IP&PR: Graphical models for Stereo

Braulio Sespede, Sara Naghedi

January 29, 2020

1 Introduction

In this report we will explain our implementation of graphical models in the context of stereo matching. More specifically, we will comment on how the cost-volume was computed using different metrics, and then on how the cost-volume was aggregated using semi-global matching (SGM).

A cost-volume represents the probability of each pixel of an image taking a certain disparity. It is usually represented a 3D volume, where the x and y axis represent the axes of the image and the z-axis is an array of disparities with their corresponding relative likelihood.

To determine such likelihood (or cost, as named in the context of graphical models), we need to find similar pixels in the corresponding stereo pair with a certain confidence. To do this efficiently, the epipolar constraint is exploited and the correspondence problem is reduced to a 1D-problem. To do this with as little error as possible, we exploit the locality principle and compare patches along the epipolar line. The final goal is to measure the horizontal distance (also known as disparity) between corresponding pixels in the epipolar line. To measure the similarity between patches for a given disparity, we use metrics such as sum of absolute differences (SAD), sum of squared differences (SSD), and normalized cross-correlation (NCC).

Since this approach only compares patches "locally" (they don't really see the whole image), the end result of a winner-takes-all approach is usually noisy (as seen in Figure 1).

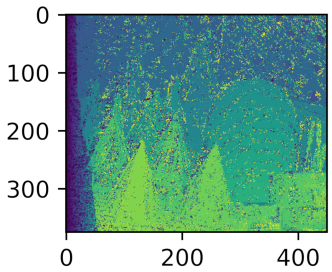

Figure 1: Example of noisy disparity map resulting from naive winner-takes-all on cost-volume built with NCC

To compensate for this, semi-global approaches exploit the benefits of both global and local stereo matching approaches (i.e. high-quality results with efficient computation times). One of such algorithms is the algorithm proposed in [1]. In this paper, the authors propose a graph-like representation where each vertex of the graph contains a list of costs, and the edges represent the likelihood of a neighboring pixel taking a certain label and then optimize along 1D chains. This energy minimization based approach minimizes the cost from neighboring pixels and then uses this information along with the information of the local cost slice to produce more "globally" accurate results along the chain. To reduce streak-like artifacts produced by optimizing along individual

scanlines (shown in Figure 2, the author proposes using the information provided by chains in several directions.

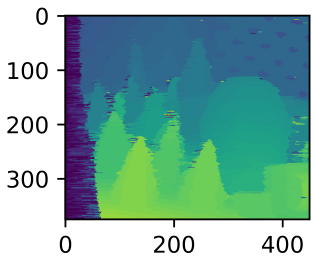

Figure 2: Example of a "streaky" disparity map due to single direction SGM aggregation.

2 Implementation

When it comes to the implementation of the construction of the cost volume, we used *view_as_windows* function in order to simplify and vectorize the implementation. The basic idea is to, for each pixel, move within a window of the same size as the maximum disparity and compute for each possible disparity the similarity given a certain metric. After the similarity for each possible disparity is computed, its stored in the 3D volume of size *(height, width, max_disparities)*. As shown in [2], taking the disparity of minimum cost results in an extremely noisy disparity map.

In order to implement the SGM method we first pre-compute the pairwise costs and broadcast it to the rest of the image (as it is the same for every pixel in our unweighted implementation). Once this has been pre-computed, we proceed to pass the messages in the 4 directions (left-to-right, right-to-left, up-to-down, down-to-up). To do this, we reuse the implementation of *dp_chain* that simply passes the message in a left-to-right fashion, and take special care to swap and flip axes of the cost-volume in a correct manner for each possible direction beforehand. The *dp_chain* method simply iterates from left-to-right passing the message as described in the assignment sheet.

Before describing the evaluation procedure and the corresponding results we will mentioned the supporting hardware and software used to develop and evaluate our implementation:

- **CPU:** Intel i7 8550U @ 1.8 GHz x8
- **OS:** Ubuntu 19.10
- **Language:** Python 3.7 (Conda)

Beyond specific implementation details, we will now mentioned the upcoming evaluation procedure. First, we will evaluate how the different similarity metrics affect the disparity maps when using a simple winner-takes-all approach. Moreover, we will analyze the effect of the patch size during disparity search. Then, we will evaluate the effect of the pairwise costs in the aggregation process done by th SGM algorithm. Before measuring the accuracy we normalized both the ground truth and our disparity maps to the range [0, 1] and then multiplied it by the maximum disparity. Afterwards, we counted the number of unoccluded pixels that were below a certain threshold of disparity with respect to the ground truth disparities.

3 Results

As previously mentioned, we will first take a look at the effect of the patch radius and disparity threshold.

- **Thresholds:** [1, 2, 3]
- **Radii:** [1, 2, 5]

The table in the Appendix shows all the accuracy computations for both the aggregated disparity maps, as well as the naively computed ones. Qualitatively, from Figure 3 we can first notice a large amount of salt and pepper noise in the unaggregated disparity maps (specially in the NCC one), which makes sense considering the disparity slices are not locally smooth as shown in [2]. We also notice that going from SAD, to SSD, and then to NCC, further increase the quality of the accuracy of the disparity maps. Clearly, the least noisy disparity maps is the one provided by the SGM algorithm using the NCC similarity metric. To our surprise, the accuracy is lower on the disparity maps using the SGM algorithm (as can be seen in the table at the Apppendix). This is particularly strange, considering the disparity maps look substantially less noisy. Possible hypotheses for these numbers, might be that the normalization of the ground-truth disparity map might be incorrect, or that the SGM algorithm might have oversmoothed, thus resulting in worse numerical results.

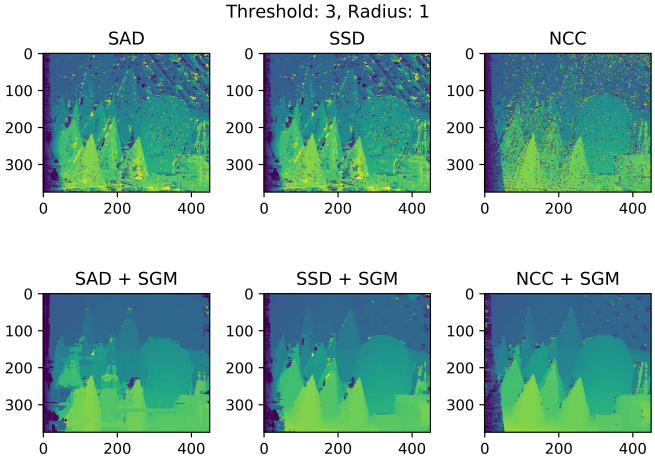

Figure 3: Comparison of different metrics, without and with aggregation.

When it comes to the radius, we notice the famous "edge fattening" effect on the disparity maps, as seen in Figure 4. Even though the result is usually less noisy as the radius increases, accuracy is lost in exchange. As the threshold increases, so does the accuracy, as it allows previously uncounted pixels to be part of the accuracy metric.

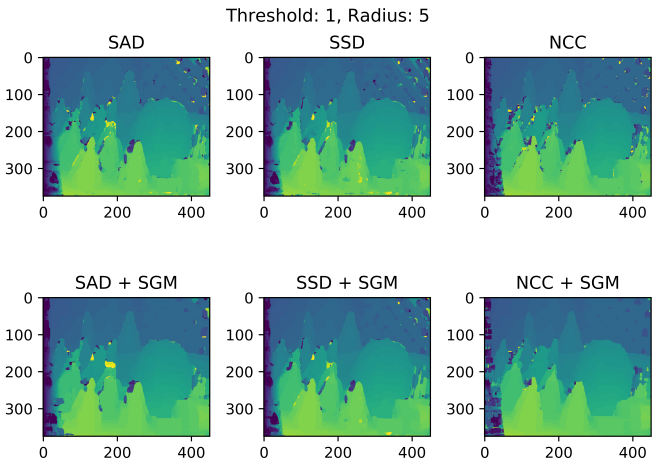

Figure 4: Increasing values of L1 and L2 respectively, and their impact on the final disparity maps. They were computed using NCC and radius 1.

Up ahead, we will briefly show the impact the hyper-parameters of the pairwise cost had on the final disparity map. As shown in the Figure 5, we can see that as both L1 and L2 increase, the noise is reduced, but at the same time details are lost and are "hatch" like artifacts appear.

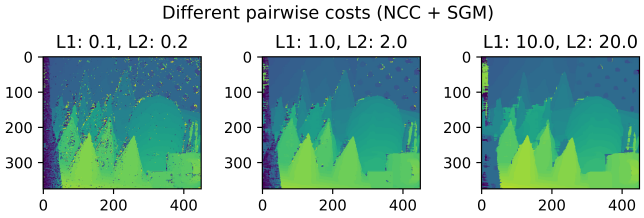

Figure 5: Increasing values of L1 and L2 respectively, and their impact on the final disparity maps. They were computed using NCC and radius 1.

4 Conclusion

To conclude this report, we will mentioned a few takeaways from this experience:

- A large radius during cost-volume computation produces edge fattening effects but reduces noise.
- The more directions we use in the SGM algorithm, the less streak-like artifacts appear, but computation time increases proportionally.
- The best similarity metric is NCC combined with SGM, as it doesn't require a large patch size to remove noise.
- The L1 and L2 parameters from the pairwise cost control the influence of neighbors: too little keeps the disparity maps noisy, too much makes "hatch" like artifacts appear and oversmoothing might happen.

References

[1] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30(2):328–341, 2007.

[2] Asmaa Hosni, Christoph Rhemann, Michael Bleyer, Carsten Rother, and Margrit Gelautz. Fast cost-volume filtering for visual correspondence and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(2):504–511, 2012.

Metric	SGM	Threshold	Radius	Accuracy
NCC	no	3	1	0.02287286522
SSD	no	3	1	0.01918346928
SAD	no	3	1	0.01657101566
NCC	no	2	1	0.01484790795
SSD	no	2	1	0.01286772369
SAD	no	2	1	0.01108208385
SSD	no	3	2	0.008282033823
NCC	no	1	1	0.007330155774
SAD	no	3	2	0.006975807012
NCC	no	3	2	0.006600614205
SSD	no	1	1	0.00657977016
NCC	no	3	5	0.006051721023
SAD	no	1	1	0.005620944096
SSD	no	2	2	0.005579256007
SAD	no	2	2	0.004620429943
NCC	no	2	2	0.00439114545
NCC	no	2	5	0.004022900657
SAD	no	3	5	0.003091866654
SSD	no	1	2	0.002994594444
SAD	yes	3	5	0.002591609577
SSD	no	3	5	0.002480441338
SAD	yes	3	1	0.002369273099
SAD	no	2	5	0.002362325084
NCC	no	1	2	0.00220252074
SSD	yes	3	2	0.002181676695
NCC	no	1	5	0.002139988605
NCC	yes	3	5	0.002139988605
SAD	no	1	2	0.002126092575
SSD	yes	3	5	0.002112196545
SAD	yes	2	5	0.001917652127
SSD	no	2	5	0.001743951753
NCC	yes	3	2	0.001730055723
SAD	yes	3	2	0.001695315648
SSD	yes	2	5	0.001653627559
SSD	yes	3	1	0.001604991454
NCC	yes	2	5	0.001507719245
SAD	yes	2	1	0.001438239095
NCC	yes	3	1	0.00143129108
SSD	yes	2	2	0.00138960299
SAD	yes	2	2	0.001028306213
NCC	yes	2	2	0.0009796701083
SAD	no	1	5	0.0009657740783
SSD	no	1	5	0.0008893459139
SSD	yes	1	5	0.0008823978989
SSD	yes	2	1	0.0008407098092
NCC	yes	1	5	0.0008337617943
NCC	yes	2	1	0.0007851256896
SAD	yes	1	1	0.0006322693606
SSD	yes	1	2	0.000576685241
SAD	yes	1	5	0.0005697372261
NCC	yes	1	2	0.0005558411962
SSD	yes	1	1	0.0004238289121
NCC	yes	1	1	0.0004099328822
SAD	yes	1	2	0.0003404527327