Assignment 2  
Mai Dabas (315026294) & Adi Green (313472417)  
Computer Vision

**Part A: Distance Tensor Computation**

1. Computing ssdd as requested in the questionnaire:

As given:

Left Right

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 | 2 |
| 3 | 3 | 3 | 3 | 3 |
| 4 | 4 | 4 | 4 | 4 |

* **ssdd(1,2,2):** corresponds to row =1, column 2 and disparity value=0.

In order to compute the ssdd(1,2,2), we need to take a 3x3 window around the pixel in row=1, column=2 in the left image, and a window around the pixel in row=1, column=2 in the right image:

Left Right

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 | 2 |
| 3 | 3 | 3 | 3 | 3 |
| 4 | 4 | 4 | 4 | 4 |

The sum of squared differences between the two windows will be identical to the one calculated in the questionnaire (values in each window are the same).

sdd(1,2,2) = 426

* **ssdd(1,2,3):** corresponds to row =1, column 2 and disparity value=1.

In order to compute the ssdd(1,2,3), we need to take a 3x3 window around the pixel in row=1, column=2 in the left image, and a window around the pixel in row=1, column=3 in the right image:

Left Right

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 | 2 |
| 3 | 3 | 3 | 3 | 3 |
| 4 | 4 | 4 | 4 | 4 |

The sum of squared differences between the two windows is the same as before (values in each window are the same).

ssdd(1,2,3) = 426

* **ssdd(2,3,0):** corresponds to row =2, column 3 and disparity value=-2.

In order to compute the ssdd(2,3,0), we need to take a 3x3 window around the pixel in row=2, column=3 in the left image, and a window around the pixel in row=2, column=1 in the right image:

Left Right

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 | 2 |
| 3 | 3 | 3 | 3 | 3 |
| 4 | 4 | 4 | 4 | 4 |

The sum of squared differences between the two windows:

|  |  |  |
| --- | --- | --- |
| 8-2 | 9-2 | 10-2 |
| 13-3 | 14-3 | 15-3 |
| 18-4 | 19-4 | 20-4 |

.^2 = 36+49+64+100+121+144+196+225+256=**1191**

* **ssdd(2,3,1):** corresponds to row =2, column 3 and disparity value=-1.

In order to compute the ssdd(2,3,1), we need to take a 3x3 window around the pixel in row=2, column=3 in the left image, and a window around the pixel in row=2, column=2 in the right image:

Left Right

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 | 2 |
| 3 | 3 | 3 | 3 | 3 |
| 4 | 4 | 4 | 4 | 4 |

The sum of squared differences between the two windows will be the same as calculated above (the values in each window is the same):

ssdd(2,3,1) = 1191

1. In the code.

**Part B: Naive Depth Map**

1. In the code.
2. The resulting image:

A picture containing graphical user interface

Description automatically generated

Forward mapping using the naïve approach:

Graphical user interface, application

Description automatically generated

As seen in the resulting images, the forward mapped image suffers from noise which appears as black dots in the image. This result corresponds to the expectation from the naïve approach, as it is very sensitive to noise. However, some regions in the image were decently labeled correctly, for example, it can be seen in the depth image that the lamp is almost entirely labeled in a darker color, meaning the lamp I recognized as closer to the camera than the other objects in the image.

**Part C: Depth Map Smoothing using Dynamic Programming**

1. In the code.
2. In the code.
3. The depth map as obtained using the dynamic programming:

Graphical user interface

Description automatically generated

Forward mapping using the dynamic approach:

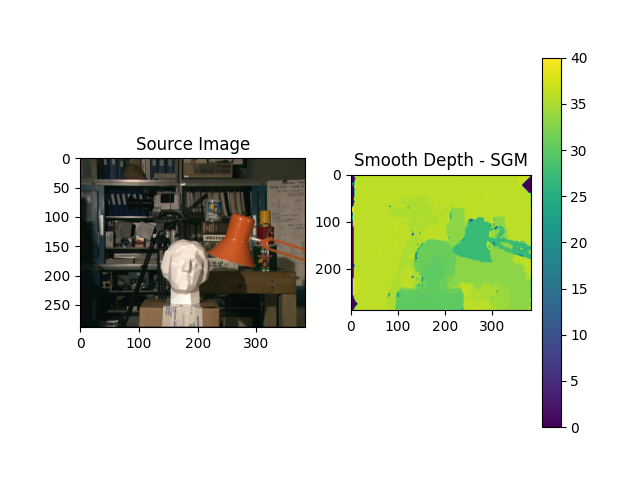
Graphical user interface, application

Description automatically generated

With respect to the depth map obtained by the naïve approach, the dynamic programming resulted much more smooth and less noisy depth map, and the depth of the different objects in the image were labeled more successfully (e.g., the lamp, the statue, the table and the background), when closer objects are colored in darker color and distanced objects in lighter color. Moreover, the forward mapped image is also less noisy and much smoother than the one obtained by the naïve approach. However, a "smearing" noise is displayed in the depth map, since, as we know, the dynamic programming is dome in one direction only (horizontally). This issue will be solved in the next sessions, when using semi-global mapping.

**Part D: Depth Image Smoothing Using Semi-Global Mapping**

1. In the code.
2. In the code.
3. In the code.
4. The depth map as obtained from averaging all directions using the dynamic programming:



Forward mapping:

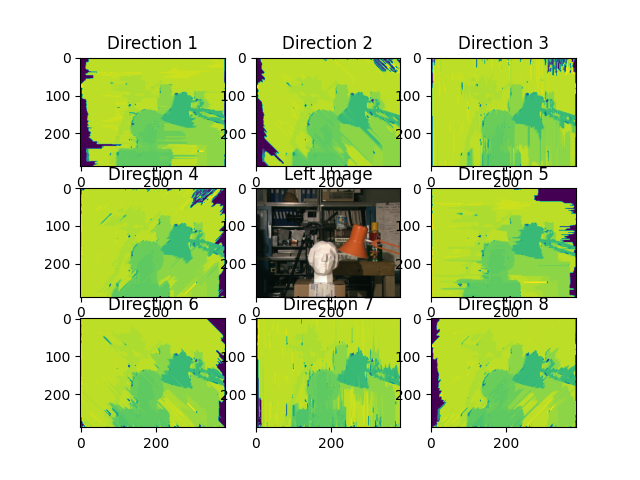
Graphical user interface, timeline

Description automatically generated

We can see that we solved the issue we faced in part C where the depth map was smeared in a single direction. Because we averaged each point based on the eight different depth maps for each direction – overall the smearing was removed.

1. In the code.

The depths maps are “smeared” in all 8 different directions, as seen below.



**Part E: Your own image**

1. Our images:

|  |  |
| --- | --- |
| Left Image | Right Image |
|  |  |