Eötvös Loránd University (ELTE)
Department of Information Systems

3D Sensing and Sensor Fusion

Stereo vision

Author
Mahmuda Rawnak Jahan (H0RQGF)

October 31, 2022

Contents

Chapter 1

Deliverable

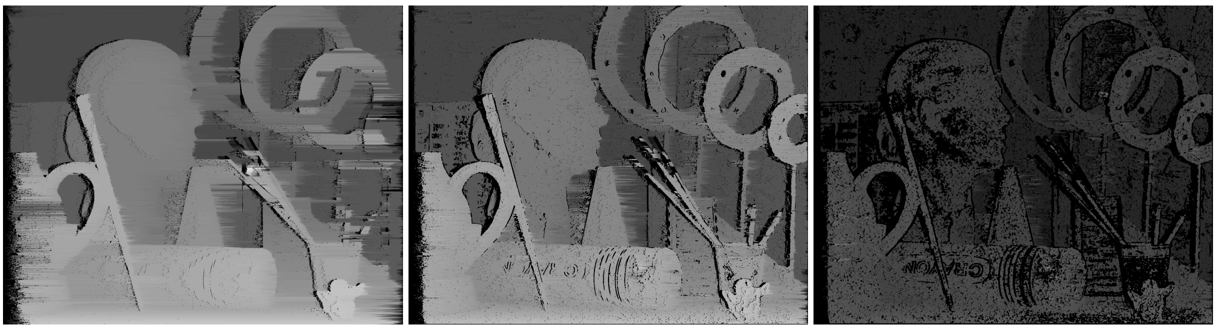
The goal of this task is to implement and compare different methods for calculating disparity maps from stereo images.

1.1 Subtask-1: Algorithms

Three different algorithms have been implemented and used for comparing 6 stereo pairs, taken from Middlebury Stereo 2005 Datasets. The Algorithms are:

1. Naive Approach
2. Dynamic Programming Approach
3. OpenCV Implementation (cv::StereoBM)

Figure 1.1 represents the disparity map generated of dynamic programming approach for window size 1, 3, 5.

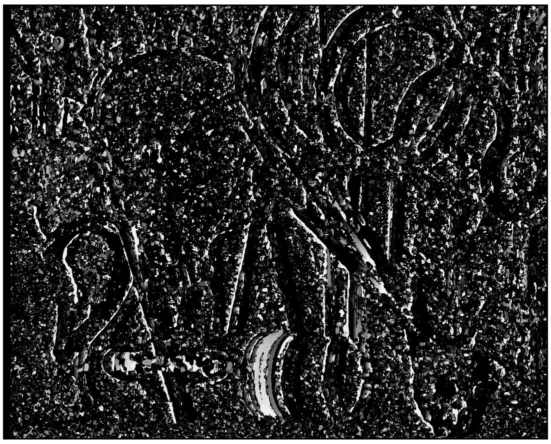


(a) window = 1x1 (b) window = 3x3 (c) window = 5x5

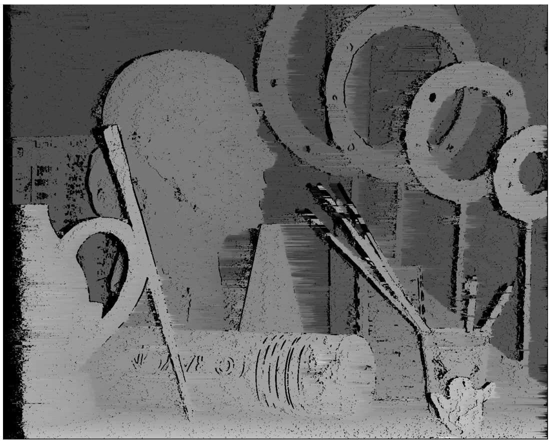
Figure 1.1: Disparity map generated with dynamic programming for different window size.

Figure 1.2 represents the output of above mentioned three algorithms for Art dataset, with 3x3 window size and the corresponding ground truth disparity image.

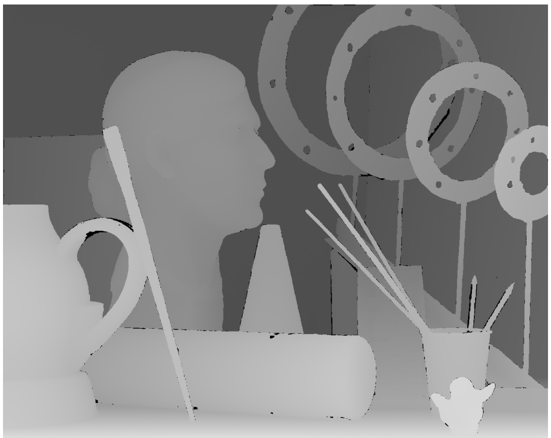
(a) Disparity map (Naive Approach)



(b) Disparity map (OpenCV StereoBM)



(c) Disparity map (Dynamic Programming)



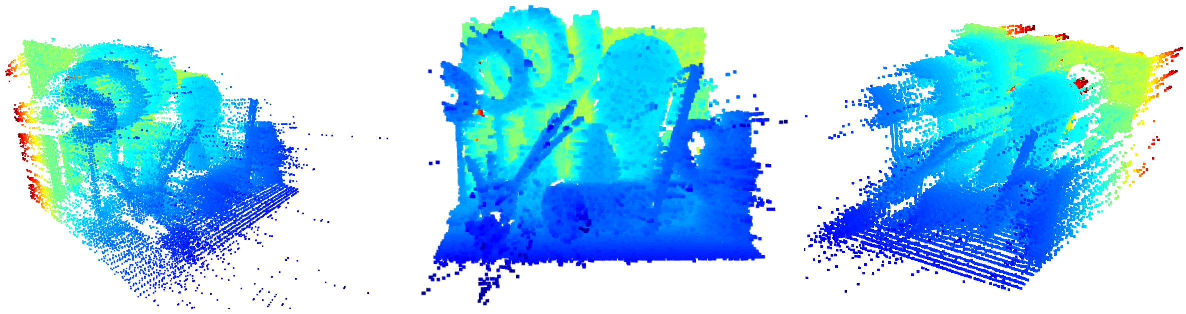
(d) Ground Truth Disparity

Figure 1.2: Disparity map generated with different algorithms and corresponding GT.

1.2 Subtask-2: 3D Display

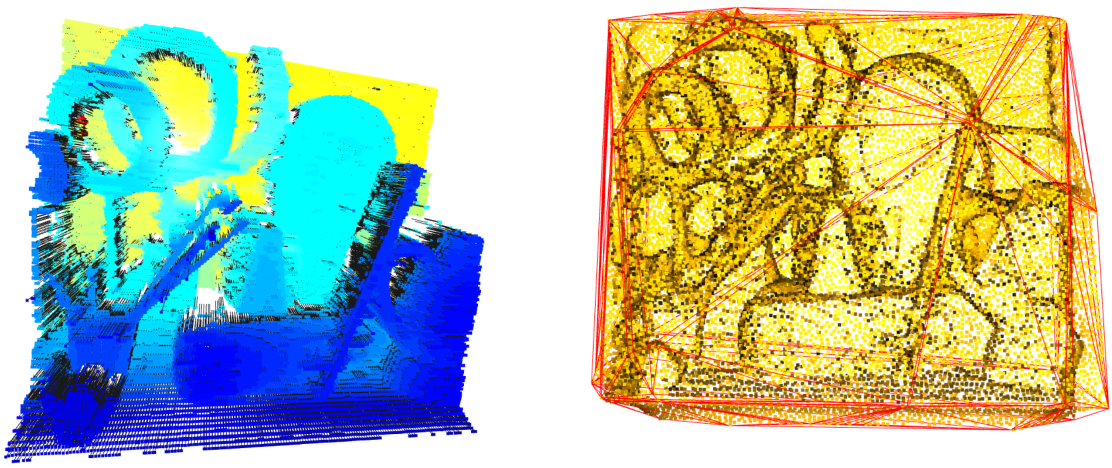
In this subtask, I've converted generated disparity map to 3D object with the help of Open3D(libraray for 3D data processing). The disparity map used for conversion was generated by dynamic programming approach with window size 3 for "Art" images from middlebury data-set.
Figure 1.3 represents the generated point cloud from different viewpoint.

Figure 1.4a illustrates the point cloud after assigning surface normal to each point. Finally, Figure 1.4b presents 3D triangulated surface.



(a) (b) (c)

Figure 1.3: Point cloud from different viewpoint



(a) Oriented point cloud (b) Triangulated Surface

Figure 1.4: Point cloud after assigning surface normal and triangulation

1.3 Subtask-3: Evaluation

For evaluating the algorithms I've taken 6 pairs of stereo images and respective ground truth disparity maps from Middlebury datasets.

1.3.1 Processing Time

Figure 1.5 presents the processing time for different window size for all the three algorithms across all the datasets. It can be noted that the dynamic approach is executed without parallalizing the code. In General, dynamic programming approach takes the most time to generate the disparity map. Whereas, the openCV implementation of StereoBM is very fast and relatively constant for all window size. For naive and dynamic approach the processing time increases rapidly with the increase of window size.

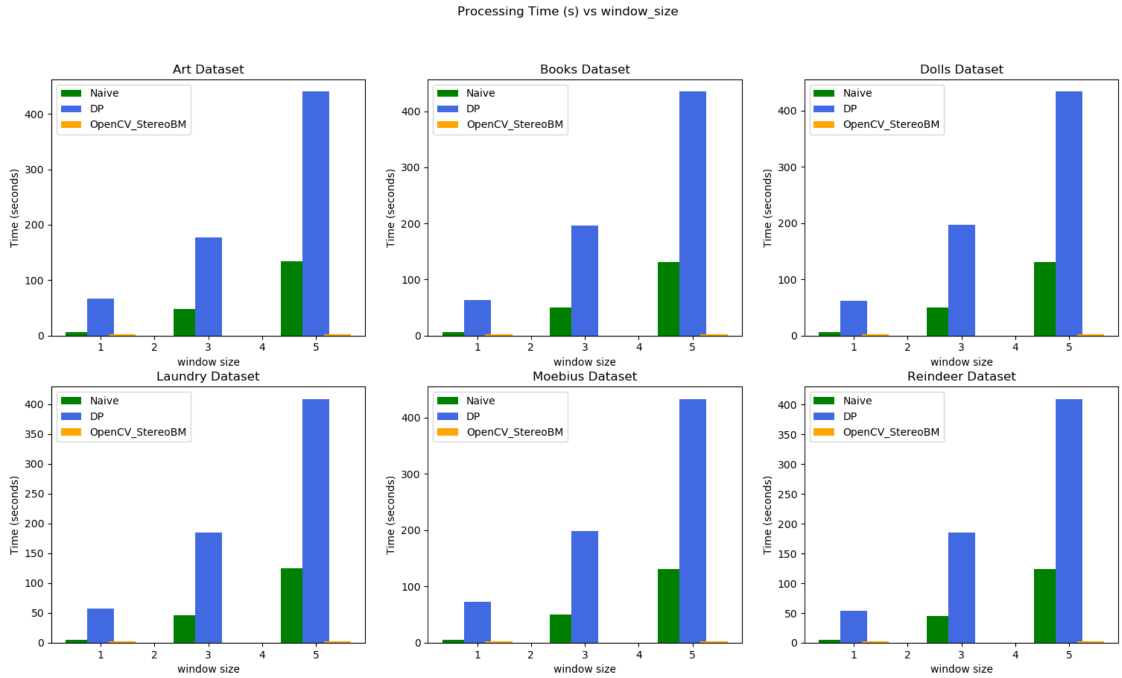


Figure 1.5: Processing time vs window size for different algorithms and all datasets.

1.3.2 Quality Comparison with Ground Truth Disparity

In this section, I've compared the quality of estimated disparity maps by dynamic approach, to ground truth disparity maps with three different matrices.

Sum of Squared Differences (SSD)

Figure 1.6 represents the results for SSD, where it can be seen that the error increases with the increase of window size. The dynamic programming approach has relatively less error as the lambda was tuned with the matrices and those values are used to generate the disparity maps.

Structural Similarity Index Measure (SSIM)

SSIM gives output as 1 if the images are perfectly matched. Figure 1.7 represents SSIM value for different window size for 1 pair from each dataset and for all algorithms. It can be noted that the dynamic programming approach has the best matches and with increasing window sizes the match result is decreasing. For naive the match value increases with increasing window size on the other hand for openCV implementation the window size doesn't affect much.

Normalized Cross Corelation (NCC)

Similarly, figure 1.8 illustrates the normalized cross correlation results.

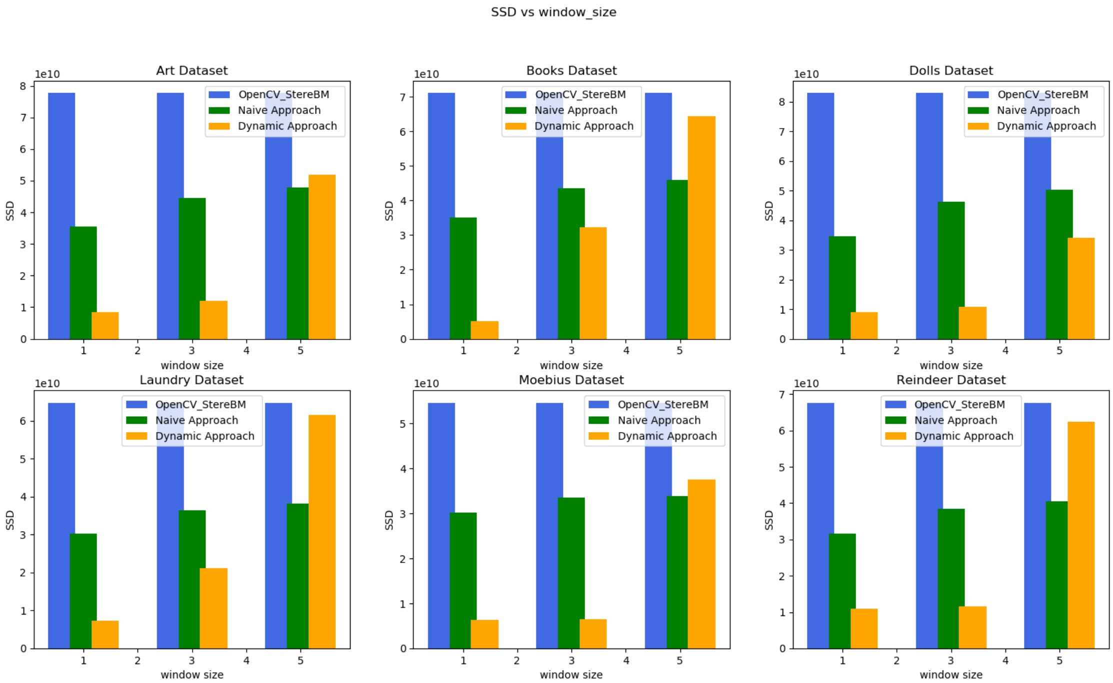


Figure 1.6: SSD vs window size for different algorithms and all datasets.

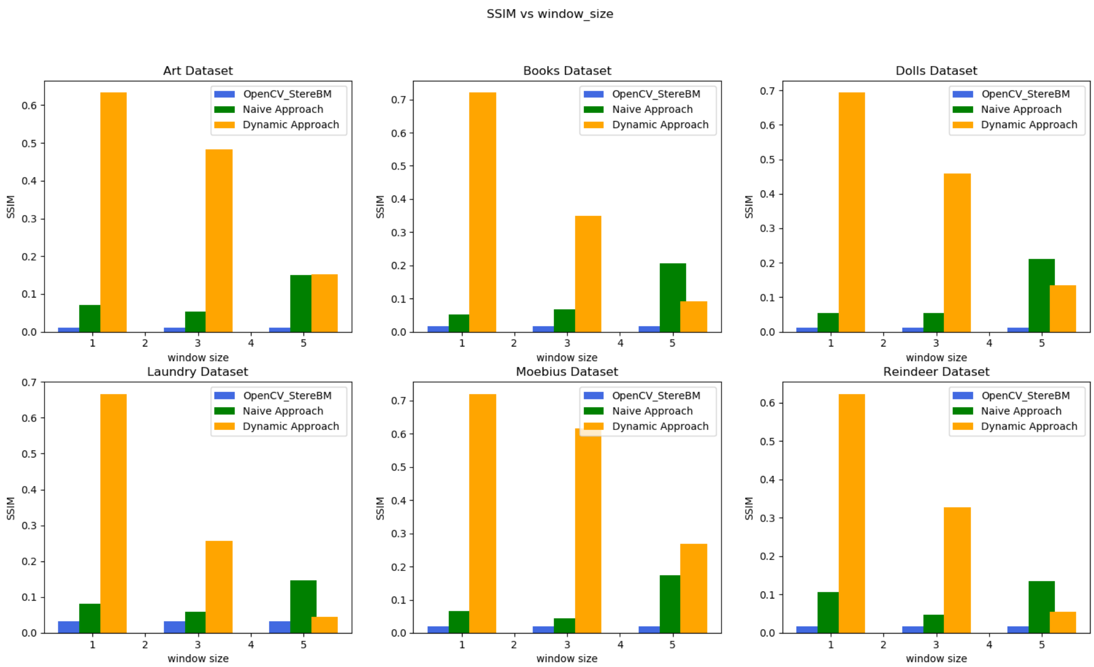


Figure 1.7: SSIM vs window size for different algorithms and all datasets.

1.3.3 Difference Image

Figure 1.9 represents the difference image between estimated disparity with dynamic approach and ground truth disparity for two different datasets. In the difference image the blue area represents the missing part of right image with respect to the left image.

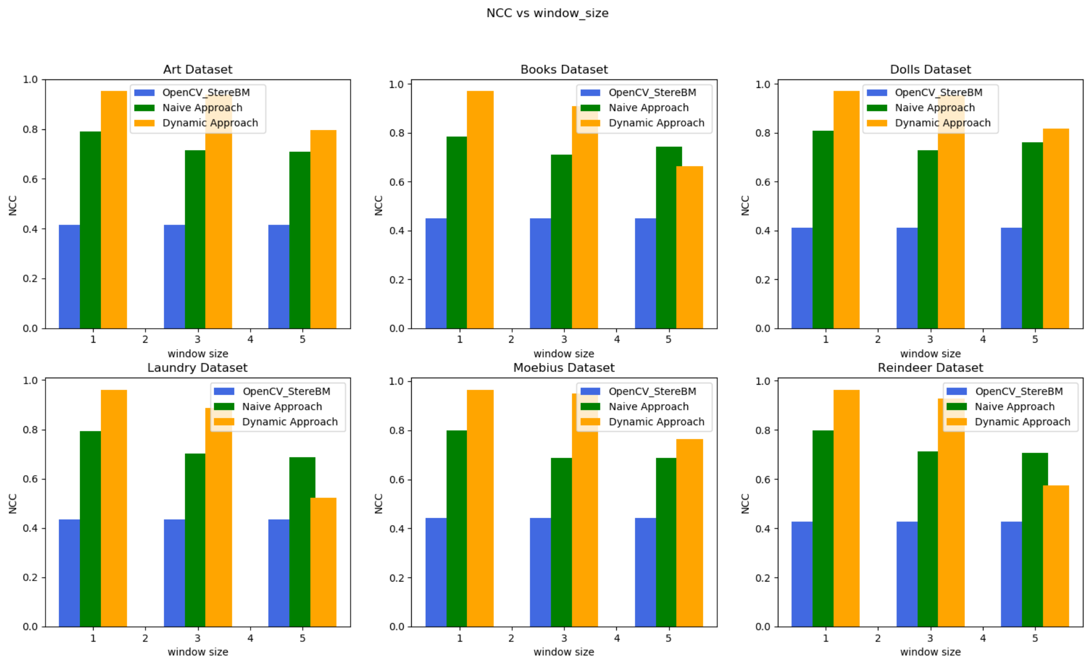
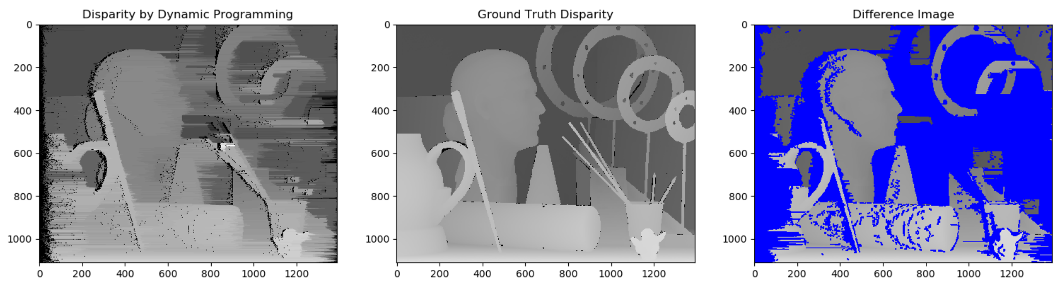
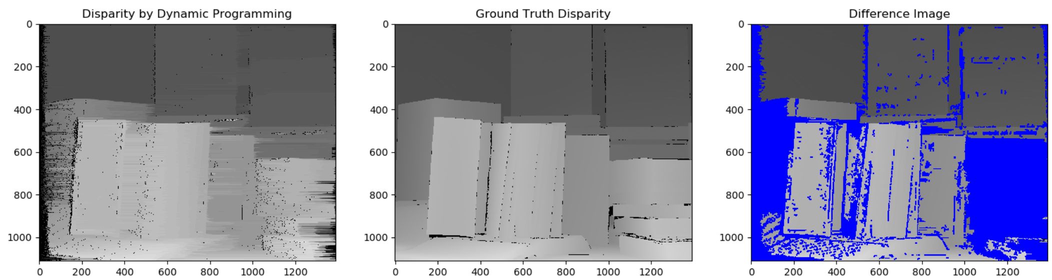


Figure 1.8: NCC vs window size for different algorithms and all datasets.



(a) For "Art" Dataset

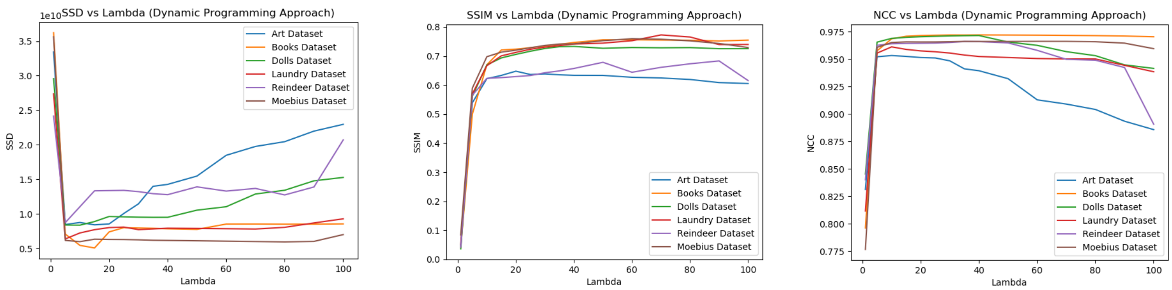


(b) For "Books" Dataset

Figure 1.9: Difference image between estimated and ground truth disparity.

1.3.4 Lambda (Occlusion Weight) Optimization

In order to find the optimal lambda across different datasets, I've chosen 15 different lambda values (1, 5, 10, 15, 20, 25, 30, 35, 40, 50, 60, 70, 80, 90, 100) and ran all the datasets with these values for window size 1. Then I've compared the generated disparity



(a) SSD Vs Lambda (b) SSIM Vs Lambda (c) NCC Vs Lambda

Figure 1.10: Disparity map generated with dynamic programming for different window size.

Table 1.1: Optimal lambda values for each metrics across each data-set

Metrics	**Dataset-1 (Art)**	**Dataset-2 (Books)**	**Dataset-3 (Dolls)**	**Dataset-4 (Laundry)**	**Dataset-5 (Reindeer)**	**Dataset-6 (Moebius)**	**Average lambda for each metrics across all datasets**
SSD	15	15	10	5	5	80	21
SSIM	20	60	40	1	1	1	20
NCC	10	40	40	10	35	60	32

map with the ground truth disparity for all the datasets with three different metrics (SSD, SSIM, NCC) and plotted them.

The plots are depicted by figure 1.10. The optimal lambda found for each metric across each data-set are shown in table 1.1. From figure 1.10 and table 1.1 it can be easily shown that the average lambda value lies between 20 to 30. For SSD, the average optimal lambda across all datasets is 21, while for SSIM and NCC it is 20 and 32. If we take further average across all metrics, the optimal lambda could be 24.