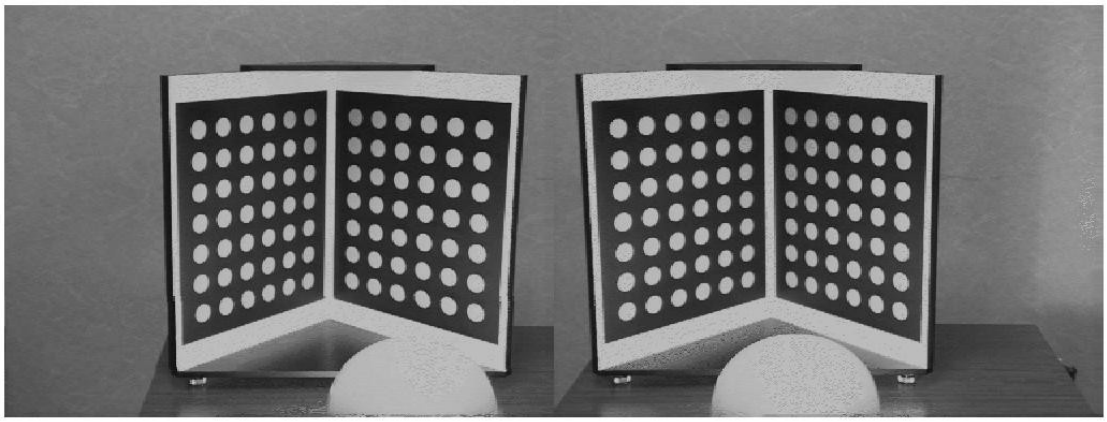
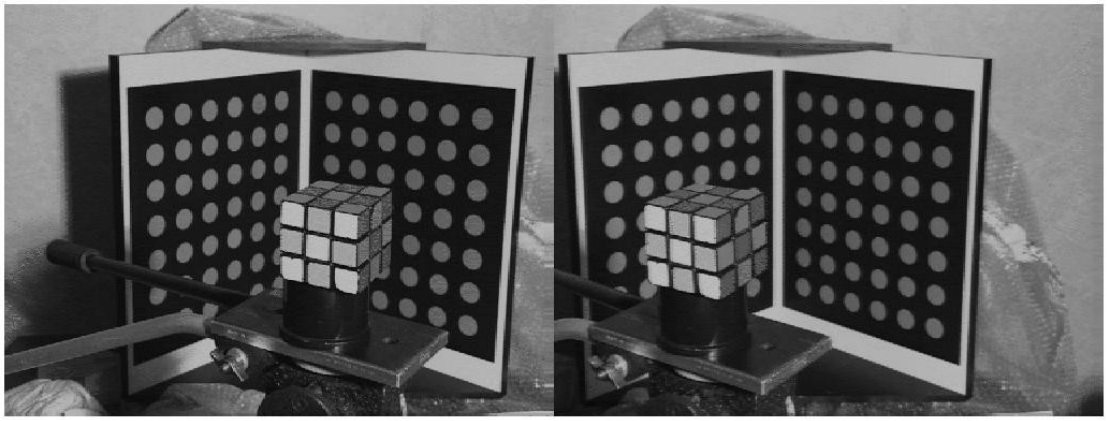
Computer Vision Lab session n. 6
15/05/2018

Fundamental matrix estimation

The aim of this lab session is to implement the 8 point algorithm to estimate the fundamental matrix F which is needed to put in relation cameras, projected points and real 3D point. We use two version for this, one normalized and other not normalized.





Davide Torielli
Luca Morando
Fabio Fusaro

First Version (EightPointsAlgorithm.m)

First thing is to compute the A matrix, to solve the system Af = 0, where f is the matrix wanted F disposed as column vector. For compute the products of A we need almost 8 correspondence points because F has 8 dof (is a 3x3 matrix but one element is dependent because we are working with homogeneous coordinates. Actually with constraint of determinant not zero real dof are 7 but this would requires another more complicated algorithm).
To solve the equation we use singular value decomposition on A : [U, D, V] = svd(A). The solution f is the last column of V, that can be easily reshaped to form the 3x3 matrix F.
Af = 0 is a overdetermined problem, which means it has more solutions. So we find the one that minimizes a cost function. To solve, we force the rank of F to be zero:
- We compute the svd on F to find [U, D, V] = svd(F).
- The diagonal matrix D has value in descending order, so a solution which minimize the cost is the one with D(3,3) = 0.

Finally we calculate the F as $F = U*D*V^T$

Second Version (EightPointsAlgorithmN.m)

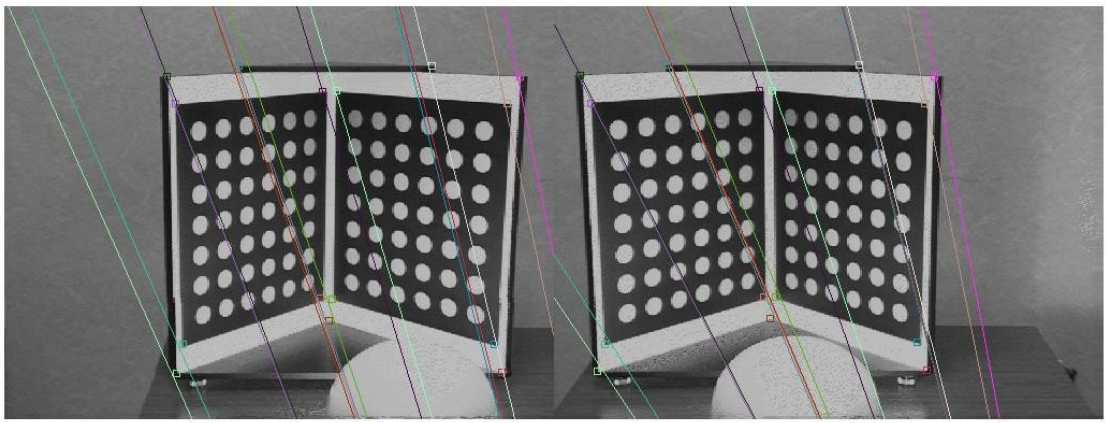
This is the same as previous, only with normalized points. We use the normalise2dpts script, compute A with these normalized point, and at the end “de-normalize” the final F.

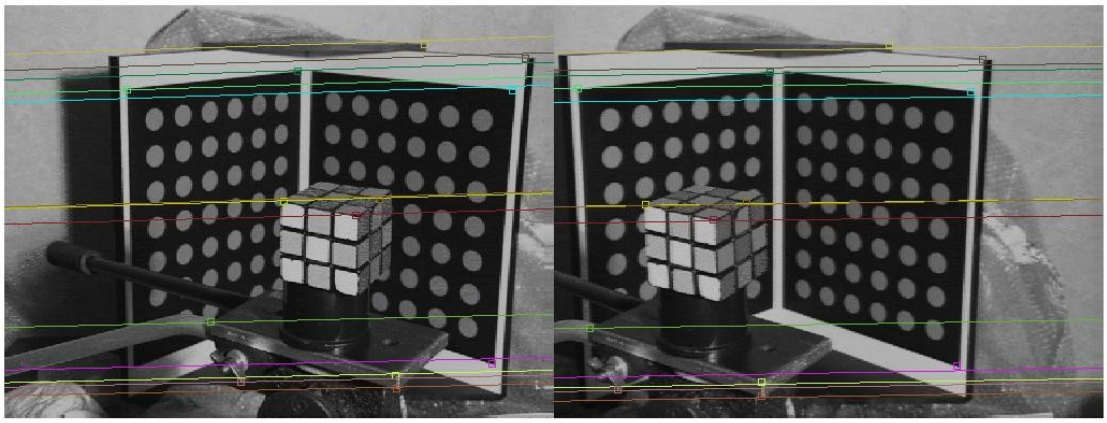
The calculation of F is good if $x'^T * F * x$ is near to zero, where x’ are the points in the second image plane and x are the points in the first one.
We see that for both couple of images this result is closer to zero when we use the normalized version.

Davide Torielli
Luca Morando
Fabio Fusaro

Here we see the epipolar lines of the corresponding points provided

Method 1

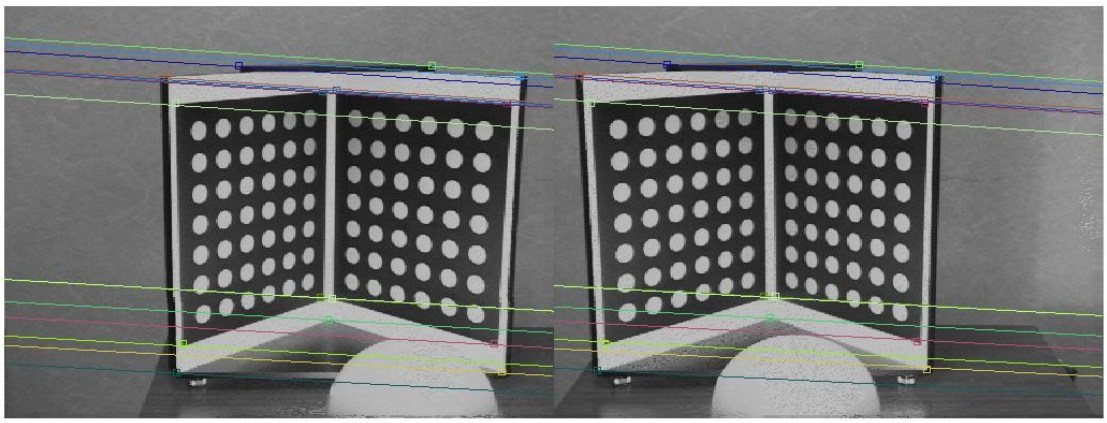


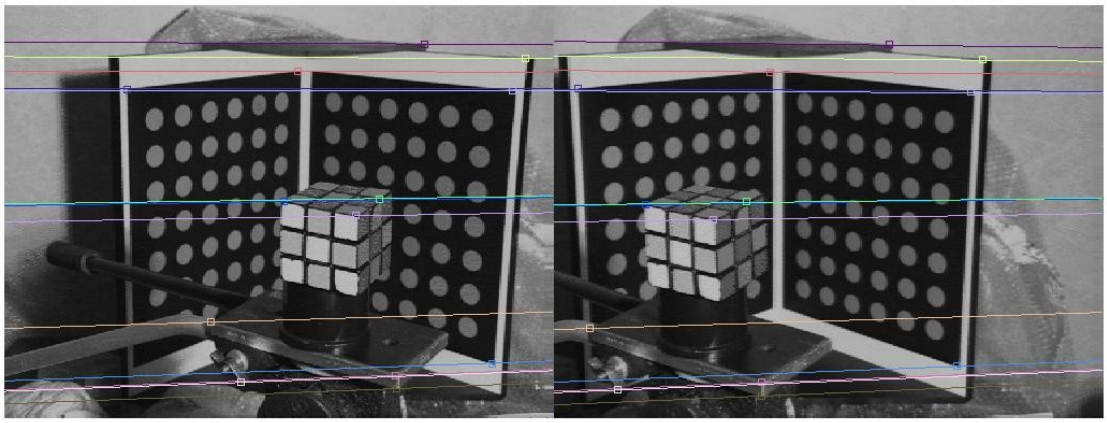


For the Rubik images (second pairs) result is not so bad. But for the first pairs result isn't acceptable. This means that it is a more "difficult" pair and it needs a better algorithm: the normalized version.

Davide Torielli
Luca Morando
Fabio Fusaro

Method 2





We can see here that the normalized version is better.

The normalized version results are better because the magnitude of products in A matrix are more similar.

Davide Torielli
Luca Morando
Fabio Fusaro

The left and right epipoles are, respectively the last column of V and U, because they are the right and left null space of F.

MIRE images
left epipole : version1 : [-0.41; -0.91; $-3.2*10^{-4}$] version2: [-1; -0.07; $-2.5*10^{-5}$]
right epipole : version1: [0.45; 0.89; $5.5*10^{-4}$] version2: [1; 0.07; $4.3*10^{-5}$]

RUBIK images
left epipole : version1 : [1; -0.03; $-2.2*10^{-5}$] version2:[-1; -0.01; $-8.7*10^{-5}$]
right epipole : version1 : [1; -0.01; $-1.1*10^{-5}$] version2:[-1; -0.02; $-1.1*10^{-4}$]

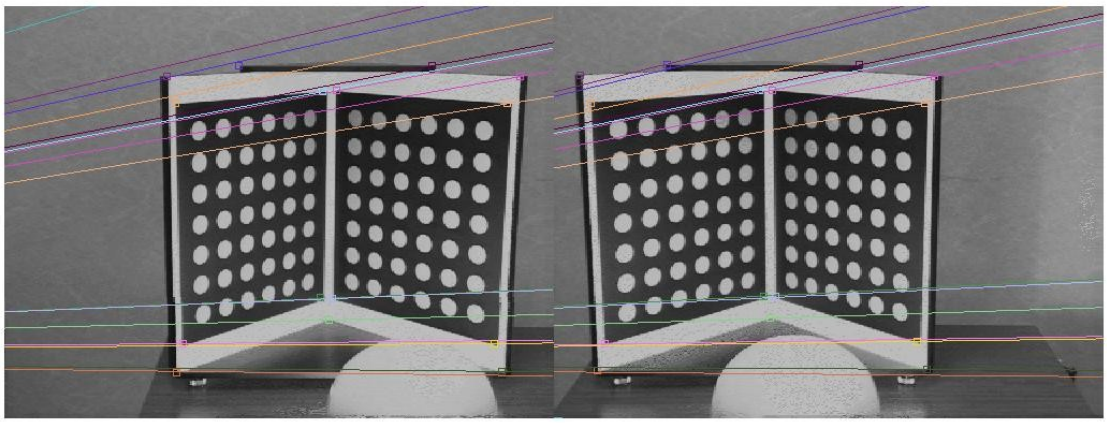
Note the third component: it is almost zero, which means that point are almost at infinite that is reasonable because epipolar lines are almost parallel.
If we divide all 3 coordinate for the last element we have epipoles in real homogeneous coordinates (with the last element equals 1) and we can appreciate better that the epipoles are far away from the plane images; this because the image planes are almost parallel.

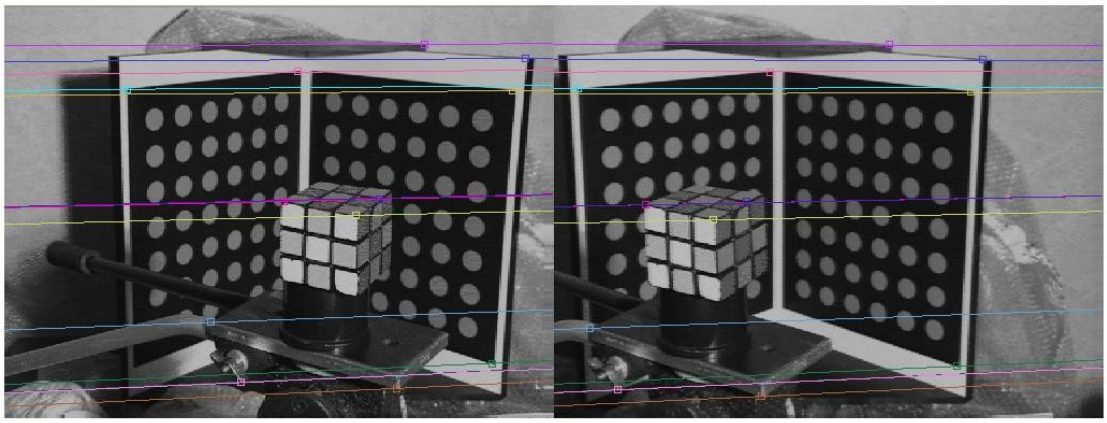
MIRE images
left epipole : version1 : [$1.3*10^{3}$; $2.8*10^{3}$; 1] version2: [$3.9*10^{4}$; $2.6*10^{3;}$ 1]
right epipole : version1: [819.9; $1.6*10^{3}$; 1] version2: [$2.3*10^{4}$; $1.7*10^{3}$; 1]

RUBIK images
left epipole : version1 : [$-4.4*10^{4}$; $1.5*10^{3}$] version2:[$1.1*10^{4}$; 108.1]
right epipole : version1 : [$9.1*10^{4}$; $-1*10^{3}$] version2:[$9.4*10^{3}$; 158.3]

Davide Torielli
Luca Morando
Fabio Fusaro

Matlab Version (estimateFundamentalMatrix)





Note : Both images are result of method RANSAC

We try to use this function provided by matlab. We add one point to the ones provided for the images.
We give a very erroneous correspondence point ([600,600] [0, 0]) to see what happen.
With method “Norm8Point” (that is the one which normalized the point as we do in version 2) results are very wrong because it need that the correspondences are very good.
With “RANSAC”, instead, results are better because this statistic method finds the outlier couples and not count them for the estimation of fundamental matrix. In this case however it detects as outliers not only the point added but also others, for this the MIRE (which is the “difficult” image) lines are different from our results.

Davide Torielli
Luca Morando
Fabio Fusaro