CS4393 Computer Vision

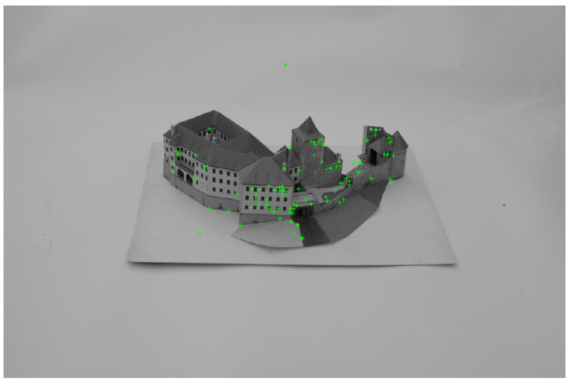
Final project
3D Reconstruction

Xin Li 4721101
Jiahui Li 4734769

June 2018

1 Keypoint detection

To detect keypoints in the frame, we first convert the rgb image to gray scale and use **vl_sift.m** to get SIFT features, we set the parameter 'PeakThresh' to 6, Figure 1 shows our result of detected keypoints, Latter we set peak thresh to 1.2 to obtain more points as in Figure 2


(a) Feature points found in frame 1

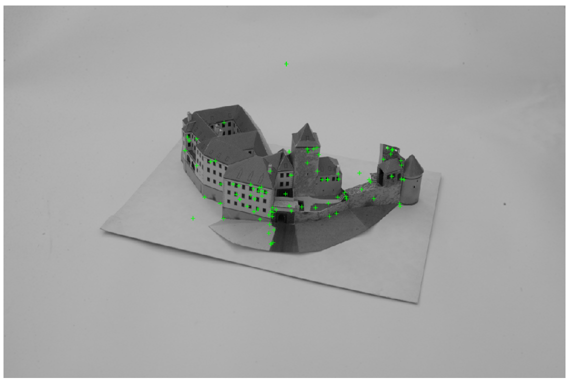
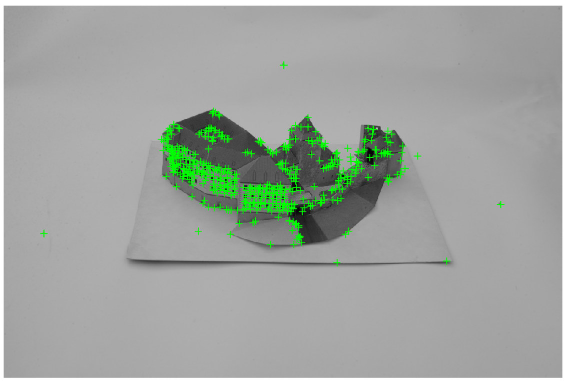

(b) Feature points found in frame 2

Figure 1: Result of SIFT, peak thresh 6


(a) Feature points found in frame 1

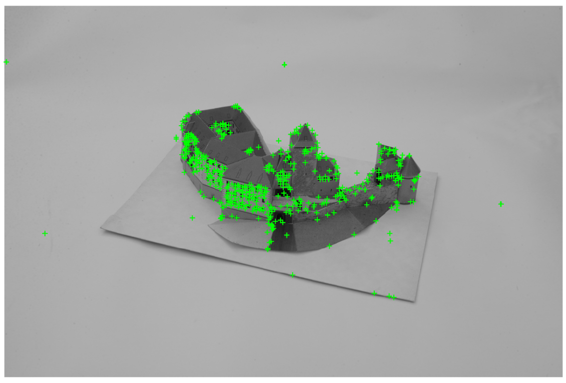
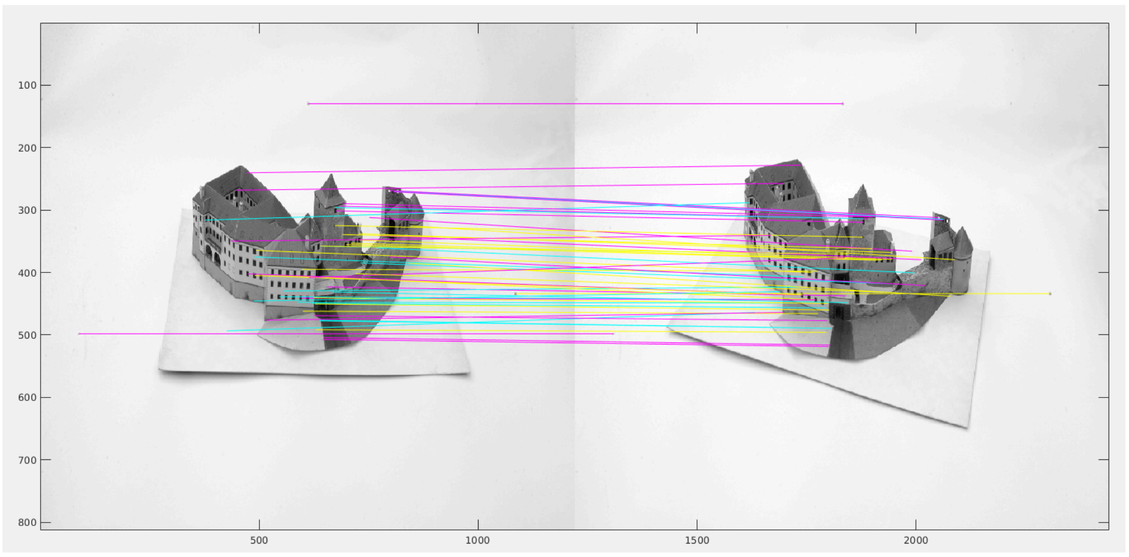

(b) Feature points found in frame 2

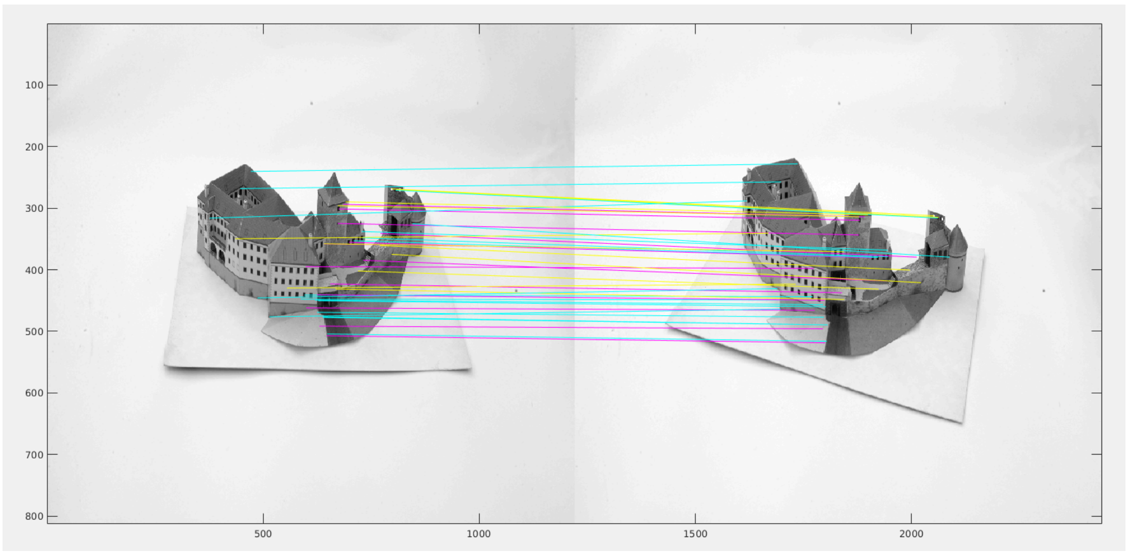
Figure 2: Result of SIFT,peak thresh 1.2

2 Matching

Then we use the function **vl_ubcmatch.m** to match those keypoints. But as showed in Figure 3a, there are many outliers presented in the background, this is because if we set a higher threshold for detection, the number of detected keypoints will decrease. Hence we use normalized eight point RANSAC attempts to exclude the outliers, code can be found in **eightpointransac.m**, number of iteration: 1000, error tolerance: 150, result shows in Figure 3b.



(a) Raw matches



(b) Matches after applying RANSAC

3 Chaining

For this process, we used the RANSAC implemented in step 2 to get the matches in every 1-2, 2-3 ..., frames, and store them in a **point-view-matrix**, The point-view matrix has views in the rows, and points in the columns. It is constructed in the same way as in (Lab assignment 6, Sec. 2) For the first 2 rows we simply place the matched point detected in frame1 and frame2 in it, then for latter rows, we match frame i and frame i+1, and find the the intersection of matches in frame i-1 and i (a), if there are matches, we place the intersected matches in frame i+1 into the same column as in (a), and add new matches in new columns, and match the first frame and last frame in the same way.

4 Stiching

For this step, we use a function called **find_points.m** to take the block that is consist of three adjacent images from the point-view-matrix, and use their corresponding locations as output.

For the locations we found in the previous step, they can be used to estimate 3D coordinates for each block, which is implemented in **sfm.m**. Applying Tomasi-Kanade factorization method, as described in (Lab assignment 4, Sec. 2), the locations firstly are transformed to shift the mean of the points to zero, so that the coordinate system would not have any aliasing. Next, the singular value decomposition will be applied to the centered points. Thus, we can find the motion matrix M and shape matrix S.

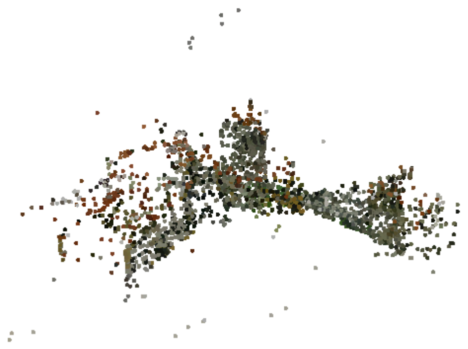
In order to stitch each 3D points from different view to a same view, the first view is chosen as the main view. By using the function **find_common_points.m**, we can find the 3D points shared by two different adjacent views, and stored them in different cell. By calling **procrustes.m**, a linear transformation can be determined between the points one by one, and the points can be transformed.

5 Eliminating affine ambiguity

In order to eliminate the affine ambiguity, the function **Isqnonlin.m** is use to find the least squares solution for the parameter of $L = inv(A' * A)$. However, the generated new L_0 may not be positive definite. To solve this problem, a function called **nearestSPD.m** is used to find the nearest (in Frobenius norm) symmetric positive definite matrix of L_0 [2]. After that, function called **chol.m** produces an lower triangular matrix to recover C, which is used to update M and S.

6 3D Model Plotting

The result of point cloud without interpolation is shown in figure 5a and 5a. From Figure 4b, a clear edge of castle can be observed. Because the points we found is not sufficient, so the interpolation works poorly in this case.


(a) The point cloud from normal view

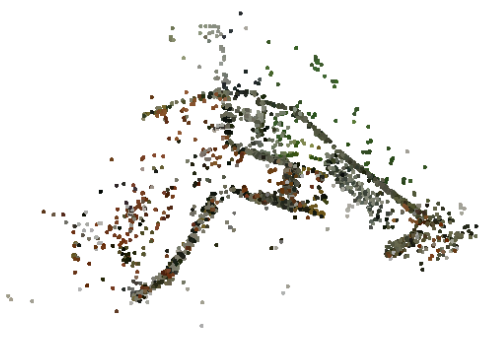

(b) The point cloud from top view

Figure 4: The 3D Model

7 Evaluation method

Without a complete ground truth model, it is difficult to evaluate our reconstructed model. Currently, the evaluation usually focuses on how much the reconstruction is consistent with the input datasets. There are several global accuracy estimation methods mentioned in paper [1].

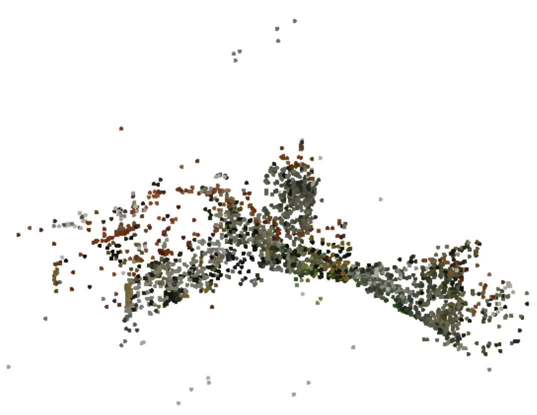
- Reconstruction error, computed as the average Euclidean distance from input points to the reconstructed surface.
- Integration error, calculates the average of the Euclidean distances between the points in the final reconstructed surface and their closest points in the input data source.
- Quantifies the accuracy of the reconstruction by measuring an average per-point distance of the range data to the reconstruction.

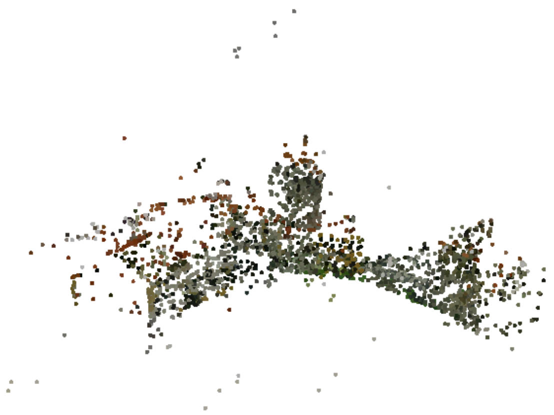
Global accuracy is used to give us a sense of whether a reconstruction tends to over- or underestimate the true shape. But it can not provide the information how consistent the local surface of the reconstruction is with its corresponding local surface on the true shape. In this case, local accuracy could be an important measurement.

8 Bundle adjustment

In order to obtain optimal 3D reconstruction, bundle adjustment is often used to minimize the re-projection error as the last step of feature-based structure and motion estimation algorithms, which is traditionally solved using a sparse variant of the Levenberg-Marquardt(LM) optimization algorithm.

For this task, we use matlab build-in nonlinear least-squares solver: **lsqnonlin** with option :LM to perform local bundle adjustment in **BA.m**. In this case we select subset of 3 views, and use three sets of camera matrix and the 3D points to reproject points on 3 image planes and calculate the reprojection error, then use the optimization function (use the origin M S as the starting point) to get rectified M and S. Also due to limited computation and the size of keypoinst matrix is quite large, we did not perform global bundle adjustment. Below is the comparison, and we can observe some tiny changes of these points.


(a) Point cloud without bundle adjustment


(b) point cloud with bundle adjustment

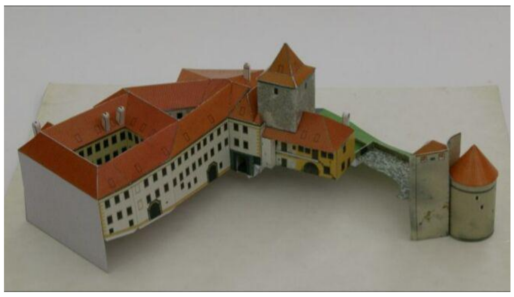


Figure 6: Raw image

References

[1] R.Hartley.*AnObjectOrientedApproachtoSceneReconstruction*,InProc.ofIEEEConf. SM&C'96, volume4, pages2475–2480, Oct, 1996.

[2] https://nl.mathworks.com/matlabcentral/fileexchange/42885-nearestspd