Report(Assignment-2)
(Entry No:2020CSY7576)

Image Mosaicing using Graph Cut Algorithm

Introduction:-
Panorama stitching is performed by properly aligning images using their homography matrix and then performing appropriate blending in the regions the two images overlap. With normal blending techniques like alpha blending, moving images may produce ghosting in the final image. This kind of ghosting may occur because such blending techniques consider the contribution of both images at each pixel location. However if we restrict portions of the final image to come from only one of the images, we'll produce better output. This can be done using graph cut algorithms which we've used below.

Methodology used:-

1)Keypoint and Descriptor matching:- Matching keypoints and descriptors using SIFT and extracting the top matches.

2)Estimating Homography Matrix:- Now that we have list of keypoints related by a homography in the two images. We run RANSAC algorithm and estimate the homography matrix, H. Suppose H aligns image2(right image) to the plane of image1(left image). Now, we've image1 and image2 aligned for stitching. (Note that we've always aligned right image to left image plane and not vice versa in our method)

3)Graph Construction:-

a)Vertex set construction:-

We extract the overlapping portion of the two images. Now, we create a graph G with number of vertices equal to the number of pixels in the overlapping region. We add two extra nodes, one is the source vertex and one is the destination vertex.

b)Edge set construction:-

The vertices in the overlapping regions whose pixels in each row are in leftmost side, are attached to the source vertex with infinite weight(or very large value) such that those pixels are constrained to come from the left image, i.e., there is a directed edge from source to all the leftmost pixels with infinite weight. The vertices in the overlapping regions whose pixels in each row are in rightmost side, are attached to the destination vertex with infinite weight(or very large value) such that those pixels are constrained to come from the right image,i.e., there is a directed edge from destination to all the rightmost pixels with infinite weight. For each vertex ‘v’ corresponding to the overlapping region pixels. There is a weighted edge from ‘v’ to its four neighbouring pixels, up, down, right, left.(if any of the pixel in up,down,right or left is not present then that edge is not created.) If A and B are the left and right images, and ‘s’ and ‘t’ be two pixel locations in the overlapping regions, then weight of a directed edge from ‘s’ to ‘t’ or from ‘t’ to ‘s’ is given by,

$$\mathrm{M(s,t,A,B)} = \frac{||A(s)-B(s)||+||A(t)-B(t)||}{2\times||A(s)-A(t)||+2\times||B(s)-B(t)||}$$

||.|| denotes a L2 norm. The numerator takes account of the fact, that the cut always pass through those edges, where the similarity between the left and right image for the corresponding edge is highest. The denominator takes care of the fact that the cut is always performed in low frequency regions than high frequency regions (as was mentioned in research paper referred for assignment), that seams often pass through low frequency regions than higher ones.

5)Applying Graph Cut:-Now that we’ve our graph, we calculate maximum flow on the graph with the Boykov-Kolmogorov algorithm which gives us the residual for each edge which is then passed into graph_tool.flow.min_st_cut() function which returns a vertex property map “partition”, which contains boolean values with true value indicating that the vertex(aka pixel) comes from left image and false value from the right images. Now, we can extract what comes from the left image and what comes from the right image which gives us an optimal seam.

6)Pyramid Blending:- This is performed using 4 levels of gaussian and laplacian pyramid between the part of left and right images that are being merged at the seam.

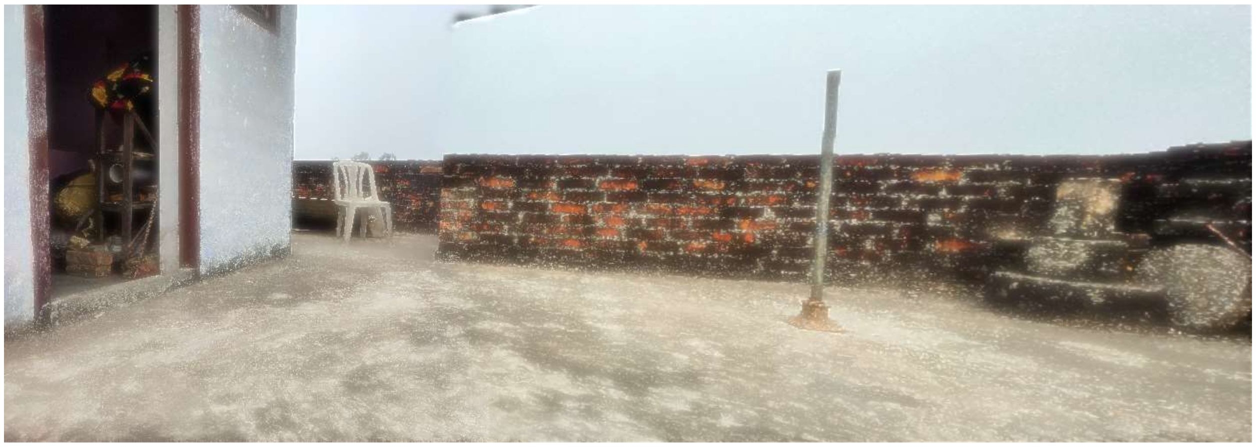
Output:-

The output for the in sample data provided is given below:

RESTAURANT

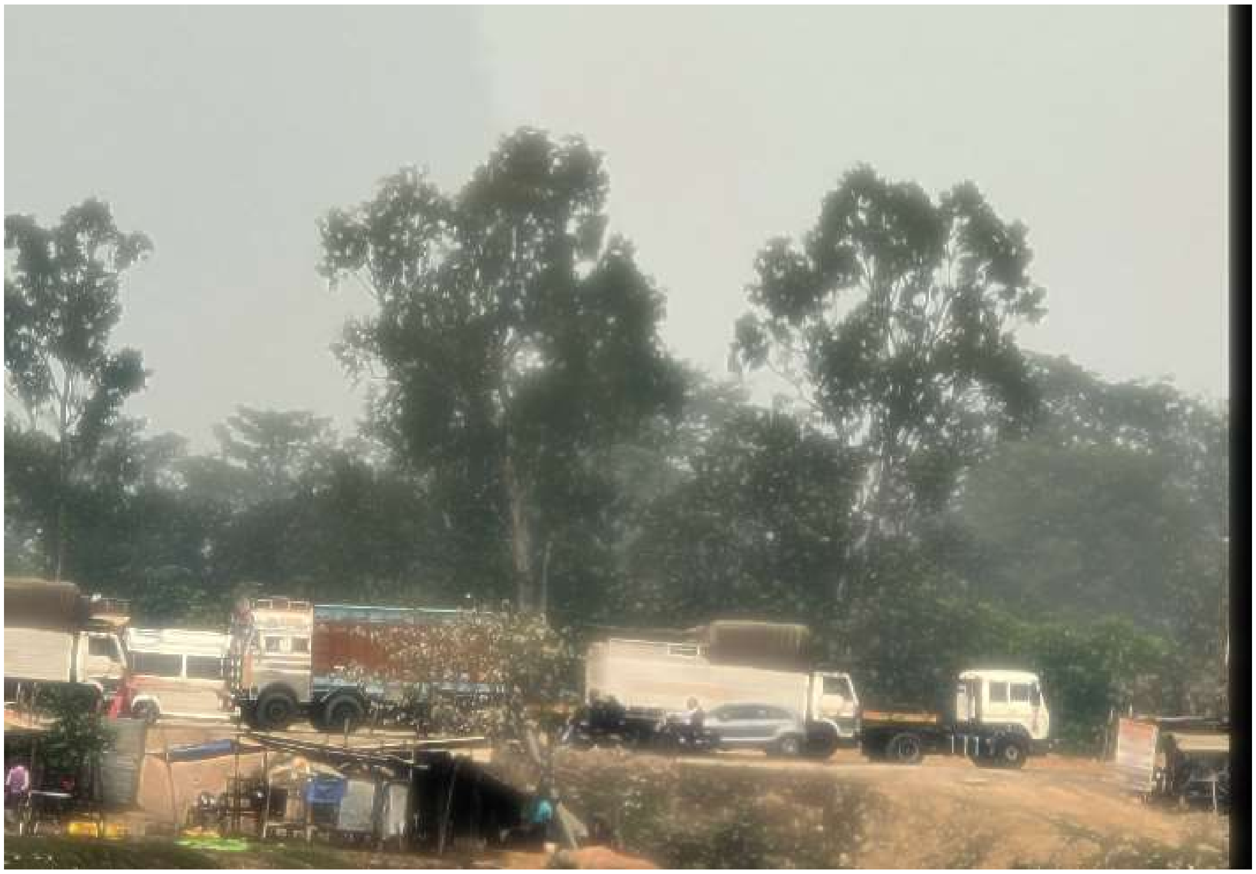
The following are the outputs of the images collected by me, alongwith reasons of failure of certain cases:

1)

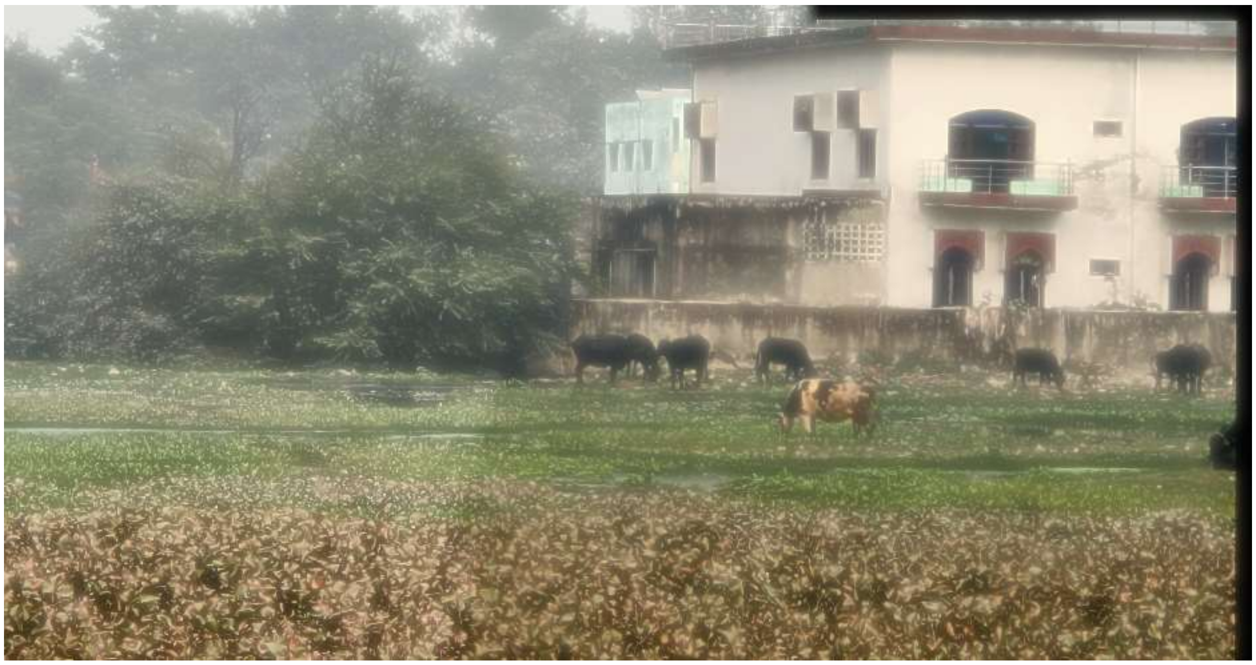


Notice the dark obvious part of seam in the top part of the image, this might be possible because it must have been the border part of the right image after it was oriented in the left image frame, which brings a sudden change in large intensity difference.

2)



3)



Notice the obvious intensity difference between left and right part of green fields, the pyramid blending couldn’t hide the seam completely. To reduce the intensity difference, the levels in pyramid blending has to be increased but then the quality of the image becomes worse.

4)

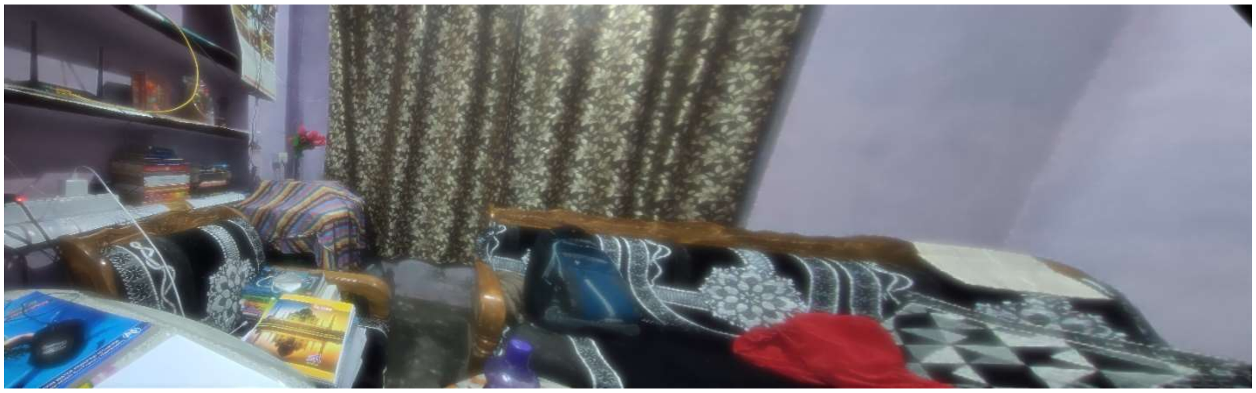
5)

6)

7)

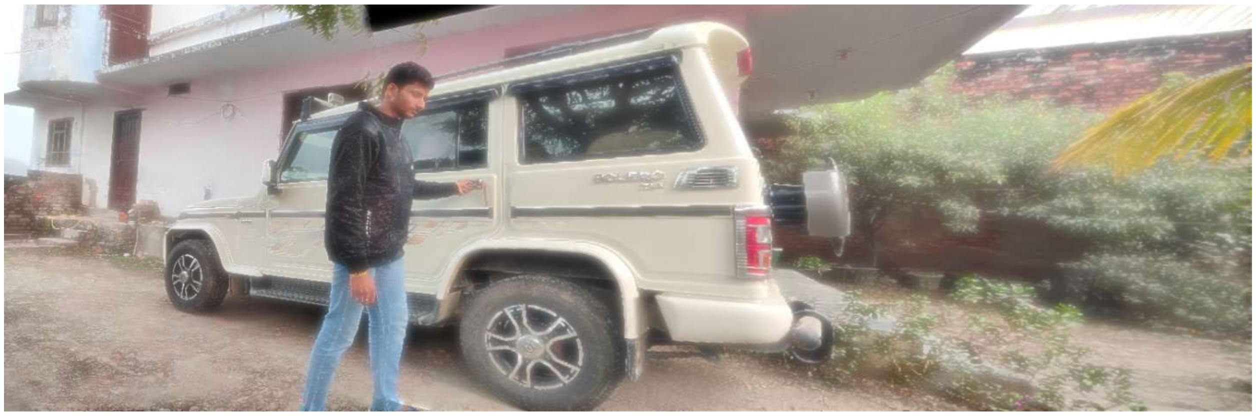
The algo failed here miserably, because it found the wrong homography alignment. This might be possible because large matching keypoints must have been present in the greyish color region(the cemented large portion of the input images), where proper orientation could not be found out as most of the portion is identical be it present at any portion of the greyish part.

8)

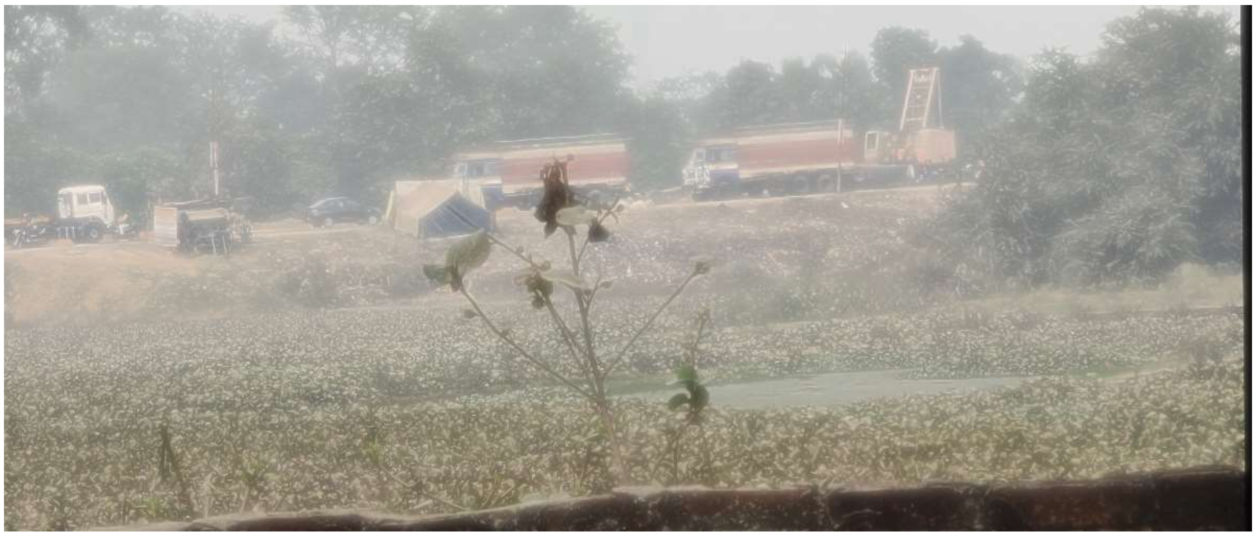


The panoroma in this case is not good, the reason can be that the images must not have been able to match the assumption of rotating camera frame, which is essential to estimate the homography and alignment.

9)



10)



The algo has failed here in the sense that it included two images of the same truck, whereas there is only one truck in the actual scene. This happened due to the fast pace of the truck, so that after capturing image 1, the truck moved forward with such a distance as to not have any overlapping region with the previous truck after homography alignment, and leaving a green region between the position of truck in image1 and that of in image 2 which largely differentiated them as two trucks. Hence, one truck displayed as two trucks.

Note that each image output may take upto 1 min to produce(depends on processor too, but just an estimation). This is the case, for we couldn't not resize the image too much and lose essential information. Please rename and convert the image pairs appropriately to “img1.png” and “img2.png” before running.