CMPE 465 ASSIGNMENT 3

Berke Evrensevdi

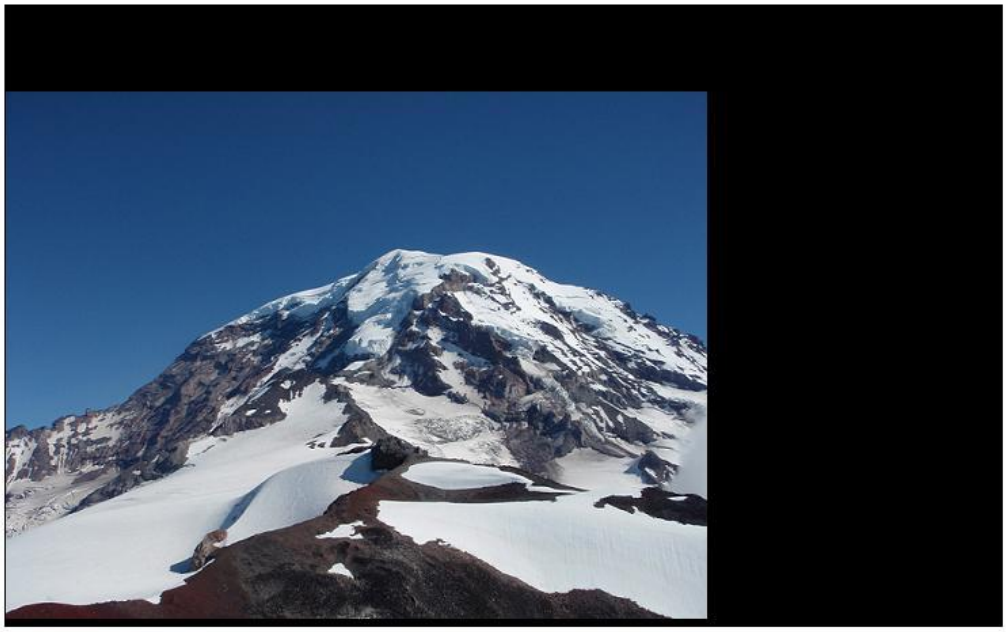
10102705512

PART 1

In part 1, initially I take the project of four corners of image2 onto image1 using homInv. After that I determine top, left, right, bottom. While determining top, I take the minimum y of topLeft or topRight. If minimum y of topLeft or topRight is negative then negative value will be equal to top. If minimum x is positive, then top will be equal to 1. While determining left, I take the minimum x of topLeft or bottomLeft. If minimum x of topLeft or bottomLeft is negative then negative value will be equal to left. If minimum x is positive, then left will be equal to 1. While determining bottom, I take the maximum y of bottomLeft or bottomRight. If one of the bottomLeft or bottomRight value is greater than height of image1, then bottom will be equal to that value. If height of image 1 is greater than the maximum y of bottomRight or bottomLeft, then bottom will be equal to height of image1. While determining right, I take the maximum x of topRight or bottomRight. If one of the bottomRight or topRight value is greater than width of image1, then right will be equal to that value. If width of image1 is greater than the maximum x of bottomRight or topRight, then right will be equal to width of image1. Afterwards, the background image's width will be equal to right-left and its height will be equal to bottom-top. The next step is to put image1 at the right location of background image called 'panorama'. This is done with adding abs(top) and abs(left) to panorama dimensions which is presented below:

```
panorama(y + abs(top), x + abs(left),z) = image1(y,x,z);
```

If top has negative value, we put image1 pixels to panorama shifted as much as abs(top) and the same goes for abs(left). After image1 put onto right location the results were like presented below:



Finally, I try to put image2 onto image1 with right angle according to image1. While doing it, I iterate top through bottom which corresponds y values and left through right which corresponds x values. Then I project each (x,y) onto image2 using hom. If projected point (xp,yp) lies within image2's boundaries, and point (x + abs(left), y+ abs(top)) corresponds to [0,0,0] which is black pixel, then new pixel value of image2(abs(yp), abs(xp)) will be put for each channels of RGB instead of [0,0,0]. The results are as presented below:

PART 2

In part 2, I implement two functions called meanShift and segmentMS. meanShift function takes parameters of data, ind, bandwidth, and stopT and returns peaks. Data is like two-dimensional rgb matrix which its first column holds red channel of each pixel, its second column holds green and the third holds blue. ind means row index which corresponds 3-d feature vector comprised of R,G and B channels. So meanShift function, takes a feature vector from 'data' with specific index (ind) as an initial centroid. Then, distance between initial centroid and all other feature vectors in data matrix is computed. If the distance is less than bandwidth, then it means that the feature vector belongs to our cluster. When we find entire cluster members, we take the mean of these members and our new centroid will be this mean of cluster members. Afterwards, again we compute the distance between new centroid and all other feature vectors in data matrix. If the distance is again less than bandwidth then we take this feature vector as a cluster member. When we gather entire members, again we take mean of the entire members and take this mean of members as a new centroid. If this newCentroid is less than the threshold (stopT), no more shift is required and this newCentroid is peak and meanShift function returns this newCentroid.

segmentMS function takes an image, a bandwidth and a threshold as a parameter and returns labels and peaks. Firstly, we need to turn the 3-d image (axbx3) into 2-d data matrix ((a*b)x3) which each row holds R, G, B values and each row corresponds to 3-d feature vector. We send each feature vector to meanShift function in order to get 'peak' which is again feature vector. We have 'peaks' array which will return from the function. So we add 'peak' to 'peaks' array if this 'peak' has distance of bandwidth/2 at least to another peaks we obtained earlier. If distance is less than bandwidth/2, then the 'peak' is discarded and counted as already existing 'peak' in 'peaks' array. Finally, we construct the 'labels' just to fill this 'labels' with our feature vectors which belong to 'peaks' array. We iterate through 'peaks' array and for each 'peak', if this 'peak' has distance of 'bandwidth' at most to a feature vector that belongs to 'data matrix' , then the corresponding row of data matrix is changed to 'peak'.

Bandwidth = 100 stopT = 20

Bandwidth = 80 stopT = 20

Bandwidth = 50 stopT = 20

Bandwidth = 30 stopT = 20

Bandwidth = 45 stopT = 30

Bandwidth = 45 stopT = 15

Bandwidth = 45 stopT = 5

When we keep stopT stable at 20, and bandwidth changes in values of 100, 80, 50, and 30. We can easily observe that when we have high bandwidth value such as 100, there will be less cluster according to other bandwidths (80, 50, 30). For instance, when bandwidth is 100, there are 5 peaks or clusters which leads to less pixel diversity of segmentation image.

When we keep bandwidth stable at 45, and change stopT values corresponds to 30, 15, and 5. While stopT decreases, pixel diversity also decreases. Also when stopT is low, algorithm for finding peak will make more shift. So lower stopT leads to higher time complexity.