Image and Vision Computing 32 (2014) 202–211

Contents lists available at ScienceDirect

Image and Vision Computing

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

Fast stereo matching using adaptive guided filtering ☆

Qingqing Yang [a,b,c], Pan Ji [a,b], Dongxiao Li [a,b,*], Shaojun Yao [a,b], Ming Zhang [a,b]

[a] Institute of Information and Communication Engineering, Zhejiang University, Hangzhou 310027, China
[b] Zhejiang Provincial Key Laboratory of Information Network Technology, Hangzhou 310027, China
[c] School of Information Science and Engineering, Ningbo Institute of Technology, Zhejiang University, Ningbo 315100, China

ARTICLE INFO

Article history:
Received 16 August 2012
Received in revised form 10 October 2013
Accepted 9 January 2014

Keywords:
Stereo vision
Local method
Adaptive guided filtering
Parallel integral image
Weighted propagation

ABSTRACT

Dense disparity map is required by many great 3D applications. In this paper, a novel stereo matching algorithm is presented. The main contributions of this work are three-fold. Firstly, a new cost-volume filtering method is proposed. A novel concept named "two-level local adaptation" is introduced to guide the proposed filtering approach. Secondly, a novel post-processing method is proposed to handle both occlusions and textureless regions. Thirdly, a parallel algorithm is proposed to efficiently calculate an integral image on GPU, and it accelerates the whole cost-volume filtering process. The overall stereo matching algorithm generates the state-of-the-art results. At the time of submission, it ranks the 10th among about 152 algorithms on the Middlebury stereo evaluation benchmark, and takes the 1st place in all local methods. By implementing the entire algorithm on the NVIDIA Tesla C2050 GPU, it can achieve over 30 million disparity estimates per second (MDE/s).

1. Introduction

Stereo matching is one of the key problems in computer vision, and it plays a significant role in many 3D applications. However, the binocular stereo correspondence problem itself is ill-posed. The captured stereo scenes are disturbed by noise from the surrounding environment, e.g. light variations and sensor noise in image formation. Some inherent ambiguities in stereo scenes, such as textureless region and occlusion, also make the problem much more challenging. Various methods have been proposed, but the problem is still not fully solved.

Many methods are proposed under certain assumptions and constraints, among which, the local smoothness assumption is the most widely used. According to the exhaustive review [1], most stereo matching algorithms can be categorized into two major classes: global methods and local methods. Global methods explicitly incorporate smoothness assumption into an energy function and formulate the problem in an energy-minimization framework, and the disparity map is determined by finding a solution that minimizes the global energy. Although global optimization techniques currently produce the best results, almost all the state-of-the-art global methods utilize image segmentation to ensure sharp depth edges [2]. In addition, iteration is inevitable in these algorithms, so they are relatively slow and do no scale well to high-resolution images.

In local methods, data term plays a dominant role, and cost aggregation is performed in local support windows. One implicit assumption embedded in cost aggregation is that, pixels in the support window are of the same disparity. An alternative view of cost aggregation in local methods is to regard it as filtering on the cost-volume. *Data costs* are locally smoothed by some specific filtering approaches [3,4]. Well performed edge preserving filters [5,6] (and their variations) can perform proper cost filtering as well as keep fine edges. It is believed that local stereo matching will play a more significant role in the future, since with the development of cameras, it will be much easier to obtain high-quality images with high resolution. Nevertheless, existing cost filtering based local algorithms usually utilize fixed-size kernel windows, which are not scalable to objects with different sizes in the same scene. They perform even worse in textureless regions due to the restriction of the fixed kernel size. Moreover, many cost filtering approaches are time-consuming [7].

In this paper, we propose a new cost-volume filtering method, whose weight kernel is a more general form of the one proposed in [6]. A novel concept named "two-level local adaptation" is introduced to guide the proposed filtering approach. Not only are the assigned support weights locally adaptive to the local patches, the sizes of local patches are also adjusted adaptively. A novel post-processing method is also proposed to handle occlusions and textureless regions. The proposed stereo matching algorithm ranks the 10th among about 152 algorithms on the Middlebury stereo evaluation benchmark, and takes the 1st place in all local methods. Moreover, a novel algorithm based on *parallel scan* is proposed to efficiently calculate an integral image

☆ This paper has been recommended for acceptance by Michael Goesele.

* Corresponding author at: Institute of Information and Communication Engineering, Zhejiang University, Hangzhou 310027, China. Tel./fax: +86 571 87952017.
E-mail addresses: qqyang@zju.edu.cn (Q. Yang), zhedajipan@zju.edu.cn (P. Ji), lidx@zju.edu.cn (D. Li), 06gksyysj@st.zju.edu.cn (S. Yao), zhangm@zju.edu.cn (M. Zhang).

0262-8856/$ – see front matter © 2014 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.imavis.2014.01.001

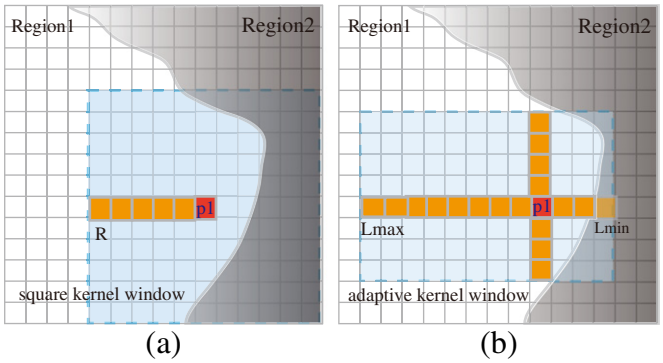

Fig. 1. Kernel windows of Eqs. (2) and (3). Pixels in shaded area represent outliers. (a) Square kernel window used in Eq. (2) and outliers presented in shaded area; (b) proposed adaptive kernel window used in Eq. (3) and outliers presented in shaded area, where fewer outliers are included.

on GPU. And the overall stereo matching algorithm can achieve over 30 million disparity estimates per second (MDE/s).[1]

The rest of this paper is organized as follows. In Section 2, related works on local stereo methods are reviewed. In Section 3, we present the proposed adaptive guided filtering approach and the overall stereo matching algorithm proposed in this paper is described in Section 4. At the end of each section, the computational complexity is analyzed. In Section 5, a hardware implementation of the cost filtering with CUDA is introduced. We report experimental results and give some discussions in Section 6, and conclusions are drawn in Section 7.

2. Related works

In local stereo matching algorithms, the disparity map is determined by selecting the value with the smallest matching cost from disparity candidates, which is also well known as winner-take-all (WTA) optimization. Thus, cost aggregation becomes the most important step in local stereo algorithms. However, it is not a trivial task as it appears to be. The most straightforward aggregation scheme is through simple low-pass filtering in the square support region. Filters with fixed-size convolution kernel, e.g. uniform (box filters), binomial or Gaussian, can be used. However, these simple methods result in poor disparity maps with fattened edges. Fixed-size kernel windows will easily overlap object boundaries, and matching costs in different regions are incorrectly aggregated. To overcome the edge fattening effect, various algorithms are proposed. Efforts on improving cost aggregation can be classified into two categories: variable support window (VSW) based approaches and adaptive support weight (ASW) based approaches.

Methods in the first category try to find support windows that fit the region size and/or shape, while preventing them from crossing object boundaries. In these methods, the piecewise smoothness assumption is utilized more explicitly, since all pixels in the support window are assumed to have the same weight. The simplest way is to filter the cost volume with a set of support windows of different sizes [1]. Its modification is shiftable windows schemes [8,9], whose window centers are anchored at different points in a set of fixed-size windows. A proper window with the most appropriate displacement is selected, which is useful at discontinuity jumps.

An alternative idea is to build a support window with variable size and/or shape tailored to the image content [10–12]. Variable window approach proposed by Veksler [12] performs well when only rectangular support windows are used. For every pixel in the reference image, a square support window is determined by minimizing the local window cost. Although the cost aggregation can be sped up by utilizing the integral image technique [13], the window adjustment step is quite time-consuming. Zhang et al. [14] proposed a fast algorithm in which non-regular support windows are used. A non-regular support window is first decomposed into horizontal and vertical slices, and cost aggregation is then performed in two directions separately.

One advantage of variable support window based approaches is that integral image technique can be utilized to speed up the aggregation procedure, which makes these cost aggregation schemes relatively efficient. However, a rectangular support window is inappropriate for pixels near object boundaries with arbitrary shapes, and a simple discontinuity reasoning method is also not strict enough to conserve edges.

Adaptive support weight based local methods, which are first introduced by Yoon and Kweon [7], adjust support weights for pixels in a local support window. In [7], the local smoothness assumption is constrained under the rules of similarity and proximity. Variations were also proposed to improve the accuracy. In [15], the authors explicitly deployed a smoothness constraint within local objects. Pixels inside the same segment in which the center pixel lies have the full weight 1, and the weights for pixels outside the segment are measured by the proximity term. Hosni et al. [16] proposed to compute the support weights by the geodesic distances, which enforce the foreground connectivity and prevent high weights from being wrongly assigned to background objects. Despite their outstanding performance, one common shortage is the high complexity. Some fast approximations [17,18] were proposed, but at the price of performance degradation. Recently, Yang [19] proposed a non-local approach, in which the cost values are aggregated adaptively on a minimum spanning tree. The support weight between two vertices is determined by their shortest distance on the tree. This literature reported better results than existing ASW based algorithms while offering extremely low computational complexity.

Few or no work tried to combine both VSW and ASW. The main reason may be: the combination of both VSW and ASW is computationally expensive with incommensurate result improvement. Readers are encouraged to refer to [20,21] for more performance study and evaluation of recent aggregation algorithms.

In recent years, cost aggregation is conducted by filtering on the cost-volume. In local stereo matching algorithms, edge preserving filters are frequently used, among which the bilateral filter [5] is the most widely used. But the brute force implementations are of high computational complexity when the kernel window is large. Many approximations [22–25] were proposed for fast implementation, but the accuracy are sacrificed due to quantization. Recently, Yang [26] proposed a recursive implementation of bilateral filtering by confining the range kernel and spatial kernel. The implementation demonstrated better accuracy than traditional bilateral filter with low complexity. But the performance of recursive bilateral filtering is still affected by the kernel confinements, this is reflected from the performance comparison with top-ranking local stereo methods. To overcome the shortages of bilateral filtering, He et al. [6] introduced the concept of guided image filtering, which has better behavior near edges. More importantly, it can be implemented exactly under linear complexity. Local methods that deployed it directly reported excellent results [3,4,27]. Based on the same filtering technique, the proposed method improves the performance by remodeling the weight kernel, following the novel concept "two-level local adaptation".

3. Adaptive guided filtering

3.1. Weight kernel remodeling

In stereo matching, the cost volume C is built in the cost computation stage. It is a three dimensional array which stores the matching costs for all possible disparity candidates. Let $C_{i,d}$ represent the cost value when pixel at $i = (x,y)$ is assigned to disparity d. Given the cost volume C and the reference image I, for each disparity candidate d, we apply

[1] The measurement of million disparity estimates per second (MDE/s) corresponds to the product of the number of pixels times the disparity range times the obtained frame-rate.

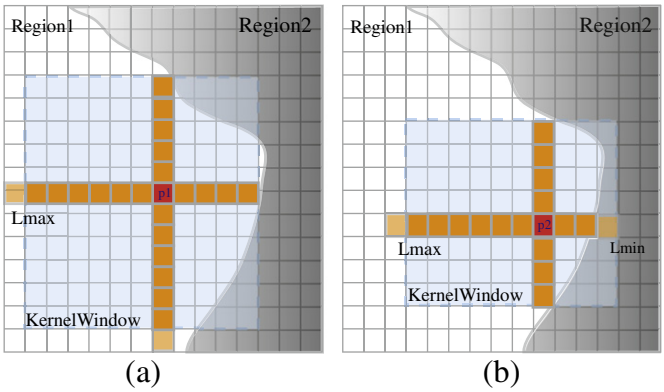

Fig. 2. Kernel window adjusting near boundary. (a) Kernel window for the center pixel $p1$, long arms are truncated by L_{max}; (b) kernel window for center pixel p2, short arms are adjusted to L_{min}.

filtering to the dth cost slice C_d. The output of filtering can be expressed as a normalized weighted sum of the input cost slice

$$C'_{i,d} = \sum_j W_{i,j}(I) C_{j,d}, \tag{1}$$

where C' is the filtered cost volume. $W_{i,j}(I)$ is the normalized weight of pixel pair (i,j), and depends on the guidance image I.

By applying the guided image filtering [6], the weight kernel can be expressed by

$$W_{i,j} = \frac{1}{|w|^2} \sum_{k:(i,j)\in w_k} \left(1 + \frac{(I_i - \mu_k)(I_j - \mu_k)}{\sigma_k^2 + \epsilon}\right). \tag{2}$$

Here, μ_k and σ_k^2 are the mean and variance of kernel window ω_k in I. ϵ is a smooth parameter, which plays an equivalent role as similarity parameter in bilateral filtering. And $|w|$ is the number of pixels in window w with fixed dimension $r \times r$.

We remodel the weight kernel by varying the kernel size. Kernel windows are adjusted adaptively for local patches. When variable kernel size is applied, the remodeled weight kernel can be expressed by the following expression:

$$W_{i,j} = \frac{1}{|w_i|} \sum_{k\in w_i} \left(\frac{1}{|w_k|} \sum_{j\in w_k} \left(1 + \frac{(I_i - \mu_k)(I_j - \mu_k)}{\sigma_k^2 + \epsilon}\right)\right), \tag{3}$$

where $|w_i|$ and $|w_k|$ are the pixel numbers in kernel window w_i and w_k respectively. This is a general form of the original weight kernel expressed by Eq. (2), while Eq. (2) is a special case when all the kernel windows have the same size, i.e. $|w_i| = |w_k| = |w|$.

With the adaptive kernel size introduced in Eq. (3), a hierarchy of two-level local adaptation is expected: the *pixel level* adaptation and the *patch level* adaptation. The *pixel level* adaptation is achieved as in existing ASW based algorithms: support weights assigned to the surrounding pixels are adaptive to the property of the local patch wherein the center pixel lies. The *patch level* adaptation ensures that the property of local patches are adaptive to the content of the guidance image.

To figure out why this additional adaptation level is meaningful in guided filtering, it is necessary to explain the characteristics of the above weight function. The numerator $(I_i - \mu_k)(I_j - \mu_k)$ in Eq. (3) is positive if I_i and I_j are located on the same side of the average value μ_k, and is negative otherwise. The value of term $1 + \frac{(I_i - \mu_k)(I_j - \mu_k)}{\sigma_k^2 + \epsilon}$ will change accordingly, so that pixel pairs on the same side are assigned large support weights and those on the different sides will be suppressed. This property ensures that sharp edges can be preserved after filtering.

The weight kernel heavily relies on two terms: mean (μ) and variance (σ^2), which represent the statistical characteristics of a local

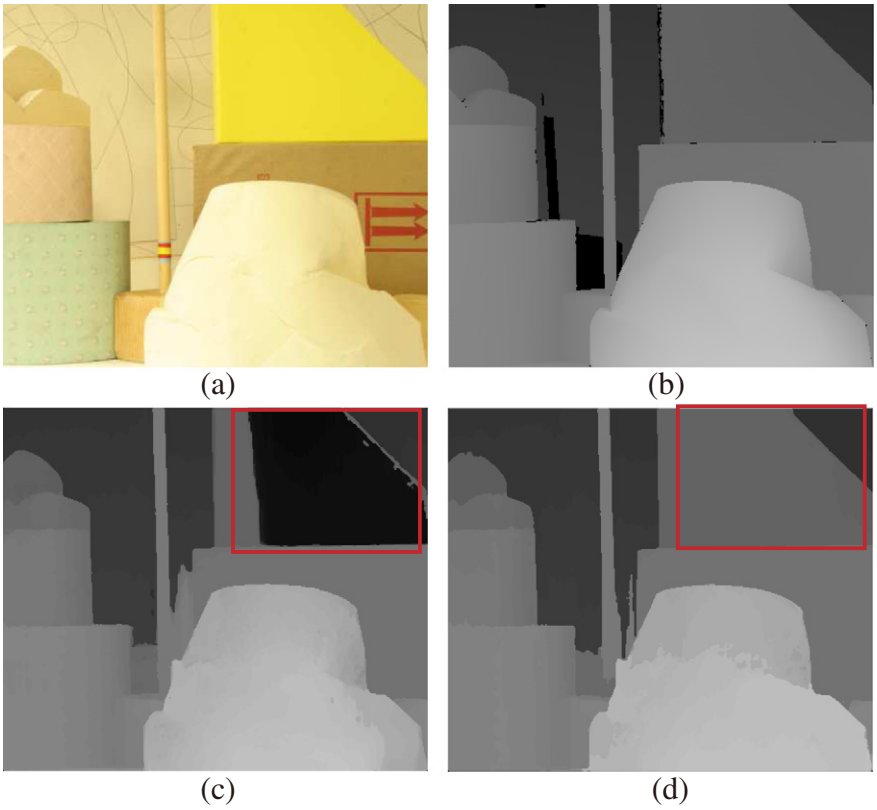

Fig. 3. Comparison of two post-processing methods. (a) Reference image; (b) ground truth; (c) result of post-processing method proposed in [3]; (d) result of the proposed post-processing method.

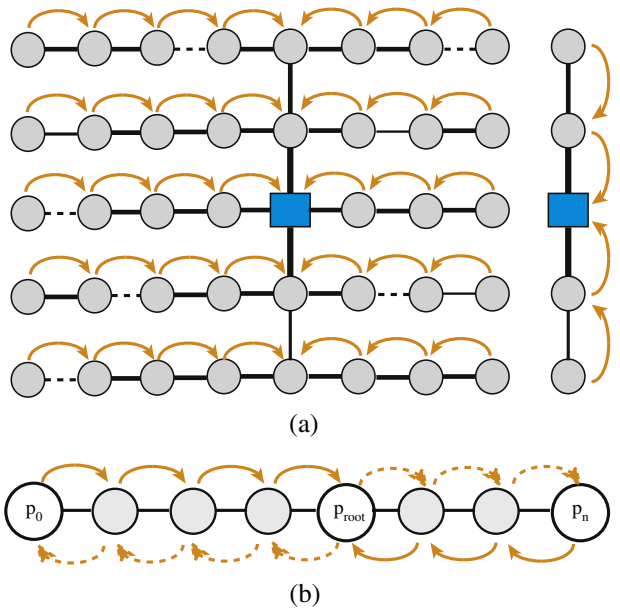

Fig. 4. Simple tree construction and weighted propagation. (a) A simple tree is constructed based on the center pixel, which is represented as blue square, branches with different transport capacity are indicated by lines with different weight, and dashed lines represent branches with the lowest weight. (b) Horizontal cost aggregation. Dashed arrows indicate unrelated calculations to process the current node.

patch. To improve the accuracy of the assigned support weights, it is meaningful to make the mean and variance represent the property of local patches more properly. As will be explained in Section 4.2 in detail, the support region (used in Eq. (3)) is now built on the skeleton stretching in four directions with four arms, which are truncated on the border of two different regions, while the original guided image filtering (expressed by Eq. (2)) utilizes square kernel windows with fixed size, resulting in much more outliers. To be more explicit, we can refer to Fig. 1. The *patch-level* adaptation makes the support weights assignment be performed in a more reasonable manner.

3.2. Linear cost-volume filtering

One advantage of the linear model is, the weight function expressed by Eq. (3) does not need to be calculated explicitly. The output can be achieved from the definition of the filter. According to the definition of the linear model, the filtered output can be expressed by

$$C'_{i,d} = \mathbf{a}_k^T \mathbf{I}_i + b_k, \forall i \in w_k. \tag{4}$$

In this paper, we assume color guide images are used. Then, $\mathbf{I}_i$ is a 3×1 vector containing color values. $\mathbf{a}_k$ is a 3×1 coefficient vector, and b_k is a scalar.

The coefficients in each kernel window are determined by minimizing the difference between the input and output. According to the cost function defined in [6], the solution given by linear regression is

$$\mathbf{a}_k = \left(\sum\nolimits_k + \epsilon \mathbf{U}\right)^{-1} \left(\frac{1}{|w_k|}\sum_{i \in w_k} \mathbf{I}_i C_{i,d} - \mu_k \overline{C}_{k,d}\right), \tag{5}$$

$$b_k = \overline{C}_{k,d} - \mathbf{a}_k^T \mu_k, \tag{6}$$

where μ_k is the mean, and $\sum_k$ is a 3×3 covariance matrix of I in kernel window ω_k respectively. $\mathbf{U}$ is a 3×3 identity matrix. $|w_k|$ is the number of pixels in ω_k, which varies according to the size of the kernel window. $\overline{C}_{k,d}$ is the mean of the input cost slice C_d in ω_k. ϵ is a regularization parameter preventing a_k from being too large, which is explained in the previous subsection. By averaging all the output values generated by all the local kernel windows $w_k (k : i \in w_k)$, we have

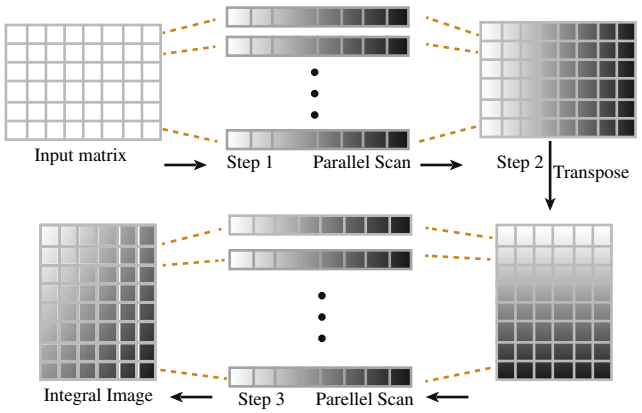

Fig. 5. Three-step parallel integral image.

$$C'_{i,d} = \overline{\mathbf{a}}_i^T \mathbf{I}_i + \overline{b}_i, \tag{7}$$

where $\overline{\mathbf{a}}_i^T = \frac{1}{|w_k|}\sum_{k \in w_i} \mathbf{a}_k^T$ and $\overline{b}_i = \frac{1}{|w_k|}\sum_{k \in w_i} b_k$. The size of support window varies with the kernel size of the local patch w_k.

To obtain the filtered output, only Eqs. (5)–(7) should be calculated. By applying the integral image technique [13], these calculations can be carried out with a sequence of box filtering operations. In the proposed adaptive guided filtering, once the anchor points (the upper-left corner and bottom-right corner) are determined, integral image technique can be applied directly. This is an extended implementation of the original linear time algorithm [6]. The complexity of the guided image filtering is not only independent of the overall kernel size, but also independent of the size varying of the kernel window (rectangular kernel window with arbitrary size can be used). Let S be the resolution of the guidance image and R be the range of disparity candidates, then the complexity of the proposed adaptive guided filtering is $O(S)$. During the cost-volume filtering process, adaptive guided filtering should be performed once for each disparity candidate. Thus the complexity of cost-volume filtering is $O(RS)$. In the rest of this paper, we will use these symbols to analyze the complexity of each subalgorithm of our stereo matching method.

4. Stereo matching algorithm

In this section, we present the overall stereo matching algorithm. Stereo pairs are assumed to be pre-rectified. Five steps are carried out to generate the final disparity map. They are: cost computation, local kernel window adjusting, cost-volume filtering, winner-take-all (WTA) disparity selecting and post-processing. Occlusions are detected and handled explicitly. A novel post-processing method is proposed to process the texureless regions, which cannot be handled well in most local algorithms.

4.1. Cost computation

Cost volume is built by computing per-pixel matching cost at given disparity values. We combine a truncated version of Birchfield and Tomasi's sampling-insensitive measure (BT) [28] and the truncated absolute difference on the gradient map. To be precise, the matching cost of pixel at $i = (x,y)$, when being assigned disparity d, can be expressed by

$$C_{i,d} = (1-\alpha)\min\left(C_{i,d}^{BT}, \tau_1\right) + \alpha\min\left(C_{i,d}^{GD}, \tau_2\right). \tag{8}$$

Table 1
Objective evaluation results of the proposed stereo matching algorithm according to the Middlebury stereo evaluation platform [34].

Algorithm	Tsukuba			Venus			Teddy			Cones			APBP
	Nonocc	All	Disc	Nonocc	All	Disc	Nonocc	All	Disc	Nonocc	All	Disc	(%)
Proposed	1.04	1.53	5.62	0.17	0.41	1.98	5.71	11.3	14.3	2.44	8.22	7.05	4.98
CostFilter [3]	1.51	1.85	7.61	0.20	0.39	2.42	6.16	11.8	16.0	2.71	8.24	7.66	5.55
NonLocalFilter [19]	1.47	1.85	7.88	0.25	0.42	2.60	6.01	11.6	14.3	2.87	8.45	8.10	5.48
P-LinearS [27]	1.10	1.67	5.92	0.53	0.89	5.71	6.69	12.0	15.9	2.60	8.44	6.71	5.68
RecursiveBF [26]	1.85	2.51	7.45	0.35	0.88	3.01	6.28	12.1	14.3	2.80	8.91	7.79	5.68
AdaptWeight [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
VariableCross [14]	1.99	2.65	6.77	0.62	0.96	3.20	9.75	15.1	18.2	6.28	12.7	12.9	7.60

Here, parameter α balances two sub cost terms, while τ_1 and τ_2 are truncation values. C^{BT} represents the cost of BT measure, and C^{GD} is the absolute difference of the reference gradient image ∇I and the target gradient image $\nabla I'$ shifted by d

$$C_{i,d}^{GD} = \left|\nabla I_{i-d}^{'} - \nabla I_i\right|. \tag{9}$$

Computed cost volume is then filtered according to the filtering approach described in Section 3. Finally, per-pixel disparity D_i is selected by simple WTA optimization

$$D_i = \arg\min_{d\in R} C_{i,d}^{'}, \tag{10}$$

where C' is the filtered cost volume, and R is the range of candidate disparity values. Both the cost computation step and WTA optimization have the complexity of $O(RS)$.

4.2. Kernel window adjusting

As expressed in Eq. (3), the mean (μ_k) and variance (σ^2) values represent the property of local patch w_k. We adjust the sizes of kernel windows aiming at excluding pixels that do not belong to the same region. A moderate percent of external pixels are allowed. Since the edge conservation mainly relies on the adaptive guided filter, the window adjusting policy is not as strict as the one used in VSW based algorithms.

In the proposed method, a support window is built upon the skeleton with four arms stretching in four directions. Four borders of the rectangular window are determined directly by the endpoints of four arms. This strategy ensures that most pixels in the local window are similar to the center pixel, and others are tolerable outliers. Arm stretching is performed in horizontal and vertical directions separately. Given a specific direction, we search the nearest pixel whose color difference exceeds the threshold τ_a to the center pixel. The endpoint of the arm in this direction is determined by it. The color difference $\Delta C_{i,j}$ is computed by

$$\Delta C_{i,j} = \min_{c\in R,G,B}\left|I_{i,c} - I_{j,c}\right|, L_{\min} \le |i-j| \le L_{\max}, \tag{11}$$

where c is one of the R, G, B color channels. $L_{\min}$ and $L_{\max}$ are truncation values that prevent arm lengths being neither too short nor too long. Fig. 2 illustrates the kernel window adjusting strategy near boundaries. In the worst case, this step needs $4L_{\max}S$ comparing operations. In most cases, only about half of this amount is required. So the complexity of the kernel window adjusting step is $O(L_{\max}S)$.

4.3. Post-processing

Despite the excellent performance of many cost-volume filtering approaches, occlusions must be handled in most local stereo matching algorithms. Many post-processing methods have been proposed in these years, especially for local algorithms. In [29] and [27], image segmentation was utilized to refine the disparity map. Rhemann et al. [3] proposed to use scanline background filling followed by bilateral weighted median filtering. It performs well in many stereo scenes. However, bilateral weighted median filtering is a local process, so it cannot handle large textureless regions even for high-quality stereo images, as shown in Fig. 3(c).

In this paper, a novel post-processing algorithm is proposed to handle occlusions as well as refine textureless regions. In the rest of this section, we describe the proposed scheme in detail.

4.3.1. Weighted propagation

For each pixel in the reference image, a simple tree graph is constructed as shown in Fig. 4(a). Each node represents a pixel and the root node is the target pixel that is to be processed. Tree branches that connect nodes are weighted by the similarity function

$$T_{p,q} = \exp\left(-\frac{\left|I_p - I_q\right|^2}{\sigma^2}\right), \tag{12}$$

where $T_{p,q}$ defines the transport capacity between two linking nodes p and q. I represents the pixel intensity, and σ adjusts the color similarity. Then, the reformulated costs are aggregated through the weighted branches from the peripheral nodes to the root.

It is work-inefficient to carry out above operations for each pixel. The complexity of the straightforward implementation is $O(RS^2)$, which is unsatisfactory for many time-constrained applications. We further simplify the approach by utilizing a two-pass model, which is widely used in many fast approximations of 2-D filters. Some dynamic-programming optimization algorithms also utilize this model to enhance inter-scanline coherence [30,31]. Weighted propagation is first carried out in the horizontal direction in separate rows. Intermediate results are then propagated in the vertical direction in separate columns in the same way. As an example, Fig. 4(b) shows how to obtain the propagated cost for the pixel p during the horizontal weighted propagation. Propagation is carried out twice. One pass is carried out from left to right, and the other is performed from right to left. The propagated result is the sum of the intermediate results from both two passes.

4.3.2. Post-processing procedure

Occlusions are first detected by a left–right consistency check, which is also well known as cross-check. Pixels that fail to pass the consistency check are marked as occluded pixels.[2] The cross-check may fail to detect many mismatched pixels, especially in low-texture regions. These pixels are further detected by peak-ratio measuring, which is expressed by

$$M_p^{PKR} = \frac{\left|C_{p,1} - C_{p,2}\right|}{C_{p,2}}, \tag{13}$$

where M_p^{PKR} is the calculated peak-ratio of pixel p. $C_{p,1}$ is the best local minimum cost, and $C_{p,2}$ is the second best local minimum cost. Pixels with peak-ratio below a specified threshold η_{PKR} are marked as unstable

[2] In fact, these pixels include both *occluded* pixels and mismatched pixels. We utilize a uniform post-processing method to refine these pixels, so they are not taken apart from each other.

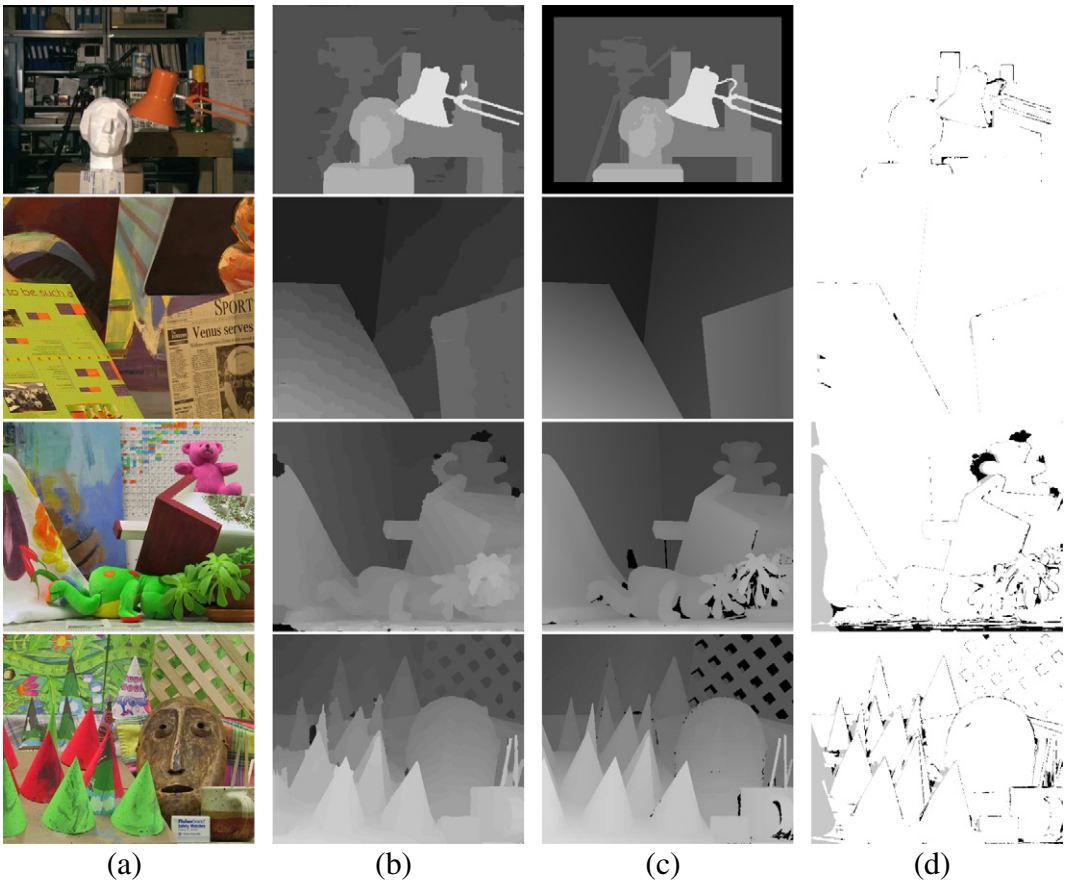

Fig. 6. Results for benchmark stereo pairs, from top to bottom: Tsukuba, Venus, Teddy and Cones. (a) Original images (left image); (b) results of the proposed method; (c) ground truth maps; (d) error maps (bad estimates with absolute disparity error > 1.0).

pixels. The cost volume C^p is then reconstructed by the following formulation:

$$C_{p,d}^{P} = \begin{cases} 0, & p \text{ is occluded,} \\ \left| C'_{p,d} - C_p^{best} \right|, & \text{otherwise.} \end{cases} \tag{14}$$

Here, $C_{p,d}^{P}$ is the reformulated cost value of pixel p at disparity candidate d. C' is the filtered cost volume, and C_p^{best} is the best cost value of pixel p after the WTA operation. Then, the reconstructed cost-volume is filtered (aggregated) by the proposed weighted propagation method, and a refined disparity map can be determined by performing another WTA operation. The final disparity map is obtained by replacing the

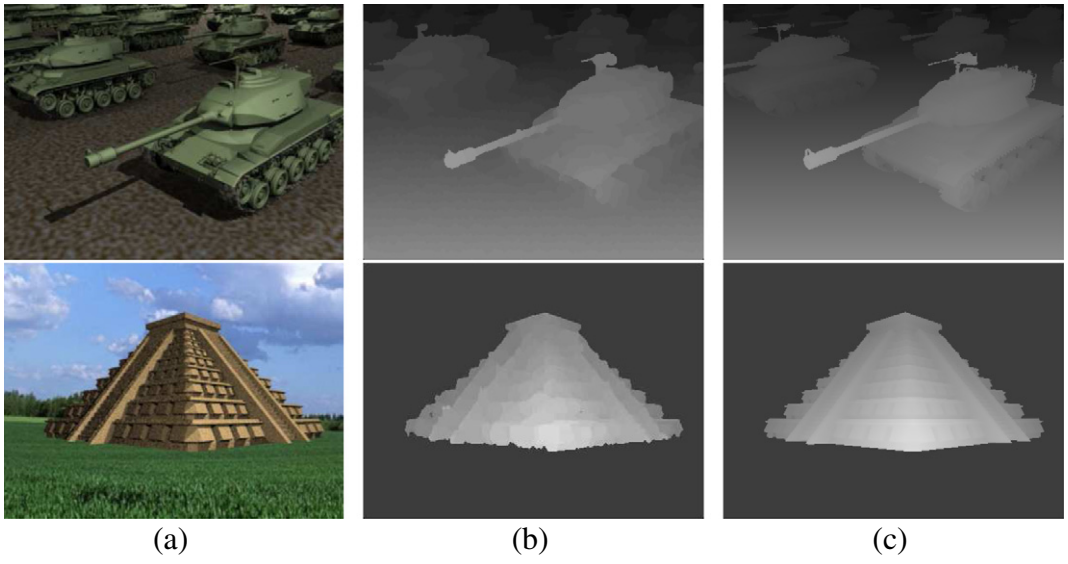

Fig. 7. Results for synthesized stereo pairs. The upper row is tanks, and the bottom row is temple. (a) Left image; (b) results of the proposed method; (c) ground truth.

Fig. 8. Results of representative data on the Middlebury website. From the top to bottom are: Dolls, Cloth2, Baby3 and Wood1. (a) Left image; (b) results of the proposed method; (c) ground truth.

disparity values of pixels that are marked *occluded* and *unstable* with the refined disparity values.

Many stereo scenes contain large regions with low texture content. For post-processing methods relying only on local processing, it is hard to recover accurate disparity for these regions. Streak-line filling will also fail if mismatches are not correctly detected, especially when only cross-check is utilized. Fig. 3(c) presents the result when the post-processing method in [3] is used. A large mismatched area occurs in the textureless region indicated by the red rectangular. The proposed method overcomes this problem by performing weighted propagation over the whole image, and mismatched pixels in textureless regions are also detected using peak-ratio measuring. As shown in Fig. 3(d), disparity in foreground textureless region can be recovered accurately.

During the two-pass weighted propagation step, two rounds of traverse are needed for each row and each column. It is carried out for each disparity candidate. Thus, the complexity of this step is $O(4RS)$, and the complexity of the final WTA operation is again $O(RS)$. Then, the complexity of post-processing is $O(RS)$, which is extremely low.

5. GPU implementation

In the cost volume filtering step, every cost slice for possible disparity levels should be filtered by the adaptive guided filter. As pointed out in Section 3.2, by applying the integral image technique [13], the adaptive guided filtering can be carried out with a sequence of box filtering operations. The efficiency of the integral image operation will make a strong impact on the performance.

Since the integral image formulation is a parallelizable task, the algorithm can be further accelerated by parallel computing devices such as GPU. In our implementation, NVIDIA's CUDA GPU computing environment is adopted. CUDA is a general-purpose parallel computing architecture and provides a C language programming interface, which makes it much more flexible and efficient.

Straightforward parallel implementation will introduce a lot of redundant computing. In this paper, the parallel implementation of integral image is further divided into prefix-sums operations in separate rows (columns).

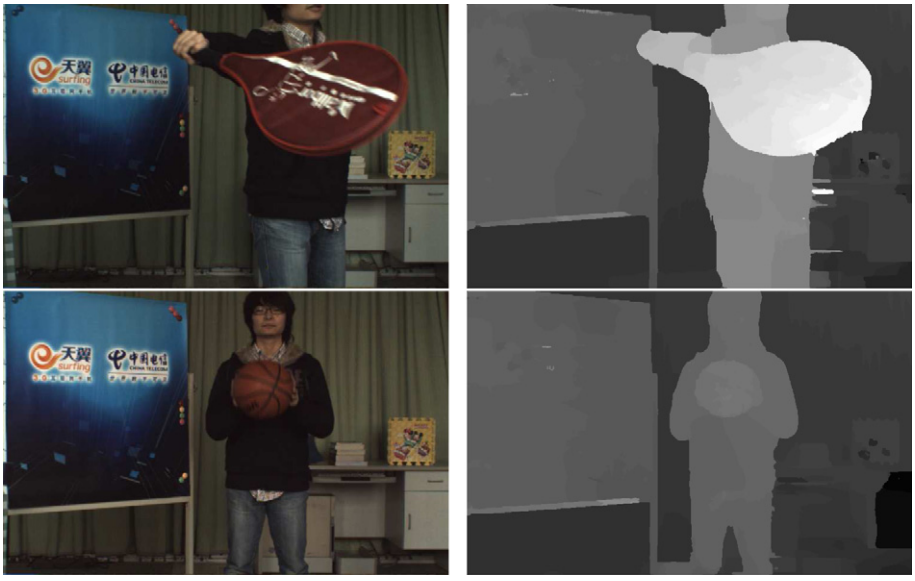

Fig. 9. Result of low-definition stereo pairs. (a) Left image; (b) results of the proposed method.

Table 2
Accuracy comparison of intermediate results after cost-volume filtering with different filtering kernel.

Algorithm	Tsukuba			Venus			Teddy			Cones			APBP
	Nonocc	All	Disc	Nonocc	All	Disc	Nonocc	All	Disc	Nonocc	All	Disc	(%)
Proposed	2.08	3.20	7.19	1.69	2.90	12.8	7.45	16.0	17.8	3.03	12.3	8.49	7.92
GF_Raw	2.33	3.27	9.09	2.01	3.28	18.8	7.95	16.6	19.2	2.92	11.8	8.25	8.79
BF_Raw	2.22	2.57	10.0	1.39	2.04	8.49	8.20	15.9	20.4	4.29	11.6	10.4	8.13

Prefix-sums on an array of data, commonly known as *scan*, was first proposed as part of the APL programming language and popularized by Blelloch [32]. Given an array of data $[a_0,a_1,\ldots,a_{n-1}]$ and a binary associative operator $\oplus$ with identity element i,[3] the output of *exclusive* scan operation is $[i,a_0,(a_0\oplus a_1),\ldots,(a_0\oplus a_1\oplus a_2\ldots\oplus a_{n-2})]$, while an *inclusive* scan operator outputs $[a_0,(a_0\oplus a_1),\ldots,(a_0\oplus a_1\oplus a_2\ldots\oplus a_{n-1})]$, which can be generated by adding the input array with the output of the *exclusive* scan.

The proposed three-step parallel integral image formulation is based on the work-efficient parallel scan [33], which can further speed up the basic *scan* operation. Thanks to the thread hierarchy provided by CUDA, it is possible to design the algorithm using the multi-level parallel programming model. Rows of the input matrix are first decomposed into separate arrays, which are scanned in parallel in their respective allocation of thread blocks. This is the *block level* parallelism. *Thread level* parallelism is distributed in each thread block, where elements in each input array are processed by threads as described in [33]. After the first step, an partially accumulated matrix is obtained.

Rather than performing *scan* on separate columns immediately, a parallel matrix transposition is added. After the matrix transposition, columns in the original matrix become rows in the transposed matrix. In the final step, parallel scan is performed on separate rows in the transposed matrix again. The only difference is that, the input array size and the number of thread blocks used are swapped. Fig. 5 presents all these three steps.

It seems that the complexity is increased by introducing an additional step. In fact, the proposed solution is much more efficient in CUDA implementation. The memory hierarchy defined by CUDA determines that it is much faster to fetch successive data from the global memory. In the global memory, multidimensional arrays are stored as one-dimensional arrays, row-by-row. Parallel data transmission along matrix columns should be avoided, since it will cause noncontinuous memory access and degrade the efficiency. A matrix transposing operation avoids such memory access at a low cost.

[3] Typical binary operators are *plus*, min, max, logical AND, logical OR etc.

6. Experimental results

6.1. Performance evaluation of the whole algorithm

The performance of the proposed stereo matching algorithm is evaluated on the Middlebury stereo evaluation website [34]. Constant parameter settings are used for all four benchmark stereo pairs: Tsukuba, Venus, Teddy and Cones. We set parameters $\{\alpha,\tau_1,\tau_2\} = \{0.11,0.027,0.008\}$ for cost computation, $\{\tau_a,L_{\min},L_{\max}\} = \{0.018,4,10\}$ for support window adjusting, $\epsilon = 5\times10^{-5}$ for cost-volume filtering and $\{\sigma,\eta_{PKR}\} = \{0.8,0.3\}$ for post-processing. The evaluation result is summarized in Table 1. Fig. 6 presents the disparity maps of these benchmark stereo pairs. Our method ranks 10th out of all 152 algorithms as of April 22th, 2013, and is the best local stereo matching method without iterative refinement. The average percent of bad pixel is 4.98% according to the evaluation website.

Apart from benchmark images, we also tested the proposed method on both synthesized and real-world stereo pairs. Fig. 7 presents the results of the proposed algorithm on two synthesized stereo pairs, tanks and temple [17]. High-quality disparity maps are generated, which is compared with the ground truth in the third column. It is clear that our algorithm performs well in both details, e.g. the gun barrels of the tanks, and large background regions.

The results of some representative data on the Middlebury website are presented in Fig. 8. They are: Dolls, Cloth2, Baby3 and Wood1 [35,36]. These stereo pairs are captured by high-end cameras in a

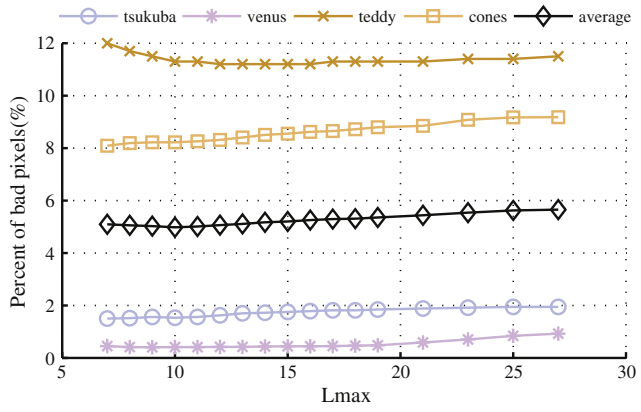

Fig. 10. Performance evaluation of the proposed method when varying the maximum allowed arm length L_{max}. Bad pixels in all regions of four benchmark images are tested.

controlled lab environment. The obtained disparity maps are quite close to the ground truth. Data Dolls presents indoor scene with many stacking objects. The proposed method can calculate accurate disparity values for most parts of the scene, and the disparity of small toys on the floor is correctly recovered. For data Cloth2, a smooth disparity map is generated for continuous cloth with repeated texture. In Baby3, objects with curved surfaces are presented, e.g. ball and cylinder. Disparity of these objects are both accurate and smooth. The disparity of the background (map) is also obtained with few mismatches. In Wood1, the texture information is much weaker, nevertheless, our method can still generate accurate and smooth disparity map that is close to the ground truth. In fact, high-definition stereo sequences are much easier to be obtained, and accurate disparity maps can be generated by the proposed method.

Our algorithm also performs well on low-definition stereo sequences, which is presented by Fig. 9. The test stereo sequences are captured by a set of low-resolution industrial video cameras without controlled lighting in our laboratory. No pre-processing, e.g. color correction, is performed after geometric rectification. Thus, it is challenging to process such stereo images. Observing the generated disparity map, sharp edges are well preserved. In textureless background like the green curtains, smooth disparity map is accurately recovered. It is proved that our algorithm is robust in such rigorous conditions. Although there are some mismatched regions on the static objects, they can be removed with proper temporal constraints, which is not discussed in this paper.

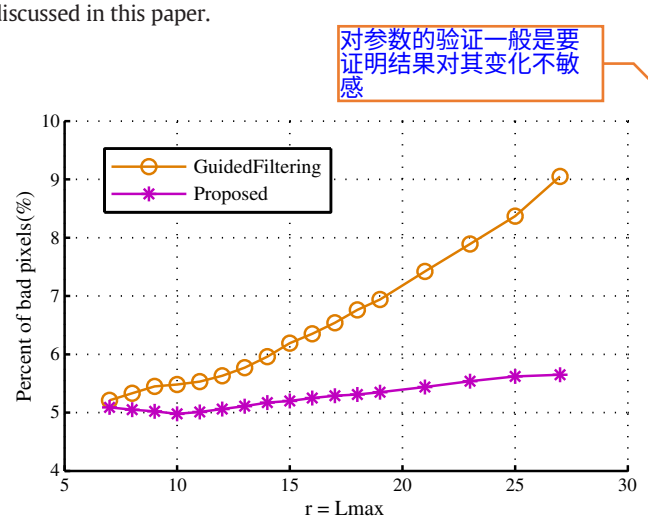

Fig. 11. Performance comparison of guided image filtering and the proposed filtering approach when varying the kernel size r of guided image filter, which is assigned to L_{max} in the proposed method.

Table 3
Runtime evaluation for benchmark stereo images.

Data set	Disparity range	CPU time (s)	GPU time (ms)	Speed up
Tsukuba	15	2.12	70	30
Venus	19	2.96	109	27
Teddy	59	8.78	312	28.1
Cones	59	8.76	308	28.5

6.2. Performance evaluation of adaptive guided filtering

To evaluate the performance of the proposed adaptive guided filtering, we make an additional comparison by implementing two other filtering kernels proposed in the literatures: the bilateral filtering kernel [5], and the original guided image filtering kernel [6]. Disparity maps without post-processing are compared on the Middlebury evaluation platform. We name these two methods as BF_Raw and GF_Raw. No symmetrical weight adjusting is deployed. Parameters used in this comparison are set as follows: for BF_Raw, a fixed kernel size 35×35 is used, and the spatial and color similarity terms are set to $\{\gamma_s,\gamma_c\} = \{17.5,0.2\}$. In our implementation, color values are handled by floating point numbers, so we adjust the color similarity term to gain a relatively good result. For GF_Raw, we set $\{\epsilon,r\} = \{0.01^2,9\}$, which was declared in [3]. The same cost computation method, which is explained in Section 4, is used to build the initial cost volume. The comparison results are reported in Table 2.

The proposed method performs better than GF_Raw and BF_Raw in most terms, especially in nonoccluded and discontinuity regions, which is gained by kernel size adjusting near discontinuities. The average percent of bad pixels (APBP) is decreased by0.87% comparing with GF_Raw and 0.21% comparing with BF_Raw. In Table 2, BF_Raw outperforms the proposed method in four 'all' columns. Since only the 'all' column counts bad pixels in the occluded regions, the improvements in these columns are gained from the occluded regions. In fact, this reflects the worse performance of BF_Raw during the cost volume filtering. The kernel of bilateral filter shows a stronger diffusion effect than the proposed method. More pixels in occluded regions are given the right values after cost filtering. This also influences foreground boundaries, which are affected by the background [16]. It is reflected by the worse performance at 'disc'. The proposed method overcomes this problem and performs better in 'disc', which is one of the most difficult regions to be handled in the stereo matching problem.

We have also tested the robustness of the proposed method by varying the maximum allowable arm length L_{max}. All other parameters are kept constant, and only L_{max} is adjusted. Bad pixels in 'all' regions of four benchmark images are evaluated in this test. The test result is reported in Fig. 10. To investigate the overall performance, we also include the average percentage of bad pixels. The test result demonstrates that the proposed method is insensitive to the change of arm length truncation. Our approach performs steadily even if the arm length truncation is set to considerable large values. The size of variable kernel window is also influenced by the minimum allowable arm length L_{min}. It should not be set to large values, since this will cover the detected edges and enlarge the kernel size at discontinuities, which runs counter to the principle of the proposed method: statistical characteristics of a local patch should be modeled with a proper patch size. Otherwise, the overall performance will be degraded. In fact, according to our experiments, L_{min} can be set to a constant value (4 in this paper) in most cases.

To compare with the performance of the original guided image filtering, we evaluated its performance by varying the overall kernel radius r. We assigned this value to the parameter L_{max} to make such comparison. L_{max} is the maximum allowed arm length, and the real kernel size in our approach is determined adaptively. Fig. 11 shows the average percentage of bad pixels for all four benchmark images. The performance of guided image filtering deteriorates significantly when

$r > 15$, while the proposed filtering method shows its robustness in this condition. Though it seems unfair to compare the performance by directly assigning r to L_{max}, we compared the overall performance. The experiment results also showed that the proposed filtering method outperforms the original guided image filtering when L_{max} varies in a wide range, even when it is set to considerable large values.

6.3. Complexity analysis and runtime evaluation

The complexity of the overall method can be computed by analyzing each subalgorithm, which is done at the end of previous subsections. The complexity of these five steps are: $O(RS)$ for cost computation, $O(L_{max}S)$ for kernel window adjusting, $O(RS)$ for cost-volume filtering, $O(RS)$ for WTA and $O(RS)$ for post-processing. Thus, the complexity of the proposed stereo matching algorithm is $O(RS)$, which only depends on the range of disparity candidates (textitR) and the resolution of the stereo pairs (S).

Both CPU implementation and GPU implementation are deployed. The CPU implementation is written in C++ and uses the OpenCV core library for basic matrix operations. The runtime is measured on a desktop with Core Duo 3.16GHz CPU and 2GB 800MHz RAM, and no parallelism technique is utilized. All operations are carried out with floating point precision. The time consumed by the benchmark stereo pairs are: Tsukuba (2.12 s), Venus (2.96 s), Teddy (8.78 s) and Cones (8.76 s). The measured time is the average of 20 separate runs. The reported runtime is competitive among the state-of-the-art algorithms, which usually need several minutes. Compared to the original guided filtering, the proposed variable kernel version needs an additional kernel window adjusting step, which takes about 16 ms for all stereo pairs when $L_{max} = 10$. Considering the precision of time measurement, the running time is negligible compared to the time consumed by the cost-volume filtering step (about 0.44 s for Tsukuba, 1.47 s for Venus, and 2.76 s for Teddy and Cones).

For the GPU implementation, the whole stereo matching algorithm is implemented on a NVIDIA Tesla C2050 GPU, whose computing capability is 2.3. The runtime is reported in Table 3. In comparison with the CPU time, the GPU code is about 28 times faster in average for benchmark stereo images. The GPU implementation can achieve over 30 million disparity estimates per second (MDE/s). For typical stereo images with disparity range 59, the average runtime of the proposed method is about 300 ms on our NVIDIA Tesla C2050 GPU.

7. Conclusions

In this paper, a new local stereo matching algorithm was introduced. The algorithm used the proposed adaptive guided filtering method, which is designed following the novel concept "two-level local adaptation". A parallel algorithm was designed to speed up the basic computing element of adaptive guided filtering, which improved the efficiency of disparity estimation significantly. We also presented a novel post-processing algorithm based on weighted propagation, which could handle both occlusions and textureless regions efficiently. The experimental results demonstrated that the proposed local algorithm can generate state-of-the-art results while preserving linear complexity.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grant Nos. 60802013, 61072081, 61271338), the National High Technology Research and Development Program (863) of China (Grant No. 2012AA011505), the National Science and Technology Major Project of the Ministry of Science and Technology of China (Grant No. 2009ZX01033-001-007), the Key Science and Technology Innovation Team of Zhejiang Province, China (Grant No. 2009R50003), and the China Postdoctoral Science Foundation (Grant Nos. 20110491804, 2012T50545).

References

[1] D. Scharstein, R. Szeliski, A taxonomy and evaluation of dense two-frame stereo correspondence algorithms, Int. J. Comput. Vis. 47 (2002) 7–42.
[2] Q. Yang, L. Wang, R. Yang, H. Stewenius, D. Nister, Stereo matching with color-weighted correlation, hierarchical belief propagation, and occlusion handling, IEEE Trans. Pattern Anal. Mach. Intell. 31 (2009) 492–504.
[3] C. Rhemann, A. Hosni, M. Bleyer, C. Rother, M. Gelautz, Fast cost-volume filtering for visual correspondence and beyond, Proceedings of the IEEE Conference on Computer Vision and, Pattern Recognition, 2011, pp. 3017–3024.
[4] A. Hosni, M. Bleyer, C. Rhemann, M. Gelautz, C. Rother, Real-time local stereo matching using guided image filtering, Proceedings of IEEE International Conference on Multimedia and Expo, 2011, pp. 1–6.
[5] C. Tomasi, R. Manduchi, Bilateral filtering for gray and color images, Proceedings of the IEEE International Conference on Computer Vision, 1998, pp. 839–846.
[6] K. He, J. Sun, X. Tang, Guided image filtering, Proceedings of the European Conference on Computer Vision, 2010, 2010, pp. 1–14.
[7] K.-J. Yoon, I.S. Kweon, Adaptive support-weight approach for correspondence search, IEEE Trans. Pattern Anal. Mach. Intell. 28 (2006) 650–656.
[8] A. Fusiello, V. Roberto, E. Trucco, Efficient stereo with multiple windowing, Proceedings of the IEEE Conference on Computer Vision and, Pattern Recognition, 1997, pp. 858–863.
[9] A.F. Bobick, S.S. Intille, Large occlusion stereo, Int. J. Comput. Vis. 33 (1999) 181–200.
[10] T. Kanade, M. Okutomi, A stereo matching algorithm with an adaptive window: theory and experiment, IEEE Trans. Pattern Anal. Mach. Intell. 16 (1994) 920–932.
[11] Y. Boykov, O. Veksler, R. Zabih, A variable window approach to early vision, IEEE Trans. Pattern Anal. Mach. Intell. 20 (1998) 1283–1294.
[12] O. Veksler, Fast variable window for stereo correspondence using integral images, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, 2003, pp. 556–561.
[13] F.C. Crow, Summed-area tables for texture mapping, SIGGRAPH Comput. Graph. 18 (1984) 207–212.
[14] K. Zhang, J. Lu, G. Lafruit, Cross-based local stereo matching using orthogonal integral images, IEEE Trans. Circuits Syst. Video Technol. 19 (2009) 1073–1079.
[15] F. Tombari, S. Mattoccia, L. Di Stefano, Segmentation-based adaptive support for accurate stereo correspondence, Proceedings of Advances in Image and Video Technology, 2007, 2007, pp. 427–438.
[16] A. Hosni, M. Bleyer, M. Gelautz, C. Rhemann, Local stereo matching using geodesic support weights, Proceedings of IEEE International Conference on Image Processing, 2009, pp. 2093–2096.
[17] C. Richardt, D. Orr, I. Davies, A. Criminisi, N.A. Dodgson, Real-time spatiotemporal stereo matching using the dual-cross-bilateral grid, Proceedings of the European Conference on Computer Vision, 2010, pp. 510–523.
[18] D. Min, J. Lu, M. Do, A revisit to cost aggregation in stereo matching: how far can we reduce its computational redundancy? Proceedings of the IEEE International Conference on Computer Vision, 2011, pp. 1567–1574.
[19] Q. Yang, A non-local cost aggregation method for stereo matching, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp. 1402–1409.
[20] F. Tombari, S. Mattoccia, L. Di Stefano, E. Addimanda, Classification and evaluation of cost aggregation methods for stereo correspondence, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp. 1–8.
[21] M. Gong, R. Yang, L. Wang, M. Gong, A performance study on different cost aggregation approaches used in real-time stereo matching, Int. J. Comput. Vis. 75 (2007) 283–296.
[22] S. Paris, F. Durand, A fast approximation of the bilateral filter using a signal processing approach, Int. J. Comput. Vis. 81 (2009) 24–52.
[23] B. Weiss, Fast median and bilateral filtering, ACM Transactions on Graphics, volume 25, 2006, pp. 519–526.
[24] F. Porikli, Constant time o(1) bilateral filtering, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp. 1–8.
[25] Q. Yang, K.-H. Tan, N. Ahuja, Real-time o(1) bilateral filtering, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 1–8.
[26] Q. Yang, Recursive bilateral filtering, Proceedings of the European Conference on Computer Vision, 2012, pp. 1–8.
[27] L. De-Maeztu, S. Mattoccia, A. Villanueva, R. Cabeza, Linear stereo matching, Proceedings of the IEEE International Conference on Computer Vision, 2011, pp. 1708–1715.
[28] S. Birchfield, C. Tomasi, A pixel dissimilarity measure that is insensitive to image sampling, IEEE Trans. Pattern Anal. Mach. Intell. 20 (1998) 401–406.
[29] H. Hirschmüller, Stereo processing by semiglobal matching and mutual information, IEEE Trans. Pattern Anal. Mach. Intell. 30 (2008) 328–341.
[30] J.C. Kim, K.M. Lee, B.T. Choi, S.U. Lee, A dense stereo matching using two-pass dynamic programming with generalized ground control points, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol. 2, 2005, pp. 1075–1082.
[31] M. Gong, Y.-H. Yang, Real-time stereo matching using orthogonal reliability-based dynamic programming, IEEE Trans. Image Process. 16 (2007) 879–884.
[32] G.E. Blelloch, Prefix sums and their applications, Technical Report CMU-CS-90-190, School of Computer Science, Carnegie Mellon University, 1990.
[33] M. Harris, S. Sengupta, J. Owens, Parallel prefix sum (scan) with CUDA, GPU Gems 3 (2007) 851–876.
[34] D. Scharstein, R. Szeliski, Middlebury stereo evaluation — version 2, http://vision.middlebury.edu/stereo/eval2002.
[35] D. Scharstein, C. Pal, Learning conditional random fields for stereo, Proceedings of the IEEE Conference on Computer Vision and, Pattern Recognition, 2007, pp. 1–8.
[36] H. Hirschmüller, D. Scharstein, Evaluation of cost functions for stereo matching, Proceedings of the IEEE Conference on Computer Vision and, Pattern Recognition, 2007, pp. 1–8.