Computer Vision

Assignment 7 : Structure from Motion

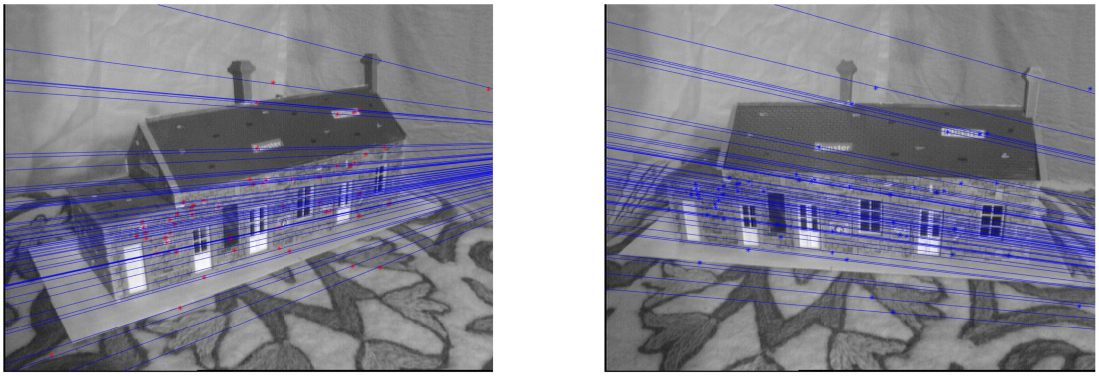
Autumn 2018

Nikita Rudin

November 21, 2018

1 Fundamental matrix extraction

In this lab we are given pictures or an object (a small house) on a turn table. We have 5 images representing the house with different degrees of rotation. The first task is to extract the epipolar geometry from two images and triangulate a first set of 3D points. The first step is to apply the SIFT detector to our two images. We then use the matches in the 8-point algorithm to compute the fundamental matrix (F). We use RANSAC to get rid of unwanted outliers. Using K we can easily get the essential matrix from F. We can decompose E to get the position and the orientation of the second camera relative to the first. Finally we can use the projection matrix we got from the decomposition of E to triangulate our matches in 3D.



(a) image 1 (b) image 3

Figure 1: Extracted Epipolar geometry

We can also plot the matches obtained after RANSAC and see that they are mostly very good. Sometimes we still see one or two errors as here there is one mismatch in the carpet (lowest line).

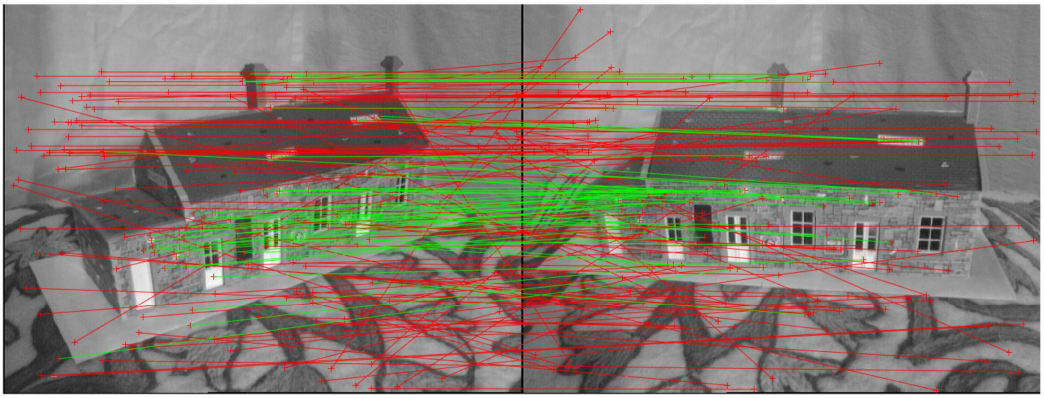


Figure 2: Matches between images 1 and 2. Inliers in green, outliers in red

2 Adding more views

Now that we have triangulate 3D points we can use them to find the position of cameras in other views. First we matches SIFT features between the first and the new image. Note that we will only use the matches from the 1st image that were inliers and as such are triangulated in 3D. Once we have this, we have a correspondance between 2d and 3D points for the new image. We can then use 6-point DLT with RANSAC to compute the projection matrix (P). In addition to the provided code, we need to add a verification to ensure that the determinant of R is positive. We then use it to get the position and orientation of the added camera. Finally we can re-project the matches to 3D and compare the results with the previous points. We repeat these steps for 3 new images. To simplify the implementation I have created the function AddViews that does the steps described above and returns the P matrix and triangulated points.

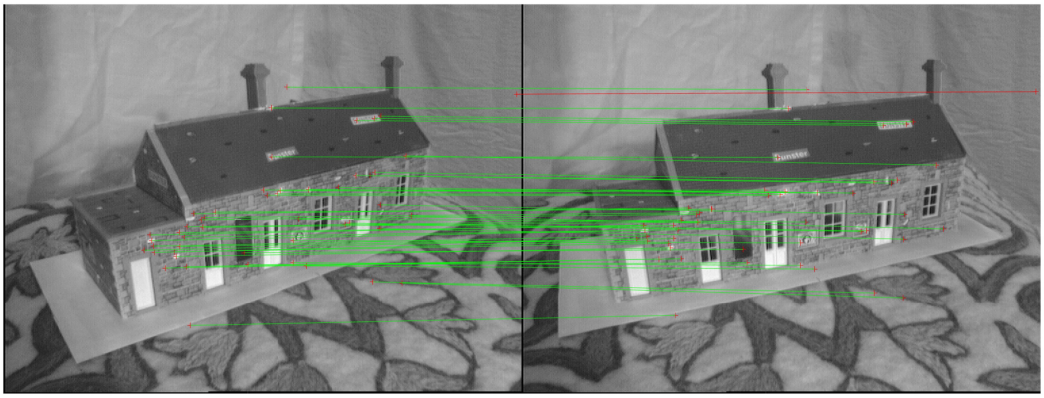


Figure 3: Matches between images 1 and 3. Inliers in green, outliers in red

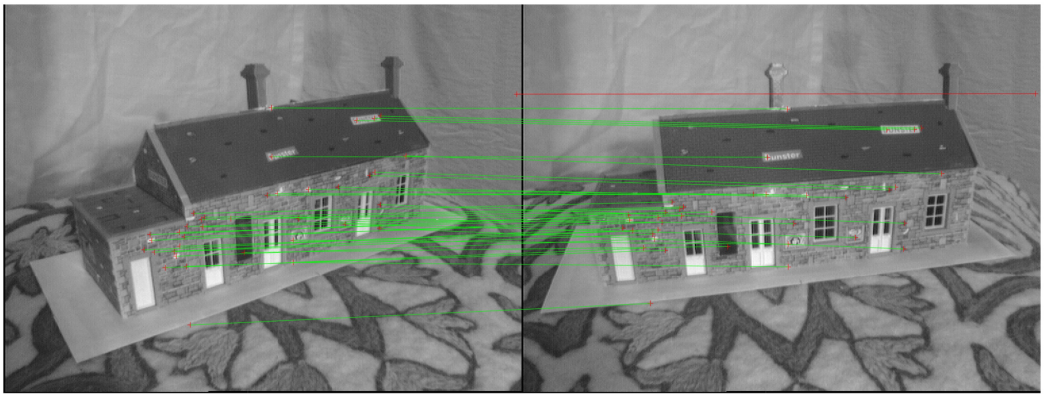


Figure 4: Matches between images 1 and 4. Inliers in green, outliers in red

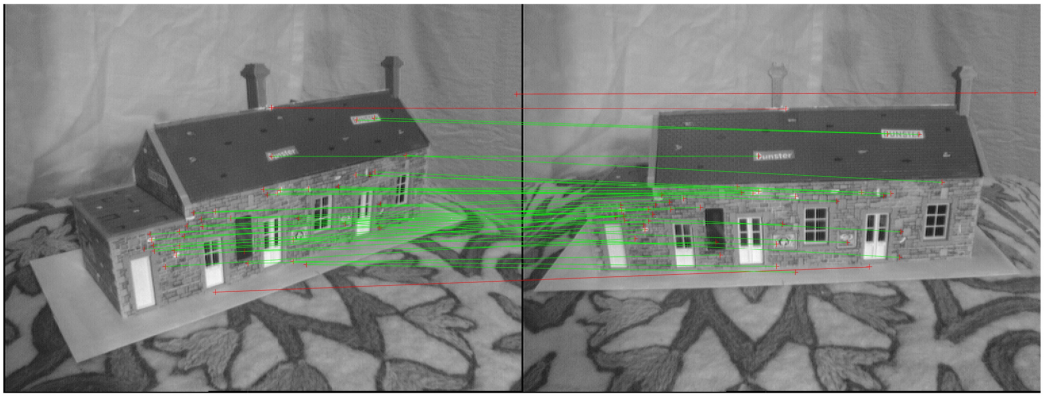


Figure 5: Matches between images 1 and 5. Inliers in green, outliers in red

3 Sparse 3D

Through the previous tasks, we have computed the triangulation of matched points in 3D. We have also estimated the positions of the camers in this 3D world. Note that the cameras don't actually move, but moving the cameras or the scene is equivalent and as such we can represent our turning object by cameras that turn around it.

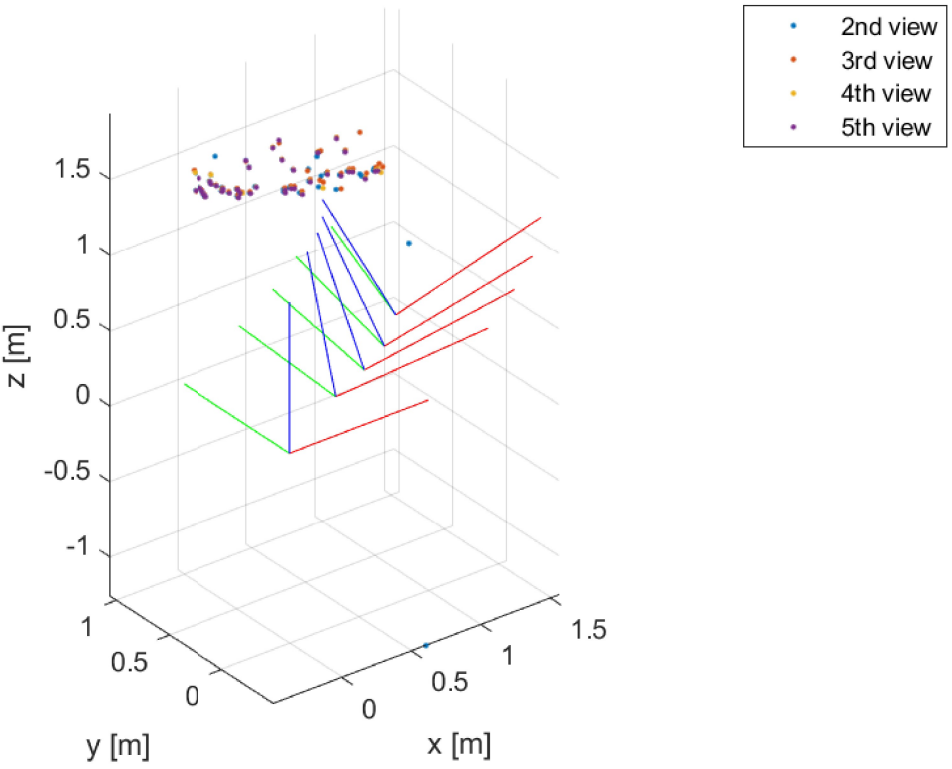


Figure 6: Matches between images 1 and 3

We can see that the cameras are correctly estimated. The points projected from different views should overlap, but since our results are not perfect we see some mismatch.

3.1 Dense 3D

I have tried to use the same approach as in the previous lab to make a dense reconstruction. First we rectify the images so that only a displacement along x is left and then we find the best disparities. This approach didn't provide any usable results because the rectification was very unstable. Retrospectively, it is probably due to the fact that this is not the right approach for scenes the orientation changes.