Computer Vision
Assignment 3

Pranjali Jain
16110119
June 28, 2020

Image Stitching for Panorama Generation

1. **Dataset**
   - Captured/Downloaded images of 5 scenes with 5 images each.
   - In a given scene, each image has at least a 20-30% overlap with the previous image.
   - The images captured are from stationary scenes with no moving objects.

2. **Grayscaling and Resizing:**
   - All the images are converted to grayscale.
   - The images in each scene are resized only if their size is too big while keeping the aspect ratio same.

3. **SIFT Feature extraction**
   - SIFT and ORB feature extractors available in python-opencv are used to find keypoints and corresponding descriptors in all the images.
   - Keypoints and descriptors help in finding correspondence between two images.
   - SIFT provides descriptors of size 128.

Now, we describe two ways for image stitching: First by implementing from scratch and second using inbuilt functions.

Implementation from Scratch

4. **Feature Matching**

- Between two images, we find the corresponding keypoints using euclidean distances or L2 norm.
- Distance between all keypoints is found using "scipy.spatial.distance.cdist", and then two keypoints from the second image with minimum distance from a given keypoint in the first image are selected.
- We compare the distances of these two keypoints from second image and select only those keypoints where there is a large difference between the difference in the keypoints distance between the keypoint with the minimum distance and the keypoint with the second minimum distance.
- Now we have point correspondences from first image and second image.

5. **Using RANSAC to find homography and inliers**
   - **Homography estimation**
     In order to find homography matrix, we randomly sample four points from the point correspondences obtained using feature matching.

     Using, $\mathbf{Ah = 0}$ we find the **h** matrix.
     h is the last row of **V** in the SVD of A, $\mathbf{A=UDV^T}$
     We reshape **h** to obtain **H**
     H is a 3x3 matrix
   - **Finding Inliers**
     We find inliers by transforming points in the source image to points in destination image using **H** matrix. The points that lie in a threshold are taken as inliers.
   - **Running RANSAC**
     We run ransac algorithm on the obtained **H** matrix and inliers, and prune the results. We run the algorithm till the size of inliers is found to be more than a threshold value(0.8*number of point correspondances).

6. **Finding Mapping of each point in destination image to the source image**
   - After estimating the **H** matrix, we find the mapping of all points on the destination image to the source image.

7. **Image warping**
   - Finally we create a canvas big enough to hold the stitched image.
   - We place the source image as it is on the canvas.
   - Next, we use the mapping obtained from the previous step, and place the destination image on the canvas
   - The canvas now shows the stitched panoramic image.

Using inbuilt functions

8. **Feature Matching**
    - Between two images, we find the corresponding keypoints using euclidean distances or L2 norm.
    - We do this using **BFMatcher** class in cv2, and **knnMatch** function. This works in a similar way to the implementation from scratch described above.
    - We find the point correspondences between two images that have the minimum distance from each other.

9. **Finding homography and inliers**
    - **Homography estimation**
      Using the point correspondences obtained using feature matching, we find the homography matrix using the function **findHomography**.

    - **Finding Inliers**
      The inlier points are given by the same function **findHomography**.

10. **Finding Mapping of each point in destination image to the source image**
    - After estimating the **H** matrix, we find the mapping of all points on the destination image to the source image using **WarpPerspective** function.

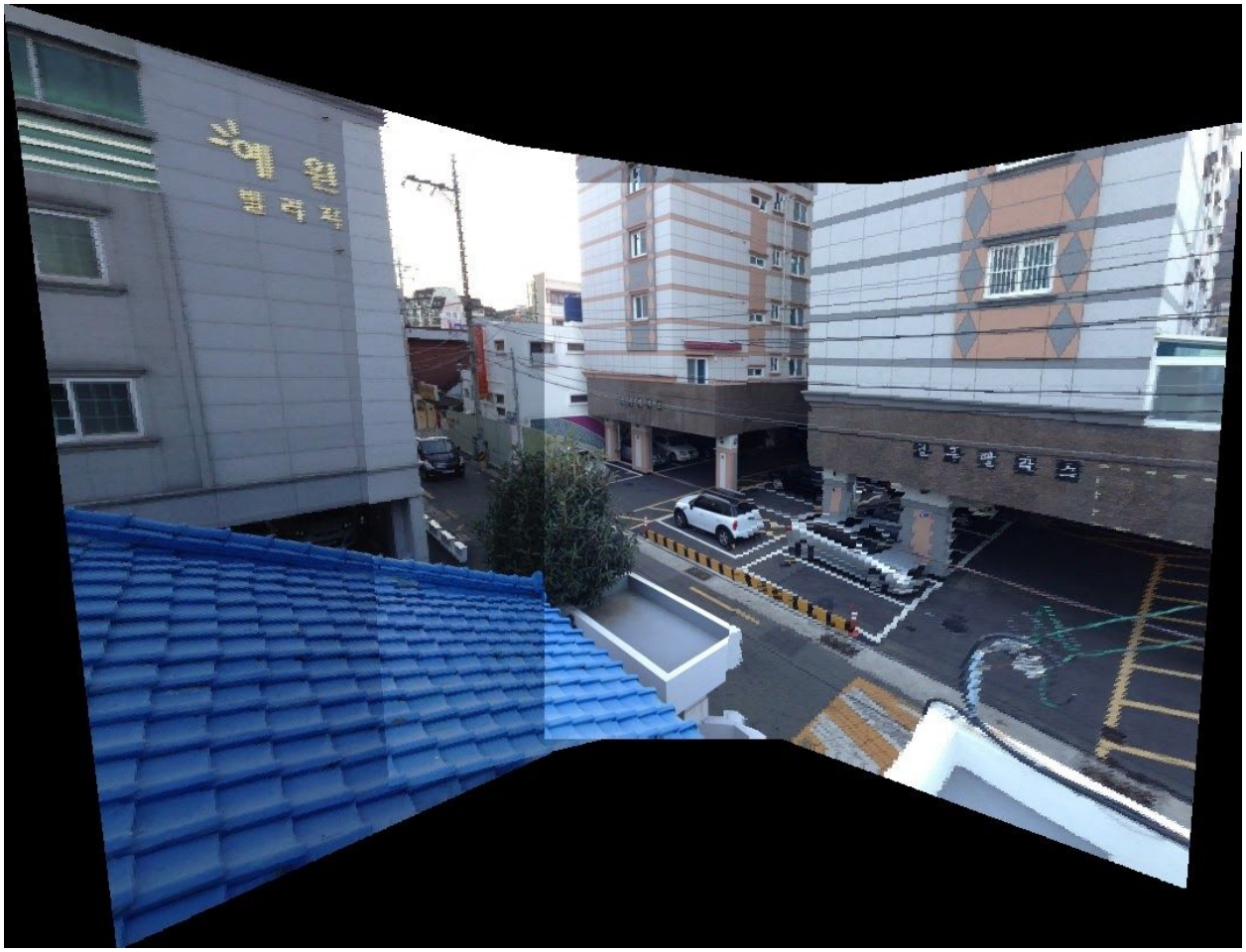
11. **Image warping**
    - A canvas big enough to hold the stitched image is also created by the **WarpPerspective** function on which the destination image is placed.
    - We place the source image as it is on the canvas.
    - The canvas now shows the stitched panoramic image.

Sample images from the dataset and Panoramas

Scene 2

Input images

Implementation from scratch



Inbuilt implementation

Scene 1

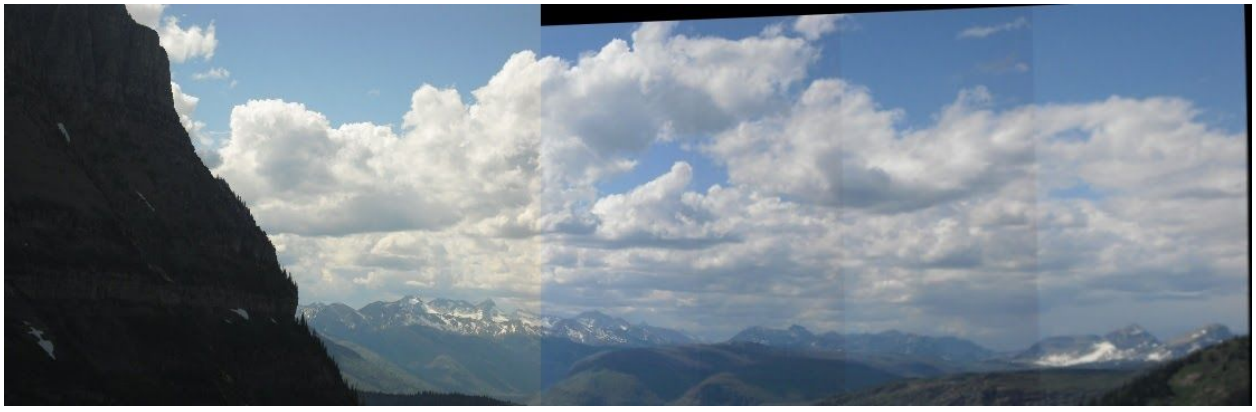
Implementation from scratch

Inbuilt implementation

Scene 3

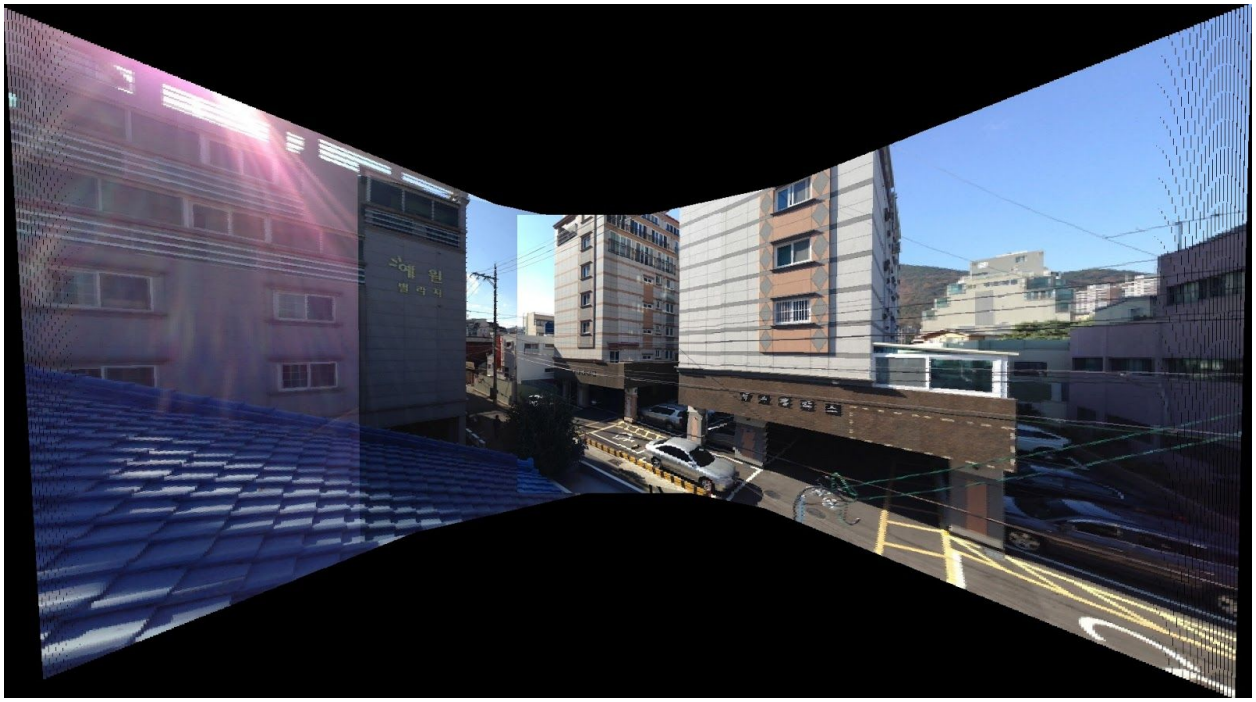
Implementation from scratch

Inbuilt implementation



Scene 4

Implementation from scratch

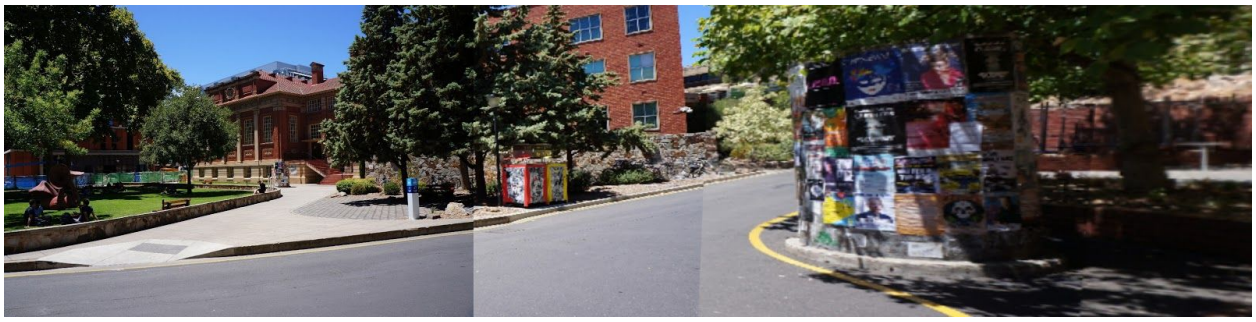


Inbuilt implementation

Scene 5

Implementation from scratch

Inbuilt implementation



References:

1. http://slazebni.cs.illinois.edu/spring18/assignment3.html
2. https://www.pyimagesearch.com/2018/12/17/image-stitching-with-opencv-and-python/
3. https://towardsdatascience.com/image-panorama-stitching-with-opencv-2402bde6b46c
4. https://kushalvyas.github.io/stitching.html
5. https://medium.com/analytics-vidhya/image-stitching-with-opencv-and-python-1ebd9e0a6d78
6. https://github.com/kushalvyas/Python-Multiple-Image-Stitching
7. https://github.com/yihui-he/panorama
8. http://6.869.csail.mit.edu/fa12/lectures/lecture13ransac/lecture13ransac.pdf
9. https://docs.opencv.org/master/d9/dab/tutorial_homography.html
10. https://inst.eecs.berkeley.edu/~cs194-26/fa18/upload/files/proj6B/cs194-26-aeh/website/