
```

These were the results of both the algorithms.

5.3 Test-3

In test-3 we were supposed to test the 4-Point algorithm with 2+1-Point algorithm with a different data by adding noise to the points of Camera-2. And then Computing everything again and analyzing the difference between both Algorithms output.

In Test-3, after defining the camera parameters, rotations and translations,

For, Camera-2 we added some noise to the data:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% ADDING WHITE NOISE TO THE POINTS OF SECOND CAMERA %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                    P2 = P2 + rand(size(P2));
```

After Adding Noise, We computed everything again and got these results:

5.3.1 Using 4-Point Algorithm – Using Translation and Rotation matrix

We computed the Homography using formula:

```
H = Rt-Tt*(R1'*N)'/d;
```

And we got result:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Homography estimation using the Translation and Rotation
%%%%%    Homography H = Rt-Tt*(R1*N)/d  :  %%%%%

H =

    1.0628   -0.4267    0.0257
    0.4247    1.0628    0.0488
   -0.0318   -0.0428    1.0000
```

Then we cross-verified the homography matrix using formula P’ = H*P we got the output:

```
HP1 =

   -0.7167
   -0.2890
    1.0000

%% Ground Truth is not =  HP1 to validate the result %%
 Which Means, it is not satisfying the equation P2(1,:) = H*P1(1,:)

GroundTruth =

   -0.7136
   -0.2876
    1.0000
```

As we can see, the results are not matching with the ground truth. Though it is very near to the ground truth but not the same because of addition of noise.

5.3.2 Using 4-Point Algorithm – Using “`homography2d`” Function

Now in this function we will compute the Homography using both Image Points (Image 1 and Image 2 points) using the SVD function to compute the homography using equation P' = H*P. And we got the result:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Homography estimation using Points in Two Images
%%%%% Using SVD,     P2(1,:) = H*P1(1,:)   %%%%%

H4pt =

    1.0610   -0.4258    0.0268
    0.4239    1.0616    0.0499
   -0.0315   -0.0426    1.0000

%%% You can see, the H and H4pt is not exactly the same %%%
The small error is due to the Noise Added to the points of second Camera
```

5.3.3 Using 2-Point Algorithm – Using Translation and Rotation matrix

We computed homography using the translation and rotation matrix using this formula:

$$H = R_y + [t_x, t_y, t_z]^T [n_x, 0, n_z]$$

And we got the Homography Matrix as:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%% 2 Point Algorithm: %%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Homography estimation using the Translation and Rotation
%%%%%    Homography H in 2-Point Algo:  %%%%%

HV2 =

    0.9272   -0.3746   -0.0148
    0.3746    0.9272    0.0147
         0         0    0.8750
```

For Validation, We got the output as;

```
 Validation of the Computed Homography:

H_Valid =

   -0.2193
   -0.4890
    0.8443

%% Ground Truth is not = H_Valid to validate the result %%
The small error is due to the Noise Added to the points of second Camera


GroundTruth =

   -0.2172
   -0.4876
    0.8457
```

The results are not matching because of the noise we added to the data.

5.3.4 Using 2-Point Algorithm – Using "`homography2d`" Function

Now in this function we will compute the Homography using both Image Points (Image 1 and Image 2 points) using the SVD function to compute the homography using equation P' = H*P. And we got the result:

```
%% True Homography : %%

TrueHomography =

    1.0596   -0.4281   -0.0169
    0.4281    1.0596    0.0168
         0         0    1.0000

%% Here, The Estimated H should be = True Homography to validate our result
But The small error is due to the Noise Added to the points of second Camera

Estimated_H =

    1.0582   -0.4274   -0.0156
    0.4274    1.0582    0.0178
         0         0    1.0000
```

Which is not exactly the same because of noise, there are some errors in the output.

Then we computed the Yaw Angle using formula shown below and validated it with the ground truth.

```
Yaw = atan2(H(2,1),H(1,1))
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Yaw Angle Computation:

Yaw =

    0.3839


YawGroundTruth =

    0.3840
```

These were the results of both the algorithms.

The results shows the difference between the original values and the Computed ones. The error can be reduced by using RANSAC algorithm and it will give us near to the original homography matrix.

5.4 Test-4

In test-4 we were supposed to test the 4-Point algorithm with 2+1-Point algorithm with a different data by adding noise to the points of Camera-2 and by adding noise to IMU. And then Computing everything again and analyzing the difference between both Algorithms output.

In Test-4, after defining the camera parameters, rotations and translations,

For IMU of camer-2 we added some noise:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% ADDING WHITE NOISE TO THE IMU OF SECOND CAMERA %%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Rotation angles in degree
roll2 = roll1 + 2*rand();
pitch2 = pitch1 + 2*rand();
yaw2 = yaw1 + 2*rand();
```

By doing this, we added 0-2 Degree of noise to each of the angle.

And for Camera-2 we added some noise to the data:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% ADDING WHITE NOISE TO THE POINTS OF SECOND CAMERA %%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                    P2 = P2 + rand(size(P2));
```

After Adding Noise, We computed everything again and got these results:

5.4.1 Using 4-Point Algorithm – Using Translation and Rotation matrix

We computed the Homography using formula:

```
H = Rt-Tt*(R1'*N)'/d;
```

And we got result:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Homography estimation using the Translation and Rotation
%%%%%    Homography H = Rt-Tt*(R1*N)/d  :  %%%%%

H =

    1.1441   -0.0002   -0.0029
    0.0004    1.1442    0.0425
    0.0195   -0.0505    1.0000
```

Then we cross-verified the homography matrix using formula P' = H*P we got the output:

```
HP1 =

   -0.4730
    0.2300
    1.0000

%% Ground Truth is not =  HP1 to validate the result %%
 Which Means, it is not satisfying the equation P2(1,:) = H*P1(1,:)

GroundTruth =

   -0.4706
    0.2319
    1.0000
```

As we can see, the results are not matching with the ground truth. Though it is very near to the ground truth but not the same because of addition of noise.

5.4.2 Using 4-Point Algorithm – Using "`homography2d`" Function

Now in this function we will compute the Homography using both Image Points (Image 1 and Image 2 points) using the SVD function to compute the homography using equation P' = H*P. And we got the result:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Homography estimation using Points in Two Images
%%%%% Using SVD,     P2(1,:) = H*P1(1,:)   %%%%%

H4pt =

    1.1429   -0.0002   -0.0017
    0.0005    1.1435    0.0437
    0.0194   -0.0498    1.0000

%%% You can see, the H and H4pt is not exactly the same %%%
The small error is due to the Noise Added to the points of second Camera
```

5.4.3 Using 2-Point Algorithm – Using Translation and Rotation matrix

We computed homography using the translation and rotation matrix using this formula:

$$H = R_y + [t_x, t_y, t_z]^T [n_x, 0, n_z]$$

And we got the Homography Matrix as:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%% 2 Point Algorithm: %%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Homography estimation using the Translation and Rotation
%%%%%    Homography H in 2-Point Algo:  %%%%%

HV2 =

    1.0000    0.0000   -0.0082
   -0.0000    1.0000    0.0192
         0         0    0.8750
```

For Validation, We got the output as;

```
 Validation of the Computed Homography:

H_Valid =

   -0.4308
   -0.1037
    0.8965

%% Ground Truth is not = H_Valid to validate the result %%
The small error is due to the Noise Added to the points of second Camera


GroundTruth =

   -0.4290
   -0.1037
    0.8973
```

The results are not matching because of the noise we added to the data.

5.4.4 Using 2-Point Algorithm – Using "`homography2d`" Function

Now in this function we will compute the Homography using both Image Points (Image 1 and Image 2 points) using the SVD function to compute the homography using equation P' = H*P. And we got the result:

```
%% True Homography : %%

TrueHomography =

    1.1429    0.0000   -0.0094
   -0.0000    1.1429    0.0219
         0         0    1.0000

%% Here, The Estimated H should be = True Homography to validate our result %
But The small error is due to the Noise Added to the points of second Camera

Estimated_H =

    1.1418   -0.0001   -0.0080
    0.0001    1.1418    0.0230
         0         0    1.0000
```

Which is not exactly the same because of noise, there are some errors in the output.

Then we computed the Yaw Angle using formula shown below and validated it with the ground truth.

```
Yaw = atan2(H(2,1),H(1,1))
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Yaw Angle Computation:

Yaw =

   1.2721e-04


YawGroundTruth =

  -3.1576e-05

%%%%% Both Yaw and YawGroundTruth should be Equal %%%%%
But, The small error is due to the Noise Added to the points of second Camer
```

These were the results of both the algorithms.

The results shows the difference between the original values and the Computed ones. The error can be reduced by using RANSAC algorithm and it will give us near to the original homography matrix.

6. Conclusion

In this Practical, we learned the difference between two different algorithms which are 4 point and 2+1 point algorithm. We tried both algorithms with different data and by adding noise to the data as well as IMU, we saw that even with noisy data the computed homograhy is quite near to the original one.

And if we use RANSAC, even if the data is noisy we will get better results than computing Homography without using RANSAC.

Also, if we add noise to many parameters like IMU and Data, the effect on the output will start expanding. But the same can be reduced by using RANSAC algorithm.