Near Real-Time Stereo Matching Using Geodesic Diffusion

Leonardo De-Maeztu, Arantxa Villanueva, *Member*, *IEEE*, and Rafael Cabeza

Abstract—Adaptive-weight algorithms currently represent the state of the art in local stereo matching. However, due to their computational requirements, these types of solutions are not suitable for real-time implementation. Here, we present a novel aggregation method inspired by the anisotropic diffusion technique used in image filtering. The proposed aggregation algorithm produces results similar to adaptive-weight solutions while reducing the computational requirements. Moreover, near real-time performance is demonstrated with a GPU implementation of the algorithm.

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

1 Introduction

Stereo matching is one of the most active areas in the field of computer vision. As a consequence, a variety of approaches have been proposed. An excellent survey of stereo matching algorithms can be found in [1]. According to this survey, most stereo algorithms can be categorized into two major classes: local methods and global methods. Local approaches use information within a finite region around the pixel for which the disparity is being computed. Global approaches incorporate explicit smoothness assumptions and determine all disparities simultaneously by applying energy minimization techniques.

When using local support regions, it is implicitly assumed that all pixels in the region are of the same depth (i.e., the fronto-parallel surfaces assumption). Multiple techniques have been proposed to choose the support for each pixel to satisfy this "same-disparity condition." Multiple-window methods select the best support window from a set of predefined windows [2], [3]. Variable-window methods do not select the best window from a predefined set, but rather compute an optimal support window for each pixel [4], [5], [6]. Adaptive-weight methods compute a support weight for each pixel inside a fixed-size square window [7], [8].

Adaptive-weight methods are the local algorithms yielding the best results, comparable to those generated using global optimization techniques. Here, the weights regulate each pixel's influence in the matching process. When two pixels have a high probability of belonging to the same object (in the fronto-parallel approach, this is equivalent to assuming that they have the same disparity), they are given high weights. However, the disparities are not known beforehand. In fact, the objective is to compute these disparities. Two of the most interesting adaptive-weight algorithms solve this problem in a similar manner, where the probability that two pixels belong to the same surface is computed according to color differences [7], [8] and the distance between them [7].

Despite the excellent results produced by adaptive-weight algorithms, there is a major problem with their implementation. In fact, pixel-wise support-weight computation is a highly time-consuming task. Anisotropic diffusion [9], a computer vision technique very similar to adaptive weighting but computationally less expensive, has been used in an attempt to reduce the computational requirements. Nevertheless, the implementation of this technique for stereo matching has so far resulted in low-quality disparity maps [10]. Min and Sohn [10] proposed an intermediate solution: They used adaptive-weight aggregation windows (smaller than those used in adaptive-weight (AW) solutions) and incorporated the iterative nature of diffusion algorithms. Yoon et al. [11] of the original adaptive-weight algorithm have also proposed a similar solution.

Here, we present a new local stereo matching algorithm inspired by anisotropic diffusion. More precisely, we propose several novel ideas to improve the effectiveness of diffusion-based stereo matching. These ideas are aimed at diffusing both matching costs and weights so that at each iteration, the relative importance of each diffusion path (i.e., the weight of this path) is known. Moreover, a coefficient is introduced to avoid loops in the diffusion process that would prevent an efficient diffusion of the matching costs. The proposed solution produces results similar to the best performing local algorithm (geodesic adaptive weighting), while considerably reducing the hardware requirements of this algorithm. Near real-time execution is demonstrated using a commercial graphics card.

The remainder of this paper is organized as follows: In Section 2, we describe previous state-of-the-art local stereo matching solutions. Our geodesic diffusion (GD) algorithm is presented in Section 3. Finally, we present the experimental results and conclusions in Sections 4 and 5, respectively.

2 Related Work

Our approach was inspired by recent advances in the area of variable cost aggregation supports that were recently evaluated in [12] and [13]. Among these solutions, the adaptive-weight algorithm proposed by Yoon and Kweon [7] deserves particular attention for its results. Hosni et al. [8] proposed a modification in the weight-computation stage of the original adaptive-weight algorithm [7], producing improved results. Both methods [7] and [8] aggregate costs using fixed-size square windows, with a different weight for each pixel. A support weight is computed for each pixel falling within the current correlation window N_p in the reference image and, correspondingly, in the correlation window $N_{\overline{p}_d}$ in the target image (Fig. 1). Let p and $\overline{p}_d$ be the respective central points of N_p and $N_{\overline{p}_d}$ for which a correspondence is being evaluated. Thus, the pointwise cost $e(q, \overline{q}_d)$ for any point q which falls within N_p corresponding to $\overline{q}_d$ belonging to $N_{\overline{p}_d}$ is weighted by a coefficient $w(p, q)$ and a coefficient $w(\overline{p}_d, \overline{q}_d)$, so that the total cost $E(p, \overline{p}_d)$ for correspondence $(p, \overline{p}_d)$ is calculated by summing all the weighted pointwise costs associated with the correlation windows and normalized by the summed weights

$$E(p, \overline{p}_d) = \frac{\sum_{q \epsilon N_p, \overline{q}_d \epsilon N_{\overline{p}_d}} w(p, q) w(\overline{p}_d, \overline{q}_d) e(q, \overline{q}_d)}{\sum_{q \epsilon N_p, \overline{q}_d \epsilon N_{\overline{p}_d}} w(p, q) w(\overline{p}_d, \overline{q}_d)}. \quad (1)$$

In [7], weights are computed using two values: the color difference and the distance between the support pixel and the central pixel of the window. The support weight is larger for pixels with colors similar to the central pixel and for pixels closer to the central pixel. The color difference is calculated as the euclidean distance between the values in the CIELab color space (Δc_{pq}), and the distance is the spatial euclidean distance that separates the two pixels in the image (Δg_{pq}). Two constants, γ_c and γ_p, are included to modulate the relative importance of each of the aforementioned

• *The authors are with the Department of Electrical and Electronic Engineering, Public University of Navarre, Arrosadia Campus, Pamplona 31006, Spain. E-mail: {leonardo.demaeztu, avilla, rcabeza}@unavarra.es.*

Manuscript received 3 Jan. 2011; revised 31 May 2011; accepted 18 Aug. 2011; published online 1 Oct. 2011.
Recommended for acceptance by A. Criminisi.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2011-01-0004.
Digital Object Identifier no. 10.1109/TPAMI.2011.192.

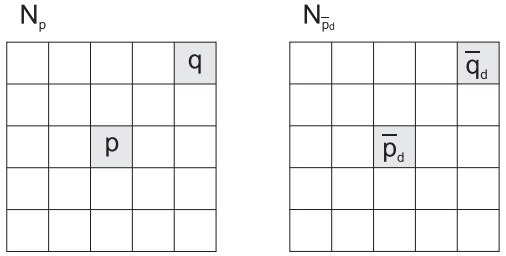

Fig. 1. Correlation windows. (a) In the reference image (N_p). (b) In the target image ($N_{\bar{p}_d}$).

parameters. The weight w, assigned to the cost of the pixel q when the disparity of the pixel p is computed, is expressed as

$$w(p,q) = \exp\left(-\left(\frac{\Delta c_{pq}}{\gamma_c} + \frac{\Delta g_{pq}}{\gamma_p}\right)\right). \quad (2)$$

The pixel-based matching cost measure used by [7] is the truncated absolute difference (TAD):

$$e(q,\bar{q}_d) = \min\left\{\sum_{c\epsilon\{r,g,b\}} |I_c(q) - I_c(\bar{q}_d)|, T\right\}, \quad (3)$$

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. [8] proposed a new method for computing support weights, specifically using the geodesic distance between two pixels. The geodesic distance $D(p,q)$ between two pixels p and q is defined as the shortest path that connects p with q in the color volume

$$D(p,q) = \min_{P\in\wp_{p,q}} d(P), \quad (4)$$

where $\wp_{p,q}$ denotes the set of all paths between p and q. A path P is defined as a sequence of spatially neighboring points in eight connectivity. The cost $d(P)$ of a path is computed as

$$d(P) = \sum_{i=2}^{n} d_C(p_i, p_{i-1}), \quad (5)$$

with $d_C(p_i, p_{i-1})$ being the euclidean distance in the RGB color space of pixels p_i and p_{i-1} (Fig. 2). According to this notation, we assume that $p = p_1$ and $q = p_n$. The function $w(p,q)$ transforms the cost of the minimum cost path into a weight with a γ parameter controlling this transformation:

$$w(p,q) = \exp\left(-\frac{D(p,q)}{\gamma}\right). \quad (6)$$

The pixel-based matching cost measure used in [8] is hierarchical mutual information (HMI). The implementation of this matching measure is described in [14], to which we refer the reader for details as this report is focused on aggregation techniques and a detailed description of HMI would unnecessarily lengthen our discussion.

The two previously described algorithms use the same optimization technique, winner-takes-all (WTA) [7], [8].

As highlighted in Section 1, adaptive-weight algorithms are not suitable for implementation on commercially available hardware. The implementation problems are related to the number of expensive exponential operations performed during the weight computation step, which is directly proportional to the square of the aggregation window size.

Anisotropic diffusion [9] is a computer vision technique similar to bilateral filtering. The bilateral filtering technique inspired the adaptive-weight method proposed by Yoon and Kweon [7]. The advantage of anisotropic diffusion is that, being a diffusion technique, only the comparison of each pixel with its immediate neighbors is necessary, thus saving computation time. Nevertheless, Min and Sohn [10] tested a direct implementation of anisotropic diffusion on stereo matching and the results were unsatisfactory in terms of accuracy. To take advantage of these lower requirements, Min and Sohn [10] and Yoon et al. [11] proposed an intermediate solution between adaptive-weights and anisotropic diffusion. The size of the windows used in these algorithms is smaller than those used in adaptive-weight algorithms. The loss in performance caused by the use of small windows was compensated for by applying several iterations of the adaptive-weight aggregation step.

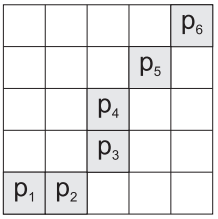

Fig. 2. One eight-connectivity path joining p_1 and p_6. The total cost of the path is computed by summing the pixel-to-pixel euclidean distance in the RGB color space.

Other diffusion techniques have been applied to stereo matching. Scharstein and Szeliski [15] tested various diffusion models. Most of the proposed methods rely on local diffusion (costs are diffused along constant-disparity planes). However, the results of these local diffusion solutions are not satisfactory near disparity edges. They also propose a Bayesian model for stereo matching in which information is also diffused between different disparity planes (i.e., a global solution).

Several stereo matching algorithms with smaller computational complexities have been proposed that try to simulate the behavior of the adaptive-weight algorithm [7] and the geodesic support-weight approach [8]. Wang et al. [16] proposed a two-pass aggregation scheme to approximate adaptive weights and enable a GPU implementation. Richardt et al. [17] recently proposed a new real-time local stereo algorithm that uses a bilateral grid for cost aggregation, achieving a 200 times improvement in speed over the original adaptive-weight algorithm. Essentially, adaptive-weight aggregation is not performed directly on the disparity space image (DSI) but on an alternative representation (the dual-cross-bilateral grid), which allows a faster execution. However, the results of both algorithms [16], [17] are less accurate than those of the adaptive-weight algorithm [7]. This is also the case for a faster version of the geodesic support-weight approach [8] proposed by Hosni et al. [18]; the simplified version is much faster but has reduced accuracy. Despite this general complexity-reduction trend, some authors have proposed algorithms based on adaptive weights but that produce more accurate results, even if the execution time of these solutions is higher than the computation time of the original adaptive-weight algorithm [7]. Tombari et al. [19] proposed the use of segmentation along with adaptive weights.

3 Geodesic Diffusion

Our algorithm was directly inspired by anisotropic diffusion [9], an iterative computer vision technique in which the intensity value, I_c, of each pixel p is updated according to the intensity value of its neighbors q_0, q_1, q_2, and q_3 (Fig. 3):

$$I_c^n(p) = I_c^{n-1}(p)\left(1 - \lambda\sum_{j=0}^{3} c(p,q_j)^{n-1}\right) + \lambda\sum_{j=0}^{3} c(p,q_j)^{n-1} I_c^{n-1}(q_j), \quad (7)$$

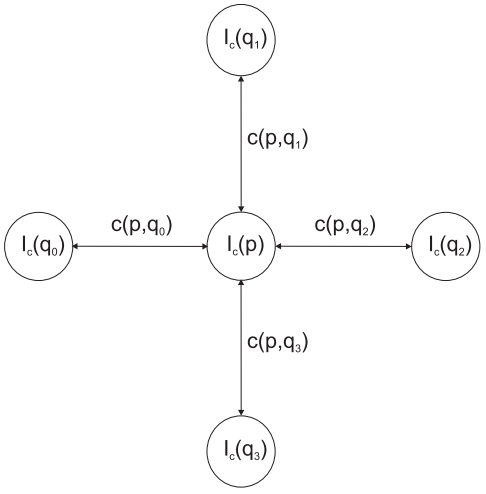

Fig. 3. A pixel p and the four direct neighbors q used for updating the intensity value according to the diffusion coefficients $c(p, q)$.

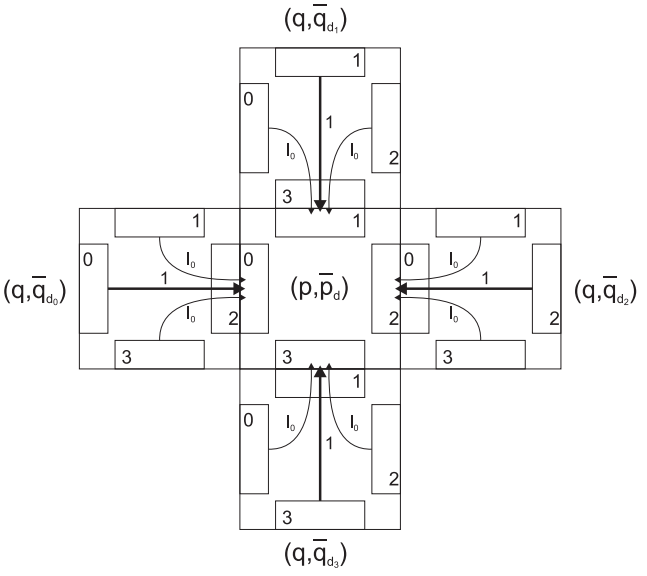

Fig. 4. Diffusion of costs (D_d) and weights (D_{dW}) between neighbors.

where $0 \leq \lambda \leq 0.25$ controls the influence of neighboring pixels and n is the iteration number. Different functions can be used to implement the diffusion coefficient $c(p, q)$. One possibility is the exponential function of the negative euclidean distance in the RGB or CIELab color space Δc_{pq} with a tuning constant γ_c, as used in adaptive-weight algorithms (2):

$$c(p, q) = \exp\left(-\frac{\Delta c_{pq}}{\gamma_c}\right). \tag{8}$$

As mentioned in Section 1, the direct implementation of anisotropic diffusion for stereo matching does not produce relevant results [10]. Nevertheless, using the idea of pixel-to-pixel diffusion according to color differences, we propose a new algorithm called geodesic diffusion based on three principles that improve the performance of diffusion to the level of state-of-the-art algorithms. First, costs and weights are diffused so that the importance of each cost value is known in each iteration. Second, in each iteration, the costs and weights at each pixel are accumulated. After the last iteration, all the support region information has been accumulated at each pixel. The final cost is obtained by dividing the accumulated cost by the accumulated weight. Third, to increase the efficiency of information diffusion and to avoid loops, turns in the direction of diffusion are penalized. These principles are formalized using equations given later in this section.

The proposed algorithm requires only the comparison of each pixel with its four direct neighbors, thus reducing the computational requirements of adaptive-weight solutions. Our algorithm also makes two contributions when compared to the methods proposed by Min and Sohn [10] and Yoon et al. [11]. First, the number of pixel comparisons is further reduced (9×9 or 5×5 windows are no longer used, being replaced by the comparison of four direct neighbors). Second, and most importantly, our algorithm does not normalize each cost using weights in each iteration. Each path conserves its weight over iterations, and the cost-weight normalization is performed only once, after diffusion has finished. This conservation of the weight of each path in our algorithm yields a performance similar to that of adaptive-weight algorithms in terms of accuracy, even when using large support regions, while the performance of [10] and [11] decreased considerably.

Our algorithm performs the diffusion of costs and weights along each disparity plane independently from other disparity planes (i.e., a fronto-parallel approach). Four data structures are used for diffusing costs and weights. The only structure that must contain all the disparity planes is the DSI D; the other three structures can be reused for every disparity plane. Here, D and a similar structure D_W accumulate the cost and weight, respectively, for each pixel during the diffusion process. D is initialized with the pixel-wise matching cost of each pixel, while in the D_W structure, all positions are initialized with ones. Diffusion is performed in two other similar structures (D_d and D_{dW}), with the only difference being that they contain four data positions per pixel instead of one. As shown in Fig. 4, each of the four positions inherits the costs and weights of each of the four direct neighbors of each pixel. Each of the four positions per pixel in D_d and D_{dW} are initialized with the values in the corresponding position of D and D_W, respectively. After the initialization stage, the diffusion process is repeated a certain number of iterations to obtain a large support region (small support regions normally produce noisy disparity maps). Based on the schematic shown in Fig. 4, the equations describing the diffusion process are

$$\begin{aligned} &D^n_{dW}(p, \overline{p}_d, i) \\ &\quad = w(p, q_i) w(\overline{p}_d, \overline{q}_{d_i}) \sum_{j=0}^{3} l((i-j) \bmod 4) D^{n-1}_{dW}(q_i, \overline{q}_{d_i}, j), \end{aligned} \tag{9}$$

$$\begin{aligned} &D^n_{d}(p, \overline{p}_d, i) \\ &\quad = \frac{\sum_{j=0}^{3} l((i-j) \bmod 4) D^{n-1}_{dW}(q_i, \overline{q}_{d_i}, j) D^{n-1}_{d}(q_i, \overline{q}_{d_i}, j)}{\sum_{j=0}^{3} l((i-j) \bmod 4) D^{n-1}_{dW}(q_i, \overline{q}_{d_i}, j)}. \end{aligned} \tag{10}$$

In these equations, i can take four different values, representing the four positions per pixel in D_d and D_{dW} (Fig. 4). Here, q_i and $\overline{q}_{d_i}$ are the left direct neighbors of p and $\overline{p}_d$ when $i = 0$, the right neighbors when $i = 2$, the upper neighbors if $i = 1$, and the lower neighbors if $i = 3$. The weights $w(p, q)$ are computed similarly to the diffusion coefficients in the anisotropic diffusion algorithm (8) using the euclidean distance between pixels in the RGB color space. As previously mentioned, performing the diffusion of weights increases the available information in each iteration. When adding the information from three paths at one position (10), weight information is used to adjust the importance of each path. Additionally, the use of four positions per pixel makes possible the distinction of four incoming directions, thus reducing the influence of loops using two simple principles. First, the cost and weight information derived from a direct neighbor is not returned to this neighbor (Fig. 4). Second, costs are only propagated with their full weights in the same direction of their propagation direction in the previous iteration. They are also propagated in the perpendicular

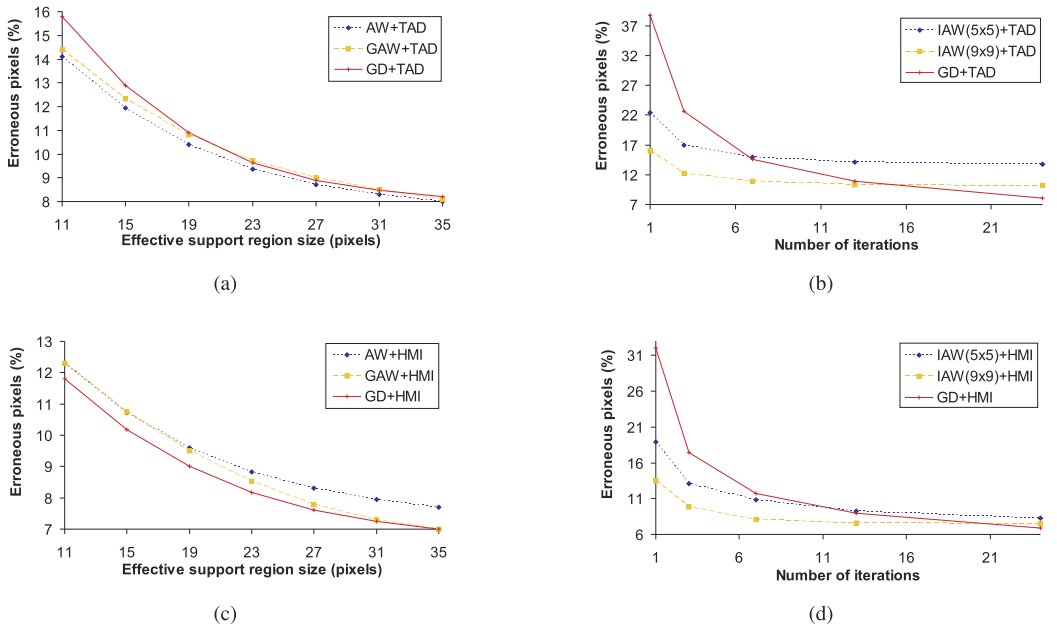

Fig. 5. Performance comparison of the different aggregation techniques studied: AW, GAW, IAW, and GD. The average percentage of erroneous pixels in nonoccluded and discontinuity areas using the Middlebury benchmark is represented in (a) and (b) using TAD as a matching measure and in (c) and (d) using HMI as a matching measure. In (a) and (c), the results are represented versus the effective support region size. In (b) and (d), the results are represented versus the number of iterations.

directions, but with a penalization parameter $0 \leq l_0 \leq 1$ (Fig. 4). These two constraints are simply formulated with the $l(i)$ function used in (9) and (10):

$$l(i) = \begin{cases} 1 & \text{if } i = 0 \\ l_0 & \text{if } i = 1 \text{ or } 3. \\ 0 & \text{if } i = 2 \end{cases} \tag{11}$$

After each iteration, the D_d and D_{dW} contents are accumulated in D and D_W, respectively:

$$D_W^n(p, \overline{p}_d) = D_W^{n-1}(p, \overline{p}_d) + \sum_{i=0}^{3} D_{dW}^n(p, \overline{p}_d, i), \tag{12}$$

$$D^n(p, \overline{p}_d) = D^{n-1}(p, \overline{p}_d) + \sum_{i=0}^{3} D_d^n(p, \overline{p}_d, i) D_{dW}^n(p, \overline{p}_d, i). \tag{13}$$

At the end of the diffusion process, the DSI costs are normalized by dividing each position in D by the corresponding position in D_W. The aggregation process by means of a diffusion method is, thus, concluded, and the disparity map is then computed by selecting the lower cost disparity for each pixel WTA.

Please note that the proposed aggregation method is based on a 4-connectivity implementation of anisotropic diffusion with several modifications incorporated to reach the same level of performance as state-of-the-art solutions. A similar approach could be proposed for an 8-connectivity implementation of anisotropic diffusion. Even if this type of diffusion requires fewer iterations to obtain a similar support area, it also introduces a series of disadvantages. First, the memory footprint of the aggregation stage is doubled. Second, the number of operations per iteration is at least doubled. Finally, new parameters should be included to model $\pm 45^\circ$ or $\pm 135^\circ$ turns in the diffusion direction.

4 Experimental Results

4.1 Quality of the Results

The Middlebury stereo evaluation website [20] provides a convenient framework for evaluating the accuracy of the reconstruction by the percentage of bad pixels in the computation of four disparity maps using four stereo pairs named Tsukuba, Venus, Teddy, and Cones. Our geodesic diffusion approach is compared with three other aggregation methods: adaptive weights [7], geodesic adaptive weights (GAW) [8], and iterative adaptive weights (IAW) [10], [11] (these last two publications both proposed a similar implementation of iterative adaptive weights, with very similar results). For IAW, 5×5 and 9×9 windows were used in our tests as proposed, respectively, by Yoon et al. [11] and Min and Sohn [10]. Two different matching measures were tested with each aggregation technique: TAD and HMI. The input stereo images were filtered for all of the aggregation methods analyzed in this section using a 5×5 bilateral filter [21] for weight computation (not for DSI computation). The reason for this choice is that high-frequency information is a very useful cue for pixel matching. In contrast, for weight computation, it is more important to associate pixels that belong to the same object, even if they have slightly different color values (because of varying texture or image noise). The parameters related to the matching measures were fixed for reasons of simplicity: the truncation value $T = 40$ for TAD and the smoothing parameter of the Gaussian filtering $\sigma = 1$ for HMI. The parameters of the bilateral filtering were not fixed because they are strongly related to the aggregation stage used. The values of these parameters and those controlling the aggregation stage were set empirically to obtain the best results. No postprocessing was applied to any of the algorithms included in Fig. 5 to highlight the performance of the raw aggregation stages.

The objective of this study was to compare the basic matching performance of the four different aggregation techniques: GD, AW, GAW, and IAW. This is why only the results on nonoccluded areas were considered and a special weight for handling occlusion was not implemented in IAW, as proposed in [11]. In fact, occlusion handling can be incorporated in most algorithms; however, for a fair comparison, only the raw aggregation algorithms were tested. Thus, the results presented in Fig. 5 are the averaged percentage of erroneous pixels using Middlebury's "nonocc" and "disc" indicators for the Tsukuba, Venus, Teddy, and Cones stereo pairs [20].

Due to the differing natures of the algorithms tested in this paper, their performance was studied using two different frameworks. The average number of erroneous pixels was compared

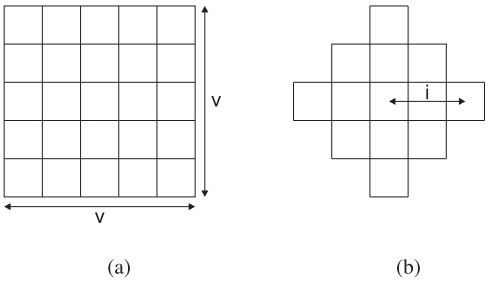

Fig. 6. Support region shape. (a) AW and GAW algorithms. (b) GD algorithm.

according to the effective support region area and according to the number of iterations. The AW and GAW algorithms are not iterative, so their performance was only tested versus the size of the support area. Additionally, the results of IAW versus the effective support region area were worse than those of AW, GAW, and GD. The reason for this difference is because IAW is an iterative algorithm that uses windows; therefore, the effective size of the support region grows rapidly with the number of iterations, but the improvement in the results with the number of iterations is less rapid. Thus, for a fair study, the comparison of the IAW algorithm against GD was performed by taking into account the number of iterations.

We defined the effective support region area as the number of pixels supporting the disparity computation for each pixel, and, to simplify the representation of the results, we also defined the effective support region size as the square root of the effective support region area. Thus, the effective support region area used in the AW and GAW algorithms with window size $v \times v$ is v^2 and the effective support region area in a GD algorithm performing i iterations is $2i^2 + 2i + 1$. The shape of these support regions is represented in Fig. 6. In Figs. 5a and 5c, the average percentage of erroneous pixels in Middlebury stereo pairs is represented versus different effective support region sizes when using TAD and HMI as matching measures, respectively. AW outperformed GAW and GD when using TAD. However, when HMI was used, GAW and GD outperformed AW, producing similar results for large support regions. Nevertheless, the percentage of erroneous pixels produced by the three techniques was very similar.

A comparison of GD and IAW versus the number of iterations is presented in Figs. 5b (matching measure TAD) and 5d (matching measure HMI). IAW is very powerful for small number of iterations (in fact, if the number of iterations is equal to 1, it is equivalent to AW). However, as the number of iterations increases, the utilization of the available information in the support region becomes poor. The performance of the basic GD idea was poor when using a small number of iterations, as expected because the size of the support region is very small for a low number of iterations, but it produced better results than IAW when the number of iterations was high. These results prove that diffusing only costs and normalizing weights to 1 after each iteration IAW limits the performance of the aggregation solution when using large support regions (a large number of iterations). However, conserving the importance of each path by diffusing costs and weights (GD) results in better performance.

To position our algorithm in the Middlebury ranking, we selected the GD+HMI implementation. The bilateral prefiltering was implemented using a 5×5 mask with the parameters $\sigma_c = 10$ and $\sigma_s = 10$, the GD aggregation stage was iterated $i = 24$ times and used the parameters $\gamma_c = 25$ and $l_0 = 0.15$. HMI was used with $\sigma = 1$ and the postprocessing step consisted of applying 3×3 median filtering and cross-checking. Disparity blobs smaller than 80 pixels were also invalidated. The disparities in these invalidated areas were filled with the content of the first validated pixel to the left or to the right (according to the nature of the occlusion). The disparity maps computed using the described algorithm are shown in Fig. 7. In Table 1, we present the results of our algorithm and compare them to the results published by the authors of the other algorithms studied. Here, we refer to the CostAggr+occ [10] and AdaptDiff [11] algorithms as IAW because both use the original scheme of AW in an iterative manner. As shown in Table 1, our GD algorithm produced a smaller average percentage of erroneous pixels than the state-of-the-art local algorithm GAW. Our algorithm has an execution time comparable to that of a simplified version of the GAW algorithm (NRTGeoSup in Table 1) proposed by Hosni et al. [18]. However, our algorithm produces more accurate results. Considering only the average percentage of erroneous pixels, GD was the best local algorithm in the Middlebury ranking. However, it did not outperform GAW in terms of its position in the Middlebury ranking [20], becoming the second local algorithm in the Middlebury ranking.

Finally, additional experiments were performed to validate the most important assumptions described in Section 3. We used the same algorithm configuration described in the previous paragraph, but modified some key steps of the algorithm (readjusting the parameter values to reach the minimum possible average percentage or erroneous pixels). For example, in Section 3, we introduced the l_0 parameter to avoid loops and enable an efficient propagation of information. When we set $l_0 = 1$, the average percentage of erroneous pixels increased from 5.49 to 10.95 percent. In addition, we mentioned that conserving the weight of each path (i.e., no renormalization of weights is performed after each iteration) should positively affect the results. When we enabled weight normalization after each iteration, the average percentage of erroneous pixels increased from 5.49 to 13.97 percent. These experiments prove the effectiveness of the proposed ideas.

4.2 Memory and Execution Time

As highlighted in the previous section, the results of the proposed aggregation algorithm were similar to the AW and GAW results and better than the IAW results. Nevertheless, the great advantage of the GD solution compared to AW and GAW is the reduction of the number of pixel comparisons from more than 1,000 (i.e., for a 33×33 window, $33 \times 33 = 1{,}089$ pixels) per pixel to just four. This reduction saves a considerable amount of computation time. After this huge reduction of exponential computations, the biggest groups of computations in our algorithm are additions and multiplications (9), (10), (12), (13), the fastest operations in today's hardware architectures.

To experimentally demonstrate the explanation in the previous paragraph, the computation time of AW, GAW, IAW, and GD are compared in Fig. 8. All algorithms were implemented in C++ and executed on an Intel Core 2 6420 CPU using just one core. GD produced results similar to AW and GAW while being a much faster aggregation technique. The improvement in speed grows with the size of the support region (see Fig. 8a).

The results in Fig. 8b show that not only did GD produce better results than IAW when using large support windows (as was shown in the previous section) but also that GD was faster, even though both types of algorithms share the advantages of having computation times that grow linearly with the effective support region size.

4.3 Near Real-Time GD

The excellent results presented in the previous section with a single CPU core suggested the implementation of GD on a massively parallel architecture. nVidia GeForce graphic cards offer TFlop performance at a low cost. To reach this level of performance, the code of the algorithms must be translated into C for CUDA, a proprietary language similar to C. A general C for

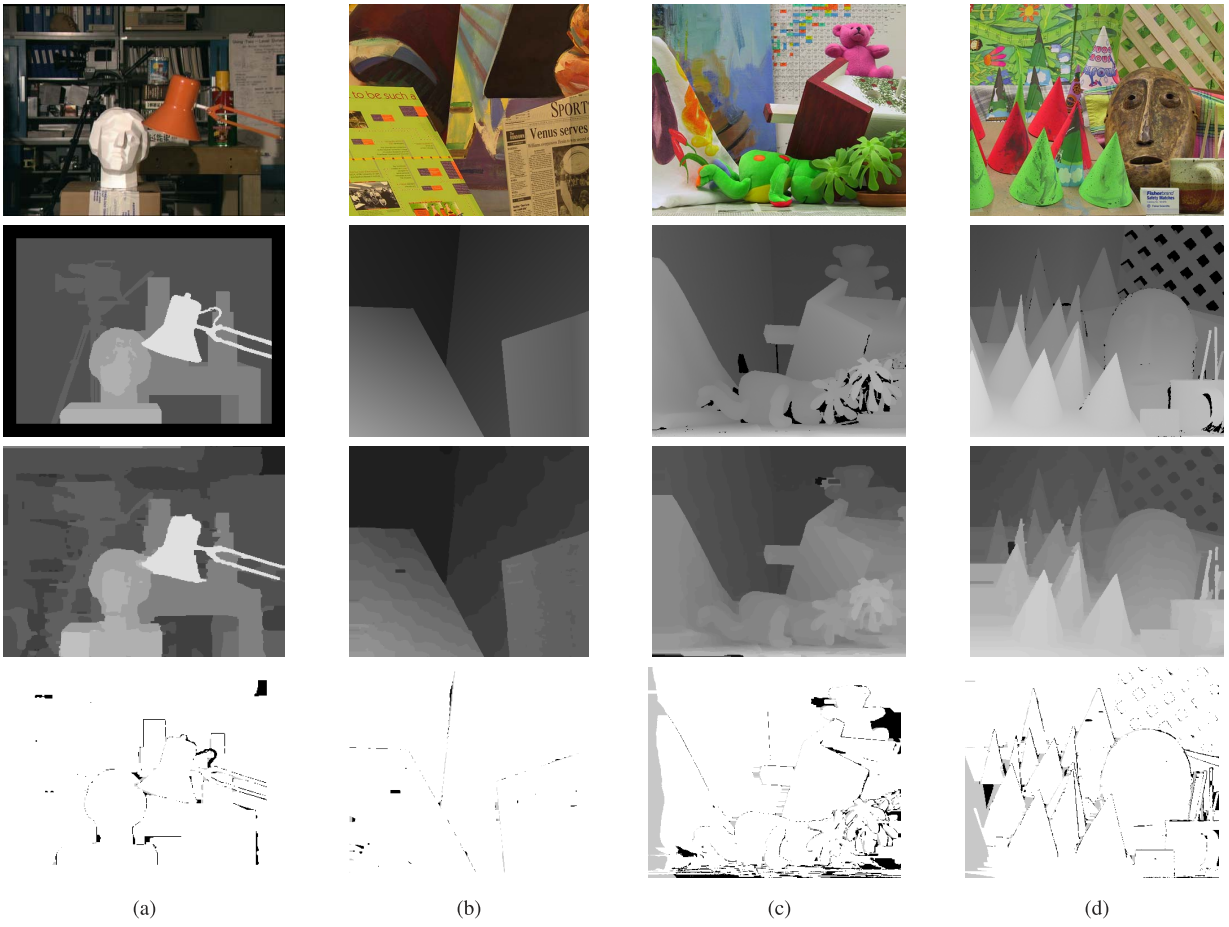

Fig. 7. Left image of each of the Middlebury data sets (first row), ground truth corresponding to each data set (second row), disparity maps computed using GD (third row), and error maps of GD results considering an error threshold of 1 pixel. (a) "Tsukuba" images. (b) "Venus" images. (c) "Teddy" images. (d) "Cones" images.

TABLE 1
Performance Comparison of the Algorithms Studied in This Publication

Algorithm	Tsukuba			Venus			Teddy			Cones			Average percent of bad pixels
	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc	
GeoDif (GD)	**1.88**	**2.35**	**7.64**	**0.38**	**0.82**	**3.02**	**5.99**	**11.3**	**13.3**	**2.84**	**8.33**	**8.09**	**5.49**
GeoSup (GAW) [8]	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
CostAggr+occ (IAW) [10]	1.38	1.96	7.14	0.44	1.13	4.87	6.80	11.9	17.3	3.60	8.57	9.36	6.20
NRTGeoSup (GAW) [18]	1.52	2.05	8.07	1.05	1.23	3.42	9.21	14.4	17.8	3.06	8.93	7.87	6.55
AdaptWeight (AW) [7]	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
AdaptDiff (IAW) [11]	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

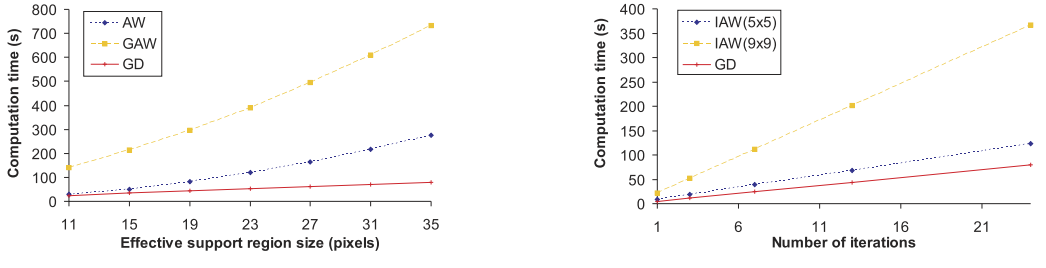

Fig. 8. Computation time for the aggregation techniques studied: AW, GAW, IAW, and GD. In (a), the results are represented versus the effective support region size. In (b), the results are represented versus the number of iterations.

CUDA implementation of our algorithm (not optimized for a specific nVidia card) ran in less than 60 milliseconds for the Tsukuba stereo pair on a GeForce 480 GTX card.

5 Conclusions

In this paper, we proposed a new local stereo matching algorithm. The new solution is based on the computational simplicity of anisotropic diffusion algorithms used in computer vision. The poor results produced by the direct implementation of anisotropic diffusion for stereo matching were greatly improved by several modifications that increased the efficiency of the aggregation process. The experimental results show that the proposed method is the second best local stereo matching algorithm in the Middlebury ranking. Moreover, we demonstrated the computational simplicity of our algorithm compared with local methods yielding similar results and showed that the algorithm is well suited for today's hardware architecture in terms of operations. We also demonstrated a near real-time implementation of the algorithm on a standard graphics card, showing that our algorithm is not just one of the best local algorithms, but that it can be used in real-world applications with low-cost hardware.

Acknowledgments

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

References

[1] D. Scharstein and R. Szeliski, "A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms," *Int'l J. Computer Vision,* vol. 47, nos. 1-3, pp. 7-42, 2002.
[2] A. Fusiello, V. Roberto, and E. Trucco, "Efficient Stereo with Multiple Windowing," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 858-863, 1997.
[3] A.F. Bobick and S.S. Intille, "Large Occlusion Stereo," *Int'l J. Computer Vision,* vol. 33, no. 3, pp. 181-200, 1999.
[4] T. Kanade and M. Okutomi, "Stereo Matching Algorithm with an Adaptive Window: Theory and Experiment," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 16, no. 9, pp. 920-932, Sept. 1994.
[5] Y. Boykov, O. Veksler, and R.I. Zabi, "A Variable Window Approach to Early Vision," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 20, no. 12, pp. 1283-1294, Dec. 1998.
[6] O. Veksler, "Fast Variable Window for Stereo Correspondence Using Integral Images," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 1, pp. I-556-I-561, 2003.
[7] K.-J. Yoon and I.S. Kweon, "Adaptive Support-Weight Approach for Correspondence Search," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 28, no. 4, pp. 650-656, Apr. 2006.
[8] A. Hosni, M. Bleyer, M. Gelautz, and C. Rhemann, "Local Stereo Matching Using Geodesic Support Weights," *Proc. Int'l Conf. Image Processing,* pp. 2093-2096, 2009.
[9] P. Perona and J. Malik, "Scale-Space and Edge Detection Using Anisotropic Diffusion," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 12, no. 7, pp. 629-639, July 1990.
[10] D. Min and K. Sohn, "Cost Aggregation and Occlusion Handling with WLS in Stereo Matching," *IEEE Trans. Image Processing,* vol. 17, no. 8, pp. 1431-1442, Aug. 2008.
[11] K.-J. Yoon, Y. Jeong, and I.-S. Kweon, "Support Aggregation via Non-Linear Diffusion with Disparity-Dependent Support-Weights for Stereo Matching," *Proc. Ninth Asian Conf. Computer Vision,* no. 1, pp. 25-36, 2009.
[12] M. Gong, R. Yang, L. Wang, and M. Gong, "A Performance Study on Different Cost Aggregation Approaches Used in Real-Time Stereo Matching," *Int'l J. Computer Vision,* vol. 75, no. 2, pp. 283-296, 2007.
[13] F. Tombari, S. Mattoccia, L.D. Stefano, and E. Addimanda, "Classification and Evaluation of Cost Aggregation Methods for Stereo Correspondence," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2008.
[14] H. Hirschmüller, "Stereo Processing by Semiglobal Matching and Mutual Information," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 30, no. 2, pp. 328-341, Feb. 2008.
[15] D. Scharstein and R. Szeliski, "Stereo Matching with Nonlinear Diffusion," *Int'l J. Computer Vision,* vol. 28, no. 2, pp. 155-174, 1998.
[16] L. Wang, M. Liao, M. Gong, R. Yang, and D. Nistér, "High-Quality Real-Time Stereo Using Adaptive Cost Aggregation and Dynamic Programming," *Proc. Third Int'l Symp. 3D Data Processing, Visualization, and Transmission,* pp. 798-805, 2007.
[17] C. Richardt, D. Orr, I. Davies, A. Criminisi, and N. Dodgson, "Real-Time Spatiotemporal Stereo Matching Using the Dual-Cross-Bilateral Grid," *Proc. 11th European Conf. Computer Vision,* pp. 510-523, 2010.
[18] A. Hosni, M. Bleyer, and M. Gelautz, "Near Real-Time Stereo with Adaptive Support Weight Approaches," *Proc. Int'l Symp. 3D Data Processing, Visualization, and Transmission,* pp. 1-8, 2010.
[19] F. Tombari, S. Mattoccia, and L.D. Stefano, "Segmentation-Based Adaptive Support for Accurate Stereo Correspondence," *Proc. Second Pacific Rim Conf. Advances in Image,* 2007.
[20] D. Scharstein and R. Szeliski, *Middlebury Stereo Evaluation—Version 2,* http://vision.middlebury.edu/stereo/eval, 2011.
[21] C. Tomasi and R. Manduchi, "Bilateral Filtering for Gray and Color Images," *Proc. IEEE Int'l Conf. Computer Vision,* pp. 839-846, 1998.