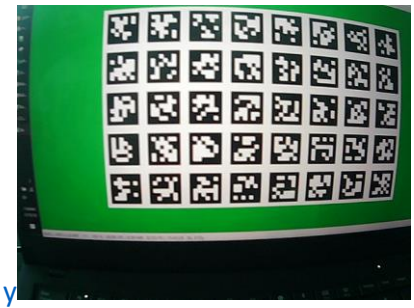
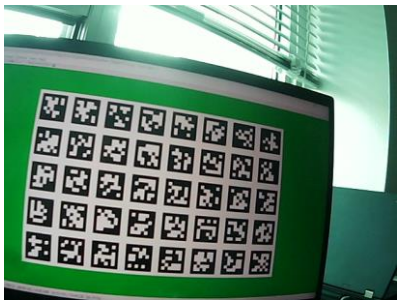
Task 6: F-matrix and Relative Pose (40 points)

Write code to answer the following questions, **document** the methods you use, and **discuss** the results you found.

1. Find and report the fundamental matrix (F-matrix) between the following two images. The raw images should be downloaded from here. (10 points)
2. Draw Epipolar lines in both images. (5 points)
3. Find the relative pose (R and t) between the two images, expressed in the left image's frame, using two different methods: a. Use F-matrix you find in 1. (10 points)
b. Do not explicitly use F-matrix (hint: this scene has a special geometric structure). (10 points)
4. Are the poses you get in 3.a and 3.b numerically close to each other? If not, analyse why. (5 points)

Hint: you should get the camera calibration matrix of the above two images by running: python demo_calib_by_photo.py after installing pyAprilTag from:
https://github.com/ai4ce/pyAprilTag#installation

y  

Solution:

1. To solve this problem, I need to find out the respective x points relative to the given images. In order to do so, I read the images and use the SIFT function. Then on I use the Python Dictionary to calculate the points keeping Flann Index as one. Then on I search 50 parameters from them and match using function of Open CV the indexed and searched parameters. Using the .match function z match our coordinates and if the distance between the points is within the certain limits then I store the Points for finding our Fundamental matrix.

Up next, I normalize the points. I use the mean and square root of standard deviations of the points to normalize the points. Once the points get normalizes, they are stacked horizontally. Then on I initialize the A matrix and fill it as mentioned in the equations. By taking the SVD of the A matrix I come up with the F_hat matrix by Reshaping the last column of V. F_hat being a full matrix again the SVD of it is taken and the resulting matrix se is used to dot product between themselves and multiplied with t1, t2 from before to come up with the Fundamental Matrix. Separately no Open CV direct feature was used to come up with the Fundamental Matrix that is why the solution is not exact.

My results for this Question is:

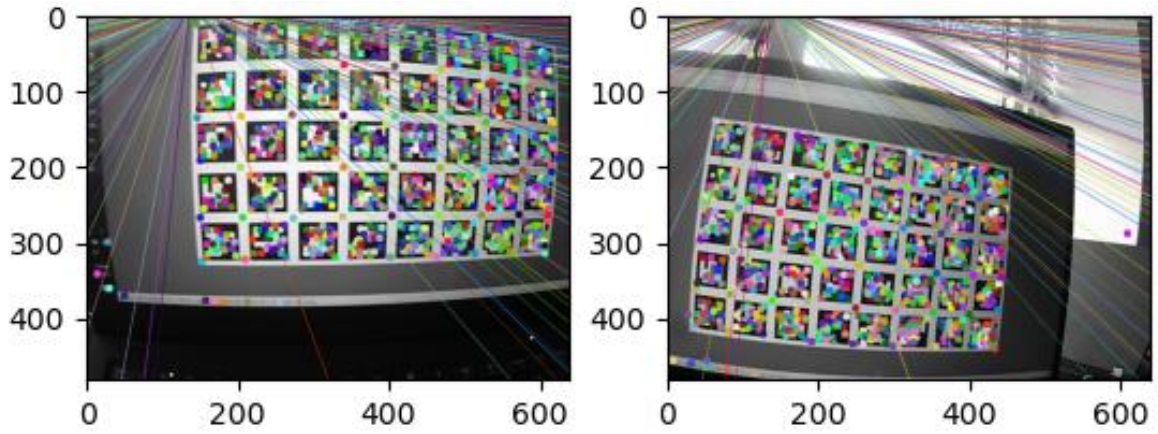
```
6. 1) Fundamental matrix is:
        [[-2.58139924e-04 -1.48663474e-04  1.47904572e-01]
        [ 4.19855030e-06  1.24361190e-03 -3.45724272e-01]
        [ 1.01060831e-01 -1.54753745e-01  1.00000000e+00]]
```

2. Open CV has an inbuilt function of computing and plotting epilines called:

```
cv2.computeCorrespondEpilines (pts1.reshape (-1, 1, 2), 2, F), where F=Fundamental Matrix
```

I use this function to come up with the lines required to plot in the images read previously. These lines are then plotted on the image using matplotlib. In addition, we set circles at desired points in the Image with commands like cv2.circle, cv2.lines etc. This is how the lines are plotted on the image using the Fundamental Matrix.

My results for this Question is:



3.a) In order to carry this part out, firstly I need to calculate the K matrices separately from both the images separately. This is one by demo_calib_by_photo.py provided by pyAprilTag. When I get the K matrices, I take the inverse of one, multiply it with the F matrix, and multiply the other K matrix with it. Therefore, I generate the E matrix. If the E matrix is a full matrix, I take the SVD of it again and I multiply the U and V together making their dimensions agreeable. Thereafter reshaping the matrix to the shape of 3*4, I get the R and T in the form of 3*3 and 3*1 together in the form of a Transformation matrix of 3*4.

My results for this Question is:

```
6.3.a)  E matrix:
        [[ -1.8818288  -24.35845875  12.10524337]
        [ 22.35372164  -4.27563478  16.1217551 ]
        [-16.20069416 -17.99219263  -0.19732173]]

The final answer is:
        [[ 0.39730119  0.05125115 -0.91625601 -0.58132566]
        [-0.27732791  0.95846136 -0.06664127  0.44483529]
        [ 0.87478054  0.28058002  0.39501121  0.68130906]]

The Rotation Vector is:
        [[ 0.22214499]
        [-1.1458681 ]
        [-0.21021808]]

 The Translation Vector is:
        [[-0.58132566]
        [ 0.44483529]
        [ 0.68130906]]
```

3.b) For 3.b, I need to use the same method as of Task 4. Firstly, I can find the IDs, corners, centres from the given two images. From there I need to find out that if the corner size is not equal to zero then I can use the Solve PnP function to find out the Rotation and Translation of each one with respect to the original picture of all the April Tags.

There after I can just multiply the two matrices and get the resulting Rotation and Translation matrix. This method is directly made from the Points we detect by pyAprilTag and the answer will not be equal to that of Task 3.a. The code is attached.

My results for this Question is:

```
6.3.b) The final answer is:
        [[ 0.95157084  0.10716696  0.28814612 -2.18857275]
        [ 0.00838376  0.92788078 -0.37278275  4.96645714]
        [-0.30731524  0.35714494  0.88204582 -2.89717243]
        [ 0.          0.          0.          1.        ]]
The Rotation Vector is:
        [[ 0.38020068]
        [ 0.31016061]
        [-0.05145365]]
The Translation Vector is:
        [[-2.18857275]
        [ 4.96645714]
        [-2.89717243]]
```

4. The principal reasons for the values being different are:

- The PnP method relates points from 3-D to 2-D and the Fundamental matrix relates it from 2-D to 2-D.
- More number of points are considered in the method of finding out the Fundamental matrix than the PnP method.
- The Solve PnP is an inbuilt optimizer, but I calculated the Fundamental matrix by Code, so more optimization is there in the PnP method.
- Lastly, The Normalizations performed in the Fundamental Matrix method is not verified and confirmed, so that might concern us with distortions in values of the resulting R and T values.