Computer Vision and Image Understanding 117 (2013) 620–632

Contents lists available at SciVerse ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

Secrets of adaptive support weight techniques for local stereo matching ☆

Asmaa Hosni *, Michael Bleyer, Margrit Gelautz

Institute for Software Technology and Interactive Systems, Vienna University of Technology, Favoritenstr. 9-11, Vienna, Austria

ARTICLE INFO

Article history:
Received 24 July 2012
Accepted 19 January 2013
Available online 4 February 2013

Keywords:
Local stereo matching
Adaptive support weights
Evaluation study

ABSTRACT

In recent years, local stereo matching algorithms have again become very popular in the stereo community. This is mainly due to the introduction of adaptive support weight algorithms that can for the first time produce results that are on par with global stereo methods. The crux in these adaptive support weight methods is to assign an individual weight to each pixel within the support window. Adaptive support weight algorithms differ mainly in the manner in which this weight computation is carried out.

In this paper we present an extensive evaluation study. We evaluate the performance of various methods for computing adaptive support weights including the original bilateral filter-based weights, as well as more recent approaches based on geodesic distances or on the guided filter. To obtain reliable findings, we test these different weight functions on a large set of 35 ground truth disparity pairs. We have implemented all approaches on the GPU, which allows for a fair comparison of run time on modern hardware platforms. Apart from the standard local matching using fronto-parallel windows, we also embed the competing weight functions into the recent PatchMatch Stereo approach, which uses slanted sub-pixel windows and represents a state-of-the-art local algorithm. In the final part of the paper, we aim at shedding light on general points of adaptive support weight matching, which, for example, includes a discussion about symmetric versus asymmetric support weight approaches.

1. Introduction

Local algorithms have a long tradition in stereo matching. They typically use squared match windows that are displaced in the second image to find a correspondence. The use of a support window leads to an implicit smoothness assumption, i.e., all pixels within the window are assumed to have the same constant disparity. The inherent problem of standard local algorithms is that this smoothness assumption is broken at depth discontinuities where the window contains pixels of the background as well as of the foreground disparity. This leads to the well-known foreground fattening effect.[1] Almost all early papers on local stereo matching have concentrated on overcoming this edge fattening effect (e.g., [1,2] to cite a few), but have not been convincingly successful in this attempt.

In 2005, Yoon and Kweon [3] introduced a surprisingly simple strategy that can clearly outperform previous local algorithms, i.e., adaptive support weight (ASW) stereo matching. This has led to the first local method that is able to compete with global algorithms in terms of quality of results (e.g., see results in the Middlebury benchmark [4]). The key idea to overcome edge-fattening is to assign an individual weight to each pixel of the match window. This weight determines the pixel's influence in the matching process. Ideally, pixels lying on the same disparity as the window's center pixel should obtain high weights (high influence) and pixels of a different disparity should obtain low weights (low influence). Defining these weights leads to a chicken and egg problem given that the disparity map is not known in advance. Yoon and Kweon's algorithm [3] as well as all subsequent ASW papers (e.g., [5–7]) exploit the appearance (color) cue to estimate the weights. The idea is that pixels whose colors are similar to the center pixel's color are also likely to be similar in disparity. We review different methods to compute these weights in the following. Note that ASW methods basically differ in the way they calculate these weights. In the first part of this paper, we present an evaluation study that aims at identifying the best weighting function, i.e., the one that leads to the lowest percentage of disparity errors on our benchmark test set.

In the original ASW paper [3], a pixel's weight inside a support window is computed from: (1) its color similarity to the window's center pixel and (2) its spatial distance from the center pixel. Note that this is equivalent to the way weights are computed in bilateral filtering, and it is known that the aggregation step of Yoon and Kweon's method [3] can be understood as filtering the cost volume

☆ This paper has been recommended for acceptance by Shahriar Negahdaripour, Ph.D.

* Corresponding author.
E-mail addresses: asmaa@ims.tuwien.ac.at (A. Hosni), bleyer@ims.tuwien.ac.at (M. Bleyer), gelautz@ims.tuwien.ac.at (M. Gelautz).

[1] Note that the assumption of constant disparity within the window is also broken for slanted surfaces where the match window contains pixels of many slightly different disparities. We will address this problem later.

1077-3142/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.cviu.2013.01.007

(disparity space image (DSI)) with a joint bilateral filter [8,7]. (This equivalence forms the foundation for various fast ASW algorithms discussed later.)

In [9] an alternative weighting function that relies on a precomputed mean-shift color segmentation is proposed. All pixels that lie in the same color segment as the center pixel are given a weight of 1, while all pixels outside the segment are given weight 0. The work of [5] refines that strategy by treating pixels that lie outside of the center pixel's segment differently. The authors use the weighting function of [3] for these pixels. (Since better results can be expected from the latter strategy, we only include [5] in our benchmark study.) Note that a problem of both algorithms is the computational overhead due to the computation of the mean-shift segmentation.

geodesic distance 是什么东西

Hosni et al. [6] define the weights within a window by computing the geodesic distance to the center pixel. The authors claim that better segmentations can be obtained by enforcing connectivity, i.e., to obtain a high weight a pixel needs to have a path of approximately constant color to the center pixel. For example, consider a plant with green leaves, each at a different disparity. For a support window that overlaps multiple leaves, the weighting function of [3] erroneously assigns high weights to all pixels regardless whether they lie on the center pixel's leaf or not, because all of them are green. Hosni et al. [6] find that there is a color edge between the center pixel's leaf and all other leaves so that pixels outside of the center pixel's leaf cannot have a path of constant color and correctly derive low weights.

有点意思

While the above papers focus on improving the quality of disparity maps, there is also a different branch of ASW algorithms that focus on computational speed. The main disadvantage of the methods above is that their computational complexity directly depends on the size of the support windows. Unfortunately, these support windows have to be large (e.g., 31 × 31 pixels) in order to handle untextured regions. In this case, computing the disparity map for a Middlebury pair can easily consume 1 min in a CPU-based implementation.

速度作为精度外的另一分支而被考虑

为何我跑了30min呢？？？

As stated above, the aggregation step of [3] is equivalent to filtering the cost volume with a joint bilateral filter. Hence, being able to implement the original ASW approach with a runtime independent of the window size boils down to the question whether it is possible to implement joint bilateral filtering with a runtime complexity independent of the filter kernel size. According to the current state of research, an O(1) implementation only works for *approximations* of the joint bilateral filter. Several authors [8,10–12] have used such approximations to derive fast implementations of the original ASW algorithm [3]. In [10,11] the joint bilateral filter is approximated by using integral histograms as described in [13]. Richardt et al. [8] uses an approximation based on the bilateral grid of [14]. Finally, the authors of [12] have presented a cost aggregation strategy that is also based on joint bilateral filtering and applies an incremental calculation scheme similar to [15]. In our benchmark, we include [8,16] as representatives of these approximate methods. The downside of these methods is that they sacrifice quality for high computational speed, and we will demonstrate this in our evaluation study.

A different strategy to derive an O(1) implementation of ASW matching is to replace the joint bilateral filter with a different filter that shares the joint bilateral filter's edge-preserving property, but can innately be implemented with a runtime independent of the filter kernel size. In this line of research, Zhang et al. [17,18] use a cross-shaped filter. Due to using a cross-shaped support region, the algorithm fails at fine structures that are neither horizontal nor vertical. Another approach is presented in [19]. The authors of that paper propose a new filtering technique, i.e., "*information permeability*" filtering. This filter can be interpreted as a hybrid approach between the cross-based filter [17] and the geodesic one [6].

A better alternative (that we include in our benchmark) is to use the recently proposed guided filter [20] for smoothing the cost volume, as has been done in [7,21]. Note that an interesting aspect of [7] is that the concept of smoothing the cost volume with an edge-preserving filter can be generalized to other computer vision problems that are typically formulated as Markov Random Fields.[2] Hence, we expect that the results of the benchmark presented in our paper are also valuable for researchers outside of stereo matching.

Moreover, it is worth mentioning the fast stereo matching method presented in [22]. The authors of that work have achieved efficient performance by: (1) reducing the computational redundancy that occurs when the aggregation is repeated at every disparity hypothesis; (2) implementing the cost aggregation step from a histogram perspective using an efficient sampling strategy.

The main contribution of this paper lies in a systematic evaluation study on ASW local stereo matching methods. We thereby focus on the role of the weights used in the cost aggregation step. Standard benchmarks such as the Middlebury online table [4] already compare a relatively large number of different ASW approaches. However, we believe that the differences in Middlebury rankings do not necessarily originate from the different weight computation methods, but rather from other ingredients such as the use of different match measures, different postprocessing procedures and the amount of parameter tuning that authors apply to optimize their ranking on the four Middlebury images. In this paper, we opt for an improved evaluation strategy by testing the weight computation algorithms under constant surrounding conditions (i.e., same match measures and postprocessing scheme) and by using a large number of 35 test images (in contrast to the four Middlebury pairs). Our benchmark shall aid other researchers in making their decisions on which weight computation method they should choose. We concentrate on two questions: Firstly, what is the best support weights computation method to achieve maximum quality? Secondly, how fast can we go in ASW matching?

精度和速度都聚焦

While we focus on standard local matching using fronto-parallel support windows in the first part of our experiments, we embed the ASW methods in an improved algorithm in the second part. This improved algorithm, i.e., PatchMatch Stereo [23], uses slanted planar support windows matched at continuous sub-pixel disparities. By moving away from the traditionally applied fronto-parallel assumption, PatchMatch Stereo achieves impressive results for slanted and rounded surfaces and shows excellent sub-pixel performance. The second series of experiments shall provide an answer to the question on how far we can go in local matching in order to maximize matching quality. Moreover, it is interesting to compare the performance of ASW strategies, if a fundamental weakness of traditional local approaches, i.e., that of using only fronto-parallel windows, is eliminated.

Another contribution of this paper is that we try to reveal several secrets of ASW stereo matching that we believe to be interesting for other researchers working in this domain. In particular, we focus on the following questions: (1) Does it make sense to compute support weights in a symmetrical manner from both images (as e.g., proposed in [3]) or is it sufficient to only use the left image in support weight computation? (2) Some researchers (e.g., in [23]) simplify the original way of ASW computation [3] by removing the spatial term to get rid of one parameter and claim that this does not considerably worsen disparity results. We investigate if this claim is true. (3) Does it make sense to apply preprocessing on the color image that is used in the weight computation (e.g., by median filtering) in order to get less noisy support weight masks?

[2] The authors demonstrate this for optical flow estimation and interactive image segmentation.

There exist several evaluation studies on the dense stereo matching problem that are discussed now. The starting point of modern stereo evaluation is the Middlebury benchmark paper [4] that has provided ground truth disparity maps for real-world stereo pairs. ([4] also conducts experiments on the aggregation step of local methods, but does not cover ASW approaches, since they have been introduced after the publication of [4]). Previous evaluation papers focus on different aspects of stereo matching algorithms including the usefulness of color information [24,25], matching using stereo pairs with poor calibration [26], evaluating confidence measures [27] and measuring the performance of radiometric insensitive match measures [28] as well as optimization algorithms [29]. The work closest to ours is [30] where the authors run an experimental comparison among different local methods. There is overlap with our study in that the ASW methods of [3,5] are investigated. The main differences are: (1) We evaluate a large set of ASW strategies, which also includes very recent ones (e.g., [7,6]). (2) We evaluate over a large set of ground truth test images, i.e., we use 35 pairs in comparison to four pairs used in [5]. (3) We have implemented all approaches on the GPU to bring these approaches to maximum speed. This allows for a fair run time comparison, as we run all algorithms on the same hardware platform and have spent considerable engineering effort to optimize the runtime behavior of *all* approaches. (4) We also evaluate ASW approaches in conjunction with slanted support windows, which leads to a more powerful local matching approach than the fronto-parallel windows used in [5]. (5) We provide general insights on ASW matching (e.g., symmetric versus asymmetric approach, the usefulness of image preprocessing). A preliminary version of this paper is found in [31].

2. Stereo algorithm

ASW stereo methods typically consist of four steps: (1) cost computation, (2) aggregation using ASW, (3) disparity selection via Winner-Takes-All (WTA) and (4) occlusion handling/postprocessing. Our study focuses on step (2) of the pipeline. Hence, in our study, we will only modify this step to test various ways for performing cost aggregation via ASW. All other steps remain constant to ensure that differences in disparity results are solely due to the use of different ASW strategies. In the following, we go through the individual steps of the pipeline.

2.1. Cost computation

In this step, the cost volume is constructed. For each pixel p of the left image and each allowed disparity d, the cost volume stores the color dissimilarity between pixel p and the pixel at coordinates $p-d$ of the right image. We follow [7] in computing these costs. In particular, we use a mixture of truncated color and gradient differences.

The color difference $M(p,d)$ for matching pixel p at disparity d is computed as:

$$M(p,d)=\sum_{i=1}^{i=3}\left|I^i_{left}(p)-I^i_{right}(p-d)\right|. \tag{1}$$

Here, $I^i(p)$ denotes the value of the ith color channel in RGB space at pixel p. The absolute difference $G()$ of gradients is expressed as:

$$G(p,d)=|\nabla_x(I_{left}(p))-\nabla_x(I_{right}(p-d))|, \tag{2}$$

where $\nabla_x(I(p))$ denotes the gradient in x direction computed at pixel p. The final cost function $C()$ is derived as:

$$C(p,d)=\alpha\cdot\min(T_c,M(p,d))+(1-\alpha)\cdot\min(T_g,G(p,d)). \tag{3}$$

Here, α balances the color and gradient terms. T_c and T_g are truncation values that help to reduce the influence of occluded pixels on the matching result. In our experiments, we set these parameters to the values proposed in [7], i.e., $T_c := 0.028$, $T_g := 0.008$ and $\alpha := 0.1$. We refer to this pixel dissimilarity measure as *Truncated Absolute Difference of Color and Gradient* (*TAD C + G*). TAD C + G represents our default match measure and if not mentioned otherwise, this is the measure used in our experiments. However, for later use, we also define the *Truncated Absolute Difference of Color* (*TAD C*), which is derived by setting $\alpha := 1$. Finally, we will also use the *Absolute Difference of Color* (*AD C*), i.e., $\alpha := 1$ and $T_c := \infty$.

Note that as explained above, we also use PatchMatch Stereo [23] in our study. PatchMatch Stereo computes disparity maps with sub-pixel accuracy and hence the parameter d of Eq. (3) becomes a float value. In this case, we compute the color and gradient values of the matching point via linear interpolation from the two closest horizontal neighbors. Also note that PatchMatch Stereo does not explicitly compute the whole cost volume. Instead it smartly traverses parts of it.

2.2. Weighted cost aggregation

As mentioned earlier, the performance of the ASW local stereo methods depends to a large extent on the support weights that are used in the aggregation step. In general, for standard ASW approaches, the aggregated matching costs $C'(p,d)$ at pixel p and disparity d are computed as:

$$C'(p,d)=\sum_{q\in w_p}W(p,q)\cdot C(q,d). \tag{4}$$

Here, w_p denotes a squared support window centered at pixel p whose size is a user-defined parameter. Note that in the case of PatchMatch Stereo [23], the aggregation is performed along a slanted plane f and the aggregation function is given by

$$C'_{PM}(p,f)=\sum_{q\in w_p}W(p,q)\cdot C(q,a_f\cdot q_x+b_f\cdot q_y+c_f), \tag{5}$$

where a_f, b_f and c_f are the three parameters of a plane f and q_x and q_y denote q's x- and y-coordinates.

Let us now focus on the function $W(p,q)$. This function computes the likelihood that pixel q lies on the same disparity with the window's center pixel p, i.e., ideally $W(p,q)$ should return 1 if pixels p and q have identical disparities and 0 otherwise. Since disparities are not known in advance, defining $W(p,q)$ is challenging and forms the subject of this study.

In the following, we define a set of weighting functions that have been proposed in the literature. We will then perform a comparison among them in the experiments section of this paper. Note that all of these weighting functions rely on the color cue to estimate a weight, which implements the assumption that within a finite neighborhood (defined by the window size) pixels of similar color are likely to lie on the same disparity (or disparity plane in the case of using slanted support windows). This assumption typically holds true on most natural images. Also note that we use an asymmetric approach for defining the weights, i.e., only the left color image is used. In the experiments section we will find that a symmetric approach (e.g., proposed in [3]) only gives a little advantage.

2.2.1. Bilateral support weights (BL and BLNoSpatial)

In the original ASW paper [3], weights are inversely proportional to (1) the color dissimilarity and (2) the spatial distance between the current pixel q and the center pixel p. As stated above, this is equivalent to the weighting function of the bilateral filter and hence we call this function bilateral support weight function. We define the function $W_{BL}(p,q)$ as:

$$W_{BL}(p,q) = \exp\left(-\left(\frac{Col(p,q)}{\gamma_c} + \frac{Dist(p,q)}{\gamma_d}\right)\right). \quad (6)$$

Here, $Col(p,q)$ is computed as:

$$Col(p,q) = \sum_{i=1}^{i=3}(|I^i(p) - I^i(q)|) \quad (7)$$

and $Dist(p,q)$ as:

$$Dist(p,q) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2} \quad (8)$$

with p_x and p_y denoting p's x- and y-coordinates. γ_c and γ_d in Eq. (6) are parameters that need to be set by the user. Here, γ_c controls the amount of smoothing in color and γ_d controls the amount of smoothing in image coordinate space.

As stated in the introduction, some authors (e.g., [23]) state that the spatial term in Eq. (6) makes very little difference on the quality of results. We will therefore also evaluate a weight function $W_{BLNoSpatial}(p,q)$ defined as:

$$W_{BLNoSpatial}(p,q) = \exp\left(-\frac{Col(p,q)}{\gamma_c}\right). \quad (9)$$

2.2.2. Geodesic support weights (GEO)

In the geodesic support weight algorithm [6], the weighting function $W_{GEO}(p,q)$ is inversely proportional to the geodesic distance between the current pixel q and the center pixel p:

$$W_{GEO}(p,q) = \exp\left(-\frac{Geo(p,q)}{\gamma_s}\right) \quad (10)$$

where the parameter γ_s controls the strength of the segmentation. $Geo(p,q)$ is defined as:

$$Geo(p,q) = \min_{P \in \mathcal{P}_{p,q}} d(P). \quad (11)$$

Here $\mathcal{P}_{p,q}$ denotes all possible paths $\langle p_1, p_2, \ldots, p_n\rangle$ that connect p and q within the support window, i.e., $p_1 = p$ and $p_n = q$. The costs of a path $d(P)$ are computed as:

$$d(P) = \sum_{i=2}^{i=n} Col(p_i, p_{i-1}). \quad (12)$$

In order to approximate $W_{GEO}(p,q)$ for all pixels q within the support window, we follow [6] and use the Borgefors algorithm [32]. Note that the geodesic support weight algorithm is sometimes criticized for being considerably slower than the bilateral support weight algorithm due to a more complex weight computation strategy. As our experiments will show, this is not true in the asymmetric case, since all required support weights $W_{GEO}(p,q)$ only need to be computed once in a preprocessing step. When looping through all allowed disparities to find the one of minimum aggregated costs (which is the computational expensive part of ASW stereo) the precomputed support masks can then be looked up from memory.

The advantage of the geodesic support weights is that connectivity is enforced in the color segmentation process, i.e., in order to obtain a high weight $W_{GEO}(p,q)$ there needs to be a path of approximately constant color between p and q. This may lead to improved segmentation results (e.g., recall the leaf example given in the introduction). However, there are also counterexamples such as a green wall paper with red dots. If the support window captures multiple red dots, all of them should ideally be given high support weights, because they all lie on the same disparity. Due to enforcing connectivity only one of the dots will be segmented and this may lead to higher ambiguity in the matching process. Whether the geodesic support weights are more powerful than the bilateral ones depends on which situation (i.e., leaf versus wall paper example) is predominant in natural images.

2.2.3. Approximate bilateral support weights ("DCBGrid" and "BLO(1)")

As stated in the introduction, ASW stereo matching is equivalent to filtering the cost volume with an edge preserving filter (also see [7]). The filter weights are thereby computed from the left color image and weighted averaging is applied on each xy-slice of the cost volume. When using a joint bilateral filter (which corresponds to the bilateral support weight function above (Section 2.2.1)), the problem is computational speed, as an exact implementation of the bilateral weights in Eq. (6) leads to an algorithm whose runtime is dependent on the match window size. However, there exist various approximations of the joint bilateral filter that remove this dependency (e.g., [33,16,34,35]) and all of them could potentially be used to derive a faster ASW algorithm. In our experiments, we include the dual-cross-bilateral grid method of [8] and the constant-time ($O(1)$) bilateral weights approximation technique based on [16]. These two methods are denoted by "DCBGrid" and "BLO(1)" in our evaluation tables. For "DCBGrid", we used the GPU code provided online by the authors, while we did a reimplementation of the latter algorithm.

2.2.4. Guided filter support weights (GF)

As stated above, the idea of filtering the cost volume is not bound to the joint bilateral filter. As an alternative, Rhemann and co-workers [7,21] suggest to use the recently proposed guided image filter [20]. The guided filter shares the edge-preserving property with the joint bilateral filter, but can be implemented in linear time (independent of the filter kernel size). (Due to this linear time property, the authors of [7] have managed to run their stereo algorithm at real-time frame rates using a GPU-based implementation.) An in-depth discussion of the guided filter would exceed the scope of this paper and the reader is referred to [20]. For completeness, we define the weight function $W_{GF}(p,q)$ as follows:

$$W_{GF}(p,q) = \frac{1}{|\omega|^2}\sum_{k:(p,q)\in\omega_k}(1 + (I_p - \mu_k)^T(\Sigma_k + \epsilon U)^{-1}(I_q - \mu_k)). \quad (13)$$

Here, I is the reference image, Σ_k and μ_k are the covariance matrix and mean vector of I in the window ω_k with dimensions $(2r+1)\times(2r+1)$ and centered at pixel k. $|\omega|$ denotes the number of pixels in this window and ϵ is a smoothness parameter. I_p, I_q and μ_k are 3×1 (color) vectors. The size of the covariance matrix Σ_k and the size of the identity matrix U is 3×3. Note that in practice the weights $W_{GF}(p,q)$ are not computed explicitly. Instead the filtered image is obtained by running a sequence of box filters whose runtime only depends on the number of pixels in the image, but not on the size of the filter kernel.

2.2.5. Weighted median support weights (WM)

We also test a filter that has not been proposed in the literature for ASW stereo matching so far, i.e., weighted median support weights. Instead of a weighted summation of cost values inside a window, the aggregation result is the weighted median of cost values. The weighted median is computed as follows:

(i) Sort all pixels of the window w_p centered on pixel p to derive a vector $\{q_1, q_2, \cdots, q_n\}$ for which $C(q_1,d) \leqslant C(q_2,d) \leqslant \cdots \leqslant C(q_n,d)$ where n is the number of pixels inside that window and d is the disparity at which the costs are computed.
(ii) Let $S(q_i)$ be a partial sum in this sorted array computed by $S(q_i) = \sum_{j=1}^{j=i} W_{BL}(p,q_j)$ with $W_{BL}()$ being defined as in Eq. (6).
(iii) Inspect the pixel q_m for which $S(q_m) \leqslant \frac{S(q_n)}{2}$ and $S(q_{m+1}) > \frac{S(q_n)}{2}$.
(iv) The weighted median of the window is then derived by $C'(p,d) = C(q_m,d)$.

Note that this method shares similarity with the disparity map postprocessing procedure of [7].

2.2.6. Segmentation-based ASW (BLSeg, GEOSeg and WMSeg)

We now focus on methods that compute an explicit color over-segmentation in a preprocessing steps. As most other segmentation-based stereo methods, we use the mean-shift based segmentation algorithm of [36] to accomplish this task. As discussed in the introduction, Tombari et al. [5] suggest assigning a weight of 1 for all pixels that lie in the same segment of the window's center pixel p. For all pixels outside of p's segment, the authors use the method of [3] (W_{BL} of Eq. (6)) to compute the weight. Formally, the weighting function $W_{BLSeg}(p,q)$ is defined as

$$W_{BLSeg}(p,q) = \begin{cases} 1.0 & \text{if } q \in S_p, \\ W_{BL}(p,q) & \text{otherwise.} \end{cases} \quad (14)$$

Here, S_p represents the set of all pixels inside p's segment. We also define a new method W_{GEOSeg}, which differs from the weighting function of Eq. (14) in that we use the geodesic weight function W_{GEO} for computing the weights of pixels outside of p's segment. Finally, we also evaluate a weighting function W_{WMSeg} where we use the weights calculated by W_{BLSeg}, but compute the weighted median (see above) instead of a weighted summation.

Note that while using a color segmentation of the left image may potentially lead to improved disparity results, a disadvantage of segmentation-based ASW methods is the computational overhead that goes along with running the segmentation algorithm in a preprocessing step. One should also be aware that there are additional parameters that need to be defined by the user, i.e., that of the segmentation algorithm.

2.3. Disparity selection

In this study, we focus on local methods, i.e., disparity optimization is performed using the WTA strategy, and leave the extension to global inference methods to future work. In the case of standard local matching using fronto-parallel windows and integer disparities, finding the disparity d_p of pixel p is straightforward. The algorithm simply checks all allowed discrete disparities for pixel p and selects the disparity d_p of minimum aggregated costs:

$$d_p = \text{argmin}_{d \in \mathcal{D}} C'(p,d). \quad (15)$$

Here, $\mathcal{D}$ represents the set of all allowed discrete disparities.

In the case of using slanted support planes and continuous disparities, we search for a plane f_p so that:

$$f_p = \text{argmin}_{f \in \mathcal{F}} C'_{PM}(p,f). \quad (16)$$

Here $\mathcal{F}$ denotes the set of all possible planes. Note that once f_p is known the disparity d_p of pixel p can simply be derived as $d_p = a_f \cdot p_x + b_f \cdot p_y + c_f$. The problem is that, since $\mathcal{F}$ is a set of infinite

Fig. 1. Test data sets used in this study. Left images and corresponding ground truth disparities are shown.

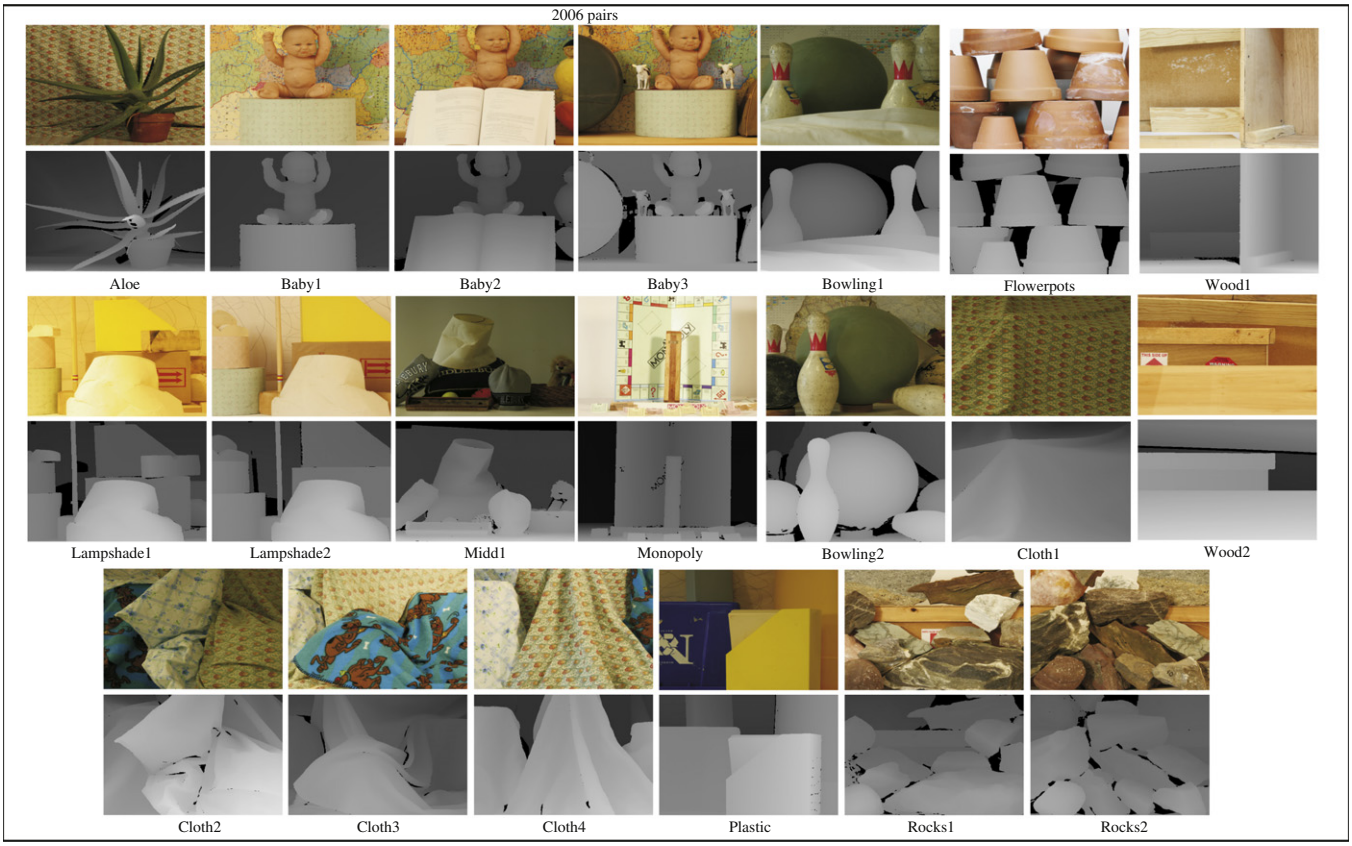

Fig. 1. (*continued*)

cardinality, the above strategy of checking all labels can no longer work. We use the strategy proposed in [23] to approximately solve this optimization problem and sketch the basic idea as follows.

The basic observation is that relatively large regions of an image can be modeled by approximately the same plane. For example, consider the Venus set in Fig. 1 that consists of four different planes. In the initialization step of the algorithm, each pixel is assigned to a plane with random parameters. It is relatively obvious that most of these random planes will be wrong, but the hope is that at least one pixel of a region carries a plane that is close to the optimal one. Note that this is very likely, since we have many guesses. For example, in the Venus image, each plane consists of approximately 50.000 pixels and we therefore have 50.000 guesses. If there is at least a single correct guess, then this is already sufficient, since PatchMatch Stereo propagates this plane to neighboring pixels. For more information on PatchMatch Stereo, the reader is referred to the corresponding paper [23]. Regarding computational speed, the authors' implementation of [23] that we use in our experiments needs approximately 1 min to compute the disparity map for one of our test pairs on the CPU.

2.4. Occlusion handling/postprocessing

Since occlusion is a global interaction between pixels, the occlusion problem cannot be modeled in a local stereo framework. We follow common practice and apply left–right consistency checking to filter out occluded pixels, i.e., we also compute the disparity map of the right image and invalidate those pixels whose disparity value is not identical to that of its matching point. To obtain a dense disparity map, we strictly follow the postprocessing strategy of [7]. We first apply a scanline filling technique. For each invalid pixel, we extract the disparity of the closest valid pixel to the left d_l and to the right d_r of the current pixel. Since occlusion occurs in the background of an image, the invalid pixel is then assigned to the minimum value of d_l and d_r. This strategy generates horizontal streaks in the disparity map and hence we apply postprocessing on the filled-in pixels. (Note that pixels that have survived left–right checking are not affected by this operation.) We perform edge-preserving smoothing on the filled-in regions by using a weighted bilateral median filter. The filter parameters can be found in [7].

3. Experiments

3.1. Benchmark setup

3.1.1. Test data

The data used in our experiments consists of 35 stereo pairs for which ground truth disparity maps are available. This data is provided by the Middlebury website [4,37–39]. Fig. 1 shows the test image pairs along with corresponding ground truth images. Note that these images are challenging due to low textured regions and complex object outlines.

3.1.2. Error metric

To measure the matching error on an image, we follow [4] and compute the percentage of pixels having an absolute disparity error larger than 1 pixel in unoccluded regions. (We only use the visible pixels to avoid that our postprocessing procedure that fills in occluded pixels has large influence on the results of our benchmark.)

3.1.3. Parameter tuning

To ensure that the results of this study are not distorted by suboptimal parameter settings, we run a separate tuning procedure for

each weight function described in Section 2.2. For each weight function, we test 26 different combinations of parameters. For each parameter setting, we compute the average error percentage over all 35 test images. Finally, we select the parameter setting that has led to the lowest average error.

3.2. Evaluation of matching accuracy

3.2.1. Fronto-parallel windows

Let us start our evaluation using the results of the standard local algorithm that applies fronto-parallel windows and integer disparities. Table 1 plots the average error computed over the 35 images for each method. The red subscripts in the table represent the rank of each method such that the method of lowest average error obtains rank 1. Note that the average error may be distorted by outliers, i.e., if one method performs particularly bad on one image, but slightly better than its competitors on all other images, it may still have a worse average error than the other methods. Hence we also plot a second error measure that is less sensitive to outliers and is also used in the Middlebury table [4], i.e. the average rank. We therefore compute the rank of each method separately on each test image, e.g., if the weight function *GF* has the lowest error percentage on the image pair Teddy it obtains rank 1. We then build the average of ranks over all test images. Note that Table 1 is sorted according to the average rank. Also note that in the following we do not include the method *BLNoSpatial* in our discussion. This method will be used in an additional experiment of Section 3.4.

Let us now look at the results of Table 1. We will first focus on the second column of the table that shows results using our default match measure, i.e., the Truncated Absolute Difference of Color and Gradient (TAD C + G) as defined in Section 2.1. The top-performing methods are the guided filter weights (*GF*) [7] as well as the original ASW function (*BL*) [3]. (While *BL* performs better according to the average rank, *GF* produces a lower average error.) However, it is noteworthy that the runtime of *GF* is considerably faster than that of *BL* as discussed later.

The 3rd rank according to both measures (average error and rank) is taken by the geodesic weight function *GEO* [6]. This is inconsistent with the rankings in the Middlebury online table where *GEO* clearly outperforms *BL*. We believe that this discrepancy comes from two reasons: (1) In [6], the authors have used a different match measure, i.e., Mutual Information. (2) We test on a large test set consisting of 35 images in contrast to the four Middlebury evaluation sets. It seems that on our large test set the connectivity property of *GEO* (also recall the leaf versus wallpaper examples from Section 2.2.2) does not lead to improvement.

Table 1
Quantitative performance of the investigated support weight computation methods for fronto-parallel aggregation. Average error percentages and ranks are shown. The subscripts represent the rank of the method in the table. We use two different match measures, i.e., TAD C + G and TAD C, defined in Section 2.1. More explanation is given in the text.

Support weights	TAD C + G		TADC	
	Avg. error (%)	Avg. rank	Avg. error (%)	Avg. rank
BL	6.20_2	2.97_1	15.46_2	3.91_2
GF	5.84_1	3.31_2	14.89_1	2.83_1
GEO	6.31_3	4.00_3	15.60_3	4.31_3
BLSeg	6.62_6	4.26_4	15.88_7	4.74_4
BLO(1)	6.72_7	4.66_5	15.81_5	5.17_5
BLNoSpatial	6.47_4	4.80_6	15.92_8	5.77_8
GEOSeg	6.55_5	5.34_7	15.80_4	5.25_6
WMSeg	9.82_9	7.83_8	24.56_{10}	8.77_{10}
WM	9.13_8	7.91_9	21.71_9	8.71_9
DCBGrid [8]	10.71_{10}	9.89_{10}	15.85_6	5.51_7

We cannot confirm that a precomputed segmentation (as proposed in [5]) helps to improve matching accuracy on our test set. As can be seen from Table 1, the segmentation-based function *BLSeg* performs worse than the original ASW function of [3]. This can also be observed when color segmentation is combined with the geodesic support weight function, i.e., *GEOSeg* performs worse than the original geodesic support weight function *GEO*. From our results it seems that there is no argument that justifies the computational overhead that goes along with running the mean shift segmentation algorithm.

Let us now focus on methods that apply approximations of the joint bilateral filter. While *BLO*(1), which is based on the O(1) joint bilateral filter approximation of [16], takes a rank in the middle of the table, the *DCBGrid* method of [8] represents the worst-performing method in our benchmark. When comparing the results of both methods against *BL*, which is the technique that both algorithms try to approximate, it becomes clear that they sacrifice quality for computational speed. However, if the speed is important we suggest using directly the guided filter method (GF) of [7], as it is even faster than the approximate methods (see discussion below) and clearly performs better in terms of matching accuracy (see Table 1).

The performance of methods based on weighted median filtering (*WM* and *WMSeg*) is relatively poor and a weighted averaging should clearly be preferred. Note that median filtering is also computationally more demanding than averaging, as a sorting operation is required.

Let us now look at the third column of Table 1 where we have used the Truncated Absolute Difference of Color (TAD C) as an alternative match measure (see Section 2.1). Note that this measure is a common choice in stereo matching. We have included these results to demonstrate that our findings about the effectiveness of aggregation schemes still hold if a different match measure is used (compare the rankings against those of the second column of Table 1). However, it is interesting to note that TAD C + G clearly outperforms TAD C, which is another finding of our evaluation study.

3.2.2. Slanted windows

We now focus on PatchMatch Stereo [23] that uses slanted support weights and continuous disparities. Note that some of the weight functions presented in Section 2.2 are impractical for PatchMatch Stereo. The reason is that PatchMatch Stereo does not filter whole *xy*-slices of the cost volume, but tests individual disparity hypotheses at each pixel. Note that when optimizing over all possible slants and continuous disparities, the corresponding cost volume has infinite size. Hence the cost filtering approach is not applicable (at least not without label quantization). Hence methods that do not explicitly perform the weight computation, but directly compute the whole filtered image are not well suited for the PatchMatch Stereo approach. This includes the guided filter approach (*GF*) and the approximations of the bilateral filter *DCBGrid* and *BLO*(1) that we exclude from the following experiment.[3] We also exclude the methods that are based on weighted median filtering as their performance in the previous experiment (see Table 1) was very poor.

Table 2 shows the results of our experiment. These results are consistent with the findings above, i.e. the bilateral weights (BL) outperform the geodesic weights (GEO) and using an explicit color segmentation (BLSeg and GEOSeg) does not lead to a better performance. Note that in general using PatchMatch Stereo instead of fronto-parallel aggregation leads to lower average errors. For

[3] In theory, one can explicitly compute the weights of the guided filter using Eq. (13). However, this is computationally extremely demanding and would not lead to a practical algorithm.

Table 2
Quantitative performance of the investigated support weights computation methods for slanted surfaces aggregation. Average ranks and average error percentages are plotted. The subscripts represent the rank of the method in the table. More explanation is given in the text.

Support weights	Avg. error (%)	Avg. rank
BL	5.69_1	1.83_1
GEO	6.20_3	2.77_2
GEOSeg	6.32_4	2.94_3
BLNoSpatial	6.11_2	3.14_4
BLSeg	6.97_5	4.31_5

example, the average error of *BL* is reduced from 6.20% (Table 1) to 5.69% (Table 2). (This performance difference would be even more drastical on images containing a large amount of slanted surfaces, e.g., street scenes.) Overall, the combination of PatchMatch Stereo and bilateral filter weights *BL* leads to the top-performing algorithm in our benchmark, i.e., the one of lowest average error percentage on our 35 test images. We believe that this combination represents the current state-of-the-art in local stereo matching. However, this does not render fronto-parallel algorithms useless as some of them have an excellent accuracy versus speed tradeoff. This is discussed below.

For a visual comparison, we show disparity and error maps for four selected stereo image pairs generated by the three top-ranked methods of Tables 1 and 2 in Figs. 2 and 3, respectively. The error maps are derived by plotting pixels whose absolute disparity error is larger than one pixel. These pixels are shown in black.

3.3. Evaluation of computational speed

Our hardware platform is an Intel Core 2 Quad 2.4 GHZ PC equipped with a GeForce GTX480 graphics card (GPU) with 1.5 GB of memory from NVIDIA. We have used CUDA [40] to implement the ASW approaches of Section 2.2 on the GPU and have spent considerable engineering effort to maximize the computational performance of all ASW strategies on our platform. Note that in the following we focus on the algorithm that is based on fronto-parallel support windows as it has real-time potential. Our CPU-based implementation of PatchMatch Stereo takes approximately 1 min per stereo pair and it is not likely that simply porting it to the GPU will lead to real-time performance due to the sequential order of some operations. Also note that the use of different ASW strategies has very little effect on the run time of PatchMatch Stereo.

In order to evaluate computational efficiency, the Million Disparity Estimations per second (MDE/s) metric is computed. The MDE/s measure for a stereo pair is computed as

$$\text{MDE/s} = imgW \cdot imgH \cdot |\mathcal{D}| \cdot \text{FPS}.$$

Here, *imgW* and *imgH* represent image width and height in pixels, $|\mathcal{D}|$ is the number of allowed discrete disparities and FPS represents the run time computed in frames per second. Note that a high MDE/s measure means that the method is fast. For each ASW method, we compute the average MDE/s measure over all 35 image pairs.

For many methods run time depends on parameter settings, i.e., the parameter that specifies the size of the match window. Hence we plot the average MDE/s measure as a function of the window size in Fig. 4.[4] Note that the window size is one parameter that we tune in order to maximize matching quality of individual methods (see Section 3.1). The actual window sizes used to generate the results of Table 1 are marked by black dots in Fig. 4[5].

[4] In the DCBGrid method [8] there is no explicit support window size. We plot the MDE/s measure as a function of the parameter σ_s that controls the amount of smoothing in the costs aggregation process. Smaller values of σ_s lead to more grid cells in the DCBGrid, which has a negative effect on the run time.

[5] This figure is an extension of a similar one used in [41] for a subset of algorithms.

As described above, the biggest advantage of $O(1)$ methods is that their run time behavior is independent of the window size, which can be seen in Fig. 4 for the graphs of the guided filter approach (*GF*) and the *BLO*(1) method. The *GF* approach is the clear winner in this plot and runs more than twice as fast as its O(1) competitor *BLO*(1). Considering a 640×480 image pair and 40 allowed disparities, the *GF* approach achieves 33.33 FPS. In contrast to the $O(1)$ methods, the run times of *BL*, *BLSeg*, *BLNoSpatial*, *GEO* and *GEOSeg* depend on the match window size (see Fig. 4) and for reasonably large windows these algorithms are considerably slower than the *GF* method. It is interesting to see that, despite a more complex weight computation strategy, the geodesic approach *GEO* is only slightly slower than the bilateral method *BL*. The reason is that we can precompute the weight masks and then retrieve them from memory when matching the window at each allowed disparity, i.e., no recomputation of weights is required (also see discussion in Section 2.2). Note that for the segmentation-based methods *BLSeg* and *GEOSeg* we have not included the run time of the mean shift segmentation, because we did not implement the segmentation algorithm on the GPU. Hence in practice these segmentation-based methods run considerably slower than what is shown in Fig. 4 due to the segmentation overhead.

3.4. Additional experiments

Let us now conduct experiments that shall provide additional insights on ASW methods.

3.4.1. Asymmetric versus symmetric support weights

In the original ASW paper [3], the authors propose to compute the weight mask from both images using a symmetric approach. In the symmetric case, aggregated matching costs are computed as:

$$C'_{SYM}(p,d) = \frac{\sum_{q \in w_p} W(p,q) \cdot W(\overline{p-d}, \overline{q-d}) \cdot C(q,d)}{\sum_{q \in w_p} W(p,q) \cdot W(\overline{p-d}, \overline{q-d})} \tag{17}$$

where $\overline{p-d}$ denotes the coordinates of p's matching point in the right image. Let us assume that the window w_p contains pixels that are only visible in the left image and will hence lead to high matching costs. In the asymmetric approach, these high-cost pixels may lead to lower aggregated costs at a wrong disparity than at the correct one. By multiplying the weights in the left image with the weights computed in the right image, these occluded pixels can be blended out, i.e., these pixels derive low overall weights after the multiplication. Hence it can be expected that the algorithm becomes more robust in the proximity of occluded regions.

Fig. 5a shows that the symmetric approach indeed performs better than the asymmetric approach, i.e., for all weight functions (*BL*, *GEO* and *GF*) the average error is lower in the symmetric case. However, the important point is that as in [3] we have used a match measure that does not truncate matching costs, i.e., the Absolute Difference of Colors (AD Cs) as defined in Section 2.1.

A different way to handle the problem of windows that partially overlap occluded pixels is to perform truncation of matching costs [42]. This truncation limits the influence of occluded pixels on the aggregated matching costs and is implemented in Eq. (3). Note that this strategy seems to be far more effective than simply relying on the symmetrical weight computation, which can be seen from considerably lower error percentages in Fig. 5b in comparison to Fig. 5a. The interesting point in Fig. 5b is that if truncation is applied, the symmetric weight computation approach does no longer outperform the asymmetrical one and even worsens the results for two of three weight functions, i.e., *BL* and *GF*. Note that the symmetric approach also leads to lower computational performance and may lead to high memory consumption (e.g., in [8]). Hence

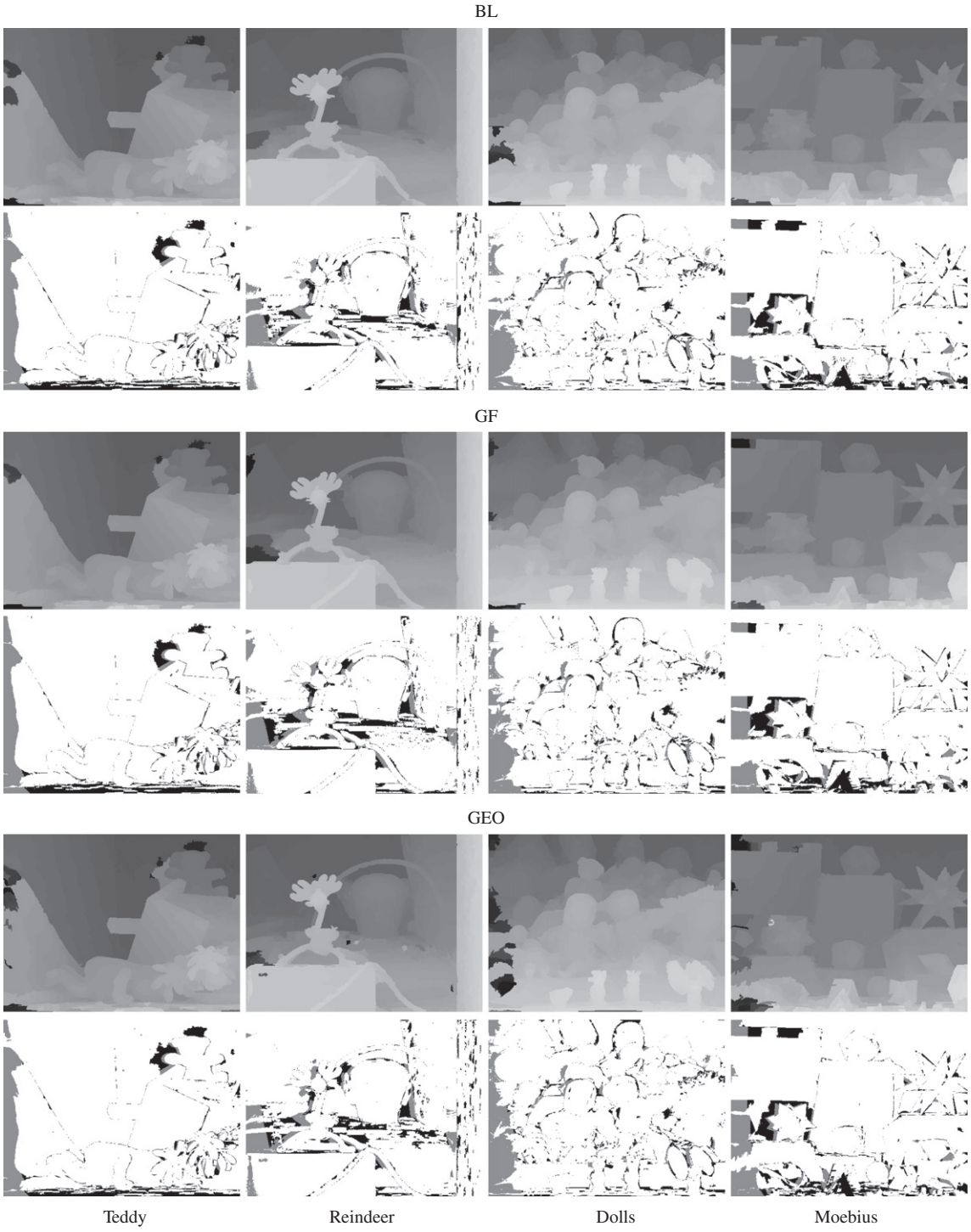

Fig. 2. Disparity and error maps for Teddy, Reindeer, Dolls and Moebius images computed by the first three ranked ASW methods in Table 1. Error pixels larger than one are shown in black in the error maps, while gray pixels correspond to errors in occluded areas.

our suggestion to other researchers is to use the simpler asymmetric approach, but to perform truncation of pixel-wise matching costs.

3.4.2. The spatial parameter in BL

As stated in the introduction, some authors claim that the spatial term in the bilateral weight computation method *BL* (Eq. (6)) is not important. To test the validity of this claim, we have also tested a method *BLNoSpatial* in Tables 1 and 2. This method is obtained by setting $\gamma_d := \infty$ in Eq. (6). Note that *BLNoSpatial* does not perform drastically worse than *BL* in both tables, i.e., for the fronto-parallel algorithm the error is increased from 6.20% (*BL*) to 6.47% (*BLNoSpatial*), while for PatchMatch Stereo the error is increased from 5.69% (*BL*) to 6.11% (*BLNoSpatial*). It is unclear whether elimination of one parameter γ_d and slightly faster processing times (see Fig. 4) outweigh the slightly decreased quality of disparity maps.

3.4.3. Preprocessing of the color image

In this experiment, we focus on the question whether we can derive better weight masks by computing the support weights on

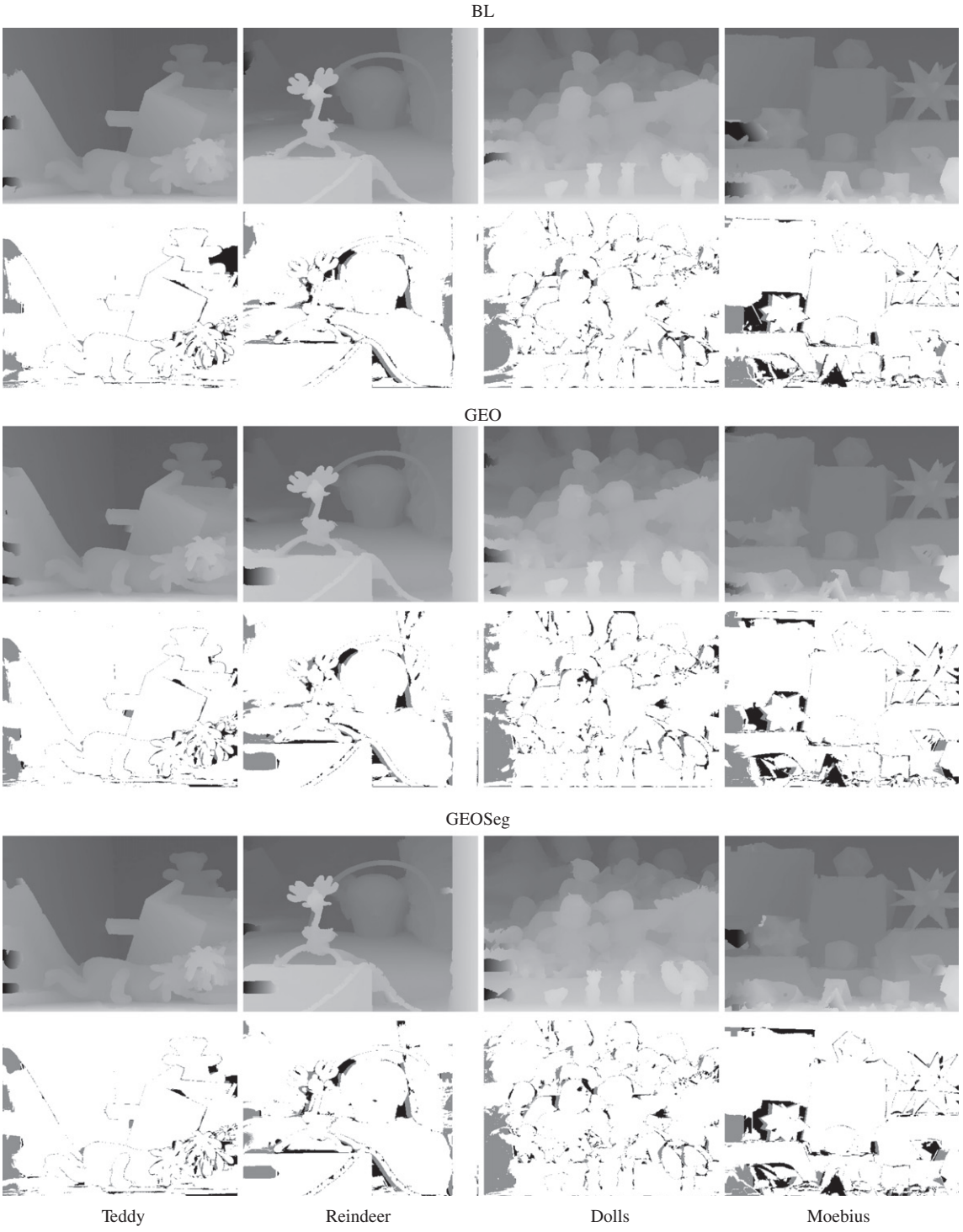

Fig. 3. Disparity and error maps for Teddy, Reindeer, Dolls and Moebius images computed by the first three ranked ASW methods in Table 2. Error pixels larger than one are shown in black in the error maps, while gray pixels correspond to errors in occluded areas.

a preprocessed color image rather than on the original left color image. The idea is to remove isolated pixels in preprocessing, i.e., single pixels whose color is considerably different from their surrounding pixels. These isolated pixels oftentimes lead to degenerated weight masks in which only the center pixel obtains high weight, while all other pixels obtain low weights. (Such degenerated weight masks cause noisy disparity results, i.e., there are many isolated pixels in the disparity map.) Note that isolated color pixels may occur due to image noise or as a consequence of the matting problem where at object borders the color of a pixel is a mixture between colors of foreground and background objects. To remove these isolated color pixels we apply standard median filtering using a small 3×3 window.[6]

Table 3 shows that this strategy does not lead to improved disparity results. For all methods listed in the table, the average error percentages are slightly higher if preprocessing of the color image

[6] We use only this filtered image to compute the support weight and still use the original color image for matching cost computation.

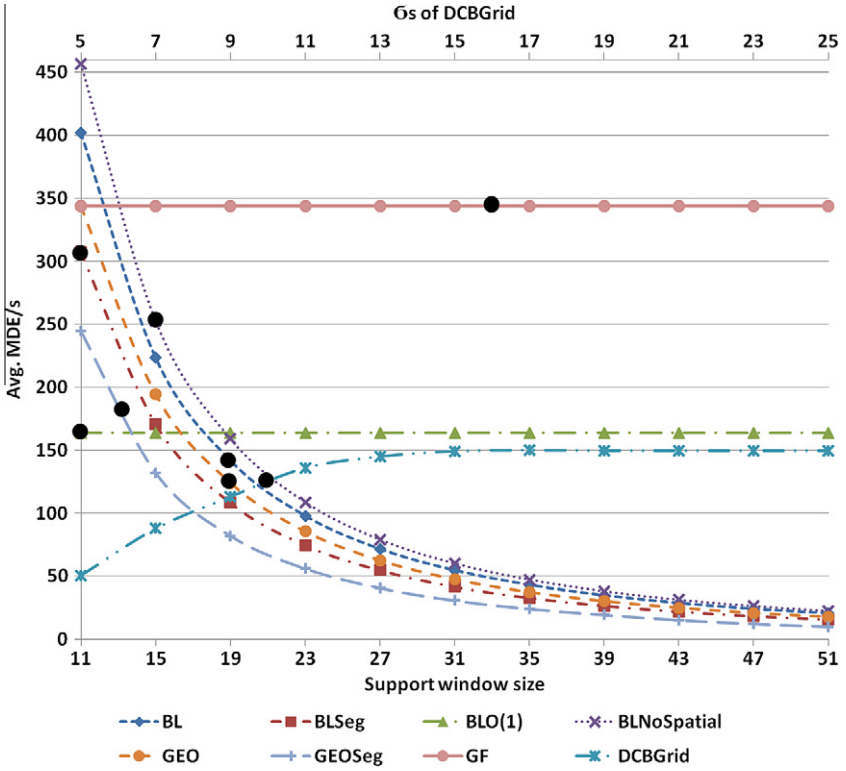

Fig. 4. Efficiency comparison of different weighting functions used in our study. Here the average MDE/s is plotted versus the support window size. The optimal window size (adjusted to get smallest average error percentage for each method individually) is marked by a black dot.

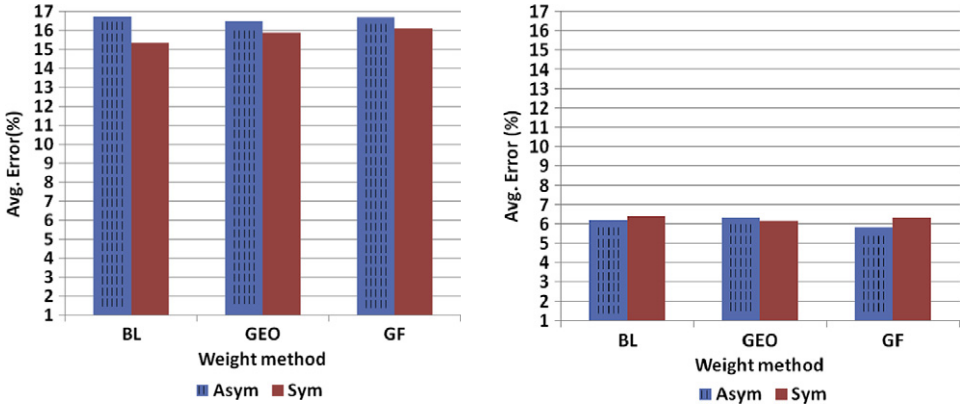

Fig. 5. Effect of using symmetric weights in the aggregation process on the average error percentage for three aggregation methods. We use two match measures. (a) Results for Absolute Difference of Color (AD C). (b) Results for Truncated Absolute Difference of Color and Gradient (TAD C + G).

Table 3
Average error of the investigated support weights computation methods with and without applying preprocessing for color images.

Support weights	W. preprocess.	W/o preprocess.
BL	6.31	6.20
BLSeg	6.68	6.62
BLO(1)	6.93	6.72
BLNoSpatial	6.60	6.47
GEO	6.44	6.31
GEOSeg	6.59	6.55
GF	5.98	5.84

Table 4
Average error of the investigated support weights computation methods with and without applying postprocessing to the disparity maps. The subscripts represent the rank of the method in the table.

Support weights	W. postprocess.	W/o postprocess.
GF	5.84_1	9.88_1
BL	6.20_2	10.15_2
GEO	6.31_3	10.99_3
GEOSeg	6.55_5	11.33_4
BLNoSpatial	6.47_4	11.82_5
BLSeg	6.62_6	11.90_6
BLO(1)	6.72_7	12.38_7

is applied. Note that, before running the Occlusion Handling Postprocessing step of our pipeline (see Section 2.4), the disparity maps obtained by using the preprocessed color image are usually less noisy. However, the left–right check is very effective in removing these isolated pixels in the disparity map so that this does not represent a big advantage. However, due to median filtering of the postprocessing step, the precision at object borders is slightly degraded.

3.4.4. Postprocessing of the disparity maps

We now focus on the effect of postprocessing, i.e., left right consistency checking and occlusion filling. Table 4 plots results with and without applying postprocessing. As can be seen, lower error percentages are achieved by applying postprocessing. This is not surprising as our postprocessing strategy is designed for filling in correct depth values in (and in the proximity of) occluded regions. It is important to note that the postprocessing step does not affect the relative ranks of the investigated support weight methods, i.e., our findings about the effectiveness of different aggregation schemes above still hold.

4. Conclusions

This paper has investigated the effect of different strategies for computing adaptive support weights (ASWs) in the aggregation step of local stereo matching methods. We have provided a systematic experimental evaluation of a variety of support weight functions suggested in the recent literature regarding the accuracy of the final disparity map and the computational efficiency of a corresponding GPU implementation. In our tests, the support weights to be evaluated are embedded into two ASW local stereo algorithms, while the remaining steps of the stereo processing chain are kept unchanged. The experiments are conducted on 35 stereo image pairs from the Middlebury data set, for which ground truth data is available.

In the first group of experiments, the adaptive support weights are plugged into a standard local stereo matching framework with the potential of real-time performance. In the second group of experiments, a suitable subset of adaptive support weight functions is tested in conjunction with the PatchMatch Stereo approach, which also accounts for slanted surfaces and fractional disparities, at the expense of computational speed. The results from the first experimental group show that the guided filter support weight approach is the overall top performer in terms of both quality and efficiency. If real-time performance is needed, it will be the first candidate method. The second part of our experiments shows that the accuracy of the disparity maps can be further improved by incorporating the adaptive support weights into the PatchMatch Stereo scheme. In this case, the best results in terms of accuracy are obtained by using an exact implementation of the bilateral filter as aggregation technique.

Another important finding is that it is not sufficient to measure the performance of a suggested support weights computation function on only the four most frequently used images of the Middlebury stereo set (i.e., the evaluation set). Since the solution for those four images is approximately reached, the performance measure will be more discriminative and fairer if it is done over all 35 image pairs. Some further insights revealed by this study include the observation that a symmetric support weights computation may be replaced by a suitable, less time-consuming asymmetric approach and that the spatial parameter of the bilateral filter can often be omitted, without loss of quality.

Since our study sheds light on potential trade-offs between the accuracy and computational efficiency of adaptive support weight techniques, we expect it to be useful for practical applications, where the choice of the most suitable stereo matching technique is strongly influenced by the requirements of the individual field.

Acknowledgments

Asmaa Hosni has been supported by the Vienna PhD School of Informatics. Michael Bleyer has been funded by the Vienna Science and Technology Fund (WWTF) through project ICT08-019.

References

[1] A. Fusiello, V. Roberto, E. Trucco, Efficient stereo with multiple windowing, in: CVPR, 1997, pp. 858–863.
[2] T. Kanade, M. Okutomi, A stereo matching algorithm with an adaptive window: Theory and experiment, TPAMI 16 (9) (1994) 920–932.
[3] K. Yoon, I. Kweon, Locally adaptive support-weight approach for visual correspondence search, in: CVPR, Vol. 2, 2005, pp. 924–931.
[4] D. Scharstein, R. Szeliski, A taxonomy and evaluation of dense two-frame stereo correspondence algorithms, IJCV 47 (1/2/3) (2002. http://www.middlebury.edu/stereo/) 7–42.
[5] F. Tombari, S. Mattoccia, L.D. Stefano, Segmentation-based adaptive support for accurate stereo correspondence, in: PSIVT, 2007.
[6] A. Hosni, M. Bleyer, M. Gelautz, C. Rhemann, Local stereo matching using geodesic support weights, in: ICIP, 2009, pp. 2093–2096.
[7] C. Rhemann, A. Hosni, M. Bleyer, C. Rother, M. Gelautz, Fast cost-volume filtering for visual correspondence and beyond, in: CVPR, 2011, pp. 3017–3024.
[8] C. Richardt, D. Orr, I. Davies, A. Criminisi, N. Dodgson, Real-time spatiotemporal stereo matching using the dual-cross-bilateral grid, in: ECCV, Vol. 6313, 2010, pp. 510–523.
[9] M. Gerrits, P. Bekaert, Local stereo matching with segmentation-based outlier rejection, in: CRV, 2006.
[10] M. Ju, H. Kang, Constant time stereo matching, in: MVIP, 2009, pp. 13–17.
[11] K. Zhang, G. Lafruit, R. Lauwereins, L. Gool, Joint integral histograms and its application in stereo matching, in: ICIP, 2010, pp. 817–820.
[12] S. Mattoccia, S. Giardino, A. Gambini, Accurate and efficient cost aggregation strategy for stereo correspondence based on approximated joint bilateral filtering, in: ACCV, 2009, pp. 371–382.
[13] F. Porikli, Integral histogram: A fast way to extract histograms in cartesian spaces, in: CVPR, Vol. 1, 2005, pp. 829–836.
[14] S. Paris, F. Durandi, A fast approximation of the bilateral filter using a signal processing approach, IJCV 81 (1) (2009) 24–52.
[15] P. Viola, M.J. Jones, Robust real-time face detection, International Journal of Computer Vision 57 (2) (2004) 137–154.
[16] Q. Yang, K. Tan, N. Ahuja, Real-time O(1) bilateral filtering, in: CVPR, 2009, pp. 557–564.
[17] K. Zhang, J. Lu, G. Lafruit, Cross-based local stereo matching using orthogonal integral images, TCSVT 19 (7) (2009) 1073–1079.
[18] X. Mei, X. Sun, M. Zhou, S. Jiao, H. Wang, X. Zhang, On building an accurate stereo matching system on graphics hardware, in: GPUCV, 2011, pp. 467–474.
[19] C. Cigla, A.A. Alatan, Efficient edge-preserving stereo matching, in: ICCV Workshops, 2011, pp. 696–699.
[20] K. He, J. Sun, X. Tang, Guided image filtering, in: ECCV, Vol. 6311, 2010, pp. 1–14.
[21] L. De-Maeztu, S. Mattoccia, A. Villanueva, R. Cabeza, Linear stereo matching, in: ICCV, 2011, pp. 1708–1715.
[22] D. Min, J. Lu, M. Do, A revisit to cost aggregation in stereo matching: How far can we reduce its computational redundancy?, in: ICCV, 2011, pp. 1567–1574.
[23] M. Bleyer, C. Rhemann, C. Rother, PatchMatch stereo - stereo matching with slanted support windows, in: BMVC, 2011.
[24] M. Bleyer, M. Gelautz, A layered stereo matching algorithm using image segmentation and global visibility constraints, ISPRS 59 (3) 128–150.
[25] M. Bleyer, S. Chambon, Does color really help in dense stereo matching?, in: 3DPVT, 2010, pp. 1 – 8..
[26] H. Hirschmüller, S. Gehrig, Stereo matching in the presence of sub-pixel calibration errors, in: CVPR, 2009, pp. 437–444.
[27] X. Hu, P. Mordohai, Evaluation of stereo confidence indoors and outdoors, in: CVPR, 2010, pp. 1466–1473.
[28] H. Hirschmüller, D. Scharstein, Evaluation of stereo matching costs on images with radiometric differences, TPAMI 31 (2009) 1582–1599.
[29] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, C. Rother, A comparative study of energy minimization methods for Markov Random Fields with smoothness-based priors, TPAMI 30 (6) (2008) 1068–1080.
[30] F. Tombari, S. Mattoccia, L.D. Stefano, E. Addimanda, Classification and evaluation of cost aggregation methods for stereo correspondence, in: CVPR, 2008, pp. 1–8.
[31] A. Hosni, M. Gelautz, M. Bleyer, Accuracy-efficiency evaluation of adaptive support weight techniques for local stereo matching, in: DAGM-OAGM, 2012, pp. 337–346.

[32] G. Borgefors, Distance transformations in digital images, Computer Vision, Graphics and Image Processing 34 (3) (1986) 344–371.
[33] F. Porikli, Constant time O(1) bilateral filtering, in: CVPR, 2008, pp. 1–8.
[34] W. Yu, F. Franchetti, J. Hoe, Y. Chang, T. Chen, Fast bilateral filtering by adapting block size, in: ICIP, 2010, pp. 3281–3284.
[35] M. Igarashi, M. Ikebe, S. Shimoyama, K. Yamano, J. Motohisa, O(1) bilateral filtering with low memory usage, in: ICIP, 2010, pp. 3301–3304.
[36] D. Comaniciu, P. Meer, Mean shift: A robust approach toward feature space analysis, TPAMI 24 (2002) 603–619.
[37] D. Scharstein, R. Szeliski, High-accuracy stereo depth maps using structured light, in: CVPR, 2003, pp. 195–202.
[38] D. Scharstein, C. Pal, Learning conditional random fields for stereo, in: CVPR, 2007, pp. 1 – 8..
[39] H. Hirschmüller, D. Scharstein, Evaluation of cost functions for stereo matching, in: CVPR, 2007, pp. 1–8.
[40] CUDA: Compute Unified Device Architecture programming guide, Tech. rep., Nvidia Corporation (2008).
[41] A. Hosni, C. Rhemann, M. Bleyer, C. Rother, M. Gelautz, Fast cost-volume filtering for visual correspondence and beyond, TPAMI, 2012. (accepted).
[42] T. Noguchi, Y. Ohta, A simple but high-quality stereo algorithm, in: ICPR, Vol. 4, 2002, pp. 351–354.