EURASIP Journal on Image and Video Processing

RESEARCH Open Access

Efficient cost aggregation for feature-vector-based wide-baseline stereo matching

Xiaoming Peng[1,2*], Abdesselam Bouzerdoum[1,3] and Son Lam Phung[1]

Abstract

In stereo matching applications, local cost aggregation techniques are usually preferred over global methods due to their speed and ease of implementation. Local methods make implicit smoothness assumptions by aggregating costs within a finite window; however, cost aggregation is a time-consuming process. Furthermore, most existing local methods are based on pixel intensity values, and hence are not efficient with feature vectors used in wide-baseline stereo matching. In this paper, a new cost aggregation method is proposed, where a Per-Column Cost matrix is combined with a feature-vector-based weighting strategy to achieve both matching accuracy and computational efficiency. Here, the proposed cost aggregation method is applied with the DAISY feature descriptor for wide-baseline stereo matching; however, this method can also be applied to a fast growing number of stereo matching techniques that are based on feature descriptors. A performance comparison with several benchmark local cost aggregation approaches is presented, along with a thorough analysis of the time and storage complexity of the proposed method.

Keywords: Stereo matching, Cost aggregation, Feature vector, DAISY

1 Introduction

Estimating depth from a pair of stereo images is a long-standing problem in computer vision. Its aim is to find a dense correspondence map between a pair of stereo images to generate either a disparity map (for rectified stereo pairs), or a depth map (for known camera calibration parameters). Stereo algorithms generally involve the following four steps: i) matching cost computation, (ii) cost aggregation, (iii) disparity computation or optimization, and (iv) disparity refinement [1]. Both *local* and *global* approaches need to perform the matching cost computation step, but they differ in the treatment of smoothness constraints. Local methods make implicit smoothness assumptions by aggregating costs within a finite window. Global approaches, by contrast, make explicit smoothness assumptions by combining the data and smoothness terms into a cost function, which is subsequently optimized using an iterative procedure. The most commonly used optimization methods for global approaches include Expectation-Maximum (EM) [2], cooperative optimization [3, 4], Graph Cuts (GC) [5], Max-Product Loopy Belief Propagation (LBP) [6], and Tree-Reweighted Message Passing (TRW) [7]. The last three methods are categorized as energy minimization for Markov Random Fields (MRFs) [8]. In practical applications, local approaches are preferred to their global counterparts due to their speed and ease of implementation.

Existing short-baseline stereo matching methods, which are mostly based on pixel intensity values, perform reasonably well and are quite fast. Di Stefano et al. proposed a local method which achieved real-time speed using Single Instruction Multiple Data (SIMD) implementation [9]. Tombari et al. compared fourteen cost aggregation techniques in terms of accuracy and computation cost [10]. They found that cost aggregation methods using adaptive weights are among the most accurate. Hirschmüller proposed a semi-global method based on mutual information [11]. Based on the gestalt principles, Yoon and Kweon

*Correspondence: xp815@uowmail.edu.au
[1]School of Electrical, Computer and Telecommunications Engineering, University of Wollongong, Northfields Ave, Wollongong, NSW 2522, Australia
[2]School of Automation Engineering, University of Electronic Science and Technology of China, Qingshuihe Campus, 2006 Xiyuan Ave, West Hi-Tech Zone, 611731 Chengdu, Sichuan, China
Full list of author information is available at the end of the article

© The Author(s). 2018 **Open Access** This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

developed an edge-preserving bilateral filter for stereo matching [12]. Subsequently, Mattoccia et al. proposed a symmetric adaptive weighting strategy using two independent spatial and range filters [13]. Instead of adopting an exact weighting strategy, Min et al. addressed the cost aggregation issue by introducing two approximations [14]. In another approach, Hosni et al. formulated the stereo matching problem in a cost-volume filtering manner [15]. The cost volume is a three-dimensional (3D) array that stores the costs for choosing a label (i.e. disparity value in stereo matching) at a given pixel. To maintain boundaries in the filtered output of the guidance image (the left image of a stereo pair), the filter weights are chosen to be those of the guided filter [16]. In [17], Yang developed a Minimum-Spanning-Tree-based cost aggregation method, which avoids the local optimality caused by manually specifying the size of the support window. Inspired by this work, Mei et al. introduced a segment-tree-based cost aggregation method [18]. More recently, Zhang et al. showed that the different cost aggregation methods essentially differ in the choice of similarity kernels and can be reformulated in a unified optimization framework [19].

Matching measures directly constructed using pixel intensity values, such as Sum of Absolute Differences (SAD), Sum of Squared Differences (SSD), and Normalized Cross Correlation (NCC), lack robustness to large perspective distortions. These measures are not suitable for wide-baseline stereo matching, where there are significant variations in the viewpoints. A better alternative to pixel-intensity-based cost aggregation is to employ local feature descriptors. In recent years, many new feature detectors and descriptors have been developed, including BRIEF [20], BRISK [21], FREAK [22] and ORB [23]. At the same time, several studies analyzed the performance of feature vectors on different tasks. Heinly et al. compared the performance of five descriptors, BRIEF, BRISK and ORB, SIFT [24], and SURF [25] in feature detection and description [26]. Khan et al. used precision-recall curves and the Wilcoxon signed rank test to compare the performance of thirteen feature vectors [27]. They found the SIFT is the most accurate performer in both image recognition and feature matching applications. The increasingly abundant feature descriptors provide alternatives to pixel-intensity-based similarity measures in dense matching scenarios, as shown by Tola et al. [28] and Liu et al. [29]. Tola et al. showed that the DAISY feature descriptor, SIFT and SURF all outperform the NCC and pixel difference in wide-baseline stereo matching [28]. To accelerate the cost aggregation step using the BRIEF feature descriptor, Zhang et al. incorporated binary masks into the cost aggregation term [30]. The binary masks are constructed using the Sum of Absolute Differences between two pixels in the CIELAB color space. However, their binary masks can only be paired with binary feature vectors like BRIEF. Thus, this method is not applicable to general real-valued feature vectors such as SIFT and DAISY.

Stereo matching using feature vectors has two difficulties. *First, feature-vector-based matching costs are much more computationally intensive than pixel-intensity-based ones.* There are several pixel-intensity-based strategies for reducing cost aggregation [13–16]. However, they do not work well when directly applied to feature vectors. For example, the computational load of the similarity kernels in the bilateral filtering methods [12, 13] and the tree-based methods [17, 18] is quite low when involving only pixel-wise intensity differences. However, their computational load is very high if feature-vector-based similarity measures are used. *Second, storing feature vectors for each image pixel requires a large amount of memory.* To facilitate the repetitive testing of different pixel correspondences, it is desirable to store the per-pixel feature vectors, as done in SIFT flow [29]. As an example of storage cost, to store a 200-element DAISY feature vector in double-precision floating-point format for each pixel of a pair of high definition images of size 1920×1080 pixels, we need approximately $(2 \times 1920 \times 1080 \times 8 \times 200)/1024^3 \approx 6.18$ GB of storage. Obviously, this is not practical on memory-limited systems.

Very recently, deep learning has been used to compare image pairs for stereo matching [31–34]. In this approach, a convolutional neural network (CNN) is deployed to compute the matching cost between a pair of image patches from the left and right images. Zagoruyko and Komodakis extracted feature descriptors from image patches at the branches of their Siamese network [31]. Their feature descriptor can be used as an alternative to hand-crafted feature descriptors, such as SIFT and DAISY. Žbontar and LeCun developped a convolutional neural network that directly outputs the matching cost between a pair of image patches [32]. The matching cost is then combined with a cross-based cost aggregation method. Chen et al. proposed a multi-scale deep embedding model to extract features from a pair of image patches [33]. The inner product of the features is large if the pair of image patches matches well, and vice versa. Luo et al. adopted a four-layer Siamese architecture for their CNN [34]. However, they observed that simply predicting the most likely configuration for every pixel using only the CNN output is not competitive with other modern stereo algorithms. To achieve a better performance, they combined their CNN with semi-global block matching and sophisticated post-processing. All these deep learning methods rely on the availability of a large pool of annotated pairs of image patches to learn a mapping between them. To address this issue, Mayer et al. established a synthetic dataset containing 35,000 stereo image pairs with ground truth disparity, optical flow, and scene flow [35].

In this paper, we propose a local cost aggregation method that operates on feature vectors. The proposed method has two major contributions. *First, we develop a feature-vector-based weighting strategy,* which can be computed much more efficiently than conventional bilateral filtering, but with only a slight reduction in accuracy. *Second, we propose a new concept called the Per-Column Cost (PCC) matrix to share the aggregated costs across different disparities during cost aggregation.* This is in sharp contrast to other cost aggregation strategies, such as Integral Images [36] and Box-Filtering [9], which are limited to a single disparity map. Although we use the DAISY feature to instantiate the proposed method, other feature vectors can also be applied.

The rest of the paper is organized as follows. In Section 2, the cost aggregation problem is formulated under the filtering framework, followed by the delineation of the proposed method. In Section 3, experimental results are presented, followed by a comprehensive analysis and discussion of the results. In Section 4, conclusions and future directions are given.

2 Cost aggregation method using feature vectors

In this section, we first cast cost aggregation as a filtering problem. Then, we analyze why existing pixel-intensity-based cost aggregation methods are not suitable for feature vectors. Finally, we describe the proposed method in detail.

2.1 Cost aggregation using a filtering framework

The proposed method works on a pair of *rectified* stereo images, where the search of corresponding points on the stereo image pair is constrained on the horizontal image scanlines. Consider a pair of *left* image I_l and *right* image I_r. The aim is to find a pixel $q = (x+d, y)$ in I_r which corresponds to a pixel $p = (x, y)$ in I_l, where d is the disparity between the pixel pair, $d \in \mathcal{D} = [d_{\min}, d_{\max}]$. Let $I_l(x, y)$ denote the intensity value (or the vector of color values) at location (x, y) in image I_l. The search for pixel q is carried out along a horizontal scan line. For a chosen feature vector $\mathbf{f}$ of length L, the dissimilarity between pixels p and q is computed by comparing $\mathbf{f}(I_l; p)$ and $\mathbf{f}(I_r; q)$, which denote the feature vectors extracted at pixels p and q in I_l and I_r, respectively. Here, we do not restrict the type of feature vector $\mathbf{f}$, so long as we can derive a scalar dissimilarity measure between $\mathbf{f}(I_l; p)$ and $\mathbf{f}(I_r; q)$. For simplicity and without loss of generality, we assume that images I_l and I_r have identical sizes (H rows, W columns). Furthermore, the search range $\mathcal{D}$ has M discrete integer values, and is the same for all the pixels p in I_l. The dissimilarity between two feature vectors $\mathbf{f}_1$ and $\mathbf{f}_2$ is denoted as $c\,(\mathbf{f}_1, \mathbf{f}_2)$.

Cost aggregation can be defined as a filtering process [19]. Let C denote the cost volume of size $W \times H \times M$, where $C(x, y, d)$ stores the cost for pixel (x, y) at disparity level d. Let $\nabla_x I$ denote the gradient component of image I along the x direction. For a pixel p in image I_l, the combined intensity and gradient cost volume, used in [15, 17, 18], is defined as

$$C(x,y,d) = (1-\alpha)\min\left(\|I_l(x,y) - I_r(x+d,y)\|_2, \tau_1\right) + \alpha \min\left(\|\nabla_x I_l(x,y) - \nabla_x I_r(x+d,y)\|_2, \tau_2\right), \tag{1}$$

where $1 \le x \le W$, $1 \le y \le H$, and $d_{\min} \le d \le d_{\max}$. The parameters τ_1 and τ_2 are two thresholds, and α balances the color and gradient terms.

Consider a 2D filter K (also called the "similarity kernel" in [19]). For a support window of size $(2w+1) \times (2w+1)$ and centered at pixel (x, y) in the left image I_l, the filter output can be expressed as

$$\tilde{C}(x,y,d) = \sum_{i=-w}^{w}\sum_{j=-w}^{w} K(i,j)\; C(x+i, y+j, d). \tag{2}$$

Existing cost aggregation methods differ mainly in the choice of K. For example, for the edge-preserving bilateral filter [12], the kernel is given as

$$K^{\mathrm{bf}}(i,j) = \frac{1}{Z}\exp\left(-\frac{\sqrt{i^2+j^2}}{\sigma_s}\right) \exp\left(-\frac{\|I_l(x,y) - I_l(x+i,y+j)\|_2}{\sigma_c}\right), \tag{3}$$

where $-w \le i, j \le w$, the parameters σ_s and σ_c control the spatial and color similarity, and Z is a normalization constant such that $\sum_{i=-w}^{w}\sum_{j=-w}^{w} K^{\mathrm{bf}}(i,j) = 1$. In the case of the guided-image filter [16], the kernel is given as

$$K^{\mathrm{gf}}(i,j) = \frac{1}{(2w+1)^2}\left[1 + \left(I_l(x,y) - \mathbf{u}\right)^T (\Sigma + \epsilon\mathbf{E}) \left(I_l(x+i,y+j) - \mathbf{u}\right)\right], \tag{4}$$

where $\mathbf{u}$ is the mean vector, Σ is the covariance matrix of the pixels in the support window, $\mathbf{E}$ denotes the identity matrix, and ϵ is a smoothness parameter. Note that for both filters, the filter kernel varies according to the coordinates of the center pixel (x, y). In the tree-based cost aggregation methods [17, 18], a connected, undirected graph $G = (V, E)$ is established for I_l, where the set of vertices V is the image pixels and the edges E connect neighboring pixels. In all these kernels, the weight between two pixels is defined as the difference between their intensity values.

One important reason that the state-of-the-art cost aggregation methods can work very fast is that the similarity kernels can be computed very efficiently, with simple arithmetic operations mostly. However, this is not the case when feature vectors rather than pixel intensity values are used. For example, if the pixel difference term $\|I_l(x,y) -$

$I_l(x+i, y+j)\|_2$ in Eq. (3) is replaced with the feature-vector-based dissimilarity term $c(\mathbf{f}(I_l; x, y), \mathbf{f}(I_l; x+i, y+j))$, we would incur a sharp increase in the computation.

2.2 Feature-vector-based cost aggregation

To formulate the feature-vector-based cost aggregation problem as a filtering problem, we first need to compute the cost volume C:

$$C(x, y, d) = c(\mathbf{f}(I_l; x, y), \mathbf{f}(I_r; x+d, y)), \tag{5}$$

where $x \in [1, W]$, $y \in [1, H]$, and $d \in [d_{\min}, d_{\max}]$. The dissimilarity measure $c(\mathbf{f}(I_l; x, y), \mathbf{f}(I_r; x+d, y))$ is problem-dependent. For example, a commonly used dissimilarity measure is the Euclidean norm of the difference between the two feature vectors. To avoid repeatedly computing and comparing feature vectors, the cost $C(x, y, d)$ is computed in a row-wise manner. For each row y, we form two $L \times W$ matrices:

$$F_l^y = \left[\mathbf{f}(I_l; 1, y), \mathbf{f}(I_l; 2, y), \ldots, \mathbf{f}(I_l; x, y), \ldots, \mathbf{f}(I_l; W, y)\right], \tag{6}$$
$$F_r^y = \left[\mathbf{f}(I_r; 1, y), \mathbf{f}(I_r; 2, y), \ldots, \mathbf{f}(I_r; x, y), \ldots, \mathbf{f}(I_r; W, y)\right].$$

The x-th column of F_l^y contains the feature vector computed at pixel (x, y) in the left image I_l. The x-th column of F_r^y contains the feature vector computed at pixel (x, y) in the right image I_r. To compute $c(\mathbf{f}(I_l; x, y), \mathbf{f}(I_r; x+d, y))$, the feature vectors $\mathbf{f}(I_l; x, y)$ and $\mathbf{f}(I_r; x+d, y)$ are retrieved directly from F_l^y and F_r^y rather than being re-computed. Next, we present in detail the proposed method for cost aggregation.

2.2.1 *Feature-vector-based weighting strategy*

Our feature-vector-based weighting strategy is inspired by the edge-preserving bilateral filter [12], but is modified to accommodate feature-vector-based dissimilarity measures. Figure 1 shows the difference between the conventional bilateral filter and the proposed bilateral filter. In the conventional bilateral filter, the weights $K^{\mathrm{bf}}(i, j)$ of Eq. (3) are computed with a single center pixel as the reference. However, in the proposed bilateral filter, the weights are computed with multiple center pixels as the reference.

Consider a support window of size $(2w+1) \times (2w+1)$ located at pixel (x, y) in an image. The weights of the proposed bilateral filter are defined as

$$G^{\mathrm{bf}}(i, j) = \exp\left(-\frac{|i|}{\sigma_1}\right) \exp\left(-\frac{c(\mathbf{f}(I_l; x, y+j), \mathbf{f}(I_l; x+i, y+j))}{\sigma_2}\right), \tag{7}$$

where $-w \le i, j \le w$, and parameters σ_1 and σ_2 control the spatial and color similarity. From Eq. (7), if $i = 0$, $G^{\mathrm{bf}}(i, j) = 1$. In other words, all the pixels in the middle column of the support window share the same weights, and act as "center" or "reference" pixels. As in the case of $K^{\mathrm{bf}}(i, j)$, the weights of $G^{\mathrm{bf}}(i, j)$ are normalized so that $\sum_{i,j} G^{\mathrm{bf}}(i, j) = 1$.

The rationale of the proposed bilateral filter can be explained as follows. Because the feature vector $c(\mathbf{f}(I_l; x, y)$ describes a local area around a pixel (x, y) instead of just the pixel itself, the cost $c(\mathbf{f}(I_l; x, y), \mathbf{f}(I_l; x+i, y+j))$ is less spatially sensitive than $\|I_l(x, y) - I_l(x+i, y+j)\|_2$. This property enables us to use multiple center pixels for the proposed bilateral filter instead of a single center pixel as

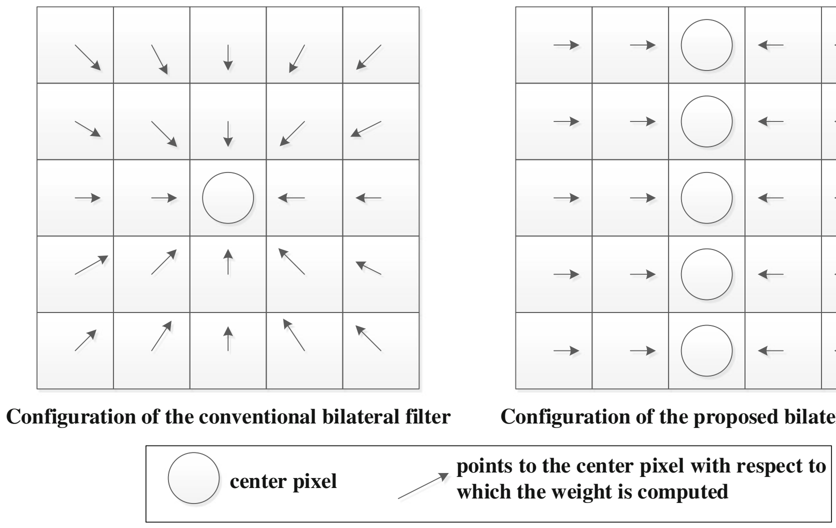
Configuration of the conventional bilateral filter
Configuration of the proposed bilateral filter
center pixel
points to the center pixel with respect to which the weight is computed

Fig. 1 Difference between the conventional bilateral filter and the proposed bilateral filter. Left: weights for the conventional bilateral filter are computed with one single center pixel as the reference. Right: weights of the proposed bilateral filter are computed with multiple center pixels as the reference

for the conventional bilateral filter. Furthermore, Eq. (7) is computed along the horizontal scan lines. If a single center is used, the term $c(\mathbf{f}(I_l; x, y), \mathbf{f}(I_l; x+i, y+j))$ needs to be computed *across* the scan lines, leading to a significant increase in the storage requirement. As confirmed later in Section 3.4, the proposed multi-center bilateral filter operates more efficiently while achieving a similar stereo matching accuracy compared with the single-center case.

2.2.2 Per-column cost matrix

The proposed weighting strategy works efficiently when combined with a new concept called the *Per-Column Cost* matrix. To illustrate this concept, Fig. 2 shows an example of three support windows of size 3×3 pixels (i.e., $w = 1$) centered at locations (x, y), $(x+1, y)$ and $(x+2, y)$ in the left image. The three support windows share a common column (colored with red). For any of these three support windows, the counterparts of this red column in the d-shifted support windows in the right image occupy the same consecutive region of $(2w+1) \times M$ pixels (shaded with gray). This fact implies that if we record the accumulated costs associated with the red column when matching either of the three support windows in the right image, we can reuse the accumulated costs and reduce the computational load by a factor of $(2w+1)$. This observation leads us to construct a matrix Γ_y to store the column-based accumulated costs:

$$\begin{aligned}\Gamma_y(x+d, x) &= \sum_{j=-w}^{w} c(\mathbf{f}(I_l; x, y+j), \mathbf{f}(I_r; x+d, y+j)) \qquad (8)\\ &= \sum_{j=-w}^{w} C(x, y+j, d),\end{aligned}$$

where the subscript y indicates that it is constructed for the y-th row.

The matrix Γ_y, of size $W \times W$, can be computed quite efficiently. First, it is quite sparse: for the x-th row of Γ_y, all entries except those with indices from $(x + d_{\min})$ to $(x + d_{\max})$ are empty. Second, starting with Γ_{w+1}, the matrix Γ_y for $y > w+1$ can be computed recursively as follows:

$$\Gamma_y(i,j) = \Gamma_{y-1}(i,j) + C(j, y+w, i-j) - C(j, y-1-w, i-j). \qquad (9)$$

An important property of Γ_y is that it is shared across different disparities, as opposed to Integral Images and Box-Filtering, which are limited to a single disparity level.

Next, we incorporate the weighting strategy described in Section 2.2.1. From Eq. (8), each element of Γ_y accumulates the unweighted costs from one column of the support window. However, the weights in Eq. (7) are computed pixel-wise. To address this incompatibility, we compute the average of the weights within one column of the support window. This averaged weight is used as a common weight for the accumulated costs within one column of the support window.

Now, we present an explanation for the above approximation. In Eq. (2), the weights $K(i,j)$ are normalized so that they sum to one for a given support window. To simplify the discussion, we rewrite Eq. (2) as

$$\tilde{C} = \sum_{s=1}^{2w+1} \sum_{t=1}^{2w+1} K_{s,t}\, C_{s,t} = \sum_{n=1}^{(2w+1)^2} K_n\, C_n, \qquad (10)$$

where s is the column index, t is the row index, and n is the linear index of the pair (s, t). Here, C_n is the cost, and K_n is the positive weight associated with the n-th element in the support window, $\sum_n K_n = 1$. Now we divide the elements in the support window into $(2w+1)$ columns, the column-wise common weights (described above) amount to approximating Eq. (10) by

$$\tilde{C} \approx \sum_{s=1}^{2w+1} \sum_{t=1}^{2w+1} \bar{K}_s\, C_{s,t} = \sum_{s=1}^{2w+1} \bar{K}_s \sum_{t=1}^{2w+1} C_{s,t}, \qquad (11)$$

where the common weight for column s is denoted as $\bar{K}_s = \frac{1}{2w+1} \sum_{t=1}^{2w+1} K_{s,t}$. Note that $\sum_{s=1}^{2w+1} (2w+1) \bar{K}_s = \sum_{s=1}^{2w+1} \sum_{t=1}^{2w+1} K_{s,t} = 1$.

Now that $\sum_{t=1}^{2w+1} K_{s,t}$ represents the accumulated weights within one column of the support window, we can compute it in a similar way as for the PCC matrix Γ. To this end, we initialize a *Per-Column-Weight* matrix Φ of size $W \times W$. To facilitate the computation of Φ, we use a 3D Per-Pixel-Weight array P of size $W \times H \times (2w+1)$.

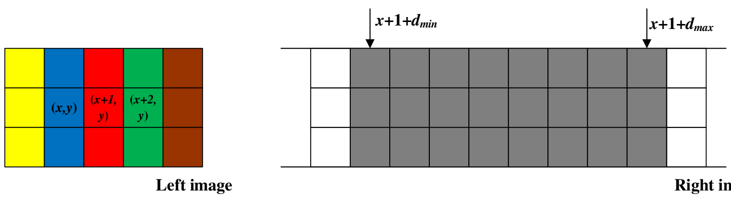

Fig. 2 An example showing the 3×3 support windows ($w = 1$) of three consecutive pixels (x, y), $(x+1, y)$ and $(x+2, y)$ in the left image. These three support windows share a common column (colored with red). For a given support window in the left image, a series of d-shifted support windows in the right image are matched against. The counterparts of this red column in these d-shifted support windows occupy an identical region of pixels in the right image (shaded with gray)

The pseudo-code for the proposed feature-vector-based cost aggregation algorithm is given in Algorithm 1. In the pseudo-code, we use F_l^y, F_r^y, Φ_y and Γ_y to explicitly emphasize that the corresponding computations are according to the y-th scanline. For simplicity, we do not consider the "border problem". However, this restriction can be avoided by replicating the border pixels. The major steps of Algorithm 1 are also summarised in Fig. 3.

An estimated disparity d is considered reliable if its aggregated cost $\mathbf{a}(d)$ is smaller than $(2w+1)^2\ \tau$, where τ is a predefined threshold. Otherwise, we regard the pixel as occluded. A large aggregated cost indicates the feature

Algorithm 1 The proposed feature-vector-based cost aggregation algorithm

```
Inputs and parameters:
  - rectified stereo image pair $\{I_l, I_r\}$ of $W \times H$ pixels in size
  - disparity search range $\mathcal{D} = [d_{\min}, d_{\max}]$ of length $M$
  - feature vector length $L$
  - support window size $(2w+1) \times (2w+1)$

Outputs:
  - disparity map $\Omega$ of size $W \times H$

1:  ▷ Initialization of the following data to zero
2:    - 3D cost volume array $C$ of size $W \times H \times M$
3:    - 3D Per-Pixel-Weight array $P$ of size $W \times H \times (2w+1)$
4:    - Per-Column-Cost matrix $\Gamma$ of size $W \times W$
5:    - Per-Column-Weight matrix $\Phi$ of size $W \times W$
6:    - two matrices $F_l$ and $F_r$ of size $L \times W$
7:    - weighted aggregated costs vector $\mathbf{a}$ of length $M$
8:    - weights vector $\mathbf{w}$ of length $2w+1$

9:  ▷ Computation of $C$ and $P$
10: for each row $y$ do
11:     Compute $F_l^y$ and $F_r^y$ using Eq. (6).
12:     Compute $C(x, y, d)$ for each $x$ and $d \in \mathcal{D}$ using Eq. (5).
13:     Compute $P(x, y, z) = \exp\left(-\frac{|z|}{\sigma_1}\right) \exp\left[-\frac{1}{\sigma_2} c(\mathbf{f}(I_l; x, y), \mathbf{f}(I_l; x+z, y))\right]$ for each $x$ and
        $z \in [-w, w]$.
14: end for

15: ▷ Computation of the weighted aggregated costs
16: for each row $y \in [w+1, H-w]$ do
17:     if $y = w+1$ then
18:         Compute $\Phi_y(x+z, x) = \Phi_y(x, x+z) = \sum_{t=-w}^{w} P(x, y+t, z)$ for each $x$ and $z \in [-w, w]$.
19:         Construct $\Gamma_y$ using Eq. (8).
20:     else
21:         Update $\Phi_y(i, j) = \Phi_{y-1}(i, j) + P(j, y+w, i-j) - P(j, y-1-w, i-j)$.
22:         Update $\Gamma_y$ using Eq. (9).
23:     end if
24:     for each $x$ do
25:         Compute $\mathbf{w}(z) = \Phi_y(x+z, x)$ for each $z \in [-w, w]$ .
26:         Normalize $\mathbf{w}(\mathbf{z})$ to sum to 1.
27:
28:         Compute $\mathbf{a}(d) = \sum_{z=-w}^{w} \frac{\mathbf{w}(z)}{2w+1} \Gamma_y(x+z+d, x+z)$ for each $d \in \mathcal{D}$.
29:         Set $\Omega(x, y) = \arg\min_d \mathbf{a}(d)$.
30:     end for
31: end for
```

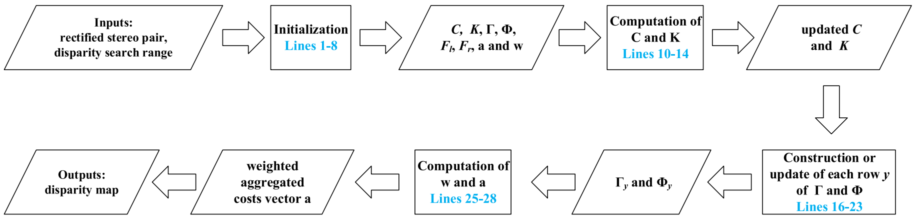
Inputs: rectified stereo pair, disparity search range
Initialization Lines 1-8
C, K, Γ, Φ, F_l, F_r, a and w
Computation of C and K Lines 10-14
updated C and K
Construction or update of each row y of Γ and Φ Lines 16-23
Γ_y and Φ_y
Computation of w and a Lines 25-28
weighted aggregated costs vector a
Outputs: disparity map

Fig. 3 Block diagram of the proposed method

vectors are not well matched, usually because the local regions lack distinctive visual appearance.

2.2.3 Computation complexity

Before discussing the algorithm complexity, we first distinguish between three types of operations involved in our method: (a) feature vector computation, (b) feature vector comparison, and (c) basic floating-point arithmetic operation. Operation (a) is the computation of a feature vector at a given pixel. Operation (b) is the computation of the dissimilarity between two feature vectors. Operation (c) obviously is much less computationally-intensive than Operations (a) and (b). For this reason, we use $O_a(1)$, $O_b(1)$ and $O_c(1)$ to denote the time complexity for each of these three operations, respectively.

The computation of the feature vectors for all pixels in the left and right image has a time complexity of $O_a(2 \times W \times H)$. To construct the cost volume array C, the term $c(\mathbf{f}(I_l; x, y), \mathbf{f}(I_r; x + d, y))$ needs to be computed for each location (x, y) in the left image and each disparity candidate d in $\mathcal{D}$, which leads to a time complexity of $O_b(W \times H \times M)$. Similarly, the construction of array P needs a time complexity of $O_b(W \times H \times [\,2w + 1\,])$.

The initial construction of matrix Γ has a time complexity of $O_c\left(W \times [2w + 1]^2 \times M\right)$. Note that $W \times M$ is the number of valid entries in Γ. Then, updating each valid entry requires only three single floating-point arithmetic operations, see Eq. (8). Thus, we have a time complexity of $O_c(3 \times W \times (H - 2w - 1) \times M) \approx O_c(3 \times W \times H \times M)$ for this purpose. Similarly, the construction and update of matrix Φ require a time complexity of $O_c\left(\frac{(2w+1)^2}{2} \times W\right)$ and $O_c(3/2 \times W \times H \times (2w + 1))$, respectively. Note that the $1/2$ factor is due to the symmetry of Φ.

Once Φ and Γ are computed, for each pixel (x, y) in the left image, the evaluation of $\mathbf{a}$ needs $3 \times (2w + 1) \times M$ floating-point arithmetic operations. The evaluation of $\mathbf{w}$ requires an extra $(2w+2)$ floating-point arithmetic operations, which is negligible compared with that of evaluating $\mathbf{a}$. Therefore, for all the pixels in the left image, the time complexity with respect to this step is $O_c(3 \times (2w + 1) \times W \times H \times M)$.

Now we analyze the storage required for implementing the algorithm. The cost volume array C and the array P require storage of $W \times H \times M$ and $W \times H \times (2w + 1)$ floating-point numbers, respectively. They account for the largest share of storage requirement. However, if storage is limited, this requirement can be significantly reduced. In fact, only the top $(2w + 1)$ rows participate in the initial construction of Γ, and we only need a small portion of C corresponding to these $(2w + 1)$ rows, which consists of $(2w+1) \times W \times M$ floating-point numbers. Afterwards, Γ is iteratively updated by "removing" the oldest contributing "row" of C and adding a new one, which means we only need to keep two "rows" of C, or $2 \times W \times M$ floating-point numbers. Summarizing these two cases, it would be sufficient to use $(2w+1) \times W \times M$ floating-point numbers to dynamically keep those "rows" of C that are needed for the construction or update of Γ. Similarly, $(2w + 1)^2 \times W$ floating-point numbers are required for the construction or update of P.

The matrices F_l and F_r both have $L \times W$ elements, and the storage requirement is dependent on the type of the chosen feature vector. For feature vectors such as DAISY and SIFT, each element is one floating-point number. For feature vectors such as BRIEF and BRISK, each element is one bit. The matrices Γ and Φ each require W^2 floating-point numbers for storage, and can be re-used for each row. The storage requirement can be further reduced if their sparsity can be exploited. Finally, the vectors $\mathbf{a}$ and $\mathbf{w}$ require M and $(2w + 1)$ floating-point numbers for storage, respectively. The time and storage complexity for the proposed method are summarized in Table 1.

3 Results and discussion

In this section, we first introduce the test data in Section 3.1 and the DAISY descriptor in Section 3.2. Then, we explain how to select the parameters of the proposed method in Section 3.3. Next, we compare the proposed feature-vector-based weighting strategy with several other weighting strategies in Section 3.4. Finally, we compare the performance of the proposed cost aggregation method with two benchmark methods on two datasets in Section 3.5.

The proposed algorithm was implemented using C++. The experiments were done on a desktop computer equipped with an Intel Core i7-4770@3.40 GHz CPU, 8

Table 1 Time and storage complexity of the proposed feature-vector-based cost aggregation algorithm

Time complexity	
Feature vector computation	$O_a(2 \times W \times H)$
Feature vector comparison	$O_b(W \times H \times M)$ for C and $O_b(W \times H \times (2w+1))$ for P
Construction and update of Γ	$O_c\left((2w+1)^2 \times W \times M\right)$ and $O_c(3 \times W \times H \times M)$, respectively
Construction and update of Φ	$O_c\left(\frac{(2w+1)^2}{2} \times W\right)$ and $O_c(3/2 \times W \times H \times (2w+1))$, respectively
Cost aggregation for all pixels	$O_c(3 \times (2w+1) \times W \times H \times M)$
Storage complexity	Unit: Number of floating-point numbers
The cost volume array C	Maximum: $W \times H \times M$
	Minimum: $(2w+1) \times W \times M$
Array P	Maximum: $(2w+1) \times W \times M$
	Minimum: $(2w+1)^2 \times W$
Matrices F_l and F_r	$L \times W$ entries, depending on the type of feature vector
Matrices Γ and Φ	W^2
Vector $\mathbf{a}$	M
Vector $\mathbf{w}$	$2w+1$

GB memory, and 64-bit Windows 7 Enterprise operating system.

3.1 Wide-baseline stereo image data

The experiments in this work used two groups of wide baseline stereo image data: i) the Fountain and HerzJesu dataset; and ii) the 2014 Middlebury Stereo dataset.

3.1.1 The fountain and HerzJesu dataset

The first group of test data is from the public dataset released by Strecha et al. [37]. Specifically, we used two data sets: the "Fountain" data set and the "HerzJesu" data set. Both data sets contain gray-scale images of size 768 × 512 pixels, along with their ground-truth depth maps and occlusion maps. The camera calibration parameters associated with each gray-scale image are available, allowing these images to be rectified. The rectified images are of size 768 × 512 pixels.

For the "Fountain" data set, eleven wide-baseline stereo images were used in our experiments. One stereo image was used for the parameter selection experiment, presented in Section 3.3. The other ten stereo images were divided into two sub-sets, denoted as Fountain-A and Fountain-B. Each sub-set contained five consecutive images. For each sub-set, one image was considered as the left image while the other four images were considered as the right images. This was repeated five times, giving twenty stereo pairs for each sub-set. For the "HerzJesu" dataset, five images were selected to form another sub-set. In all, the first group of test data contained 60 stereo pairs. For brevity, we call this set "Fountain and HerzJesu Dataset" hereafter.

For this group of test data, the performance of a given stereo matching method was measured using precision, recall and running time. Let m be the number of *correctly* estimated non-occluded pixels, n the number of non-occluded pixels, and N the number of ground truth non-occluded pixels. For a given non-occluded pixel, if the relative error between its estimated depth value and the ground-truth depth value is less than 5%, the estimation is considered to be correct. The precision and recall are defined as

$$\begin{aligned} precision &= \frac{m}{n}, \\ recall &= \frac{n}{N}. \end{aligned} \tag{12}$$

3.1.2 The 2014 Middlebury stereo dataset

The second group of test data is from the 2014 Middlebury Stereo Dataset [38]. The stereo image pairs in this dataset were generated using a structured lighting system, and were meant to present new challenges for the next generation of stereo algorithms. Of the 33 stereo image pairs in the 2014 Middlebury Stereo Dataset, only 23 of them have accompanying ground-truth disparity maps. Therefore, we used these 23 stereo pairs in quarter resolution to form the second group of test data: 1) adirondack, 2) jadeplant, 3) motorcycle, 4) piano, 5) pipes, 6) playroom, 7) playtable, 8) recycle, 9) shelves, 10) vintage, 11) backpack, 12) bicycle1, 13) cable, 14) classroom1, 15) couch, 16) flowers, 17) mask, 18) shopvac, 19) sticks, 20) storage, 21) sword1, 22) sword2, and 23) umbrella.

Because this group has no accompanying occlusion maps, the performance of a given method is measured by the *overall disparity estimation accuracy*, which is defined as the fraction of pixels with correctly estimated disparity values. If the difference between the estimated disparity and ground-truth disparity of a pixel is less than two pixels, the disparity is considered as correctly estimated.

3.2 Implementation of the DAISY feature vector

In this work, we selected the DAISY feature descriptor to implement and test the proposed method. The DAISY descriptor gets its name from its flower-like shape. The center of the flower is located at the center of an image patch. There are Q concentric rings surrounding the flower center, each ring containing T evenly distributed circles. These $Q \times T$ circles form the flower petals. The interested reader is referred to Fig. 6 of [28] for a visual appearance of the DAISY descriptor. The flower center and petals are each described by a histogram of length H, which is the convolved orientation map computed at the flower center or a petal. Thus, a DAISY descriptor contains $H \times (Q \times T + 1)$ elements. Our experiments used the default parameters published in [28]: $Q = 3$, $T = 8$, and $H = 8$, for a feature vector of 200 elements.

The DAISY feature descriptor has been shown to outperform SIFT and SURF in wide-baseline stereo matching [28]. In addition, DAISY is more computationally efficient than SIFT because it reuses the descriptor computation of other pixels. A disadvantage of the DAISY descriptor is that it is not scale- and rotation-invariant. However, since we use rectified images as input, the scale and rotation disparities between the stereo image pair are mostly compensated during the image rectification step. A C++ implementation of the DAISY descriptor is publicly available from http://cvlab.epfl.ch/software/daisy.

For two DAISY descriptors $\mathbf{f}_1$ and $\mathbf{f}_2$, their dissimilarity is defined as

$$c(\mathbf{f}_1, \mathbf{f}_2) = \frac{1}{S} \sum_{k=1}^{S} \|\mathbf{f}_1^k - \mathbf{f}_2^k\|_2, \tag{13}$$

where S is the total number of non-occluded histograms, and $\mathbf{f}_1^k$ and $\mathbf{f}_1^k$ are the k-th normalized histogram of $\mathbf{f}_1$ and $\mathbf{f}_2$, respectively [28]. Of the $(Q \times T + 1) = 25$ histograms of a DAISY descriptor, some may be occluded because their corresponding petals lie outside the image plane. Hence, only the non-occluded histograms are used for matching. Each of the non-occluded histograms is normalized to unity norm. The dissimilarity measure $c(\mathbf{f}_1, \mathbf{f}_2)$ ranges between 0 (perfect match) and 2 (complete non-match).

3.3 Parameter selection for the proposed method

The proposed method has two important parameters, the support window size (determined by w) and the threshold value τ. The algorithm was run with various combinations of w and τ on two stereo pairs shown in Fig. 4. These two stereo pairs were not included in the Fountain and HerzJesu Dataset described in Section 3.1. The τ values were varied between 0.1 and 0.9 with a step of 0.1. The running times for different sizes of the support window were averaged. The results in terms of precision versus recall and average computation time are shown in Fig. 5. Each τ value corresponds to a cross on the precision-recall curves. Figure 5 shows that for every support window size, the recall increases and the precision decreases for increasing τ. Another observation is that a large support window size does not necessarily increase the performance in terms of precision-recall, despite an increase in running time. As discussed in Section 2.2.1, a feature vector describes the statistics in an image patch around a pixel. For all the pixels inside a support window, the union of their corresponding image patches exceeds the support window itself. For this reason, we can use a small support window, instead of a large one as is often required by pixel-intensity-based stereo matching. Based on this analysis, a combination of $w = 3$ and $\tau = 0.3$ was used in the subsequent experiments.

3.4 Analysis of weighting strategies

In this simple experiment, we analyzed the proposed feature-vector-based weighting strategy and three other weighting strategies. The experiment used the two stereo pairs presented in Section 3.3.

We first compared the proposed method with two weighting strategies: i) the conventional bilateral filter with intensity difference [12]; ii) the bilateral filter with a single center. The first weighting strategy is represented by Eq. (3). The second weighting strategy is an

Fig. 4 Two stereo pairs used for parameter selection for the proposed method

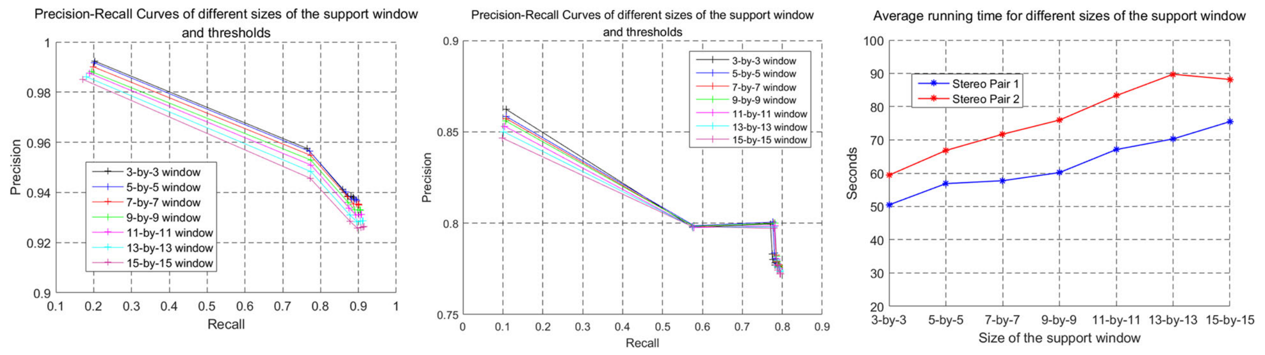

Fig. 5 Performance of the proposed method with various combinations of w and τ on the two stereo pairs. Left: the precision-recall curves of different w and τ values for the first stereo pair. Middle: the precision-recall curves of different w and τ values for the second stereo pair. Right: average running time for different values of w

extension of Eq. (3), by replacing the term $\|I_l(x,y) - I_l(x+i,y+j)\|_2$ with the feature-vector-based dissimilarity term $c(\mathbf{f}(I_l;x,y),\mathbf{f}(I_l;x+i,y+j))$.

For fair comparison between the proposed method and the first two strategies, the DAISY feature vector was used to compute the initial cost volume in Eq. (1), instead of the truncated absolute differences. This way, the performance differences in cost aggregation were solely determined by the different weighting strategies.

Figures 6 and 7 show the precision versus recall of the three weighting strategies on the two stereo pairs, for different support window sizes. The proposed weighting strategy significantly outperforms the bilateral filter with

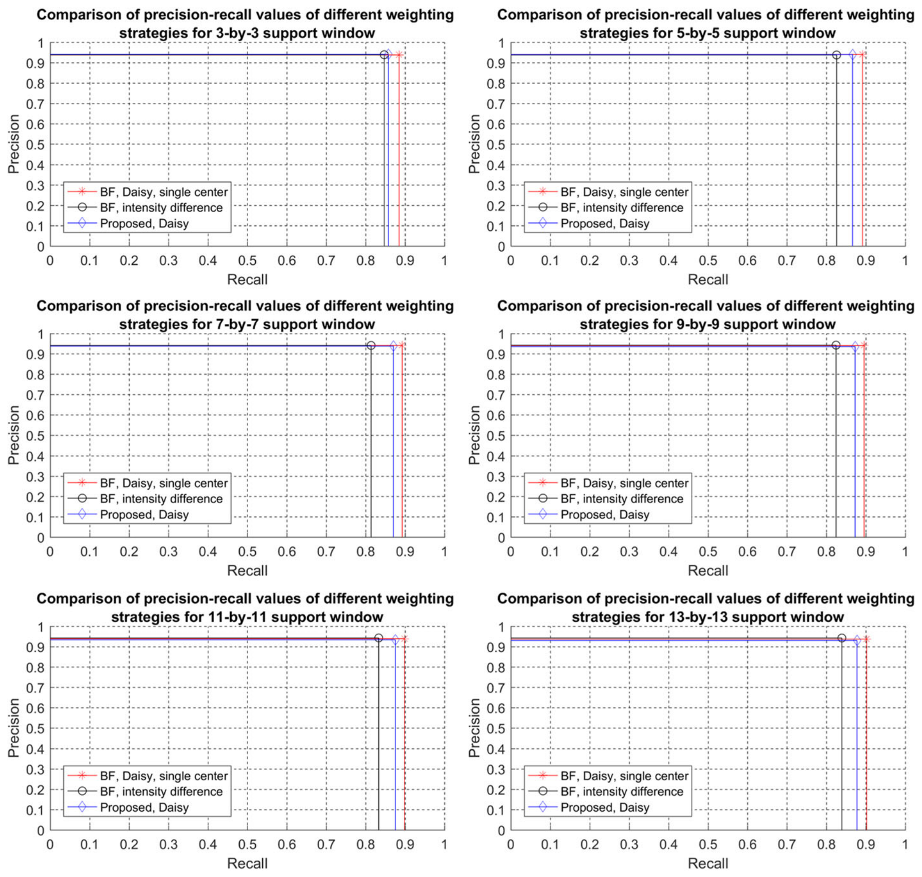

Fig. 6 Performance comparison of the proposed and two other bilateral-filtering-based weighing strategies on the stereo pair in Fig. 4(a)

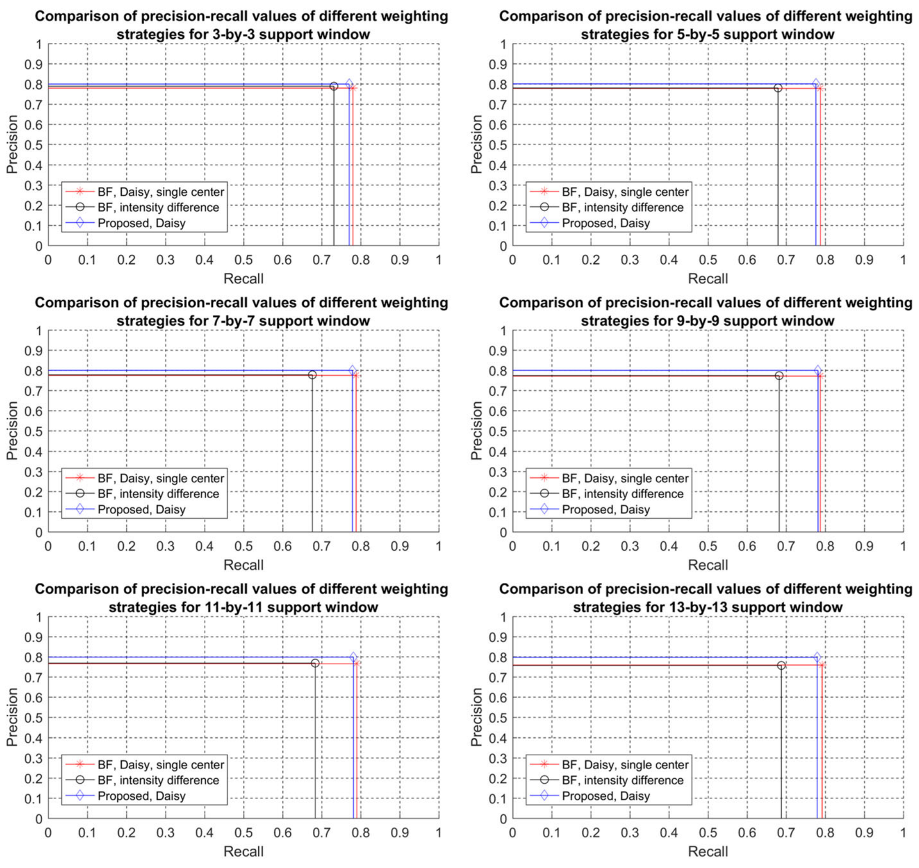

Fig. 7 Performance comparison of the proposed and two other bilateral-filtering-based weighing strategies on the stereo pair in Fig. 4(b)

intensity difference. The recalls of the proposed weighting strategy are only slightly lower than those of the bilateral filter with a single center. However, the proposed weighting strategy is much faster than the bilateral filter with a single center.

Next, we analyzed another weighting strategy, which is based on the guided-image filter [15]. The cost volume was computed in two ways. One way was to use a combination of the truncated absolute difference of the color and the gradient, as in [15]. The other way was to use the DAISY feature vector. On the two stereo pairs, this strategy did not produce good disparity estimates, even using both ways of computing the cost volumes. This result suggests that the weighting strategy based on the guided-image filter may not be suitable for wide-baseline stereo matching.

3.5 Comparison with other methods

In this section, we compare, using the two datasets described in Section 3.1, the proposed cost aggregation method with two benchmark methods: Min et al.'s method [14], and the census transform [39, 40].

Min et al. developed an approximate strategy to optimize cost aggregation [14]. This strategy, originally based on the truncated absolute difference (TAD) matching cost, consists of two parts: *disparity candidate selection* and *joint histogram-based aggregation*. In our implementation, the strategy was extended to feature vectors, by replacing the TAD matching costs with the dissimilarities between feature vectors. To enable a fair comparison, the DAISY descriptors were stored in the memory for the disparity candidate selection.

The census transform is one of the most popular techniques to compute matching costs for stereo vision. This method creates an encoded bit string for the pixels in a window. If the intensity of a pixel is lower than that of the center pixel of the window, the corresponding bit is set to one; otherwise, it is set to zero. This way, the census transform describes the spatial structure in the window. A census-transformed image pair is matched by computing the Hamming distance between the bit strings. Our C++ implementation of the census transform was adapted from Banks and Corke's source code that accompanies their publication [41]. In the experiments, we used

Table 2 Performance comparison of three methods on the fountain and HerzJesu dataset

Method	Fountain-A		Fountain-B		HerzJesu		Time (sec)
	Precision	Recall	Precision	Recall	Precision	Recall	
Min et al. [14]	0.904	0.838	0.831	0.765	0.907	0.845	128
Census transform [41]	0.981	0.788	0.975	0.714	0.978	0.759	105
Proposed method	0.953	0.809	0.934	0.751	0.967	0.823	53

a census window size of 31×31 pixels, which is compatible with the computation of the DAISY feature vector.

Both benchmark methods rely on the left-right cross-check to detect pixels with unreliable disparities. That is, the disparity map for both the left and right images are computed. First, for a pixel p in the left image, its counterpart q in the right image is found. Then, for pixel q in the right image, its counterpart p' in the left image is found. Finally, the disparity value for pixel p is considered as correctly estimated if $\|p - p'\| \leq \epsilon$, where ϵ is a small threshold. For a fair comparison, we also applied the left-right crosscheck with the proposed method. An evaluation of different values for ϵ indicated that $\epsilon = 2$ gave a good trade-off between the precision and recall rate. Therefore, in the following experiments we selected $\epsilon = 2$.

3.5.1 *Comparison on the fountain and HerzJesu dataset*

In this experiment, we compared three methods on the Fountain and herzJesu Dataset. Table 2 presents the performance in terms of precision, recall and running time of the three methods on the Fountain and HerzJesu Dataset. Min et al.'s method [14] yields the highest recall rate and the lowest precision rate among the three methods. The census transform provides the lowest recall rate and the highest precision rate among the three methods. The high precision rate by the census transform is attributed to its ability to capture the spatial structure of local windows using the encoded bit strings. In comparison, the proposed method yields a middle rank in terms of precision and recall rates among the three methods. In terms of processing time, the proposed method is 2.42 times faster than Min et al.'s method, and 1.98 times faster than the census transform.

To further investigate stereo matching performance, we compute *the proportion of correctly estimated non-occluded pixels*, which is the product of the precision and recall rates, see Eq. (12). The results, shown in Fig. 8, indicate that the proposed method has a higher *precision* $\times$ *recall* score than the other two methods in most of the 60 stereo pairs. The average value of the *precision* $\times$ *recall* for the proposed method is 0.757, higher than that of the census transform (0.738) and Min et al.'s method (0.721).

Figure 9 presents representative results of depth estimation on this dataset. In this figure, Columns 1 and 2 are the input stereo pair; Column 3 is the output of the proposed method, where occluded pixels are shown in pink color. Column 4 compares the proposed method and Min et al.'s method [14]: If the depth of a pixel is correctly estimated by the proposed method but incorrectly estimated by Min et al.'s method, it is represented with

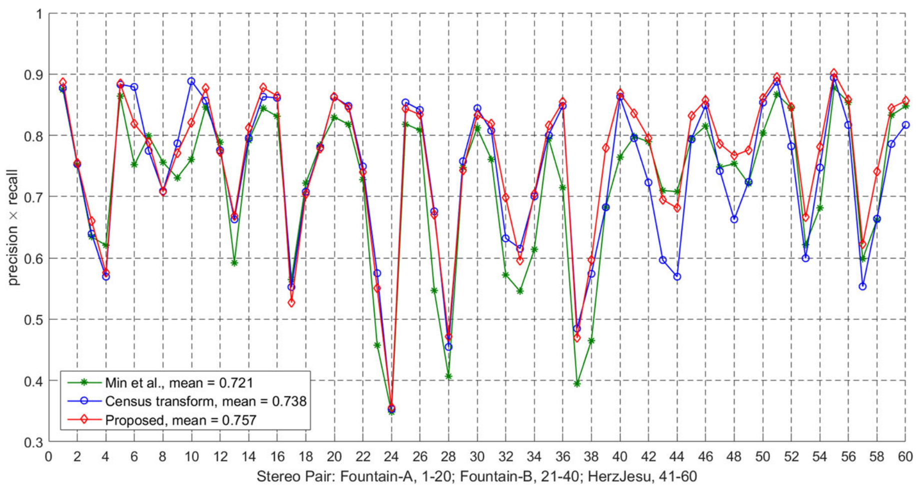

Fig. 8 Performance comparison of three methods on the Fountain and HerzJesu Dataset

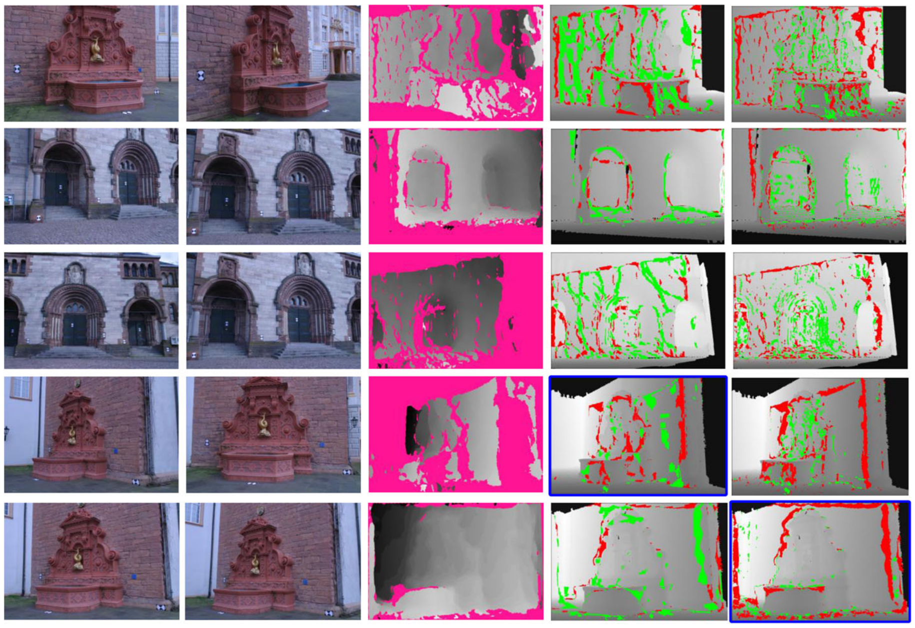

Fig. 9 Representative examples of depth estimation from the Fountain and HerzJesu Dataset. Columns 1 and 2: the stereo pairs. Column 3: depth estimation results by the proposed method, where pink pixels denote occluded pixels. Column 4: comparison of the proposed method with Min et al.'s method [14]. If the depth of a pixel is correctly estimated by the proposed method but incorrectly by Min et al.'s method, the pixel is shown in *green* color; otherwise, it is shown in *red* color. Column 5: comparison of the proposed method with the census transform using the same color convention. A blue border around the image indicates the case where the proposed method performs worse compared to the other methods

green color; otherwise, it is represented with *red* color. Similarly, Column 5 compares the proposed method with the census transform using the same color convention. In Columns 4 and 5, a blue border around the image indicates the case where the proposed method performs worse than the other method. The distribution of the color pixels in Columns 4 and 5 also reveal the stength of a given method in different parts of the image. Generally, the proposed method outperforms Min et al.'s method in most parts of the image. However, for sharp boundaries (e.g., the boundaries between the red wall and the white wall in the last two rows of Fig. 9), the census transform works better. This performance gap can be attributed to the different ways of forming the feature descriptors. The DAISY descriptor samples at sparse locations in a local area (only at the center and petals of the DAISY "flower"),

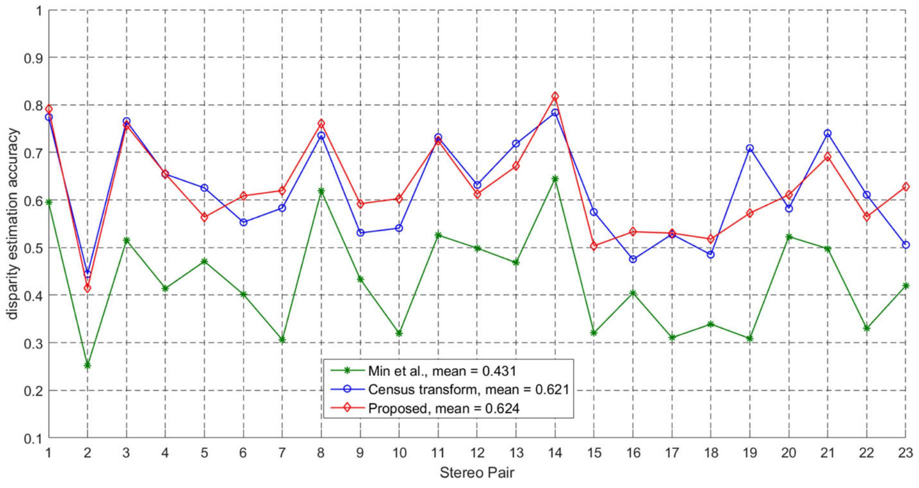

Fig. 10 Performance comparison of three methods on the 2014 Middlebury Stereo Dataset

while a bit string used in the census transform takes every location in the local area into account. Consequently, the DAISY descriptor may miss out some boundary pixels if its sample locations are far from them. This performance gap between the proposed method and the census transform can be reduced by using more advanced feature descriptors.

3.5.2 Comparison on the 2014 Middlebury stereo dataset

In this experiment, we compared three methods on the 2014 Middlebury Stereo Dataset. Figure 10 presents the disparity estimation accuracy of the three methods on this dataset. For a fair comparison of the three methods, we discarded borders of half the census window width from the results. This is because the census transform generated zero pixels along the borders of census-transformed images. The proposed method and the census transform perform significantly better than Min et al.'s method. The average disparity estimation accuracy for the proposed method, the census transform, and Min et al.'s method is 0.624, 0.621, and 0.433, respectively. Of the 23 stereo pairs, the proposed method achieves the highest disparity estimation accuracy for 13 stereo pairs.

Note that the 2014 Middlebury Stereo Dataset contains less texture compared with the Fountain and HerzJesu Dataset. This result indicates that the proposed method and the census transform are more stable for textureless scenes.

Figure 11 presents representative examples of disparity estimation on the 2014 Middlebury Stereo Dataset. Columns 1 and 2 are the input stereo pairs. Column 3 is the output of the proposed method, where pixels with incorrect disparity estimations are shown in pink. Columns 4 and 5 compare the proposed method with Min et al.'s method and the census transform, using the same color convention as in Fig. 9. However, because this dataset has ground-truth disparity maps instead of depth maps, the outputs are overlaid on the ground-truth disparity maps. The results again show that at sharp boundaries (e.g., the "sticks" stereo pair in the last row of Fig. 11), the proposed method may perform less successfully than the census transform.

4 Conclusion

In this paper, a feature-vector-based cost aggregation algorithm was proposed for wide-baseline stereo matching, and evaluated using the DAISY feature vector. The proposed algorithm improved the efficiency of cost aggregation by combining a Per-Column-Cost matrix and a feature-vector-based weighting strategy. The paper also

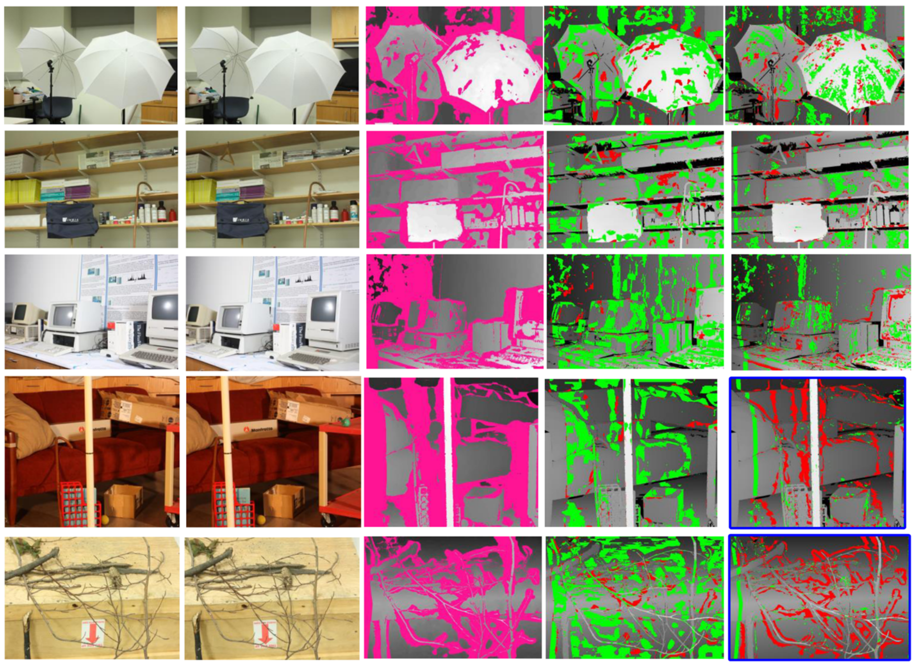

Fig. 11 Representative examples of disparity estimation from the 2014 Middlebury Stereo Dataset. Columns 1 and 2: the stereo pairs. Column 3: disparity estimation results by the proposed method, where *pink* denote pixels with incorrect disparity estimations. Column 4: comparison of the proposed method with Min et al.'s method [14]. If the disparity of a pixel is correctly estimated by the proposed method but incorrectly by Min et al.'s method, the pixel is shown in *green* color; otherwise, it is shown in *red* color. Column 5: comparison of the proposed method with the census transform using the same color convention. A blue border around the image indicates the case where the proposed method performs worse compared to the other methods

presented a detailed analysis of both time and storage complexity of the proposed method. The new method was extensively tested and compared with two benchmark methods on two wide-baseline datasets. With growing research in feature detectors and visual descriptors, it can be envisaged that the proposed method will be attractive for stereo matching applications where feature vectors are used.

Among several possibilities, one direction for future work is to further accelerate the speed of the proposed method. For example, once the Per-Column-Cost matrix Γ and the Per-Column-Weight matrix Φ are computed, the disparity values for the pixels in a row can be computed in parallel. Another direction is to combine the proposed method with feature vectors that are found via deep learning [31–34].

Acknowledgements
We gratefully thank the anonymous reviewers and Associate Editor for the constructive and detailed comments that helped improve the paper.

Funding
This research was partly supported two grants from University of Electronic Science and Technology of China (Grant No. LJT115010701037 and Grant No. Y02002010701026), and a grant from the Australian Research Council.

Availability of data and materials
The Fountain and HerzJesu Dataset is available from http://cvlab.epfl.ch/software/daisy. The 2014 Middlebury Stereo Dataset is available from http://vision.middlebury.edu/stereo/data/2014.

Authors' contributions
The original idea of the research was proposed by XP, but was largely inspired by his discussions with AB. SLP contributed to the experiment analysis and paper writing. All three authors worked closely during the preparation and revision of the manuscript. All authors read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Author details
[1]School of Electrical, Computer and Telecommunications Engineering, University of Wollongong, Northfields Ave, Wollongong, NSW 2522, Australia. [2]School of Automation Engineering, University of Electronic Science and Technology of China, Qingshuihe Campus, 2006 Xiyuan Ave, West Hi-Tech Zone, 611731 Chengdu, Sichuan, China. [3]College of Science and Engineering, Hamad Bin Khalifa University, Education City, PO Box 5825, Doha, Qatar.

Received: 27 April 2017 Accepted: 22 January 2018
Published online: 11 April 2018

References

1. D Scharstein, R Szeliski, A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. Int. J. Comput. Vis. **47**(1-3), 7–42 (2002)
2. C Strecha, R Fransens, L Van Gool, in *Proc. Computer Vision and Pattern Recognition (CVPR)*. Combined depth and outlier estimation in multi-view stereo, (2006), pp. 2394–2401
3. X Huang, in *Proc. 26th DAGM Symposium*. Cooperative optimization for energy minimization in computer vision: a case study of stereo matching, (2004), pp. 302–309
4. Z Wang, Z Zheng, in *Proc. Computer Vision and Pattern Recognition (CVPR)*. A region based stereo matching algorithm using cooperative optimization, (2008), pp. 1–8
5. Y Boykov, O Veksler, R Zabih, Fast approximate energy minimization via graph cuts. IEEE Trans. Pattern Anal. Mach. Intell. **23**(11), 1222–1239 (2001)
6. JS Yedidia, WT Freeman, Y Weiss, in *Advances in Neural Information Processing Systems (NIPS)*. Generalized belief propagation, (2000), pp. 689–695
7. MJ Wainwright, TS Jaakkola, AS Willsky, Map estimation via agreement on trees: message-passing and linear programming. IEEE Trans. Inf. Theory. **51**(11), 3697–3717 (2005)
8. R Szeliski, R Zabih, D Scharstein, O Veksler, V Kolmogorov, A Agarwala, M Tappen, C Rother, A comparative study of energy minimization methods for Markov random fields with smoothness-based priors. IEEE Trans. Pattern Anal. Mach. Intell. **30**(6), 1068–1080 (2008)
9. L Di Stefano, M Marchionni, S Mattoccia, A fast area-based stereo matching algorithm. Image Vis. Comput. **22**(12), 983–1005 (2004)
10. F Tombari, S Mattoccia, L Di Stefano, E Addimanda, in *Proc. Computer Vision and Pattern Recognition (CVPR)*. Classification and evaluation of cost aggregation methods for stereo correspondence, (2008), pp. 1–8
11. H Hirschmüller, Stereo processing by semiglobal matching and mutual information. IEEE Trans. Pattern Anal. Mach. Intell. **30**(2), 328–341 (2008)
12. KJ Yoon, IS Kweon, Adaptive support-weight approach for correspondence search. IEEE Trans. Pattern Anal. Mach. Intell. **28**(5), 650–656 (2006)
13. S Mattoccia, S Giardino, A Gambini, in *Proc. Asian Conference on Computer Vision (ACCV)*. Accurate and efficient cost aggregation strategy for stereo correspondence based on approximated joint bilateral filtering, (2009), pp. 371–380
14. D Min, J Lu, MN Do, in *Proc. International Conference on Computer Vision (ICCV)*. A revisit to cost aggregation in stereo matching: how far can we reduce its computational redundancy? (2011), pp. 1567–1574
15. A Hosni, C Rhemann, M Bleyer, C Rother, M Gelautz, Fast cost-volume filtering for visual correspondence and beyond. IEEE Trans. Pattern Anal. Mach. Intell. **35**(2), 504–511 (2013)
16. K He, J Sun, X Tang, Guided image filtering. IEEE Trans. Pattern Anal. Mach. Intell. **35**(6), 1397–1409 (2013)
17. Q Yang, in *Proc. Computer Vision and Pattern Recognition (CVPR)*. A non-local cost aggregation method for stereo matching, (2012), pp. 1402–1409
18. X Mei, X Sun, W Dong, H Wang, X Zhang, in *Proc. Computer Vision and Pattern Recognition (CVPR)*. Segment-tree based cost aggregation for stereo matching, (2013), pp. 313–320
19. K Zhang, Y Fang, D Min, L Sun, S Yang, S Yan, Q Tian, in *Proc. Computer Vision and Pattern Recognition (CVPR)*. Cross-scale cost aggregation for stereo matching, (2014), pp. 1590–1597
20. M Calonder, V Lepetit, M Ozuysal, T Trzcinski, C Strecha, P Fua, BRIEF: computing a local binary descriptor very fast. IEEE Trans. Pattern Anal. Mach. Intell. **34**(7), 1281–1298 (2012)
21. S Leutenegger, M Chli, RY Siegwart, in *Proc. International Conference on Computer Vision (ICCV)*. BRISK: Binary robust invariant scalable keypoints, (2011), pp. 2548–2555
22. A Alahi, R Ortiz, P Vandergheynst, in *Proc. Computer Vision and Pattern Recognition (CVPR)*. FREAK: fast retina keypoint, (2012), pp. 510–517
23. E Rublee, V Rabaud, K Konolige, G Bradski, in *Proc. International Conference on Computer Vision (ICCV)*. ORB: an efficient alternative to SIFT or SURF, (2011), pp. 2564–2571
24. DG Lowe, Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**(2), 91–110 (2004)
25. H Bay, A Ess, T Tuytelaars, L Van Gool, Speeded-up robust features (SURF). Comput. Vis. Image Understanding. **110**(3), 346–359 (2008)
26. J Heinly, E Dunn, JM Frahm, in *Proc. European Conference on Computer Vision (ECCV)*. Comparative evaluation of binary features, (2012), pp. 759–773
27. N Khan, B McCane, S Mills, Better than SIFT? Mach. Vis. Appl. **26**(6), 819–836 (2015)
28. E Tola, V Lepetit, P Fua, DAISY: an efficient dense descriptor applied to wide-baseline stereo. IEEE Trans. Pattern Anal. Mach. Intell. **32**(5), 815–830 (2010)
29. C Liu, J Yuen, A Torralba, SIFT flow: dense correspondence across scenes and its applications. IEEE Trans. Pattern Anal. Mach. Intell. **33**(5), 978–994 (2011)
30. K Zhang, J Li, Y Li, W Hu, L Sun, S Yang, in *Proc. International Conference on Pattern Recognition (ICPR)*. Binary stereo matching, (2012), pp. 356–359

31. S Zagoruyko, N Komodakis, in *Proc. Computer Vision and Pattern Recognition (CVPR)*. Learning to compare image patches via convolutional neural networks, (2015), pp. 4353–4361
32. J Žbontar, Y LeCun, in *Proc. Computer Vision and Pattern Recognition (CVPR)*. Computing the stereo matching cost with a convolutional neural network, (2015), pp. 1592–1599
33. Z Chen, X Sun, L Wang, Y Yu, C Huang, in *Proc. International Conference on Computer Vision (ICCV)*. A deep visual correspondence embedding model for stereo matching costs, (2015), pp. 972–980
34. W Luo, AG Schwing, R Urtasun, in *Proc. Computer Vision and Pattern Recognition (CVPR)*. Efficient deep learning for stereo matching, (2016), pp. 5695–5703
35. N Mayer, E Ilg, P Hausser, P Fischer, D Cremers, A Dosovitskiy, T Brox, in *Proc. Computer Vision and Pattern Recognition (CVPR)*. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation, (2016), pp. 4040–4048
36. F Crow, Summed-area tables for texture mapping. Comput. Graphics. **18**(3), 207–212 (1984)
37. C Strecha, W von Hansen, L Van Gool, P Fua, U Thoennessen, in *Proc. Computer Vision and Pattern Recognition (CVPR)*. On benchmarking camera calibration and multi-view stereo for high resolution imagery, (2008), pp. 1–8
38. D Scharstein, H Hirschmller, Y Kitajima, G Krathwohl, N Nesic, X Wang, P Westling, in *Proc. German Conference on Pattern Recognition (GCPR)*. High-resolution stereo datasets with subpixel-accurate ground truth, (2014), pp. 31–42
39. R Zabih, J Wood, in *Proc. European Conference on Computer Vision (ECCV)*. Non-parametric local transforms for computing visual correspondence, (1994), pp. 151–158
40. H Hirschmüller, D Scharstein, Evaluation of stereo matching costs on images with radiometric differences. IEEE Trans. Pattern Anal. Mach. Intell. **31**(9), 1582–1599 (2009)
41. J Banks, P Corke, Quantitative evaluation of matching methods and validity measures for stereo vision. Int. J. Robot. Res. **20**(7), 512–532 (2001)

Submit your manuscript to a SpringerOpen journal and benefit from:

► Convenient online submission
► Rigorous peer review
► Open access: articles freely available online
► High visibility within the field
► Retaining the copyright to your article

Submit your next manuscript at ► springeropen.com