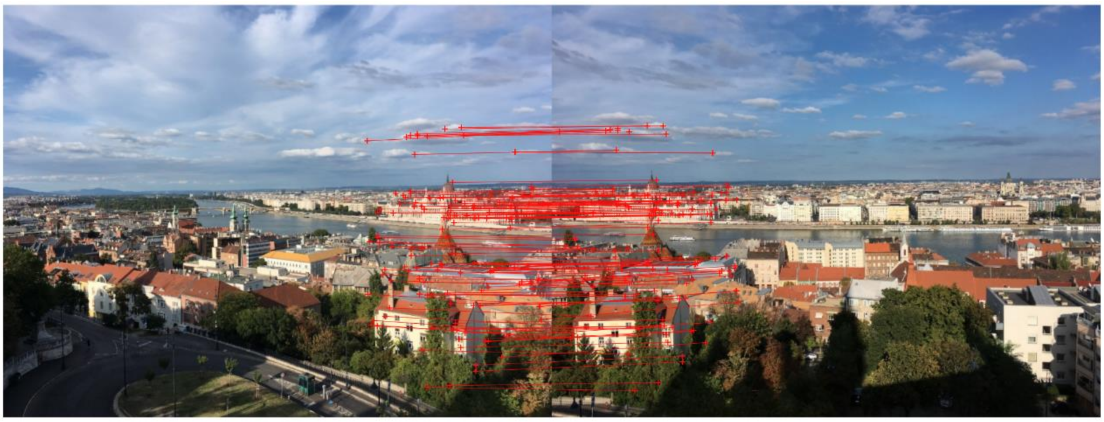
Q1:

3.

Putative matches: I selected all pairs of key features whose spatial distance is below the 0.03 * max(spatial distance), here's the result.



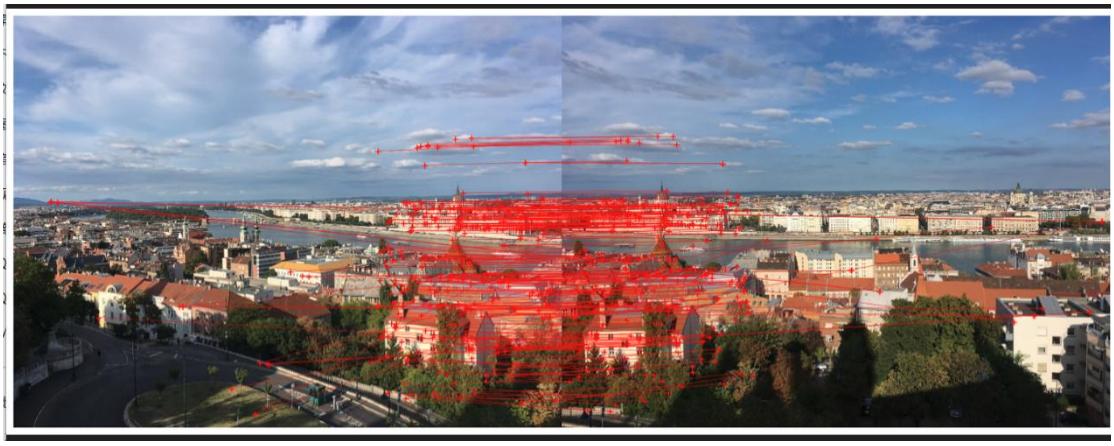
4.

The RANSAC algorithm here is used to find the best homography matrix that maps correspondence between two sets of the points. The hyperparameters used in this implementation are:

1. distance_threshold: The maximum distance between a correspondence and its projection under the computed homography matrix for it to be considered an inlier. The default value is 2.
2. outlier_ratio: The maximum fraction of correspondences that are allowed to be outliers. The default value is 0.2.
3. p: The probability of finding the correct model given a set of correspondences. The default value is 0.99.
4. s: The size of the random subset of correspondences that is selected for each RANSAC iteration. The default value is 4.
5. e: The maximum fraction of correspondences that are allowed to be outliers. The default value is 0.2.

The algorithm uses a loop that runs for N iterations, where N is calculated based on the values of p, s, and e. At each iteration, s random correspondences are selected from the input data and a homography matrix is computed using those correspondences. The homography matrix is then used to project all the correspondences and the distance between the projected points and the actual points is computed. Correspondences that are within the distance threshold are considered inliers, while the rest are considered outliers. If the number of inliers is greater than the number of inliers found in previous iterations, the homography matrix, inliers, and residuals are updated accordingly. The loop terminates early if the number of outliers is less than the specified outlier ratio.

The average residual is 0.753, and the number of inliers is 144.



5. This is the stitched image:

Q2:

1.

The residuals for both unnormalized/normalized method:

method	Residual unnormalized method	Residual normalized method
Residual 1	0.14912309939434457	0.05484509861965783
Residual 2	0.17921336681147462	0.060334458479578496
Combined	0.1641682331029096	0.057589778549618165

Here’s the 3*3 fundamental matrix for both of the methods:

Fundamental matrix Unnormalized:

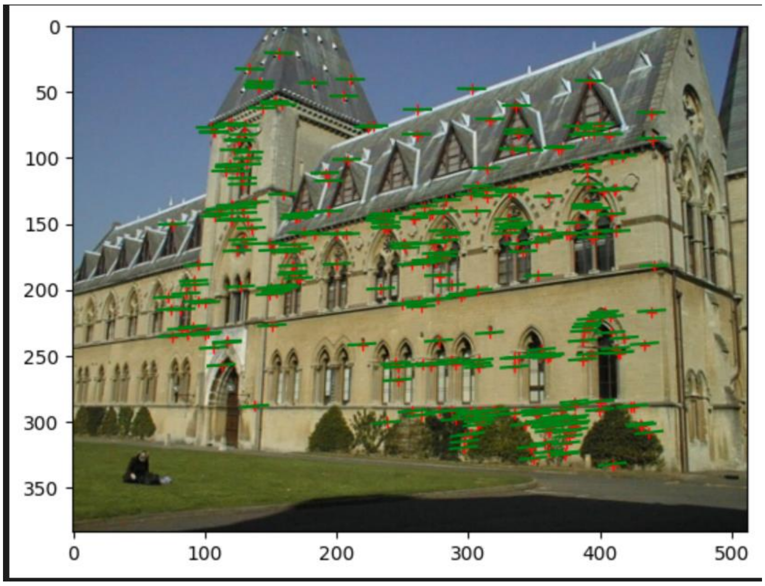
[[1.32546895e-06 -1.36852466e-05 6.83862987e-04]
[2.88625175e-05 -2.66854091e-07 -4.09703775e-02]
[-5.63235250e-03 3.73349826e-02 1.00000000e+00]]

Fundamental matrix Normalized:

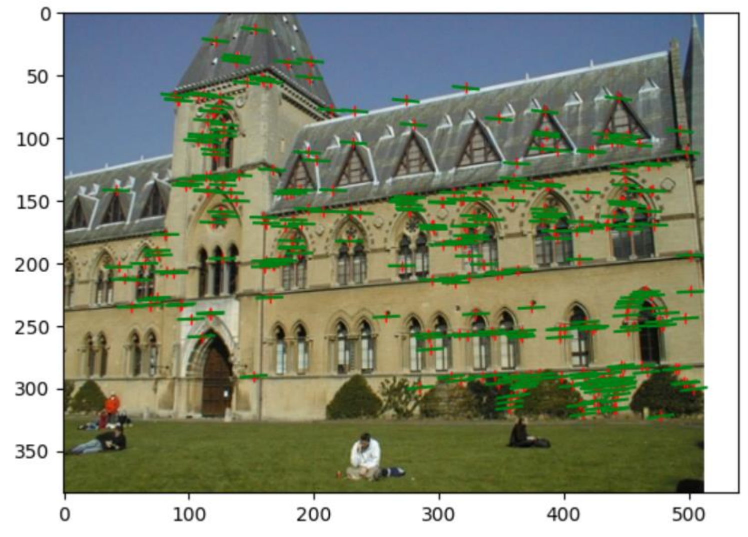
[[1.86520131e-07 -3.82943867e-06 5.65229482e-04]
[2.22767397e-05 2.16768623e-07 -4.11451100e-02]
[-5.30028152e-03 3.69650848e-02 1.00000000e+00]]

Visualizations:

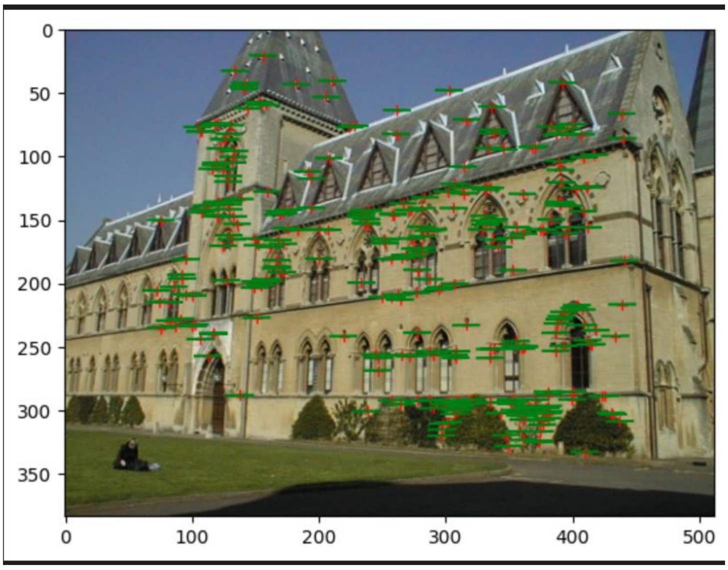
Unnormalized on I2:



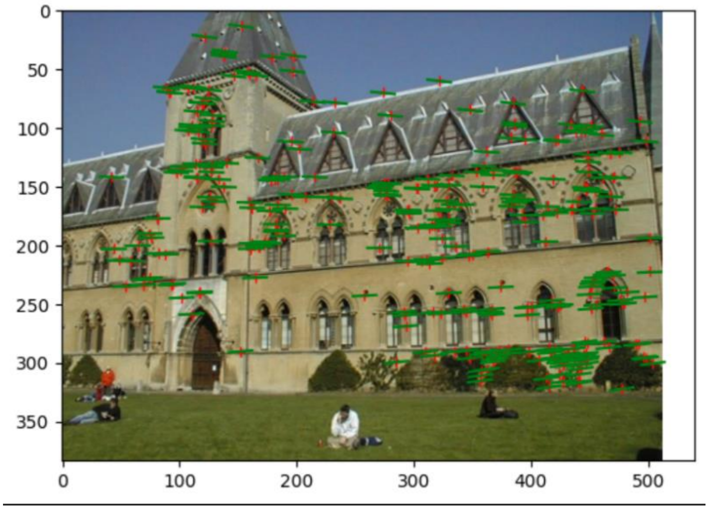
Unnormalized on I1



Normalized on I2:



Normalized on I1:



```
Fundamental Unnormalized
[[ 1.32546895e-06 -1.36852466e-05  6.83862987e-04]
 [ 2.88625175e-05 -2.66854091e-07 -4.09703775e-02]
 [-5.63235250e-03  3.73349826e-02  1.00000000e+00]]
library: residual in frame 2 (non-normalized method) =  0.17921336681147462
library: residual in frame 1 (non-normalized method) =  0.14912309939434457
library: residual combined   (non-normalized method) =  0.1641682331029096
Fundamental Normalized
[[ 1.86520131e-07 -3.82943867e-06  5.65229482e-04]
 [ 2.22767397e-05  2.16768623e-07 -4.11451100e-02]
 [-5.30028152e-03  3.69650848e-02  1.00000000e+00]]
library: residual in frame 2 (normalized method) =  0.060334458479578496
library: residual in frame 1 (normalized method) =  0.05484509861965783
library: residual combined   (normalized method) =  0.057589778549618165
```

2.

lab 1 camera projection
[[-2.33260962e+00 -1.10025080e-01 3.37513233e-01 7.36686567e+02]
 [-2.31044166e-01 -4.79515070e-01 2.08722206e+00 1.53627263e+02]
 [-1.26377057e-03 -2.06774255e-03 5.14712340e-04 1.00000000e+00]]

lab 2 camera projection
[[6.93154686e-03 -4.01684470e-03 -1.32602928e-03 -8.26700554e-01]
 [1.54768732e-03 1.02452760e-03 -7.27440714e-03 -5.62523256e-01]
 [7.60946050e-06 3.70953989e-06 -1.90203244e-06 -3.38807712e-03]]

residuals between the observed 2D points and the projected 3D points:
residual in lab1: 13.545832894768449
residual in lab2: 15.544953471723453

3.

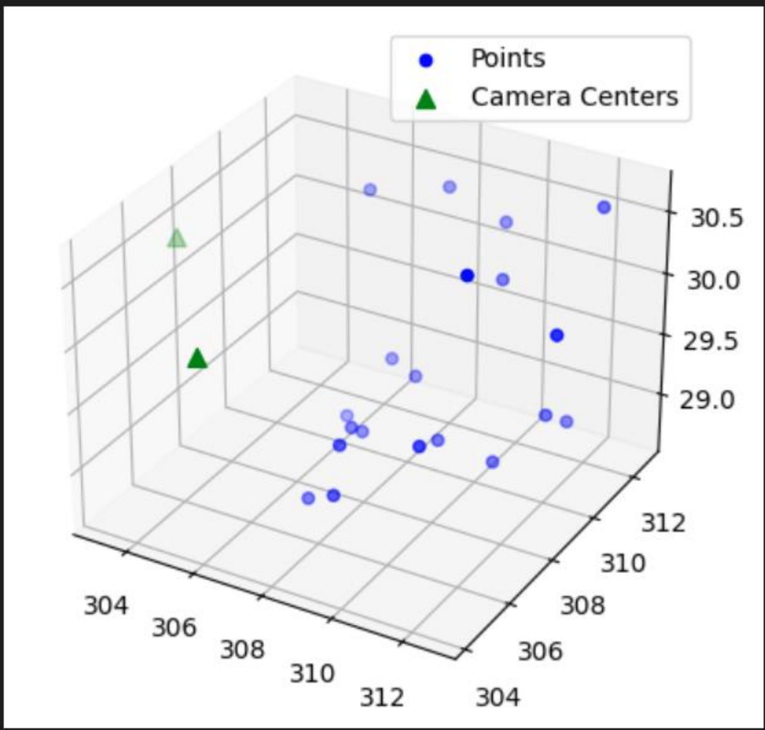
lab1 camera center [305.83276769 304.20103826 30.13699243]
lab2 camera center [303.10003925 307.18428016 30.42166874]
library1 camera center [7.28863053 -21.52118112 17.73503585]
library2 camera center [6.89405488 -15.39232716 23.41498687]

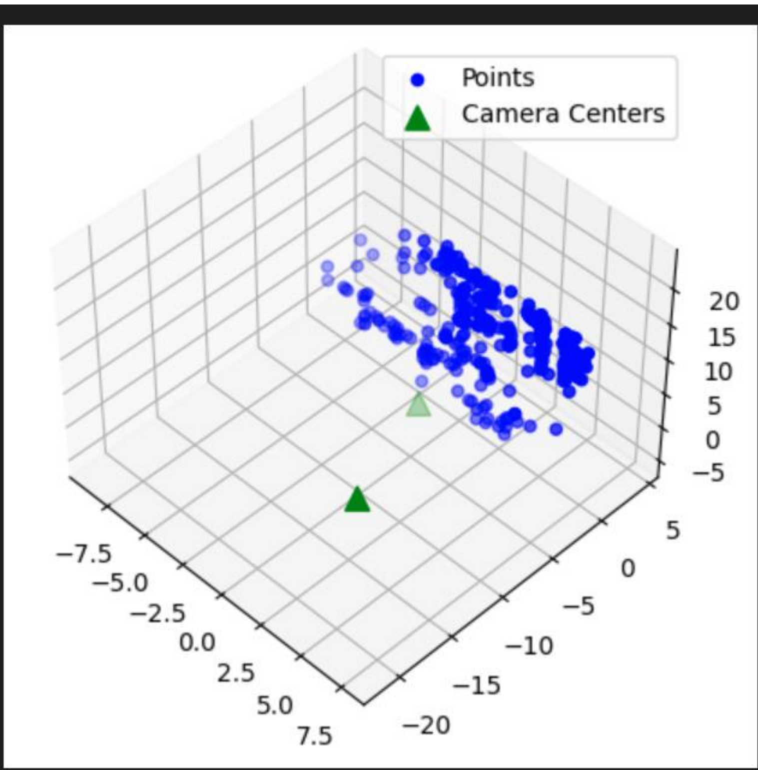
4.

Here are the 3D reconstruction errors for both lab pairs and library pairs, also the visualization.

```
Mean 3D reconstuction error for the lab data:  0.01332
2D reprojection error for the lab 1 data:  10.899446033014147
2D reprojection error for the lab 2 data:  1.5485148119181338
2D reprojection error for the library 1 data:  24.662071196868627
2D reprojection error for the library 2 data:  28.649537735261134

<matplotlib.legend.Legend at 0x20588740be0>
```





6. **Extra credits**

For the extra credits, I used the ransac to get the fundamental matrix, here's the results for both image pairs.

Lab:

```
Fundamental Normalized
[[-1.13377315e-06  1.55687381e-05 -3.89040871e-03]
 [ 1.07835509e-05 -2.65638358e-06  3.13466221e-02]
 [-2.33026057e-04 -4.30707351e-02  1.00000000e+00]]
lab: residual in frame 2 (normalized method) =  0.5497313255920021
lab: residual in frame 1 (normalized method) =  0.5833496860082279
lab: residual combined   (normalized method) =  0.566540505800115
```

The performance are relatively similar to the results from the ground truth matches. The number of inliers is 18 and the average residual is 0.61.

Library:

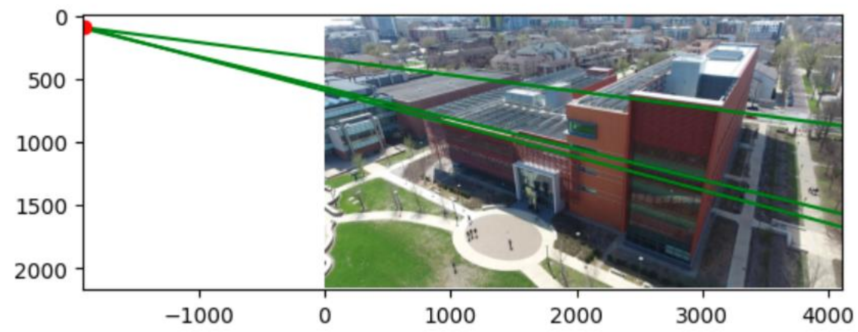
```
Fundamental Normalized
[[ 1.86520131e-07 -3.82943867e-06  5.65229482e-04]
 [ 2.22767397e-05  2.16768623e-07 -4.11451100e-02]
 [-5.30028152e-03  3.69650848e-02  1.00000000e+00]]
library: residual in frame 2 (normalized method) =  0.060334458479578496
library: residual in frame 1 (normalized method) =  0.05484509861965783
library: residual combined   (normalized method) =  0.057589778549618165
```

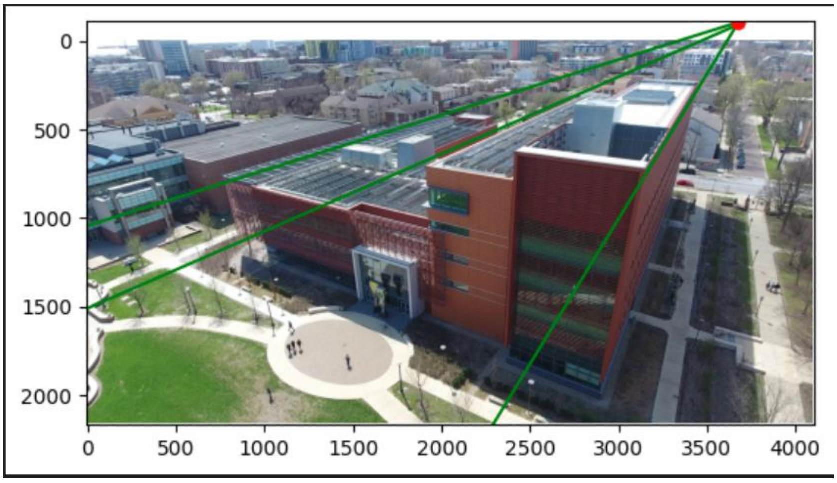
The number of inliers is 7 and the average residual is 0.67.

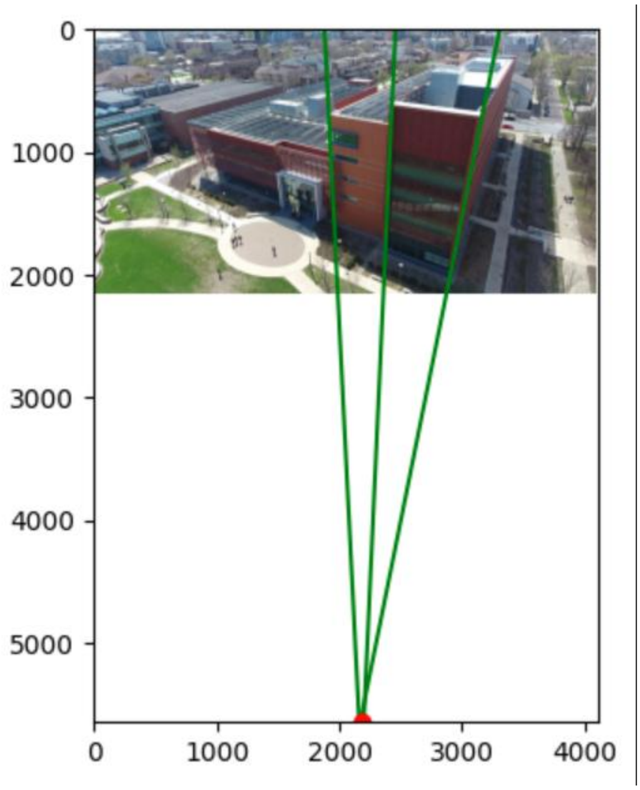
Q3:

1.

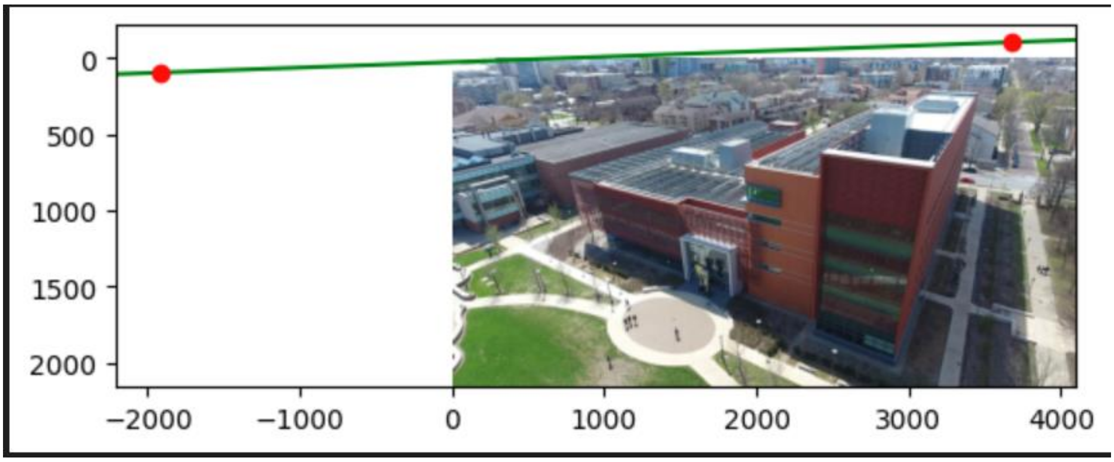
For three parallel lines in total, we have the vanishing points at (-1.91080224e+03, 9.38131166e+01,1), (3.67966328e+03, -1.05848093e+02, 1), (2.17582618e+03, 5.63670907e+03,1)







2.



The parameter is a = 0.03569183871611272
b= 0.9993628433402271
c = -25.55329751895859

```
# Part (2) Computing and plotting the horizon
# <YOUR CODE> Get the ground horizon line
horizon_line = get_horizon_line(vpts[:, 0], vpts[:, 1])
print(horizon_line[0])
print(horizon_line[1])
print(horizon_line[2])
# <YOUR CODE> Plot the ground horizon line
fig = plt.figure(); ax = fig.gca()
plot_horizon_line(vpts[:, 0], vpts[:, 1], im, ax)
fig.savefig('Q3_horizon.pdf', bbox_inches='tight')
```

[24]

```
...  0.03569183871611272
     0.9993628433402271
     -25.55329751895859
```

3.

The camera parameter is f = 5274144.54223915, optical center (principal point) of the camera is:

[[5.27414454e+06 0.00000000e+00 2.01457715e+03]
 [0.00000000e+00 5.27414454e+06 1.12177522e+03]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]

```
# Part (3) Computing Camera Parameters
# <YOUR CODE> Solve for the camera parameters (f, u, v)
f, u, v, K = get_camera_parameters(vpts)
print(f,u,v,K)
```

```
5274144.54223915 2014.5771455644888 1121.7752244881442 [[5.27414454e+06 0.00000000e+00 2.01457715e+03]
 [0.00000000e+00 5.27414454e+06 1.12177522e+03]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
```

4.The rotation matrix of the camera is :

[[-9.98796950e-01 9.99586526e-01 2.64829340e-10]
 [4.90372493e-02 -2.87537461e-02 7.41515741e-09]
 [9.91081853e-11 5.15062867e-11 -1.00000000e+00]]

```python
# Part (4) Computing Rotation Matrices
# <YOUR CODE> Solve for the rotation matrix
R = get_rotation_matrix(vpts, K)
print(R)
```

```
[[-9.98796950e-01  9.99586526e-01  2.64829340e-10]
 [ 4.90372493e-02 -2.87537461e-02  7.41515741e-09]
 [ 9.91081853e-11  5.15062867e-11 -1.00000000e+00]]
```