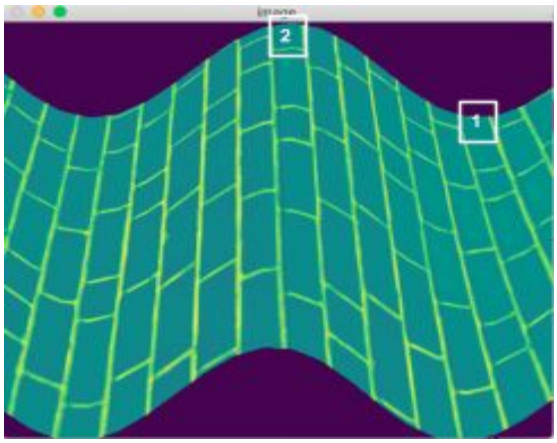
Reflection Essay

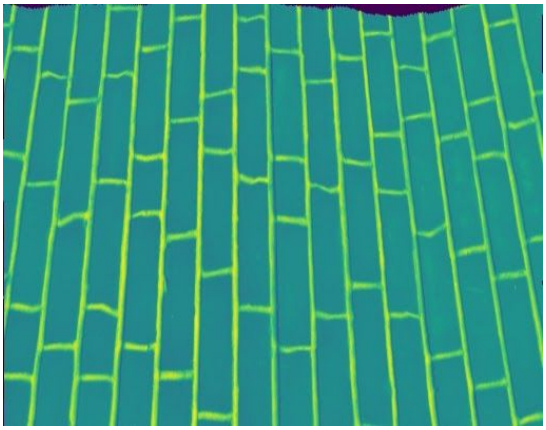
Nijil 183079009
Arjun 193079014
Vinuthna 19307R003

1) Piecewise Affine Transformation

As the distorted image is sinusoidal in nature, we need to find its amplitude and the number of cycles in order to reconstruct it in the best possible way. Here I have used a mouse callback function cv2.EVENT_LBUTTONDOWN to attain 2 points. Choose the points in the specified order as shown below in the distorted image.



a) Distorted image(input)

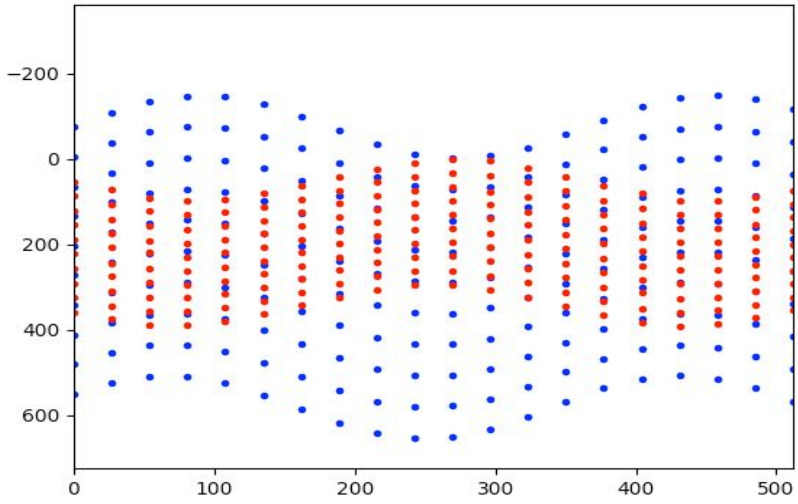


b) output

The difference in vertical coordinates (here it's ~100)gives twice the amplitude of the sinusoid. The difference in horizontal coordinates gives the length of half cycle(~ 170). Thus from size of the image i.e number of columns (512 in this case) and this difference we can obtain the number of cycles.(~ 1 ½ cycles).

Assuming a meshgrid using 20 columns as source which is to be attained. The destination would be the sinusoid as mentioned. Scikit image library has functions for various transforms, among the various transforms tform = PiecewiseAffineTransform() is used for our problem.
tform.estimate(dst,src) predicts src from dst.

Q): How to select the grid-point correspondences for this problem? Explain the process.



The red dots indicates the portion in the src or the distorted image given to be transformed as a rectangular image as obtained in the output. Thus some of the points in the src are mapped with points outside the image in dst(i.e the black portions).

References:
https://scikit-image.org/docs/dev

2.1) Manual Mosaicing

a)

4 point are actually needed , but to get good result we need more than 4 points. We had taken atleast 6 points.
WarpPerspective() has been used , We may want to map back the intersecting points back to parallel form or vice versa.

b)
We downloaded remaining pictures of “campus” dataset and used two related images(ie, with some correspondances)

c)
i)Output after mosaicing will be unchanged, if the homography is Identity .It is also possible for the size of the output and re to be same and both images same, if I_2 is actually a subset of reference image.
Corresponding points influences the stitched result, if the points are completely uncorrelated ,then we would see a resultant(stitched image)which will not resembles the real world out there.
ii)
100pixels each
some pts in the range [10,20] of J_1 maps to some points in the range [85,95] of J_2

Case 1: Let J_1 be the reference image and J_2 be the image to be stitched to the reference.
$AX_2+B=X_1$, choice of A and B for different mapping can prevent the output size from exceeding 100 pixels.(Its mapping

For example: 85----->19(of J_1)
95----->20 if A=0.1 and B=10.5

So point like X_2=1----->10.6(not present in discrete grid)
X_2=100--->20.5(not in discrete grid)
This will preserve the size of the resulting image.

Figure 2.a-Image 1 is kept as reference

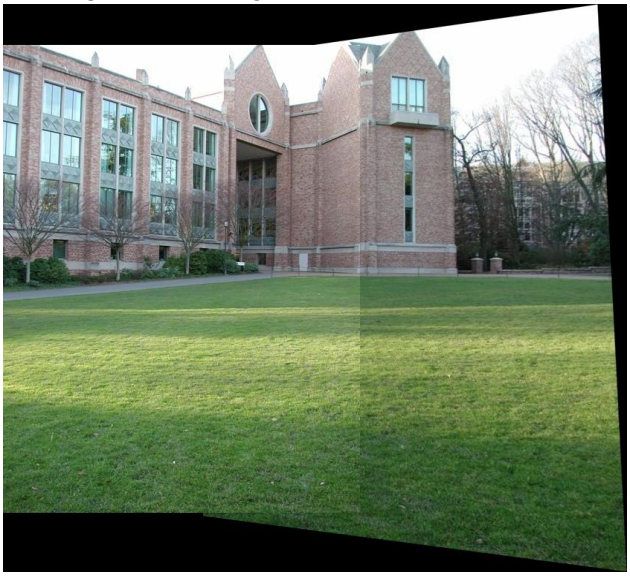


Figure 2.b- Image 2 is kept as reference

Figure 2.1 a) image 1, image 2 and image after mosaicing the left

Figure 2.1 b) image 1, image 2 and image after mosaicing

Figure 2.1 c) image 1, image 2 and image after mosaicing

2.2) Auto Mosaicing

2.b

i)

We didn't use any explicit filter. We took the best matched features from the two images. These were sorted on the basis of Hamming Distance. There was no need to filter these points ,as the first 25+ correspondences were correct for all three images.

ii)

Descriptors of each feature of both images were computed, Hamming Norm was used to compare between descriptors .Second parameter ,crossCheck is set to true, then the BFMatcher will only return a consistent pair. Such technique usually produces best results with minimal number of outliers when there are enough matches. These Points were sorted on basis of their distance and best
10,20 etc correspondence were taken for different dataset.

Iii)

Yes, it in fact worked better using the automatic algorithm, earlier case(manual) chances of human induced error while marking was quite high,so we may possibly see many misalignment , here since features are found automatically, there is no need to worry about human induced error.

Figure 2.2 a) image 1, image 2 and image after mosaicing the left

Figure 2.2 b) image 1, image 2 and image after mosaicing

Figure 2.2 c) image 1, image 2 and image after mosaicing

2.3) Let's generalize ($3 \leq n \leq 5$ input images)

Show the results for the given datasets as well as on your own datasets (of 5 images). Mention the results when your code is working and when it is not working for some reference images or some datasets (if that's the case). (Why do we say this?)

The quality of the results we got depended on the keypoints that were identified and the number of correct point correspondences that were given by the inbuilt function. We found that for certain datasets certain maximum number of point correspondences worked best, so we kept it as a hyperparameter. This hyperparameter had to be tuned to get the best results for individual datasets.

Q.2.3.1: Explain the method you followed. Do you get the result if you choose any image to be the reference image? If no, what issues are you facing? Give your opinion on the reason behind these issues.

The method followed for stitching two images include the following steps: finding keypoints, matching corresponding keypoints using features (hamming distance), finding homography between the matched keypoints, applying homography to the query image and overlaying the reference image over the transformed query image. This process had to be done multiple times for stitching multiple images. However, the order in which the images are stitched is important, since the images in the dataset are given in a correct order we could sequentially add the stitch the images one after the other. For sequentially stitching the images the entire set of images had to be split into images to the left and right of the reference image. Position of the place where the overlay of the reference image had to be was kept constant at the top left corner. So for stitching the images to the left of the reference image reflection about the y-axis had to be done.

Q.2.3.2: Here, we assumed that you know the sequence in which you have to stitch the images. How would you modify your approach if you don't know the sequence? Describe the approach briefly.

If the images are not ordered then we would find the image pairs with the most area overlap and join them first. A quantitative way to determine the amount of overlap would be to find the keypoints of all the images and find the pairwise matching between the images, the image pair having the best point correspondence match can be concluded to be adjacent images and they can be stitched together.This process can be repeated for other images.

Q.2.3.3: How does the functionality you are providing (or could provide) differ from the Stitcher class.

The stitcher class is based on the Brown and Lowe method and is insensitive to:

- Ordering of images
- Orientation of images
- Illumination changes
- Noisy images that are not actually part of the panorama

Additionally, image stitching method is use of gain compensation and image blending resulting in better results.The main difference that our method has with the stitcher class are the above

Q.2.3.4: The mosaicing results show a “seam". What techniques can be used to remove the seam?

Seam can be removed by calculating the final value of pixels by performing weighted average of pixel values of images that are overlapping over this pixel. Weight can be based on the distance from the pixel to the edge of the image. If the pixel is closer to the center of the image that it belongs to, then is more important and the weight is bigger.

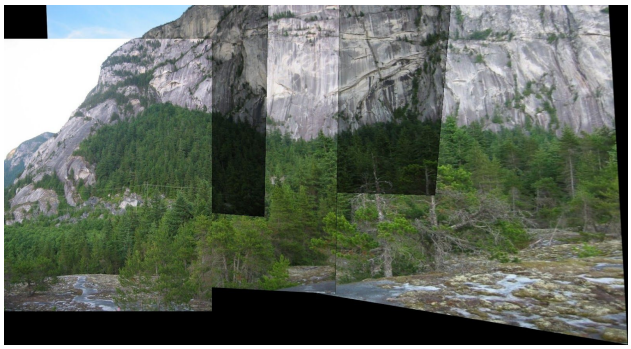
Results

1. **Mountain**

   This dataset had a good amount of overlap between the images and so the outputs for the same are also good. The reference image used also determines the quality of the stitched image. 20 correspondences were used for this dataset.

   Input:

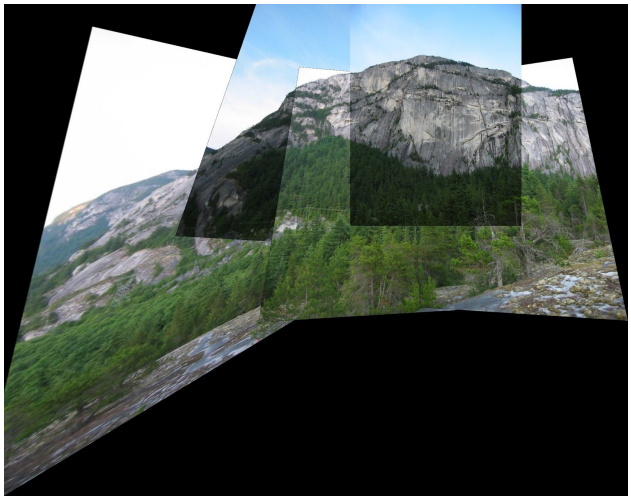
   Output:



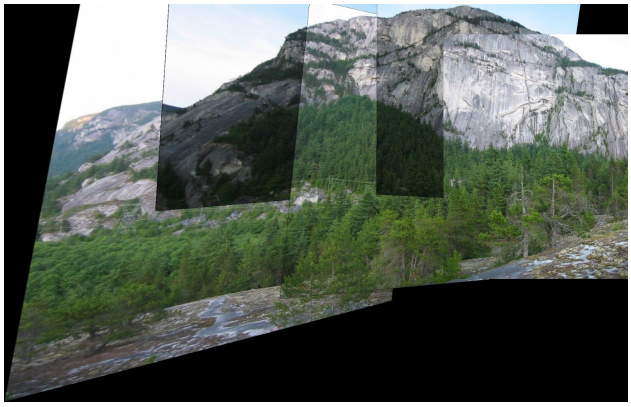
Reference image = 1

Reference image = 2

Reference image = 3



Reference image = 4



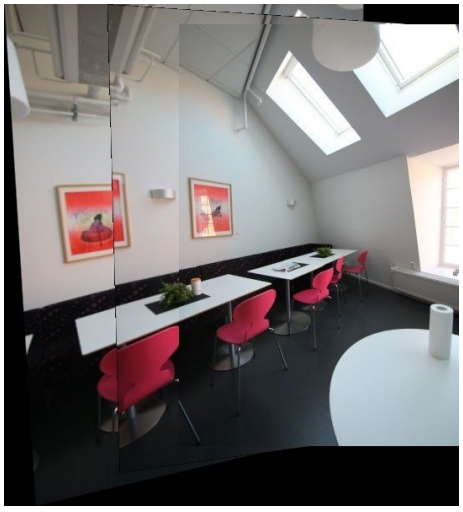
Reference image = 5

2. **Room**

This dataset had very distinct points that can be used as keypoints, which meant the correspondences that were identified was good. The dataset gave good results for almost all correspondences we tried. 25 point correspondences were used.

Input

Output:



Reference image = 1,2,3 respectively

3. **Yard**

This dataset caused some problems because of repeating patterns in the images. The keypoints that were identified could not be matched perfectly at times. Additionally the images were ordered from right to left physically. We used 25 correspondence points. It did not give good results for the fifth image as the reference image.

Input:

Output:

Ref image = 5

Ref image = 4


Ref image = 3

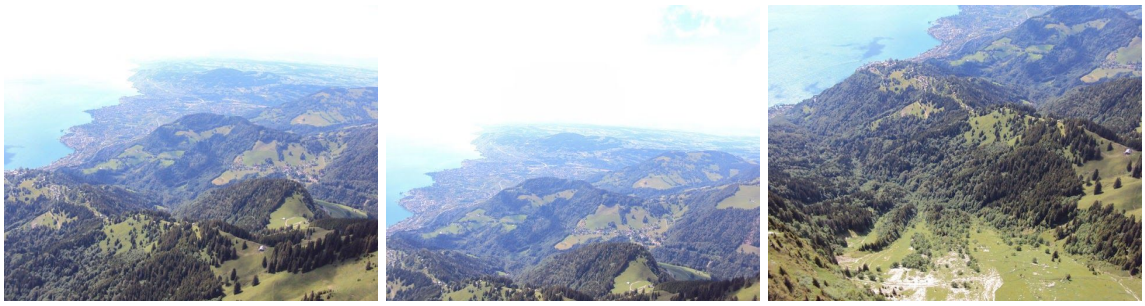

Ref image = 2


Ref image = 1

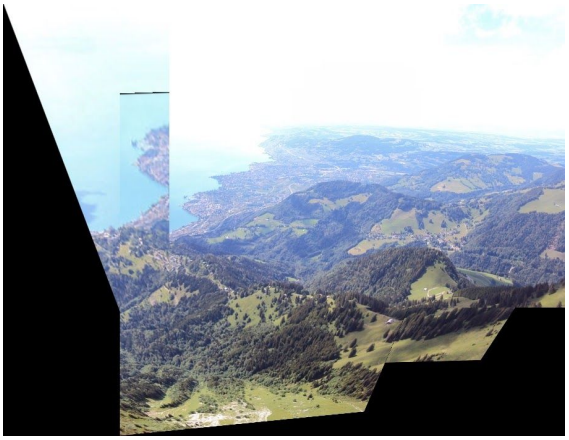
4. **Ledge**

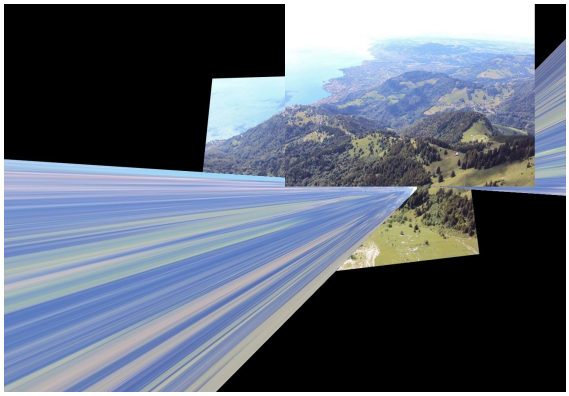
   This dataset had similar points which can give wrong wrong correspondences. We used 20 correspondences for matching. The stitching done using the first image as the reference image gave good results. However, the stitching done using the other two images as reference gave some unsatisfactory output because of correspondence points mismatch.

   Input:



Output:



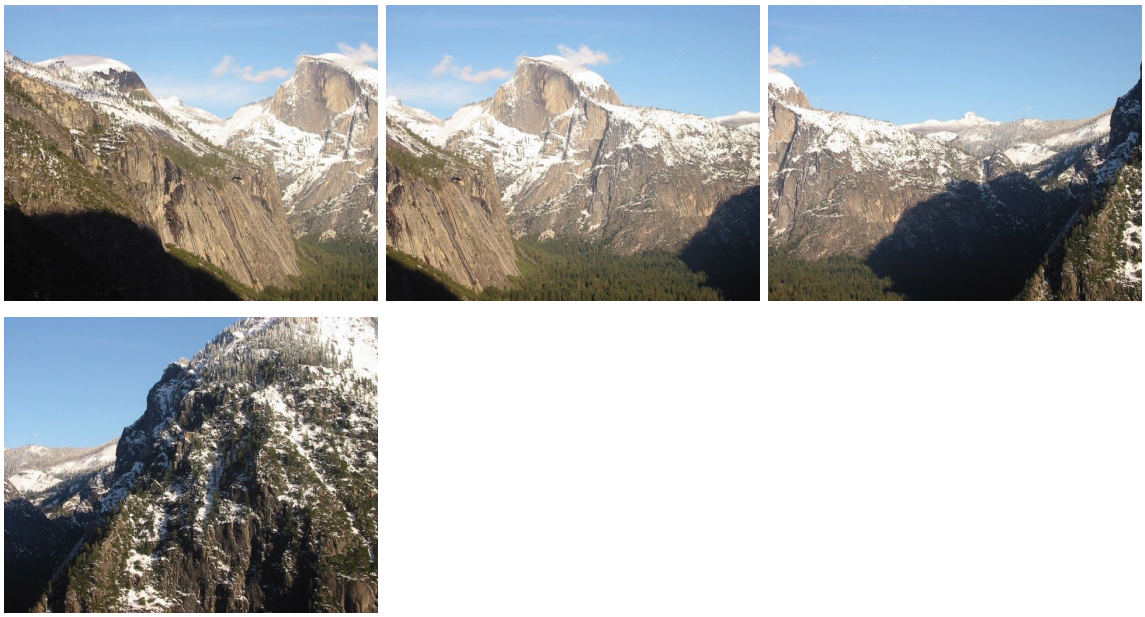


Reference image = 1,2,3 respectively

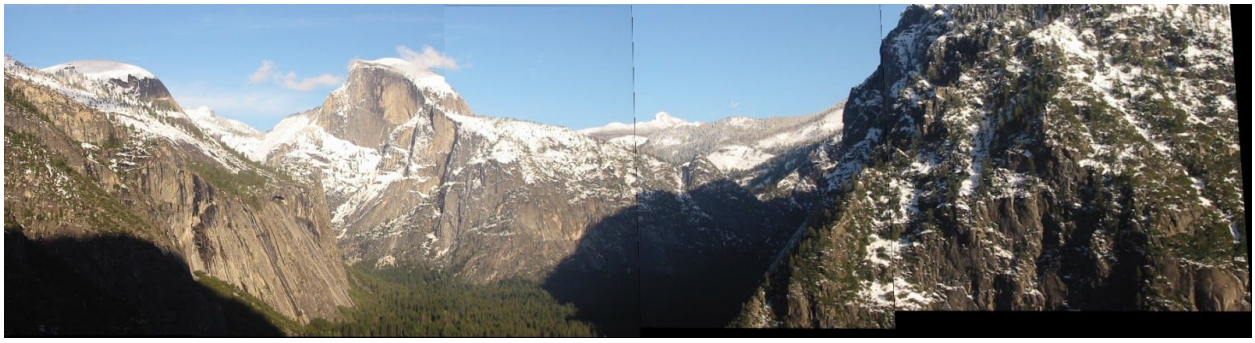
5. Yosemite

For this dataset, the images were taken with not much overlap and also there were repeating patterns that led to correspondence mismatches leading to unsatisfactory results. The output from taking the first image as the reference image was very good. We used 9 correspondences.

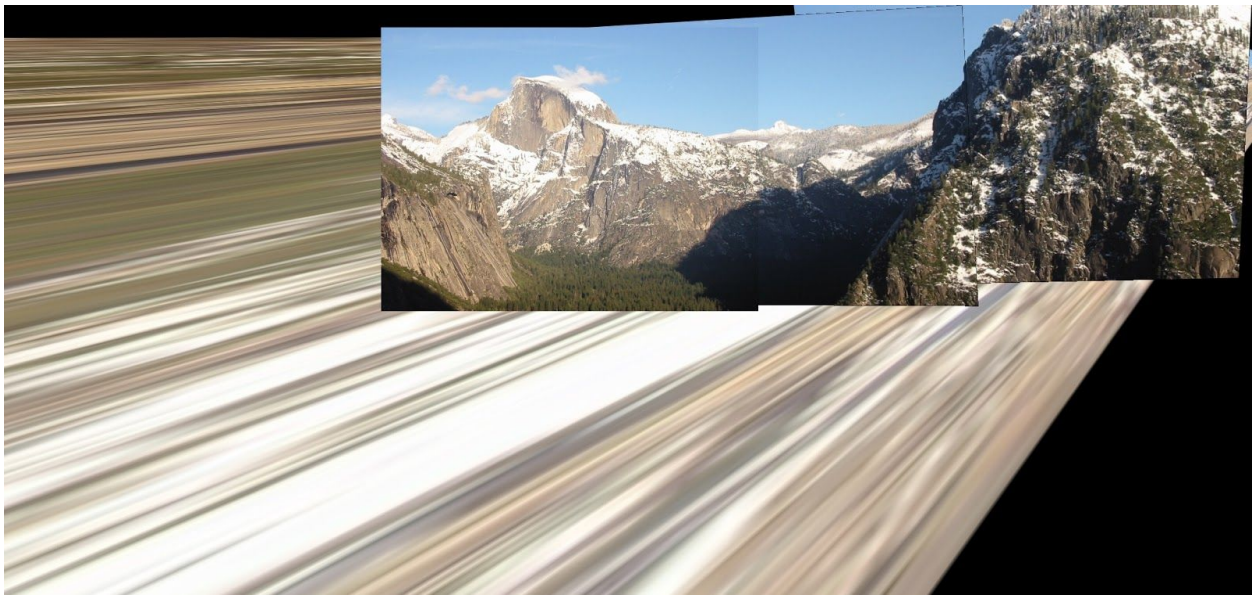
Input:



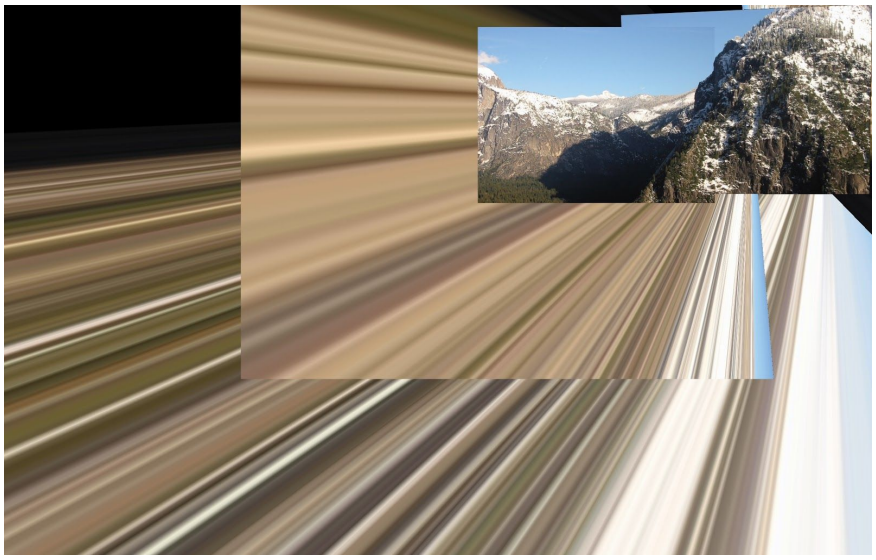
Output:



Reference image = 1



Reference image = 2



Reference image = 3

Reference image = 4

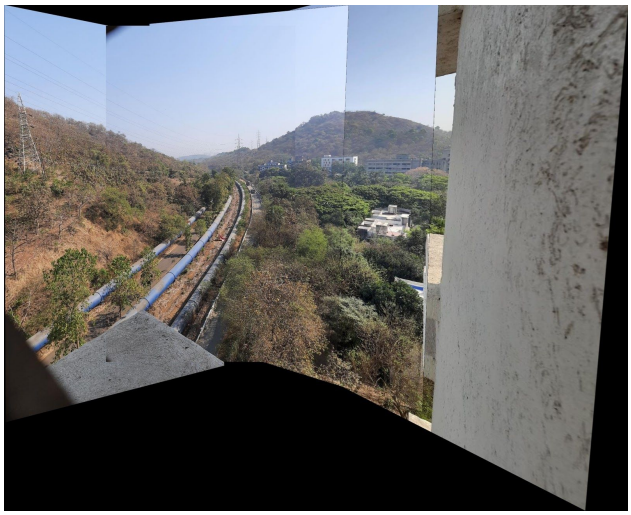
6. **Sameer (own dataset)**

   This is a self-made dataset of Sameer hills and the neighbouring area. It gave good results as the overlapping area between the images are good. We used 25 point correspondences.

   Input:

   Output:

Reference image = 1


Reference image = 2


Reference image = 3


Reference image = 4

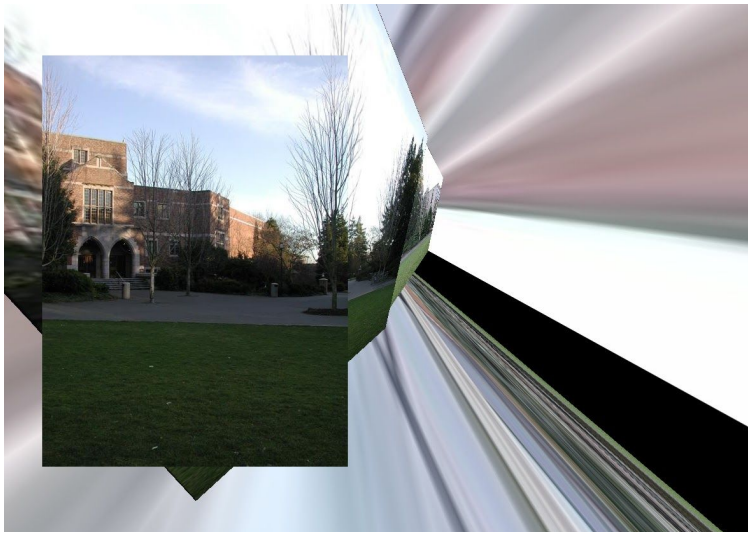

Reference image = 5

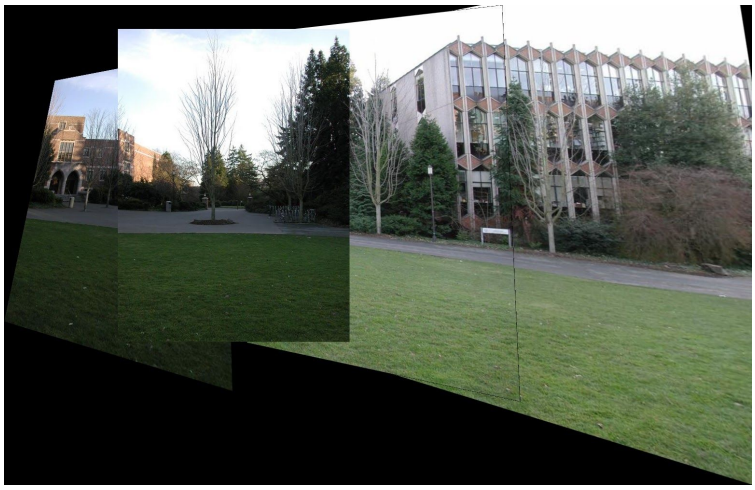
7. **Campus (own dataset)**

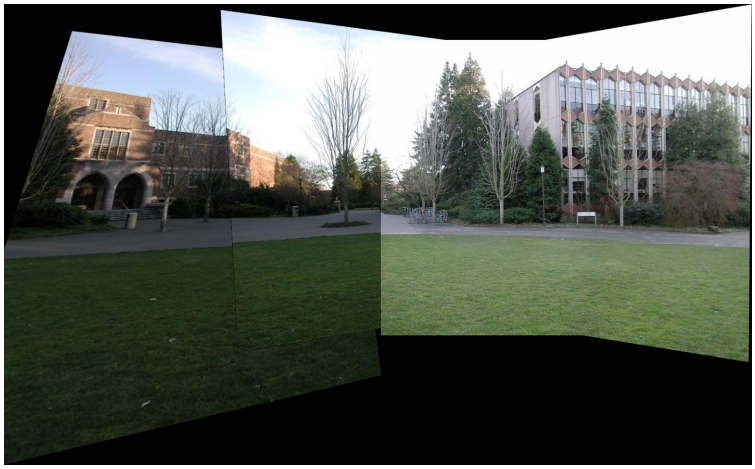
   This dataset was taken from the internet. It gave good results for 20 correspondence points. The stitching done with the fourth image as reference did not yield satisfactory results, but for remaining images as reference images the stitching was good.

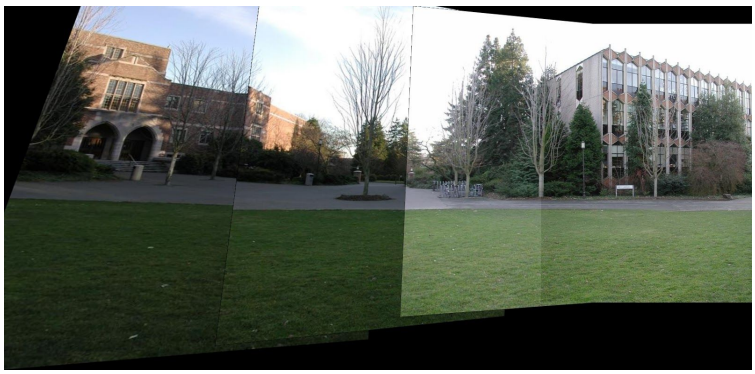
   Input:

   Output:


Reference image = 4


Reference image = 3


Reference image = 2


Reference image = 1

Reference

1. https://stackoverflow.com/questions/24563173/stitch-multiple-images-using-opencv-python#:~:text=Step%20by%20step%2C%20assuming%20you,Update%20H_2%20%3D%20H_3%20*%20H_23
2. https://www.pyimagesearch.com/2018/12/17/image-stitching-with-opencv-and-python/
3. https://medium.com/analytics-vidhya/image-stitching-with-opencv-and-python-1ebd9e0a6d78
4. https://morioh.com/p/bd1b6fc9d9eb
5. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_matcher/py_matcher.html
6. https://docs.opencv.org/3.4.2/d1/d89/tutorial_py_orb.html
7. https://docs.opencv.org/3.4.2/d2/d8d/classcv_1_1Stitcher.html
8. http://bigwww.epfl.ch/publications/thevenaz0701.pdf