Maastricht University

Assignment Report

Image Stitching

Prepared by:
Foti Kerkeshi

May 2022

Table of contents

Problem definition

Given two images from the same scene, stitch these images side-by-side by finding common parts (features) that overlap with each other.

Approach

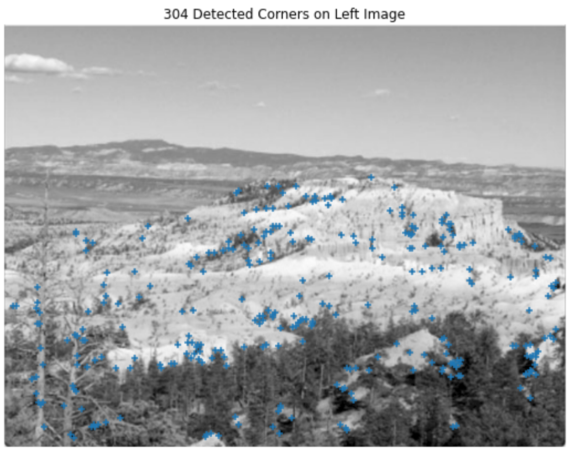
The steps followed to solve the problem are as follows:

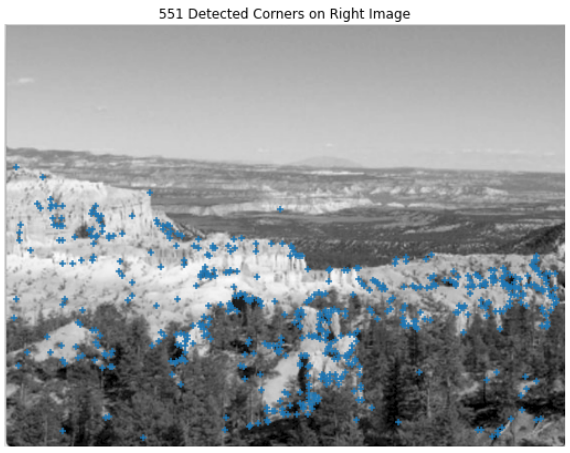
1. Detect feature points using Harris corner detection.
2. Create key-point descriptors by creating patches around them
3. Estimate key-point matches based on the Euclidean distance
4. Perform RANSAC to find the affine transformation matrix
5. Warp and apply geometric transformation

Harris Corner Detection

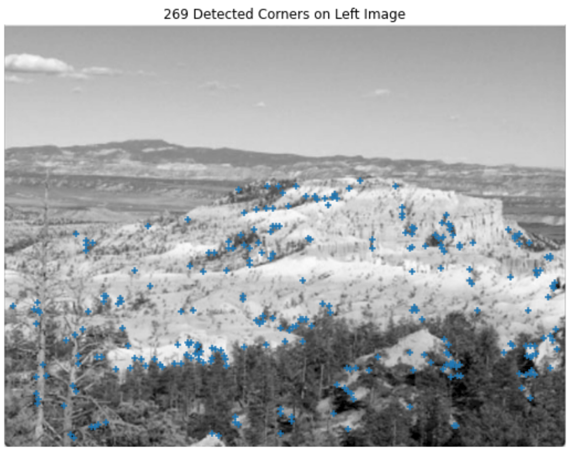
For this, the cornerHarris method from OpenCV was used to detect corners in the image with the following hyperparameter values:

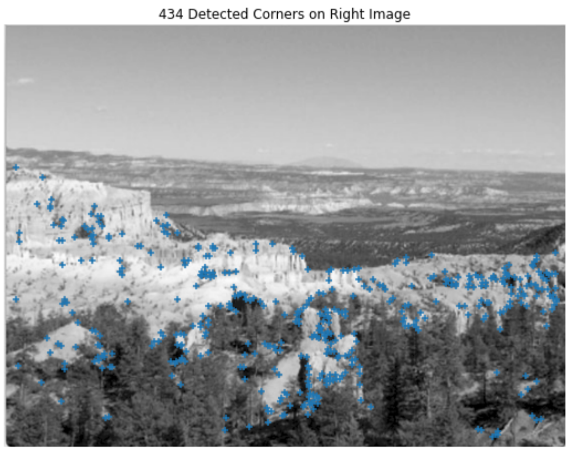
- *blockSize* (neighborhood size) = 3
- *kSize* – aperture parameter for the Sobel operator = 5
- *k* – Harris detector free parameter = .03





Increasing the *kSize* parameter above, the algorithm is not able to detect sharp corners so it ends up detecting fewer corners this time (see image below).





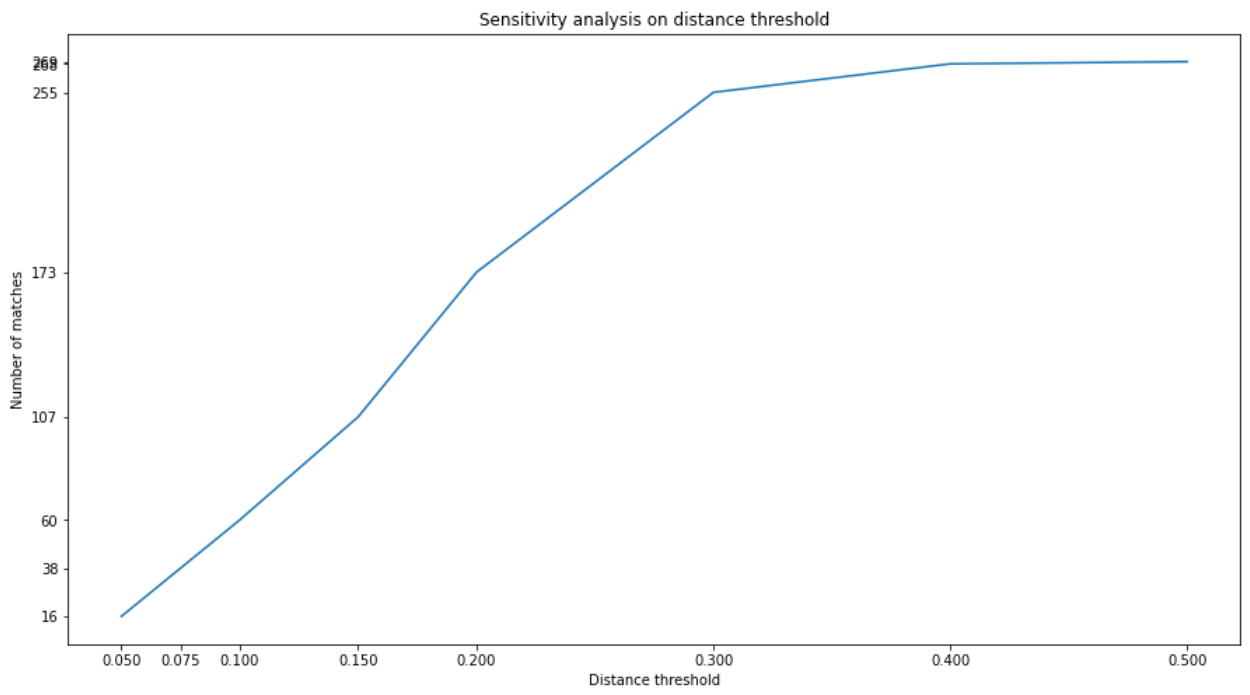
After detecting corners, peaks in the corner measure response image were found using corner_peaks from the skimage library which filters out many key points based on the min_distance parameter. Increasing this parameter leads to discarding more key points from the detected Harris corners set.

There are many ways to deal with border pixels like padding the original image with extra rows and columns based on the filter size, reflecting the image at the borders, and wrapping the image values. Here, the above-mentioned method offers the possibility to exclude border key points, which in the next step of the key-points description will suffer from missing information while creating patches around it. So, we exclude detected corner pixels that fall within the range of half patch size from the edge.

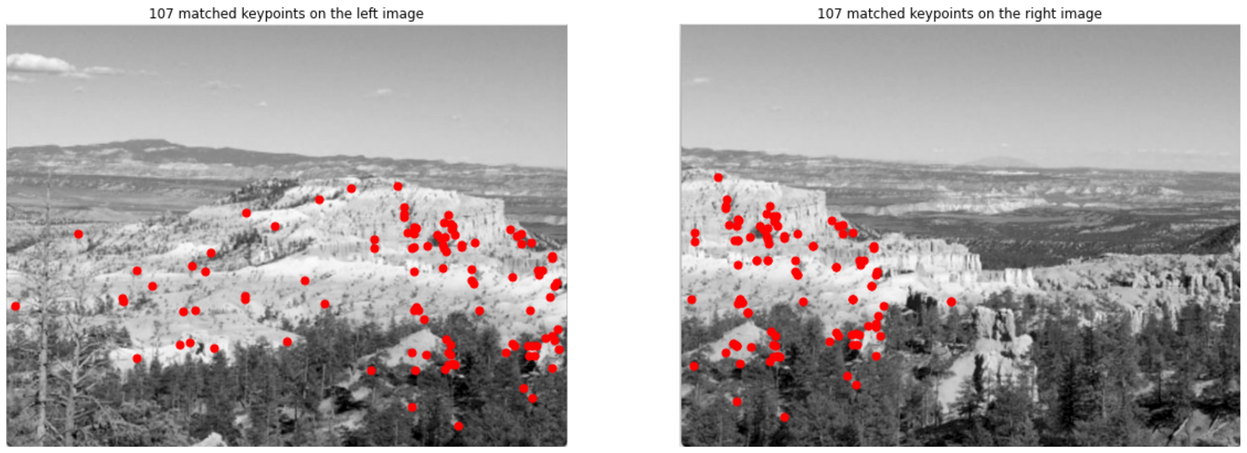
Key-point description

Each key-point filtered out from the Harris Corner detector in the above step was described by extracting fixed N-size patches around it and which results in a N x N sized vector.

After that, the distances between each descriptor from the first image were measured with each descriptor from the second image using the **euclidean distance** after normalizing each descriptor and normalized correlation. Then we iterate over the descriptors of both images and find the smallest distance for each pair. For both implementations, a threshold parameter was determined if the key-descriptor pair was a good match. Below, the sensitivity analysis is performed on this threshold parameter.

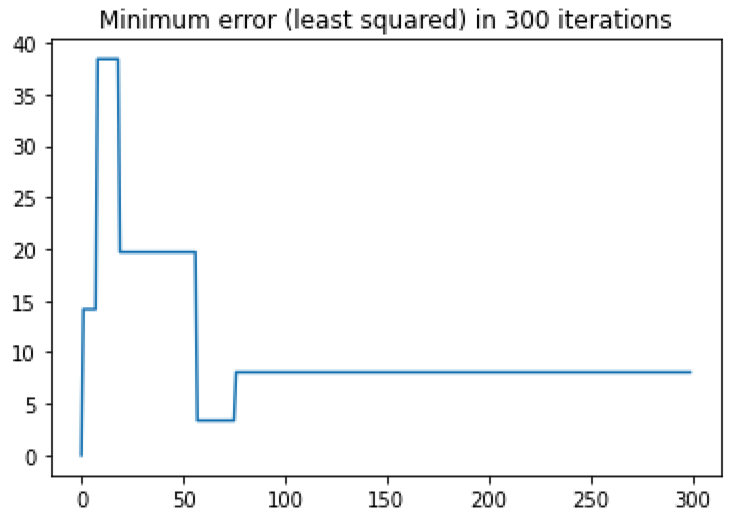


Since we only need a few hundred matches from all the detected key points, a threshold value in the range of 0.1 and 0.4 would be ideal.


107 matched keypoints on the left image
107 matched keypoints on the right image

Visually we can more or less estimate the performance of the matching algorithm. Notice that the same image patterns which are more distinct, like the mountains rocks, and some parts of the trees are detected in both images. Although, we can still notice that there are a lot of outliers and wrong matches between the image, while in reality these key points are not shared between the images.

Sensitivity analysis was performed also on the patch size that served as a descriptor of the keypoint. Although, the final results were compared after doing hyperparameter tuning on the other parameters in the next steps. But for the matter of covering the keypoint description topic in this section, the results will be provided here.



RANSAC

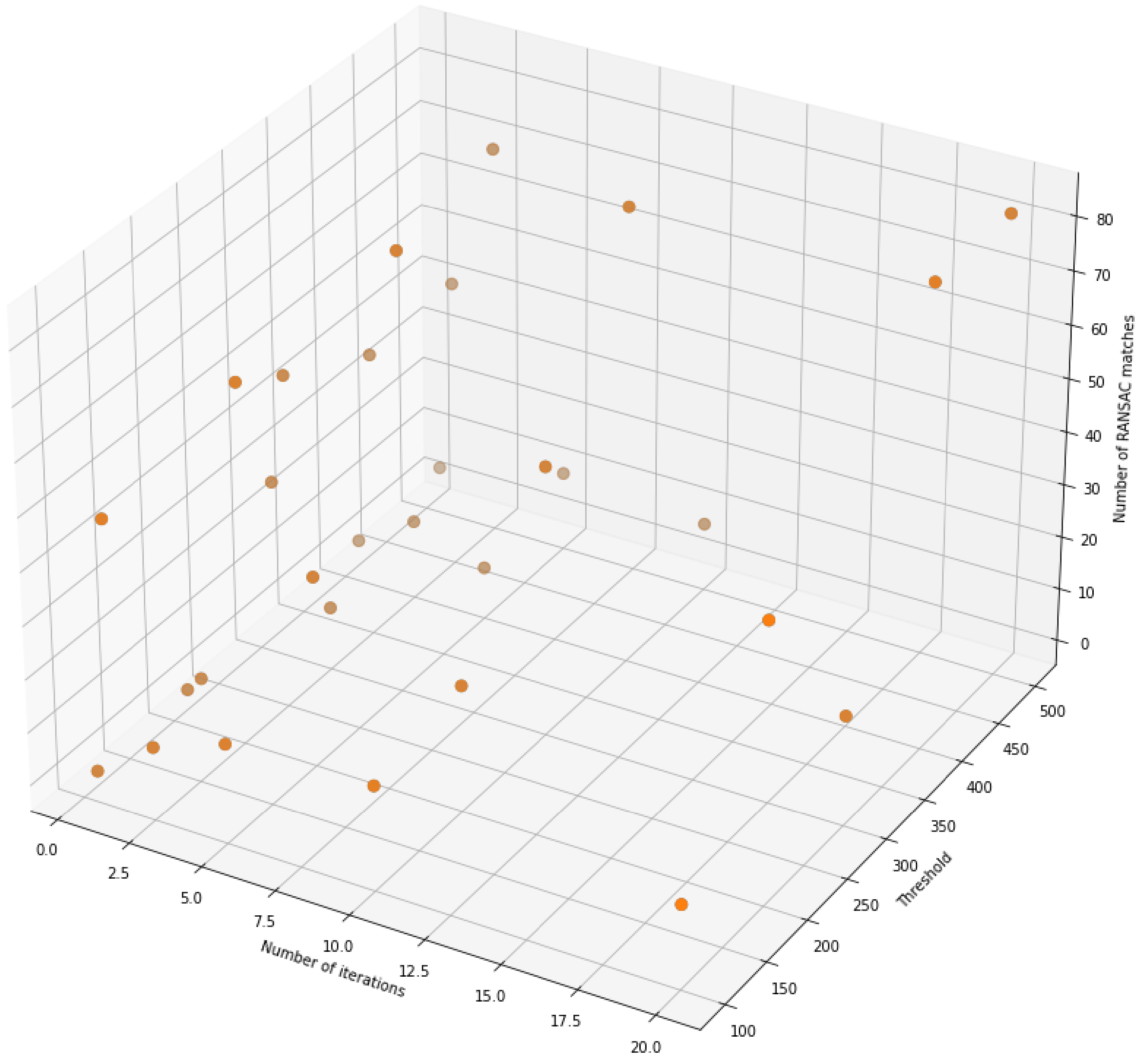
RANSAC algorithm was used to determine the model parameters of the affine transformation matrix. The big advantage of this algorithm is that it is robust to outliers.

For each iteration the steps are as follows:

- Select 3 to 5 random feature pairs
- Compute affine transformation matrix H
- Compute inliers where the squared distance between the point coordinates in one image and the transformed coordinates of the matching points in the other image is less than the specified threshold
- Keep the largest set of inliers
- Recompute the least-squares H estimate using all of the inliers

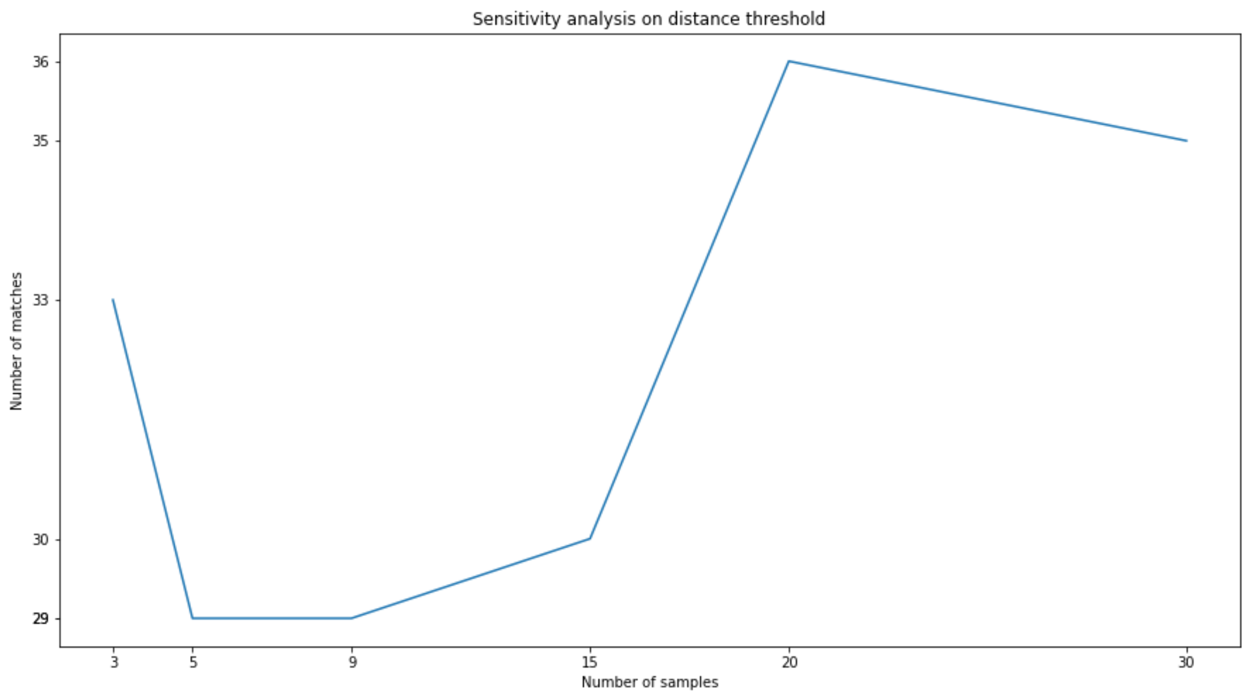
Evaluation of the algorithm was done by fine-tuning three parameters, the number of iterations, the number of sample size, and the inlier threshold (compared to the squared distance between point coordinates in one image, and the transformed coordinates of the matching points in the other image).

Sensitivity analysis was performed initially on the 2 first parameters and then on the number of samples with the following results:

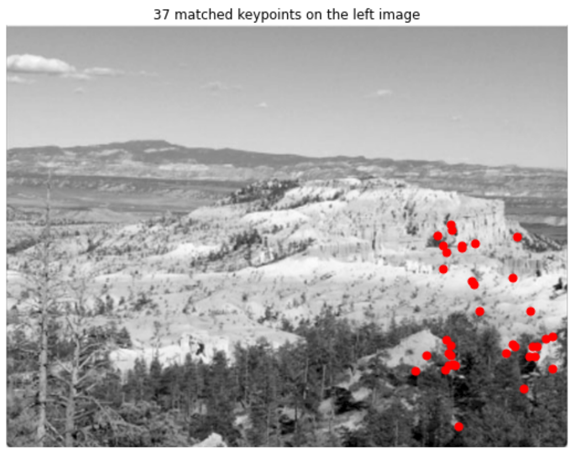


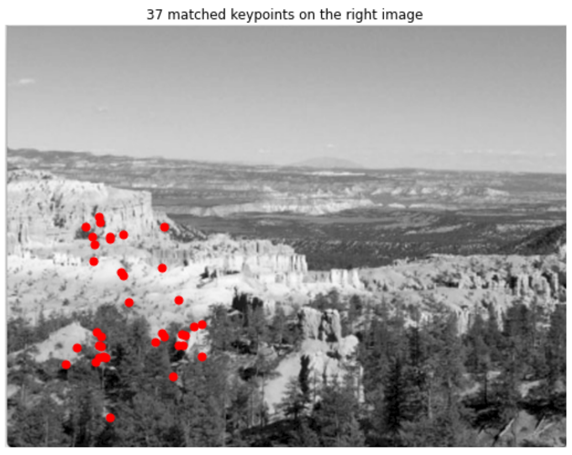
The above plot visualizes very clearly which parameter pair (number of iterations and threshold) achieves the highest number of inliers (number of RANSAC matches). A good number of matches would be 20 to perform the affine transformation.

With the acquired hyperparameters above, a sensitivity analysis was performed on the third parameter, the number of samples (image below):



After performing RANSAC the best matches were these:

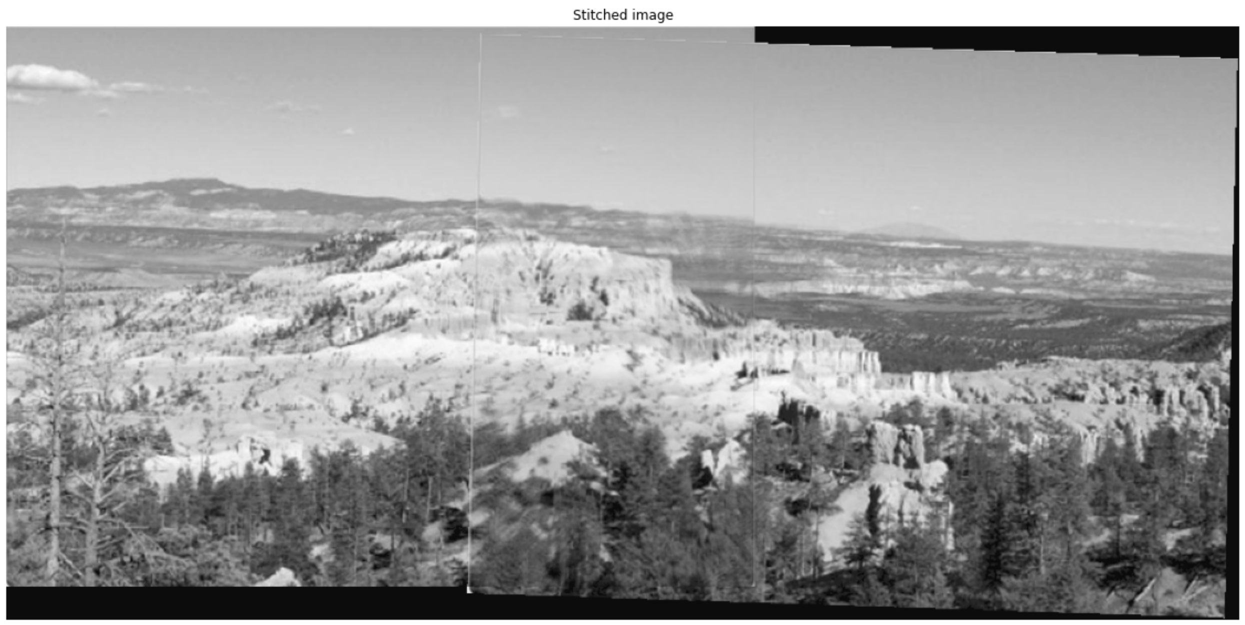




RANSAC estimated a good model removing all the outliers that previously were considered wrong features, and keeping only the inliers to apply the affine transformation in the next step.

Warp and transformation

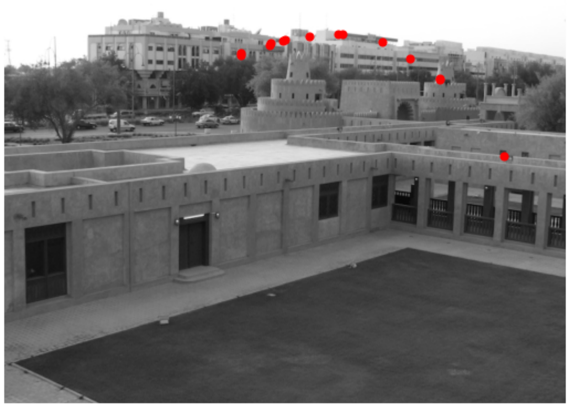
Now that we computed the affine transformation matrix, we know how each point is transformed in the destination image. The last step of the image stitching is to obtain the final stitched panoramic image using the homography matrix.



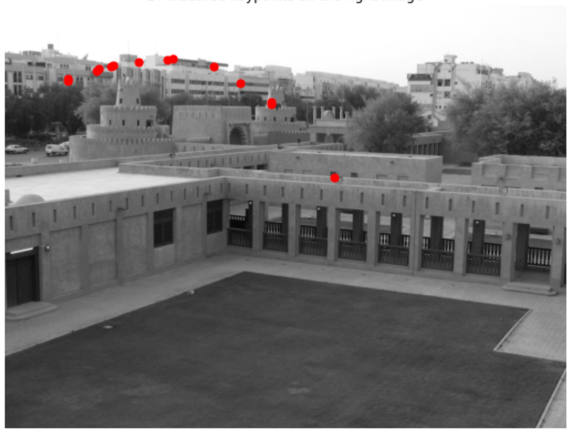
From the final panoramic image, we can notice a few blurry regions and stripes that cut off the image. There are many ways to achieve better merging to make the panorama look more natural like linear gradient blending etc.

Other experimentations

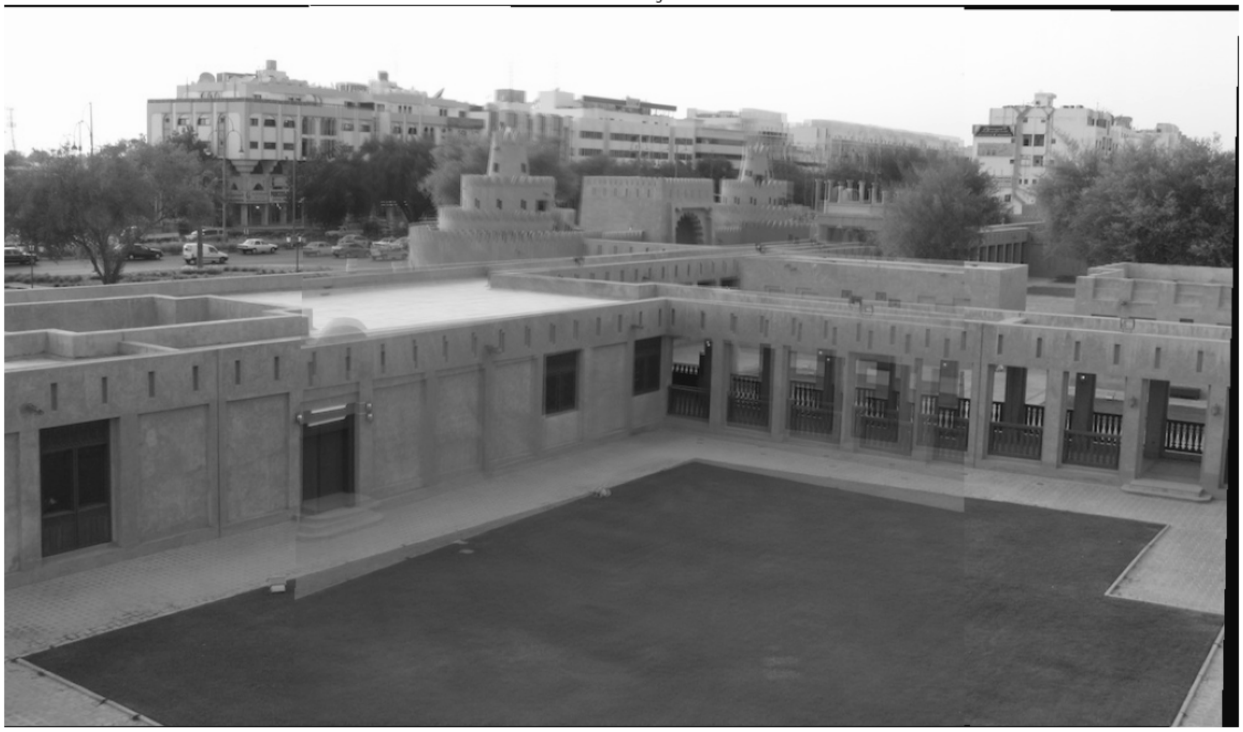
14 matched keypoints on the left image



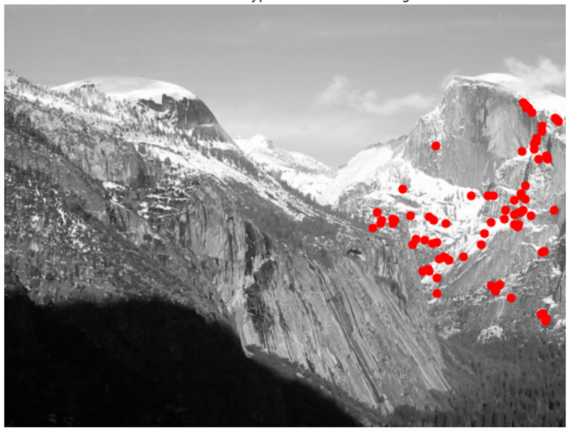
14 matched keypoints on the right image



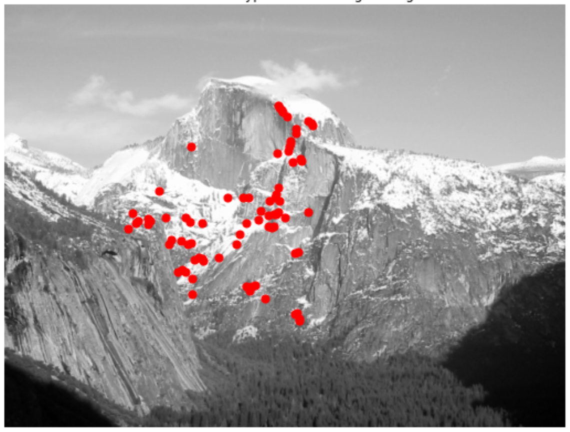
Stitched image



79 matched keypoints on the left image



79 matched keypoints on the right image



Stitched image