COL780 : Computer Vision
Assignment 2

Gaurav Jain : 2019CS10349

1 Method

I have implemented the Hessian method for corner detection. To match the detected corners between adjacent frames, I have used proximity based matching and filtered them using a sum of square difference (SSD) threshold. To stitch the frames together, I have used affine transformation and warped them using OpenCV methods.

1.1 Hessian Corner Detector

The Hessian Corner Detector method is based on computing the Hessian of the image pixels. The eigen values given by singular value decomposition of the Hessian matrix are used to determine whether a point can be used as an important point in panorama generation.

But, the same information can be extracted from the determinant and trace of the original matrix. So, I use the quantity $Det(H) - \alpha\ Trace(H)^2$ and keep a threshold as suggested in the literature. The values of α is 0.05 and the value of threshold is 1% of the maximum value of the above mentioned quantity. I also do non-maximal suppression by taking the point only if it is the maximum in its neighborhood.

Before calculating the Hessian, I have smoothed the image using OpenCV's `GaussianBlur` function ($\sigma = 1$). To calculate the Hessian, I use the discrete differentiation method. To do so, I have utilized the filters for I_{xx}, I_{yy} and I_{xy}. To pass the filter over the complete image, I have used the `filter2D` function of OpenCV. I have again smoothed the calculated values of I_{xx}, I_{yy} and I_{xy} for better detection.

The code for this part is present in the `hessian_corner_detection` function.

1.2 Matching Corners in Adjacent Frames

After calculating the corners for all frames using the above discussed method, I match adjacent frames using proximity where I take the closest (Euclidean distance) corner point in the second frame to a corner point in the first frame.

To refine the matched corner pairs, I put a threshold on the sum of square difference of the neighborhood around the corner points of the two frames. The value of threshold is 250 in my implementation.

The code for this part is present in the `proximity_matching` function.

1.3 Stitching Frames to create Panorama

After producing the matched corners between adjacent frames, I estimate the affine matrix using these matched corner pairs by applying the least square fitting method. I convert the points to projective co-ordinate system before applying the least square fitting method. The code for this part is present in `estimate_affine_matrix` function.

Since the homography of homography is also homography, we can combine the estimated affine matrices and directly project all frames to a particular frame. I chain together the affine matrices of adjacent frames to calculate the affine matrices with respect to the first frame. This method is present in the `main` function.

To calculate the dimensions of the panorama, we need to find the positions of the corner points of all frames after projecting them to the first frame. To do so, we require the affine matrices derived in the previous paragraph. I determine the new positions using these affine matrices and take the maximum and minimum of the projected corner points to calculate the dimensions. This method is present in the `get_dimensions` function.

I warp all the frames using OpenCV's `warpAffine` function and estimated affine matrices. Finally, I take the bitwise OR of all the warped frames to generate the final merged frame. I remove the black borders present in the final image. (Present in the `main` function)

2 How to run the code?

Run the following command:

```
python assign2.py input_dir_path output_path dataset_name(1/2/3/4/5)
```

3 Generated Panoramas

The generated panoramas of the second dataset (*dataset2*) is shown below:

Figure 1: Generated Panorama of *dataset2/1/*

Figure 2: Generated Panorama of *dataset2/2/*

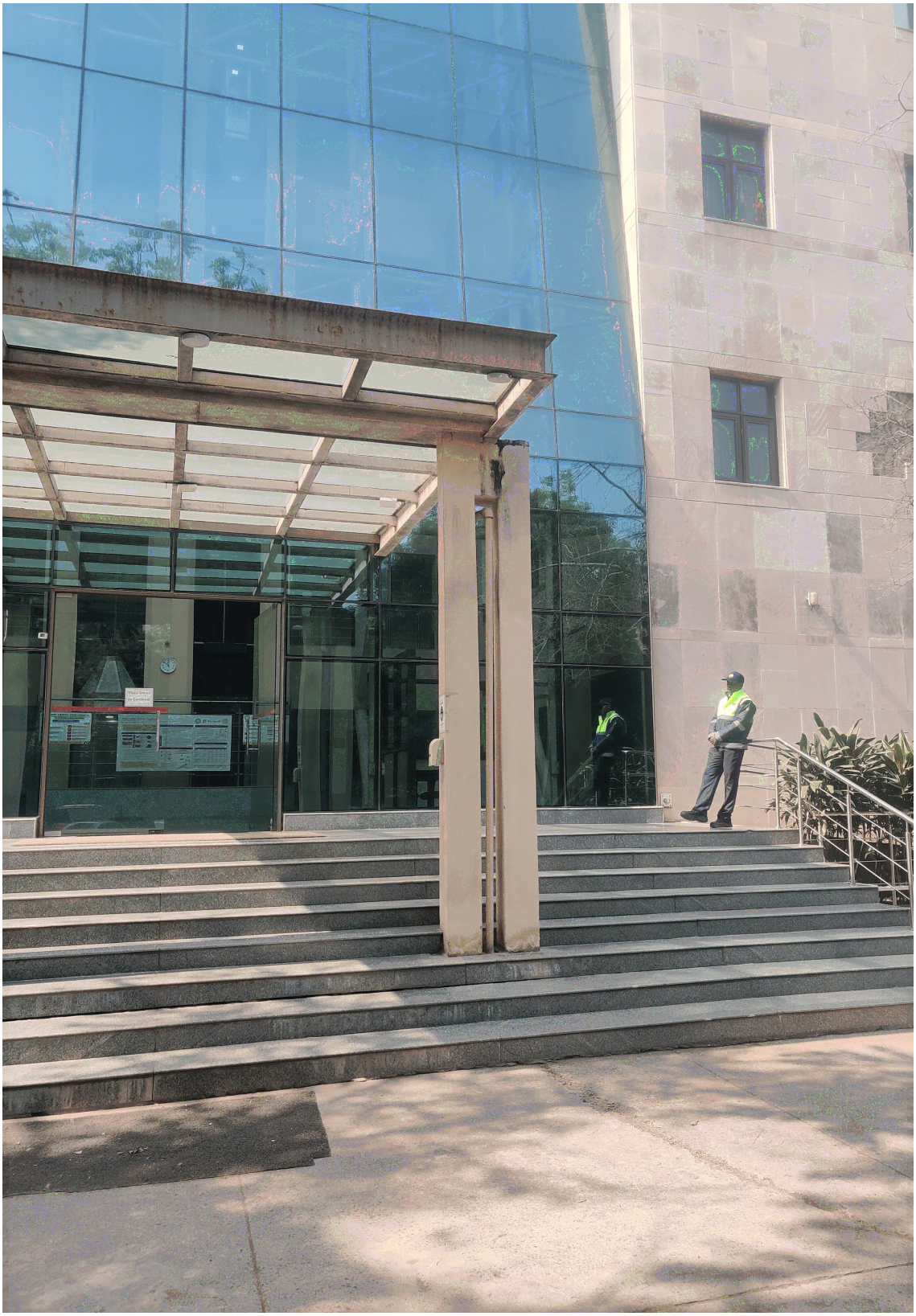

Figure 3: Generated Panorama of *dataset2/3/*

Figure 4: Generated Panorama of *dataset2/4/*

Figure 5: Generated Panorama of *dataset2/5/*

4 Link

This link contains the generated panoramas of *dataset2* in the original quality.