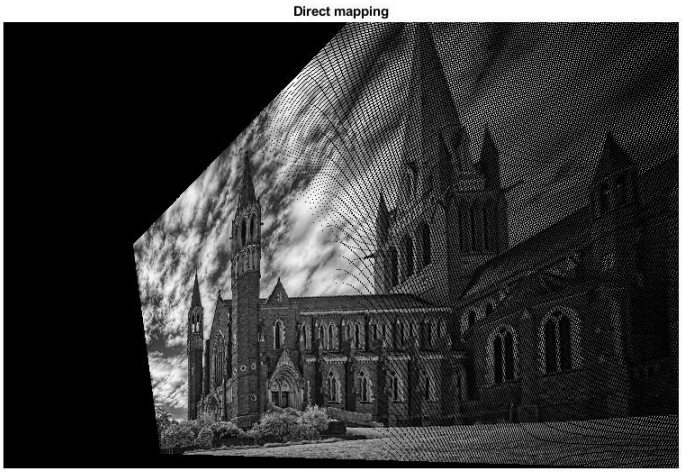
Computer Vision Lab session n. 8
04/06/2018

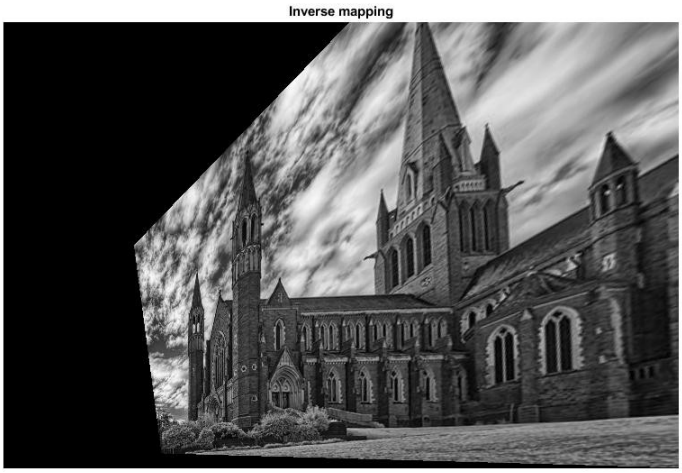
Playing with homographies

Correcting the perspective distortion of an image

With the provided code, we select 4 couples of point to correct the distortion and make the subject of the image flat, viewing it in front. The above one is a difficult case: we can't correct distortion of every areas because there are a lot of different planes. We choose to correct the distortion for the lateral face of the cathedral.

Fabio Fusaro
Davide Torielli
Luca Morando





The direct mapping is full of black lines; in fact this is a direct transformation which means that for each pixel in the original image we find a correspondence pixel in the transformed image. So some pixels in the transformed one will have no value. These black lines are more evident in the areas where the pixels are shifted more.

Instead in the inverse mapping each point has a correspondence because we start computing the result from the pixels of the transformed image.

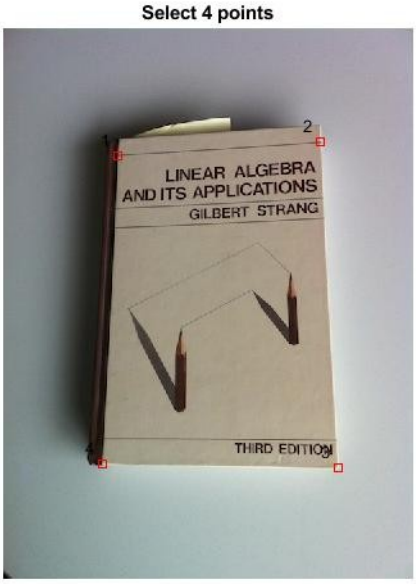
Fabio Fusaro
Davide Torielli
Luca Morando

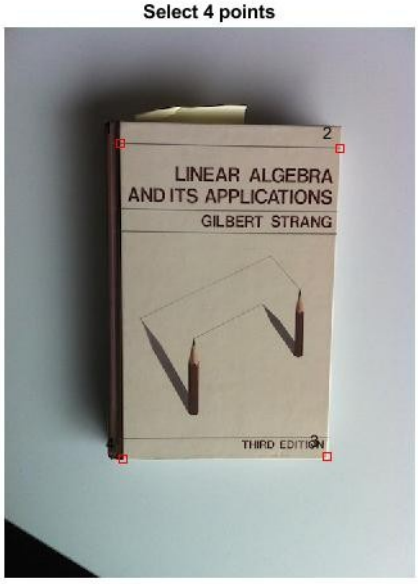
Bilinear interpolation

In addition, in the inverse code there is the bilinear interpolation which makes the image more uniform (less “grainy”) as we can see above (the points where we can detect this more is between the sky and the upper building).
This is because pixels will have a colour more similar to the near ones, thanks to the interpolation.

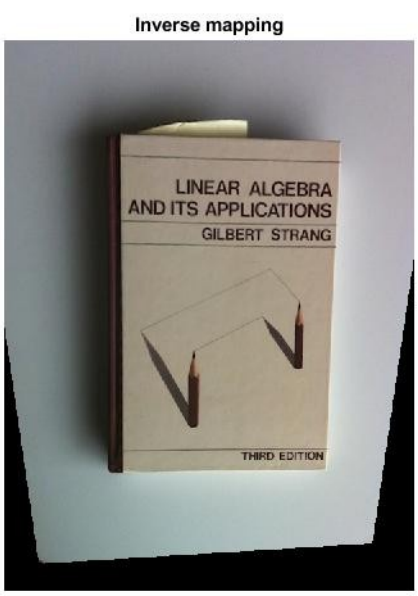
Fabio Fusaro
Davide Torielli
Luca Morando

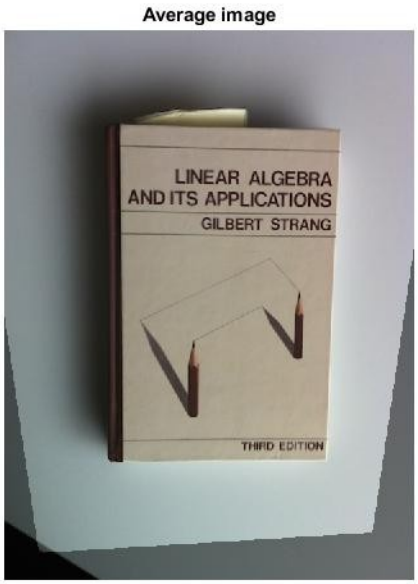
Estimating the homography between a pair of images





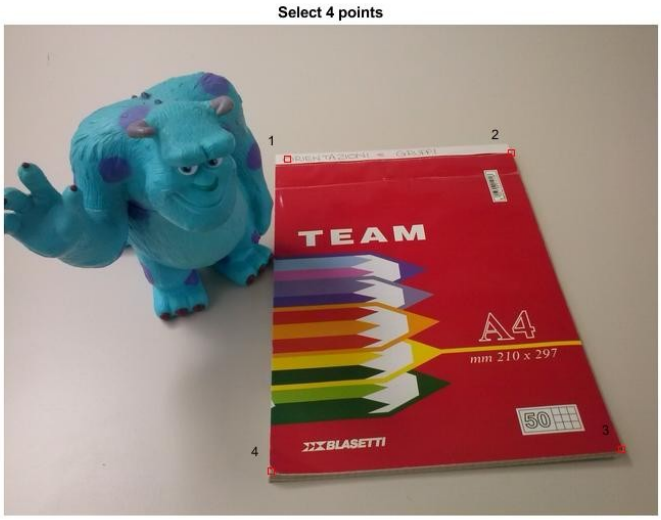
Here we choose couples of point in different stereo images. With these correspondence, the code estimate the homography matrix H (as in the previous problem) and try to modify the first image to make it equals to the second, through a inverse mapping (same inverse mapping function as problem 1).

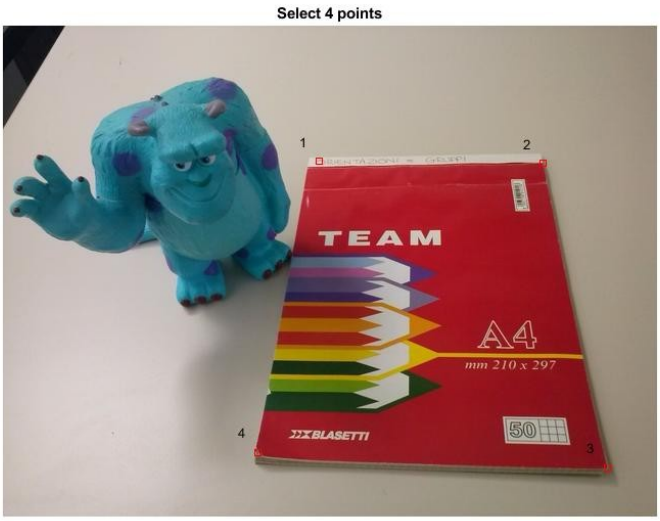




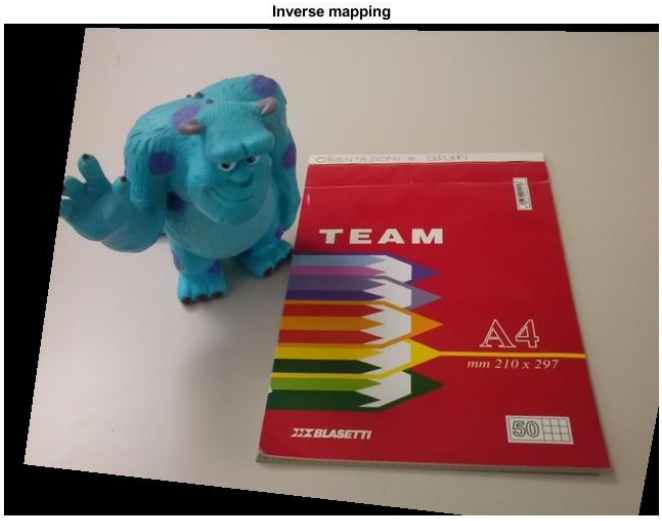
Fabio Fusaro
Davide Torielli
Luca Morando

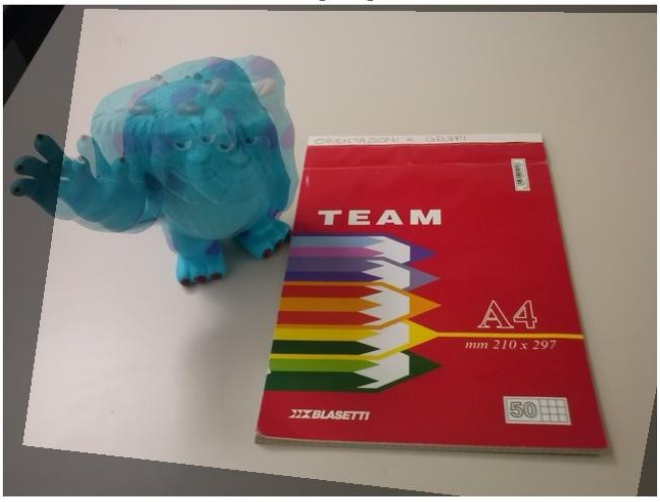
No 2D image





To estimate the homography we need an image which represent (almost) flat thing (2D image).
In fact we see that choosing the couple above the results are bad :



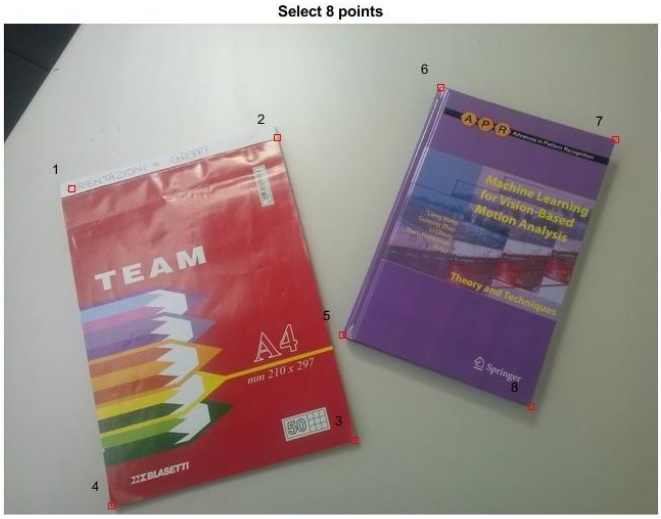


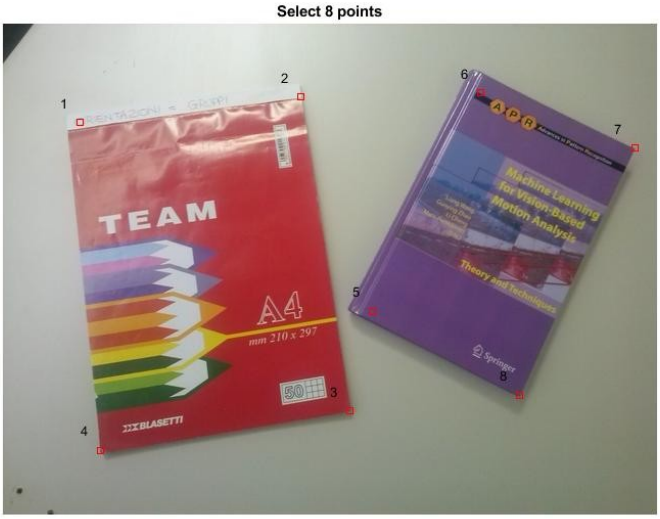
Things would not be better if we add more correspondence points, even using RANSAC method which would discard possible bad correspondences.

Fabio Fusaro
Davide Torielli
Luca Morando

RANSAC Method

The RANSAC is a statistic method which computes the H matrix different times with different correspondences (each time it takes 4 couples because 4 are the minimum to compute the homography matrix H, because H has 4 degrees of freedom) and choices the most suitable H according to a given threshold.
Obviously the number of points chosen must be greater than 4, otherwise the choice of which couples to select is unique and we can't see the differences from the normal method.

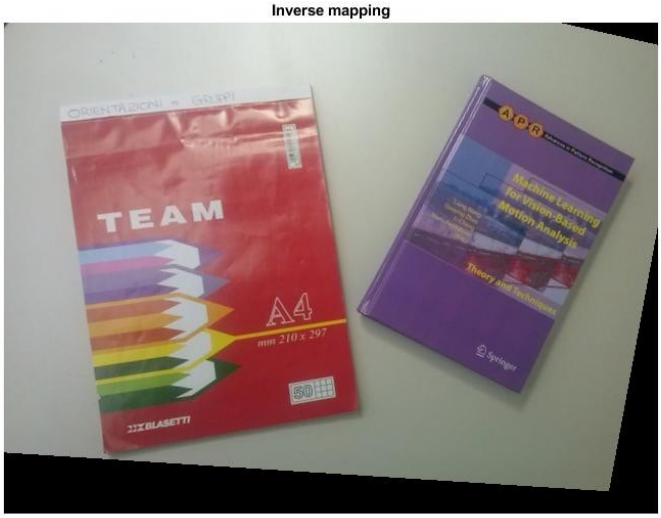




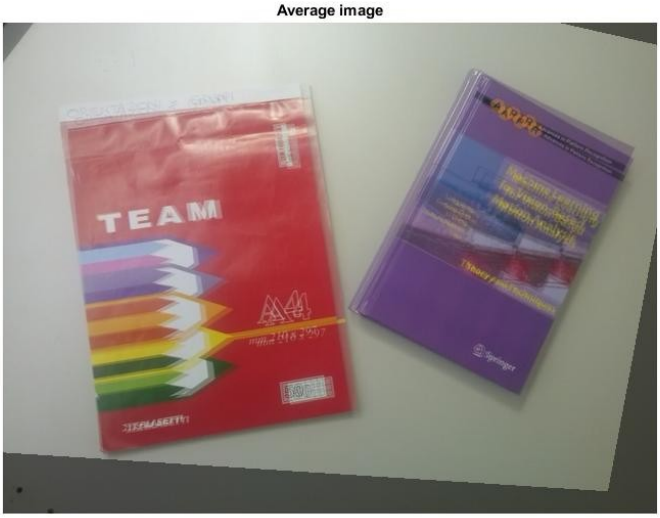
We try this code adding more point (8 point in total) give some (2) bad correspondence.

Fabio Fusaro
Davide Torielli
Luca Morando

Inverse mapping

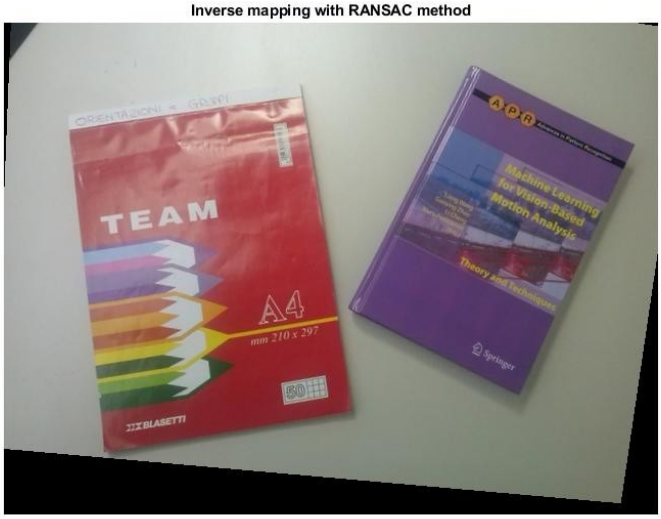


Average image

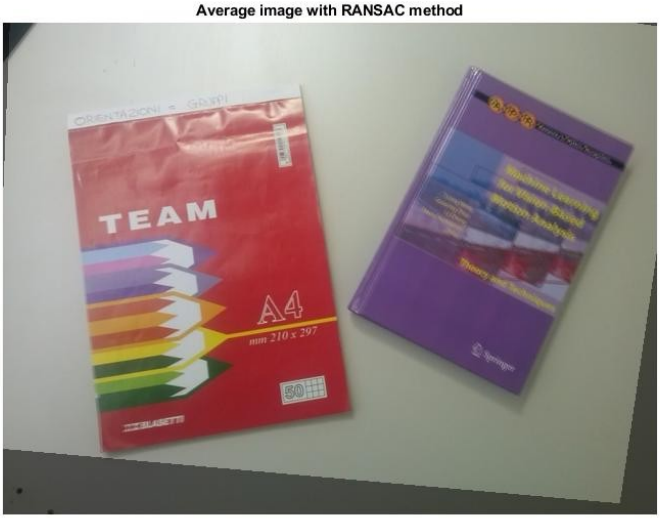


Above we see the result without ransac: the bad correspondence points are take into consideration and result are not so good.

Inverse mapping with RANSAC method



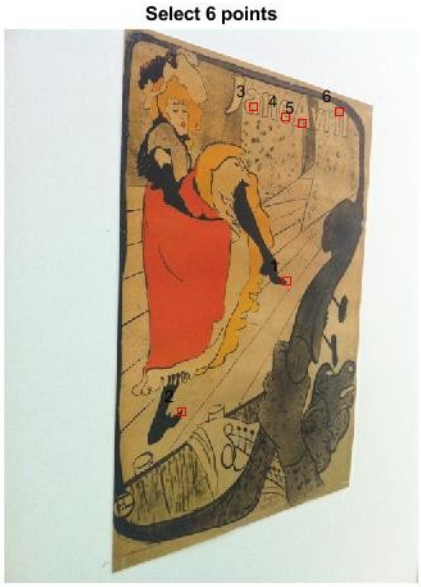
Average image with RANSAC method



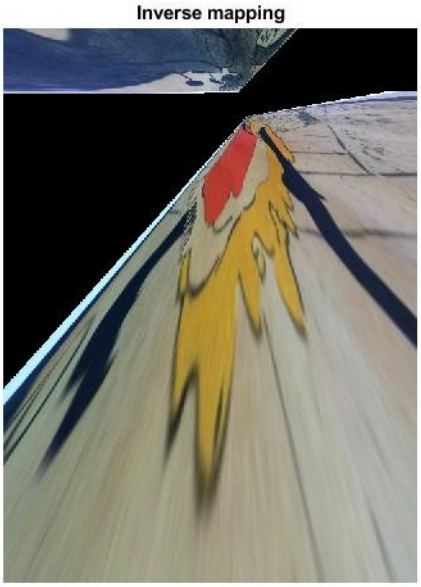
Instead with RANSAC (above) the result are very good, it has discharged the outliers.

Fabio Fusaro
Davide Torielli
Luca Morando

Normalized method



Another thing we can modify is to use normalized points to estimate the H matrix. We try with 6 points (selecting 3 of them not so correct).



As we can see above, the vanilla version is very bad.

Fabio Fusaro
Davide Torielli
Luca Morando

Inverse mapping with normalized points

Average image with normalized points

Instead, the normalized version manages to work better.

The RANSAC method, in this case, drops two outliers but one must be taken because minimum number of point is 4.

Inverse mapping with RANSAC method

Average image with RANSAC method

Result is different from normalized but in terms of overall errors it is just as good.

Fabio Fusaro
Davide Torielli
Luca Morando

Building a mosaic from a sequence of images

With this code we compute a “panoramic” image built from a set of images.
This is similar to the second problem but now it is more difficult because we need 8 correspondence points. In fact the images are not 2D, and so the homography matrix has 8 degrees of freedom (as a fundamental matrix).
The correspondence is done taking one reference image (the one in the middle) and selecting the correspondence points between this and all the other images.
If we take with care correct points, the result is pretty good:

Fabio Fusaro
Davide Torielli
Luca Morando