

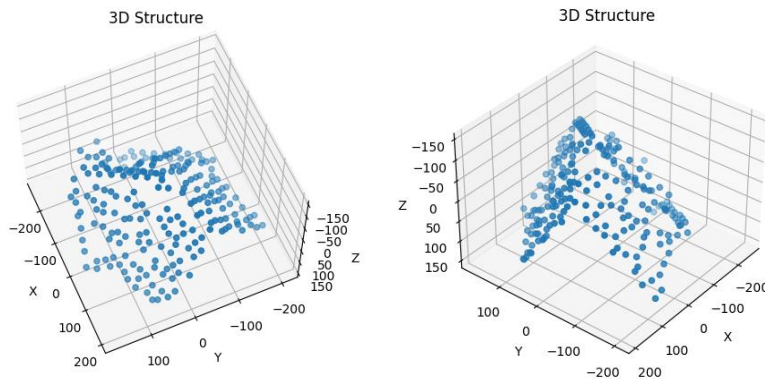
CS543/ECE549 Assignment 5

Name: Yiyang Liu

NetId: yiyang34

Part 1: Affine factorization

A: Display the 3D structure (you may want to include snapshots from several viewpoints to show the structure clearly). Report the Q matrix you found to eliminate the affine ambiguity. Discuss whether or not the reconstruction has an ambiguity.

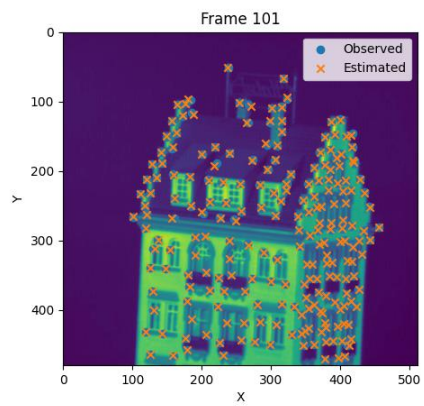
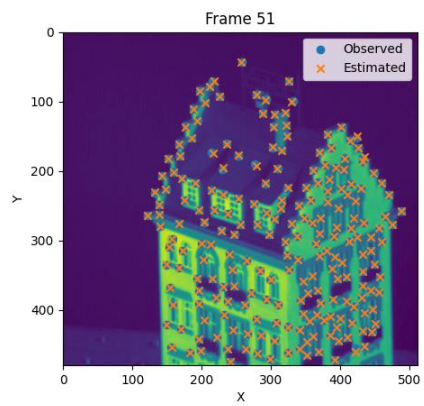
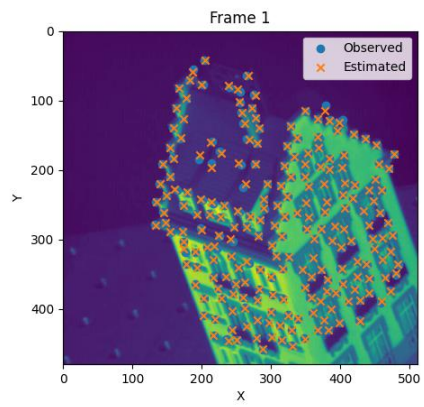


Recovered Q:

```
[[0.07967536  0.          0.          ]
 [0.00075718  0.08491909  0.          ]
 [0.00261378  0.00474187  0.03999408  ]]
```

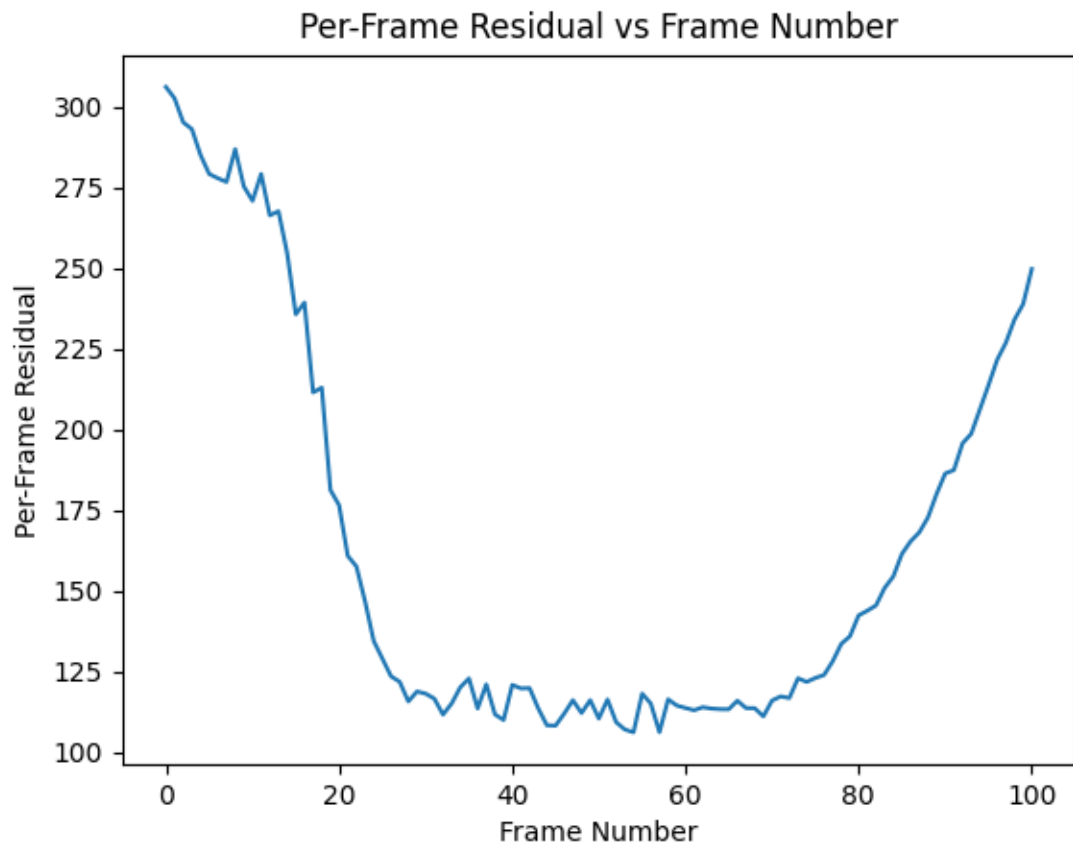
The hotel structure reconstruction is free from affine ambiguity, as orthographic projection precludes the occurrence of non-orthogonal axes.

B: Display three frames with both the observed feature points and the estimated projected 3D points overlaid.



C: Report your total residual (sum of squared Euclidean distances, in pixels, between the observed and the reprojected features) over all the frames, and plot the per-frame residual as a function of the frame number.

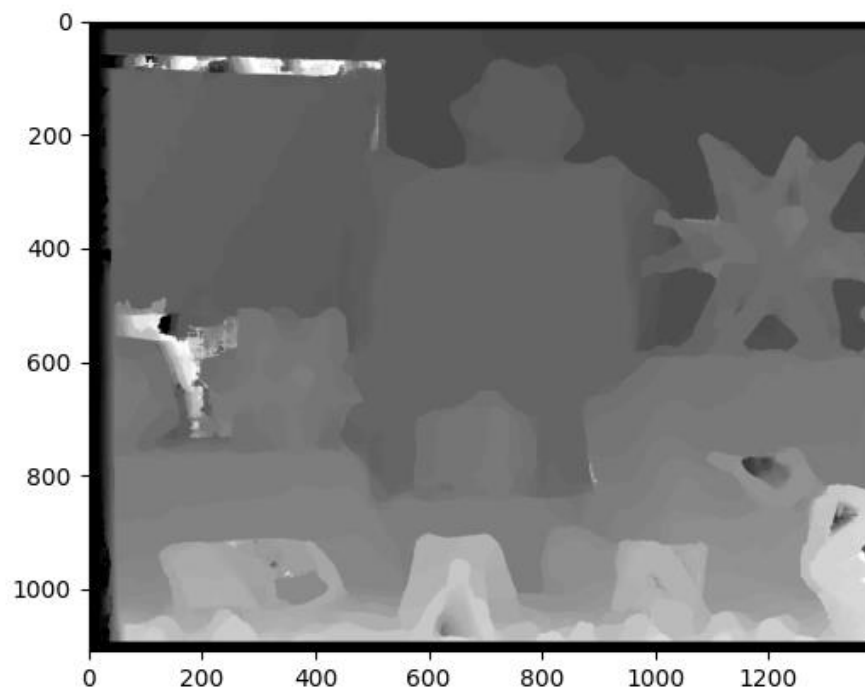
Total Residual: 16428.332063028152



Part 2: Binocular stereo

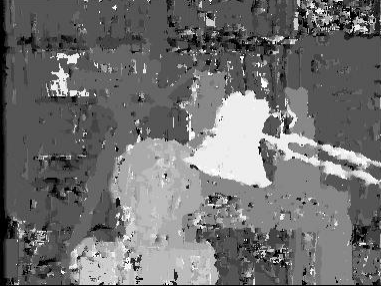




A: Display best output disparity maps for both pairs.




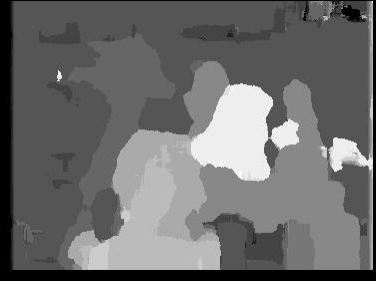
tsukuba block size 15 disparities 16



B: Study of implementation parameters:

1. **Search window size:** show disparity maps for several window sizes and discuss which window size works the best (or what are the tradeoffs between using different window sizes). How does the running time depend on window size?


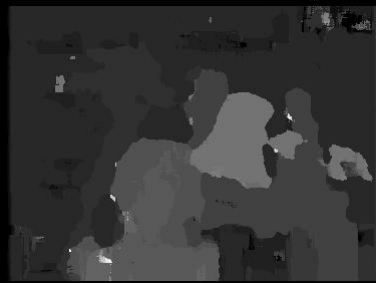


Block size	# of disparities	Disparity map	Running time (s)
5	16		8.618
7	16		8.553
9	16		8.567
11	16		8.567
13	16		8.556

15	16		8.536
17	16		8.410
19	16		8.465
21	16		8.471

When the block size is 15, the disparity map looks the best.

The above running is the average time of running the code 10 times with respect to the parameters. As we can see, the running time is **NOT** depend on window size.

2. **Disparity range:** what is the range of the scanline in the second image that should be traversed in order to find a match for a given location in the first image? Examine the stereo pair to determine what is the maximum disparity value that makes sense, where to start the search on the scanline, and which direction to search in. Report which settings you ended up using.




Block size	# of disparities	Disparity map		
15	16			
15	32			
15	48			
15	64			
15	80			

I selected 16 for the number of disparities because this parameter yielded the optimal visual result. The x coordinates of the scanlines begin at half the window size and extend to the width of the image minus half the window size. The search direction is horizontal, given that the original image is already rectified.

Furthermore, the running time increases significantly with an increase in the number of disparities, as illustrated below.

```
time for tsukuba block size 15 disparities 16: 7.331877946853638 s
time for tsukuba block size 15 disparities 32: 14.014255285263062 s
time for tsukuba block size 15 disparities 48: 20.40999460220337 s
time for tsukuba block size 15 disparities 64: 26.620291709899902 s
time for tsukuba block size 15 disparities 80: 33.034539461135864 s
```

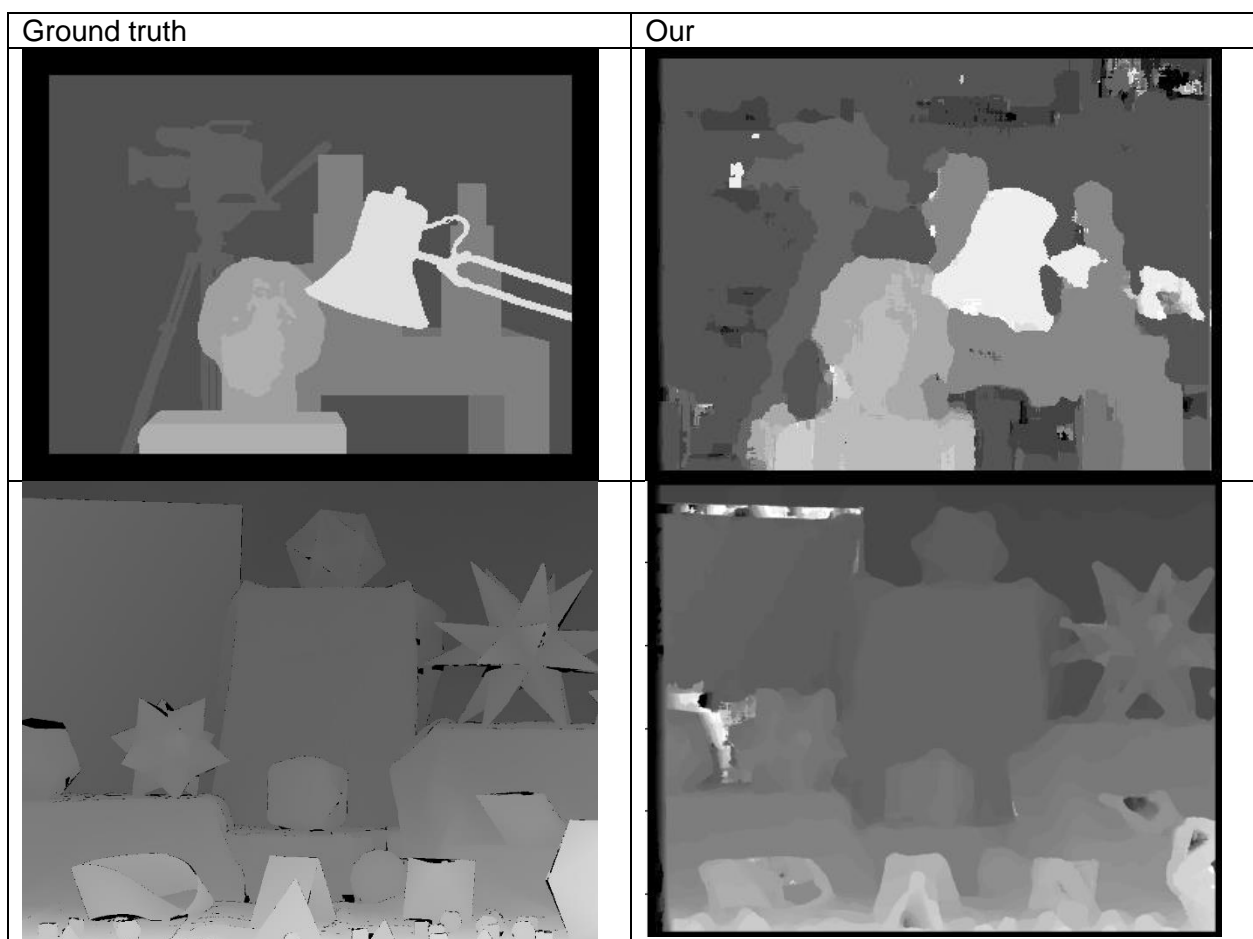

3. **Matching function:** try sum of squared differences (SSD), sum of absolute differences (SAD), and normalized correlation. Show the output disparity maps for each. Discuss whether there is any difference between using these functions, both in terms of quality of the results and in terms of running time.

Block size	# of disparities	Method	Disparity map	Running time (s)
15	16	SSD		6.541
		SAD		6.533
		normalized correlation		6.507

As demonstrated above, the outcomes obtained using the sum of squared differences (SSD) and normalized correlation exhibit significant similarity. It is crucial to note that employing the

absolute differences (SAD) without normalizing the original input may lead to unexpected overflow issues. To address this, image normalization is imperative to ensure that all pixel values fall within the range of 0 to 1 when using SAD. However, if use this normalization step, the application of SAD can result in the blurring of object boundaries in the image. Consequently, the SAD results may not be as accurate as anticipated.

C: Discuss the shortcomings of your algorithm. Where do the estimated disparity maps look good, and where do they look bad? What would be required to produce better results? Also discuss the running time of your approach and what might be needed to make stereo run faster.



Shortcomings:

1. There are some errors in the right above corner of the disparity map.



2. The boundary of each object in the disparity map is not shape enough. And sometimes the boundaries are not successive.
3. The running time is too long. It takes 8 seconds for an image which size is (288, 384).

Good things:

The disparity map accurately represents the depth of objects in the image, with objects farther from the camera appearing in darker colors. Additionally, objects within the disparity map exhibit a close color range within specific contiguous regions.

With my own code, it takes 8 seconds for an image which size is (288, 384). Therefore, I also attempted the function using the cv2 library in Python, as demonstrated below. The following code executes significantly faster, completing in less than 1 second for each stereo pair. This improvement is attributed to the default exclusion of the full-scale two-pass dynamic programming algorithm, which would otherwise consume $O(W*H*numDisparities)$ bytes. This optimization is especially noteworthy for stereo images of size 640x480 and becomes even more pronounced for larger HD-size pictures.

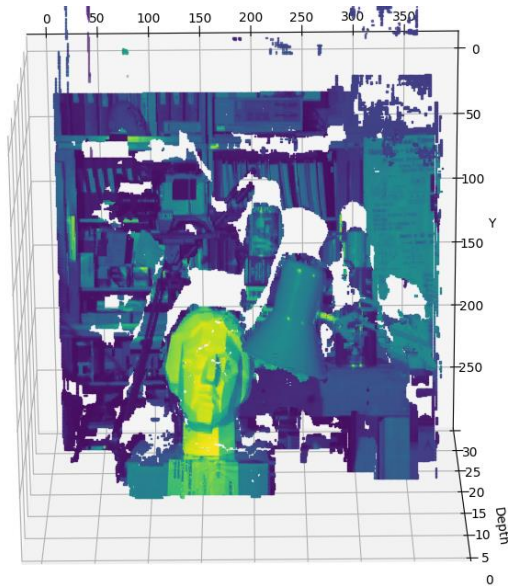
```
# Initiate and StereoBM object
stereo = cv2.StereoBM_create(numDisparities=64, blockSize=31)
# compute the disparity map
disparity = stereo.compute(imgL,imgR)
```

Part 3: Extra Credit

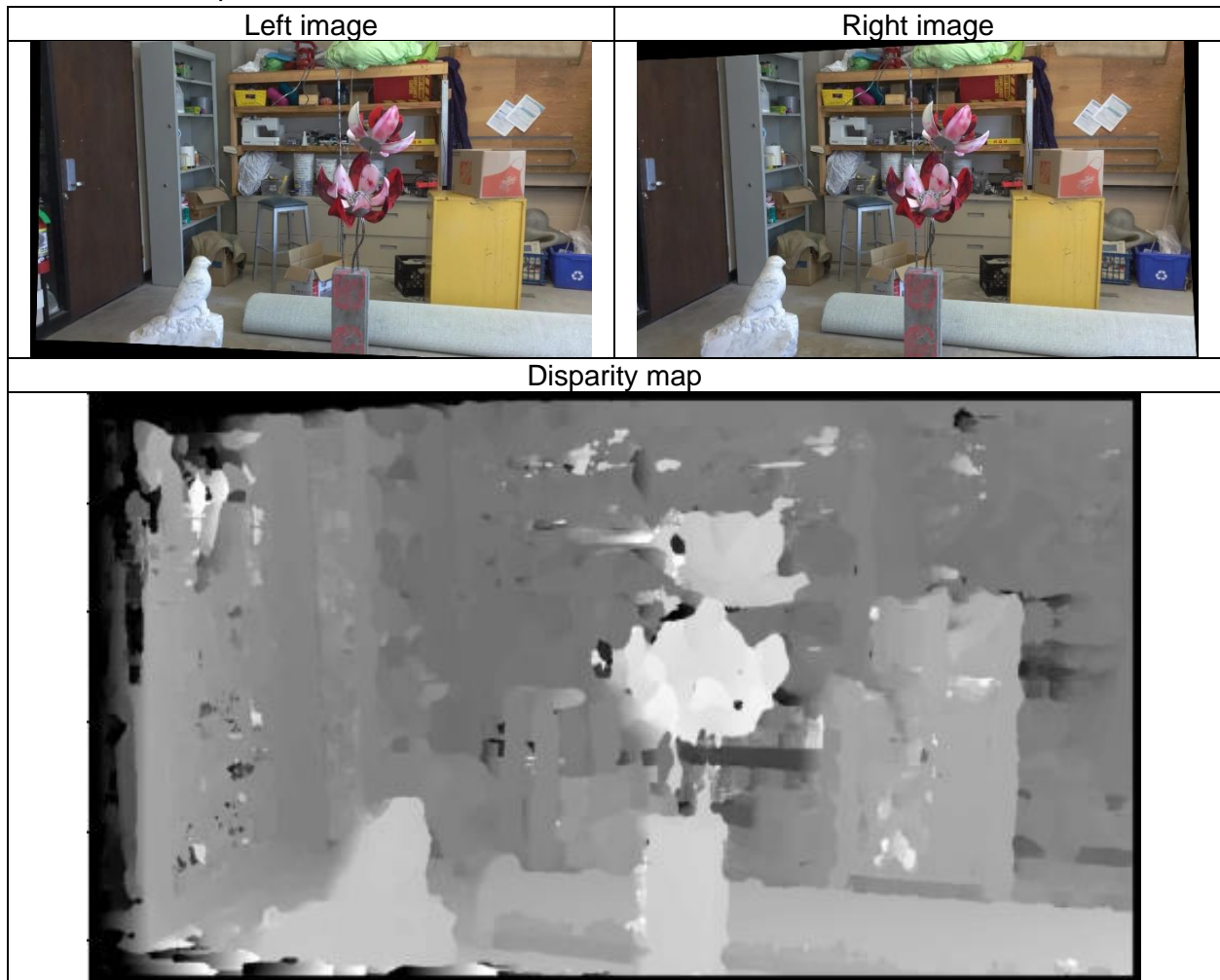
Post any extra credit for parts 1 or 2 here. Don't forget to include references, an explanation, and outputs to receive credit. Refer to the assignment for suggested outputs.

Part 2:

1. Convert your disparity map to a depth map and attempt to visualize the depth map in 3D. Just pretend that all projection rays are parallel, and try to "guess" the depth scaling constant. Experiment with displaying a 3D point cloud, or computing a Delaunay triangulation of that point cloud.




2. Find additional rectified stereo pairs on the Web and show the results of your algorithm on these pairs.



3. Try to incorporate non-local constraints (smoothness, uniqueness, ordering) into your algorithm. You can come up with simple heuristic ways of incorporating these constraints, or try to implement some of the more advanced methods discussed in the course (dynamic programming, graph cuts). For this part, it is also fine to find code on the web.

According to https://docs.opencv.org/4.x/d2/d85/classcv_1_1StereoSGBM.html

Parameters	Disparity map result
<pre>block_size = 5 min_disp = 0 max_disp = 16 uniquenessRatio = 5 speckleWindowSize = 50 speckleRange = 2 disp12MaxDiff = 9</pre>	
<pre>block_size = 3 min_disp = 0 max_disp = 64 uniquenessRatio = 5 speckleWindowSize = 50 speckleRange = 2 disp12MaxDiff = 20</pre>	