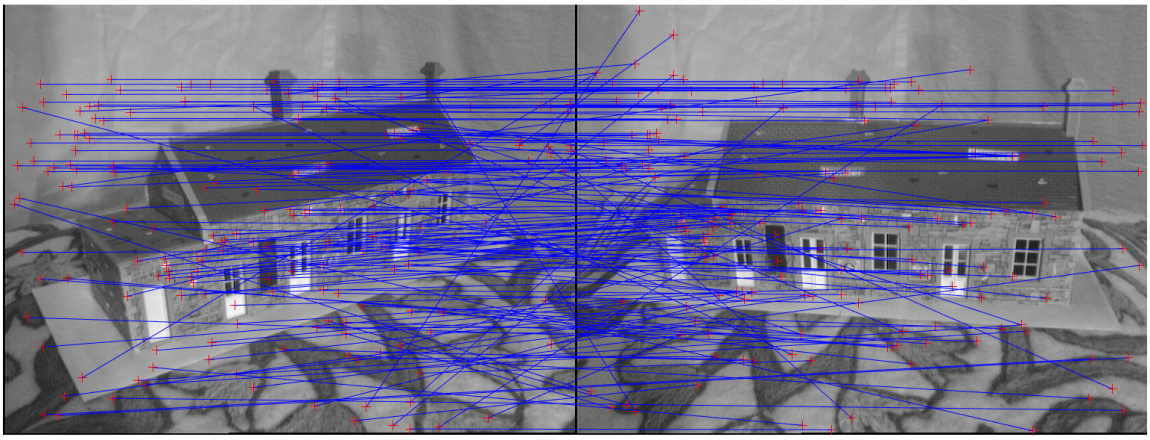
Computer Vision Exercise 7: Structure from Motion

Soomin Lee (leesoo@student.ethz.ch)

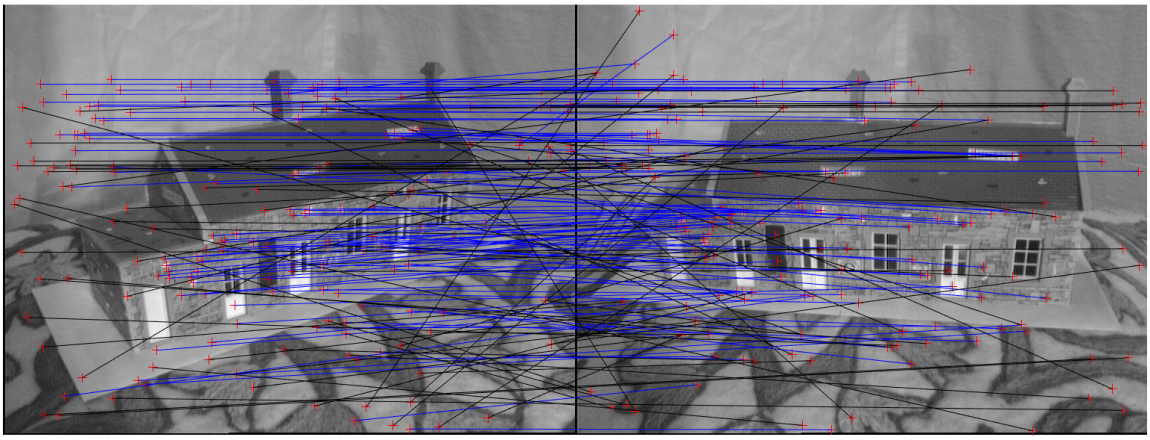
November 27, 2020

1. Feature extraction and initialization with epipolar geometry

To initialize the pipeline, the first and last images are taken in order to have a relatively large baseline for a better quality of triangulation. Then using `VLFeat`, SIFT features are extracted and the matching between the descriptors are found. With the matched features we estimate the fundamental matrix F using RANSAC (8-point RANSAC algorithm). The matching between the features is shown in Figure 1, (a) showing the original matching before performing any verification and (b) showing only the matches that are classified as inliers when estimating the fundamental matrix with the RANSAC algorithm. Also, the epipolar lines are shown in Figure 2.

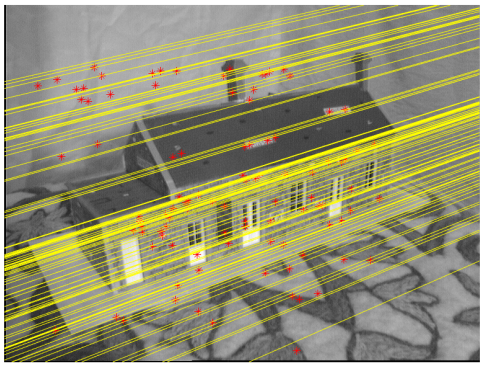


(a) Original matches.

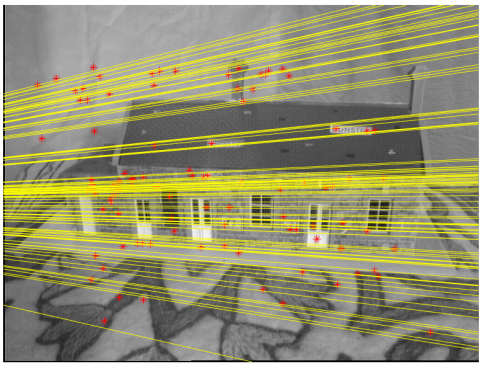


(b) Inlier matches (**blue**) and outlier matches (**black**) after 8-point RANSAC algorithm.

Figure 1: Extracted features (red cross) and matching between the them (line). The images are the first (left, `img000`) and the last (right, `img004`) image of the sequence respectively.



(a) The first image of the sequence (`img000`).



(b) The last image of the sequence (`img004`).

Figure 2: Epipolar lines (yellow line) calculated with the fundamental matrix.

Since the calibration matrix K is known and is identical for all the images, we can then calculate the essential matrix using the equation $E = K^T F K$. Furthermore, we can decompose the essential matrix into R and t through SVD so we can get the projection matrix. As we can only know the relative rotation and translation, we set the projection matrix of the first image as an identity matrix. Lastly, we can triangulate the inlier matches with the computed projection matrices. Therefore, we now have 3D-2D point correspondences to start with, meaning that we can triangulate additional images one by one on top of the triangulation from the current initialization step.

2. Triangulation and adding new views

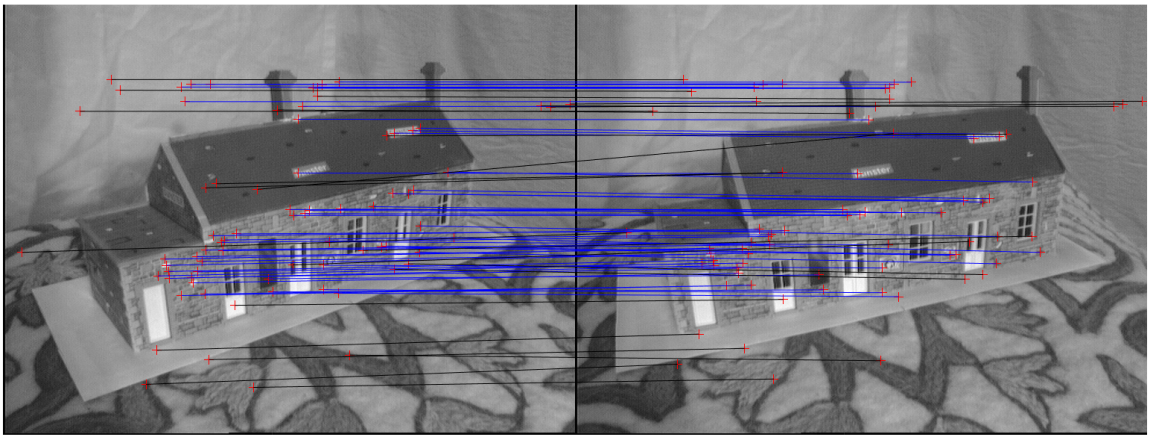
From now on, we take a new image and extract SIFT features in the same way as before, then we match the features against the features from the first image (`img000`) that were triangulated. Since we obtained the 3D-2D correspondences for the first image in the previous step, by matching the features from the new image to those from the first image, we have acquired the 3D-3D correspondences for the new image as well. Therefore, we can estimate the projection matrix directly with RANSAC algorithm (6-point RANSAC algorithm) instead of calculating the fundamental matrix. The inlier matches after adding each image (`img001`, `img002`, `img003`) are shown in Figure 3.

3. Plotting

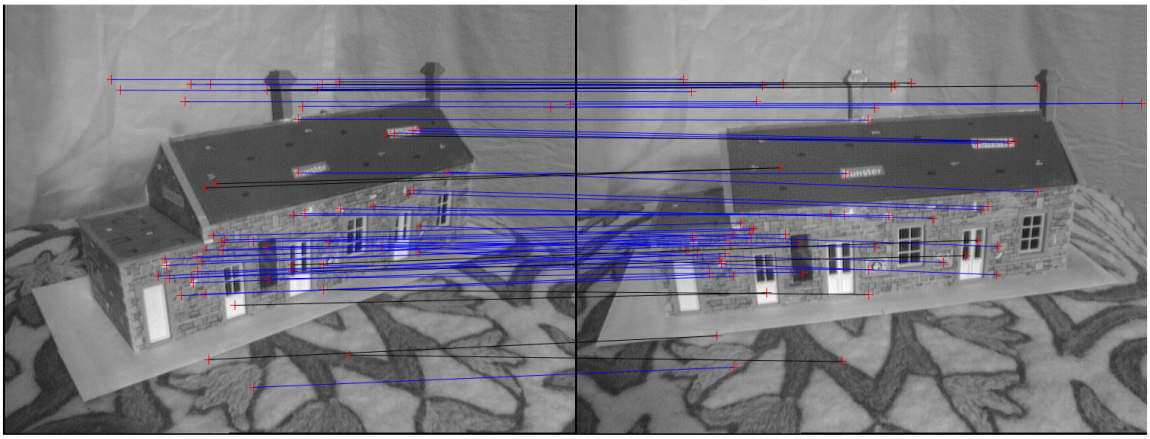
The result of each triangulation is plotted together in Figure 4, as well as the camera poses for each view. As one can expect from the image sequence, the camera pose moves and rotates consistently. Yet, the points are not aligned perfectly. It can be later improved by running an optimization algorithm such as bundle adjustment.

4. Dense Reconstruction

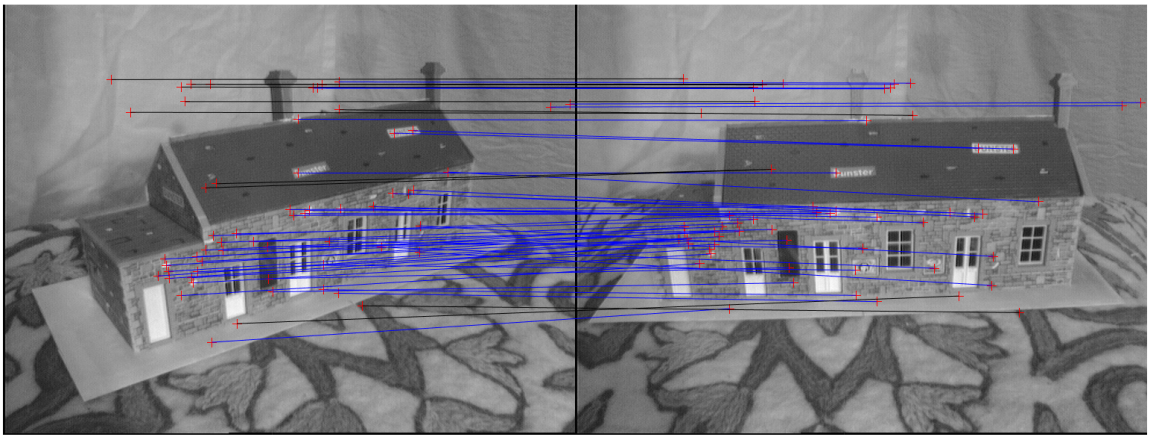
To get a dense reconstruction instead of the sparse reconstruction as in Figure 4, we can compute the disparity map using the graph-cut method as in the previous stereo matching exercise and project every pixel that has depth into the scene. The first (`img000`) and the second image (`img001`) are selected for the stereo matching. Figure 5 shows the rectified images, and Figure 6 shows the disparity map. Finally, the scene with the depth information together with the image is shown in Figure 7 and 8. Figure 7 was generated by using the same method as the previous exercise and visualized with `meshlab`, while Figure 8 was created using the given function `create3DModel`, in particular using the method with `trisurf`. The depth map was calculated by the equation $Z = bf/d$, where Z is the depth, b is the length of the baseline,



(a) `img000` (left) and `img001` (right).



(b) `img000` (left) and `img002` (right).



(c) `img000` (left) and `img003` (right).

Figure 3: Inlier matches (**blue**) and outlier matches (**black**) after 6-point RANSAC algorithm.

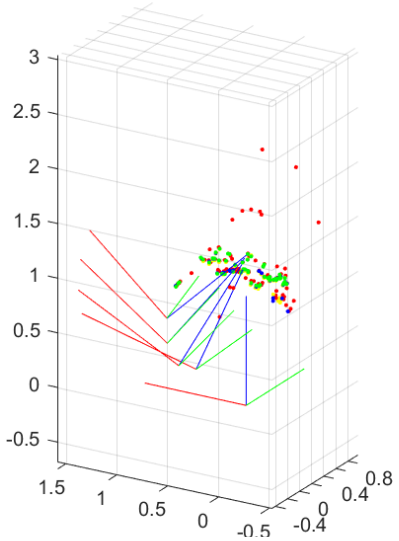


Figure 4: Triangulated points from each step (order: red, green, blue, yellow) and the camera poses.

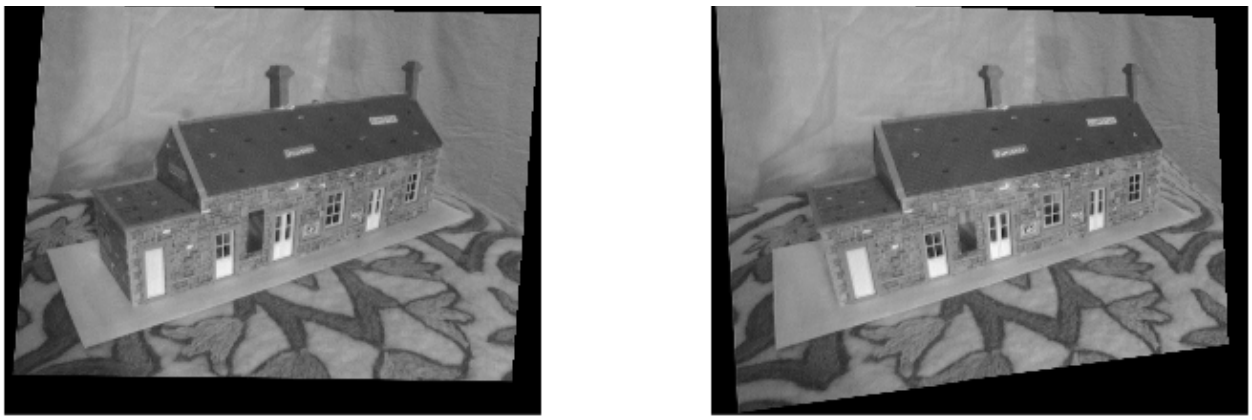


Figure 5: Rectified images (left: `img000`, right: `img001`).

f is the focal length, and d is the disparity. Also, only the disparity values that agree between two images are used. Further optimizing can be still done by setting a better disparity range for instance.

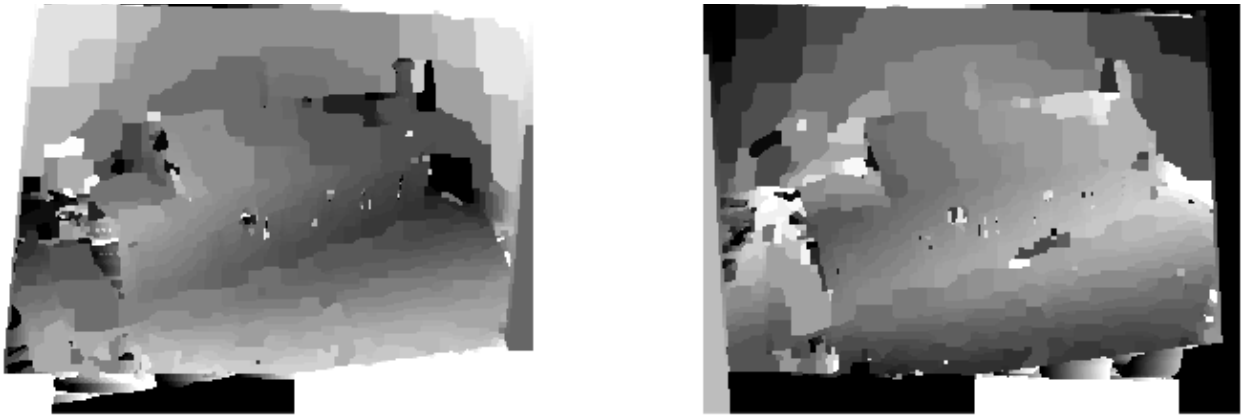


Figure 6: Disparity map (left: `img000`, right: `img001`).

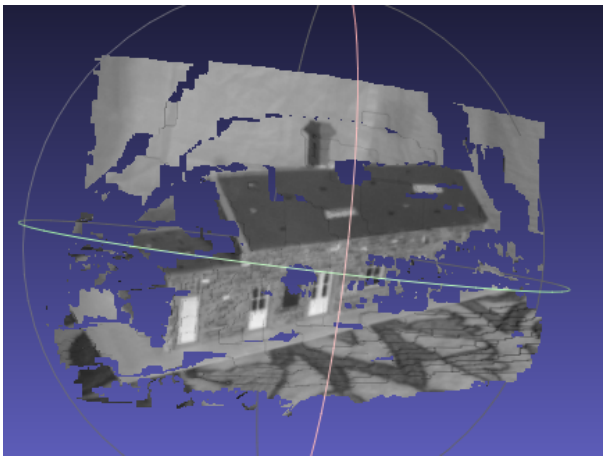
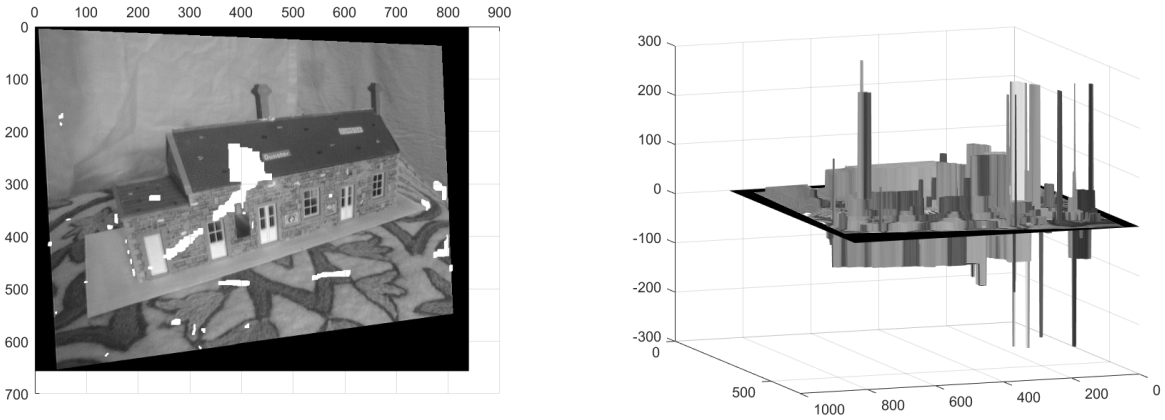


Figure 7: Reconstructed scene using `MeshLab`.



(a) View from front. (b) View from side.

Figure 8: Reconstructed scene using the provided function `create3DModel`.