Computer Vision
Exercise 5

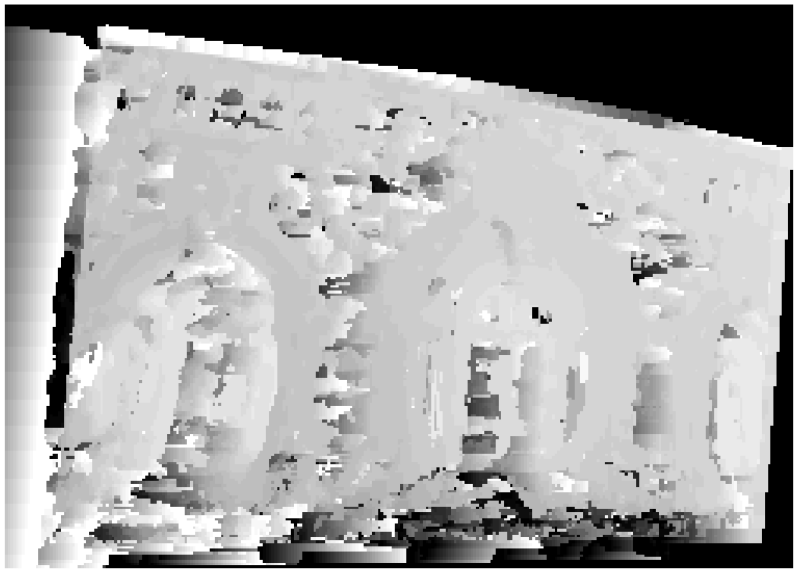
Amrollah Seifoddini

First, I resized the images because my laptop configuration does not allow this much of visual processing. Then I rectified the resized images and used the rectified ones for all the following tasks.

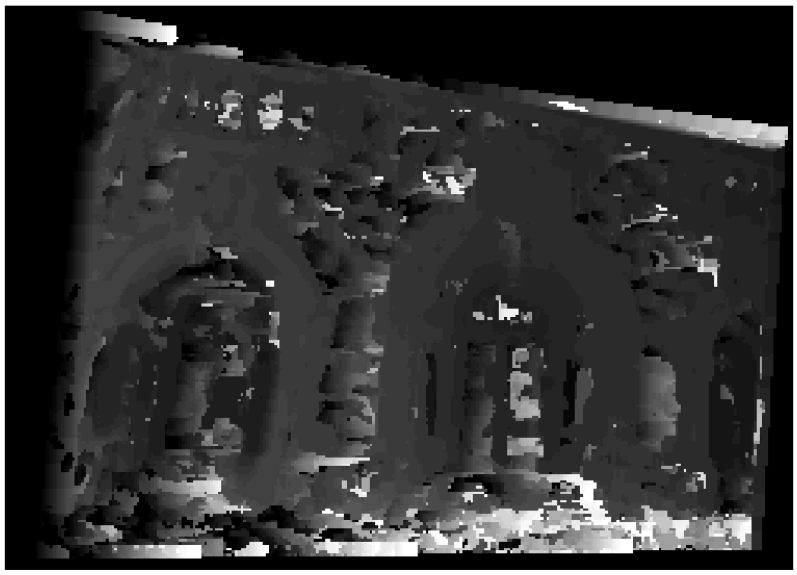
Part 1)

In this part, I used a simple “for” loop over all possible disparity values. In each loop iteration, I shift the img1 by the respective disparity value, and then subtract the result image from the img2 to compute the SSD. Then I convolve it with an average filter generated by fspecial function to smooth the local patches. Outside of the loop, I find the minimize value for each pixel in the image. I repeated this procedure for different window sizes (including 3, 5, 7). The results show that disparity maps get smoother while window size grows. For example, here I demonstrated a 7*7 window size map (first one) versus a 5*5 window size map (second one). More result images are available in the “results” folder beside codes.

7*7 window size Left map result (WTA)



5*5 window size Left map result (WTA



5*5 window size Right map result (WTA)

Part 2)

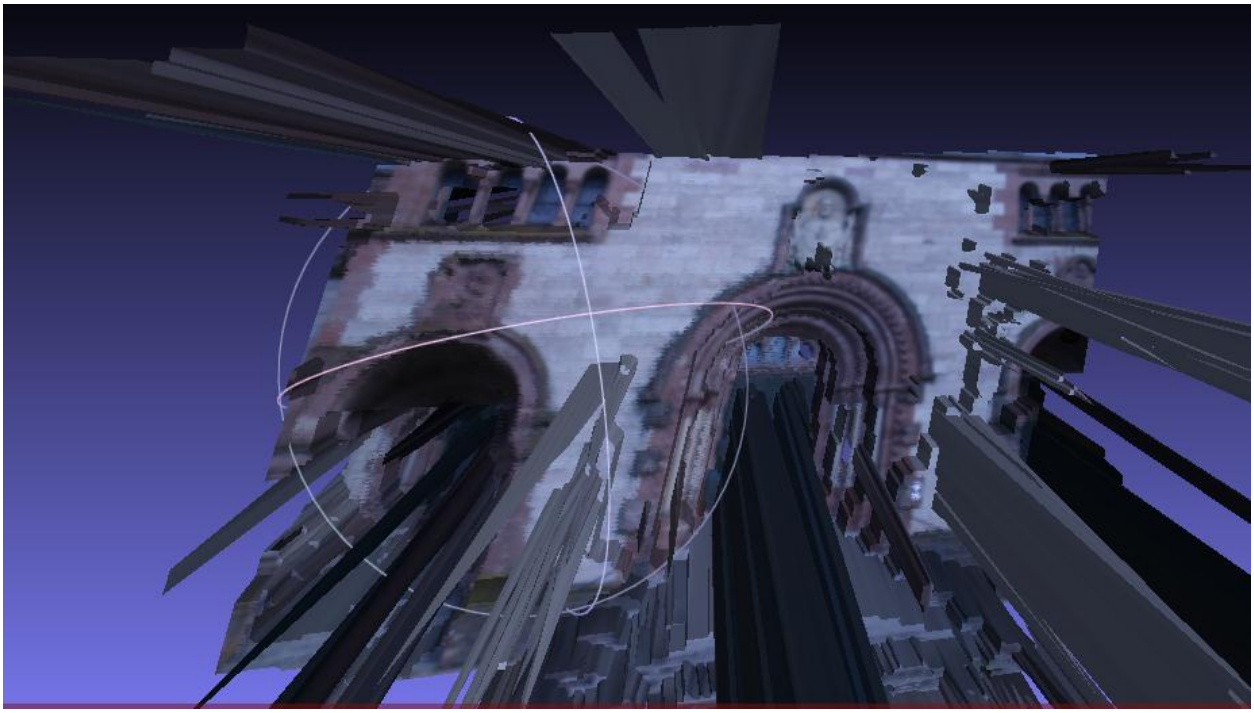
In this part, I just put my code for the first part in context of a file named diffGC.m. The rest of the code remains unchanged. I played somehow with the size of Gaussian filter in gcDisparity.m but couldn't see much improvement. So, I just used the default parameters. The part is also repeated with different window sizes and the results confirm the smoothness effect of growing window size. But the effect is less considerable as opposed to part 1 because here, the GraphCut algorithm enforces smoothness anyway. Here is a graphcut disparity map for window size 7. We can see that nearby pixels merged to each other and get the dominant disparity in that neighborhood. Thus, the edges are visible very good and completely distinct with the wall surface, although we lost some fine details. It is obvious that we want a balance between smoothness and fine details of pixel varieties in disparity.

7*7 window size Left map result (GraphCut)

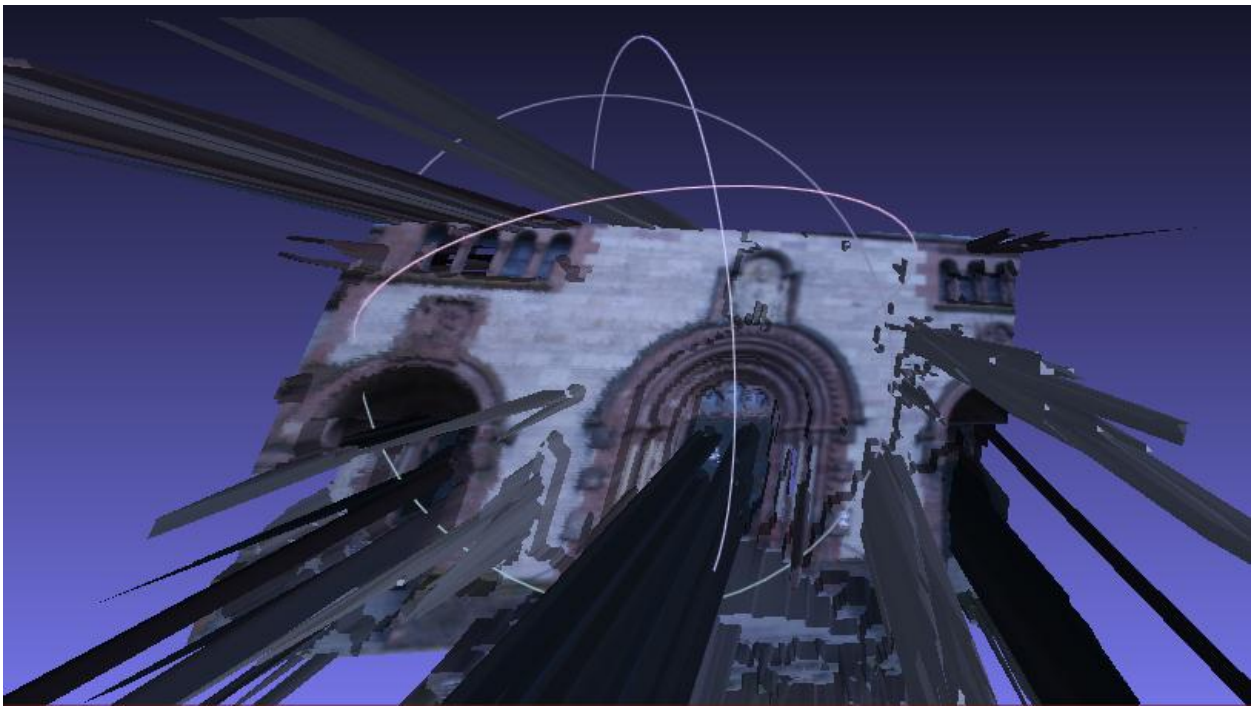
7*7 window size Right map result (GraphCut)

Part 3)

In this part, I edited the generatePointCloudFromDisps file and used linTriang function to get the 3d coordinates of points. For each image pixel, I passed the (x,y) and (x+disp(x,y),y) as corresponding 2d points to LinTriang function. Of course, the camera matrixes are also required for this function. After getting the 3d coordinates, I multiply scale matrix in them to get real 3d values for creating the 3d object. Then, I used the generateObjFile function to create .obj and .mlt file for 3d model of our stereo. I tried to open the obj file with Meshlab software but every time, it crashed while importing the file. This is probably because of my retarded graphic cart which is onboard and only has 256 MB shared memory. I asked my friend to open that file for me. Here are the result of my **old** 3d model. I still didn't have a chance to ask my friend to open my newest obj files. I also attached my newest obj files, so you can open it and have a look at it.

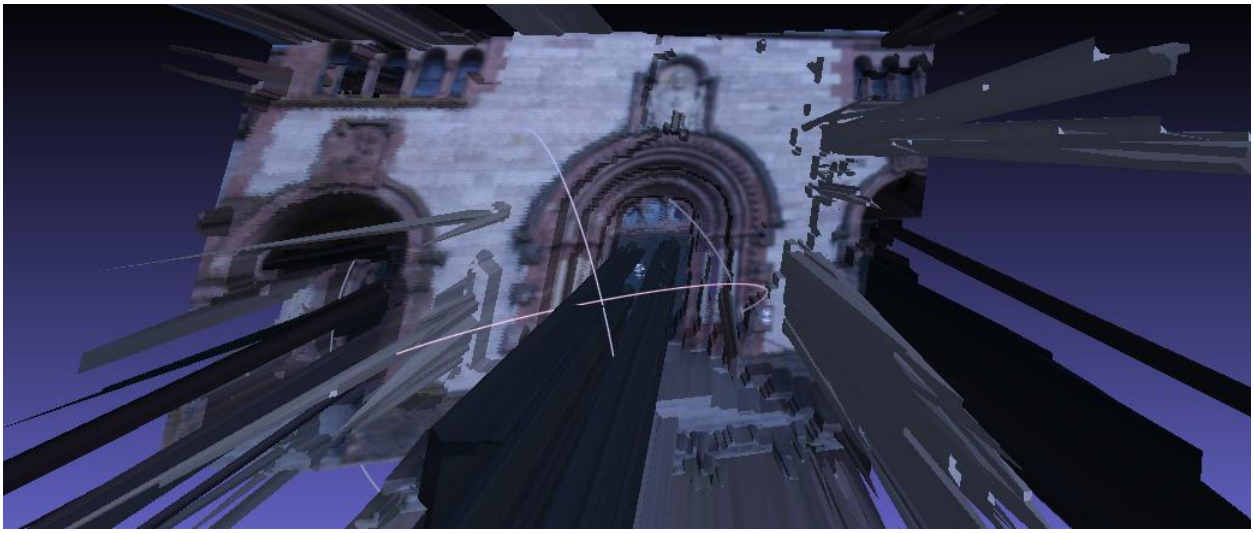


Left view of Winner Takes All algorithm.



Right view of Winner Takes All algorithm.

Left view of GraphCut algorithm.



Right view of GraphCut algorithm.

Part 4) Bonus:

For this part, Firstly, I extract sift features from both images. Then I find a matching between two image features. The SIFT features matching are displayed below (including inliers and outliers). After that, I use the **ransac5pE** algorithm from last exercise to refine our matching and get rid of outliers. After finalizing the list of inliers, I get difference of x-coordinates of these keypoints in the images. At the end, I can get an estimate of min and max disparity values by getting min and max over all these differences. However,

because of randomness nature of ransac5p algorithm, the result range varies a lot. Therefore, I decided to limit the RANSAC threshold to 0.8 (to decrease the number of inliers) and apply upper bound and lower bound on the returned disparity range.