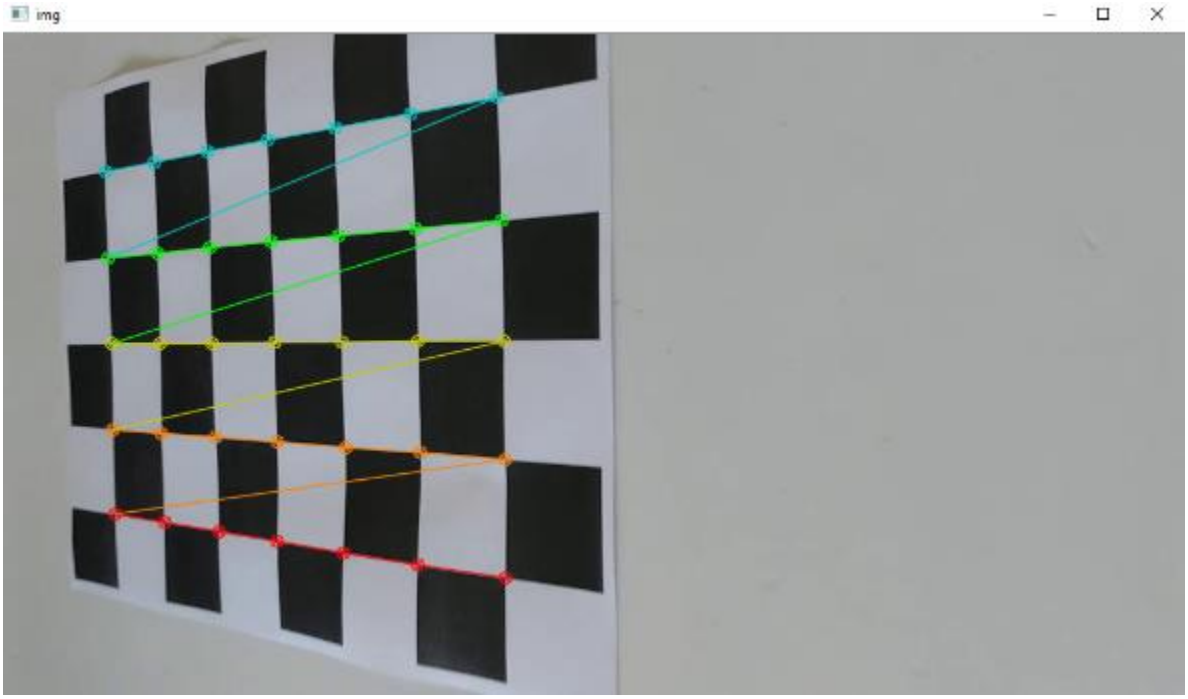
Task 1




This image shows the camera calibration stage of the program (subtasks A and B). Camera calibration simply refers to finding the intrinsic parameters of a camera, such as the focal length and optical center. In this image, the program found the image points of each interior corner in the checkerboard pattern. Since we know in the real world, each corner point is a unit space from the next, we can construct a camera matrix that maps the 3D image points (0,0,0), (1,0,0), (2,0,0) ... (6,5,0) to the 2D image points shown by the circles in the image above. The calibration process is done in the file "camera_calibration.py" in lines 33 – 54. The program iterates through each calibration image and finds the 2D image points using the "cv2. findChessboardCorners" method. It then calculates the camera matrix K and distortion coefficients on line 54 using the "cv2.calibrateCamera" method and saves them to a file.

```
Camera matrix:
[[994.93741654   0.         485.13219679]
 [  0.         953.82995311 286.16775089]
 [  0.           0.           1.        ]]
Distortion coefficients:
[[ 3.40589672e-01 -1.91044041e+00  1.88896990e-03 -3.99447870e-03
   3.16561429e+00]]
```

Shown above are the resulting camera calibration matrix K as well as the distortion coefficients. The camera calibration contains the intrinsic parameters including two different focal lengths (which should be the same in a true pinhole camera) as well as the principal point.


Shown above is the result of the KLT feature-point tracking throughout the entirety of the window (subtasks C and D). As expected by a perspective projection, all feature points shifted toward the vanishing point in the frame. The feature tracking process initiates by finding good feature points to track in the first frame using the “cv2.goodFeaturesToTrack” method. Then, the Kanade-Lucas-Tomasi feature tracker tracks the movement of these features from frame to frame. Each feature point is stored in an array “tracks.” If the point disappears or moves out of the window, the track is removed from the array “tracks.” The entire feature tracking algorithm is located in lines 27 – 78 in the file “optical_flow.py.” The green lines show the movement of feature points from frame to frame. In the end, the feature points in the initial frame and the final frame are extracted from “tracks” to establish the correspondences between the images.

Task 2


Shown above is the result of running 10000 iterations to calculate the best fundamental matrix with the fewest outliers and the smallest inlier variance. The green lines indicate correspondences between the initial and final frames that were inliers, while the red lines indicate correspondences that were outliers. The algorithm calculating the fundamental matrix is located in file “fundamental_mat_calc.py” in lines 16 – 226. The algorithm starts by calculating the average of both x coordinates and y coordinates in the initial and final frames (lines 42 – 55). It then uses these averages, as well as the standard deviations, to calculate the holography matrices for both frames for subtask B (lines 61 – 67). Then, the 10000-iteration process starts. For each iteration, the algorithm selects 8 random points and adds a row in the A matrix using the corresponding coordinates (subtask C – lines 92 - 105). After constructing the A matrix, the fundamental matrix is calculated by finding the eigenvector corresponding to the least eigenvalue in matrix A (also subtask C). The fundamental matrix is then denormalized on line 129 (subtask D). For subtasks E and F, the algorithm goes through the remaining correspondences in lines 135 – 161 and calculates the variance metric and determines if the correspondence is an outlier or inlier. If the correspondence is an outlier, it saves the index of the correspondence in the array “tracks” to an array “outlier_indices.” If it is an inlier, the variance is added to the inlier sum. At the very end of the algorithm (line 168), the fundamental matrix with the least outliers and the lowest inlier sum is selected (subtask G). Then the epipoles are calculated in lines 213 – 221 and outputted (subtask H).

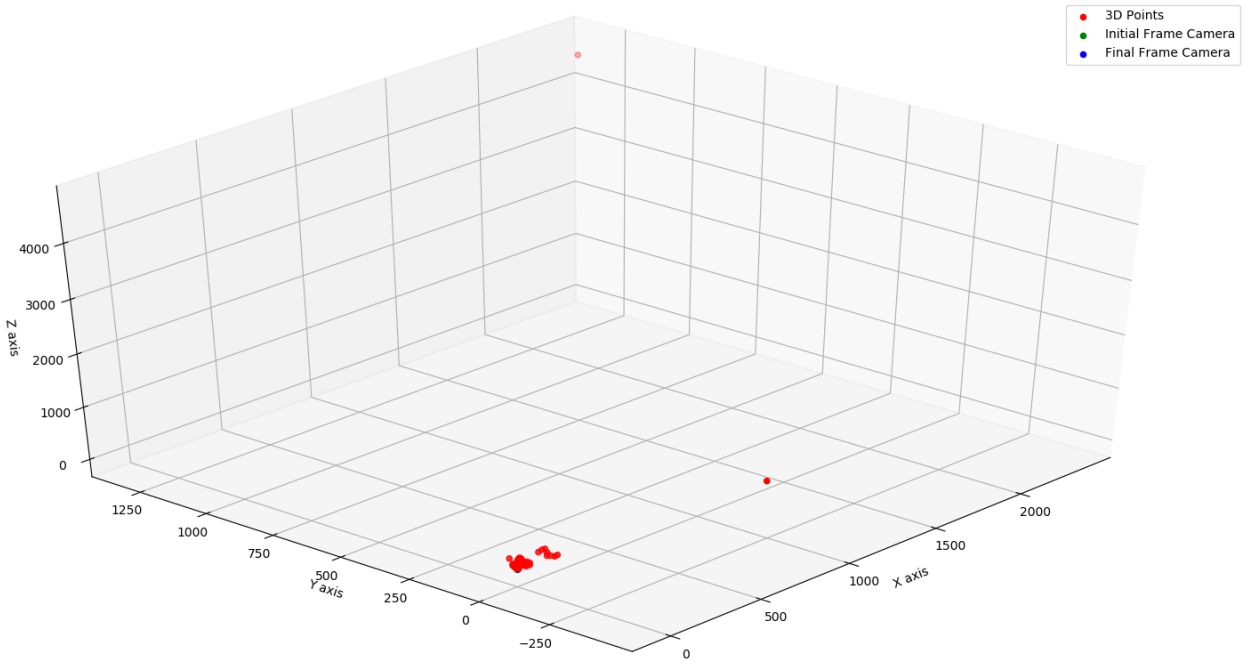
```
Running iterations
Finished fundamental matrix iterations
Fundamental matrix:
[[-6.96847805e-08 -2.07285271e-05  6.44317389e-03]
 [ 2.06920879e-05 -1.40993771e-07 -9.20604470e-03]
 [-6.32019036e-03  9.36591956e-03 -7.19755092e-02]]
Number of outliers:
18
Inlier sum:
106.25089750506524
Outlier indices:
[0, 2, 5, 8, 12, 14, 26, 27, 37, 40, 42, 45, 60, 62, 77, 82, 83, 88]
**Press escape after done viewing**
First frame epipole:
[447.01428076 309.3333056    1.        ]
Last frame epipole:
[449.749316   306.95457538   1.        ]
```

Here is the result after running 10000 iterations. Shown is the best fundamental matrix which has 18 outliers and an inlier sum of about 106. The outlier indices (the correspondence indices within the array “tracks”) are also shown.

Task 3

```
Essential matrix:
[[-5.08740360e-03 -9.99065695e-01  2.40692818e-02]
 [ 9.99013225e-01 -4.93655776e-03  3.83939615e-02]
 [-2.19524269e-02 -3.54221120e-02  1.92540323e-05]]
```

Shown above is the resulting essential matrix that is calculated from the fundamental matrix and camera matrix K (subtask A). The essential matrix is calculated in the file “essential_mat_3d.py” on line 68. In accordance with the directions of subtask A, the constraints of the essential matrix (non-zero singular values being identical and rotation matrices of SVD belonging to SO(3)) are enforced in lines 74 – 78. For subtask B, the four possible combinations of R and t are calculated and stored within variables “R_1”, “R_2”, and “T” in lines 88 – 91. The four possible combinations are: (R_1, T), (R_1, -T), (R_2, T), and (R_2, -T). The scale of the z coordinate of the translation vector is calculated using the properties of the video on line 94. The 3D points are then calculated on lines 20 – 39 for each combination of R and T (subtask C). The 3D points are calculated by finding the vectors m and m_prime on lines 23 and 24 using the inverse camera calibration matrix (mapping 2D points to 3D points). The distance component is then found by solving the linear system in lines 27 – 32. The combination of R and T that resulted in the fewest points behind the frames is selected on line 114.




Here is the resulting 3D plot showing the 3D points calculated from the correspondences and essential matrix (subtask D).


The reprojection error of the 3D coordinates (shown in red) with respect to the original correspondences (shown in green) are shown in the initial frame (subtask E). As you can see, most points coincide.


The reprojection error of the 3D coordinates (shown in red) with respect to the original correspondences (shown in green) are shown in the final frame (subtask E).