Computer Vision Assignment 7

Luca Ebner (ebnerl@student.ethz.ch)

November 27, 2020

The goal of this exercise is to create a simple structure from motion pipeline.

1. Feature Extraction and Initialization with Epipolar Geometry

To initialize the pipeline, we need to consider two images. If we extract feature matches from the image pair, we can use an 8-point Random Sample Consensus (RANSAC) algorithm to compute the epipolar geometry. From there, we can compute the essential matrix and projection matrices.

To extract feature matches, we can use the `vl_sift` and `vl_ubcmatch` functions from the `VLFeat` toolbox (`https://www.vlfeat.org/`). With the 8-Point RANSAC algorithm, we can then compute the fundamental matrix **F** similarly as we did in *"Lab Assignment 4 - Model Fitting"*. In this exercise, the function `ransacfitfundmatrix` was provided to do so. In Figure 1 and 2 the feature matches counted as inliers in the RANSAC algorithm and the resulting epipolar geometry can be seen.

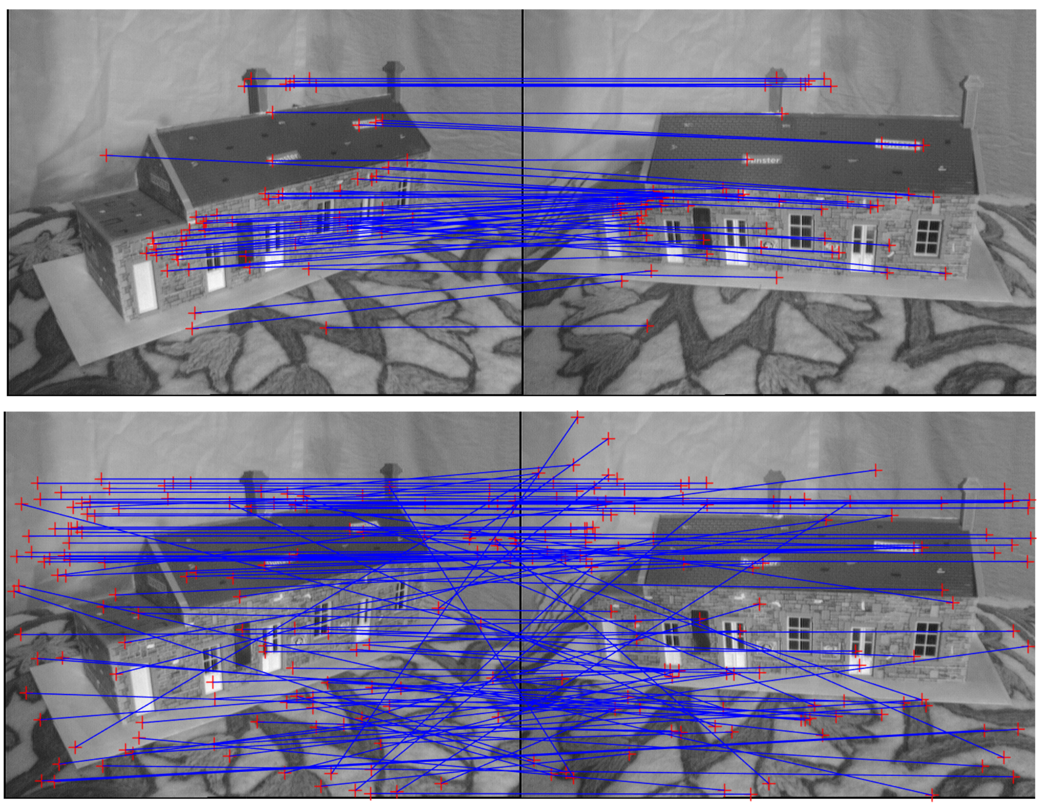

Figure 1: RANSAC inliers (top) and outliers (bottom) of feature point matches between the images 0 and 4 used for the initialization. Note that both the inlier and outlier matches can still contain errors.

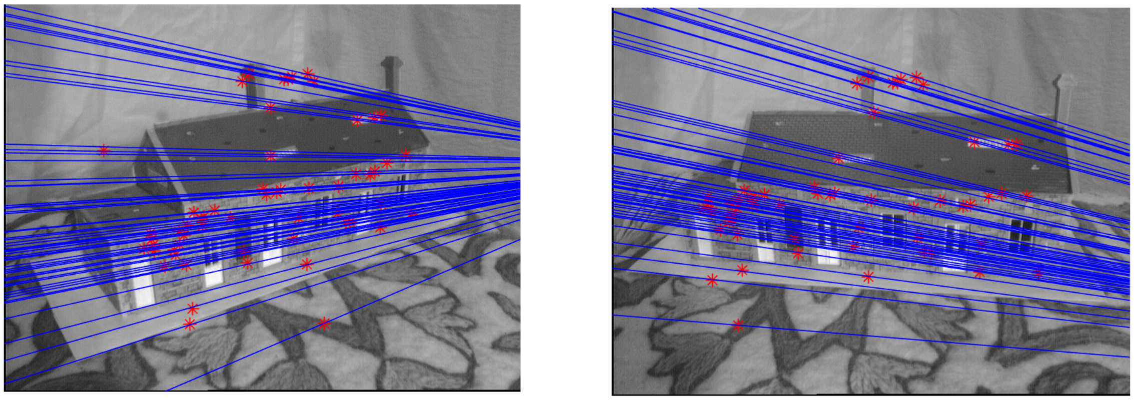

Figure 2: Epipolar lines resulting from the fundamental matrix found in the 8-point RANSAC algorithm from the initialization image pair.

If we have found **F**, the equation for the essential matrix **E** is given as

$$E = (K')^T * F * K, \tag{1}$$

where K', K are the the intrinsics camera calibration matrices of the two images. Note that in our case it holds that $K' = K$, since we use the same camera for both images.

If we assume that the projection matrix **P** for the first image is $[I|0]$, we can use the provided function `decomposeE` which decomposes **E** into a translation vector **t** and a rotation matrix **R** in order to find the projection matrix **P** for the second image.
Once we have retrieved the projection matrices of both images, we can use them to triangulate the matched feature points, store their 3D position and get 3D-2D point correspondences. For this, we can use the provided function `linearTriangulation`. A graph including the triangulated 3D points from all image pairs in the exercise can be seen in Figure 6.

2. Triangulation and Adding new Views

We can extend our results from the previous exercise by adding more views from our scene. As explained in the exercise sheet, we can achieve this by matching features from additional views with one of the existing views from the previous exercise. These matches can then also be seen as 3D-2D matches between the 2D point in the new image and the 3D point found in the triangulation from the previous exercise.

As in the first exercise, we start by extracting feature matches between one of the images from the initialization and an additional image. We can then perform a 6-point DLT algorithm similar to the one we used in *"Lab Assignment 1 - Camera Models and Calibration"* and combine it with RANSAC to fit a projection matrix between the newly added image 2D feature points and the corresponding 3D triangulation points. For this we can use the provided function `ransacfitprojmatrix`.
Figures 3-5 show the respective in- and outliers between the image pairs of the added views chosen for this exercise.

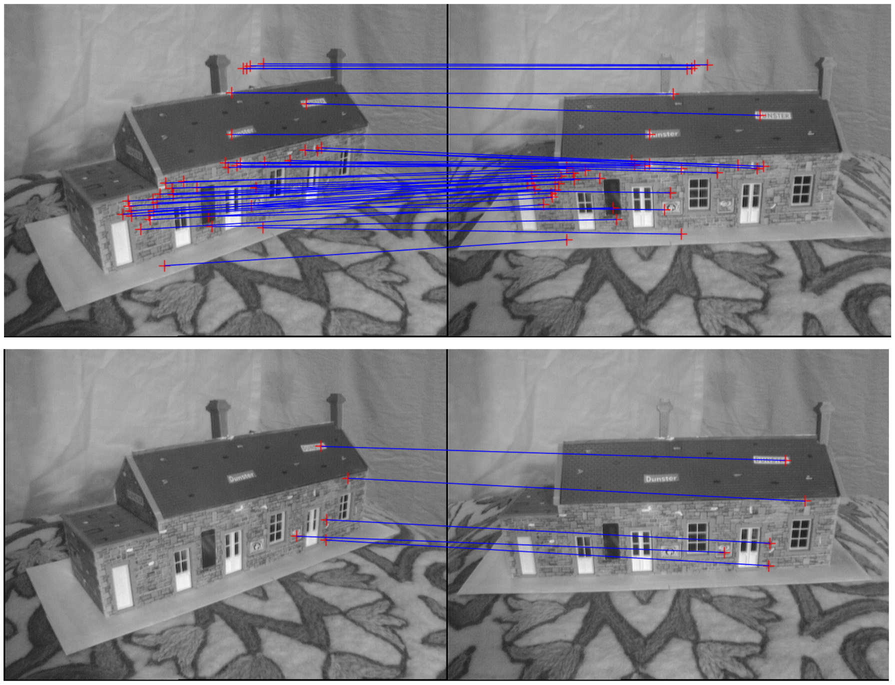

Figure 3: RANSAC inliers (top) and outliers (bottom) of feature point matches between the images 0 and 3. Note that both the inlier and outlier matches can still contain errors.

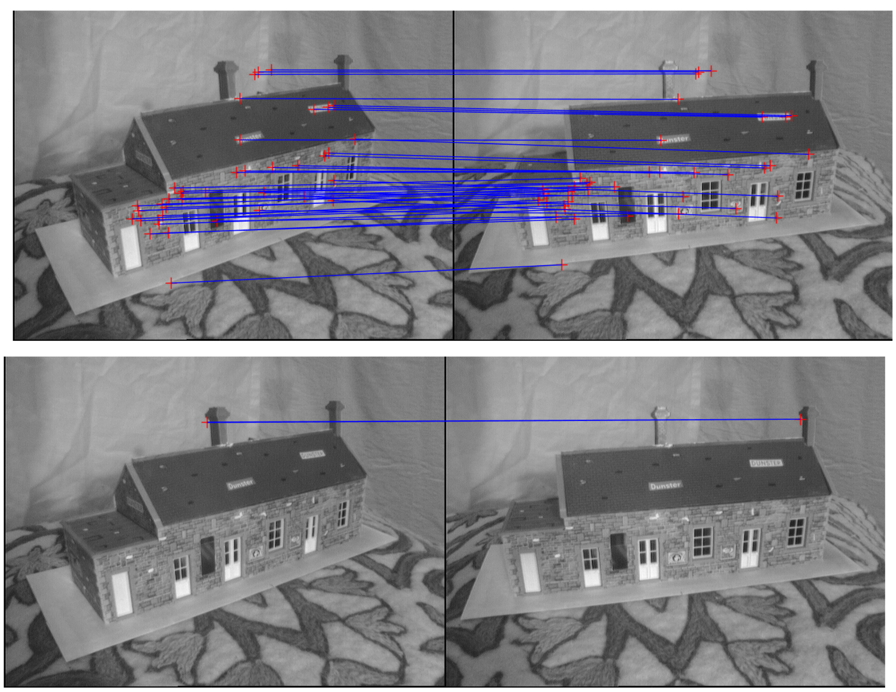

Figure 4: RANSAC inliers (top) and outliers (bottom) of feature point matches between the images 0 and 2. Note that both the inlier and outlier matches can still contain errors.

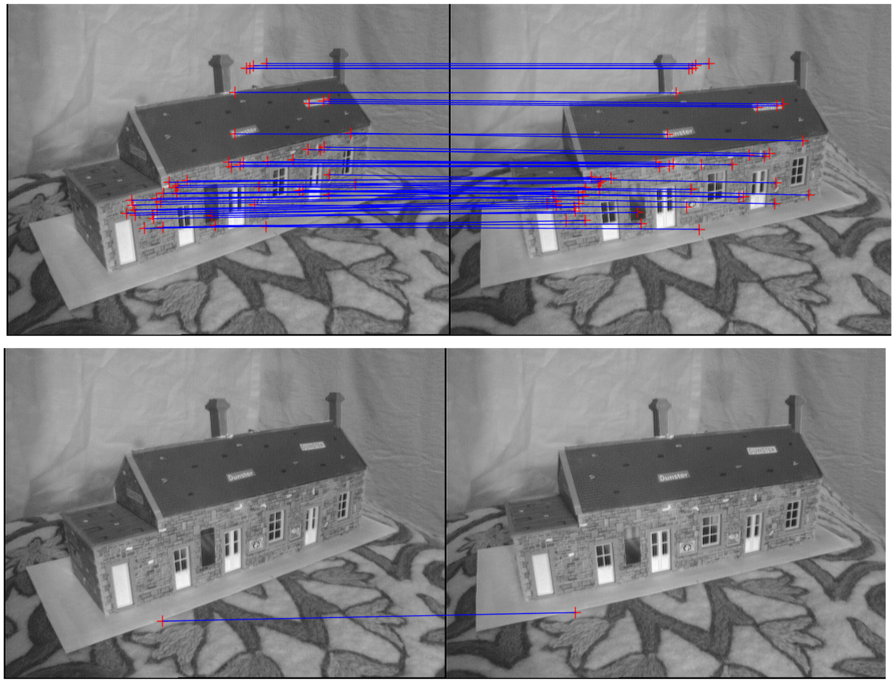

Figure 5: RANSAC inliers (top) and outliers (bottom) of feature point matches between the images 0 and 1. Note that both the inlier and outlier matches can still contain errors.

3. Plotting

In the end, we can investigate our results by plotting the triangulation of all our RANSAC inlier matches between the views. Figure 6 shows a visualization of our 3D triangulation points, where we used the image pair 0-4 for initialization and 0-1, 0-2, 0-3 for additional views.

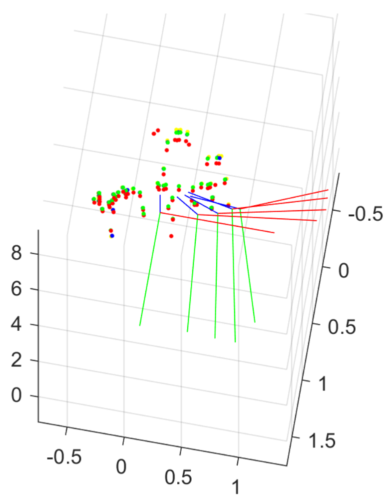

Figure 6: Visualization of the 3D triangulation points. As expected, the first camera frame lies in the origin and the others are are rotated around the triangulated points.