TEL AVIV UNIVERSITY

Computer Vision

2022-2023

Assignment #2

Students

ID1: 301613501

ID2: 305207946

Part A: Distance Tensor Computation

1. Sum of Squared Difference Distance – Manual Calculation

Given the following rectified images, we shall calculate the ssdd values as required.
The ssdd values below calculated by the following form:

$$ssvd = \sum_{i=0}^{8} \left(Yellow(i) - Blue(i)\right)^2$$

$$\cdot\, Where\ Yellow\ is\ the\ left\ kernel\ and\ blue\ is\ the\ right\ one$$

ssdd(1,2,1) 426

Left

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

Right

1	1	1	1	1
2	2	2	2	2
3	3	3	3	3
4	4	4	4	4

ssdd(1,2,2) 426

Left

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

Right

1	1	1	1	1
2	2	2	2	2
3	3	3	3	3
4	4	4	4	4

ssdd(1,2,3) 426

Left

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

Right

1	1	1	1	1
2	2	2	2	2
3	3	3	3	3
4	4	4	4	4

ssdd(2,3,0) 1191

Left

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

Right

1	1	1	1	1
2	2	2	2	2
3	3	3	3	3
4	4	4	4	4

ssdd(2,3,1) 1191

Left

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

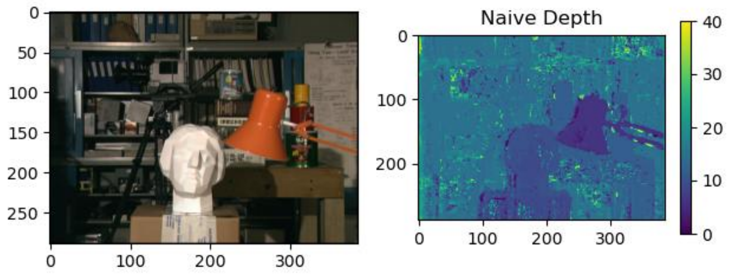
Right

1	1	1	1	1
2	2	2	2	2
3	3	3	3	3
4	4	4	4	4

2. See implementation of **ssd_distance** in **solution.py**

Part B: Naive Depth Map

3. See implementation of **naive_labeling** in **solution.py**



4. Naive results explanation.

In a naive approach, the SSDD tensor used to compare two images by simply calculating the sum of the squared differences between the pixel values of the images at each corresponding position (comparing only a single pixel each time).

In the previous section we created a map matrix (called ssdd_tensor) that stores those SSDD values for a range of different disparity values.

We learned in class that the SSD distance profile with respect to disparity has a convex profile.

Therefore, the naive labeling function in this section will return to us the disparity label of the minimum SSD distance for each pixel.

However, this approach can be problematic for several reasons:

- It does not take into account the spatial relationship between pixels, which means that images that are slightly shifted or rotated may be considered dissimilar even if they are visually similar.
- The local Naive method considers for each pixel its surroundings only, making it more prone to noise. Such that, this method is sensitive to noise and other perturbations in the images, which means that small differences between the images can lead to large differences in the SSDD which affect the disparity value (and of course the label).
- It does not account for changes in lighting or other imaging conditions, which means that images taken under different conditions may be considered dissimilar even if they are actually of the same object or scene.

From the **Naive Depth** map some 'silhouettes' of the same shape as of some objects located at different depths in the **Source Image** can be identified, like the white statue and the orange lamp, which obtained a lower disparity value (corresponds to its depth in the Source Image).

We can see that many pixels in the background of these 'silhouettes' (and inside them) seem not "smooth", which means some pixels have a significantly different disparity value from their neighborhood. This phenomenon is more common at places where the pixel does not differ much from its neighbors in the **Source Image**, i.e., intensity values are quite alike, making the pixel a bad "point of interest" and the obtained disparity profile is noisy.

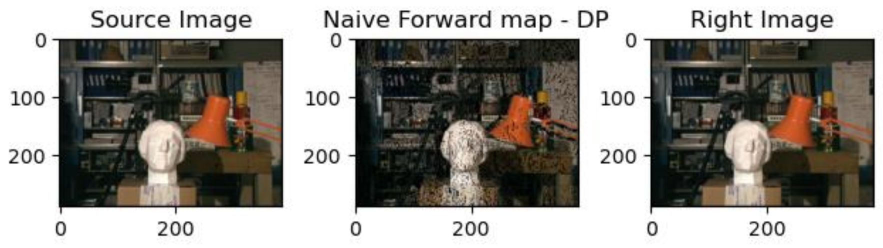
Implementing the forward mapping pixels from the left image to the corresponding pixels on the right image according to the disparity values doesn’t return very clean image, such that many black holes appear in the image.

A black hole in the **Naive Forward Map** is caused when the minimum SSDD we get for a particular pixel is not obtained from the defined disparity range, but is obtained from somewhere else on the epipolar line on the source image.

We can also say that the different angles from where the two rectified images were taken also make their effects where some details may be seen slightly different, or may not be seen at all, from different angles.

Moreover, since there is no smooth prior assumption (saying that nearby pixels share a similar disparity values) is made in the naive labeling method, there are some discontinuities in the **Naive Forward Map** (like in the orange lamp and the white statue). The **Naive Depth** map is not "smooth".

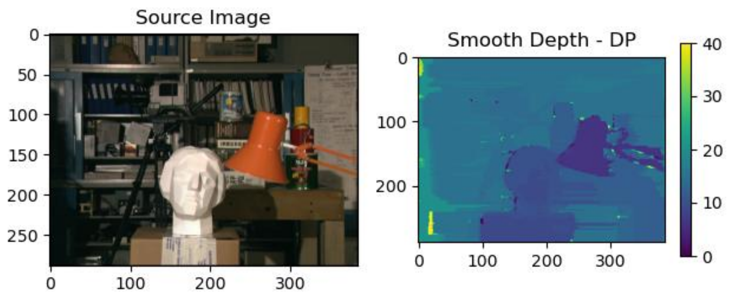
Summarizing the above, we can say that it is even stands out that all the noises and discontinuities that we can see on the depth image in the previous section were translated to the forward mapping image as black holes.



Part C: Depth Map Smoothing using Dynamic Programming

5. See implementation of **dp_grade_slice** in **solution.py**

6. See implementation of **dp_labeling** in **solution.py**



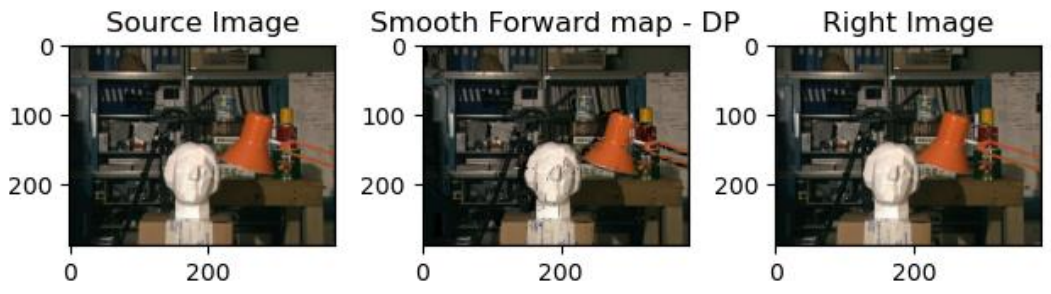
7. Comparing the smooth and the naive approaches.

Unlike the naive method, the Depth Map Smoothing Dynamic Programming (DP) method is taking into account a larger environment (slice) from the SSDD tensor matrix such that it is much less affected by noises and other perturbations.

Due to the fact that nearby pixels are assumed to share a similar disparity value, the map at each row is much "smother" compared to the map obtained using the **Naive** method.

However, since this restriction of having similar disparity values is only enforced in the horizontal direction (along rows), the disparity map of one row might be inconsistent with another's, resulting in a slightly "soft"/"blurred" boundaries of the objects (Streaking Artifacts).

The Smooth Forward map obtained here is much "smoother" than that obtained in the **Naive** method. Also, the number of black pixels is reduced comparing to the Forward map obtained for the **Naive** map.

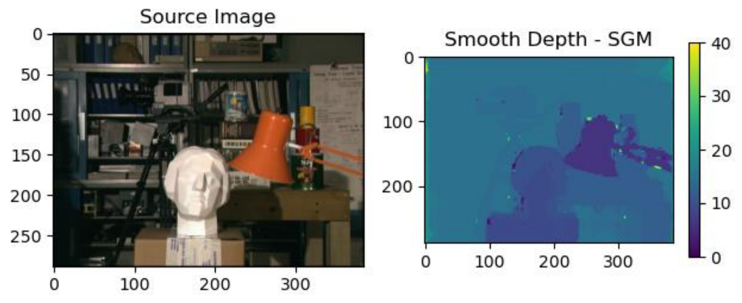


Part D: Depth Image Smoothing Using Semi-Global Mapping

8. See implementation of **slice_per_direction** in **solution.py**

9. The Function **dp_grade_slice** support general length of slices.

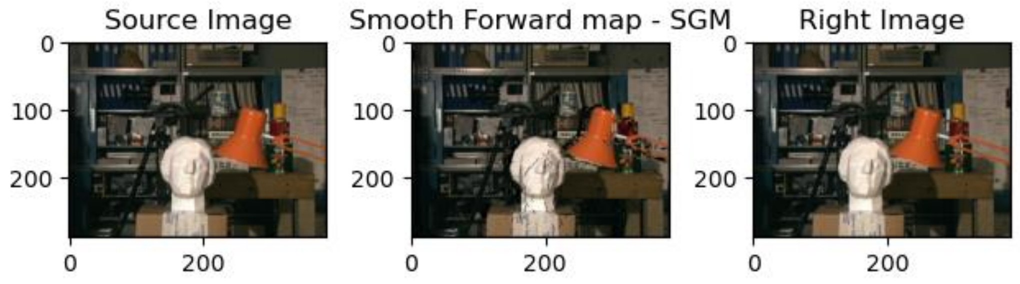
10. See implementation of **sgm_labeling** in **solution.py**



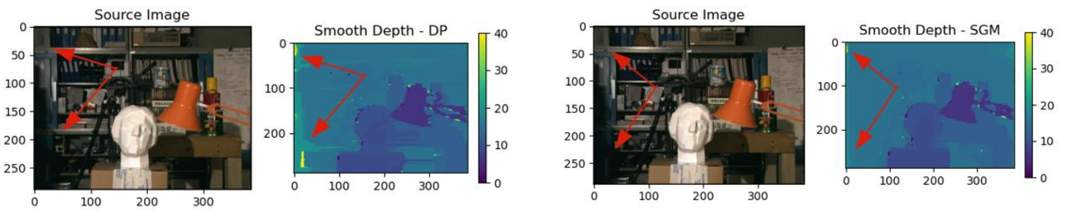
11. Explaining the SGM (Semi-Global Mapping) approach and compare to the previous sections.

Since SGM applies the Smooth Prior assumption to all directions and makes the combination of it along the algorithm (i.e., the disparity value of each pixel is consistent with pixels from different directions), the depth map received from this method is the clearest and most accurate of all methods introduced along this assignment: the 'silhouettes' are smoother and the edges are well defined comparing to the results obtained for the previous methods.

The obtained forward map still suffers from black pixels due to the same reasons mentioned before.

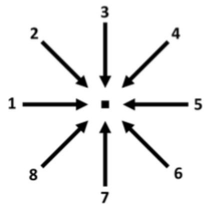


We shall notice that in our image, most of the improvement of the SGM algorithm is no longer seen in the forward mapped image, because of the fact that the left side of the image is very dark (farther explanation by example – if the left corner of the image would not be dark so it was able see the improvement by the SGM algorithm also in the forward mapping image, in the places that we see the improvement in the depth map (see red arrows below).

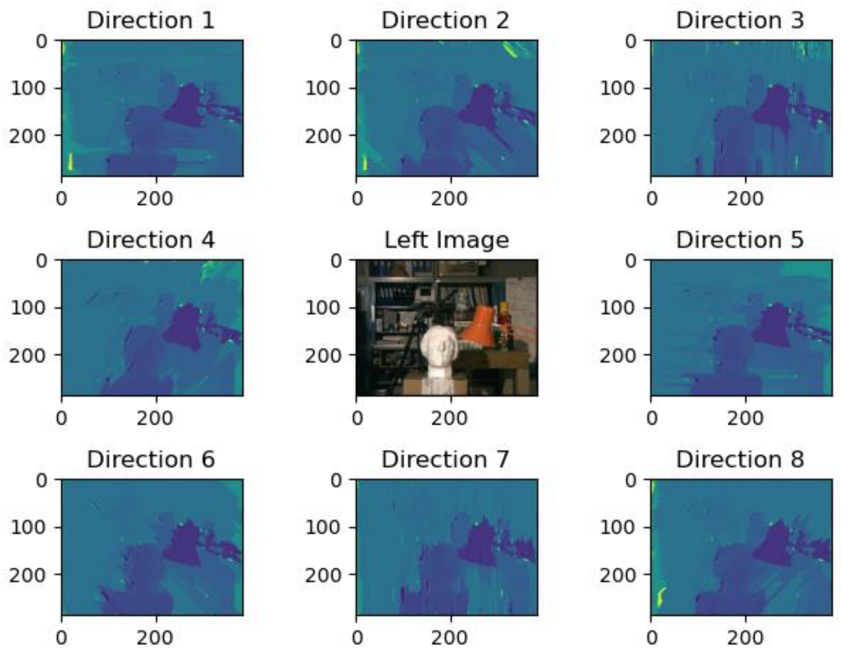


12. See implementation of **dp_labeling_per_direction**in **solution.py**

First, it is clear that the case of direction 1 is similar to the case from 6.
In the image below it is possible to see that each image is a little smeared in the requested direction that we implemented.



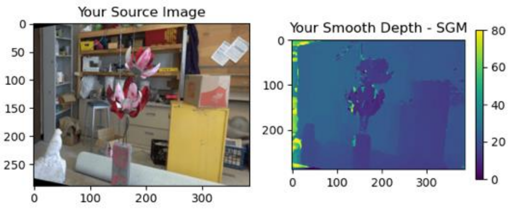
The impact of forcing the Smooth Prior assumption in different directions is easily seen: from each depth map the "smoothness" and "soft" edges of the objects in the image are at its corresponding direction, so as the noise. The averaging of all directions results in a "cleaner" and less "noisy" map, as obtained in Q10.



Part E: Your Own Image

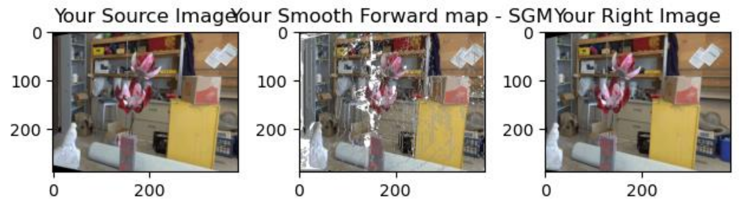
13. Our Images

We ran all the relevant algorithms from the previous sections. Since the horizontal distance is more significant on our images, we increased the disparity range to 40, and just after this change we got the following result with the ability to see the depth of the original image by the SGM mapping image.

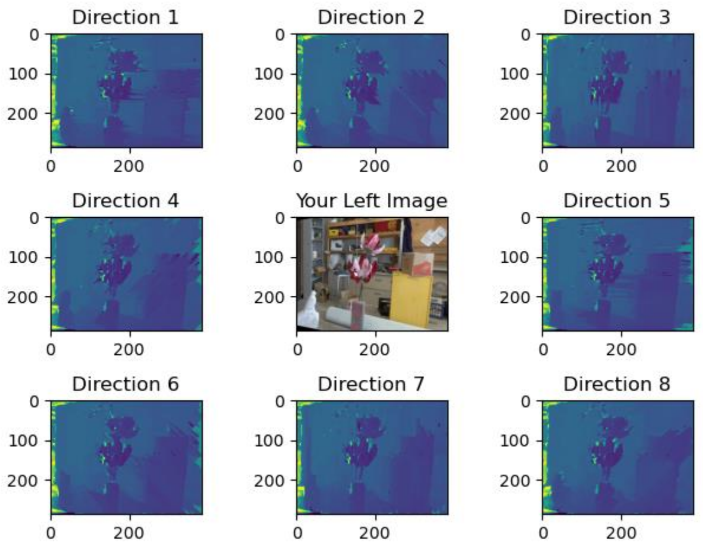


Also here, the Smooth Prior assumption taken at all directions yields a clear disparity map: 'silhouettes' of the different objects in the source image can clearly be seen, so that the low disparity values corresponding to the objects' depth in the image (low disparity values for closer located objects).

In the next forward mapping image, we have another view on the pixels on the objects and the image's edges that were not mapped ideally in the SGM depth map above, due to the same reasons we already mentioned earlier.



In addition, in the next image we can get another view on those pixels (on the edges). While the different directions appliance was not perfectly smooth the edges in contrast to the previous demonstration that we saw in question 12. This is because this time the distance from which both pictures were taken is higher than of ones we used in the previous questions, what makes the difference in the fields of view more significant.

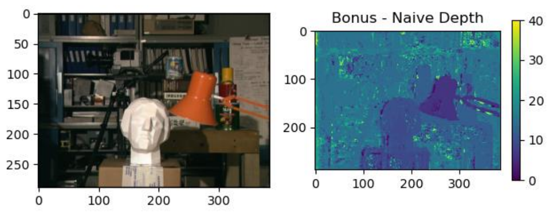
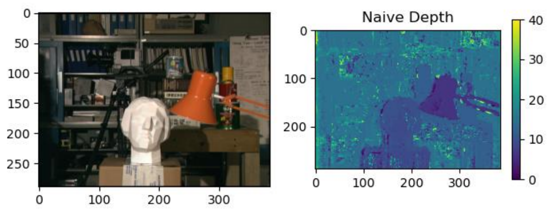
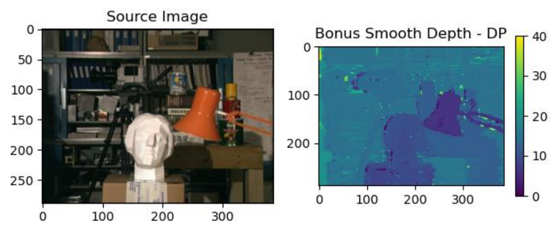
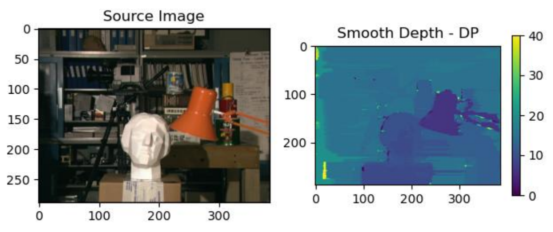
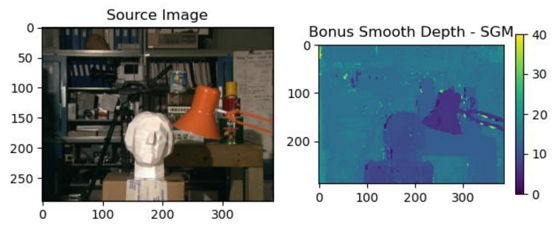
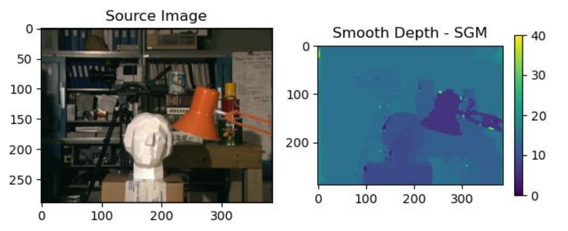


Bonus Part

14. A new metric to measure

Very similar to the original implementation we can choose any measurement method for the distance. For example, instead of using squared difference we can measure the absolute difference. Since this calculation is not power the difference, we can guess that the result will be slightly less sensitive to noise.

In the comparison table below, it can be seen that there are ranges that the absolute difference ignores some noises (mainly in image edges of the single direction DP) but in general, the SSDD gave mush better performance.

New Metric ABS Calculation	**SSDD**
 Bonus - Naive Depth	 Naive Depth
 Source Image Bonus Smooth Depth - DP	 Source Image Smooth Depth - DP
 Source Image Bonus Smooth Depth - SGM	 Source Image Smooth Depth - SGM