J Real-Time Image Proc (2014) 9:447–461
DOI 10.1007/s11554-012-0275-4

SPECIAL ISSUE

Real-time stereo using approximated joint bilateral filtering and dynamic programming

Liang Wang · Ruigang Yang · Minglun Gong · Miao Liao

Received: 1 May 2012 / Accepted: 16 August 2012 / Published online: 5 September 2012

Abstract We present a stereo algorithm that is capable of estimating scene depth information with high accuracy and in real time. The key idea is to employ an adaptive cost-volume filtering stage in a dynamic programming optimization framework. The per-pixel matching costs are aggregated via a separable implementation of the bilateral filtering technique. Our separable approximation offers comparable edge-preserving filtering capability and leads to a significant reduction in computational complexity compared to the traditional 2D filter. This cost aggregation step resolves the disparity inconsistency between scanlines, which are the typical problem for conventional dynamic programming based stereo approaches. Our algorithm is driven by two design goals: real-time performance and high accuracy depth estimation. For computational efficiency, we utilize the vector processing capability and parallelism in commodity graphics hardware to speed up this aggregation process over two orders of magnitude. Over 90 million disparity evaluations per second [the number of disparity evaluations per seconds (MDE/s) corresponds to the product of the number of pixels and the disparity range and the obtained frame rate and, therefore, captures the performance of a stereo algorithm in a single number] are achieved in our current implementation. In terms of quality, quantitative evaluation using data sets with ground truth disparities shows that our approach is one of the state-of-the-art real-time stereo algorithms.

Keywords Real-time stereo · Cost aggregation · Bilateral filtering · Dynamic programming · Disparity map · Stereo video

Electronic supplementary material The online version of this article (doi:10.1007/s11554-012-0275-4) contains supplementary material, which is available to authorized users.

L. Wang (✉)
Applied Sciences Group, Microsoft Corporation,
15001 NE 40th St, Redmond, WA 98052, USA
e-mail: liangwan@microsoft.com

R. Yang
Computer Science Department, University of Kentucky,
329 Rose Street, Lexington, KY 40506, USA
e-mail: ryang@cs.uky.edu

M. Gong
Department of Computer Science, Memorial University
of Newfoundland, St.Johns, NL A1B 3X5, Canada
e-mail: gong@cs.mun.ca

M. Liao
Sharp Laboratories of America, 5750 NW Pacific Rim Blvd.,
Camas, WA 98607, USA
e-mail: liaom@sharplabs.com

1 Introduction

Stereo is one of the most actively researched topics in computer vision. Thanks to the Middlebury benchmark evaluation system [35], recent stereo research in this area has significantly advanced the state-of-the-art in terms of accuracy. However, in terms of speed, the best stereo algorithms typically take several seconds or minutes to compute a disparity map, limiting their applications to off-line processing. There are many interesting applications, such as robot navigation, automatic driving and augmented reality, in which high-quality depth estimation at video frame rate is crucial. For real-time online stereo processing, the options are rather limited that, in general, only local correlation-based approaches [14] and scanline-based optimizations [11] are feasible. Most local approaches, although being fast, are quite fragile and prone to have difficulties within textureless regions or near occlusion boundaries. Scanline-based optimization utilizes dynamic programming (DP) to produce better quality depth estimation. Unfortunately, as each scanline is optimized independently, erroneous horizontal strokes, i.e., the "streaking" artifacts, often arise in the resultant disparity maps.

This work is inspired by the idea of cost-volume filtering via edge-preserving filters, started from the work of Yoon and Kweon [51], which introduces cost aggregation schemes that use a fix-sized window with per-pixel varying support weights. The support weights are computed based on the color similarity and geometric distance to the center pixel of interest. In fact, taking both proximity and similarity into account to construct the filter kernel is the key idea of bilateral filtering [41]. Although bilateral filter-based aggregation methods have proven to be effective, their applications in real-time stereo are limited by their speed. It is nonlinear, and its computational complexity grows quadratically with the kernel size. Brute-force implementations are on the order of minutes for generating a small depth map [51].

In this paper, we attempt to reformulate Yoon and Kweon's [51] aggregation algorithm using a fast separable implementation of the bilateral filter. In the first pass, the raw cost-volume is bilaterally filtered in the horizontal direction using a 1D kernel and the intermediate matching costs are bilaterally filtered subsequently in the vertical direction. This two-pass separable approximation reduces the complexity of the aggregation approach from $O(|I|N\ell^2)$ to $O(|I|N\ell)$, where $|I|$ and N are the image size and disparity search range, respectively and ℓ is the kernel width of a square window. Our approximation leads to fast cost-volume filtering and satisfactory results. Motivated by its suitability for hardware acceleration, we propose a GPU implementation which further improves the speed by one or two orders of magnitude compared to the CPU implementation.

In addition to the GPU-based local "winner-takes-all" (WTA) solution, we further incorporate our two-pass aggregation scheme into a scanline optimization framework for improved reconstruction accuracy. We found that changing the window shape from conventional square to vertical rectangle allows robust performance near depth discontinuities and effectively alleviates DP's scanline inconsistency problem. The smoothed cost-volume is transferred back from the GPU to the CPU memory for DP optimization. Thus, our approach not only makes use of both CPU and GPU in parallel, but also makes each part do what it is best for: the graphics hardware performs cost aggregation in massive parallelism, and the CPU carries out DP that requires more flexible looping and branching capability. The current implementation is capable of running at video frame rate. Quantitative evaluation using data sets with ground truth disparities shows that our approach is among the state-of-the-art real-time stereo algorithms. Combined with its high speed capability, our algorithm is suitable for many real-time applications that require high-quality depth estimates. This stereo formulation that was built on fast approximated bilateral cost-volume smoothing and dynamic programming optimization is the main contribution of this work.

This paper builds upon and extends our previous work [45], with an amended stereo model, detailed description of the algorithm, implementation details, and new experimental comparisons. The rest of paper is organized as follows: after reviewing the related work and background materials of bilateral filtering in Sects. 1.1 and 1.2, we present our stereo algorithmin in Sect. 2. Implementation details of our GPU acceleration scheme are reported in Sect. 3. In Sect. 4, we report experimental results and finally conclude in Sect. 5.

1.1 Related work

A large number of algorithms have been published to address the depth from stereo problem. For the scope of this paper, of particular interest are real-time and bilateral filtering-based algorithms. Interested readers are referred to an excellent survey and evaluation [36] for more detailed taxonomies of dense stereo matching algorithms.

In general, stereo algorithms can be categorized into two major classes: local and global methods. Local methods establish pixel correspondences by measuring the similarity between image regions and usually have fast implementations [13, 42]. Representative early real-time local methods are described elsewhere [21, 24, 31, 49, 50]. The central problem of local correlation-based algorithms is how to determine the size and shape of the aggregation window. For accurate depth estimation, a window must be large enough to cover sufficient intensity variations, while small enough to avoid crossing depth discontinuities. An improper window causes problems such as incorrect disparities in textureless regions and blurred occlusion boundaries. In order to resolve these dilemmas, there has been work on varying window size and shape [3, 17, 20, 22, 43]. The key idea is to evaluate a variety of windows and select the one with the optimal cost. Although performing better than fixed window methods, variable window methods are, in general, less accurate than global methods, which make explicit smoothness assumptions about the scene and minimize certain cost functions [4, 40].

More recently, Yoon and Kweon [51] present a correspondence search algorithm that yields high-quality results comparable to those obtained by global methods. The success of Yoon and Kweon's work [51] lies in the use of joint bilateral filter for cost-volume filtering. The most attractive property of their approach [51] is that a large window can be used to aggregate information without over-blurring occlusion boundaries. On the other hand, the approach used by these authors [51] is computationally demanding. Its execution time is comparable to that required by global methods, diminishing the efficiency advantage of local approaches. For this reason, several *adaptive weights*-based approaches have been proposed, aiming at improving Yoon and Kweon's [51] speed performance. Mattoccia et al. [27] present a block-based aggregation strategy that can obtain a disparity map in a few seconds. Gupta and Cho [16] introduce an adaptive binary window approach. While strong results are demonstrated, their algorithm takes 0.46 s for a 384 $\times$ 288 image with 16 disparity candidates. Yu et al. [52] develop a high performance stereo system using *exponential step size adaptive weight* (ESAW) technique. Their approach demonstrates high data parallelism and can be mapped to a GPU platform. Richardt et al. [33] present an approximated but real-time implementation of the bilateral filtering aggregation method. However, due to the large amount of memory required for processing full-color images, the support weights are computed using grayscale intensities rather than three-channel color vectors, giving poor results near object boundaries. Rhemann et al. [32] present a filter framework which achieves fast and high-quality disparity estimation. Their approach is based on the recently proposed guided filter [19], which has the edge-preserving property and a runtime independent of the filter size.

Besides from local methods, efficient global stereo algorithms have also been studied. Among the various energy minimization methods, DP is of particular interest for real-time applications due to its low computational complexity. Sun [39] proposes an early DP-based stereo algorithm that executes near real-time. The image is divided into nonuniform rectangular subregions to reduce the disparity search range. Gong and Yang [15] present a stereo algorithm based on reliability-based DP. Their algorithm can be implemented on the GPU and yields near real-time performance. By using a coarse-to-fine scheme and MMX instructions, Forstmann et al. [11] present an accelerated DP algorithm whose implementation achieves about 100 MDE/s on an AMD Athlon XP 2800+ 2.2G processor. Traditional DP algorithms optimize the disparity assignments on a scanline by scanline basis and the inter-scanline consistency is not enforced. A number of approaches have been proposed to address this limitation [18, 23, 26, 44]. For example, Kim et al. [23] introduce a two-pass DP scheme that performs optimization both along and across the scanline; Lei et al. [26] optimize a global energy function defined on a 2D tree structure whose nodes are over-segmented image regions. Unfortunately, these advanced approaches, in general, involve more computational cost and are typically slow. In addition to DP, Yang et al. [48] propose a near real-time GPU implementation of the hierarchical belief propagation (BP) algorithm [10]. It produces better accuracy than fast DP-based algorithms but runs slower at about 17 MDE/s. Yu et al. [52] propose a real-time "exponential step size message propagation (ESMP)" algorithm. As an extension of the ESAW method mentioned earlier, by incorporating the smoothness prior, ESMP improves the accuracy at the cost of lower speed in comparison with ESAW.

1.2 Bilateral filter and its application in cost aggregation

Before describing our proposed stereo algorithm, we start with a brief review of bilateral filtering and Yoon and Kweon's adaptive weights stereo algorithm [51].

The bilateral filter is a filtering technique to smooth an image while preserving edges [41]. One of its variants, the joint bilateral filter [25], smoothes an image with respect to edges in a different image. Its basic formulation is very similar to Gaussian convolution: each pixel is replaced by a weighted average of its neighbors. The core difference is that the bilateral filter takes into account the dissimilarity in pixel values with the neighbors when constructing the blur kernel. More formally, given an image I and a central pixel p (we use the notation I_p for the pixel value at position p), the support weight w (p, q) of p's neighbor q is written as:

$$w(p,q) = \exp\left(-\frac{\|I_p - I_q\|}{\sigma_c} - \frac{\|p-q\|}{\sigma_g}\right), \tag{1}$$

where $\|I_p - I_q\|$ and $\|p-q\|$ represent the color dissimilarity (Euclidean distance between pixel values) and the spatial distance between p and q, respectively. The bilateral filter is controlled by two parameters σ_c and σ_g. These two values respectively control the influence from similarity and proximity. An image filtered by a bilateral filter $\mathrm{BF}(\cdot)$ is defined by

$$\mathrm{BF}(I)_p = \frac{\sum_{q\in\Omega_p} w(p,q)\cdot I_q}{\sum_{q\in\Omega_p} w(p,q)}, \tag{2}$$

where Ω_p denotes the set of all pixels in the support region and the normalization factor $\sum_{q\in\Omega_p} w(p,q)$ ensures support weights sum to one. More interesting properties, implementation details, and applications of bilateral filtering can be found in Ref. [29].

Yoon and Kweon [51] utilize the bilateral filtering as an aid in local stereo. Given a pair of stereo images $\{I, I'\}$, the raw matching cost between pixels is written as $\widetilde{C}(p,d)$

where p represents the pixel location in the reference view I, and d is a disparity hypothesis. In [51], the final cost-volume is computed as a weighted average of raw matching costs

$$C(p,d) = \frac{\sum_{q\in\Omega_p, q'\in\Omega'_p} w(p,q)w(p',q')\widetilde{C}(q,d)}{\sum_{q\in\Omega_p, q'\in\Omega'_p} w(p,q)w(p',q')}, \quad (3)$$

where $p' = p - d$ represents $p's$ corresponding pixel in I' given disparity d.

It is worth noting that unlike the conventional bilateral filtering, equation (3) takes into account the support weights in both images. According to the authors' explanation and our experimental observations, employing the support weights in both windows helps to improve correspondence search, especially for pixels near occlusion boundaries. A direct implementation of the proposed method, however, is computationally more expensive than other window-based methods. The reported running time for the benchmark "Tsukuba" image with a 35 × 35 support window is about 1 min on an AMD AthlonXP 2700+ 2.17G processor [51].

Several approaches have been proposed to accelerate the bilateral filtering [5, 8, 28, 46]. They all rely on approximations that yield various degrees of speed–accuracy trade-off. Among the existing approaches, to compute a joint bilateral filter for real-time stereo, the bilateral grid approach proposed in Refs. [5, 8] is feasible because it achieves high-quality outputs and real-time performances even on high-resolution images. For instance, Richardt et al. [33]'s work discussed earlier is built upon the bilateral grid technique.

2 Algorithm description

In this section, we present the proposed stereo formulation. Given multiple images taken from different viewpoints, the goal of a stereo algorithm is to establish pixel correspondences across images. For the scope of this paper, we focus on dense two-frame stereo and assume the input stereo images $\{I, I'\}$ are rectified, i.e., the epipolar lines are aligned with corresponding scanlines. In this case, the correspondence can be expressed as a disparity value d, i.e., if $p(x,y)$ and $p'(x',y')$ are corresponding pixels in the left and right images, respectively, then the disparity between p and p' is defined as the difference of their horizontal image coordinates as $d = x - x'$. Note that $y = y'$ since corresponding pixels must be on the same scanline for rectified images. The output of a stereo algorithm is a disparity map D that stores the disparity value for every pixel in the reference image. In the following, when there is no confusion, we will omit (x,y) and write $d = p - p'$ for conciseness and notation clarity.

Following the taxonomy in [36], our algorithm consists of three major steps: (1) matching cost computation; (2) adaptive cost-volume filtering; and (3) disparity optimization via DP. Details about each module are presented below. Besides from these key components, in all experiments, a 3 × 3 median filter-based disparity refinement step is employed to remove isolated noises from disparity maps.

2.1 Matching cost computation

The matching cost computation step initializes the cost-volume $\widetilde{C}(p,d)$ by computing raw pixel-wise matching costs. In this paper, we use the simple absolute difference (AD) dissimilarity function to measure the difference between two pixels as

$$\widetilde{C}(p,d) = \min\left(\frac{\sum_{c\in\{R,G,B\}} |I_p^c - I_{p-d}^c|}{3}, C_{\max}\right), \quad (4)$$

where I^c is the intensity of the color band c, and the parameter $C_{\max}$ $(0 < C_{\max} \leq 255)$ is a truncation value. The truncation is necessary to make the matching costs robust to occlusion and non-lambertian objects that violate the brightness constancy assumption. For every pixel $p(x,y) \in I$, we loop through all disparity hypotheses to calculate their matching costs using equation (4). In the end, we obtain the initial cost volume $\widetilde{C}$, which is a three-dimensional array that can be indexed by x, y, and d.

2.2 Fast cost-volume smoothing

We modify Yoon and Kweon's [51] approach as our baseline cost aggregation algorithm. Two minor changes are: (1) In Yoon and Kweon's work, the similarity between two pixels within the support window is measured in the CIELab color space. Our approach instead measures the color proximity in the RGB color space for simplicity and efficiency; (2) Inspired by [33], we reformulate equation (1) as

$$w(p,q) = \exp\left(-\frac{\|I_p - I_q\|}{\sigma_c}\right)\sqrt{\exp\left(-\frac{\|p - q\|}{\sigma_g}\right)}, \quad (5)$$

where the square root is applied to the proximity weight, so that $w(p,q)\cdot w(p',q')$ in equation (3) involves the proximity weight only once. In our baseline implementation, we employ a 35 × 35 support window. The running time for the "Tsukuba" sequence is about 25 s on an Intel 2.66 GHz processor with our not fully optimized implementation.

As can be seen, the full-kernel implementation of the bilateral cost-volume filtering is computationally expensive because the support weights need to be recomputed for every pixel. Unfortunately, unlike box and Gaussian filters which have very fast separable implementations, bilateral

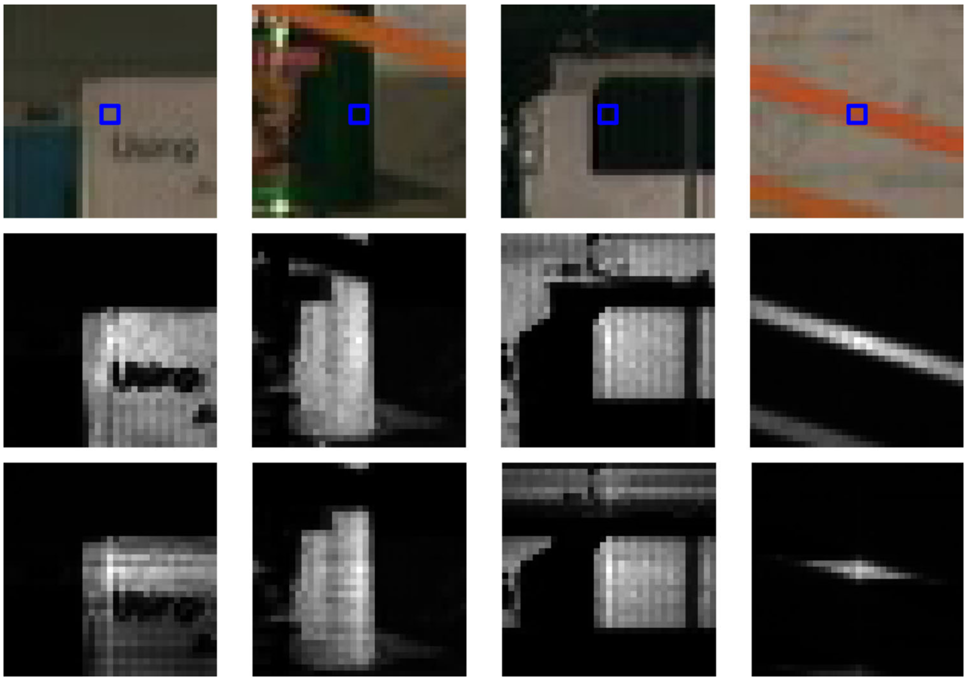

Fig. 1 A comparison of full-kernel with approximated support weights. *Top row* close-up views at several pixel locations in the “Tsukuba” image. The *blue square* marks the center pixel of interest. *Second row* the original 35 × 35 support weights. *Third row* the corresponding support weights computed using our two-pass approximation

filter is not separable in theory due to the color-dependent term in equation (5). Nevertheless, in order to address the speed issue, Pham and Van Vliet [30] attempt to approximate the full-kernel bilateral filter using two separate 1D kernels. Their separable implementation is applied to video enhancement and compression. Ansar et al. [1] first apply bilateral filtering to stereo and conclude that a separable approximation is adequate. However, their work addresses the pre-processing of input imagery rather than cost aggregation. Bilateral filtering is used to smooth the stereo images instead of the cost-volume.

In this paper, we revisit the separable approximation and attempt to speed up the bilateral aggregation process using a two-pass implementation: a 1D bilateral filter is applied to smooth the cost-volume along the first dimension (either horizontal or vertical) and the intermediate results are filtered in the subsequent dimension. In essence, this simplified approach reformulates equation (3) as

$$C^{\mathrm{tmp}}(p,d) = \frac{\sum_{u=x-\frac{\ell}{2}}^{u=x+\frac{\ell}{2}} w[p(x,y),q(u,y)]w(p',q')\widetilde{C}(q,d)}{\sum_{u=x-\frac{\ell}{2}}^{u=x+\frac{\ell}{2}} w[p(x,y),q(u,y)]w(p',q')} \qquad (6)$$

$$C(p,d) = \frac{\sum_{v=y-\frac{\ell}{2}}^{v=y+\frac{\ell}{2}} w[p(x,y),q(x,v)]w(p',q')C^{\mathrm{tmp}}(q,d)}{\sum_{v=y-\frac{\ell}{2}}^{v=y+\frac{\ell}{2}} w[p(x,y),q(x,v)]w(p',q')} \qquad (7)$$

where C^{tmp} is a temporary buffer to store the matching costs obtained from the first pass.

As mentioned above, this separable implementation does not produce exactly the same results as the full-kernel filtering because of the non-separability of $w(\cdot,\cdot)$ in equation (3). Figure 1 shows both the original and approximated support weights for several selected pixels in the “Tsukuba” image. In most cases, especially for uniform areas and axis-aligned edges, the original support weights are very similar to their approximated counterparts. For the rightmost patch which contains two diagonal line structures, our approach still tends to assign higher weights to pixels that are closer or with similar color, but spatially the support weights attenuate much faster compared to the original 2D kernel. In this scenario where there are thin diagonal structures, two-pass approximation is similar to a full-kernel with a smaller support region.

In Fig. 2, we further provide visual and quantitative comparisons of the disparity maps. Compared to the full-kernel filtering, the separable approximation still performs edge-preserving cost-volume smoothing. While visually similar disparity maps can be obtained, as expected, quantitative evaluation with ground truth data confirms that the two-pass approximation yields slightly less accurate results, especially for textured regions. It is worth noting that neither implementation achieves the accuracy numbers as reported elsewhere [51]. We believe the difference is

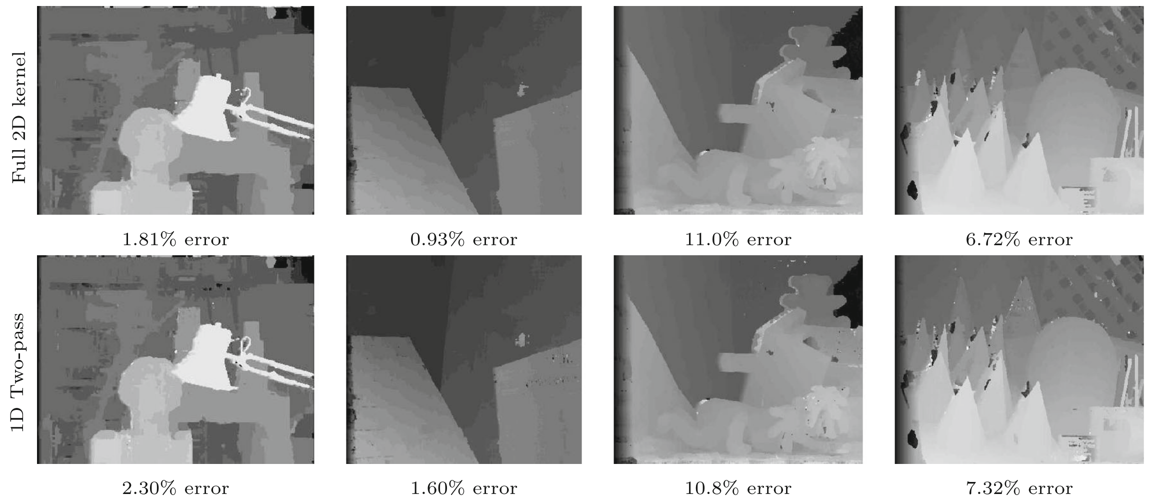

Fig. 2 Disparity maps for the Middlebury benchmark data generated from (*top row*) full-kernel (35 × 35) bilateral cost aggregation and (*bottom row*) the separable two-pass approximation, respectively. Identical parameter settings are used to generate these results. Error disparity percentages are measured in non-occluded areas.

mainly due to the left–right consistency check and occlusion filling post-processing steps employed in [51]. Similar observation and conclusion can also be found in [33]. In terms of speed, this two-pass acceleration dramatically speeds up the computation, reducing the complexity per disparity estimation from $O(\ell^2)$ to $O(\ell)$. For instance, for the "Tsukuba" image, our result is generated in 1.9 s while the full-kernel approach takes about 32 s (kernel width $\ell = 35$). On the other hand, the downside of this approximation is that its resultant disparity maps are less smooth than the brute-force implementation and there is no formal characterization of their differences in accuracy. As a consequence, this two-pass aggregation scheme produces an interesting trade-off between accuracy and speed.

2.3 Disparity optimization via DP

In this section, DP is performed for disparity optimization. As an early framework introduced for the stereo correspondence problem, DP is still one of the most popular techniques for its 1D optimization capability and high efficiency.

DP-based algorithms formulate stereo correspondence as a least-cost path finding problem. Given an image scanline $S_y = \{p(\cdot, y)\}$, DP finds an optimal path through a 2D slice $C(\cdot, y, \cdot)$ of the 3D cost-volume. The optimal path is equivalent to a disparity assignment function $f(p)$ that minimizes the global cost function

$$E(f) = E_{\text{data}}(f) + E_{\text{smooth}}(f), \tag{8}$$

where the first term, the data term,

$$E_{\text{data}}(f) = \sum_{p \in S_y} C[p, f(p)] \tag{9}$$

penalizes disparity assignments that are inconsistent with the observed image data, whereas the smoothness term encourages neighboring pixels to have similar disparities based on the assumption that the scene is piecewise smooth. In this work, $E_{\text{smooth}}(f)$ is defined as

$$E_{\text{smooth}}(f) = \lambda_s \cdot \sum_{p \in S_y} \sum_{q \in \xi_p} \max\left[\exp\left(-\frac{|I_p - I_q|^2}{\sigma_s}\right), \epsilon\right] \cdot \min[|f(p) - f(q)|, \tau], \tag{10}$$

where λ_s is the rate of increase in the smoothness cost; $\xi_{p(x,y)} = \{p(x-1, \ y), \ p \ \ (x+1, \ y)\}$ and $\exp(-|I_p - I_q|^2/\sigma_s)$ is a monotonically decreasing function of intensity differences that lowers smoothness penalty costs at high intensity gradients; parameters σ_s and ϵ control the sharpness and lower bound of the exponential function, respectively. In order to allow for sharp depth edges, the smoothness cost stops growing after the disparity difference becomes large. Parameter τ controls the upper bound of discontinuity penalty between neighboring pixels.

Energy functions with the form defined in equation (8) can be minimized by DP. For each scanline S_y in the reference view, we construct a cost matrix M and an ancestor matrix A. Both M and A have $N \times W$ entries, where N and W represent the disparity range and image width, respectively. Each entry is a potential place along the path. We traverse M from left-to-right updating the entries in M and A. The complexity of the

Algorithm 1 Three-state DP for optimal path extraction

```
1: for $d = 0$ to $N - 1$ do
2:     $M(d, 0) = C(0, y, d)$;
3: end for
4: for $x = 1$ to $W - 1$ do
5:     compute the smoothness cost $\lambda$ between $(x - 1, y)$ and $(x, y)$ based on equation (10);
6:     $nvocc = 0$;
7:     for $d = N - 1$ to 0 do
8:         $cmin0 = C(x, y, d) + M(d, x - 1)$; //match state
9:         $cmin1 = C(x, y, d) + M(d - 1, x - 1) + \lambda$; //diagonal occlusion
10:        $cmin2 = M(d + 1, x) + (nvocc < \tau\ ?\ \lambda : 0)$; //vertical occlusion
11:        $M(d, x) = \min(cmin0, cmin1, cmin2)$;
12:        $if(M(d, x) == cmin0)\ A(d, x) = (d, x - 1);\ nvocc = 0$;
13:        $if(M(d, x) == cmin1)\ A(d, x) = (d - 1, x - 1);\ nvocc = 0$;
14:        $if(M(d, x) == cmin2)\ A(d, x) = A(d + 1, x);\ nvocc = nvocc + 1$;
15:    end for
16: end for
```

brute-force implementation is $O(WN^2)$ per-scanline since updating $M(d,x)$ requires considering N previous entries $M(0,x-1)\ldots M(N-1,x-1)$. Inspired by others' work [2, 6], we impose the common occlusion and monotonic ordering constraints [12] and employ the *three-state* (horizontal match, diagonal occlusion, and vertical occlusion states) scanline optimization algorithm as outlined in Algorithm 1 to construct the optimum path. By assuming the ordering rule, three instead of N potential moves need to be considered, which significantly reduce the complexity of the pathfinding problem. After the rightmost column is filled, the optimum path can be extracted via back-tracking [6]. This DP process is repeated over all the scanlines to generate a dense disparity map.

DP's scanline-independent computational structure allows us to easily take advantage of the multi-core architectures that are ubiquitous in today's CPUs. In our DP implementation, the reference image is evenly divided into K sub-regions, where $K \in \{1, 2, \ldots\}$ and the region size is $W \times \lceil H/K \rceil$ (H represents the image height). At runtime, multiple threads are evoked with each thread assigned to a region and there is no data interchange between regions during the energy minimization process. As will be later reported in Sect. 4, exploiting thread-level parallelism enables a significant speed performance improvement without any loss in accuracy.

2.4 Vertical aggregation

Global approaches usually use the raw pixel-based matching costs and skip the aggregation step. In this paper, we instead combine the strengths of the edge-preserving cost-volume smoothing and the DP optimization framework to achieve high accuracy depth estimation. Motivated by DP's well-known difficulty of enforcing inter-scanline consistency (resulting in horizontal "streaks" in the estimated disparity maps), we enforce vertical smoothness by constructing the data term with an approximated $\ell_y \times \ell_x$ rectangular bilateral filter (aggregation window), where $\ell_y \geq \ell_x$ guarantees that the dominant aggregation direction is orthogonal to image scanlines. After vertical aggregation, pixels between scanlines are likely to have a more consistent cost if they are sharing the same color (and therefore, more likely to belong to the same surface). The inter-scanline consistency of DP can be better enforced in the final disparity map.

Figure 3 illustrates the results of combining vertical smoothing and DP optimization. With a 5×1 Gaussian filter, noise and "streaking" artifacts are reduced compared to performing DP alone (no cost aggregation step applied). However, pixels near occlusion boundaries tend to be blurred and thin structures are not well preserved (e.g., lamp bar in Fig. 3b). Similar observation is reported

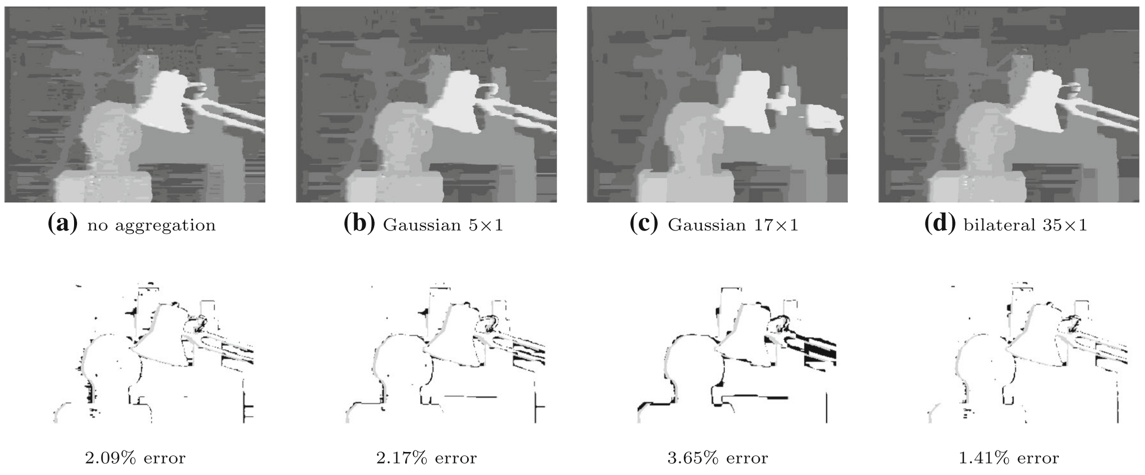

Fig. 3 Comparison of cost-volume smoothing with Gaussian and bilateral filtering. Disparity maps are computed using DP after the aggregation step. *Top row* (**a**)–(**c**): disparity maps from $\ell \times 1$ support window with Gaussian weights, where (**a**) $\ell = 1$, (**b**) $\ell = 5$, and (**c**) $\ell = 17$, respectively. Disparity (**d**) is obtained from 35×1 bilateral filtering aggregation. Quantitative error rates in non-occluded regions (bad pixels labeled in *black*) are given in the *bottom row*.

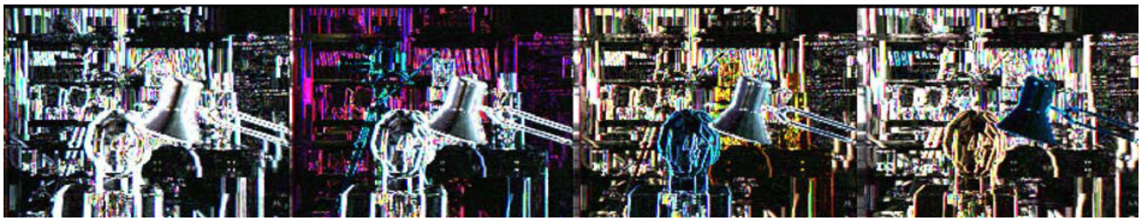

Fig. 4 The texture used to store matching costs. The four-color channels of a single pixel in the texture store the matching costs of a pixel under four different disparity hypotheses.

in [11], in which the costs from the previous scanline is aggregated. With a large 17×1 Gaussian kernel, the disparity map is smoothed at the cost of occlusion boundaries being substantially blurred. In contrast, using a 17×1 bilateral filter can preserve sharp object boundaries, suppress noise, and enforce inter-scanline consistency.

3 Acceleration using graphics hardware

To achieve real-time performance, we take advantage of GPU's massively data parallel architectures and implement the matching cost computation and cost-volume smoothing steps on graphics hardware to improve the computation speed of our algorithm.

In the matching cost computation stage, the input stereo images are stored as two textures. For each disparity hypothesis d, we draw a 2D rectangle aligned with two input textures, one of them being shifted horizontally by d pixels. We use the pixel shader, a programmable unit in the graphics hardware, to compute the per-pixel absolute difference and the results are written to an output texture. Since the graphics hardware is most efficient at processing four-channel (RGB + alpha) color images, we compute four disparity hypotheses at a time and store the absolute-difference images in different channels. To search over N disparity hypothesis, $\lceil N/4 \rceil$ rendering passes are needed.

Similar to existing real-time stereo GPU implementations [14, 15], the matching costs obtained are stored as 8-bit integers in GPU memory instead of floating points for lower computational overhead. Representing matching costs with 8 bits makes accurate disparity estimation more challenging since small cost differences cannot be presented due to the limited precision. In our GPU implementation, the matching costs in equation (4) are truncated and scaled to make better use of the range of a single byte as

$$\widetilde{C}(p,d) = \min\left(\frac{\sum_{c\in\{R,G,B\}} |I_p^c - I_{p-d}^c|}{3}, C_{\max}\right) \cdot \frac{255}{C_{\max}}. \tag{11}$$

After truncating and scaling, the resultant 3D cost-volume is stored as a stack of 2D images. Four adjacent disparity hypotheses are packed into one color image to utilize the vector processing capacity of GPUs. The color images are tiled together to form a large matching cost texture. An example is shown in Fig. 4.

For the cost aggregation step, we first compute the per-pixel adaptive weights for both images. Similar to the cost computation process, we shift the image over itself to compute the pixel-wise weights according to equation (1) and store them in textures. The 1D kernel width is always set to a multiple of four to facilitate the four-vector processing capability on GPU. After computing the weights for bilateral filters, we can step through the cost-volume to compute the weighted average. A fairly complex pixel shader program is implemented to index into both the matching cost textures and weighting textures to calculate the final cost. Aggregating over N disparity hypotheses with an approximated $\ell_y \times \ell_x$ bilateral filtering kernel requires $\lceil N \cdot (\ell_y + \ell_x)/16 \rceil$ rendering passes in our implementation.

The advantage of using graphics hardware mainly comes from the parallelism inherent in today's GPU. Both cost computation and aggregation are regular per-pixel operations that can benefit most from GPU's parallel architecture. The smoothed cost-volume can be used by a WTA selection scheme on GPU (as in [14]), or it can be read back to CPU memory for CPU processing using DP. It should be noted that it is possible to implement the entire DP optimization process on GPU. However, as reported [15], a GPU-based DP implementation is actually slower than its CPU counterpart. This is mainly due to the significant number of rendering passes needed and the lack of true branching capability on GPU [15]. Therefore, we adopted a co-operative approach, using the GPU to compute the cost volume and the CPU to carry out DP. With the new PCI-Express interface between CPU and GPU, the communication bandwidth is huge, removing a long-existing bottleneck between GPU and CPU. Our approach not only makes use of both CPU and GPU in parallel, but also makes each part do what it is best at: the GPU performs image warping and aggregation in massive parallelism while the CPU carries out DP which requires more flexible looping and branching capability.

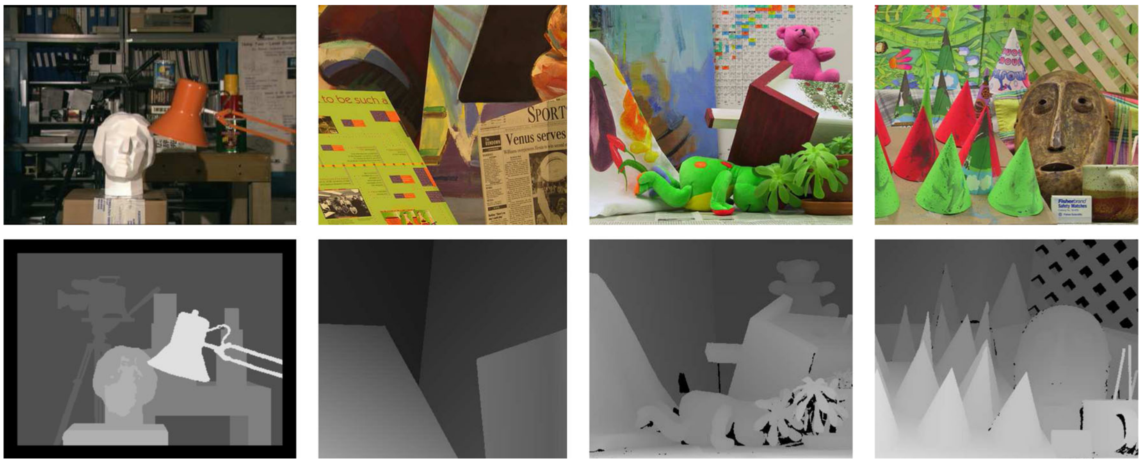

Fig. 5 Reference images of "Tsukuba", "Venus", "Teddy" and "Cones" stereo pairs and their ground truth disparities.

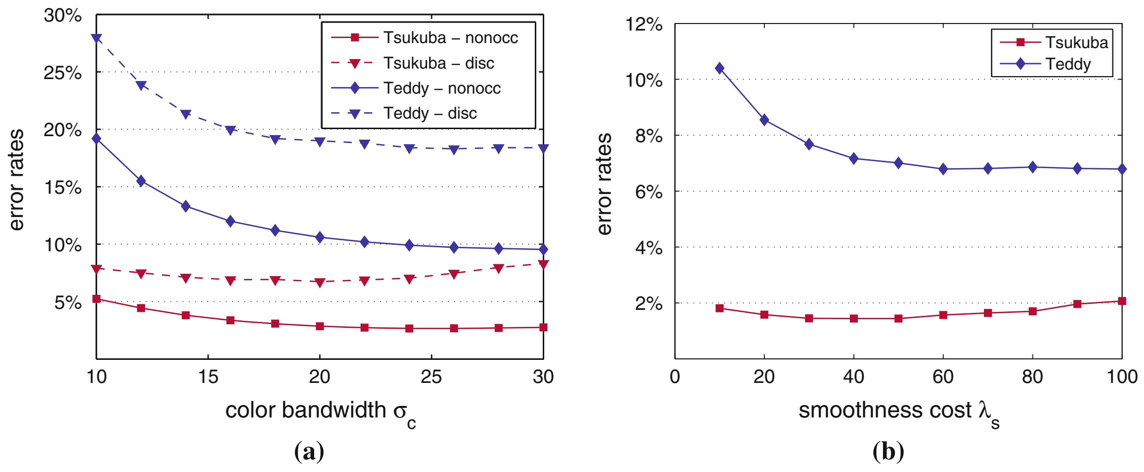

Fig. 6 **a** Error rate with respect to the color bandwidth σ_c for bilateral filtering (Eq. 1). Statistics in non-occluded regions (nonocc) and areas near depth discontinuity boundaries (*disk*) are both reported. Disparity maps are generated using WTA and two-pass (35×35) bilateral aggregation; **b** error rate as a function of the smoothness penalty cost λ_s (Eq. 10). Disparity maps are generated using DP and vertical (35×1) bilateral aggregation

4 Experiments

4.1 Static images

In this section, we verify the effectiveness of the proposed algorithm and report experimental results. Our experiments were conducted on a 4× Intel Xeon(R) 2.67 Ghz CPU with a GeForce GTX 580 GPU from NVIDIA. To enable a quantitative evaluation of different algorithms, we adopted the widely used stereo data sets with ground truth disparity maps available [36, 37] (as shown in Fig. 5). The reconstruction accuracy is measured by the percentages of bad matching (where the absolute disparity error is greater than 1 pixel) via the Middlebury evaluation system.

The main parameters in our algorithm can be divided into three sets: (1) truncation value $\{C_{\max}\}$ for matching cost computation; (2) four parameters $\{\sigma_c, \sigma_g, \ell_x, \ell_y\}$ for cost aggregation; and (3) $\{\sigma_s, \epsilon, \lambda_s, \tau\}$ for disparity selection using DP. Following the experimental observations in [14], $C_{\max}$ is set to 25 throughout. Parameters σ_c and σ_g are color and spatial bandwidths for the bilateral filtering, respectively. Figure 6a shows the performance of two-pass aggregation for the "Tsukuba" and "Teddy" images as a function of σ_c. In this experiment, we keep the width of the support window and σ_g constant, $\ell_x = \ell_y = 35$, $\sigma_g = 17.5$ (radius of the support window), and use WTA to select the disparities. Note that besides from error rates in non-occluded areas, we also plot error percentages for pixels near depth discontinuities to assess the parameter's edge-

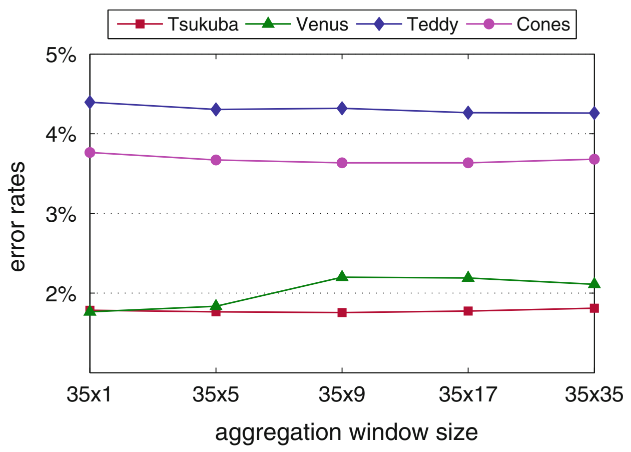

Fig. 7 Error rate with respect to different aggregation window sizes. Disparity maps are generated using DP

preserving capability. In this paper, σ_c is set to 20 for all test images according to the results learned from this plot.

Among the four-DP parameters, σ_s, ϵ and τ are less sensitive and we empirically set $\sigma_s = 400$, $\epsilon = 0.4$ and $\tau = 2$. To determine λ_s, namely the rate of increase in the smoothness cost, we set $\ell_y = 35$, $\ell_x = 1$ and plot the error rates with respect to λ_s in Fig. 6b. Note that we use truncated and scaled matching costs in equation (11) for these experiments. As can been seen, the optimal λ_s varies for different images. Fortunately, $\lambda_s \in [40, 60]$ typically generates good results. For Middlebury quantitative evaluation, we fix $\lambda_s = 60$.

Finally, we evaluate the effects of cost aggregation using windows with different sizes. In Fig. 7, we fix kernel height $\ell_y = 35$ and plot the error rates as a function of width ℓ_x. It is worth noticing that since DP performs horizontal optimization, we let $\ell_y \geq \ell_x$ to ensure the dominant aggregation direction is orthogonal to image scanlines. Figure 7 suggests that increasing the width of the support window, in general, tends to marginally improve the accuracy. When $1 < \ell_x \leq 35$ in three of the four data sets, approximated bilateral filtering achieves better (or comparable) results compared to the 1D vertical smoothing. And for the "Venus" sequence, the increase in error is mainly caused by the constant parameter setting $\lambda_s = 60$ used in our experiments, which is considered to be too large for "Venus". On the other hand, it also reveals the risk of over-smoothing the results when performing both 2D aggregation and DP optimization.

Using the Middlebury online system at [35], we compare our method against other relevant stereo algorithms listed in the evaluation table and summarize the results in Table 1. With DP optimization, the vertical aggregation window is set to 35 $\times$ 1 for the CPU implementation (VAggCPU+DP) or 32 $\times$ 1 for the GPU counterpart (VAggGPU+DP). For two-pass bilateral aggregation with WTA disparity selection, 35 $\times$ 35 and 32 $\times$ 32 windows are used by CPU (2PassAggCPU) and GPU (2PassAggGPU) implementations, respectively. The average percent of bad pixels in non-occluded regions in the second column is used as the metric by which the table is sorted. Corresponding disparity maps from our approach are shown in Fig. 8. In addition to quantitative error percentage, run time in MDE/s is also reported to provide readers with a more clear picture of the compared algorithms[1]. We provided more details on runtime analysis in Section 4.2.

The VAggGPU+DP algorithm outperforms other DP-based real-time or near real-time solutions [15, 34, 44, 45] in terms of both matching accuracy and speed. There are two DP-based approaches [7, 26] (not listed in Table 1) that yield better accuracy than ours. However, they both require color segmentation and are typically slow for real-time applications. In comparison with [48] which performs full-frame optimization via BP, our proposed algorithm can achieve much higher throughput at comparable accuracy. Another near real-time BP-based algorithm [47] relies on color segmentation and plane fitting. Even through with the segmentation and BP implemented on a GPU, it is much slower than our approach. Compared to most edge-preserving filter-based local methods [16, 27, 33, 52, 53], our proposed algorithm achieves better trade-off between accuracy and efficiency. Our accuracy falls behind a GPU local method [32]. Note that Rhemann et al. [32] refines the final disparity maps by employing advanced post-processing steps such as mutual consistency check [9] (required to compute both left and right disparity maps) and hole filling. For results reported in this paper, only a 3 $\times$ 3 median filtering is applied to refine the disparity maps. Incorporating effective and efficient disparity refinement step into our existing stereo framework is a future research direction. The approximated 2PassAggGPU approach can produce reasonably accurate disparity maps in real-time. Compared to VAggGPU+DP, although being less accurate, it has the advantage that the computations are completely carried out by the GPU, leaving the CPU free to handle other tasks.

Our GPU implementation of the bilateral aggregation attains an average speed-up factor of 245 compared to its CPU counterpart, with some sacrifice in accuracy. The principal source of accuracy loss is our choice of GPU precision. Although the pixel shader performs computation in 32-bit floating point numbers, we store the filtered

[1] It is important to note that some previous methods are not implemented on the same processors. The MDE/s numbers reported in this table do not reflect a fair comparison across different platforms, but only a reference for the readers.

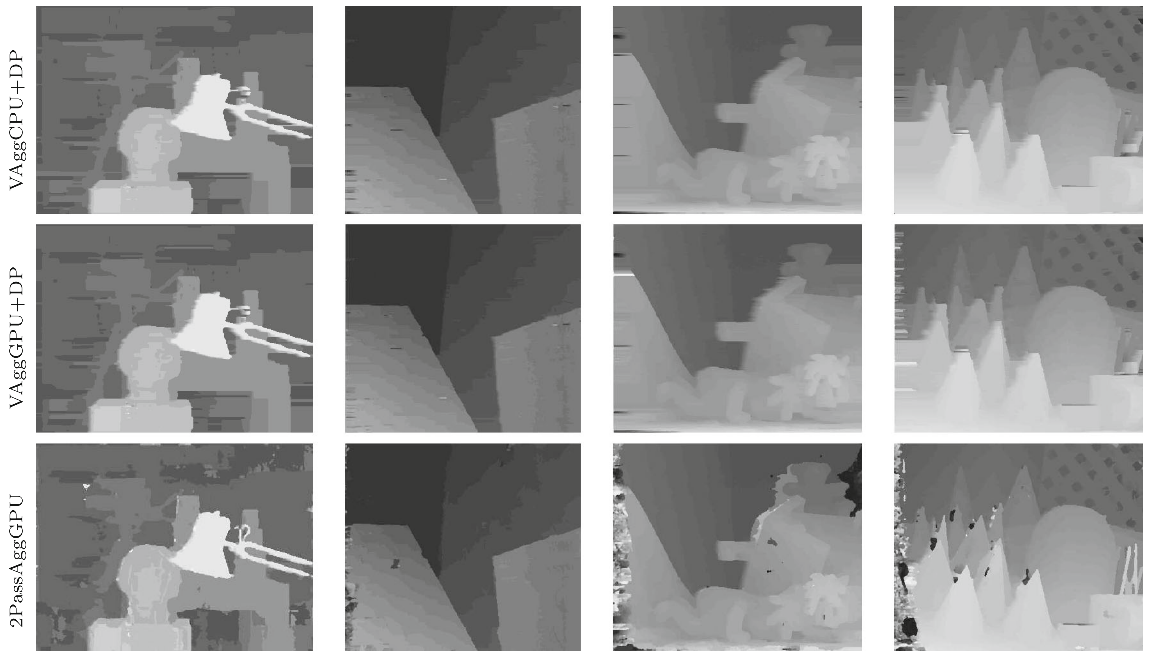

Fig. 8 Disparity maps for the Middlebury benchmark data generated from full-kernel (35 × 35) bilateral cost aggregation (*top row*) and the separable two-pass approximation (*bottom row*). Identical parameter settings are used to generate these results. Error disparity percentages are measured in non-occluded areas.

Table 1 Accuracy and speed comparison of related stereo algorithms in the Middlebury online evaluation system [35]

Algorithm	Non-occ error %				Avg. error %	MDE/s
	Tsukuba	Venus	Teddy	Cones		
CostFilter [32]	1.51	0.20	6.16	2.71	2.65	145.7
PlaneFitBP [47]	0.97	0.17	6.65	4.17	2.99	9.4
VAggCPU+DP	1.57	1.53	6.79	5.53	3.86	2.62
RealtimeBP [48]	1.49	0.77	8.72	4.61	3.90	20.9
FastBilateral [27]	2.38	0.34	9.83	3.10	3.91	0.3
VAggGPU+DP	1.57	1.47	6.93	6.07	4.01	155.7
OptimizedDP [34]	1.97	3.33	6.53	5.17	4.25	19.0
RealtimeABW [16]	1.26	0.33	10.7	4.81	4.28	3.9
RealtimeGPU [45]	2.05	1.92	7.23	6.41	4.40	52.8
2PassAggCPU	1.47	1.40	9.48	5.27	4.41	1.43
ESAW [52]	1.92	1.03	8.48	6.56	4.50	194.8
RealtimeBFV [53]	1.71	0.55	9.90	6.66	4.71	106.9
2PassAggGPU	1.66	1.86	10.3	5.47	4.82	350.1
DCBGrid [33]	5.90	1.35	10.5	5.34	5.77	133.6
ReliabilityDP [15]	1.36	2.35	9.82	12.9	6.61	20.0

Note that our DP implementation uses two threads to process the upper and lower part of the image, respectively in parallel

VAggCPU+DP, dynamic programming with CPU-based vertical bilateral aggregation (35 × 1); VAggGPU+DP, dynamic programming with GPU-based vertical bilateral aggregation (32 × 1); 2PassAggCPU, two-pass CPU-based approximated bilateral aggregation (35 × 35); 2PassAggGPU, two-pass GPU-based approximated bilateral aggregation (32 × 32)

matching costs in 8-bit textures for lower computational and read-back overhead. Although this precision problem can be addressed by truncating-then-scaling the original matching costs obtained, the resulting algorithms are more sensitive to the selection of the truncation value than the corresponding CPU implementations. And also note that the parameters are tuned for the CPU implementations only and may not be optimal for their GPU counterparts.

Table 2 Real-time performance

Image Size	# of Disp.	Runtime MDE/s					
		CPU		GPU	VAggGPU+DP		
		2PassAggCPU	VAggCPU+DP	2PassAggGPU	1 Core	2 Cores	4 Cores
320 × 240	16	1.38	2.59	292.1	90.4	147.5	235.9
	32	1.56	2.90	326.9	99.4	164.2	253.6
640 × 480	16	1.28	2.42	353.9	97.6	158.8	248.2
	32	1.48	2.55	427.6	102.8	173.8	270.0

The test system is a 4× Intel Xeon(R) 2.67 Ghz processor with a GeForce GTX 580 graphics card from NVIDIA.

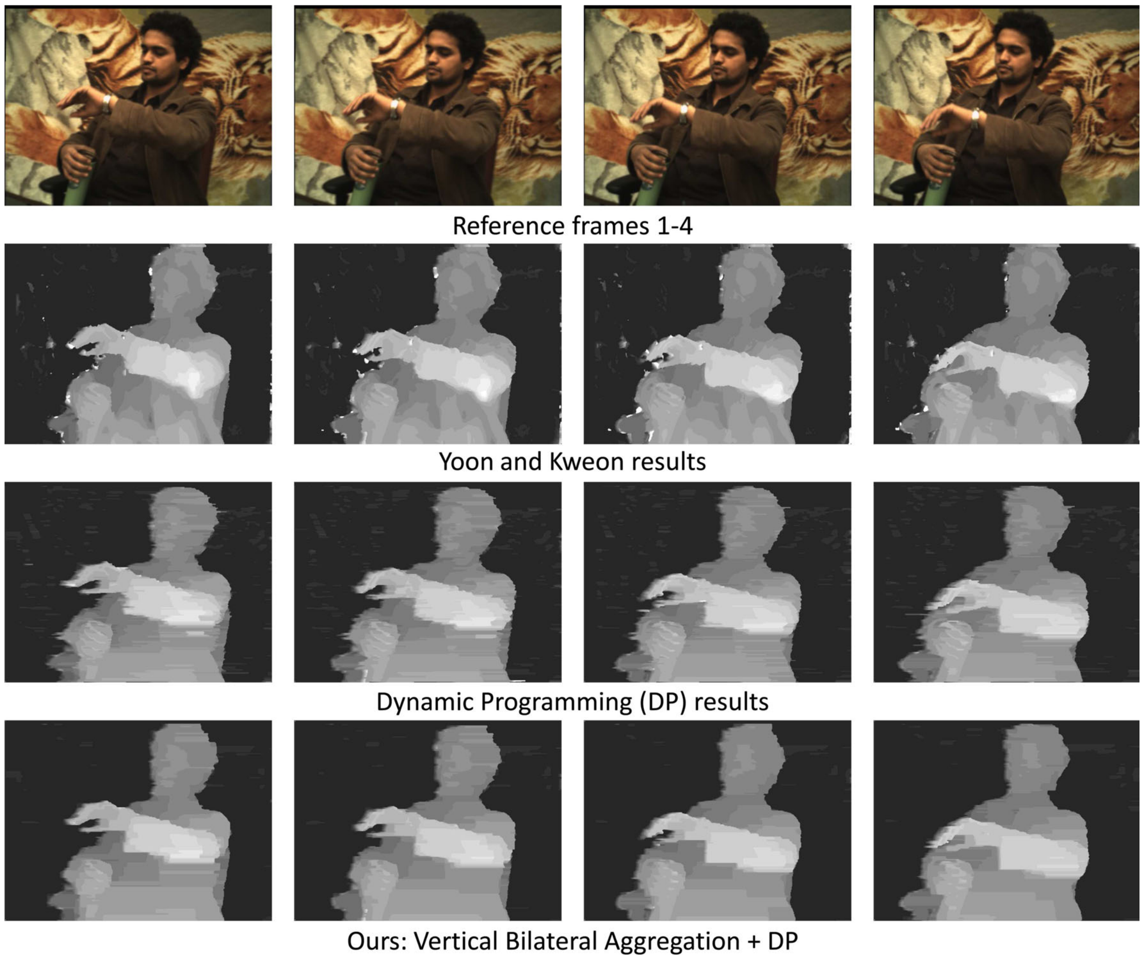
Reference frames 1-4
Yoon and Kweon results
Dynamic Programming (DP) results
Ours: Vertical Bilateral Aggregation + DP

Fig. 9 Selected disparity maps for a stereo video of dynamic scene (this data set was publicized by [38]). *First row* reference frames 1–4 of the stereo video. *Second row* results obtained using our implementation of Yoon and Kweon's bilateral aggregation algorithm [51]. *Third row* results from the three-state DP algorithm similar to [2]. *Last row* results from vertical bilateral aggregation (32 × 1) and DP optimization. A 3 × 3 median filter is applied to refine the disparity maps for all three approaches. Note the improved spatial and temporal consistency from our algorithm.

4.2 Video sequences of dynamic scenes

In addition to performing well on static stereo images, we have applied our method to stereo videos of dynamic scenes. Even though the videos are processed on a frame by frame basis without incorporating temporal smoothness constraints, Fig. 9 shows that combining vertical bilateral aggregation and DP yields more temporally coherent depth estimation than using either edge-preserving filtering [51] or DP optimization [2].

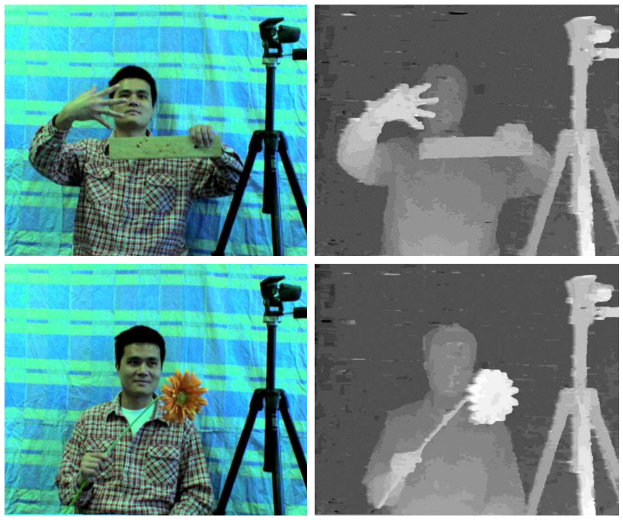
Fig. 10 Two sample images and their depth maps from our live system on a four-core 2.66 GHz PC with a NVIDIA's GeForce GTX 580 graphics card. Our algorithm can achieve 192 fps with 340×240 input images and 16 disparity levels.

We also integrated our algorithm into a stereo system with live video input. The input images are rectified with lens distortion removed. This preprocessing is implemented on the graphics hardware using texture mapping functions. Figure 10 shows some live images from our system. Notice the fine structures and clean object boundaries that our approach is able to produce. The speed performance with respect to different image resolutions and disparity ranges is summarized in Table 2. Our GPU-accelerated version is two orders of magnitude faster than its CPU counterpart. Exploiting thread-level parallelism is a simple yet effective way to speed-up DP and we point out that multi-thread technique could also be coupled with data-level parallelism (i.e., executing the same operation on multiple data using the SIMD instructions), currently supported by most off-the-shelf processors, so as to obtain further (2-3 times) speed up.

5 Conclusion and future work

In this paper, we present a stereo framework that operates at real-time while still estimating high-quality depth information for live stereo video sequences. Our proposed algorithm combines edge-preserving cost-volume filtering and DP optimization. The use of a color and distance-weighted cost aggregation window in the vertical direction significantly reduces DP's "streaking" artifacts. Experimental results show that it is among the best performing real-time stereo algorithms in terms of both disparity estimation accuracy and efficiency. In addition, an approximation for the 2D bilateral aggregation is developed, which leads to a fully GPU-accelerated implementation to achieve two orders of speed-up compared to the original approach in [51]. This simplified approach can produce reasonably accurate disparity maps in real time.

Looking into the future, optimizing DP using SIMD instructions (as in [11]) will further improve the speed performance. We would also like to investigate the precision issue on the graphics hardware. Current graphics hardware does provide limited support for high-precision texture maps, at the cost of significant performance degradation (the hardware is optimized to work with 8-bit textures). From an algorithmic standpoint, our DP implementation enforces the ordering constraint for speed consideration. Very fine 3D structures (e.g., the flower stem in Fig. 10) may disappear if it is far away from the background objects. We plan to investigate the use of scanline optimization [36], which enforces the smoothness constraint directly without employing the ordering constraint. Another interesting venue to explore is to enforce the temporal consistency in the video to reduce the flickering artifacts in the final disparity maps.

Acknowledgments The authors would like to thank the editor and all the reviewers for their constructive comments. This work is supported in part by University of Kentucky Research Foundation, US National Science Foundation award IIS-0448185, CPA-0811647, MRI-0923131, National Science Foundation of China grant No. 60872069, and Zhejiang Provincial Natural Science Foundation of China grant 2011C11053.

References

1. Ansar, A., Castano, A., Matthies L.: Enhanced real-time stereo using bilateral filtering. In: Proceedings of the International Symposium on 3D Data Processing, Visualization and Transmission (2004)
2. Bobick, A.F., Intille, S.S.: Large occlusion stereo. Int. J. Comput. Vis. **33**(2), 181–200 (1999)
3. Boykov, Y., Veksler, O., Zabih, R.: A variable window approach to early vision. IEEE Trans. Pattern Anal. Mach. Intell. **20**(12), 1283–1294 (1998)
4. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Trans. Pattern Anal. Mach. Intell. **23**(11), 1222–1239 (2001)
5. Chen, J., Paris, S., Durand F.: Real-time edge-aware image processing with the bilateral grid. In: Proceedings of the ACM Siggraph Conference (2007)
6. Cox, I.J., Hingorani, S.L., Rao, S.B., Maggs, B.M.: A maximum likelihood stereo algorithm. Comput. Vis. Image Underst. **63**(3), 542–567 (1996)
7. Deng, Y., Lin, X.: A fast line segment based dense stereo algorithm using tree dynamic programming. In: Proceedings of the European Conference on Computer Vision (2006)
8. Durand, F., Dorsey, J.: Fast bilateral filtering for the display of highdynamic-range images. In: Proceedings of the ACM Siggraph Conference (2002)

9. Egnal, G., Wildes, R.P.: Detecting binocular half-occlusions: empirical comparisons of five approaches. IEEE Trans. Pattern Anal. Mach. Intell. **24**(8), 1127–1133 (2002)
10. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient belief propagation for early vision. Int. J. Comput. Vis. **70**(1), 41–54 (2006)
11. Forstmann, S., Ohya, J., Kanou, Y., Schmitt, A., Thuering, S.: Real-time stereo by using dynamic programming. In: Proceedings of CVPR Workshop on Real-time 3D Sensors and Their Use (2004)
12. Geiger, D., Ladendorf, B., Yuille, A.: Occlusions and binocular stereo. Int. J. Comput. Vis. **14**(3), 211–226 (1995)
13. Georgoulas, C., Andreadis, I.: A real-time fuzzy hardware structure for disparity map computation. J. Real-Time Image Process. **6**(4), 257–273 (2011)
14. Gong, M., Yang, R., Wang, L., Gong, M.: A performance study on different cost aggregation approaches used in real-time stereo matching. Int. J. Comput. Vis. **75**(2), 283–296 (2007)
15. Gong, M., Yang, Y.-H.: Near real-time reliable stereo matching using programmable graphics hardware. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (2005)
16. Gupta, R.K., Cho, S.-Y.: Real-time stereo matching using adaptive binary window. In: Proceedings of International Symposium on 3D Data Processing, Visualization and Transmission (2010)
17. Gupta, R.K., Cho, S.-Y.: A correlation-based approach for real-time stereo matching. In: Proceedings of International Conference on Advances in Visual Computing (2010)
18. Gutemberg, G.-F.: An optimal time space algorithm for dense stereo matching. J. Real-Time Image Process. **7**(2), 69–86 (2012)
19. He K., Sun J., and Tang X.: Guided image filtering. In: Proceedings of European Conference on Computer Vision (2010)
20. Kanade, T., Okutomi, M.: A stereo matching algorithm with an adaptive window: theory and experiment. IEEE Trans. Pattern Anal. Mach. Intell. **16**(9), 920–932 (1994)
21. Kanade, T., Yoshida, A., Oda, K., Kano, H., Tanaka M.: Stereo method for video-rate dense depth mapping and its new applications. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (1996)
22. Kang, S.B., Szeliski, R., Chai, J.: Handling occlusions in dense multi-view stereo. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (2001)
23. Kim, J.C., Lee, K.M., Choi, B.T., Lee, S.U.: A dense stereo matching using two-pass dynamic programming with generalized ground control points. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (2005)
24. Kimura, S., Shinbo, T., Yamaguchi, H., Kawamura, E., Naka, K.: A convolver-based real-time stereo machine (sazan). In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (1999)
25. Kopf, J., Cohen, M.F., Lischinski, D., Uyttendaele, M.: Joint bilateral upsampling. In: Proceedings of ACM Siggraph Conference (2007)
26. Lei, C., Selzer, J., Yang, Y.-H.: Region-tree based stereo using dynamic programming optimization. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (2006)
27. Mattoccia, S., Giardino, S., Gambini, A.: Accurate and efficient cost aggregation strategy for stereo correspondence based on approximated joint bilateral filtering. In: Proceedings of Asian Conference on Computer Vision (2009)
28. Paris, S., Durand, F.: A fast approximation of the bilateral filter using a signal processing approach. Int. J.Comput. Vis. **81**(1), 24–52 (2009)
29. Paris, S., Kornprobst, P., Tumblin, J., Durand, F.: Bilateral filtering: theory and applications. Found. Trends Comput. Graph. Vis. **16**(9), 1–73 (2009)
30. Pham, T.Q., van Vliet L.J.: Separable bilateral filtering for fast video preprocessing. In: Proceedings of IEEE International Conference on Multimedia and Expo (2005)
31. Digiclops trinocular stereovision system. http://www.ptgrey.com/products/triclopsSDK/triclops.pdf
32. Rhemann, C., Hosni, A., Bleyer, M., Rother, C., Gelautz, M.: Fast cost-volume filtering for visual correspondence and beyond. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (2010)
33. Richardt, C., Orr, D., Davies, I., Criminisi, A., Dodgson, N.: Real-time spatiotemporal stereo matching using the dual-cross-bilateral grid. In: Proceedings of European Conference on Computer Vision (2010)
34. Salmen, J., Schlipsing, M., Edelbrunner, J., Hegemann, S., Lueke, S.: Real-time stereo vision: making more out of dynamic programming. In: Proceedings of International Conference on Computer Analysis of Images and Patterns (2009)
35. Scharstein, D., Szeliski, R. http://vision.middlebury.edu/stereo/
36. Scharstein, D., Szeliski, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. Int. J. Comput. Vis. **47**(1), 7–42 (2002)
37. Scharstein, D., Szeliski, R.: High-accuracy stereo depth maps using structured light. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (2003)
38. Smith, B.M., Zhang, L., Jin, H.: Stereo matching with nonparametric priors in feature space. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (2009)
39. Sun, C.: Fast stereo matching using rectangular subregioning and 3D maximum-surface techniques. Int. J. Comput. Vis. **47**(1–3), 99–177 (2002)
40. Sun, J., Zheng, N.-N., Shum, H.-Y.: Stereo matching using belief propagation. IEEE Trans. Pattern Anal. Mach. Intell. **25**(7), 787–800 (2003)
41. Tomasi, C., Manduchi, R.: Bilateral filtering for gray and color images. In: Proceedings of IEEE International Conference on Computer Vision (1998)
42. Tombari, F., Mattoccia, S., Stefano, L.D., Addimanda, E.: Classification and performance evaluation of different aggregation costs for stereo matching. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (2008)
43. Veksler, O.: Fast variable window for stereo correspondence using integral images. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (2003)
44. Veksler, O.: Stereo correspondence by dynamic programming on a tree. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (2005)
45. Wang, L., Liao, M., Gong, M., Yang, R., Nister, D.: High-quality real-time stereo using adaptive cost aggregation and dynamic programming. In: Proceedings of International Symposium on 3D Data Processing, Visualization and Transmission (2006)
46. Weiss, B.: Fast median and bilateral filtering. In: Proceedings of ACM Siggraph Conference (2006)
47. Yang, Q., Engels, C., Akbarzadeh, A.: Near real-time stereo for weakly-textured scenes. In: Proceedings of British Machine Vision Conference (2008)
48. Yang, Q., Wang, L., Yang, R., Wang, S., Liao, M. Nister, D.: Real-time global stereo matching using hierarchical belief propagation. In: Proceedings of British Machine Vision Conference (2006)
49. Yang, R., Pollefeys, M.: Multi-resolution real-time stereo on commodity graphics hardware. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (2003).
50. Yang, R., Pollefeys, M., Yang, H., Welch, G.: A unified approach to real-time, multi-resolution, multi-baseline 2D view synthesis and 3D depth estimation using commodity graphics hardware. Int. J. Image Graph. **4**(4), 1–25 (2003)
51. Yoon, K.-J., Kweon, I.-S.: Adaptive support-weight approach for correspondence search. IEEE Trans. Pattern Anal. Mach. Intell. **28**(4), 650–656 (2006)

52. Yu, W., Chen, T., Franchetti, F., Hoe, J.C.: High performance stereo vision designed for massively data parallel platforms. IEEE Trans. Circuits Syst. Video Technol.**20**(11), 1509–1519 (2009)
53. Zhang, K., Lu, J., Lafruit, G., Lauwereins, R., Van Gool, L.: Real-time accurate stereo with bitwise fast voting on cuda. In: Proceedings of IEEE Workshop on Embedded Computer Vision (2009)

Author Biographies

Liang Wang is a researcher in Microsoft Corp.'s Applied Sciences Group, where he has been working on the invention and development of novel human–computer interfaces using computer vision technologies since 2010. His research interests include a variety of topics in computer vision, computer graphics, and image processing, especially on stereo matching, structure from motion, 3D reconstruction, image-based modeling and rendering, segmentation and matting for live video and multi-projector display system. Liang Wang received his Bachelor's degree in Computer Science and Engineering from Beihang University, Beijing, China in 2004. He then received his Doctor of Science degree in Computer Science from University of Kentucky in 2012. He is a member of the IEEE.

Ruigang Yang received the MS degree in computer science from Columbia University in 1998 and the PhD degree in computer science from the University of North Carolina, Chapel Hill, in 2003. He is an associate professor in the Computer Science in the Department at the University of Kentucky. His research interests include computer graphics, computer vision, and multimedia. He is a recipient of the US National Science Foundation CAREER award in 2004 and a member of the IEEE, the IEEE Computer Society and the ACM.

Minglun Gong is an associate professor at the Memorial University of Newfoundland and an adjunct professor at the University of Alberta. He obtained his Ph.D. from the University of Alberta in 2003, his M.Sc. from the Tsinghua University in 1997, and his B.E. from the Harbin Engineering University in 1994. After graduation, he was a faculty member at the Laurentian University for 4 years before joining the Memorial University. Minglun's research interests cover various topics in the broad area of visual computing (including computer graphics, computer vision, visualization, image processing and pattern recognition). So far, he has published over 70 referred technical papers in refereed journals and conference proceedings and submitted 2 patent applications. He has served as a program committee member for top-tier conferences, such as ICCV and CVPR, and reviewer for prestigious journals including IEEE, TPAMI and IJCV. He was the recipient of the Izaak Walton Killam Memorial Award and the CFI New Opportunity Award.

Miao Liao received his B.S. degree from Tsinghua University in 2005 and the PhD degree in Computer Science from the University of Kentucky, in 2011. He is currently a senior researcher in Sharp Labs of America in Camas, Washington. His research interests include computer vision, computer graphics, image processing and multimedia. He has served as a program committee member and reviewer for international journals and conferences.