Q1

To find the keypoints, I applied the SIFT library which is used in week 7 tutorial.
To find the correspondences, I applied the FLANN library which is also in week 7 tutorial.

Q2

Step a: shift and scale the pixel coordinates

To shift and scale the coordinates, I first follow the method discussed on Ed discussion board.

You might have an image that is 640x480 pixels. A reasonable normalisation might be to put (0,0) in the middle of the image and have the x range go from -1 to 1. Thus you might have:

$$\text{input} = \begin{pmatrix} x \\ y \end{pmatrix} \qquad \text{output} = \frac{1}{320}\begin{pmatrix} x - 320 \\ y - 240 \end{pmatrix}$$

This would result in new (output) values that are between -1 and 1 for both x and y (so both +ve and -ve values).

Deduct height/2, width/2 from x and y respectively then divide height/2, width/2 to get the coordinates from -1 to 1. However, using this method, it is very hard to transform the calculated F to the original pixel coordinate.

Then I found another method to normalize the coordinates.
https://www.cc.gatech.edu/classes/AY2016/cs4476_fall/results/proj3/html/sdai30/index.html

1. Compute the centroid of all keypoints.
2. Recenter all the points by subtracting the centroid.
3. Define a scale term for two images.

$$\mathbf{s} = \frac{\sqrt{2}}{\left(\frac{1}{n}\sum_{i=1}^{n}(\tilde{u}_i^2 + \tilde{v}_i^2)\right)^{1/2}}$$

$$\mathbf{s}' = \frac{\sqrt{2}}{\left(\frac{1}{n}\sum_{i=1}^{n}\left(\tilde{u'}_i^2 + \tilde{v'}_i^2\right)\right)^{1/2}}$$

4. Construct transformation matrix

$$T_a = \begin{pmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & -\bar{\mathbf{u}} \\ 0 & 1 & -\bar{\mathbf{v}} \\ 0 & 0 & 1 \end{pmatrix}$$

$$T_b = \begin{pmatrix} s' & 0 & 0 \\ 0 & s' & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & -\bar{\mathbf{u}}' \\ 0 & 1 & -\bar{\mathbf{v}}' \\ 0 & 0 & 1 \end{pmatrix}$$

5. Normalize the points using the above transformation matrix

$$\hat{\mathbf{x}}_\mathbf{i} = T_a\mathbf{x}$$
$$\hat{\mathbf{x}}'_\mathbf{i} = T_b\mathbf{x}$$

By applying the above method, it is easy to retrieve F in the original coordinate using the formula.

$$F_{orig} = T_b^T * F_{norm} * T_a$$

Step b: compute design matrix

For this step, I randomly selected 8 points and compute the matrix according to the lecture slide.

$$\begin{pmatrix} p_1q_1 & p_2q_1 & q_1 & p_1q_2 & p_2q_2 & q_2 & p_1 & p_2 & 1 \\ p_1q_1 & p_2q_1 & q_1 & p_1q_2 & p_2q_2 & q_2 & p_1 & p_2 & 1 \\ p_1q_1 & p_2q_1 & q_1 & p_1q_2 & p_2q_2 & q_2 & p_1 & p_2 & 1 \\ p_1q_1 & p_2q_1 & q_1 & p_1q_2 & p_2q_2 & q_2 & p_1 & p_2 & 1 \\ p_1q_1 & p_2q_1 & q_1 & p_1q_2 & p_2q_2 & q_2 & p_1 & p_2 & 1 \\ p_1q_1 & p_2q_1 & q_1 & p_1q_2 & p_2q_2 & q_2 & p_1 & p_2 & 1 \\ p_1q_1 & p_2q_1 & q_1 & p_1q_2 & p_2q_2 & q_2 & p_1 & p_2 & 1 \\ p_1q_1 & p_2q_1 & q_1 & p_1q_2 & p_2q_2 & q_2 & p_1 & p_2 & 1 \end{pmatrix}$$

Step c: perform an SVD of the design matrix to find its null space

I simply used the library, np.linalg.svd to perform SVD.

Step d: compose the draft fundamental matrix F

After performing svd, we get u, s, vh.
Where vh is a 9*9 matrix and the last row is null space.
Reshape the null space to get the draft fundamental matrix.

Step e: perform an SVD, set the smallest singular value to zero and reassemble

Perform a svd of the draft fundamental matrix to get u_F, s_F, vh_F.
Set s_F[-1] = 0 then reassemble F

```
F = (u_F * s_F) @ vh_F
```

Step f: calculate which correspondences are inliers and which are outliers

To decide which one is inlier and which one is not, we need to check whether the point lies on the epipolar line. If the point lies around the line within a reasonable range e.g. 1-2 pixels, then we decide this point is an inlier, otherwise it is an outlier.
Since we have computed the fundamental matrix F and we have all the 8 points, we can compute the corresponding epipolar line for each correspondence using the formula.
Correspondence: x and x'
$l = Fx'$

Then calculate the distance between the point and the line, if the distance is less than the threshold e.g. 2 pixels. Then we say the point is an inlier. For the method to calculate distance, I follow the one discussed on the Ed discussion board.

$$\hat{l}' = \frac{1}{\sqrt{a^2 + b^2}} \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$ scale the line then calculate distance $$\left| \hat{l}' \cdot p' \right|$$

However, in order to calculate the distance in the original pixel coordinate. We must transform the fundamental matrix F to the original coordinate then use the original point coordinate to calculate the distance.

$$F_{orig} = T_b^T * F_{norm} * T_a$$

Then we can calculate the distance and check whether a correspondence is an inlier or an outlier using the above method.

Step g: wrap in a RANSAC loop and run enough times to have probability > 99% of finding 8 inliers and computing a good F

To perform RANSAC, first we need to know how many iterations we are going to run. I found a formula on the website mentioned above.

$$\mathbf{k} = \frac{log(1-p)}{log(1-w^n)}$$ where k is the number of iterations, p is the probability, w is the proportion of inliers and n is the number of points. p = 0.99, n = 8 in this case.

We can see that w is unknown here, so we must make some assumptions. For example, assume 50% of the correspondences are inliers. Then we have k = $\frac{\log(1-0.99)}{\log(1-0.5\text{^}8)}$ = 1177.

With 50% inliers, we need to run around 1200 iterations. The less the inliers, the more iterations we need to run. For some of the pictures, we might need less than 1000 iterations since the inlier proportion is large. For other pictures where there are not many inliers, we might need 10000 iterations. So the number of iterations depend on the pictures, and I adjust the number of iterations for each picture set.

Step h: re-estimate F using all the inliers

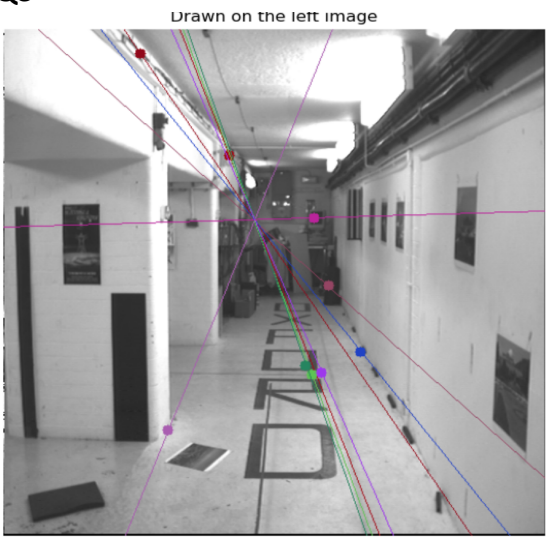
After doing RANSAC, we have found a good F and all the inliers using this F.
Then, we re-estimate F using these inliers instead of using random 8 points according to step a-e.

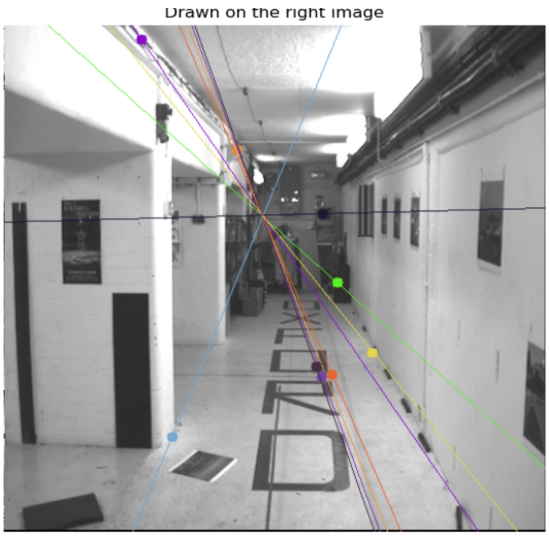
Step i: compute F in terms of the original pixel coordinate

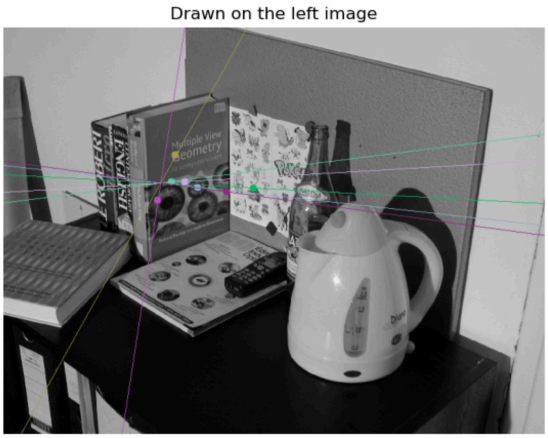
Finally, transform F in terms of the original pixel coordinate using the formula.

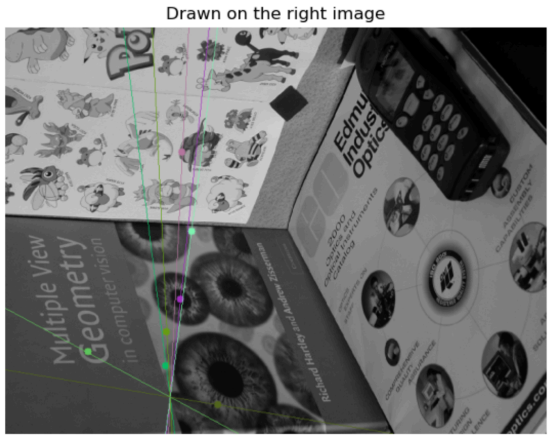
$$F_{orig} = T_b^T * F_{norm} * T_a$$

Q3

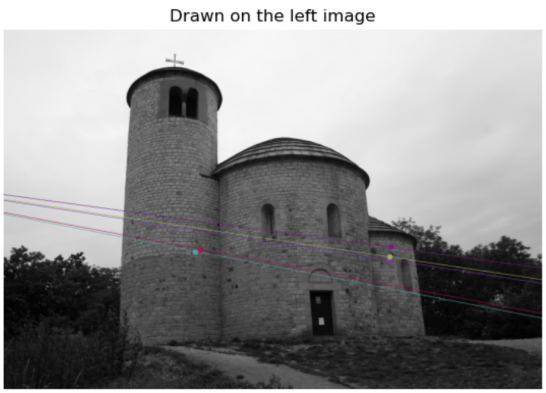


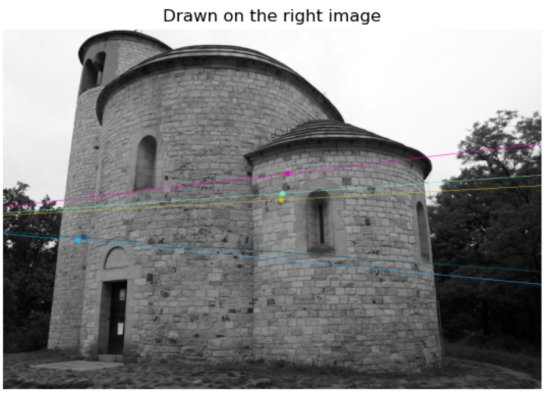


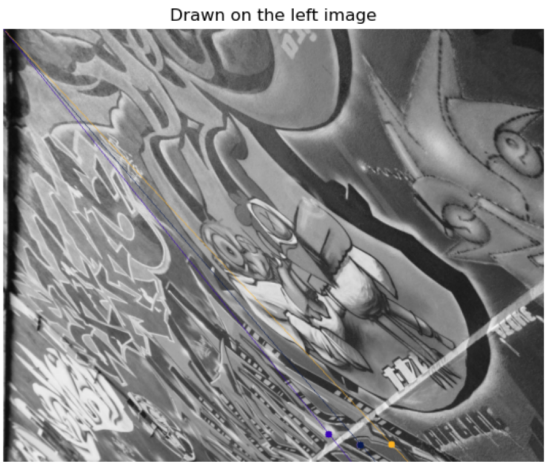




For images like the above two, the camera takes the pictures almost from the same angle which makes the proportion of inliers very high. So we only need a small number of iterations for RANSAC to get a good result.







For images like these two, the camera takes these pictures from different angles. And there are not many good correspondences between the images. The number of inliers is probably less than 10. So, we need a huge number of iterations in order to compute a good F.