1 Methodology

Finding keypoints and correspondences

SIFT from the cv2 module was used to compute keypoints in both images. FLANN (a knn algorithm) from the cv2 module was then used to match keypoints to determine correspondences. At this point, a findHomography, another cv2 module was used to calculate inliers for visualisation purposes - no data was from this function otherwise, apart from obtaining an initial estimate of inlier rate.

Note that much of this step's code was derived from workshops 7 and 8.

Shifting and scaling pixel co-ordinates

When calculating the fundamental matrix, a poorly distributed set of correspondence points across the image co-ordinate domain may lead to an ill-defined null space and therefore an inaccurate fundamental matrix. To counteract this, correspondence point co-ordinates are shifted and scaled to be centered at (0, 0) and have average distance of sqrt(2) from the origin.

This was achieved by constructing a transformation matrix T defined as:

$$T = \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & -xc \\ 0 & 1 & -yc \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

where s is the scale factor, and xc, yc are centroids of the correspondence points. The new pixel co-ordinates are then calculated as $x_{new} = Tx$ for each co-ordinate point.

Computing design matrix

Eight random points were chosen from all correspondence points. The reason to use only 8 points is because it is the minimum to form the fundamental matrix, and using more would only increase the computation time taken to find a fundamental matrix free of outliers. The design matrix was formed thus:

$$\begin{bmatrix} p_1^{(1)}q_1^{(1)} & p_2^{(1)}q_1^{(1)} & q_1^{(1)} & p_1^{(1)}q_2^{(1)} & p_2^{(1)}q_2^{(1)} & q_2^{(1)} & p_1^{(1)} & p_2^{(1)} & 1 \\ p_1^{(2)}q_1^{(2)} & p_2^{(2)}q_1^{(2)} & q_1^{(2)} & p_1^{(2)}q_2^{(2)} & p_2^{(2)}q_2^{(2)} & q_2^{(2)} & p_1^{(2)} & p_2^{(2)} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ p_1^{(n)}q_1^{(n)} & p_2^{(n)}q_1^{(n)} & q_1^{(n)} & p_1^{(n)}q_2^{(n)} & p_2^{(n)}q_2^{(n)} & q_2^{(n)} & p_1^{(n)} & p_2^{(n)} & 1 \end{bmatrix} \tag{2}$$

where $p^{(n)}$ refers to the co-ordinates of the nth corresponding point in the left image, and $q^{(n)}$ to the right.

Performing SVD on the design matrix

Numpy's svd function is used to perform SVD on the design matrix, returning $A = UDV$ where A is the design matrix.

Composing F

To find the fundamental matrix, the final column from V was extracted. As numpy's svd function returns results in decreasing order of magnitude, the least singular value is always from the final column. The fundamental matrix is simply the final column re-arranged into a 3x3 matrix.

Performing SVD on F

SVD is performed on F, setting the last row of D to zero and reassembling. This has the effect of changing the determinant of F to zero, making it a rank-2 matrix. This is important to reduce the effects of noisy image co-ordinates, which may cause the definition $q^T Dp = 0$ to fail.

Calculating inliers

The first step is to calculate epilines. To find the epilines for the left image (right image follows similarly):

$$a, b, c = F_u q_n \tag{3}$$

where:

$$ax + by + c = 0 \tag{4}$$

F is unscaled and unshifted back to image co-ordinates by applying:

$$F_u = T_R^T \times F \times T_L \tag{5}$$

where T_R is the translation matrix from the right image and T_L is the translation matrix from the left image.

The distance from each point to their epiline is then calculated thus:

$$d = \begin{bmatrix} a_R & b_R & c_R \end{bmatrix} \times F \times \begin{bmatrix} a_L \\ b_L \\ c_L \end{bmatrix} \tag{6}$$

Whie inliners should theoretically lie upon their epilines exactly, numerical error leads to slight distances so an error tolerance of 1e-3 was used as the distance cutoff for inliner identification.

RANSAC loop

The above steps were wrapped into a RANSAC loop, randomly choosing 8 correspondence points to form the design matrix each time. The RANSAC loop was ran for 5000 iterations. The iteration number was chosen by the process below:

First, initial inlier rate estimates were collected from the first step. Inlier rates varied from 5% to 80% depending on the images. Then the iteration count required to have 99% confidence of finding 8 inliers was then calculated from:

$$\frac{\log(1 - 0.99)}{\log(1 - r^8)} \tag{7}$$

where r was the inlier rate for that image.

As some images required absurd amounts of iterations to form 99% confidence, I manually chose a value of 5000 iterations that satisfied the majority of non-absurd iteration counts. This strikes a balance between obtaining 99% confidence for most images with considerations of run-time becoming untenable.

The iteration which yielded the highest amount of inliers was then chosen as the output fundamental matrix.

Re-estimate F

Finally, one more iteration was ran, using all inliers from the output fundamental matrix as part of the design matrix to return the final high-quality fundamental matrix.

Re-compute F

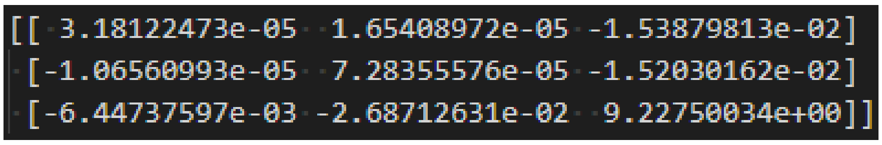
The final fundamental matrix was then unscaled and unshifted back to the original pixel co-ordinates thus:

$$F_u = T_R^T \times F \times T_L \tag{8}$$

similar to previously when calculating inliers.

2 Results

File: booksh
Calculated number of iterations needed: 17656
Fundamental matrix:



```
[[ 3.18122473e-05  1.65408972e-05 -1.53879813e-02]
 [-1.06560993e-05  7.28355576e-05 -1.52030162e-02]
 [-6.44737597e-03 -2.68712631e-02  9.22750034e+00]]
```

Figure 1: booksh fundamental matrix

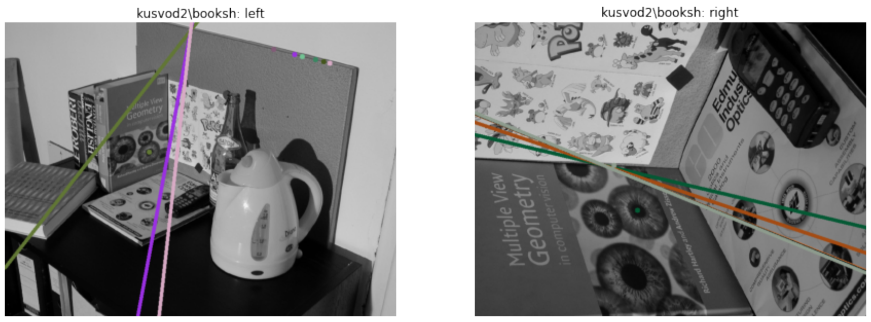


Figure 2: booksh epilines and inliers

File: box
Calculated number of iterations needed: 323
Fundamental matrix:

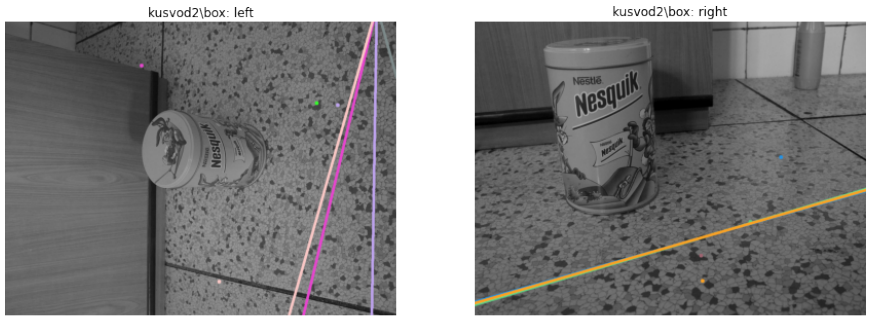
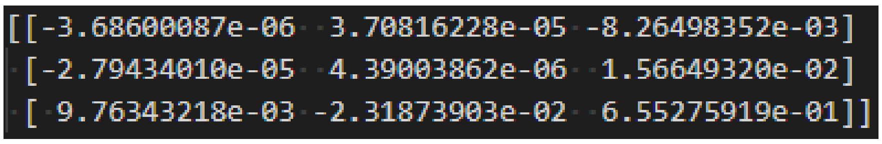


Figure 3: box epilines and inliers

File: castle
Calculated number of iterations needed: 2436
Fundamental matrix:



```
[[-3.68600087e-06  3.70816228e-05 -8.26498352e-03]
 [-2.79434010e-05  4.39003862e-06  1.56649320e-02]
 [ 9.76343218e-03 -2.31873903e-02  6.55275919e-01]]
```

Figure 4: castle fundamental matrix

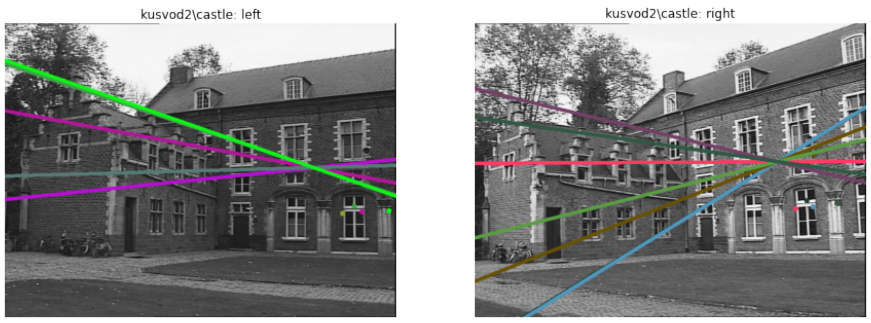
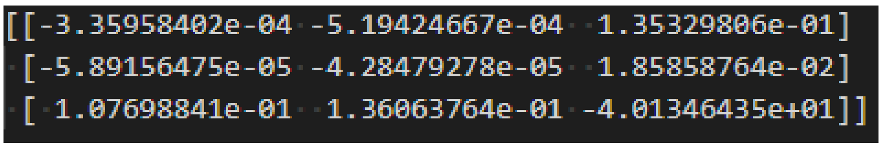


Figure 5: castle epilines and inliers

File: corr
Calculated number of iterations needed: 21
Fundamental matrix:



```
[[-3.35958402e-04 -5.19424667e-04  1.35329806e-01]
 [-5.89156475e-05 -4.28479278e-05  1.85858764e-02]
 [ 1.07698841e-01  1.36063764e-01 -4.01346435e+01]]
```

Figure 6: corr fundamental matrix

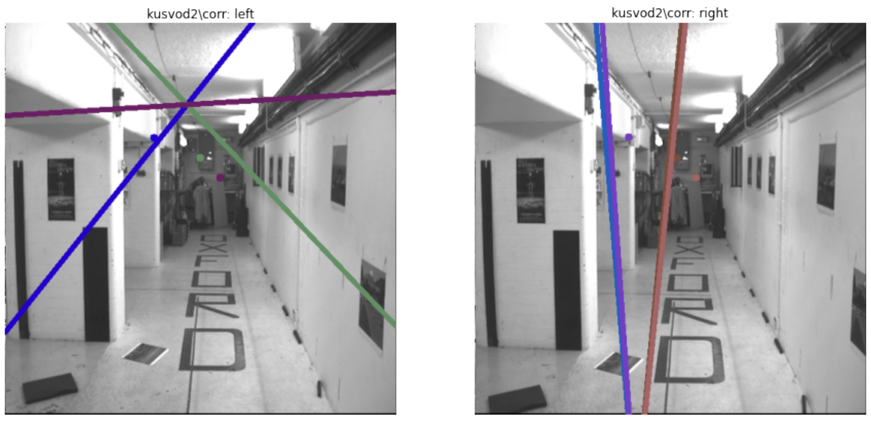
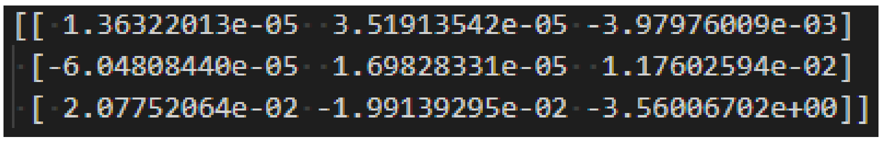


Figure 7: corr epilines and inliers

File: graff
Calculated number of iterations needed: 333684762580
Fundamental matrix:



```
[[ 1.36322013e-05  3.51913542e-05 -3.97976009e-03]
 [-6.04808440e-05  1.69828331e-05  1.17602594e-02]
 [ 2.07752064e-02 -1.99139295e-02 -3.56006702e+00]]
```

Figure 8: graff fundamental matrix

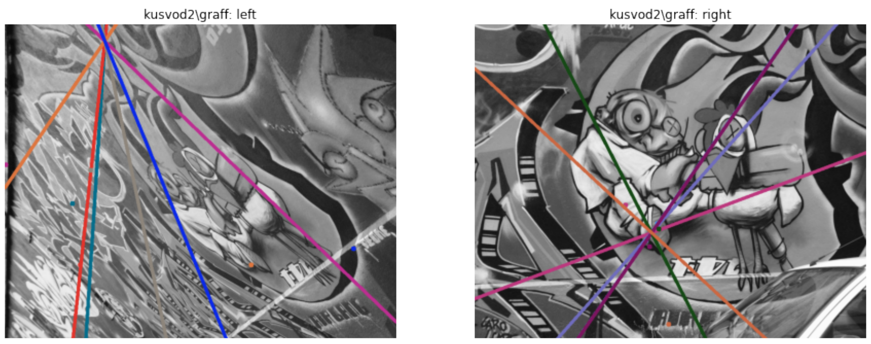
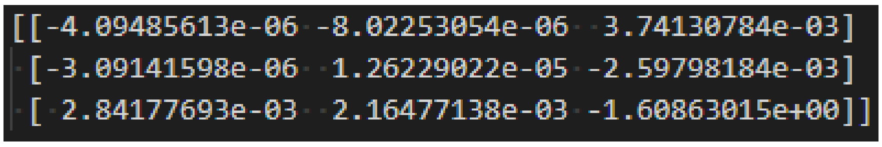


Figure 9: graff epilines and inliers

File: head
Calculated number of iterations needed: 361900
Fundamental matrix:



```
[[-4.09485613e-06 -8.02253054e-06  3.74130784e-03]
 [-3.09141598e-06  1.26229022e-05 -2.59798184e-03]
 [ 2.84177693e-03  2.16477138e-03 -1.60863015e+00]]
```

Figure 10: head fundamental matrix

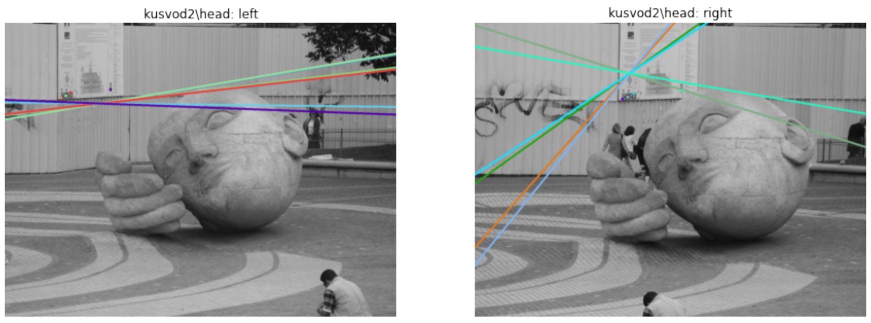
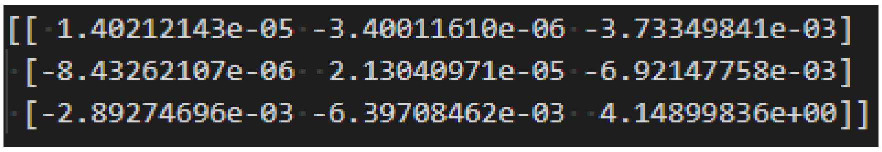


Figure 11: head epilines and inliers

File: kampa
Calculated number of iterations needed: 3484
Fundamental matrix:



```
[[ 1.40212143e-05 -3.40011610e-06 -3.73349841e-03]
 [-8.43262107e-06  2.13040971e-05 -6.92147758e-03]
 [-2.89274696e-03 -6.39708462e-03  4.14899836e+00]]
```

Figure 12: kampa fundamental matrix

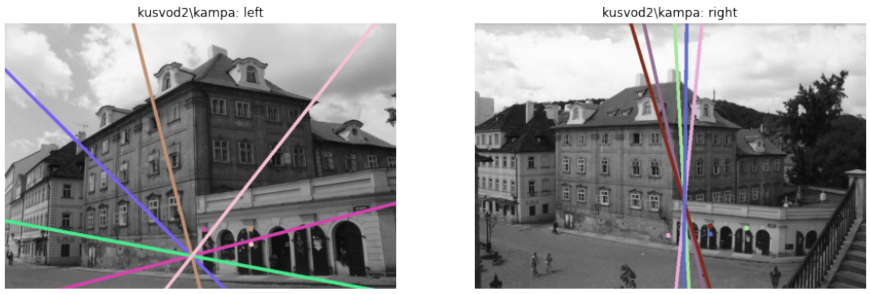
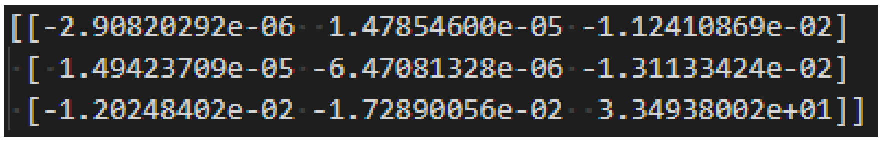


Figure 13: kampa epilines and inliers

File: Kyoto
Calculated number of iterations needed: 451255715
Fundamental matrix:



```
[[-2.90820292e-06  1.47854600e-05 -1.12410869e-02]
 [ 1.49423709e-05 -6.47081328e-06 -1.31133424e-02]
 [-1.20248402e-02 -1.72890056e-02  3.34938002e+01]]
```

Figure 14: Kyoto fundamental matrix

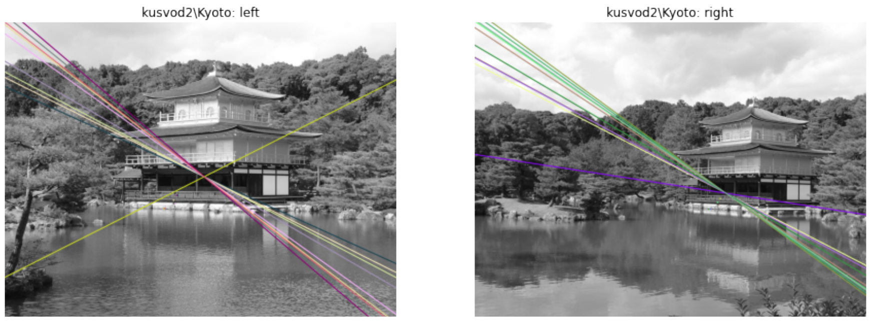
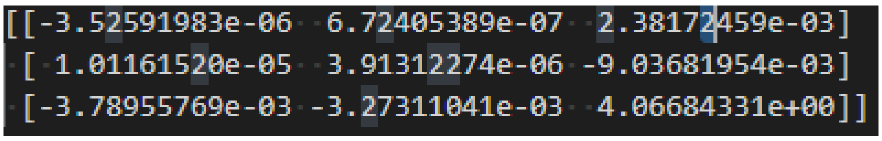


Figure 15: Kyoto epilines and inliers

File: leafs
Calculated number of iterations needed: 54232373972
Fundamental matrix:



```
[[-3.52591983e-06  6.72405389e-07  2.38172459e-03]
 [ 1.01161520e-05  3.91312274e-06 -9.03681954e-03]
 [-3.78955769e-03 -3.27311041e-03  4.06684331e+00]]
```

Figure 16: leafs fundamental matrix

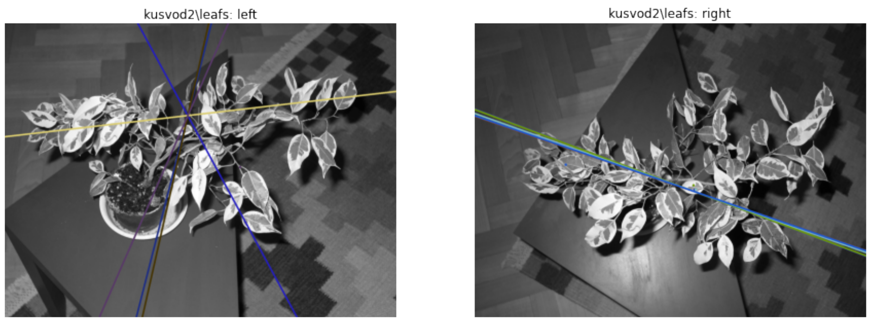
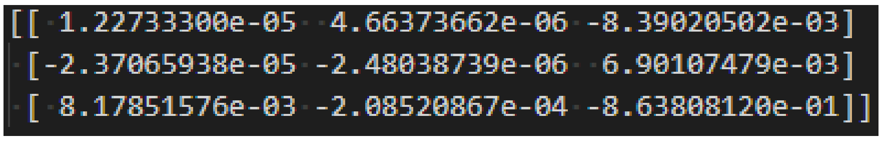


Figure 17: leafs epilines and inliers

File: plant
Calculated number of iterations needed: 155610550
Fundamental matrix:



```
[[ 1.22733300e-05  4.66373662e-06 -8.39020502e-03]
 [-2.37065938e-05 -2.48038739e-06  6.90107479e-03]
 [ 8.17851576e-03 -2.08520867e-04 -8.63808120e-01]]
```

Figure 18: plant fundamental matrix

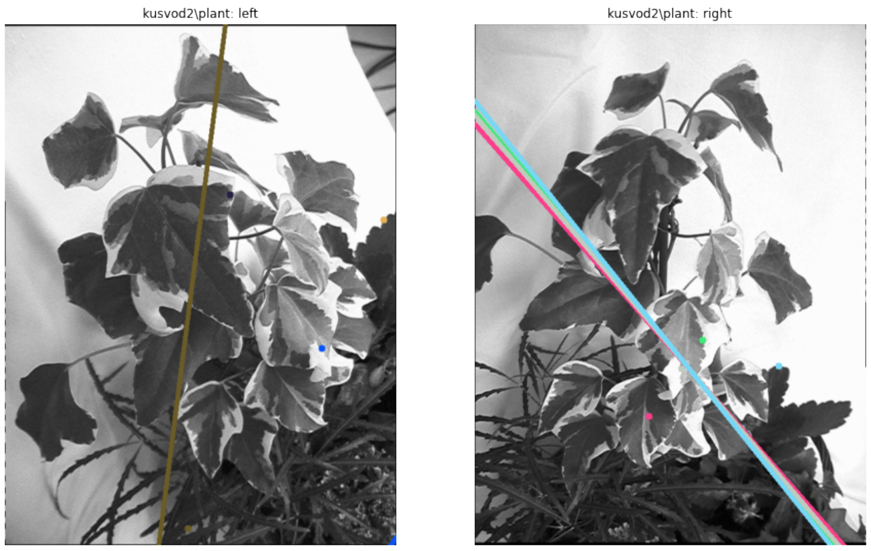
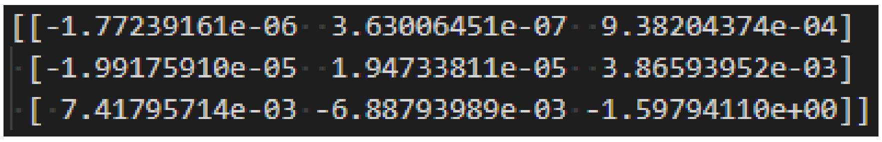


Figure 19: plant epilines and inliers

File: rotunda
Calculated number of iterations needed: 941216
Fundamental matrix:



```
[[-1.77239161e-06  3.63006451e-07  9.38204374e-04]
 [-1.99175910e-05  1.94733811e-05  3.86593952e-03]
 [ 7.41795714e-03 -6.88793989e-03 -1.59794110e+00]]
```

Figure 20: rotunda fundamental matrix

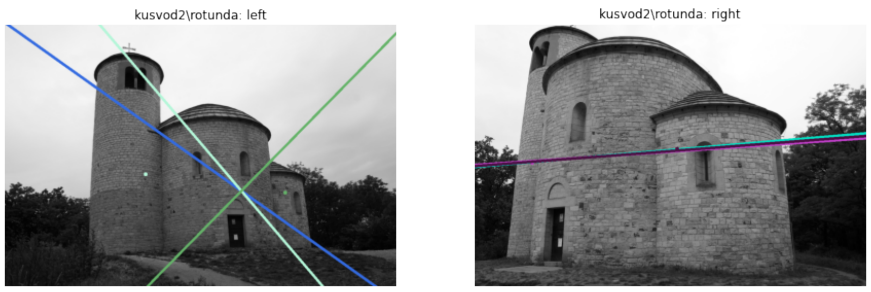
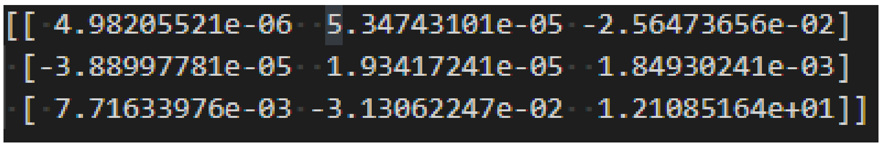


Figure 21: rotunda epilines and inliers

File: shout
Calculated number of iterations needed: 6569
Fundamental matrix:



```
[[ 4.98205521e-06  5.34743101e-05 -2.56473656e-02]
 [-3.88997781e-05  1.93417241e-05  1.84930241e-03]
 [ 7.71633976e-03 -3.13062247e-02  1.21085164e+01]]
```

Figure 22: shout fundamental matrix

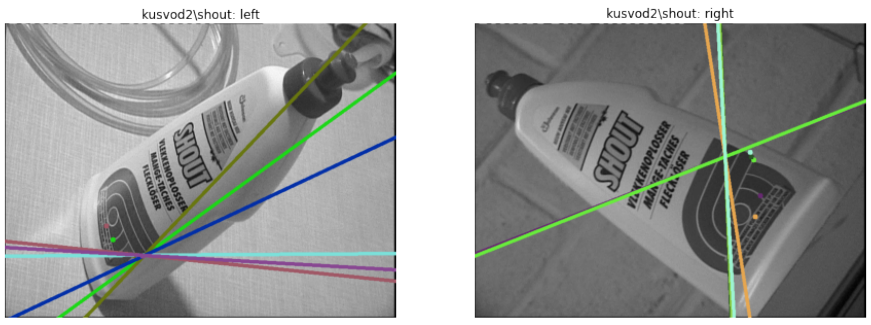
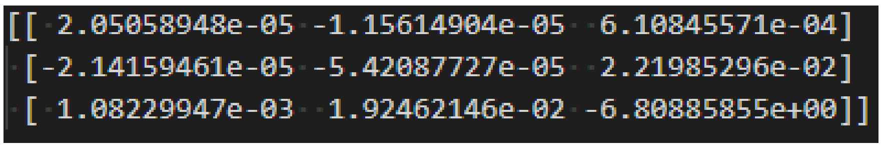


Figure 23: shout epilines and inliers

File: valbonne
Calculated number of iterations needed: 489
Fundamental matrix:



```
[[ 2.05058948e-05 -1.15614904e-05  6.10845571e-04]
 [-2.14159461e-05 -5.42087727e-05  2.21985296e-02]
 [ 1.08229947e-03  1.92462146e-02 -6.80885855e+00]]
```

Figure 24: valbonne fundamental matrix

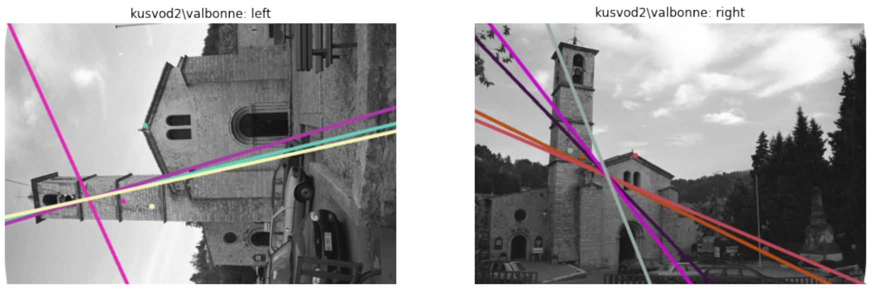


Figure 25: valbonne epilines and inliers

File: wall
Calculated number of iterations needed: 172695723
Fundamental matrix:

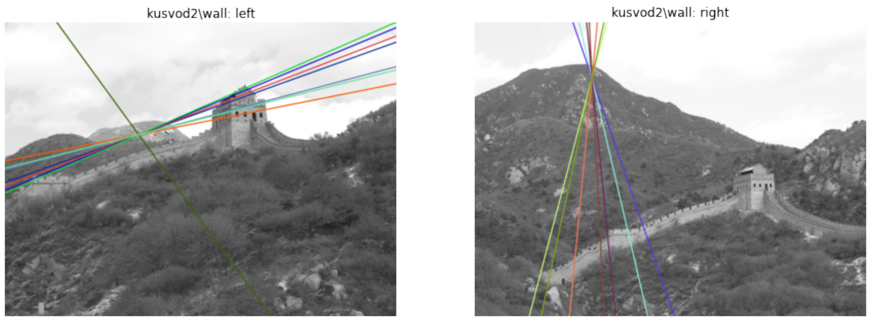


Figure 26: wall fundamental matrix

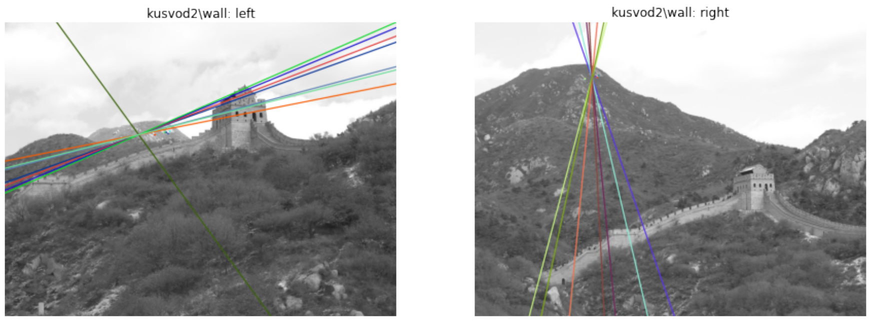
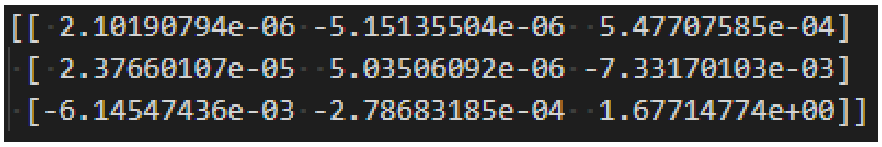


Figure 27: wall epilines and inliers

File: wash

Calculated number of iterations needed: 2591030
Fundamental matrix:



```
[[ 2.10190794e-06 -5.15135504e-06  5.47707585e-04]
 [ 2.37660107e-05  5.03506092e-06 -7.33170103e-03]
 [-6.14547436e-03 -2.78683185e-04  1.67714774e+00]]
```

Figure 28: wash fundamental matrix

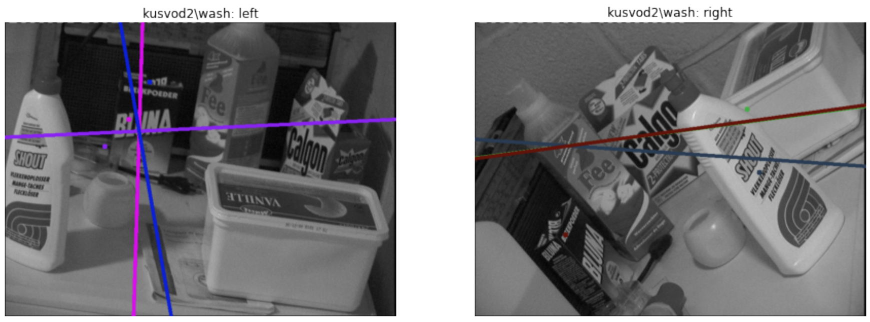
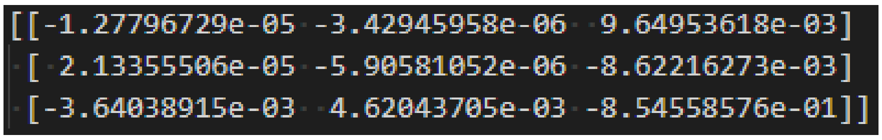


Figure 29: wash epilines and inliers

File: zoom
Calculated number of iterations needed: 23240
Fundamental matrix:



```
[[-1.27796729e-05 -3.42945958e-06  9.64953618e-03]
 [ 2.13355506e-05 -5.90581052e-06 -8.62216273e-03]
 [-3.64038915e-03  4.62043705e-03 -8.54558576e-01]]
```

Figure 30: zoom fundamental matrix

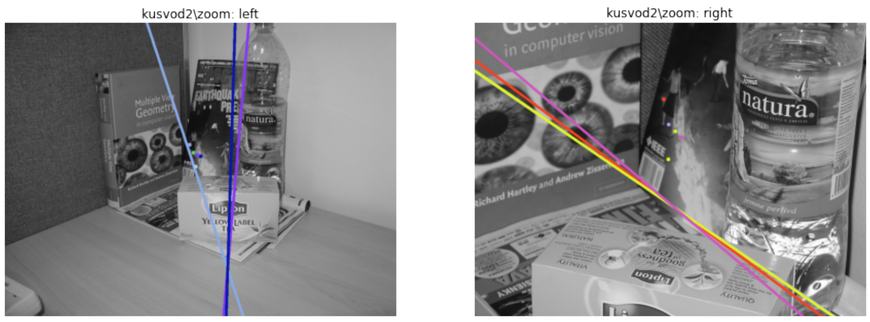


Figure 31: zoom epilines and inliers

3 Discussion

Clearly the fundamental matrices calculated are of low quality and the epilines drawn in the images are of dubious accuracy. This is possibly due to an insufficient number of iterations ran for the RANSAC loops due to time constraints. However, more likely is a mistake in step 1, with finding correspondences which then propagated to the final results. This is evident due to the incredibly absurd iteration counts calculated from inlier rates. Another possibility is errors in the fundamental matrix calculations.

Regardless, images ”box”, ”graff”, ”plant”, and ”wall” seemed to be most difficult to analyse.
Image ”box” contains a complicated table surface pattern which may have confused the correspondence calculations, leading to inaccurate correspondences.
Image ”graff” consists of two very different angles, so the extreme angle difference may have lead to difficulty in calculating correspondences again.
Image ”plant” has difficult to identify edges.
Image ”wall” seems to have an entire mountain visible in the right image which is missing in the left image, along with other missing features from both sides, leading to many false positives in correspondence point analysis.