Assignment 2

Panorama Stitching

Hari Krishnan CK

2021EEY7583

Panorama stitching includes the following steps which are discussed in detail. They are

- Gaussian blurring: blurring the image using the gaussian blur technique
- Hessian corner detection: to find the key points
- Descriptor finding: to find the descriptor around the key points
- Matching descriptors: to find the best match descriptors between the frames
- Affine transformation: to find the affine transformation matrix between the frames
- RASAC : to fithe nd best fit affine transformation matrix
- Pairwise Stitching: Stitching the two images together after applying best-fit fit affine transform
- Multi Stitching: an algorithm to stitch all the images together using a pairwise stitching function

Hessian Corner Detection

Hessian corner detection is a method to find the key points especially corners using the Hessian matrix for the image at a pixel level. We calculate the Hessian matrix for a pixel which is given by $[[I_{xx}, I_{xy}], [I_{xy}, I_{yy}]]$ after which eigenvalues are found. If the smallest eigenvalue is greater than a threshold (cornerThreshold in code) we say the pixel is a key point.
An example result of corner detection is shown below:

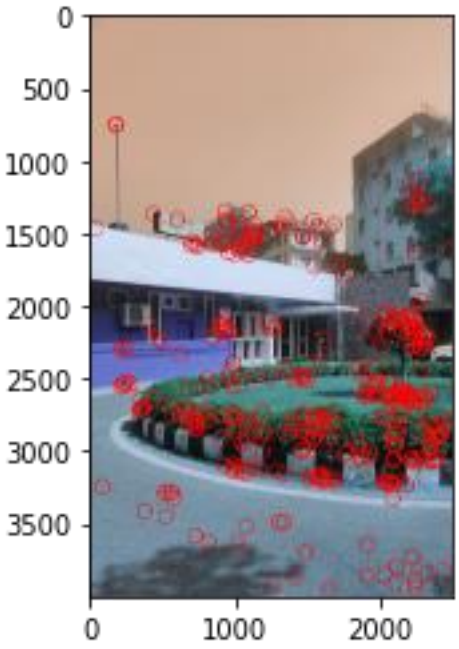


Figure 1: Corners detected in an image

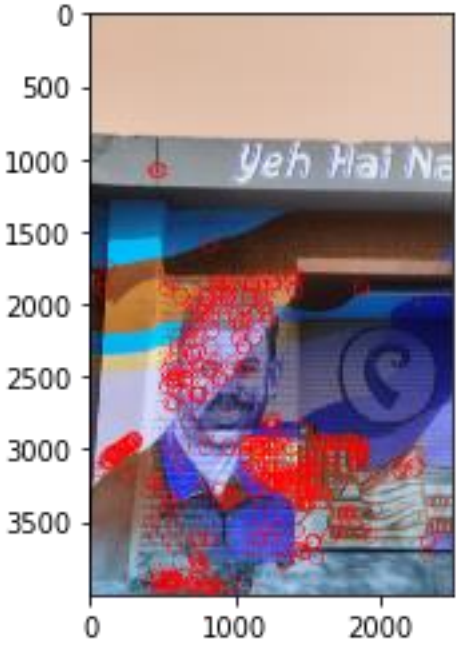


Figure 2: Corners detected in another image

Descriptor Finding

After we have keypoints we need to find the descriptor for key points so that we can match them with the next image. Two methods were implemented and tried:

- HOG descriptor: For each pixel gradient magnitude and angle is found out and depending on the angle which lies between (-90 to 90), 9 bins are used (-

90 to -70, -70 to -50,etc). Depending on the gradient angle, gradient magnitude is added to the corresponding bin.

- Patch around the keypoints: A small patchsize (10X10 in implementation) around the keypoint is extracted after which the patch is mean normalized to have zero mean and unit variance. This 100 dimensional vector is considered as the descriptor.

Both approaches gave similar results and there was not significant changes in both the approaches.

Matching Keypoints

To match the keypoints obtained using the Hessian corner detection, we use the proximity and sum of squares approach. Initially a small patch around the keypoint is extracted and normalized with zero mean and one standard deviation. This is done to reduce the variance between the images due to illumination, etc. Then the best match with the keypoint patch in the second image is found using the L2-norm (sum of squares between the patches). If the L2-norm between the patches is less than a threshold (matchingThreshold in code) the two patches are considered a match.

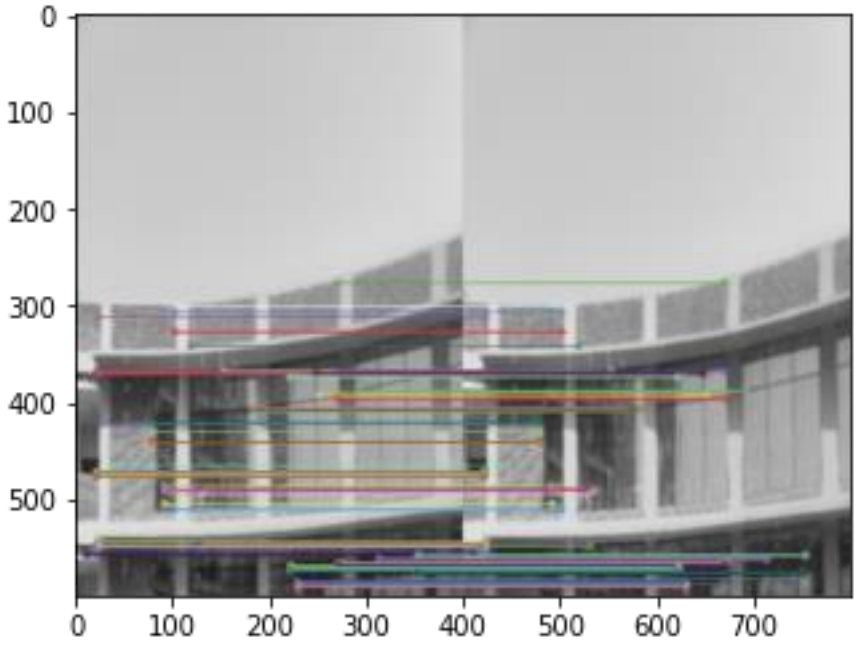


Figure 3: Matching descriptors between adjacent frames

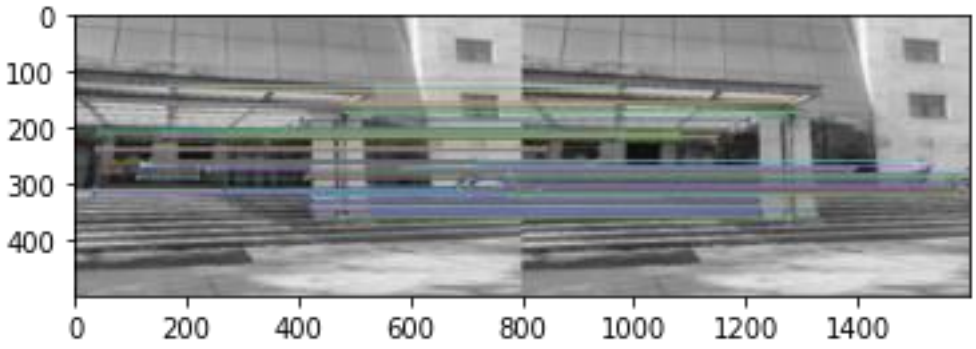


Figure 4: Matching descriptors between adjacent frames

Affine Transformation

Once we have the keypoint matches between two pictures we can find the affine transformation from one image to another i.e AX=Y (np.linalg.lstsq is used in code) where X=keypoints in image 1, Y= keypoints in image 2 and A is the affine transformation between the two images. But we have a lot of keypoints between the images and we need to find an affine transform removing the wrong matches and hence finding the best fit.

RANSAC

To find the best fit affine transformation matrix between the images, the RANSAC algorithm is used which has the following steps implemented.

- Step 1: Sample nPoints data from the total sample data (let's call this idxSelected). In code it is implemented as nPoints=int(max(nPoints,seedThreshold*len(X)))
- Step 2: Fit the affine transform to these idxSelected points.
- Step 3: Calculate the error on non-selected points.
- Step 4: Find the samples in non-selected where the error is less than pointInlineError and if that number is greater than fitThreshold*len(non-selected) then continue to Step 5 else go to Step 1.
- Step 5: Fit the affine transform on the idxSelected points and the samples in Step 4.
- Step 6: Check if the total error on all the samples is lowest, then consider the new affine matrix as the best one obtained else continue

Parameters used in simulations are

pointInlineError=20,iter=2000,nPoints=3,fitThreshold=0.8,seedThreshold=0.4

Pairwise Stitching

Pairwise Stitching between two images has the following steps implemented:

- Centering both images (padding with zeros on all sides)

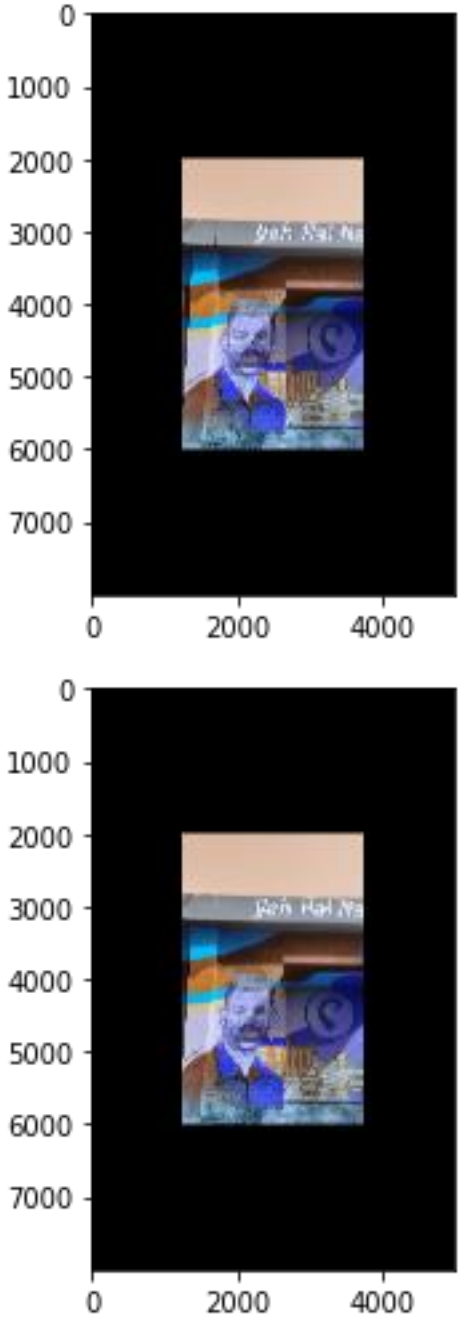


Figure 5: Centering both the images before applying affine transform

- Applying the affine transformation on the right image
- Combining the two images in such a way that if left image pixel is non-zero take left image value else take the right image pixel

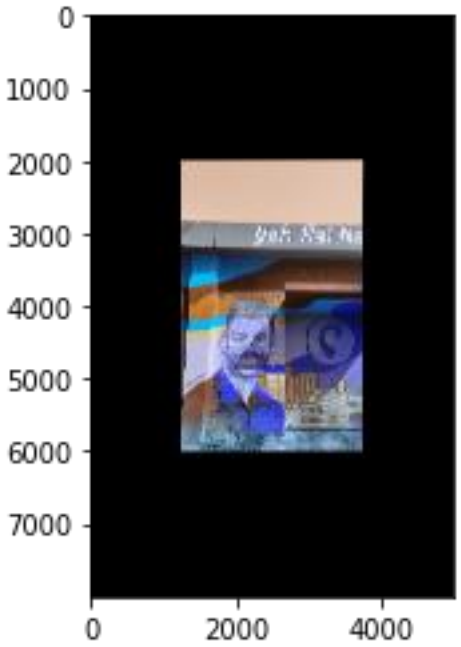


Figure 6: Combing the two figures in Figure 5 by taking the left pixels if non-zero

- Removing the black borders from the combined image

Figure 7: Final image after removing black borders

Multi Stitching

Now we have to stitch all the images continuously. Two algorithms were implemented for this purpose

Left-right Stitching: We take the first frame given as the reference frame and keep on stitching as the next frames are given .i.e stitch frame 1 and 2 then stitch frame 3 with the

result of stitching frame 1 and 2. The total number of calls to pairwise stitching will be N-1 where N is the total number of frames to stitch. In the datasets given, this approach works fine. In code it is implemented as PanaromaMatching.

Pyramid Stitching: In this stitching technique, two consecutive frames are stitched together in one go. i.e Frame 0 and 1 let's call it stitched frame01, Frames 2 and 3 let's call it stitched frame23, etc and in next iteration stitched frame01 and stitched frame23, and so on on till there is only one image left . Then these stitched frames are stitched together in a recursive way. The total number of calls to pairwise stitching will be N-1 where N is the total number of frames to stitch. In code it is implemented as PairWisePanaromaMatching.

Results:

In this section, results for panaroma matching for the given datasets is given below

Parameters or thresholds used are

cornerThreshold=350

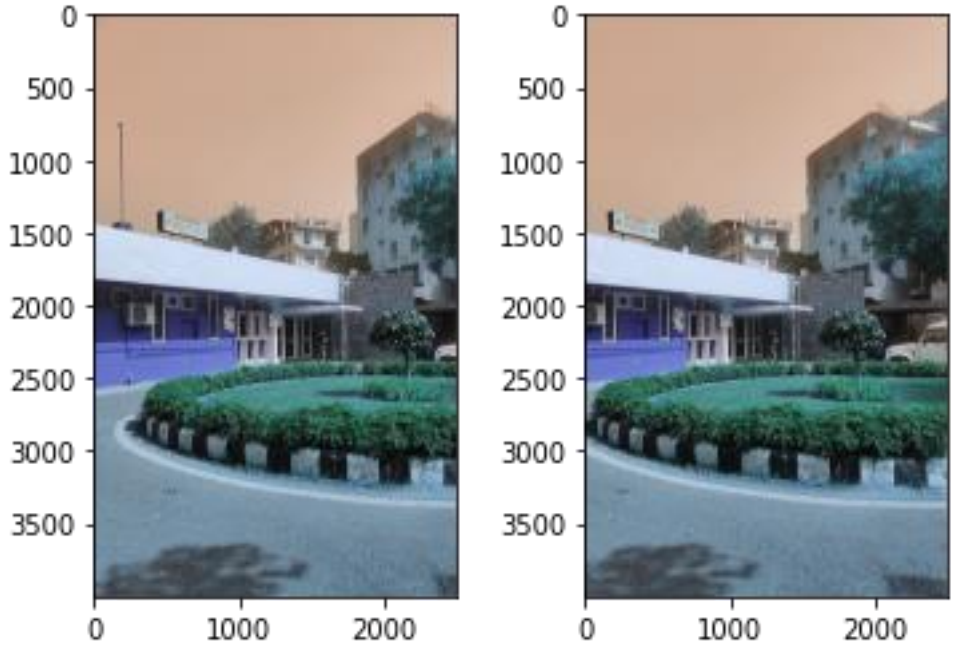
matchingThreshold=20

proximityThreshold=300

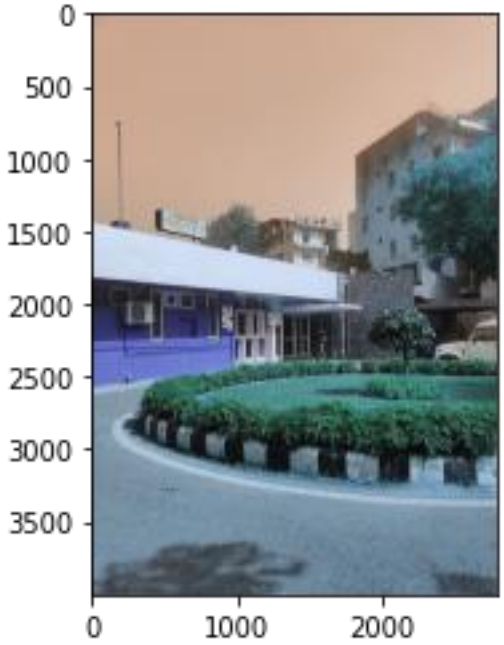
Results for the datasets given are shown below:

An example explanation for one image is given below

First Frame and Last Frame



Stitched image




अस्पताल
H HOSPITAL H
IIT DELHI


Yeh Hai Nayi
@Delhistreetart