B657-COMPUTER VISION

ASSIGNMENT 2

sdasara-simang-aboppana

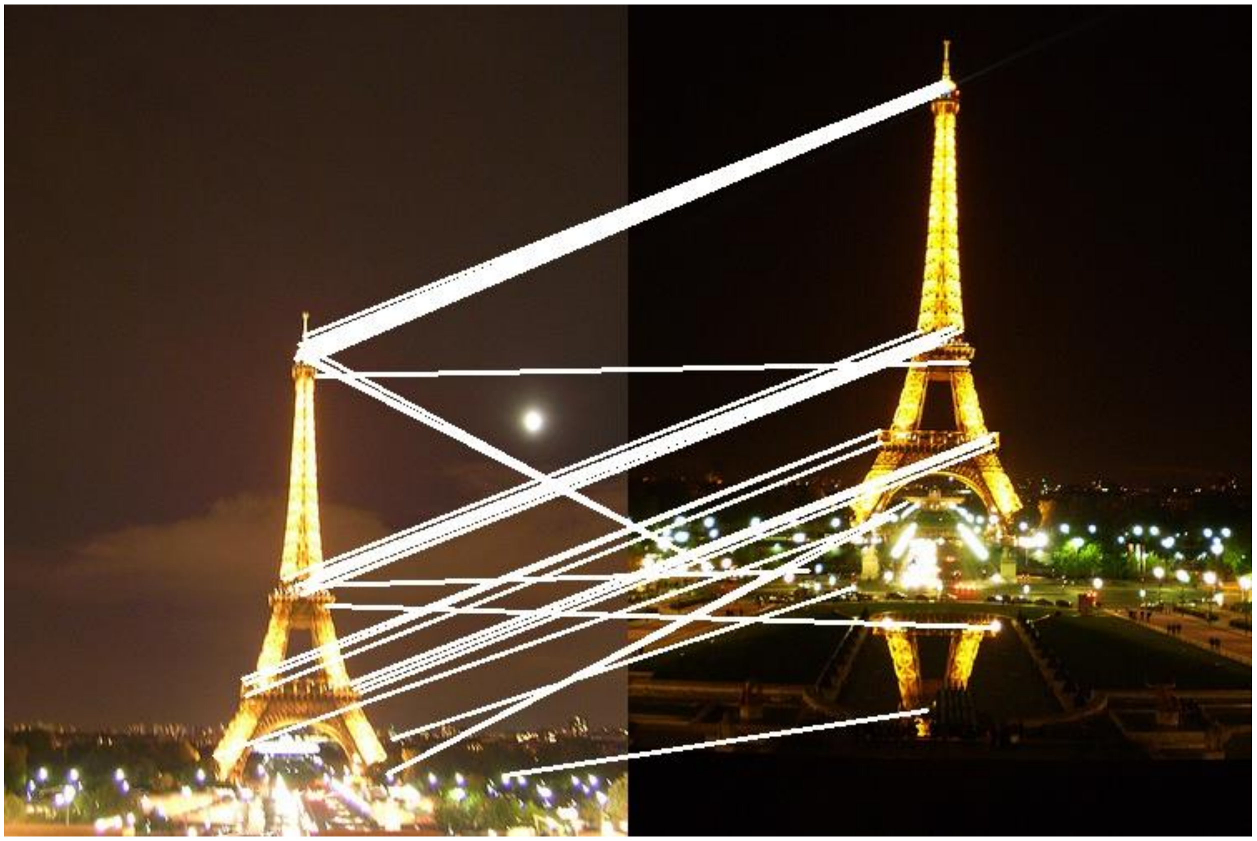
Part 1: Image matching and clustering

Design decisions, simplifications and assumptions:

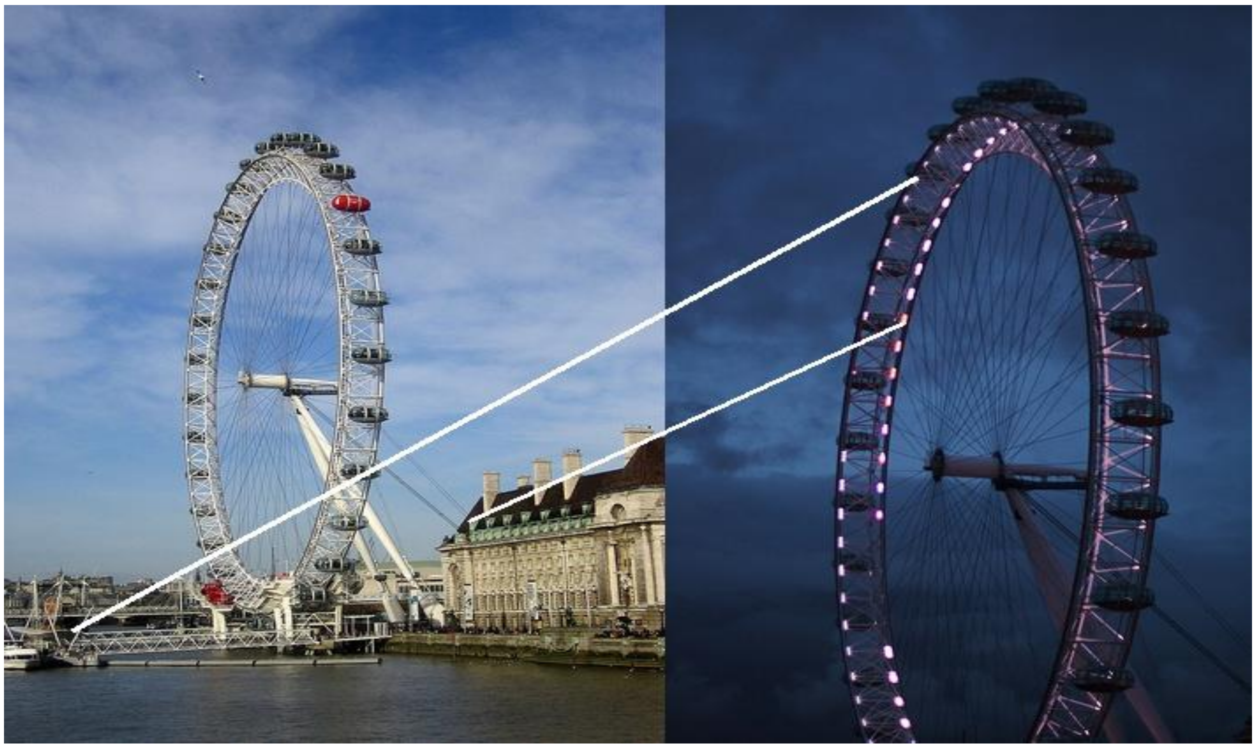
- We used OpenCV implementation of *ORB* algorithm for feature keypoint detection.
- Bitwise Hamming distance was used as the distance between the descriptor features.
- Match method was implemented that counts the number of matching descriptors for keypoints based on threshold between first closest and second closest matches.
- The chamfer distance like matching method above is not symmetric.
- *Response* attribute of the keypoint object was used to filter out the large number of keypoints in each image.
- Keypoints whose *response* attribute value was less than the mean of all *response* attribute values in an image were discarded to make computation faster.
- The threshold was set at *0.8* to get optimal matching.
- A similarity matrix is constructed for the images to be clustered. Only the bottom half of the diagonal is filled and is copied over to the top half to reduce computation time.
- The clustering algorithm implemented is a variant of K-means where we take the cluster centroid as the image with the maximum matches within its cluster.
- Clustering algorithm uses the similarity matrix and image indices to perform clustering.
- Clustering algorithm recenters the cluster centre based on the maximum number of matches with the other images in the same cluster.
- The number of iterations was set to 700.

Results and observations:

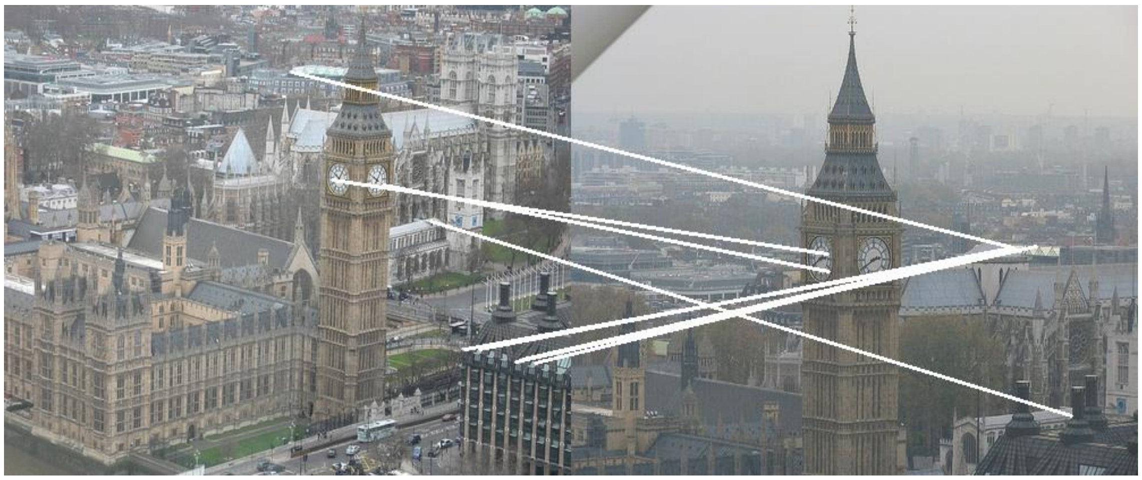
- ORB feature detection seems to work well on images that have a clear scene in the background and the images have the same hue. Eg. eiffel_18.jpg and eiffel_19.jpg



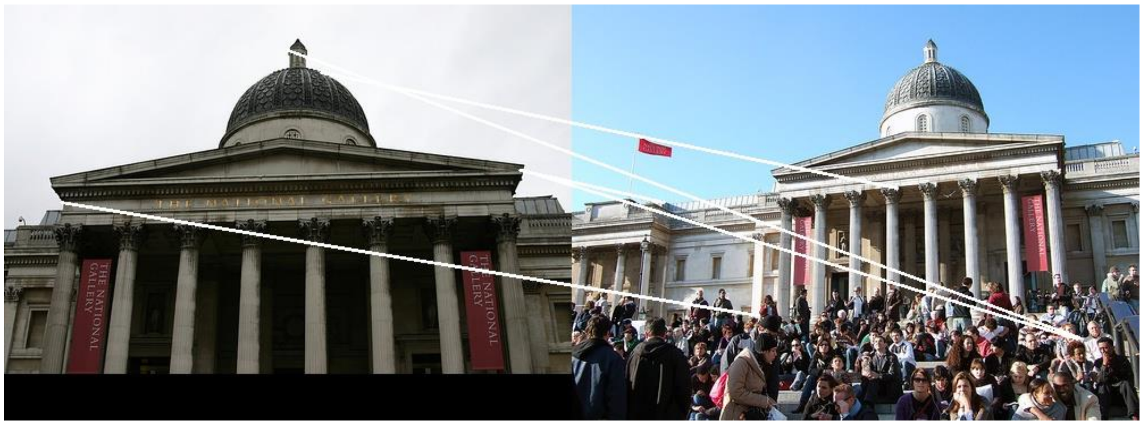
- This case for londoneye_2.jpg and londoneye_8.jpg seems to fail because of the difference in the hue of the image.



- ORB matching matches all elements of the image that seem similar even if they are not scene related.



- A major drawback of this matching method is if the descriptors of keypoints from non-scene objects match, it can disturb our matching.



- The clustering on Part1 folder of 93 images fetched the following results:
  True positive is 134
  True negative is 6508
  Accuracy is : 0.7762973352033661

The Overall computational time is around **35 minutes** for part1

Part 2: Image Transformations

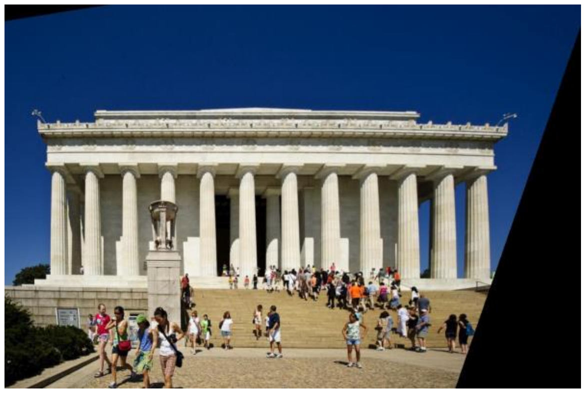
1. Transforming an image using transformation matrix

Design and specifications:

- Given a transformation matrix the transform function transforms the image coordinates to new coordinates and uses bilinear interpolation to fill in the pixel values
- Inverse warping and bilinear interpolation are used to avoid holes in the resulting image
- If the transformed coordinates are out of bounds in the original image, they are left blank

Results:

Original image



Transformed image

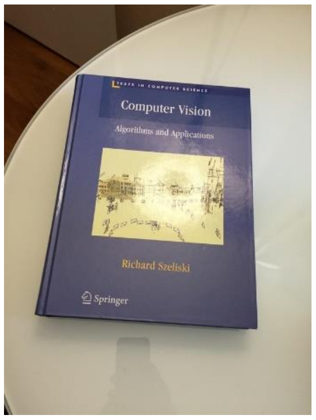
2. Finding the transformation matrix from points and applying it to input image

Design, specifications and assumptions:

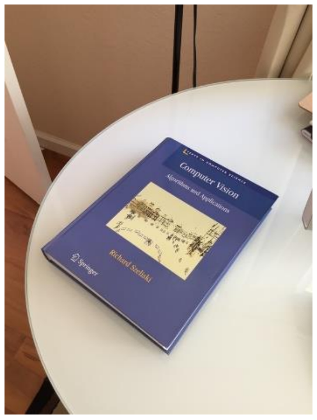
- This function uses the pair of corresponding points from images and then computes the transformation matrix
- A set of linear equations were solved based on the number of pair of points
- For translation, just a difference between the points would result in the translation vectors
- For Euclidian, affine and projective there are 4,6,8 equations to be solved
- The function only outputs a transformation matrix if the coefficients matrix of the linear equations is non-singular
- Then it takes the transformation matrix and applies it to the input image to transform it into the orientation of second image
- The resulting image is not entirely captured because the coordinates don't fit the original image size

Results:

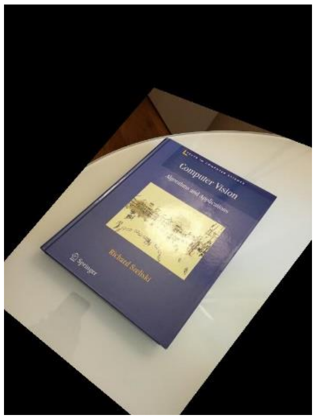
- The output image shown below is in the same orientation as the desired image



Input image



Desired orientation



Output image

Part 3: Image matching and transformations

Design, specifications and assumptions:

- The matching keypoints between two images are obtained from part 1 and they are used to find the transformation matrix
- The points are then fed into RANSAC algorithm which selects the best hypothesis with maximum support i.e. inliers. Hypothesis is the transformation matrix
- The RANSAC algorithm starts with fixed number of iterations (2500) and high inlier to total points ratio of 0.8. The Euclidian distance threshold is set to 5 pixels between transformed coordinates and original coordinates to count a point pair as an inlier. 4 points are randomly sampled and used to find the hypothesis at a time.
- Further, it checks for best hypothesis with inlier ratio over 0.8, if no hypothesis is found then the inlier ratio is reduced by 0.05 and the hypothesis search is recursive until the best hypothesis is returned
- Using the best hypothesis returned from RANSAC i.e. the transformation matrix the images are then stitched together to create a panorama
- For creating the panorama, a blank canvas of the size of both images combined horizontally is created and then the first image is stored, aligned to the leftmost axis
- The second image is then filled into the canvas using inverse warping and bilinear interpolation using the inverse of the transformation matrix
- For overlapping locations, the pixel values are averaged

Results:

- The results for four test cases we have experimented are below
- For case 1, case 2 and case 3 the results are near perfect with the images aligning accurately
- For case 4 the images are aligning correctly but the texture information is lost as we are using bilinear interpolation while stitching the images. The image contains dense texture of sandy mountain. So, this adds a little blur or distortion in the output

Case 1

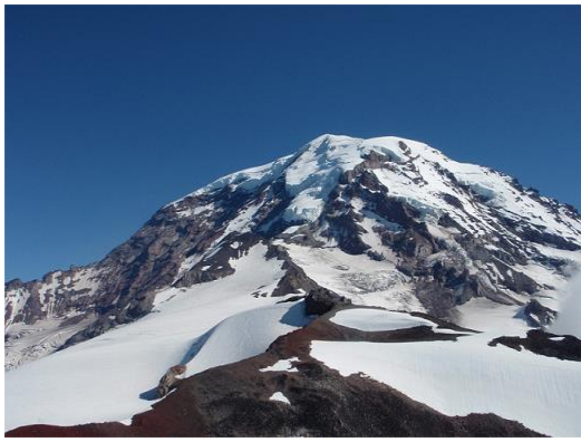


Image 1

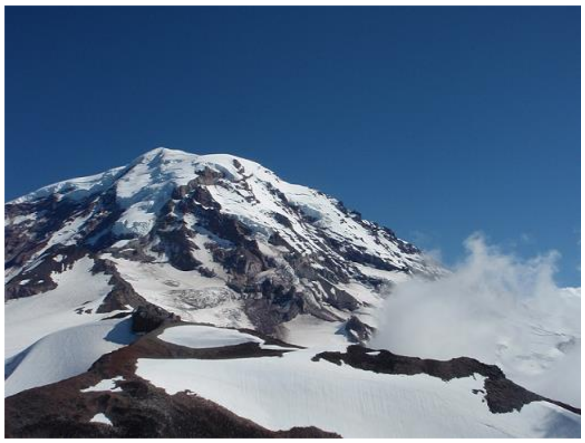
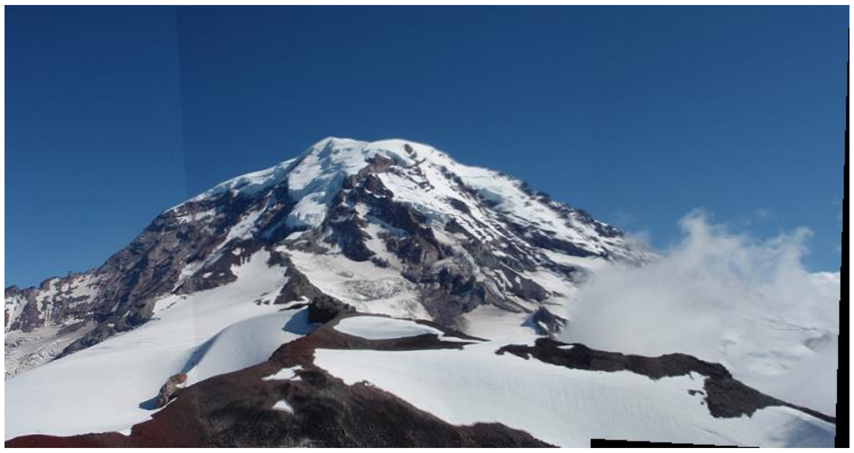


Image 2


Panorama image

Case 2

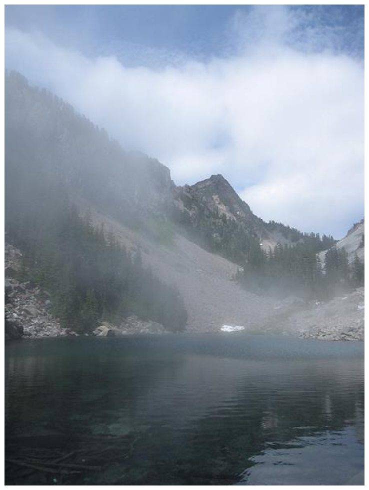


Image 1

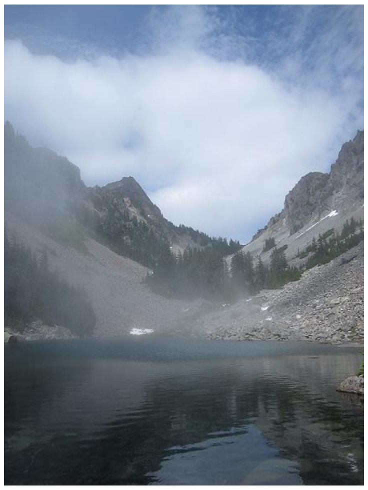
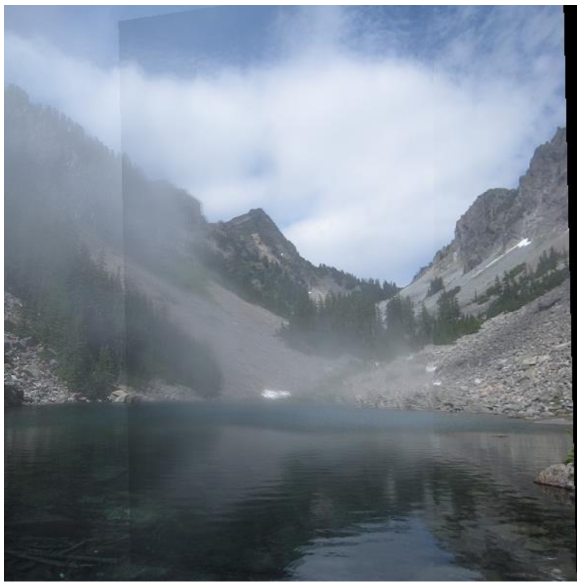


Image 2


Panorama image

Case 3

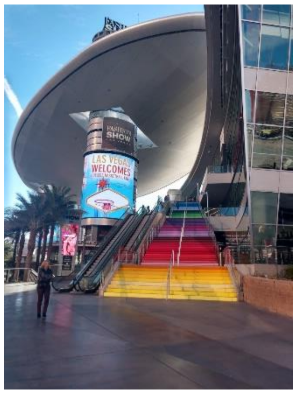


Image 1

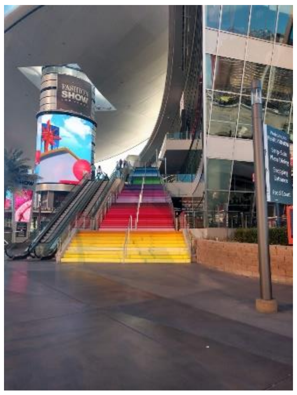
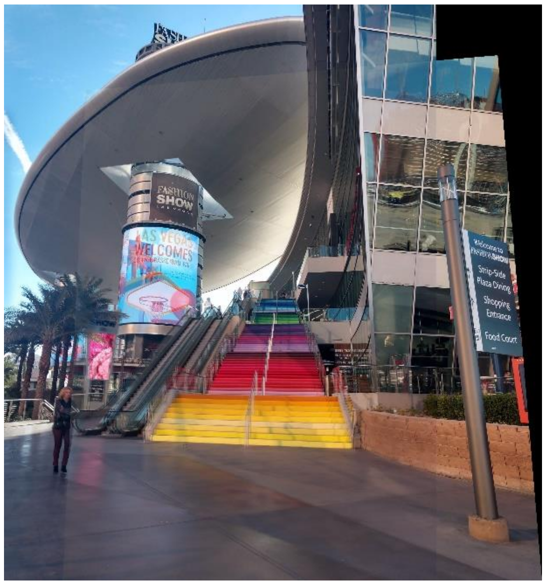


Image 2



Panorama image

Case 4

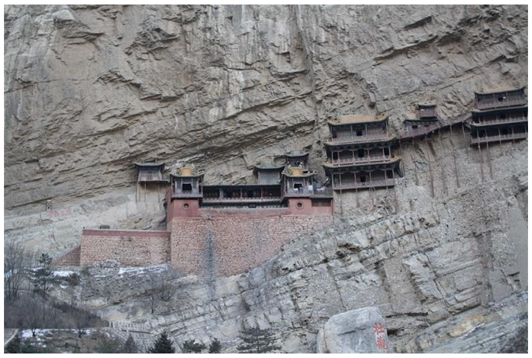


Image 1

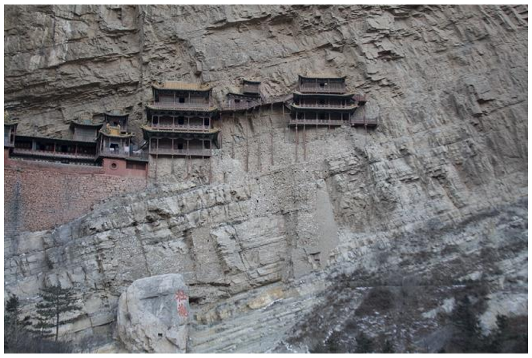
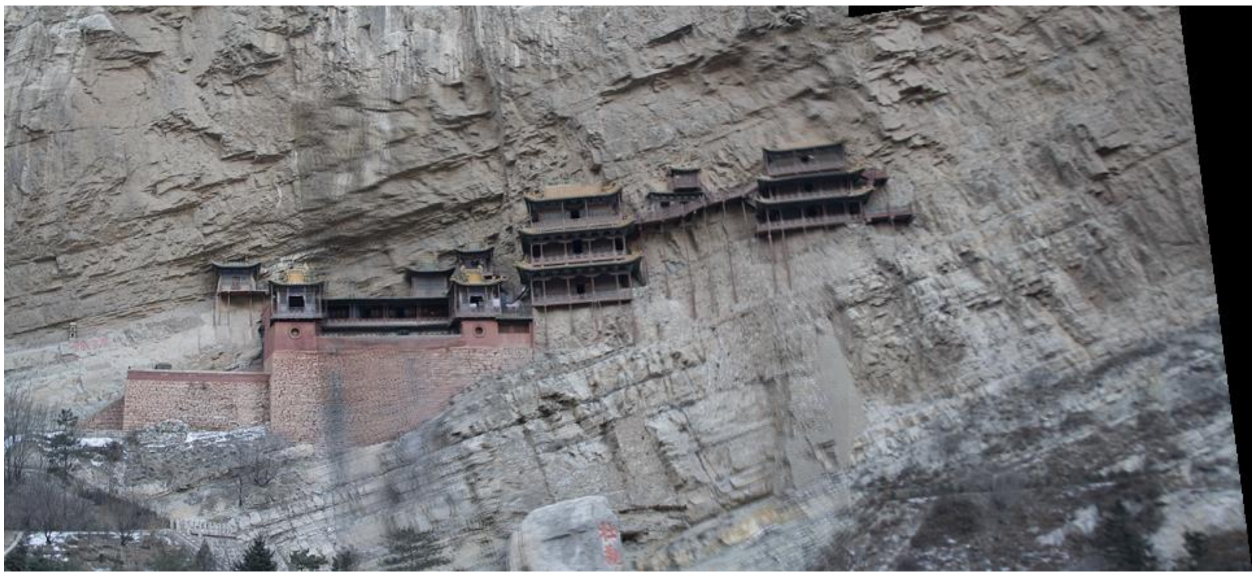


Image 2



Panorama image