Computer Vision

Assignment 4 : Model fitting

Autumn 2018

Nikita Rudin

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Computer Vision
and Geometry Lab

1 Ransac linear regression

In this exercise we apply RANSAC to a linear regression problem and compare the results with least squares. The RANSAC algorithm selects two random points and gets the line passing through them. It then checks the number of points within the treshold distance to the line (inliers). It then repeats these steps a given number of times until

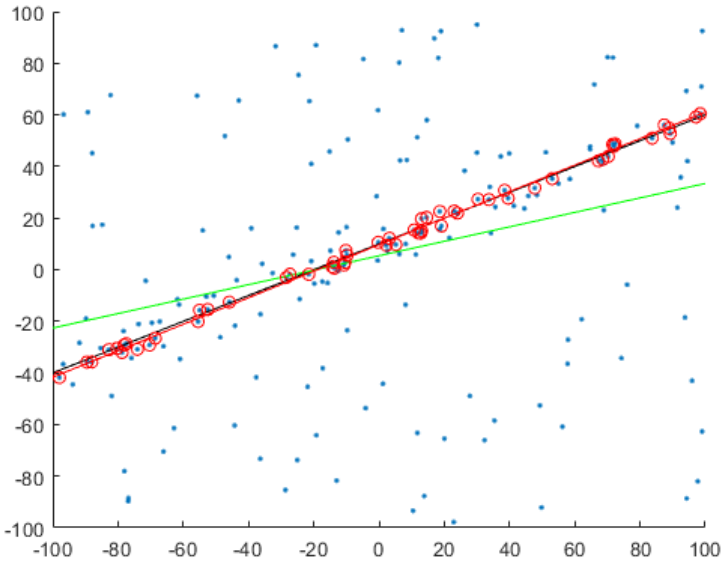


Figure 1: RANSAC Algorithm applied to a linear regression

As expected we see that the least squares solution is heavily impacted by noise while the RANSAC line is very close to the true model.

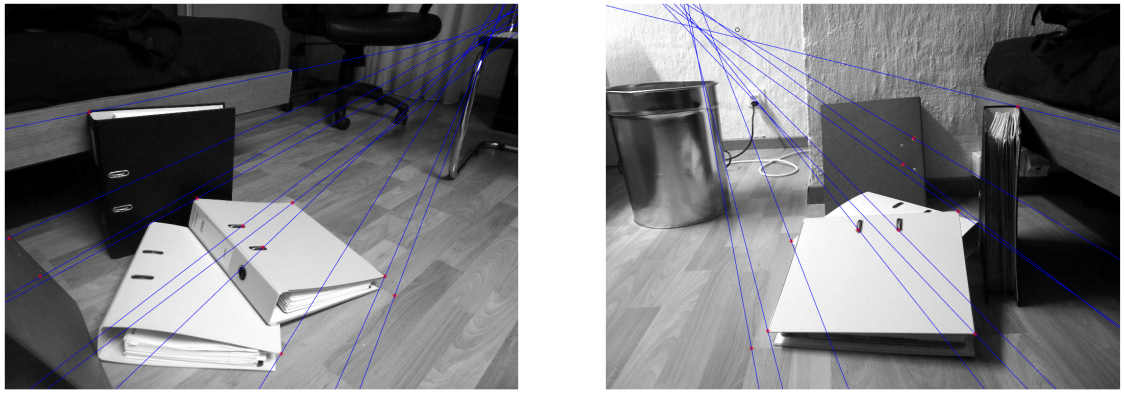
Real	RANSAC	least squares
44.5695	164.4821	44.5695

Here the errors are only computed on inliers. The least square solution is trying to minimize the total error, which explains why it is much larger when considering the inliers only. The RANSAC error appears to be even smaller than the real error, but this is due to random chance.

2 Fundamental Matrix

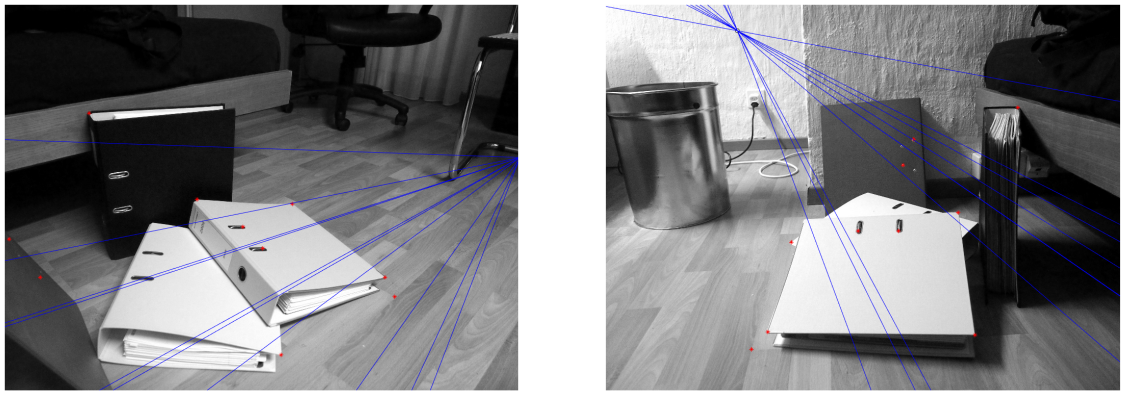
In the second part of this assignment we extract the fundamental matrix from a series of hand-clicked matches on two images. The images represent the same scene, but with a change of point of view. We apply the 8 point algorithm. We construct the A matrix from the point matches and then extract F as the right null vector using SVD decomposition.

We then impose the singularity constraint by setting the last singular value to 0. 10 points are used for robustness to noise in the matches. The points are normalized for numerical issues so the final F matrix has to be denormalized.



(a) Image 1 (b) Image 2

Figure 2: Projection of the epipolar lines using the non singular matrix F



(a) Image 1 (b) Image 2

Figure 3: Projection of the epipolar lines using the singular matrix F_h

As we can see the results are different between F and F_h. If we do not impose the singularity constraint the lines pass exactly through the matches but they don't converge to a single epipole. Moreover the projected epipole (black circle) is also misplaced. When we use the singularity version of the matrix, all lines converge to the same point and the epipole is projected exactly at that point, but now the lines don't pass through the matches any more.
It is interesting that we get such a big difference in the projected line, because the difference in the matrix elements is only in the order of 10^{-6}.

3 Essential matrix

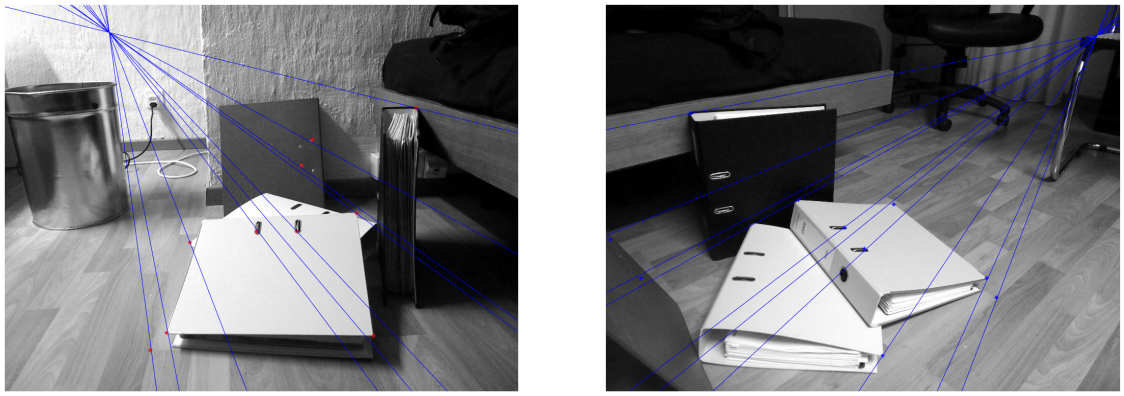
We will now repeat the previous steps but this time computing the essential matrix. It is computed in the same way as the fundamental matrix, but the points have to be premultiplied with the known camera matrix K.

$$x_n = K^{-1}x$$

The singularity constraint is also slightly different. In addition to setting the last singular value to 0, we set the two others to the same value (their mean). We can also diretly compute the E matrix from F.

$$E = K^T F K$$

Comparing both results we get differences of the order of 10^{-3} without the singularity constraint and up to 1.5 with the constraint, so there are noticeable differences between the two approaches.



(a) Image 1 (b) Image 2

Figure 4: Projection of the epipolar lines using the singular matrix E_h

We see that the results look better than those of the F matrix. The lines pass close to the points and also converge to a single point, the epipole. It is normal that this approach yields better results because we are giving more information to the algorithm by providing K.

Having computed E, we can get the relative rotation and position of the 2nd camera with respect to the 1st one. When doing the computation we get multiple solution and we need to select the one that puts all points in front of the camera. finally we can project points in 3D using triangulation and show the estimated cameras in that space. We see that the results look close to what we can assume by looking at the images.

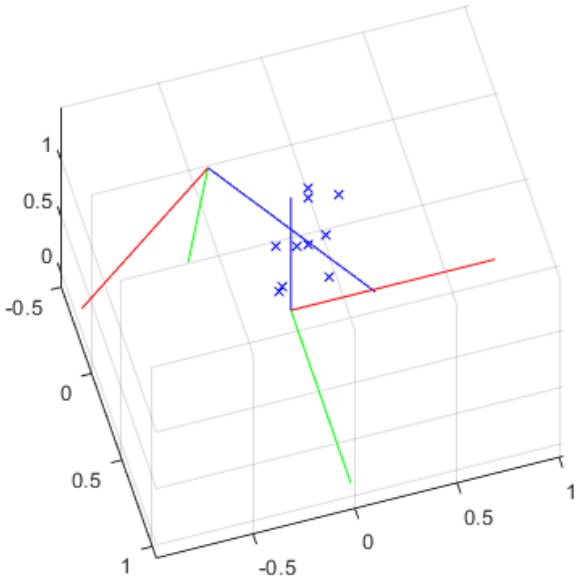


Figure 5: 3D projection of matches with estimated cameras

4 RANSAC for the 8 point Algorithm

The last step of this assignment is to implement the 8 point algorithm on matches detected automatically by a SIFT detector. We use RANSAC to determine the correct matches. We can see that the matches provided by the detector are not perfect. These few errors can heavily impact the final result.

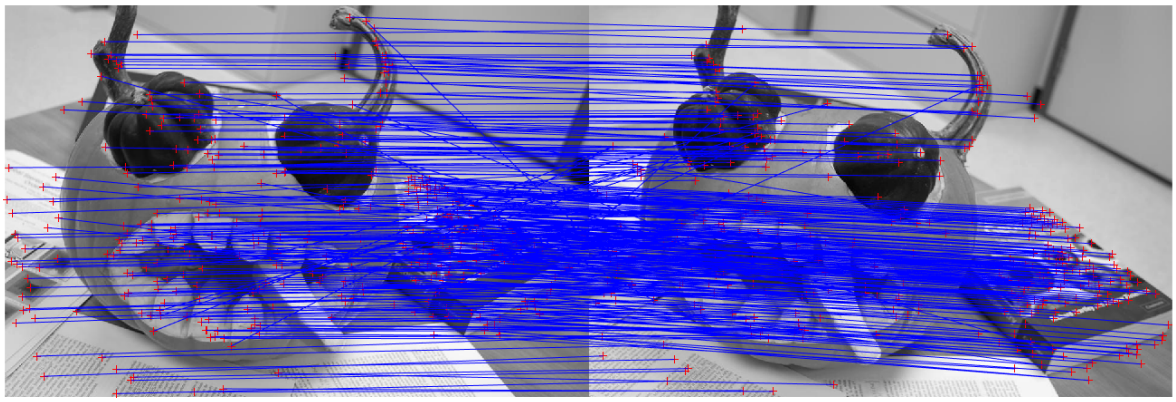


Figure 6: All matches detected by SIFT

We implement RANSAC in the following way. We select 8 random points from the

matches and calculate the fundamental matrix. We then compute the distance between the projected lines and the matches in both images. If this distance is below a threshold we consider the point an inlier. In the first case we apply this algorithm for a fixed number of times(1000) and use the largest group of inliers to compute the final F matrix. We can make the end condition adaptive. We calculate the probability of having found a solution without outliers and if this probability is above 99% we stop the search. The probability is given by

$$1 - (1 - r^N)^M$$

with $N = 8$ the number of points selected each time, M the number of iterations and r the ratio of inliers, which is estimated as the best ratio found so far.

Figure 7: Incorrect matches detected by RANSAC after 367 iterations

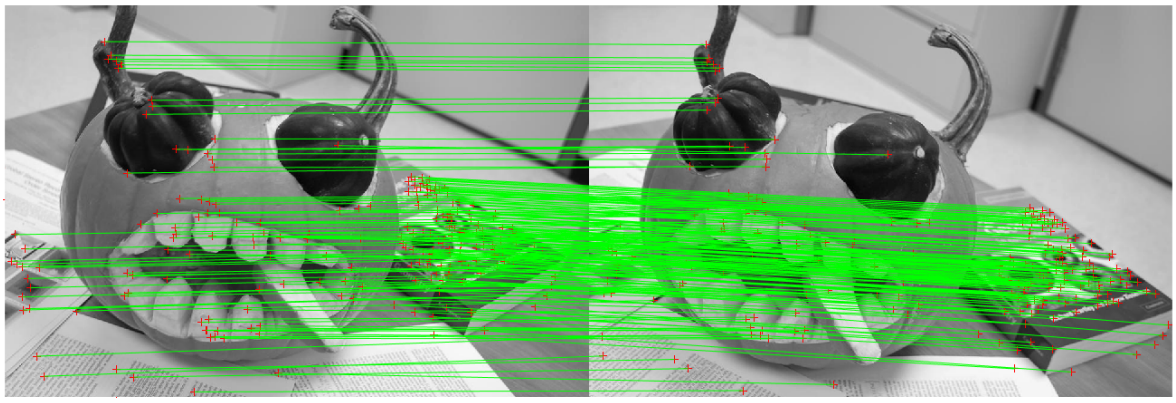


Figure 8: Correct matches detected by RANSAC after 367 iterations

It appears that after a little bit of tuning, the inlier matches provided by RANSAC are all correct for the pumpkin image. Also we get these results after only 367 iterations so

the adaptive conditions allows us to avoid some useless computations We can re project the points in 3D and we can even imagine the pumpkin and the book in the 3D view.

It is important to note that the results are not perfect on other images for example we

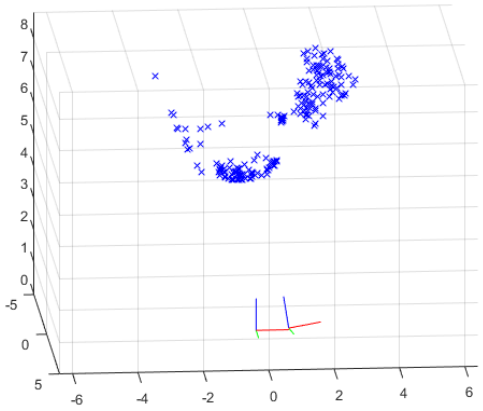


Figure 9: 3D projection of the matches

can see below that there are incorrect matches classified as inliers. This is due to the far lower inlier ration in the detected matches, which can be explained by the complexity of the scene and the large change in point of view. The adaptive algorithm takes much longer to finish because r is much lower.

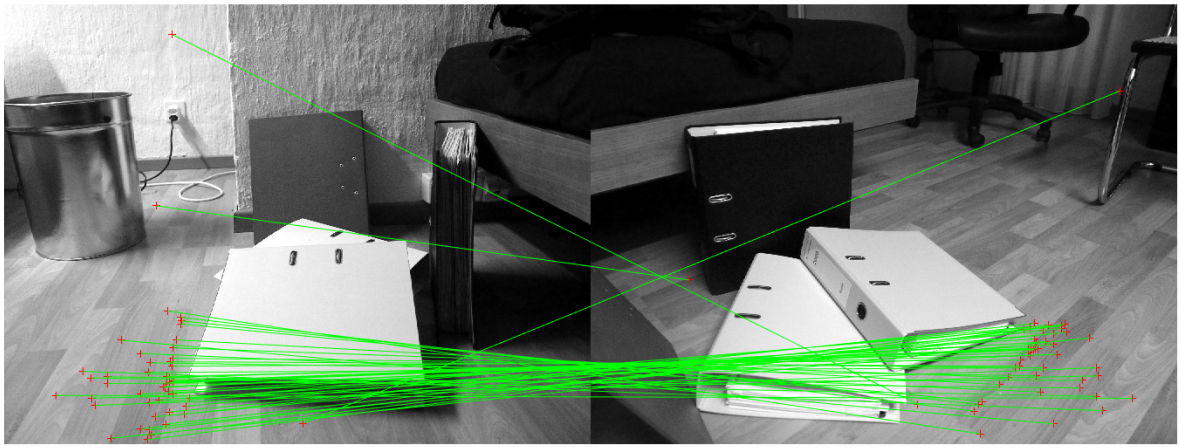


Figure 10: Inliers detected by RANSAC after 367 iterations