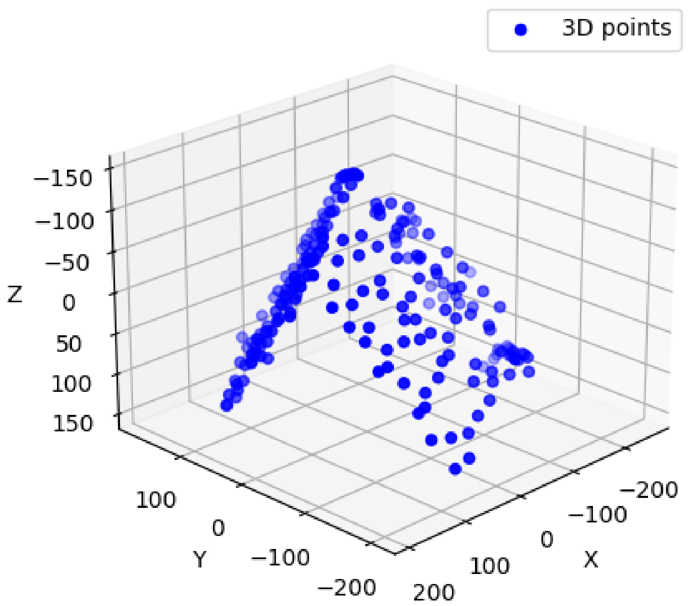
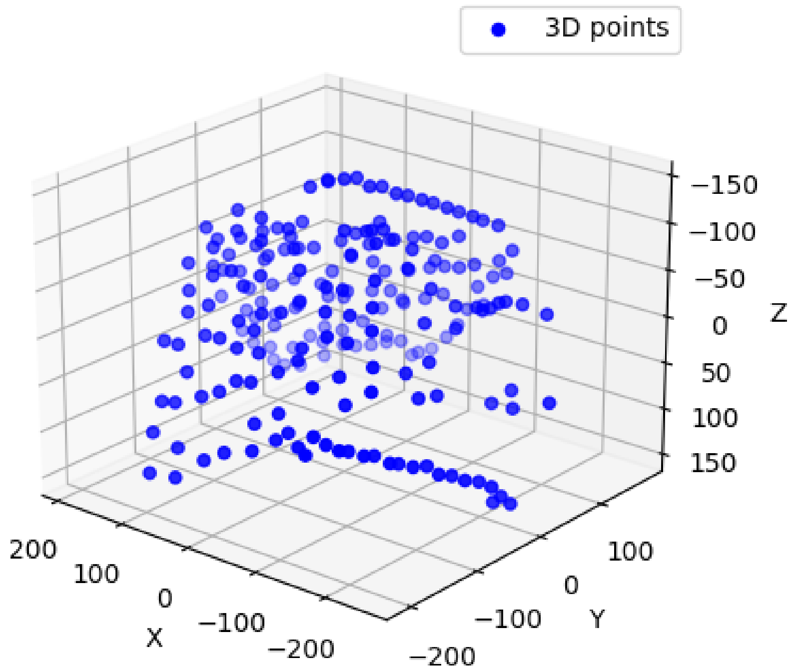
CS543/ECE549 Assignment 5

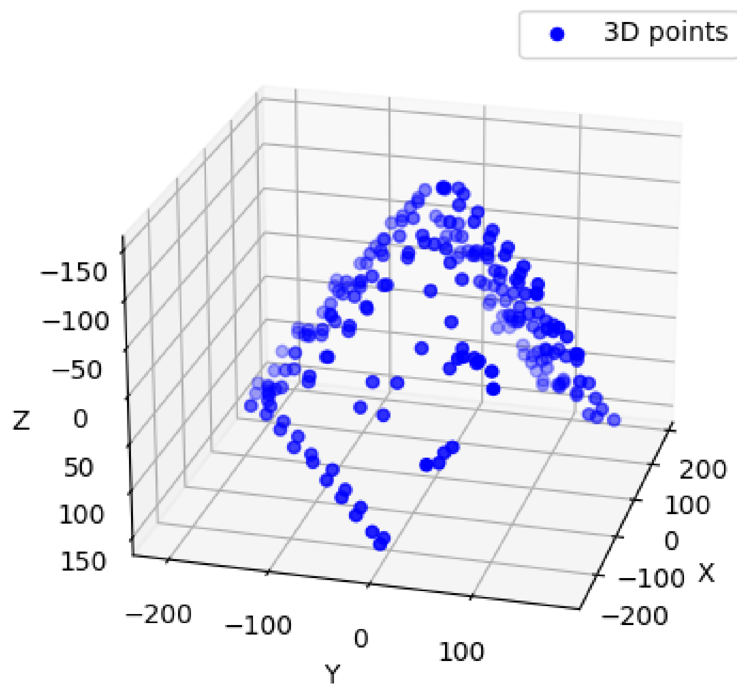
Name: Zihan Hu
NetId: zihanhu2

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.




3D points
−150
−100
−50
0
50
100
150
Z
200
100
0
−100
−200
X
100
0
−100
−200
Y


3D points
−150
−100
−50
0
50
100
150
Z
−200
−100
0
100
Y
200
100
0
−100
−200
X

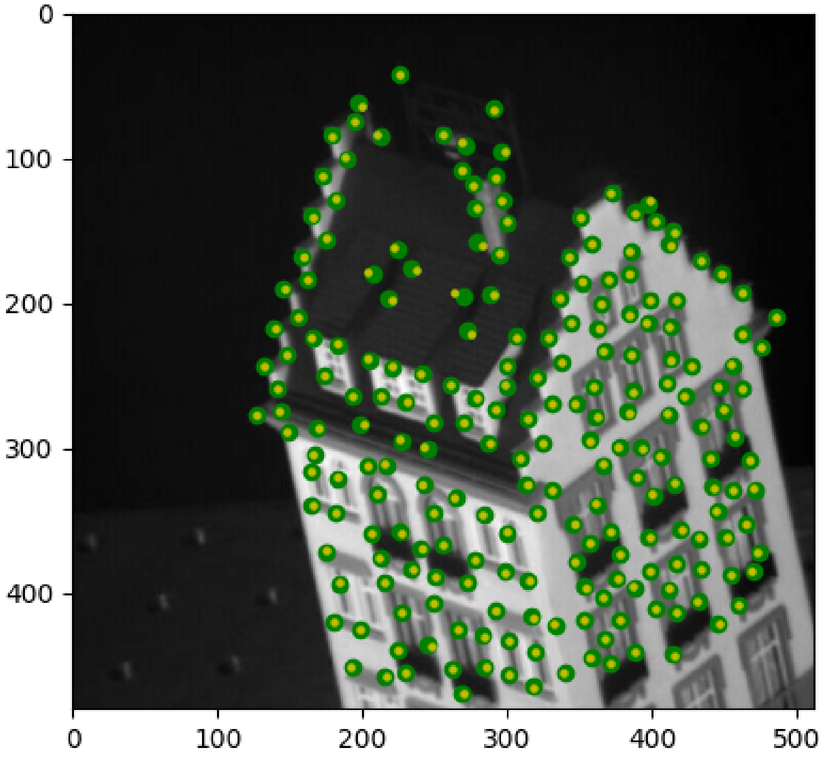
Matrix Q :

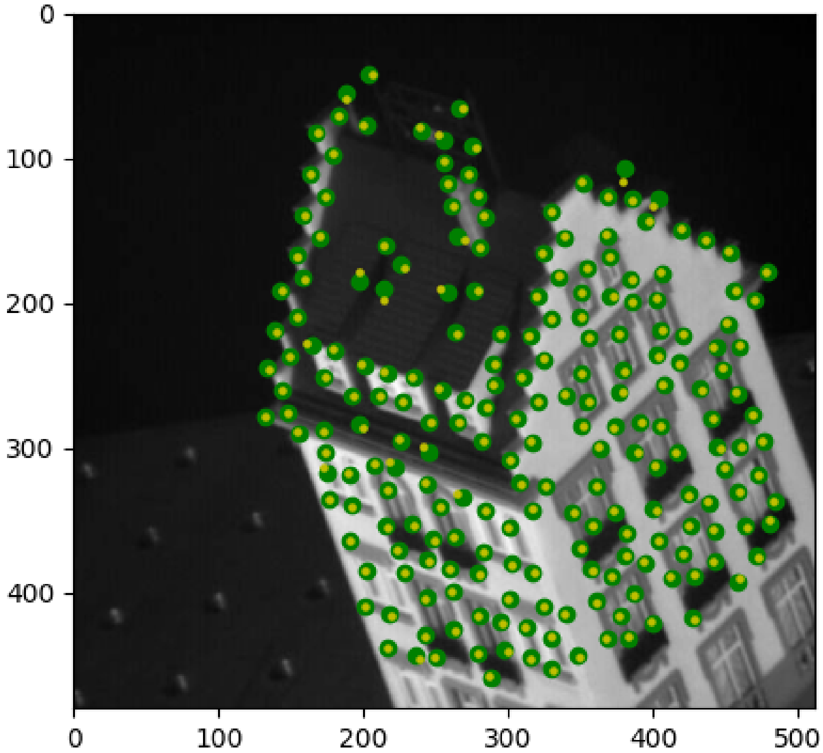
[[0.0796754 0.   0.   ]
[0.00075727 0.08491913 0.   ]
[0.00261389 0.00474193 0.03999405]]

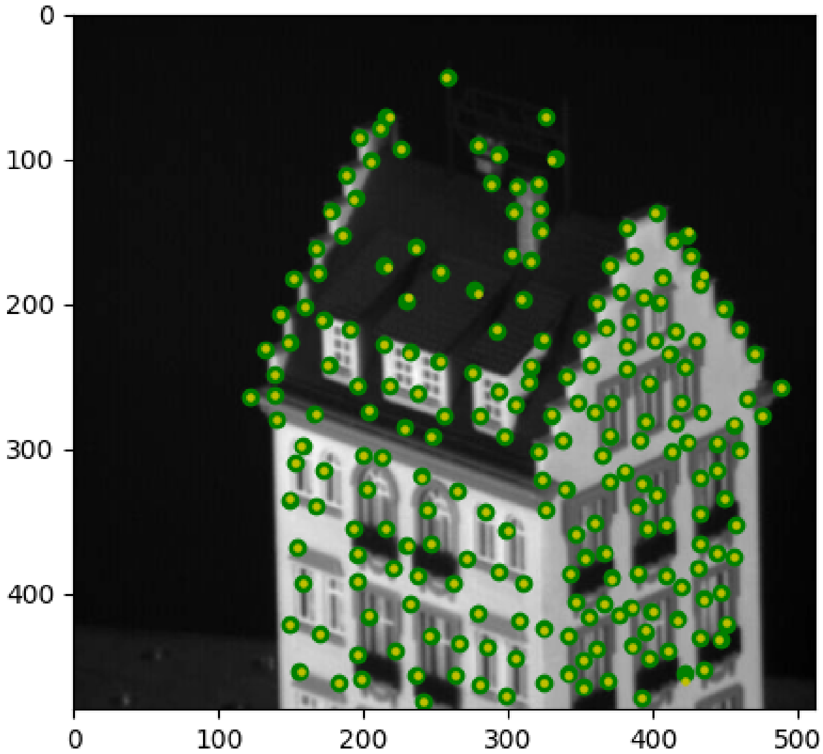
The original output has multiple solutions instead of one solution. I think the current reconstruction does not have an ambiguity based on my output since I eliminate the affine ambiguity by calculating the Q matrix to represent the orthographic projection.

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

observed feature points are green, and estimated projected 3D points are yellow marks.

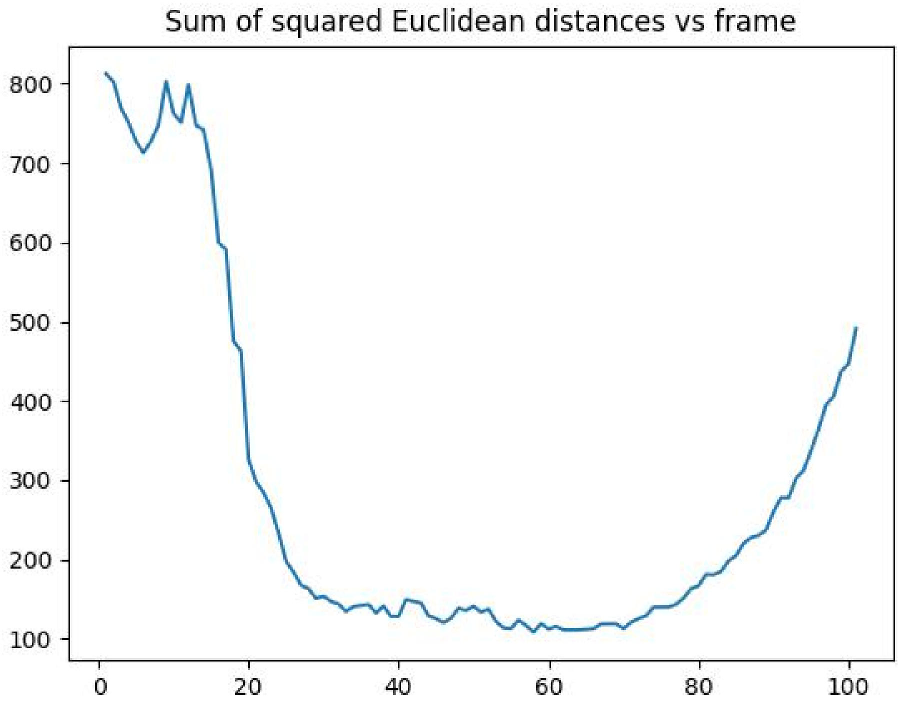



0
100
200
300
400
0
100
200
300
400
500


0
100
200
300
400
0
100
200
300
400
500

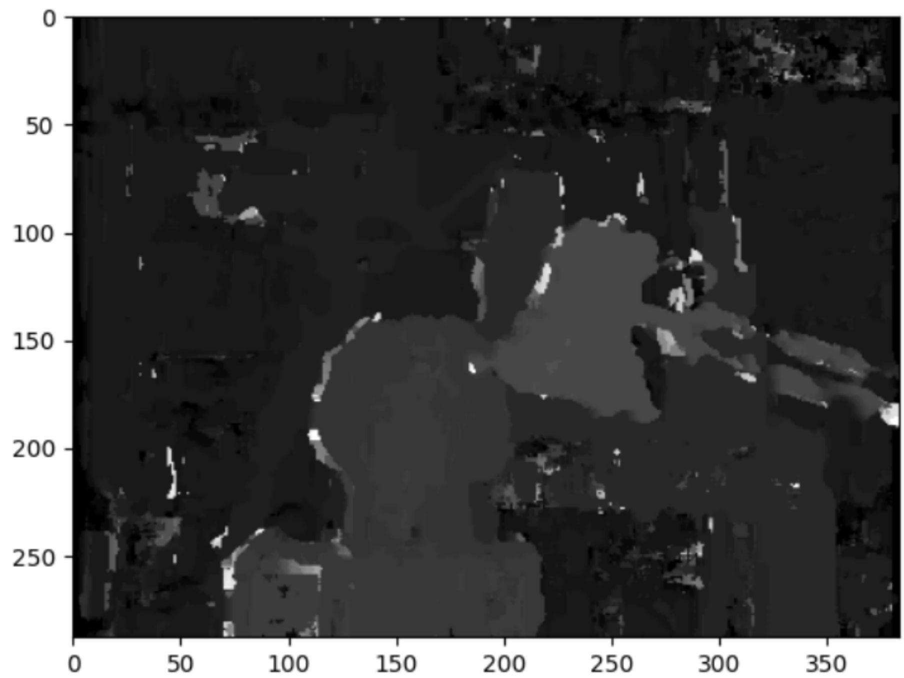
C: Report your total residual (sum of squared Euclidean distances, in pixels, between the observed and the reprojected features) over all the frames, and plot the per-frame residual as a function of the frame number.

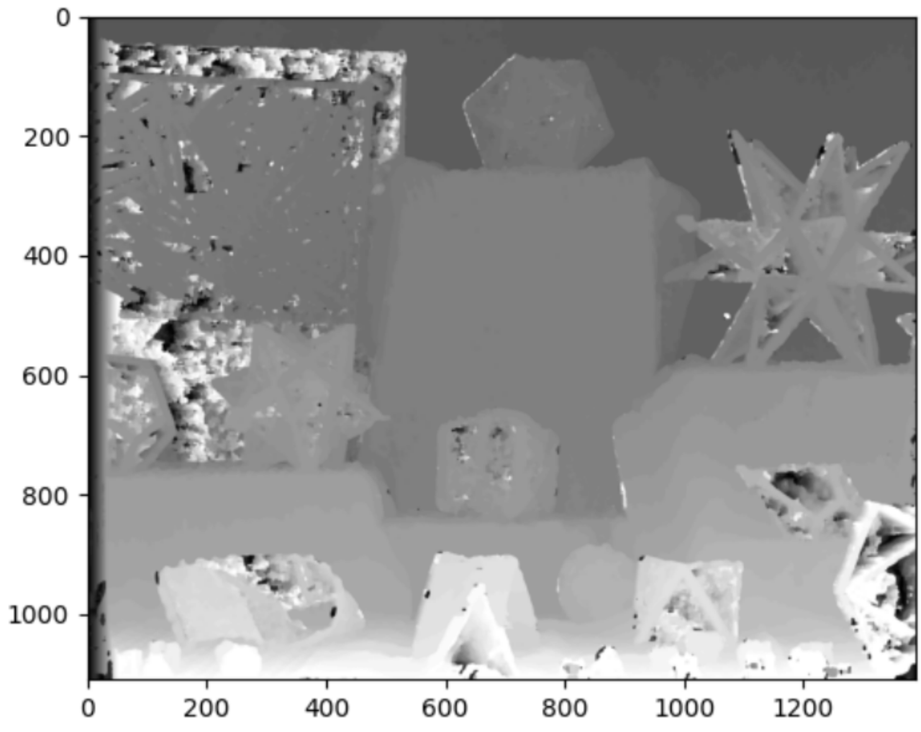
residual = 28583.41464



Part 2: Binocular stereo

A: Display the best output disparity maps for both pairs.

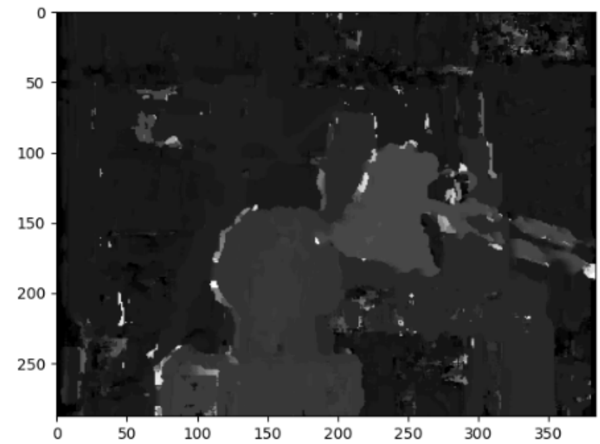


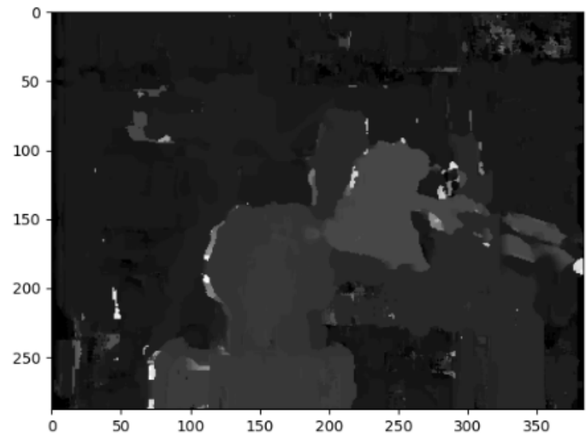


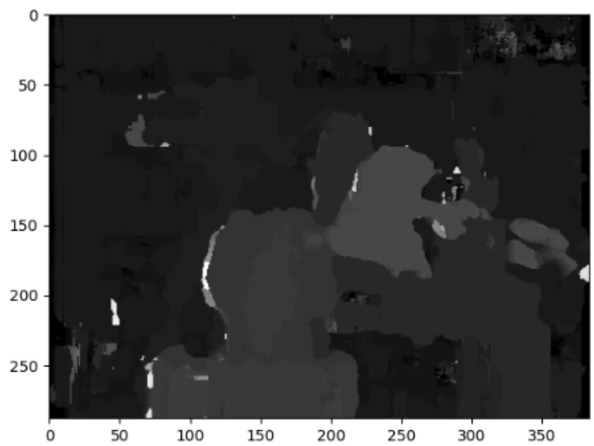
B: Study of implementation parameters:

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?

My first image has a window size of 7, the second image has a window size of 11 and the third image has a window size of 13. Personally speaking, I think window size 13 has the best. The larger window size can capture more information when comparing two patches but it needs more computation time.



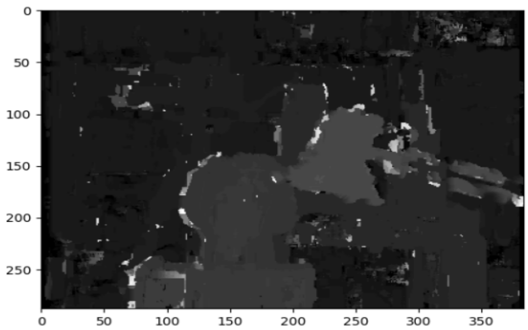


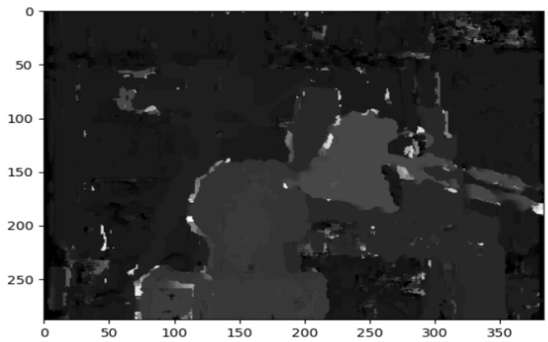


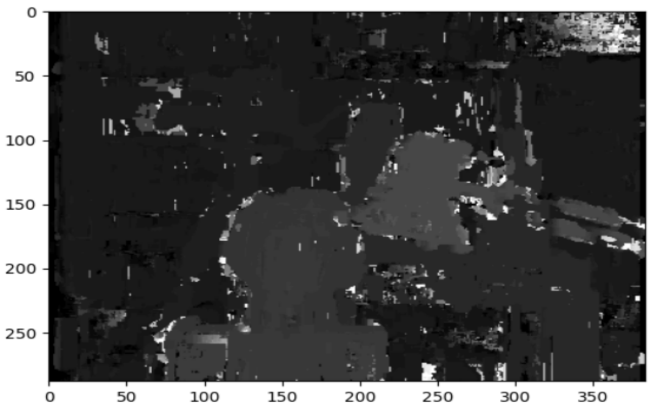
2. **Disparity range:** what is the range of the scanline in the second image that should be traversed in order to find a match for a given location in the first image? Examine the stereo pair to determine what is the maximum disparity value that makes sense, where to start the search on the scanline,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

The disparity range I ended up with is [-50, 10]. The range of the scanline should be traversed in order to find a match is min = max(0, disparity) max = min(width, width + disparity), and the direction of the search are from left to right

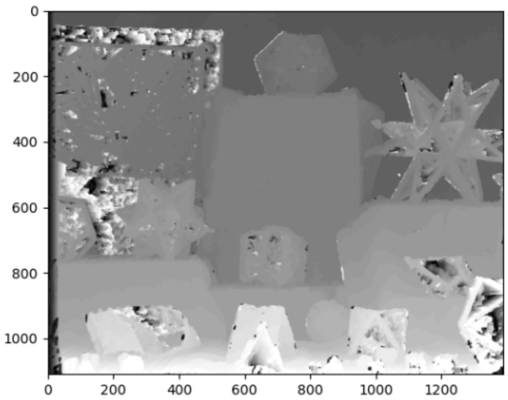
3. **Matching function:** try the sum of squared differences (SSD), the sum of absolute differences (SAD), and normalized correlation. Discuss whether there is any difference between using these functions, both in terms of the quality of the results and in terms of running time.

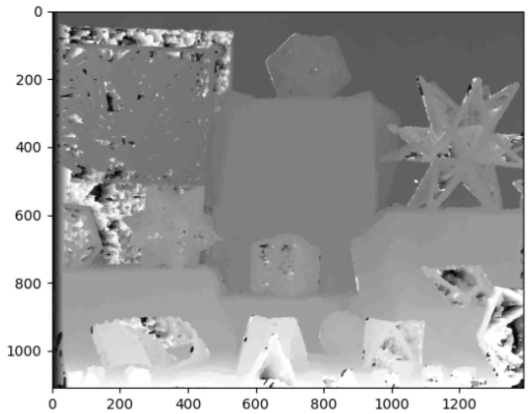


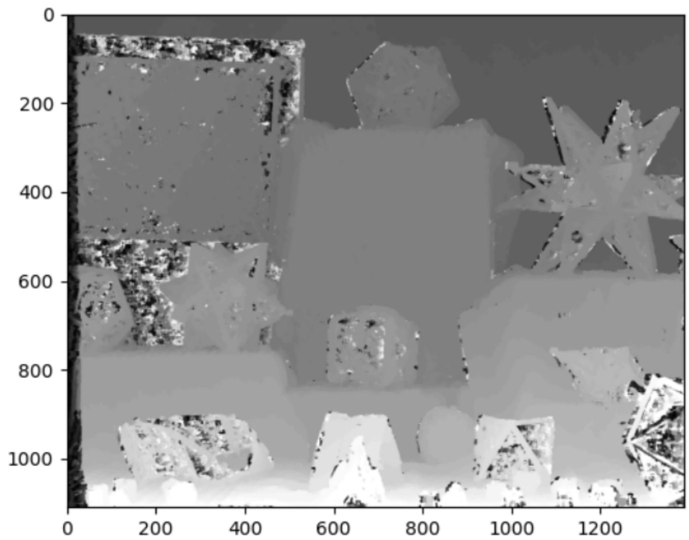




The first image is using SSD, the second image is using SAD, and the third image is using NCC. For comparison, SAD outperforms the other two approaches for the first image. However, NCC will cost much more time compared to SSD and SAD.







The first image is using SSD, the second image is using SAD, and the third image is using NCC. For comparison, NCC outperforms the other two approaches for the second image. However, NCC will cost much more time compared to SSD and SAD.

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.

The shortcomings of my algorithm are the long-running time. The sliding window approach needs a lot of computation resources.

The estimated disparity maps look good when the patch has a distinctive pattern such as a Lambertian surface. However, it gets worse when there is a repetition of the pattern, similar patches, and occlusions.

We can use some optimization methods to improve our results such as dynamic programming and graph cuts.

My running time is so long. I believe we can utilize the conclusion to speed up the process by separate into two linear operations. In addition, if we can do some parallel programming that can improve our speed.