Fundamental Matrix Estimation

Authors: Ierardi Ambra, Scorrano Andrea, Trovatello Alessandro, *Università degli Studi di Genova*

There will be only *one version* of the code and the report submitted, since the group worked together over every part of the assignment.

KEYWORDS: 8 points algorithm, ransacF, fundamental matrix

1 INTRODUCTION

This assignment is focused on finding the **foundamental matrix** given by the study of two stereo images. This tool plays a crucial role in computer vision, in the field of stereo vision. It is used to describe the geometric relationship between two camera views of a scene: it relates corresponding points in the two images and encapsulates the projective geometry of the imaging process. To find the fundamental matrix, two different algorithms were implemented: the **8 points algorithm** and the **RANSAC**. The 8-point algorithm is a method for estimating the fundamental matrix F using at least 8 corresponding points in two images. Two versions of this method were implemented: using normalized and non-normalized points. More promising results are expected with the normalized set of points. RANSAC (RANdom SAmple Consensus) is a robust statistical method used to estimate the fundamental matrix from a set of data points contaminated with noise.

2 REPORT TASKS

2.1 8 points algorithm

The *8 points algorithm* is a procedure in computer vision that allows computation of the **fundamental matrix** form a stereo acquisition of an image, meaning that the same scene has been acquired by two different points of view by the same camera, or by two different cameras. The starting point of this process is the choice of the correspondences, that, in this case, were already given, inside the files *Rubik1.points*, *Rubik2.points*, *Mire1.points* and *Mire2.points*. From these couples of points the **affinity matrix** was built, and, using the *singular value decomposition*, the *fundamental matrix* was obtained, forcing its rank to be equal to two, to have a corresponding line in the second image, instead of a point.
Two different versions were tested: one with the couples of points as they were given, and another one taking care of the normalization of the points, with the function **normalise2dpts.m**, before running the same algorithm and, as last point, denormalizing the resulting *fundamental matrix* with the *transformation matrices* of the two sets of points, respectively T_1 and T_2, in the following way:

$$F = T_2^T \cdot F \cdot T_1 \tag{1}$$

2.2 Fundamental matrix estimation with Ransac

The second point to cover in this assignment was finding the **foundamental matrix** using the given *Ransac* algorithm. The *Ransac* algorithm requires as input the matches between the two images. In order to find the matching points, the provided function **findMatches.m** was used, which can exploit two different algorithms: NCC, which compares features considering patches and the euclidean distance between points, and SIFT, that is more suitable to deal with scaling and rotation, more severe view-point changes. Once obtained the matches, the *Ransac* algorithm selects randomly a subset of the M pairs, uses the 8 points algorithm to estimate the fundamental matrix, computes the error obtained by applying F to all pairs, detects inliers, which are the points with a residual of the epipolar constraint under a chosen threshold, and outliers, the points which do not fulfill the previous condition. This algorithm must be repeated for k iterations, with k given by:

$$K = \frac{\log(1-p)}{\log(1-\omega^M)} \tag{2}$$

where p is the probability of obtaining a set of M good pairs (inliers) and ω^M are the best M inliers.

3 RESULTS

3.1 8-point algorithm function

The use of the two different pair of images provides different results. In the case of the *Rubik* images, the original set of correspondences gave as result the following *fundamental matrix*:

$$\mathrm{F} = \begin{bmatrix} 7.350 \cdot 10^{-8} & 1.263 \cdot 10^{-6} & 0.001 \\ 2.527 \cdot 10^{-6} & -3.562 \cdot 10^{-6} & 0.118 \\ -0.004 & -0.119 & 0.985 \end{bmatrix}$$

The correctness of the resulting matrix can be evaluated checking if the *epipolar constraint* is satisfied, with the following operation:

$$x_2^T \cdot F \cdot x_1 = 0 \tag{3}$$

where x_1 is the set of points from the first image and x_2 from the second one. This equation, in real applications, is never satisfied, in terms of accuracy: the resulting vector will never be exactly made of zeros, but, if the values are small enough, this constraint can be considered satisfied.
With these data and this matrix, the resulting vector has a mean of -0.3384, which means that it is not much accurate. Despite that, it is possible to notice in the following two images that the epipolar lines are quite precise, and the matches too are good enough:

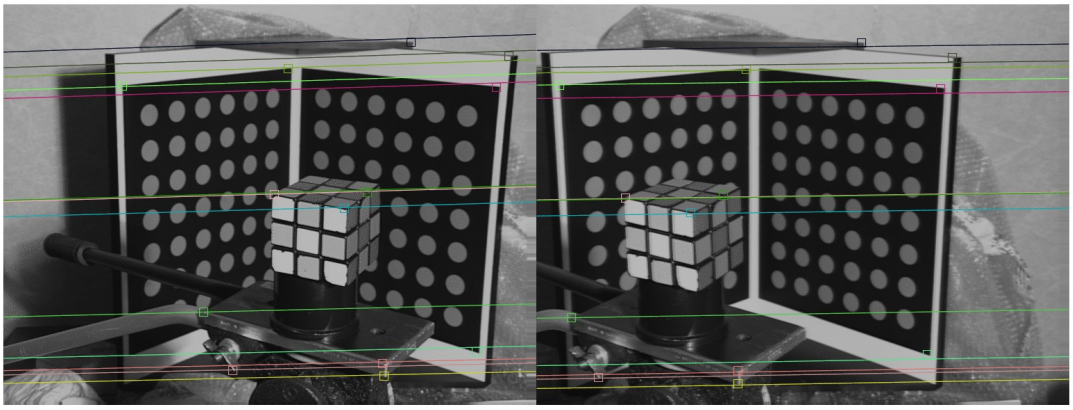

Figure 1: Epipolar lines with not normalized points, Rubik images

On the other hand, the normalized set of points of the *Rubik* image gives this *fundamental matrix*:

$$\mathrm{F} = \begin{bmatrix} -2.074 \cdot 10^{-9} & -4.701 \cdot 10^{-7} & 7.458 \cdot 10^{-5} \\ 3.842 \cdot 10^{-7} & -2.919 \cdot 10^{-8} & -0.004 \\ -4.129 \cdot 10^{-5} & 0.004 & -0.007 \end{bmatrix}$$

The mean of the vector resulting from the evaluation of the epipolar constraint is $8.463 \cdot 10^{-4}$, which is three order of magnitude smaller than the one given by the not normalized set of points, as expected because normalization involves scaling and centering the points to a standard scale, which can lead to more stable and accurate results.
Also, it is possible to notice that these epipolar lines are passing exactly through the small squares, that are the matching points, while in the previous case there was a small error, still visible: this is quite evident in the top right match, which is in a black color in *Figure 1* and in light blue in *Figure 2*.

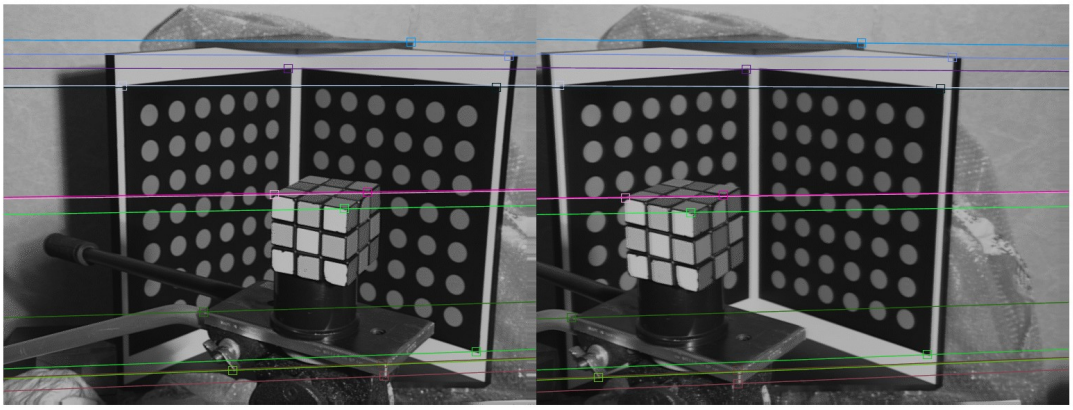

Figure 2: Epipolar lines with normalized points, Rubik images

In the case of the *Mire* images, the original set of points give the resulting *fundamental matrix*:

$$\mathrm{F} = \begin{bmatrix} -2.080 \cdot 10^{-6} & 2.143 \cdot 10^{-6} & -0.003 \\ -2.024 \cdot 10^{-6} & 8.223 \cdot 10^{-8} & 0.002 \\ 0.005 & -0.002 & -0.99 \end{bmatrix}$$

The mean of the epipolar constraint vector is 0.0194, which is an order of magnitude less with respect to the previous set of points, but it is possible to notice in the *Figure 3* that the epipolar lines are almost vertical, which may be due to low quality of the image.

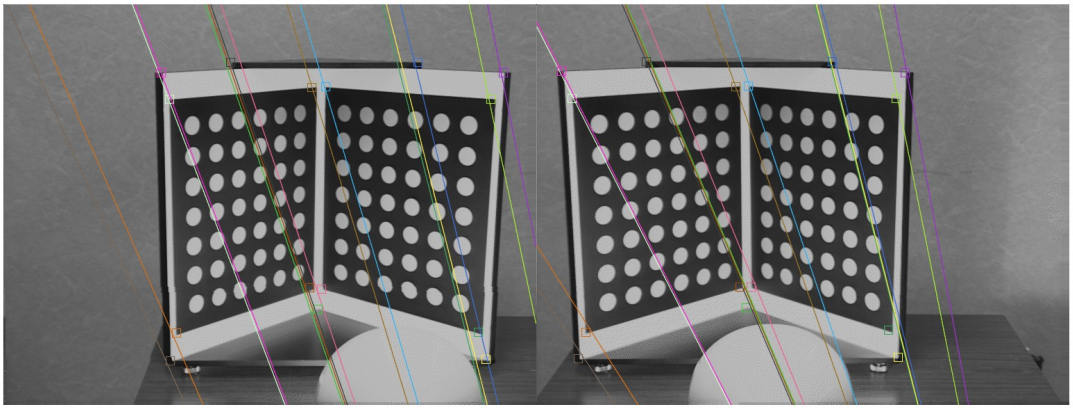

Figure 3: Epipolar lines with not normalized points, Mire images

Regarding the normalized set of points, the resulting *fundamental matrix* has the following values:

$$\mathrm{F} = \begin{bmatrix} -4.203 \cdot 10^{-9} & 1.755 \cdot 10^{-7} & -2.977 \cdot 10^{-4} \\ -1.124 \cdot 10^{-7} & 7.269 \cdot 10^{-8} & 0.004 \\ 2.850 \cdot 10^{-4} & -0.004 & -0.035 \end{bmatrix}$$

The mean of the epipolar constraint vector is 0.0044, which is an order of magnitude lower than the one resulting from the not normalized set of points, as expected.

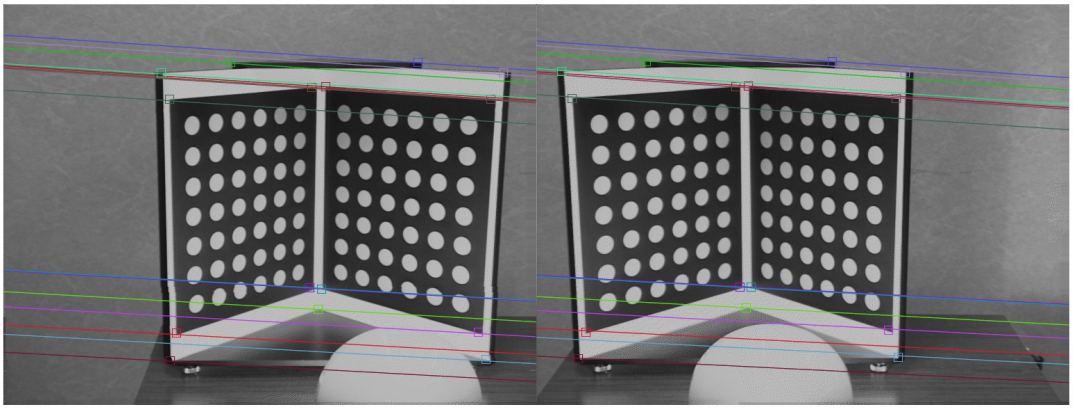

Figure 4: Epipolar lines with normalized points, Mire images

3.2 Ransac method

3.2.1 Assignment functions, Mire images

The results of the **ransacF.m** function depends to the result of the **findMatches.m** function; so, if the matches between images are bad, consequently the result of the *Ransac* will not be good. In the following images are shown the results of the **findMatches.m**, computed with the SIFT method, on the *Mire* images:

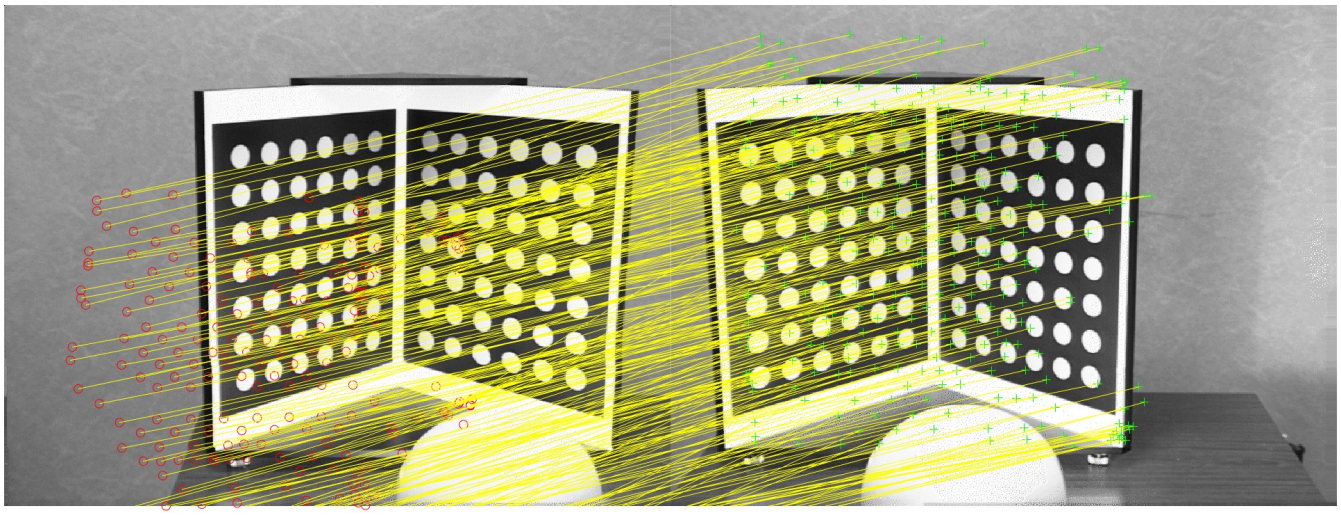

Figure 5: Matches between Mire images, computed by **findMatches.m**

As it is possible to note, the matches are so bad that it's difficult to find one correct match. Same result with the **findMatches.m** function computed with NCC method.
Once the **ransacF.m** has been computed with these matches, the resulting fundamental matrix has the following values:

$$F = \begin{bmatrix} -3.191 \cdot 10^{-9} & -4.883 \cdot 10^{-7} & 2.551 \cdot 10^{-4} \\ 3.711 \cdot 10^{-6} & -2.359 \cdot 10^{-6} & -0.005 \\ -0.001 & 0.005 & -0.785 \end{bmatrix}$$

and the mean of the epipolar constraint vector is -0.0063. These values have to be examined with a certain attention because the fundamental matrix changes every time the **ransacF.m** function is called, due to the random choice of the 8 points, to find the *bestF*. The epipolar lines drawn by **ransacF.m** are as follows:

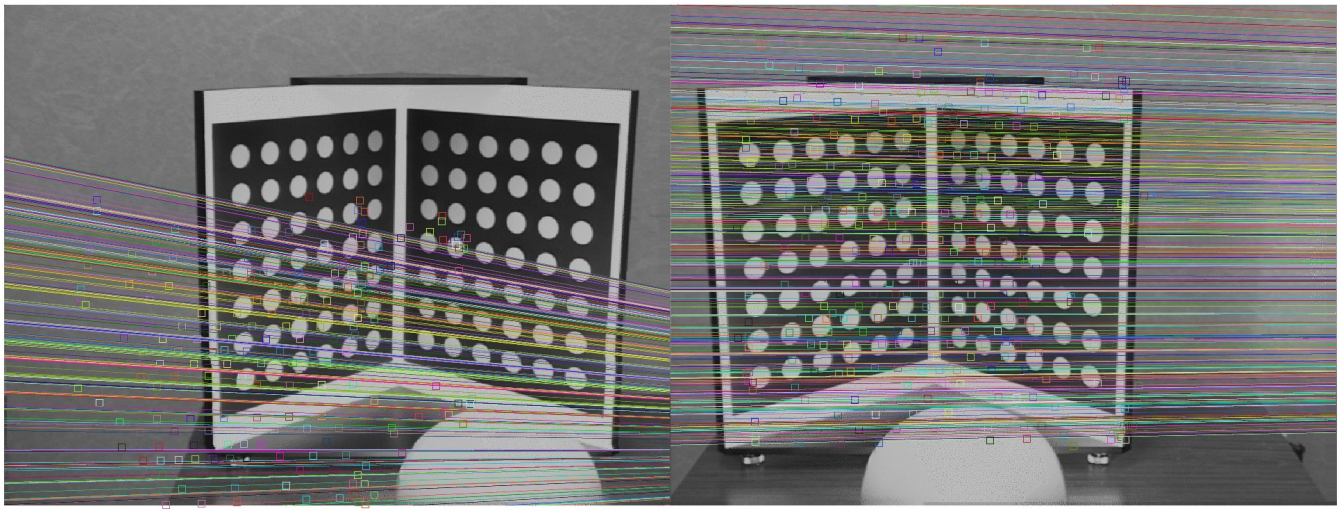

Figure 6: Epipolar lines on Mire images, resulting from **ransacF.m**

As expected, the epipolar lines are very wrong due to the matches found by **findMatches.m**.

3.2.2 Assignment functions, Rubik images

Now, if the input is changed with Rubik images, the resulting matches and the relative *Ransac* don't change much, take a look at the top right group of matches: the distance between those ones and the right corner of the object is visibly different.

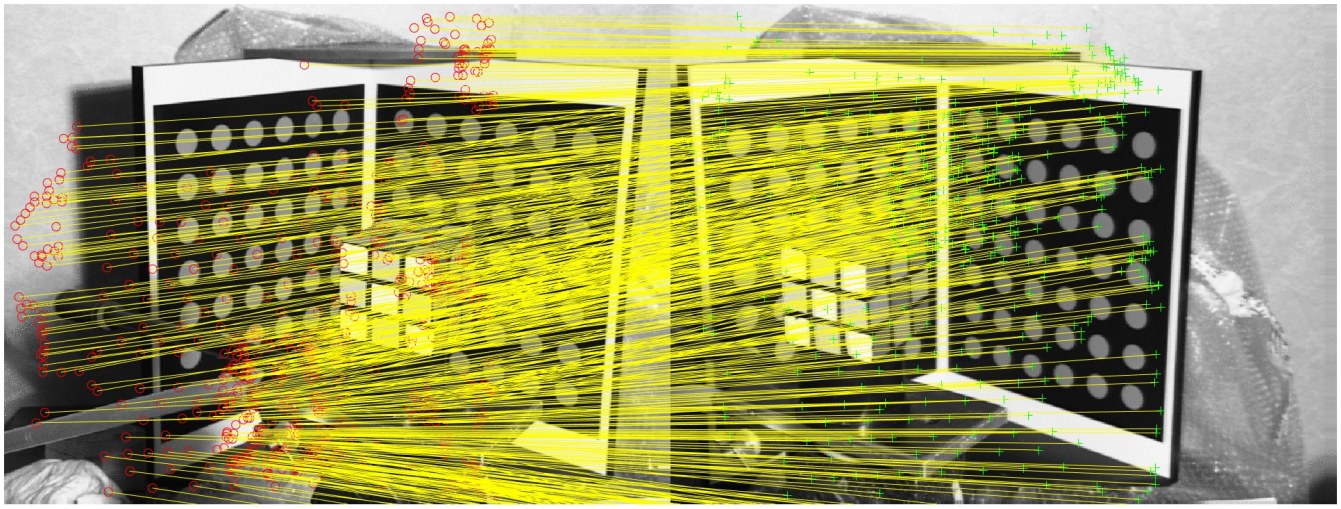

Figure 7: Matches between Rubik images, computed by **findMatches.m**

Calculating the **ransacF.m** with these matches, the resulting fundamental matrix is as follows:

$$F = \begin{bmatrix} 1.751 \cdot 10^{-7} & 1.395 \cdot 10^{-6} & -0.001 \\ -1.006 \cdot 10^{-6} & -1.160 \cdot 10^{-5} & 0.008 \\ 7.188 \cdot 10^{-4} & -0.001 & -0.317 \end{bmatrix}$$

The mean of the epipolar constraint vector is 0.0034. The epipolar lines are the following ones:

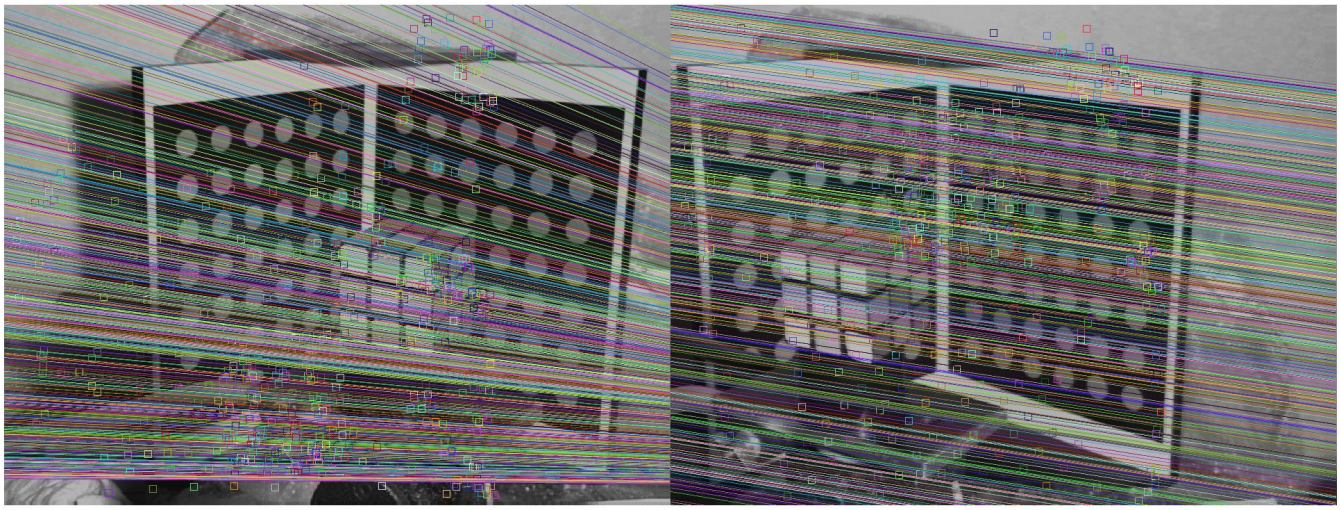

Figure 8: Epipolar lines on Rubik images, resulting from **ransacF.m**

As expected, the epipolar lines are very bad due to matches found by **findMatches.m** also with these input images.

3.2.3 Built-in MATLAB functions, Rubik images

Alternatively, it is possible to use the built-in MATLAB functions to find good matches between images, but unfortunately they cannot get over the check of **ransacF.m**, so it was used another built-in *Matlab* function to compute the *Ransac* algorithm. These functions are: **extractFeatures** and **matchFeatures**, to find matches, and **estimateFundamentalMatrix** with 'RANSAC' method, to compute the *Ransac* algorithm.
Repeating the same steps as before with the built-in functions, the matches found by built-in functions in Rubik images are as follows:

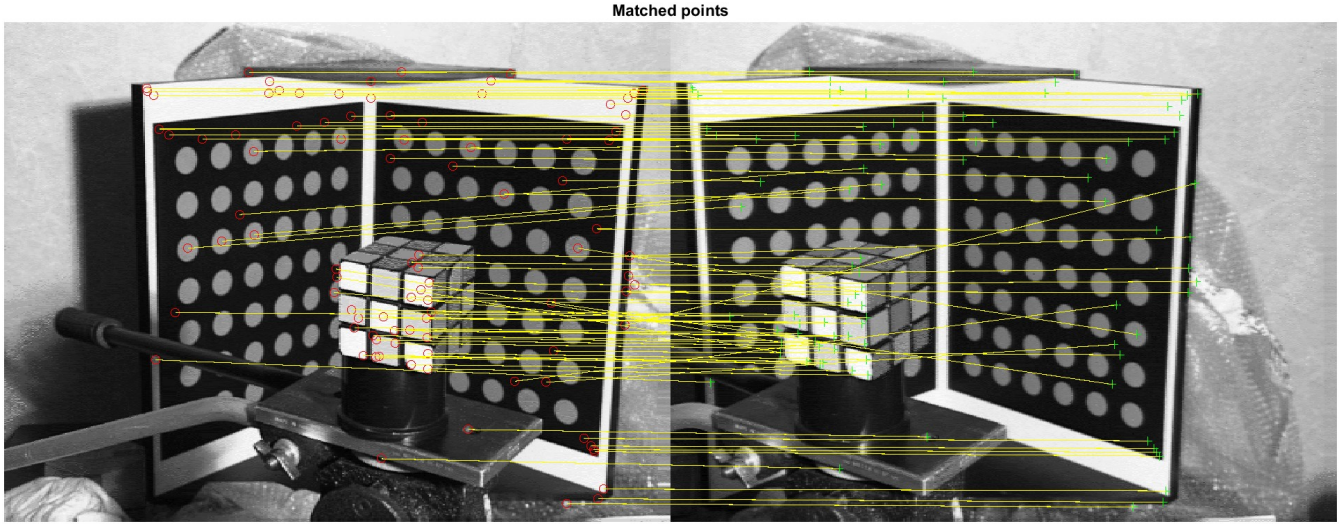

Figure 9: Matches between Rubik images, computed by **extractFeatures** and **matchFeatures**

As it is possible to note, there are much more good matches. Once the **estimateFundamentalMatrix** with Ransac method has been computed, the resulting fundamental matrix has the following values:

$$F = \begin{bmatrix} 2.357 \cdot 10^{-7} & 7.374 \cdot 10^{-5} & -0.012 \\ -6.117 \cdot 10^{-5} & 7.145 \cdot 10^{-6} & 0.605 \\ 0.008 & -0.611 & 0.629 \end{bmatrix}$$

The epipolar lines computed by **estimateFundamentalMatrix** are as follows:

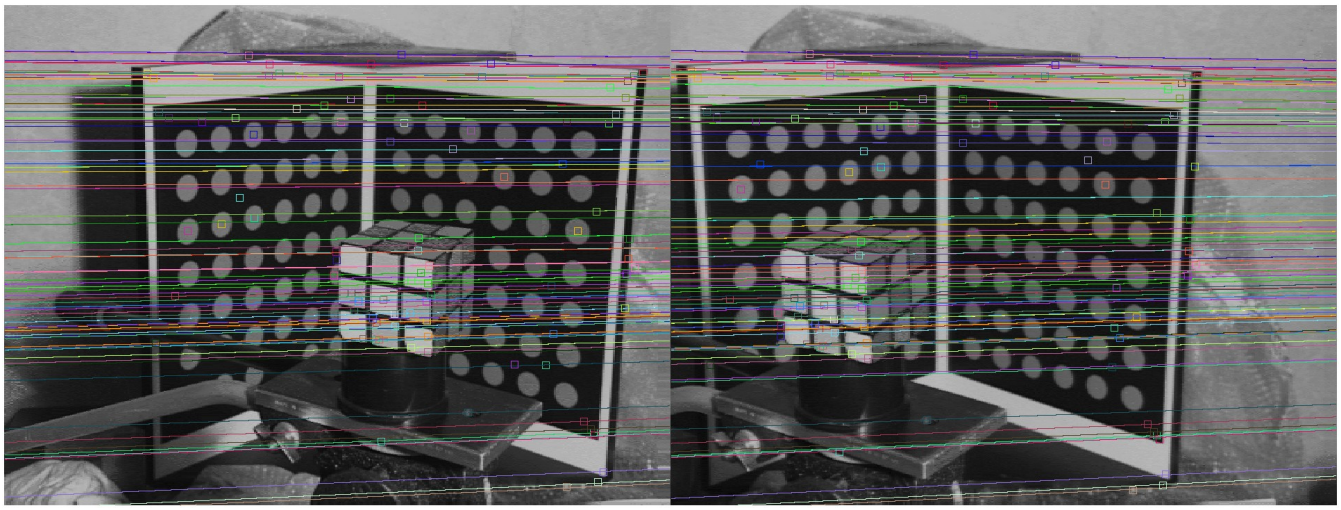

Figure 10: Epipolar lines on Rubik images, resulting from **estimateFundamentalMatrix**

It is possible to note that it is definitely better than **ransacF.m** function.

3.2.4 Built-in MATLAB functions, Mire images

The following matches are obtained giving the Mire images as input, using the same built-in functions previously mentioned:

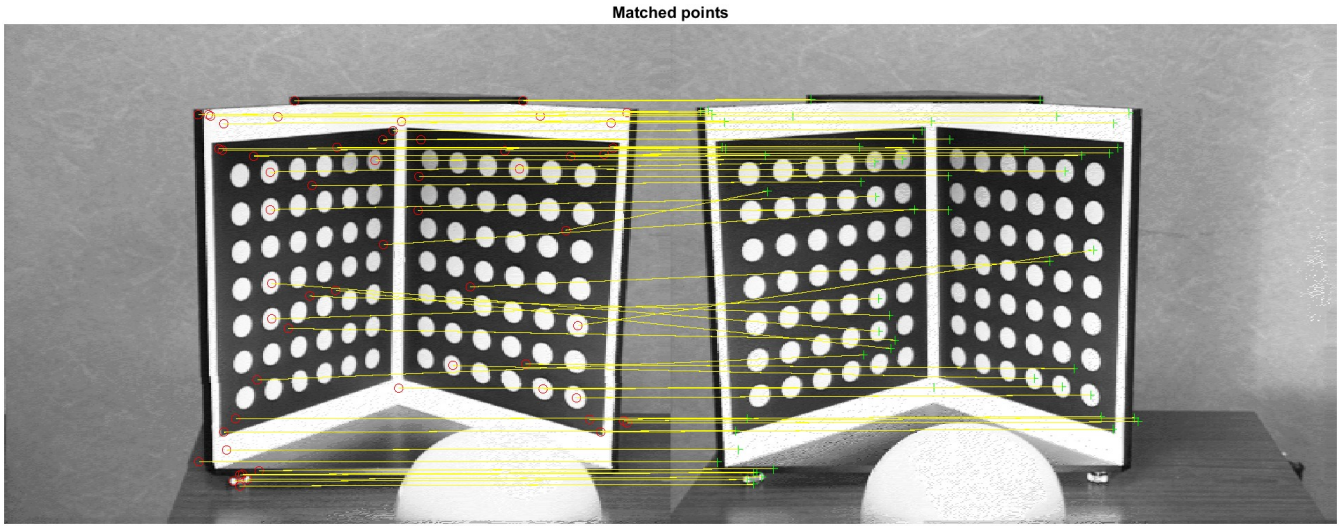

Figure 11: Matches between Rubik images, computed by **estimateFundamentalMatrix**

As in the previous images, it is possible to find more good matches than the bad ones, and the fundamental matrix computed by the **estimateFundamentalMatrix** has the following values:

$$F = \begin{bmatrix} -2.500 \cdot 10^{-7} & -3.229 \cdot 10^{-5} & -8.057 \cdot 10^{-4} \\ 3.897 \cdot 10^{-5} & 3.738 \cdot 10^{-7} & 0.261 \\ -9.463 \cdot 10^{-4} & -0.269 & 0.929 \end{bmatrix}$$

The epipolar lines are the following:

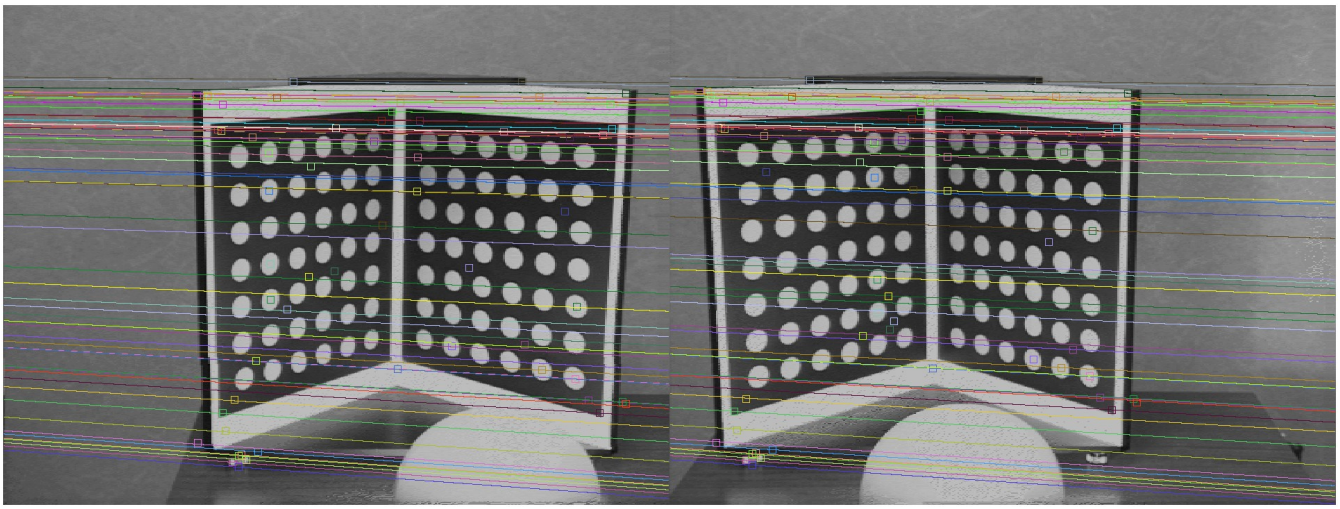

Figure 12: Epipolar lines on Mire images, resulting from **estimateFundamentalMatrix**

As before, the results are better than the ones obtained with **ransacF.m**.

4 SUMMARY

This assignment has the focus on the fundamental matrix estimation. To see the correctness of the resulting matrix, it was checked whether the epipolar constraint was satisfied. This constraint was verified, but it wasn't perfectly satisfied, as happens in real applications , there is always an error due to noise or bad matches. The results show that the 8-point algorithm works better using the normalized points. The given *Ransac* method doesn't find good fundamental matrices if doesn't have a good matches, found by the provided function **findMatches.m**. Using the built-in *Matlab* functions to look for the matches and to compute the fundamental matrix, it's possible to obtain good results. It's important to note that the *Ransac* will never returns the same fundamental matrix, due to the random permutation of chosen points used every time.