EFFICIENT AGGREGATION VIA ITERATIVE BLOCK-BASED ADAPTING SUPPORT-WEIGHTS

Leonardo De-Maeztu[1]*, Stefano Mattoccia*[2]*, Arantxa Villanueva*[1] *and Rafael Cabeza*[1]

[1]Department of Electrical and Electronic Engineering, Public University of Navarre, Pamplona, Spain
[2]Department of Electronics, Computer Sciences and Systems, University of Bologna, Bologna, Italy

ABSTRACT

Local stereo matching algorithms based on adapting-weights aggregation produce excellent results compared to other local methods. In particular, they produce more accurate results near disparity edges. This improvement is obtained thanks to the fact that the support for each pixel is accurately determined based on information such as colour or spatial distance. However, the computation of the support for each pixel results in computationally complex algorithms, especially when using large aggregation windows. Iterative aggregation schemes are a potential alternative to using large windows. In this paper we propose a novel iterative approach for adapting-weights aggregation which produces better results and outperforms most previous adapting-weights methods.

Index Terms— Stereo, 3D/stereo scene analysis.

1. INTRODUCTION

Dense stereo matching is one of the most important problems in the field of computer vision. According to [9], stereo methods can be classified into two types of approaches (local and global) according to the strategies used for estimation. In general, local approaches are much faster and more compatible to a practical implementation than global approaches producing less accurate results. Nevertheless, recent local algorithms based on the adapting-weights strategy [13, 5] produce results comparable to algorithms based on global optimization techniques. However, the complexity of adapting-weights strategies is large.

In recent years, techniques aimed at reducing the computational complexity of local stereo matching algorithms based on the adapting-weights strategy have been proposed. Unfortunately, in some cases, the reduced complexity is also accompanied by a reduced accuracy of the algorithm [3, 8] compared to [13, 5]. On the other hand, in some cases the reduction of complexity does not affect the accuracy of the results [6].

Another possible alternative to speedup adapting-weights is to reduce the size of the aggregation window. To compensate for the loss in accuracy caused by the use of smaller windows, the aggregation stage is iterated several times. In [12] 5×5 aggregation windows are used. Because such small windows aggregate costs around a small neighborhood, a large number of iterations is needed to obtain accurate disparity maps.

In this paper we propose to use a block-based iterative strategy for costs aggregation instead of a pixel-based one. The implementation of this type of technique in an iterative framework allows to use larger windows with similar complexity. Thanks to using larger windows for costs aggregation, the convergence of the results is much faster and less iterations are needed.

The rest of this paper is organized as follows. In Section 2, we review state-of-the-art adapting-weights stereo matching algorithms. In Section 3 we present our iterative block-based approach for costs aggregation. We report experimental results in Section 4. Finally, we draw conclusions in Section 5.

2. RELATED WORK

The field of variable costs aggregation has been recently reviewed in [3, 11]. The most remarkable step in this field in terms of accuracy improvement was proposed in [13]. The adapting-weights technique described in [13] enabled a huge reduction of the accuracy gap between global and local stereo matching algorithms. Other techniques (i.e., geodesic support-weights aggregation [5]) have proposed further accuracy improvements relying on a modified adapting-weights aggregation methodology. Both methods [13] and [5] aggregate costs using fixed-size square windows, with a different weight for each pixel.

The pixel-based matching cost measure used by [13] is the truncated absolute difference (TAD). Considering a pixel q in the reference image and the corresponding pixel $\overline{q}_d$ in the target image for a disparity d, TAD can be expressed as

$$e(q, \overline{q}_d) = \min\{ \sum_{c \epsilon \{r,g,b\}} |I_c(q) - I_c(\overline{q}_d)|, T\}, \qquad (1)$$

The work described in this study was supported by the Spanish Ministry of Science and Innovation with a FPU grant (AP2007-02468).

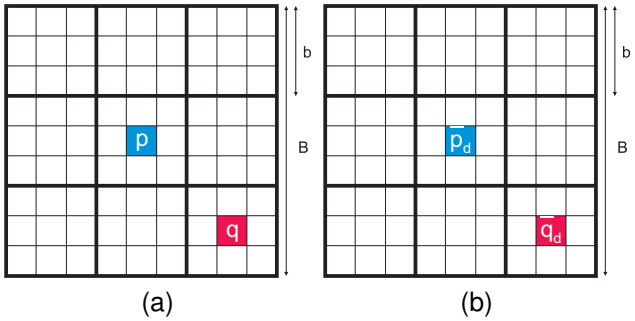


Fig. 1. Pixel notation inside the correlation windows in a block-based adapting-weights aggregation strategy: (a) in the reference image (N_p); (b) in the target image ($N_{\overline{p}_d}$).

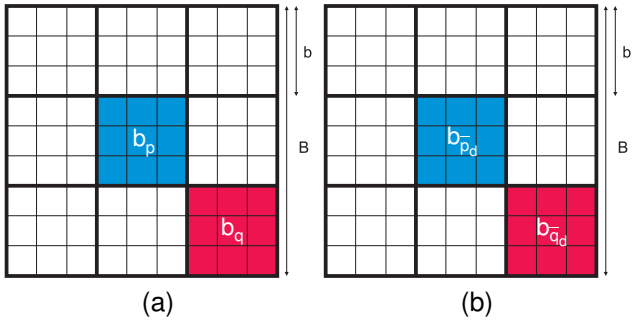


Fig. 2. Block notation inside the correlation windows in a block-based adapting-weights aggregation strategy: (a) in the reference image (N_p); (b) in the target image ($N_{\overline{p}_d}$).

where I_c is the intensity of the color band c in the RGB color space, and T is the truncation value used to limit the influence of outliers.

Hosni et al. [5] proposed a new method for computing support weights, specifically, using the geodesic distance between two pixels. The pixel-based matching cost measure used in [5] is Hierarchical Mutual Information (HMI). The implementation of this matching measure is described in [4].

The two previously described algorithms use the same optimization technique, Winner-Takes-All (WTA) [13, 5].

The authors of the original adapting-weights algorithm proposed an iterative implementation of adapting-weights [12]. The window size used in this algorithm is smaller than those used in adapting-weights algorithms. The loss in performance caused by the use of small windows was compensated by applying several iterations of the adapting-weights aggregation step. However, a large number of iterations is needed for the algorithm to converge, so it is not especially faster than the non-iterative version of adapting-weights.

One of the main disadvantages of adapting-weights aggregation is that it is computationally expensive. Several authors have proposed faster local methods inspired by the adapting-weights algorithm. Gong et al. [3] proposed a two-pass 1D algorithm instead of using a 2D square window for aggregation. Richardt et al. [8] recently proposed a new real-time local stereo algorithm based on the adapting-weights method. They used a bilateral grid for costs aggregation, achieving a 200× speedup over the original adapting-weights algorithm. However, the results of both algorithms [3, 8] are less accurate than those of the adapting-weights algorithm [13].

Fast bilateral stereo [6] is another stereo matching technique inspired by the pixel-based adapting-weights aggregation method. Execution time is accelerated aggregating costs on a block basis instead of on a pixel basis. This method computes approximated weights for reference and target images on a block basis assigning to each point within a block a single value. It is worth to note that all block-level computations can be efficiently implemented by means of incremental calculation schemes such as box-filtering [7] or integral images [1].

The block-based adapting-weights strategy has two interesting properties. First, it is faster than pixel-based adapting-weights stereo matching. Moreover, when it is implemented with $b = 3$, it is not only around one order of magnitude faster than pixel-based adapting-weights, but it produces equivalent results (see [6]). Second, the block-based adapting-weights strategy adds flexibility to the adapting-weights method. By varying b, accuracy can be traded for computational complexity. In fact, block-based adapting-weights links pixel-based adapting-weights [13] ($b = 1$) and fixed-window [9] (one block of the same size of the aggregation window).

3. PROPOSED METHOD

Given two pixels p and $\overline{p}_d$ for which a correspondence has to be evaluated, and the associated supports N_p and $N_{\overline{p}_d}$ both of size $B \times B$ we partition the two supports in $\frac{B}{b} \times \frac{B}{b}$ blocks of size $b \times b$ as shown in Figure 1. As it is described in Figure 2, we refer to the block for which the central pixel is p as b_p. For each block (b_q and $b_{\overline{q}_d}$) of the two supports N_p and $N_{\overline{p}_d}$ we independently assign two weights according to

$$w(b_p, b_q) = \exp\left(-\left(\frac{\Delta c_{pq}}{\gamma_c} + \frac{\Delta g_{pq}}{\gamma_p}\right)\right), \qquad (2)$$

where Δc_{pq} is the Euclidean distance between the average values in the RGB color space of the pixels inside the block, and Δg_{pq} is the spatial Euclidean distance that separates the central points of the two blocks in the image. Two constants, γ_c and γ_p, are included to modulate the relative importance of each of the aforementioned parameters.

Once we obtain the block-based weights, we combine the weights to obtain a symmetric block-based weighted support so that the total cost $E(p, \overline{p}_d)$ for the correspondence associated to the pixels $(p, \overline{p}_d)$ is calculated by summing all the weighted pointwise costs average inside each block $e(b_q, b_{\overline{q}_d})$ and normalized by the summed weights:

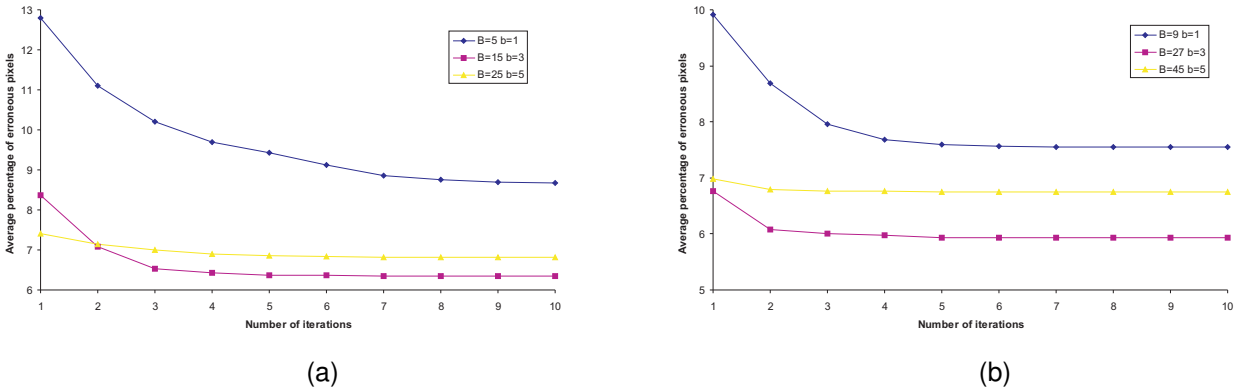


Fig. 3. Comparison of iterative block-based adapting-weights aggregation of algorithms with different block size: (a) 5×5 blocks; (b) 9×9 blocks.

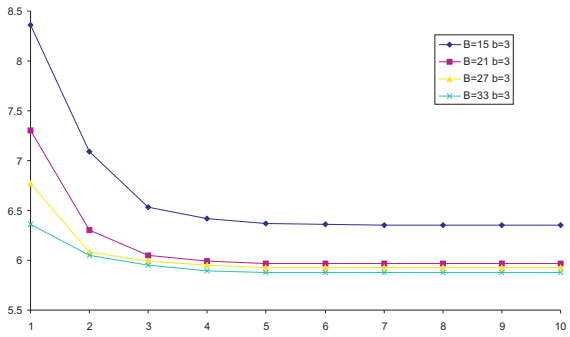


Fig. 4. Comparison of iterative block-based adapting-weights aggregation of algorithms with $b = 3$.

$$E(p,\overline{p}_d) = \frac{\sum_{b_q \epsilon N_p, b_{\overline{q}_d} \epsilon N_{\overline{p}_d}} w(b_p,b_q) w(b_{\overline{p}_d}, b_{\overline{q}_d}) e(b_q, b_{\overline{q}_d})}{\sum_{b_q \epsilon N_p, b_{\overline{q}_d} \epsilon N_{\overline{p}_d}} w(b_p,b_q) w(b_{\overline{p}_d}, b_{\overline{q}_d})}. \quad (3)$$

We use the same implementation of HMI proposed in [4] for computing the pixel-wise matching cost. We refer the reader to this publication for details. Compared to pixel-based aggregation, the number of calculations is intrinsically reduced by a factor $b \times b$. The aggregation process is repeated a certain amount of iterations to obtain the aggregated costs. Finally, WTA is performed to obtain the final disparity map.

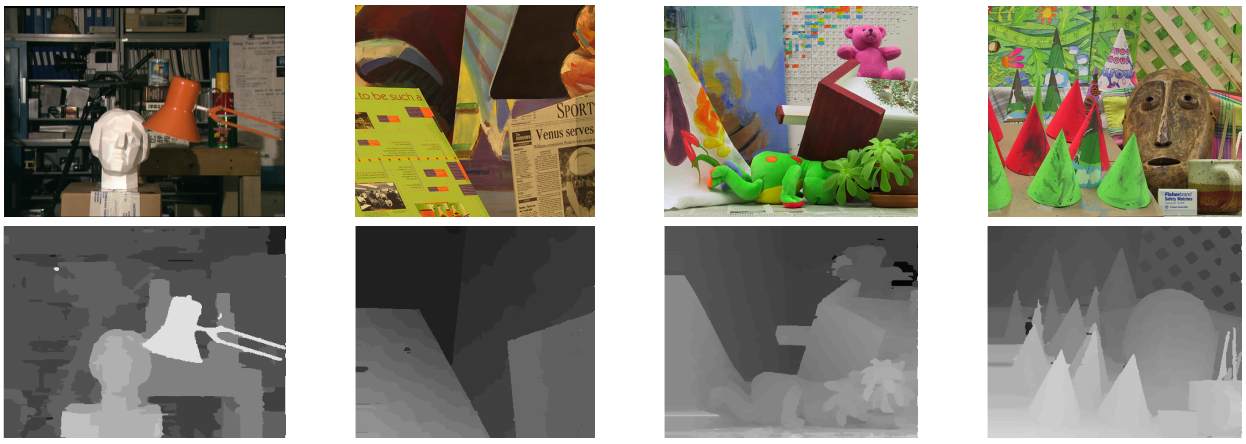
After two disparity maps are obtained (one corresponding to the left image and one corresponding to the right image of the stereo pair), 3×3 median filtering is applied to both maps and their results are cross-checked to detect occlusions. Invalidated regions are filled with the content of their first neighbor to the left or to the right depending on the nature of the occlusion. Finally, 3×3 median filtering is again applied to the final disparity map.

4. EXPERIMENTAL RESULTS

We evaluated the performance of the proposed stereo matching algorithm using the Middlebury stereo pairs with ground truth [9, 10]. In Figure 3, the average percentage of erroneous pixels for the four Middlebury stereo pairs is represented versus the number of iterations. The algorithms compared in each graph have a similar complexity (i.e., the algorithms compared in each graph use the same number of blocks, each with a different block size). From the results it is clear that the block-based approach ($b > 1$) is more appropriate for this iterative framework than the pixel-based approach ($b = 1$). Not only the block-based approach converges more rapidly than the pixel-based approach, but it also produces better results.

According to Figure 3, $b = 3$ seems to be the optimal value for this type of solution, because it combines a rapid convergence and a lower percentage of erroneous pixels than other b values. In Figure 4 the average percentage of erroneous pixels is represented versus the number of iterations for different $b = 3$ configurations. A window size of $B = 21$ pixels iterated 3 times seems a good trade off between accuracy and complexity. Further increases in the window size or the number of iterations produce negligible improvements in the accuracy of the resulting disparity maps.

Figure 5 contains the four aforementioned stereo pairs used for stereo algorithms benchmarking and the results produced by our algorithm with $B = 21$, $b = 3$, 3 iterations, $\gamma_c = 9$ and $\gamma_p = 12.5$ and $\sigma = 1$ for HMI. In Table 1 the accuracy and execution time on the Teddy stereo pair of the $B = 21$, $b = 3$ implementation of the proposed algorithm is compared to other state-of-the-art stereo matching algorithms based on adapting-weights. As it was mentioned in the previous section, the disparity maps of our method are refined using a post-processing step that includes a 3×3 median filter that helps to remove salt and pepper noise for a fair com-



(a) (b) (c) (d)

Fig. 5. Left images of each of the Middlebury datasets (upper row), and disparity maps computed using our method (lower row). (a) "Tsukuba" images. (b) "Venus" images. (c) "Teddy" images. (d) "Cones" images.

Table 1. Performance comparison of state-of-the-art adapting-weights algorithms.

Algorithm	Tsukuba			Venus			Teddy			Cones			Average percent of bad pixels	Comput time (sec)
	nocc	all	disc	nocc	all	disc	nocc	all	disc	nocc	all	disc		
GeoSup [5]	1.45	1.83	7.71	0.14	0.26	1.90	6.88	13.2	16.1	2.94	8.89	8.32	5.80	610
Our method	**1.78**	**2.10**	**7.57**	**0.31**	**0.50**	**2.17**	**7.94**	**12.8**	**17.1**	**3.07**	**8.73**	**8.46**	**6.05**	**18.2**
AdaptWeight [13]	1.38	1.85	6.90	0.71	1.19	6.13	7.88	13.3	18.6	3.97	9.79	8.26	6.67	175
AdaptDiff [12]	1.14	1.93	6.12	0.73	1.21	3.81	8.02	14.2	21.97	4.12	10.58	7.75	6.80	1125
FastBilateral [6]	2.38	2.80	10.4	0.34	0.92	4.55	9.83	15.3	20.3	3.10	9.31	8.59	7.31	25.4

parison with the other algorithms. However, big erroneous patches caused by occlusions or noise (e.g. white patch in Tsukuba or black patches in Cones) cannot be removed using only median filtering. In fact, the GeoSup algorithm implements a more complex and accurate post-processing technique to refine the produced disparity maps (weighted median filtering). AdaptDiff performs median filtering like our algorithm [12]. AdaptWeight [13] and FastBilateral [6] also perform some type of post-processing, even if it is not explicitly mentioned. This can be deduced because the results in the Middlebury ranking for FastBilateral are more accurate than those published in the paper (where it is stated that the results included in the paper are not post-processed). In the case of AdaptWeight, the results are commonly believed to be the result of a post-processing step, because several authors [6, 8, 2] have obtained less accurate disparity maps when implementing the algorithm proposed in [13].

The accuracy results in Table 1 are those reported by the authors in the Middlebury ranking [10] (our algorithm iFBS can also be found in the ranking), and the timing measurements were performed using our C implementation of their techniques on an Intel Core 2 6420 CPU using just one core. The proposed aggregation method can be fully parallelized using massively parallel architectures, however the implementation was done using only one core for a fair comparison with other algorithms. Our proposal is very close to the GeoSup algorithm [5] in terms of accuracy, being more than one order of magnitude faster. The rest of algorithms in the Table are outperformed by our proposal, even if they take more time to compute the disparity maps (our technique is the fastest method in the Table).

As shown in Table 1, the computation time of the proposed method for the Teddy stereo pair is 18.2 seconds. The most time consuming stages are those related to the proposed costs aggregation technique (1.2 seconds for the weights computation step and 12.5 seconds for the weighted costs aggregation step).

5. CONCLUSION

In this work we have presented a new iterative costs aggregation algorithm for stereo matching that produces more ac-

curate results than most state-of-the-art adapting-weights solutions. The combination of block-based aggregation using small windows along with an iterative framework improves the accuracy of previous non-iterative or iterative but pixel-based solutions. Moreover we have experimentally shown that the execution time in considerably reduced compared to other adapting-weights solutions.

6. REFERENCES

[1] F. Crow. Summed-area tables for texture mapping. *Proc. Special Interest Group on Graphics*, pages 217–212, 1984.

[2] L. De-Maeztu, A. Villanueva, and R. Cabeza. Stereo matching using gradient similarity and locally adaptive support-weight. *Pattern Recognition Letters*, 32(13):1643–1651, 2011.

[3] M. Gong, R. Yang, L. Wang, and M. Gong. A performance study on different cost aggregation approaches used in real-time stereo matching. *Intl J. Computer Vision*, 75(2):283–296, 2007.

[4] H. Hirschmüller. Stereo processing by semiglobal matching and mutual information. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 30(2):328–341, 2008.

[5] A. Hosni, M. Bleyer, M. Gelautz, and C. Rhemann. Local stereo matching using geodesic support weights. In *Proc. IEEE Intl Conf. on Image Processing*, pages 2093–2096, 2009.

[6] S. Mattoccia, S. Giardino, and A. Gambini. Accurate and efficient cost aggregation strategy for stereo correspondence based on approximated joint bilateral filtering. In *Asian Conf. Computer Vision*, pages II: 371–380, 2009.

[7] M. McDonnell. Box-filtering techniques. *Computer Graphics and Image Processing*, 17(1):65–70, 1981.

[8] C. Richardt, D. Orr, I. Davies, A. Criminisi, and N. Dodgson. Real-time spatiotemporal stereo matching using the dual-cross-bilateral grid. In *European Conf. Computer Vision*, volume 6313, pages 510–523. Springer Berling / Heidelberg, 2010.

[9] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Intl J. Computer Vision*, 47(1-3):7–42, 2002.

[10] D. Scharstein and R. Szeliski. *Middlebury stereo evaluation - version 2*, http://vision.middlebury.edu/stereo/eval.

[11] F. Tombari, S. Mattoccia, L. Di Stefano, and E. Addimanda. Classification and evaluation of cost aggregation methods for stereo correspondence. *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[12] K.-J. Yoon, Y. Jeong, and I.-S. Kweon. Support aggregation via non-linear diffusion with disparity-dependent support-weights for stereo matching. In *Asian Conf. on Computer Vision*, number 1, pages 25–36, 2009.

[13] K.-J. Yoon and I.-S. Kweon. Adaptive support-weight approach for correspondence search. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 28(4):650–656, 2006.