Multimedia Tools and Applications (2019) 78:15779–15800
https://doi.org/10.1007/s11042-018-6993-2

Oriented-linear-tree based cost aggregation for stereo matching

Wenhuan Wu[1,2] · Hong Zhu[1] · Qian Zhang[3]

Received: 25 April 2018 / Revised: 6 November 2018 / Accepted: 28 November 2018 /
Published online: 8 December 2018

Abstract
Matching cost aggregation is one of the most important steps in dense stereo correspondence, and non-local cost aggregation methods based on tree structures have been widely studied recently. In this paper, we analyze the shortcomings of both the local window-based aggregation methods and the non-local tree-based aggregation methods, and propose a novel *oriented linear tree* structure for each pixel to perform the non-local cost aggregation strategy. Firstly, each pixel in the image has an oriented linear tree rooted on it and each oriented linear tree consists of multiple 1D paths from different directions. Compared to other spanning trees, our oriented linear trees don't need to be additionally constructed beforeh and since they are naturally embedded in the original image. Moreover, each root pixel not only gets supports from adjacent pixels within its local support window, but also receives supports from the other pixels along all 1D paths. Secondly, for each pixel lying on the same 1D path, we can at the same time calculate their aggregated cost along their path by traversing the path back and forth twice. Finally, the final aggregated cost for each root pixel can be calculated by summing the aggregated costs from all 1D paths. Performance evaluation on the Middlebury and KITTI datasets shows that the proposed method outperforms the current state-of-the-art aggregation methods.

Keywords Stereo matching · Cost aggregation · Oriented linear tree · Cost volume · Edge-aware filtering

✉ Hong Zhu
zhuhong@xaut.edu.cn

[1] School of Automation and Information Engineering, Xi'an University of Technology, Xi'an 710048, China

[2] School of Electrical and Information Engineering, Hubei University of Automotive Technology, Shiyan 442002, China

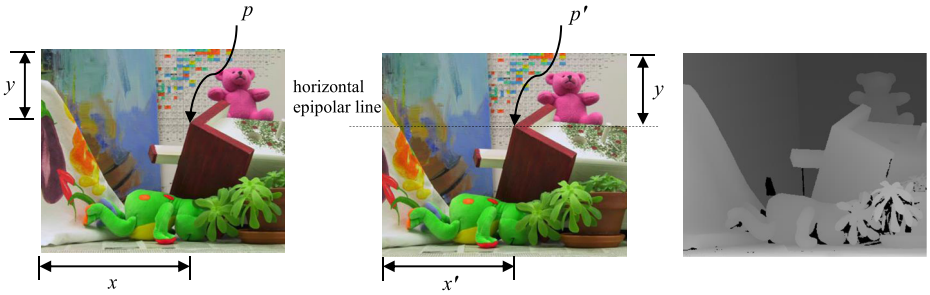
[3] Department of Information Science and Technology, Taishan University, Tai'an 271021, China

1 Introduction

Dense two-frame stereo matching has been one of the fundamental and most extensively studied problems in computer vision, and it is employed in a variety of vision applications such as 3D scene reconstruction [7, 41], image-based rendering [21, 57] and autonomous driving [10, 14], since it represents an inexpensive way for recovering 3D information of a real world scene. The inputs of stereo matching are a pair of images from the same scene which are captured by a stereo camera system. For stereo matching, it is assumed that the input image pair is rectified [48] as well as one of the input two images is the reference image and the other is the target image. The task of stereo matching is to find the corresponding pixel in the target image for each pixel of the reference image. Because the input image pair is rectified and their epipolar lines are horizontal, the search of pixel correspondences between the image pair can be performed along horizontal lines as illustrated in Fig. 1. The *disparity* is the difference between horizontal coordinates of a pair of corresponding pixels. The resultant output of stereo matching is a dense disparity map by finding all the correspondences between the input image pair. The disparity map can be used to estimate the depth information according to the triangulation principle [12]. As such, the disparity map can not only be applied to reconstruct 3D structures in the scene, but it can also be used as an important clue or feature of visual tracking [24–29] to improve the tracking performance. Due to the ambiguous nature of the matching problem and existing noise, occlusion, or distortion in the images, stereo matching is very challenging and the recovery of an accurate disparity map still remains an open problem.

According to the analysis and taxonomy scheme proposed in [39], stereo matching algorithms can be categorized into global and local methods. Stereo matching algorithms generally perform the following four steps:

1. cost computation / cost volume estimation;
2. cost aggregation;
3. disparity computation / optimization;
4. disparity refinement.



(a) Reference (left) image (b) Target (right) image (c) Disparity map

Fig. 1 Stereo matching of Middlebury [40] *Teddy* image pair. Here the left image is the reference image and the right image is the target image. For the corner pixel p in the left image whose coordinates are (x, y), its corresponding pixel in the right image must be on the horizontal epipolar line with vertical coordinate y. Thus we have to search for the corresponding pixel of pixel p along the horizontal line shown by the dotted line in the right image. Supposing that the corner pixel p' in the right image with coordinates (x', y) is the true corresponding pixel of pixel p, the disparity d of pixel p is the difference between the horizontal coordinate of pixels p and p', i.e., $d = x - x'$. The disparity map in (c) can be generated by finding all the correspondences between the image pair

In cost computation, a 3D cost volume as the initial *disparity space image* is generated by computing matching costs for each pixel at all possible disparity levels. In cost aggregation, the matching costs are aggregated over each pixel's support region. Then, the initial disparity map can be generated by computing the disparity of each pixel with global or local optimization methods. As such, stereo matching algorithms can be classified as either global methods or local methods. Lastly, the final disparity map can be obtained by refining the initial disparity map with various post-processing methods. Similar to the multi-task learning categorization problem, global stereo matching methods [2, 15, 22, 23, 35, 42, 46, 47, 49, 52] simultaneously assign an optimal disparity level for each pixel by minimizing an energy function [22] defined on the Markov random field model. Global stereo matching methods incorporate spatial smoothness between neighboring pixels as a regularization term into the energy function. In contrast, for some multi-task learning problems in time domain, such as complex activity recognition [30–32] and career path prediction [33], the temporal relatedness among different tasks is imposed into an appropriate multi-task learning model. Generally speaking, the energy function of global stereo matching methods can be minimized through a discrete global optimization algorithm, e.g., graph cut [2, 23], belief propagation [42, 52] or dynamic programming [15, 46]. Despite the reliable matching results are obtained, global algorithms are often time-consuming. In contrast, after performing cost aggregations within support regions, local algorithms [1, 4, 6, 9, 17, 18, 20, 34, 38, 43, 45, 53, 55] perform a local "*winner-take-all*" (WTA) optimization [39], that is, for each pixel simply choose the disparity associated with the minimum aggregated cost value as its final disparity. Local algorithms are usually computationally cheaper than global algorithms. Among these steps, the quality of cost aggregation has significant impact on the accuracy and the efficiency of stereo algorithms as described in [19, 39, 44]. It is a key ingredient for state-of-the art local algorithms [4, 17, 18, 34, 38, 43, 53, 55] and a primary building block for some top-performing global algorithms [35, 49, 52]. Therefore, in this paper, we mainly concentrate on cost aggregation.

The cost aggregation of a pixel is traditionally performed by summing or averaging the costs of the pixel itself and all its neighboring pixels within a local window. Actually, window-based cost aggregation can be viewed as image filtering over the cost volume [18, 56]. While simple box filter is adopted for cost aggregation, it tends to blur the depth boundaries and produces well known "edge fatten" effect. Hence, to obtain accurate results at depth boundaries, variable or multiple windows-based algorithms [6, 20, 45] were proposed to find an optimal support window for each pixel adaptively. In fact, rectangular and constrained-shaped window models may be inappropriate for pixels near arbitrarily shaped depth discontinuities. Additionally, doing cost aggregation over multiple windows for each pixel is computationally expensive. To resolve this problem, segmentation-based methods [1, 9] use segmented regions with arbitrary sizes and shapes as support windows, which is implicitly assumed that the disparity varies smoothly in each segmented region. However, these methods require precise color segmentation [5] that is very difficult when dealing with highly textured images.

The adaptive support weight (ASW) methods [4, 17, 18, 34, 38, 43, 53] try to assign appropriate support weights to the pixels in a support window based on the proximity and color distances to the center pixel. These methods can be treated as segmenting the reference image in a "soft" way. Yoon and Kweon [53] firstly introduced the bilateral filter to compute the support weights of pixels and its cost aggregation process can be understood as filtering the cost volume with bilateral filter [34]. Despite bilateral filter can effectively preserver depth boundaries, this method is computationally expensive. Since then, various edge-aware filtering techniques have been explored for cost aggregation, such as segmentation-based adaptive

support [43], dual-cross-bilateral filter [38] and geodesic weight support [17]. Rhemann et al. [18] successfully applied the guided image filter [13] with running time independent of the kernel size to cost aggregation. This method can outperform most local methods on Middlebury benchmark [40] in terms of both speed and accuracy. However, the selected support regions of the ASW-based algorithms are still limited in a pre-defined window of fixed size. Due to this reason, as the same as the traditional window-based cost aggregation algorithms, this kind of algorithms is vulnerable to low or repetitive texture.

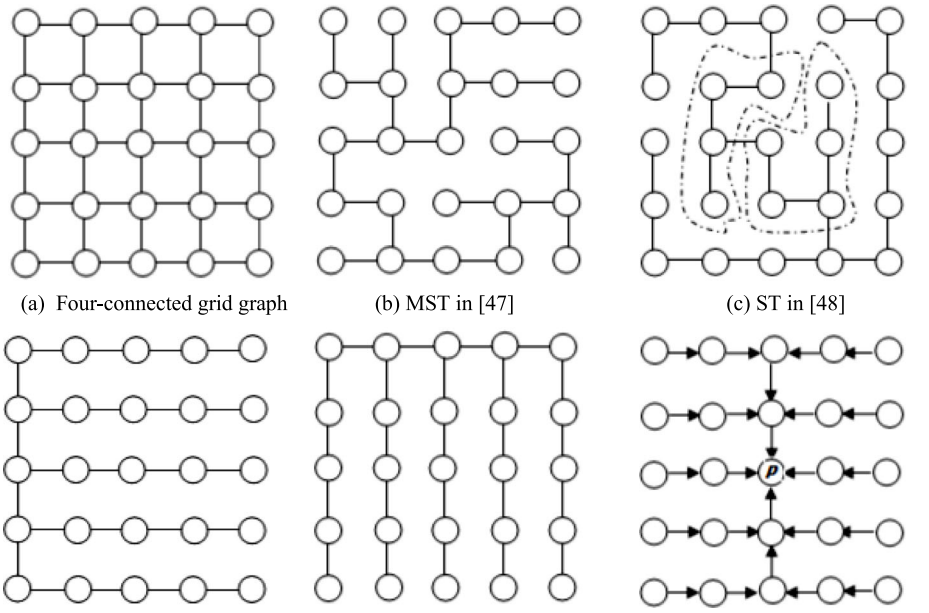
Recently, Yang first proposed a non-local cost aggregation algorithm [50, 51] by constructing a minimum spanning tree (MST) over the reference image. Then a pixel can get supports from all the other pixels of the image along their shortest paths on the tree structure. Different from previous local methods that rely on local support regions, the non-local method performs cost aggregation for each pixel over the whole image. By enforcing tight connections for the pixels inside the same segment, Mei et al. proposed a segment-tree structure (ST) instead of MST to perform the non-local cost aggregation [36]. Considering that the tree structure for MST or ST is not unique since there are a lot of edges which have the same weight, a cross-trees structure consisting of a horizontal tree and a vertical tree is employed in [3]. Then the non-local cost aggregation is done twice by traversing the two crossed trees successively. However, due to the cross-shaped structure like in [4, 55], the connectivity between two adjacent pixels in diagonal directions cannot be maintained.

In this paper, a novel non-local algorithm based on oriented linear tree structure is proposed for cost aggregation. In the reference image, each pixel as a root node corresponds to an individual oriented linear tree. Each oriented tree includes multiple 1D paths from different directions and these 1D paths intersect at the root node. Compared to other spanning trees such as MST and its variants, our oriented linear trees don't need to be additionally constructed since the reference image naturally contains these linear trees. Any root pixel not only receives supports from all adjacent pixels within its local support window, but also gets the supports from all the other pixels in the whole image. It means that the proposed method integrates the advantages of local window-based aggregation and non-local aggregation. Furthermore, the final aggregated cost for each pixel can be obtained by summing the aggregated costs from all its paths. Obviously, a brute force implementation would be really time-consuming. However, for each pixel lying on the same 1D path, we can in one breath calculate their aggregated cost along their path by using two-passes traversals. This results in the low computational complexity of the proposed algorithm which is linear to the number of pixels and disparity levels.

The remainder of this paper is organized as follows. In Section 2, we summarize the related work. The cost aggregation problem is formulated in Section 3. In Section 4, we describe the novel oriented linear trees structure and detail how to perform the cost aggregation on it. Experiments results and analyses are presented in Section 5. Finally, we draw the conclusions and discuss the future work in Section 6.

2 Related work

Here we mainly focus on the non-local tree-based cost aggregation algorithms by comparing different tree construction techniques. Detailed comparisons and discussions of various local cost aggregation methods can be found in recent surveys [19, 44].



(a) Four-connected grid graph (b) MST in [47] (c) ST in [48]

(d) The horizontal tree in [49] (e) The vertical tree in [49] (f) Cost aggregation of pixel p in [49]

Fig. 2 Different tree structures of previous non-local methods. **a** Four-connected grid of the image. **b** The minimum spanning tree (MST) in [50]. Each pixel gets supports from all other pixels of the image along unique paths on the MST. **c** The segment Tree (ST) in [36]. Each segment enveloped by dotted curve contains a sub MST. Each pixel receives supports from all other pixels of the image along unique paths on the ST. **d** The horizontal tree in [3]. **e** The vertical tree in [3]. **f** For pixel p, its sequential cost aggregation on the two crossed trees. Firstly, the initial cost volume is aggregated on the whole horizontal tree and intermediate results are produced. Next, the final aggregated cost of pixel p can be obtained by aggregating the intermediate results on the p's column of the vertical tree

In fact, all the non-local cost aggregation methods essentially still adopt the adaptive support weight strategy, but they extend the support window to the entire image. In previous tree-based methods, the reference image is treated as a four-connected, undirected planar graph as shown in Fig. 2a. The nodes are all image pixels and the edges are all the edges between neighboring pixels. The edge weight between two neighboring pixels is their color difference. In [50], a minimum spanning tree (MST) as shown in Fig. 2b is derived from this graph. The intuition is that an edge with small weight is less likely to cross the depth boundaries since its two endpoints have high color similarity. Edges with large weights thus will be removed during spanning tree construction. By traversing all nodes of the tree during the cost aggregation process, each pixel can receive different supports from all the other pixels in the image. Here, if an edge is selected, its two endpoints are connected directly and they can get support from each other. Additionally, two nodes which are not connected directly by an edge can also get support from each other through a shortest path on the tree. From the above analysis, it can be seen that MST structure is not unique since there are a lot of edges with equal weights. The tree is also sensitive to highly textured region and local noise. Moreover, since the image graph is the four-connected, for any node, only less than four connected edges are often selected and the diagonal connected edges with 8-connectivity cannot be selected. An observation is that MST neglects the spatial information between adjacent pixels.

By incorporating the segmentation information into MST, segment-tree (ST) structure [36] is constructed as shown in Fig. 2c. In this method, the image graph is firstly divided into a set of coherent segment, then for each segment a sub-tree is constructed, and all the sub-trees are connected to build the ST structure. Though its accuracy is improved, obviously, ST structure is easily influenced by the accuracy of image segmentation and constructing a sub-MST in each segment may still suffer from the problems of the MST structure. Recently, a novel cross-trees structure consisting of a horizontal-tree and a vertical-tree is proposed [3]. Fig. 2d, e show the structures of the two crossed trees. As illustrated in Fig. 2f, firstly the cost aggregation is performed on the horizontal tree, and the matching cost of each pixel is updated with the aggregated cost. Then, the cost aggregation is carried out on the vertical lines to estimate the final aggregated cost. In this method, there is an explicit smoothness assumption by truncating the color difference between two neighboring pixels. Considering that performing the non-local cost aggregation on the cross-trees directly will blur the depth boundaries, some prior information such as edge prior is used to prevent the false smoothing at the depth boundaries.

As can be seen from the above discussion, the above proposed tree structures emphasize the color similarity only and then the siblings of most nodes are removed in the process of building a special tree structure, so that the original Markov random field embedded in the reference image may be broken. No doubt, the proper supports from the adjacent pixels within local region (e.g., eight-connected region) are very important for any pixel to reduce the matching ambiguity. In this paper, a novel oriented linear tree structure consisting of multiple 1D paths is presented. In this structure, different weights between nodes on the same path are given based on their geodesic distance. In addition, any pixel not only gets correct support from all adjacent pixels in a local window, but the other pixels outside the local window also support it along various linear paths.

3 Cost aggregation formulation

In this section, the cost aggregation is formulated by filtering the cost volume. Under this formulation, different choices of weight functions lead to different cost aggregation methods.

Firstly, let I and I' be respectively the left image and right image of a rectified stereo image pair [48] and $\mathcal{D}$ denote the set of allowed disparity levels. Here the left image I is chosen as the reference image. The matching cost computation step is formulated as a function f: $\mathbb{R}^{W\times H\times M} \times \mathbb{R}^{W\times H\times M} \longmapsto \mathbb{R}^{W\times H\times|\mathcal{D}|}$, where W, H are the width and height of input images, M represents color channels of input images (i.e. $M = 3$ for RGB color images or $M = 1$ for gray images) and $|\mathcal{D}|$ is the number of disparity levels in $\mathcal{D}$. Thus, for the given stereo image pair I, $I' \in \mathbb{R}^{W\times H\times M}$, by applying cost computation function:

$$C = f\left(I, I'\right) \tag{1}$$

We can get the 3D cost volume $C \in \mathbb{R}^{W\times H\times|\mathcal{D}|}$, which represents matching costs for each pixel in the reference image at all possible disparity levels. For a single pixel p with coordinates (x, y), its cost at disparity level $d \in \mathcal{D}$ can be denoted as a scalar, $C(p, d)$.Various methods can be

used to compute the cost volume as described in [16]. For example, the *AD-Gradient* (i.e. absolute difference of color and gradient) cost function defined in [18] can be formulated as:

$$\begin{aligned} C(p,d) = (1-\alpha)\cdot\min\left(\frac{1}{M}\left\|I(p)-I'(p_d)\right\|_1, \tau_1\right) \\ + \alpha\cdot\min\left(\left|\nabla_x I(p)-\nabla_x I'(p_d)\right|, \tau_2\right) \end{aligned} \tag{2}$$

Here $I(p)$ denotes the color vector of pixel p. ∇_x is the grayscale gradient in x direction. p_d in the image I' is the corresponding pixel of p with a disparity d, i.e., $p_d = (x-d, y)$. To make the similarity measurement between p and p_d be linear with color difference, the average absolute difference of $I(p)$ and $I'(p_d)$ is used. α balances the color and gradient terms. τ_1, τ_2 are the truncation values to reduce the influence of occluded pixels. $C(p,d)$ actually expresses how well the pixel p in the reference image I matches the pixel p_d in the target image I'.

Now the $|\mathcal{D}|$ slices of the cost volume C are filtered respectively. To be more precise, for the d^{th} xy-slice, the output of the filtering at pixel p at disparity d is a weighted average of all pixels in a support window w_p of p in the d^{th} xy-slice, which can be formulated as:

$$C^A(p,d) = \sum_{q\in W_p} K(p,q)C(q,d) \tag{3}$$

where $K(\cdot)$ is a weighting function (also known as kernel function [37]) and the support window w_p is closely related to the weighting function. $K(p,q)$ measures the support weight between pixels p and q. Then all the weights can be calculated by performing the weighting function $K(\cdot)$ on the whole reference image.

Once all the slices of the cost volume are filtered, the final aggregated cost volume C^A is obtained and the disparity at pixel p denoted by d_p is simply chosen with the WTA strategy as:

$$d_p = arg\ \min_{d\in\mathcal{D}} C^A(p,d) \tag{4}$$

Then the final disparity map D can be obtained by mapping each pixel to the optimal disparity level following the above steps.

4 Cost aggregation on oriented linear trees

In this section, a novel oriented linear tree structure is firstly proposed. A linear time cost aggregation with two-passes traversals technique on single 1D path is presented afterwards. The computational complexity is discussed finally.

4.1 The oriented linear tree structure

In our method, the reference image I is represented as a connected (not 4-connected) and undirected graph $G = (V,E)$ where each node in V corresponds to a pixel in I, and each edge in

E connects a pair of neighboring pixels. For an edge e connecting pixels u and v, its weight is decided as follows:

$$\omega_e = \omega(u, v) = \frac{1}{M} \|I(u) - I(v)\|_1 \tag{5}$$

Here $I(u)$ and $I(v)$ represent the color vector of pixels u and v respectively, as well as M denotes color channels of the reference image. When comparing two neighboring pixels in the same image to decide whether they can support each other, as the same as other ASW methods, it is assumed that the neighboring pixels which have similar color are likely to have similar disparity. In order to ensure that the similarity between two neighboring pixels is linear with color difference like in Eq. (2), the average absolute difference of their color vectors is used as their edge weight. Experiments also verify that using $L1$ norm of the vector as the similarity measurement in Eq. (5) performs better than using other norms such as $L2$ norm.

For the local filtering-based methods, a pixel gets supports only from the other pixels inside its local window. For handling the poor texture regions, a window should thereby be large enough to capture sufficient intensity variation. Unfortunately, some filters such as bilateral filter have high computational complexity when the kernel size is large (e.g. 35×35). It is impossible for different data sets to find an optimal window that shows good performance both in the result quality and in computational speed. Thus, the non-local methods [3, 36, 50] try to use *tree filtering* [51] to effectively and efficiently perform the cost aggregation, where each pixel receives supports from all the other pixels of the same image through unique paths on a tree. However, during the process of constructing some special tree structure, some useful connectivity constraints between adjacent pixels may be lost and the original Markov random field embedded in the reference image may be destroyed. This results in relatively poor performance in ambiguity discrimination in highly textured regions. Hence, for each pixel, on the one hand, the inherent connectivity between it and adjacent pixels should be held like the local filtering-based methods; on the other hand, for handling the low texture regions, more other pixels in the whole reference image should be also used to support it.

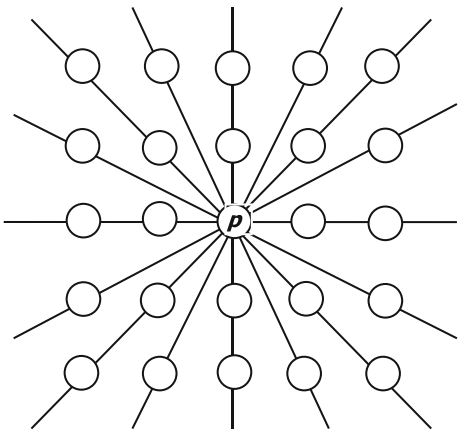


Fig. 3 An oriented linear tree rooted on pixel p with $L = 8$ 1D paths. The pixel p receives the other image pixels' contributions along eights 1D paths from different directions. The oriented linear tree naturally contains a local window centered on pixel p for reducing the ambiguity in highly textured regions and depth boundaries. The other pixels far away from p on all the 1D paths can be used for handling low textured areas

According to the above observations, a new idea that each pixel receives all the other pixels' supports along 1D paths from all directions is proposed. In Fig. 3, paths from various directions intersect at the central pixel p, which can be treated as a tree rooted on node p. We refer to this tree as the *Oriented Linear Tree*. Here eight symmetric directions in the tree are showed. We can see that pixel p not only gets the supports from adjacent pixels within a local support window (e.g. 5×5 pixels) that is naturally formed, but the other pixels outside the local window in the reference image also contribute to pixel p along various 1D paths. Let pixels q and p be two nodes on a 1D path and an orderly sequence of nodes $\langle q = q_1, \ q_2, \ \cdots, \ q_n = p \rangle$ be gone through successively when q reaches p along this path. The geodesic distance $S(p, q)$ between p and q can be directly defined as the sum of the edge weights along their path as follows:

$$S(p,q) = S(q,p) = \sum_{i=1}^{n-1} \omega(q_i, q_{i+1}) \tag{6}$$

Note that the spatial proximity is employed since the distance is accumulated along the 1D path. The support weight $K(p, q)$ between p and q is defined as follows:

$$K(p,q) = K(q,p) = \exp\left(-\frac{S(p,q)}{\sigma}\right) \tag{7}$$

where σ is a user-defined parameter used to adjust the distance between two nodes. The pixel with shorter distance from p is assigned a larger weight.

It follows that every pixels in the reference image has an oriented linear tree similar to the pixel p. The number of 1D paths must be at least four, that is, the horizontal, vertical and two diagonal paths must be included in the oriented linear tree. Actually, the oriented linear tree with the above four 1D paths can bring desirable results. More 1D paths used will produce more accurate results but also increase the runtime complexity. In this paper, for trading off accuracy against speed, we use eight 1D paths. In addition to the above four paths, the other four additional paths respectively correspond to the directional angles $actan\left(\pm 1/2\right)$ and $actan(\pm 2)$.

4.2 Cost aggregation on an oriented linear tree

Since all the paths in the tree are separated from each other, of course we separately aggregate the matching costs along different paths. Let the oriented linear tree rooted on pixel p contain L(e.g. $L = 8$) 1D paths and $\mathcal{P}_{\mathrm{r}}$ denote the 1D path from the direction r. Let $C_r^A(p, d)$ denote the aggregated cost along the path $\mathcal{P}_{\mathrm{r}}$ for pixel p and disparity d. According to Eq. (3), $C_r^A(p, d)$ can be defined as follows:

$$C_r^A(p,d) = \sum_{q \in \mathcal{P}_{\mathrm{r}}} K(p,q) C(q,d) \tag{8}$$

Accordingly, the final aggregated cost $C^A(p, d)$ for the root p and disparity d can be calculated by summing the aggregated costs $C_r^A(p, d)$ of all 1D paths. Note that the matching cost $C(p, d)$ of the root p at disparity d is added L times in the sum, so the repeat addition of $C(p, d)$ should

be subtracted from the sum. Hence, the final aggregated cost $C^A(p,d)$ of the root p at disparity d is given as follows:

$$C^A(p,d) = \sum_{r=1}^{L} C_r^A(p,d) - (L-1)C(p,d) \tag{9}$$

Lastly, the final disparity result can be obtained from Eq. (4). As far as image filtering is concerned, the weights $K(p,q)$ should be normalized. It is noteworthy that all the weights $K(p,q)$ in the reference image remain unchanged for different disparity levels and then all the disparity levels share a normalization constant $\sum_{q \in I} K(p,q)$. Therefore, the normalization step is usually not required in the cost aggregation since the disparity with minimum aggregated cost for each pixel is simply chosen using the WTA strategy as expressed in Eq. (4).

4.3 Cost aggregation on 1D path

Obviously, if we directly compute the aggregated cost for each root node of so many trees according to Eqs. (8) & (9), this can easily become computationally intractable. Benefiting from the linearity of each 1D path in the oriented trees as well as the geodesic distance used between two nodes, by using two-passes traversals on each path $\mathcal{P}_r$ from one end to the other end, we can calculate the aggregated cost $C_r^A(p,d)$ at disparity d for each node p lying on the path $\mathcal{P}_r$ at the same time. This leads to the linear computational complexity of our aggregation algorithm.

Here we show that how the values of $C_r^A(p,d)$ at disparity d for each node p on the path $\mathcal{P}_r$ are determined with two-passes traversals. Since the cost aggregation problem for all nodes on $\mathcal{P}_r$ is studied under the same disparity level, here the disparity d can be treated as a constant for ease of description. In Fig. 4, let s and t be the two endpoints of 1D path $\mathcal{P}_r$ and they must lie on the image borders. In addition, for any node p on the path $\mathcal{P}_r$ that isn't the endpoint, let p-1 and $p+1$ on the path $\mathcal{P}_r$ be the two neighboring nodes of node p. Let $F_r^A(p,d)$ denote an intermediate aggregated cost value of p

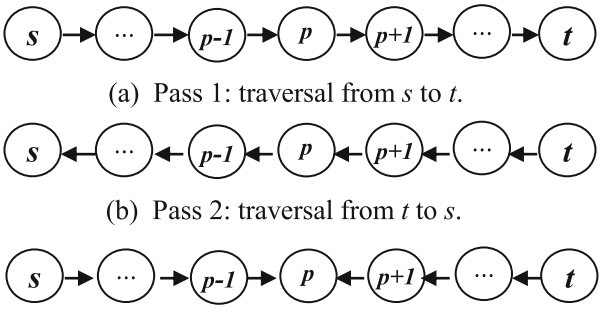


Fig. 4 The two-passes traversals on the path $\mathcal{P}_r$ from one end to the other end. **a** In pass 1, $F_r^A(\cdot,d)$ of each node on $\mathcal{P}_r$ is computed by traversing $\mathcal{P}_r$ from s to t. **b** In pass 2, $B_r^A(\cdot,d)$ of each node on $\mathcal{P}_r$ is computed by traversing $\mathcal{P}_r$ from t to s. **c** The aggregated cost $C_r^A(p,d)$ for each node p lying on the path $\mathcal{P}_r$ is derived by combining the results from the above two passes

supported by all nodes from s to p. According to Eqs. (7) & (8), the recursive formula of $F_r^A(p,d)$ can be deduced as follows:

$$\begin{aligned} F_r^A(p,d) &= \sum_{q\in[s,p]} K(p,q)C(q,d) \\ &= C(p,d) + \sum_{q\in[s,p-1]} K(p,q)C(q,d) \\ &= C(p,d) + \sum_{q\in[s,p-1]} K(p,p-1)K(p-1,q)C(q,d) \\ &= C(p,d) + K(p,p-1)\sum_{q\in[s,p-1]} K(p-1,q)C(q,d) \\ &= C(p,d) + K(p,p-1)F_r^A(p-1,d) \end{aligned} \tag{10}$$

where $[s,p]$ represents an orderly sequence of nodes on $\mathcal{P}_r$ from s to p. Therefore, in the first pass as shown in Fig. 4a, the cost aggregation is performed by traversing $\mathcal{P}_r$ from s to t and the value $F_r^A(p,d)$ of each node on the path $\mathcal{P}_r$ can be calculated in turn since the value $F_r^A(s,d)$ for the starting node s is known, i.e. $F_r^A(s,d) = C(s,d)$.

In the second pass, as can be seen from Fig. 4b, the path $\mathcal{P}_r$ is traced from node t to node s where the traversal direction is just opposite to the first pass. Here let $B_r^A(p,d)$ denote another intermediate aggregated cost value of p supported by all nodes from t to p. Then $B_r^A(p,d)$ can be written recursively in a similar way to $F_r^A(p,d)$ as,

$$\begin{aligned} B_r^A(p,d) &= \sum_{q\in[t,p]} K(p,q)C(q,d) \\ &= C(p,d) + K(p,p+1)B_r^A(p+1,d) \end{aligned} \tag{11}$$

Hence, in the second pass, from the starting node t with $B_r^A(t,d) = C(t,d)$, the value $B_r^A(p,d)$ of each node on the path $\mathcal{P}_r$ can be recursively calculated until the end s is reached.

It follows that we can rewrite $C_r^A(p,d)$ from Eqs. (8), (10) & (11) as follows:

$$\begin{aligned} C_r^A(p,d) &= \sum_{q\in\mathcal{P}_r} K(p,q)C(q,d) \\ &= \sum_{q\in[s,p]} K(p,q)C(q,d) + \sum_{q\in[t,p]} K(p,q)C(q,d) - C(p,d) \\ &= F_r^A(p,d) + B_r^A(p,d) - C(p,d) \end{aligned} \tag{12}$$

Figure 4c illustrates the above calculation process. Note that the above two passes traversals can be implemented separately and independently. After accomplishing the two-passes traversals on the path $\mathcal{P}_r$, from Eq. (12) we can obtain the aggregated cost $C_r^A(p,d)$ at disparity d for each node p lying on the path $\mathcal{P}_r$ at the same time.

Once the aggregated cost $C_r^A(p,d)$ along each 1D path $\mathcal{P}_r$ for the root p is calculated by using the two-passes traversals, the final aggregated cost $C^A(p,d)$ of pixel p at disparity d is easily obtained according to Eq. (9).

4.4 Computational complexity

For each pixel p and each disparity level $d\in\mathcal{D}$, computing $C_r^A(p,d)$ by traversing single 1D path $\mathcal{P}_r$ back and forth needs to visit the pixel p twice. Because the total number of disparity levels is $|\mathcal{D}|$ and an oriented tree only contains eight 1D paths in this paper, the overall computational complexity of our oriented-linear-trees based cost aggregation

algorithm is $O(|\mathcal{D}|WH)$, where W, H are the width and height of the reference image. In addition, since the proposed oriented linear trees don't need to be additionally constructed and only include a few separable and linear 1D paths, the non-local cost aggregation based on oriented linear trees is very suitable for parallel calculations.

5 Experimental results

In this section, we use Middlebury [40] and KITTI [8] datasets in order to validate the effectiveness of the proposed method denoted as OLT. To evaluate the performance of our method, we compare it with four state-of-the-art cost aggregation methods: aggregation with guided image filter (denoted as GF) [18], aggregation with minimum spanning tree (denoted as MST) [50, 51], aggregation with segment tree (denoted as ST) [36], aggregation with cross-trees and edge prior (denoted as Cross-E) [3]. For ST, we directly use the C++ code provided by the authors,[1] and all the parameter setting are identical to those used in their implementations. For GF, we implemented our own C++ code by referring to the author-provided Matlab program[2] in order to process high-resolution images from KITTI dataset efficiently. In the same way, we implement our algorithm and re-implement MST and Cross-E by using C++ language. For MST and Cross-E, their parameters follow the settings of the corresponding papers.

5.1 Experimental settings

We mainly focus on two benchmarks: the Middlebury benchmark [40] from the indoor scene, and the KITTI benchmark [8] from the outdoor scene. The Middlebury benchmark is a de facto standard for comparing existing stereo matching algorithms. To date, more than 150 methods have been evaluated on the Middlebury benchmark, using four standard pairs of stereo images, (i.e., Tsukuba, Venus, Teddy, and Cones). In order to more comprehensively evaluate the performance of the proposed method, our experiments are conducted not only on the four standard pairs, but also on the other 23 pairs of images (i.e., Middlebury 2005 (6 image pairs) and Middlebury 2006 (17 image pairs) data sets). The Middlebury benchmark provides corresponding ground truth disparity maps of these 27 stereo image pairs. Note that these Middlebury image pairs from various complex indoor scenes are challenging due to low texture, repetitive texture and complex object structures. In addition, we also carry out the experiments on the KITTI benchmark to further verify the adaptability and robustness of our method. The KITTI dataset contains 195 test image pairs and 194 training image pairs for evaluating stereo matching algorithms. These image pairs from various real complex road scenes are taken by a pair of high-resolution cameras equipping on an autonomous driving platform. All the KITTI images are captured under the real-world illumination condition, and then most image pairs contain large textureless regions, e.g., sky, walls and cars, and illumination change, e.g., shades and light reflection. Hence, the KITTI benchmark is more challenging than the Middlebury benchmark. During our experiment on the KITTI benchmark, we use the whole 194 training image pairs with ground truth disparity maps available to evaluate our method.

[1] http://xing-mei.net/resource/page/segment-tree.html
[2] https://www.ims.tuwien.ac.at/publications/tuw-202088

As described in the Section 1, cost aggregation as a key step of stereo matching can only be implemented after the matching cost calculation is completed. Therefore, we need to choose an appropriate matching cost calculation method to construct the initial cost volume for cost aggregation. The above four cost aggregation methods (i.e., GF [18], MST [50, 51], ST [36] and Cross-E [3]) quantitatively evaluated their performance on the above 27 Middlebury image pairs and used the *AD-Gradient* cost function in Eq. (2) proposed in [18] to calculate the initial cost volume. As the same with them, for the Middlebury dataset we also adopt this cost function to compute the initial cost volume and set the parameters of the *AD-Gradient* cost function to be the same values as these four methods for fair comparison, i.e., $\{\alpha, \tau_1, \tau_2\} = \{0.1, 7/255, 2/255\}$. For the KITTI dataset, considering that most image pairs have large illumination variation, we adopt the *AD-Census* cost function proposed in [35] to compute the initial cost volume for all the five cost aggregation methods since Census Transform [54] is proven to be robust against illumination changes. Then the related parameters of the *AD-Census* cost function follow the settings in [35]. In fact, the proposed cost aggregation method only has a single parameter, namely, the parameter σ of our weighting function in Eq. (7). The parameter σ is set to 0.06 and remains constant for all data sets. We also investigate the performance of the proposed cost aggregation method with respect to the parameter σ in the following experiment part.

5.2 Evaluation on Middlebury dataset

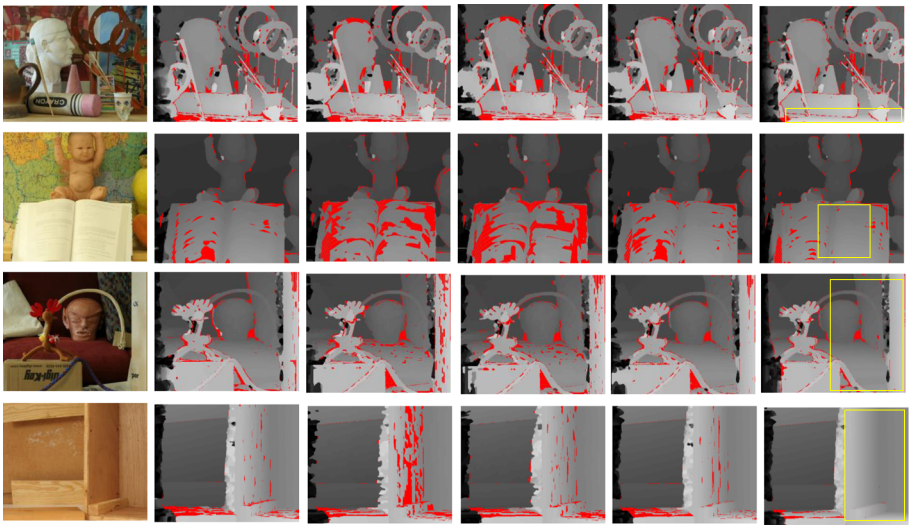
We first evaluate the proposed method on 27 stereo image pairs from the Middlebury benchmark [40]. Firstly, the initial cost volume is calculated by applying the *AD-Gradient* cost function defined in [18] and then it is used as a common input to all the five cost aggregation methods. Next, we respectively perform each cost aggregation algorithm on the initial cost volume and then establish the corresponding initial disparity map with the WTA strategy. In order to compare different cost aggregation methods more reasonably, here no post-processing technique for refining the initial disparity maps is employed. Note that all the ground truths of disparity maps for these 27 image pairs are provided by the benchmark. Once the initial disparity map is estimated, the disparity error of a pixel is the absolute difference between the estimated disparity and the ground truth, and the disparity error of a region is the mean error of all pixels in it. In the same way as other state-of-the-art methods [3, 1819, 36, 39, 44, 50, 51], the aggregation accuracy is evaluated by the error rate and the disparity error in non-occluded areas, where the error rate is the percentage of bad pixels having disparity error larger than 1 pixel.

The quantitative evaluation results are presented in Table 1, where the best result among the five algorithms is marked in bold for each stereo instance. For visual comparison, the disparity maps established by using the five algorithms are shown in Fig. 5. Firstly, as can be seen in Table 1, both the average error rate and the average disparity error of the proposed method are lowest among all the five methods. Moreover, our average rankings in both the error rate and the disparity error are also highest, and our method outperforms the other four methods on most stereo instances. We can find that the number of the best results for the proposed method, no matter whether in the error rate or in the disparity error, is more than the sum of the other four methods. The average error rate and average disparity error of the proposed method respectively decrease by 1.38% and 0.26 px comparing with GF. Although GF has been proven to be the top performer among all local methods, it still establishes locally optimized results within user-specified local support region. It also shows that MST and ST are less

Table 1 Quantitative evaluation of the five cost aggregation algorithms on the Middlebury dataset in terms of the cost aggregation accuracy and the rank displayed with subscript, where the best results are shown in bold

Data	Error rate					Disparity error				
	GF	MST	ST	Cross-E	OLT	GF	MST	ST	Cross-E	OLT
Aloe	4.95% $_5$	4.40% $_3$	4.40% $_3$	3.62% $_2$	**2.91%** $_1$	0.79 px $_4$	0.76 px $_3$	**0.75 px** $_1$	0.80 px $_5$	**0.75 px** $_1$
Art	9.05% $_3$	10.83% $_5$	10.52% $_4$	8.75% $_2$	**6.91%** $_1$	1.47 px $_3$	1.62 px $_5$	1.60 px $_4$	1.45 px $_2$	**1.22 px** $_1$
Baby1	**3.70%** $_1$	8.29% $_5$	4.52% $_3$	4.58% $_4$	4.03% $_2$	**0.75 px** $_1$	1.03 px $_5$	0.82 px $_2$	0.93 px $_4$	0.85 px $_3$
Baby2	**4.52%** $_1$	12.69% $_4$	15.20% $_5$	6.50% $_3$	4.86% $_2$	**0.69 px** $_1$	0.85 px $_4$	0.97 px $_5$	0.74 px $_3$	0.71 px $_2$
Baby3	4.94% $_2$	5.71% $_5$	5.07% $_3$	5.20% $_4$	**4.89%** $_1$	1.33 px $_2$	1.41 px $_3$	1.42 px $_4$	1.44 px $_5$	**1.32 px** $_1$
Books	8.22% $_3$	9.04% $_4$	9.41% $_5$	**7.78%** $_1$	8.07% $_2$	**1.08 px** $_1$	1.16 px $_5$	1.12 px $_3$	1.12 px $_3$	**1.08 px** $_1$
Bowling1	15.14% $_2$	18.15% $_4$	18.88% $_5$	15.40% $_3$	**14.52%** $_1$	3.98 px $_5$	**2.04 px** $_1$	2.31 px $_3$	2.72 px $_4$	**2.04 px** $_1$
Bowling2	6.49% $_2$	10.21% $_4$	11.13% $_5$	8.78% $_3$	**6.11%** $_1$	0.72 px $_2$	0.77 px $_3$	0.87 px $_5$	0.79 px $_4$	**0.68 px** $_1$
Cloth1	0.66% $_5$	0.43% $_4$	0.41% $_3$	0.15% $_2$	**0.11%** $_1$	0.41 px $_3$	0.41 px $_3$	0.41 px $_3$	**0.39 px** $_1$	**0.39 px** $_1$
Cloth2	2.57% $_3$	3.35% $_4$	3.36% $_5$	1.76% $_2$	**1.30%** $_1$	0.53 px $_2$	0.62 px $_4$	0.63 px $_5$	0.54 px $_3$	**0.52 px** $_1$
Cloth3	1.76% $_5$	1.73% $_4$	1.57% $_3$	1.12% $_2$	**0.93%** $_1$	0.46 px $_5$	0.45 px $_3$	0.45 px $_3$	0.44 px $_2$	**0.43 px** $_1$
Cloth4	1.16% $_3$	1.24% $_4$	1.27% $_5$	0.76% $_2$	**0.74%** $_1$	0.46 px $_3$	0.47 px $_4$	0.47 px $_4$	0.44 px $_2$	**0.43 px** $_1$
Dolls	4.66% $_3$	6.00% $_5$	5.17% $_4$	4.50% $_2$	**3.79%** $_1$	0.66 px $_2$	0.67 px $_4$	0.66 px $_2$	0.67 px $_4$	**0.61 px** $_1$
Flowerpots	13.15% $_2$	17.68% $_5$	14.91% $_4$	14.31% $_3$	**11.35%** $_1$	1.07 px $_2$	1.12 px $_5$	1.09 px $_4$	1.07 px $_2$	**1.05 px** $_1$
Lampshade1	13.86% $_5$	11.94% $_4$	10.31% $_2$	10.36% $_3$	**8.27%** $_1$	2.15 px $_5$	1.49 px $_2$	1.62 px $_4$	1.59 px $_3$	**1.36 px** $_1$
Lampshade2	22.38% $_5$	15.53% $_3$	21.65% $_4$	14.70% $_2$	**12.20%** $_1$	4.76 px $_5$	**1.88 px** $_1$	3.74 px $_4$	2.96 px $_3$	2.39 px $_2$
Laundry	15.07% $_5$	12.38% $_2$	13.83% $_4$	13.59% $_3$	**11.98%** $_1$	1.85 px $_5$	**1.21 px** $_1$	1.32 px $_3$	1.49 px $_4$	1.29 px $_2$
Moebius	8.85% $_5$	8.41% $_4$	8.19% $_2$	**8.11%** $_1$	8.25% $_3$	1.06 px $_5$	0.85 px $_2$	**0.84 px** $_1$	0.97 px $_4$	0.87 px $_3$
Reindeer	5.96% $_3$	8.47% $_5$	7.73% $_4$	5.35% $_2$	**4.11%** $_1$	1.00 px $_3$	1.12 px $_5$	1.09 px $_4$	0.97 px $_2$	**0.89 px** $_1$
Rocks1	2.18% $_3$	2.37% $_5$	2.34% $_4$	1.52% $_2$	**1.27%** $_1$	0.59 px $_4$	0.58 px $_3$	0.59 px $_4$	0.57 px $_2$	**0.56 px** $_1$
Rocks2	1.43% $_3$	1.86% $_5$	1.61% $_4$	1.24% $_2$	**1.02%** $_1$	0.48 px $_4$	0.48 px $_4$	0.47 px $_3$	**0.46 px** $_1$	**0.46 px** $_1$
Wood1	4.02% $_3$	9.86% $_5$	5.08% $_4$	3.80% $_2$	**1.17%** $_1$	0.90 px $_3$	1.00 px $_5$	0.91 px $_4$	0.84 px $_2$	**0.60 px** $_1$
Wood2	1.22% $_4$	1.08% $_3$	3.06% $_5$	0.75% $_2$	**0.69%** $_1$	0.87 px $_4$	**0.53 px** $_1$	1.21 px $_5$	0.61 px $_3$	0.58 px $_2$
tsukuba	2.28% $_5$	**1.67%** $_1$	1.85% $_2$	2.24% $_4$	2.06% $_3$	0.20 px $_3$	**0.17 px** $_1$	0.19 px $_2$	0.23 px $_5$	0.20 px $_3$
venus	0.94% $_5$	0.65% $_4$	0.64% $_3$	0.58% $_2$	**0.54%** $_1$	0.30 px $_2$	0.31 px $_5$	0.30 px $_2$	0.30 px $_2$	**0.29 px** $_1$
Teddy	8.35% $_5$	**7.30%** $_1$	7.67% $_3$	7.51% $_2$	7.69% $_4$	**0.82 px** $_1$	0.87 px $_2$	0.92 px $_5$	0.89 px $_3$	0.91 px $_4$
Cones	**2.89%** $_1$	3.63% $_5$	3.55% $_4$	3.50% $_3$	3.42% $_2$	**0.47 px** $_1$	0.55 px $_5$	0.53 px $_2$	0.53 px $_2$	0.53 px $_2$
Avg. error	6.31%	7.22%	7.16%	5.79%	**4.93%**	1.11 px	0.91 px	1.01 px	0.96 px	**0.85 px**
Avg. rank	3.41	3.96	3.78	2.41	**1.41**	3.00	3.30	3.37	2.96	**1.52**

The average error and the average rank are given in the bottom



(a) The reference image (b) GF (c) MST (d) ST (e) Cross-E (f) OLT

Fig. 5 The initial disparity maps without post-processing on Middlebury dataset. The 1st row: *Art*; the 2nd row: *Baby* 2; the 3rd row: *Reindeer*; the 4th row: *Wood1*. The bad pixels in the disparity maps are marked red. **a** The reference images. **b** Disparity maps computed with GF. **c** Disparity maps computed with MST. **d** Disparity maps computed with ST. **e** Disparity maps computed with Cross-E. **f** Disparity maps computed with the proposed method (i.e., OLT). See the regions of **f** within the yellow boxes, where the proposed method has fewer bad pixels than the other methods

accurate than GF in the evaluation. It is because that MST and ST remove some 8-connected siblings of most pixels, which results in the low discrimination for matching ambiguity especially in highly-textured regions as shown in Fig. 5. Besides, our algorithm achieves better performance than Cross-E. Our average error rate and average disparity error are respectively 0.86% and 0.11 px lower than Cross-E. Note that the latter needs edge prior and its results are also easily influenced by the accuracy of the extracted edges. The visual

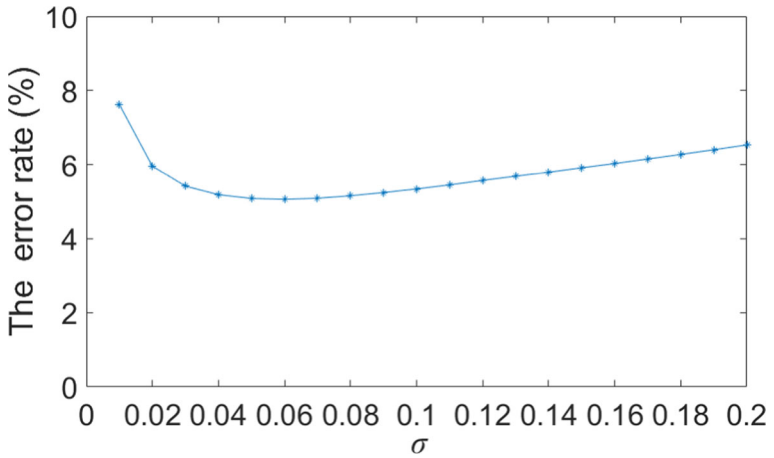
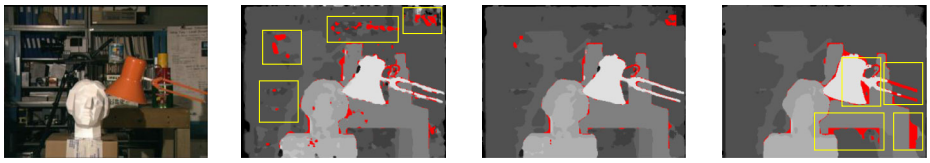


Fig. 6 Performance of the proposed cost aggregation method with respect to the parameter σ



(a) The reference image (b) $\sigma = 0.01$, **4.59%** (c) $\sigma = 0.06$, **2.06%** (d) $\sigma = 0.2$, **4.23%**

Fig. 7 The initial disparity maps of *tsukuba* image pair under three different σ values. The bad pixels in the disparity maps are marked red. The bold numbers under the disparity maps are the error rate. The reference image is in (**a**). In (**b**), the disparity map under $\sigma = 0.01$ has a large number of noise and bad pixels especially in low texture regions shown in the yellow boxes. For the disparity map when $\sigma = 0.2$ in (**d**), the bad pixels mainly occur at the boundaries of foreground objects and the thin lamp poles are partly erased as shown in the yellow boxes. In contrast, the disparity map under $\sigma = 0.06$ in (**c**) contains fewer bad pixels

comparison in Fig. 5 shows that the proposed method not only produces less bad pixels in highly textured regions, but also performs better in large planar surfaces with low texture.

The single parameter used in our cost aggregation method is the parameter σ of the weighting function in Eq. (7) which is used to adjust the support weight between two pixels. It is necessary to study the performance of the proposed cost aggregation method with respect to parameter σ. Firstly we still use the *AD-Gradient* cost function to compute the initial cost volume, and keep all the parameter settings of this cost function constant. Then we vary the parameter σ from 0.01 to 0.2. The error rate of all the 27 Middlebury image pairs in non-occluded areas is evaluated in this test. Figure 6 shows the test results with different σ. It can be seen from Fig. 6 that the variance of the error rate is very low when σ ranges from 0.02 to 0.18, and the error rate is minimum when $\sigma = 0.06$. The result demonstrates that our cost aggregation method is insensitive to the change of the parameter σ. It is worth noting that our weighting function in Eq. (7) actually adopts the Gauss kernel. Thus, the support weight between any two pixels is positively and non-linearly related to the smoothing parameter σ. If the value of the parameter σ is too small (e.g. $\sigma < 0.02$), the support weights which a pixel gets from the other pixels are also relatively small. This will degrade the performance of cost aggregation. In particular, as illustrated in Fig. 7b, the matching ambiguities in low texture or repeat texture regions can not be effectively reduced because a pixel within these regions only receives very little supports from the other pixels from those regions with texture variation. On the other hand, if the value of the parameter σ is too large (e.g. $\sigma > 0.18$), the support weights which a pixel gets from the other pixels become larger. This may bring numerous disparity errors at the boundaries of objects if the color difference between the background and the foreground is not large enough as shown in Fig. 7d.

For complexity comparison, Table 2 reports the average running time of the five algorithms on the four standard Middlebury data sets consisting of Tsukuba (384×288 pixels, 16 disparity levels), Venus (434×383 pixels, 20 disparity levels), Teddy (450×375 pixels, 60 disparity levels) and Cones (450×375 pixels, 60 disparity levels). We test the C++ programs

Table 2 The complexity comparison among the five algorithms

	GF	MST	ST	Cross-E	OLT
Avg. running time (s)	14.46	4.73	5.47	4.92	4.85

For each algorithm, this table reports the average running time on the four standard Middlebury data sets

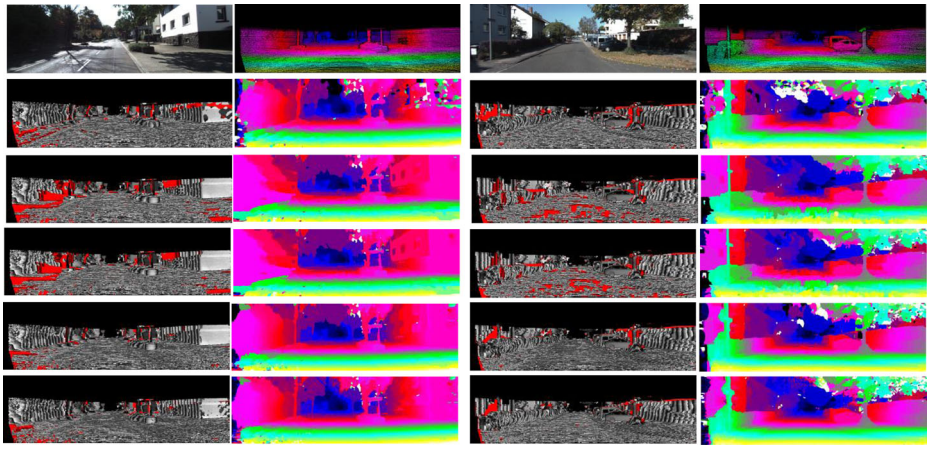
Table 3 Quantitative evaluation of the five cost aggregation algorithms on the KITTI training dataset

Methods	Avg. disparity Error		> 2 pixels		> 3 pixels		> 4 pixels		> 5 pixels	
	Avg-Noc	Avg-All	Out-Noc	Out-All	Out-Noc	Out-All	Out-Noc	Out-All	Out-Noc	Out-All
GF	3.20 px	4.25 px	15.54%	17.15%	10.87%	12.55%	8.89%	10.60%	7.81%	9.52%
MST	3.47 px	4.15 px	31.21%	32.28%	23.27%	24.41%	18.54%	19.69%	15.36%	16.50%
ST	3.38 px	4.10 px	29.24%	30.37%	21.60%	22.79%	17.13%	18.34%	14.16%	15.36%
Cross-E	2.36 px	3.28 px	13.98%	15.46%	9.67%	11.17%	7.69%	9.18%	6.51%	7.99%
OLT	**2.25 px**	**3.22 px**	**11.82%**	**13.57%**	**8.98%**	**10.77%**	**7.15%**	**8.76%**	**6.27%**	**7.64%**

Performance is measured in terms of the average disparity error, as well as the error rate with different thresholds. The best results among the five algorithms are shown in bold

Out-Noc the error rate in non-occluded areas, *Out-All* the error rate in total, *Avg-Noc* the average disparity error in non-occluded areas, *Avg-All* the average disparity error in total.

of the five algorithms on a PC platform with a 3.20GHz Intel Core (i5-6500) CPU and 8GB Memory. Our method has the similar efficiency with MST and Cross-E and it is faster than GF and ST. Although the complexity of GF is independent of the kernel size, if color guide image is used, it still requires about 18 box filtering to finish the guided filtering process. ST needs to build a specific tree structure like MST and Cross-E; additionally, it has to implement image segmentation, which makes ST slow. By contrast, the proposed method does not need to additionally create any tree structure or carry out image segmentation beforehand.



(a) Frame: #000000_10 (b) Frame: #000008_10

Fig. 8 The initial disparity maps without post-processing on the KITTI training dataset. The 1st row: two reference images (frames: #000000_10, #000008_10) and the ground truths of their disparity maps which are displayed in false color. The 2nd row: the disparity results of GF. The 3rd row: the disparity results of MST. The 4th row: the disparity results of ST. The 5th row: the disparity results of Cross-E. The 6th row: the disparity results of the proposed method. For each method, the estimated disparity maps are displayed in false color, and the corresponding error maps in non-occluded areas from ≤1 pixel error in black to ≥ 5 pixels error in white are shown here. In the error maps, the bad pixels with disparity error more than default threshold 3 pixels are marked red

5.3 Evaluation on KITTI dataset

Previous cost aggregation algorithms only evaluate their methods on the Middlebury benchmark. However, in our experiment, we also evaluate the proposed method on the KITTI benchmark [8] to further prove the feasibility of our method. During our experiment, we use the whole 194 training image pairs with the ground truths of disparity maps available. Due to the illumination variation on the KITTI dataset, we adopt the *AD-Census* cost function proposed in [35] to compute the matching cost because *Census Transform* [54] is robust against illumination changes as described in [11, 16, 35]. In the same way with the Middlebury benchmark, the obtained initial cost volume is used as a common input to all the five cost aggregation methods. According to the evaluation metric provided by the KITTI Benchmark [8], here the performance in aggregation accuracy is measured in terms of the average disparity error, as well as the error rate with different thresholds. Compared to the Middlebury Benchmark, for each stereo pair in the KITTI Benchmark, there are two different regions needed to be evaluated, that is, the whole reference image region denoted as "All" and the non-occluded region denoted as "Noc". Accordingly, each region has a ground truth of disparity map available.

Table 3 reports the quantitative evaluation of the five cost aggregation algorithms on the KITTI dataset. Here the disparity results are also computed only with the WTA strategy and no post-processing is performed. Figure 8 illustrates the visual comparisons, where the false color disparity maps are displayed by using the KITTI development kit [8]. From Table 3, we can see that our proposed algorithm performs better than the other methods in terms of both average disparity error and average error rate. As can be seen in Fig. 8, there are a lot of poor textured regions and noise such as walls and roads in the images because they are captured in the real-world illumination condition. The local window-based cost aggregation methods such as GF cannot effectively handle these regions. The non-local cost aggregation methods such as MST and ST are expected to perform well in the low textured regions. Unfortunately their performance is worse than GF in these regions especially in roads as shown in Fig. 8b. The main reason may be that these methods are vulnerable to the noise since they omit the inherent spatial proximity constraints between local adjacent pixels. Besides, the implicit assumption for Cross-E method that the color edges are consistent with the depth boundaries may degrade the cost aggregation in highly textured or noise regions. It can be found in Fig. 8 that the disparity maps generated by our method are more smoothing and contain less bad pixels.

6 Conclusions

This paper describes a novel non-local cost aggregation based on oriented linear trees. This proposed cost aggregation method can effectively overcome the disadvantages of the traditional window-based aggregation methods and non-local tree-based aggregation methods presented recently. The quantitative evaluation on, no matter whether Middlebury benchmark of the indoor scene or KITTI benchmark of the outdoor scene, demonstrates that the proposed method not only outperforms state-of-the-art local cost aggregation methods, but also offers better performance than the other non-local aggregation methods. The experimental results also show that the proposed aggregation method not only produces accurate results, but also keeps linear computational complexity. In the future work, the proposed aggregation method will be accelerated with CUDA

implementations on GPU systems to achieve real-time performance. In addition, it is noteworthy that the result of the matching cost calculation has a direct impact on the accuracy of cost aggregation and the final output of the stereo matching. Therefore, outstanding and robust matching cost calculation methods adapted to various conditions will be considered and then integrated in our cost aggregation framework in order to further improve the performance of stereo matching.

Acknowledgements This work was supported by National Natural Science Foundation of China (No.61673318, No.61703301, No.61771386, No.61801005); by Research project of Hubei Provincial Department of Education (B2017080), China.

References

1. Bleyer M, Gelautz M (2005) A layered stereo matching algorithm using image segmentation and global visibility constraints[J]. ISPRS J Photogramm Remote Sens 59(3):128–150
2. Boykov Y, Veksler O, Zabih R (2001) Fast approximate energy minimization via graph cuts [J]. IEEE Trans Pattern Anal Mach Intell 23(11):1222–1239
3. Cheng F, Zhang H, Sun M et al (2015) Cross-trees, edge and superpixel priors-based cost aggregation for stereo matching [J]. Pattern Recogn 48(7):2269–2278
4. Cigla C, Alatan AA (2011) Efficient edge-preserving stereo matching[C]. In: Proceedings of the IEEE international conference on computer vision workshops. IEEE, pp 696–699
5. Comaniciu D, Meer P (2002) Mean shift: a robust approach toward feature space analysis [J]. IEEE Trans Pattern Anal Mach Intell 24(5):603–619
6. Fusiello A, Roberto V, Trucco E (2000) Symmetric stereo with multiple windowing[J]. Int J Pattern Recognit Artif Intell 14(08):1053–1066
7. Geiger A, Ziegler J, Stiller C (2011) Stereoscan: dense 3d reconstruction in real-time[C]. In: Proceedings of the IEEE intelligent vehicles symposium. IEEE, pp 963–968
8. Geiger A, Lenz P, Urtasun R (2012) The KITTI vision benchmark. [Online]. Available: http://www.cvlibs.net/datasets/kitti/eval_stereo_flow.php?benchmark=stereo. Accessed March 2018
9. Gerrits M, Bekaert P (2006) Local stereo matching with segmentation-based outlier rejection[C]. In: Proceedings of the 3rd Canadian conference on computer and robot vision. IEEE, pp 66–66
10. Guney F, Geiger A (2015) Displets: resolving stereo ambiguities using object knowledge[C]. In: Proceedings of the IEEE conference on computer vision and pattern recognition. IEEE, pp 4165–4175
11. Hafner D, Demetz O, Weickert J (2013) Why is the census transform good for robust optic flow computation?[C]. In: Proceedings of the international conference on scale space and variational methods in computer vision. Springer, pp 210–221
12. Hartley R, Zisserman A (2003) Multiple view geometry in computer vision[M]. Cambridge university press, Cambridge
13. He K, Sun J, Tang X (2013) Guided image filtering [J]. IEEE Trans Pattern Anal Mach Intell 35(6): 1397–1409
14. Hermann S, Klette R (2012) Iterative semi-global matching for robust driver assistance systems[C]. In: Proceedings of the Asian conference on computer vision. Springer, pp 465–478
15. Hirschmuller H (2008) Stereo processing by semiglobal matching and mutual information[J]. IEEE Trans Pattern Anal Mach Intell 30(2):328–341
16. Hirschmüller H, Scharstein D (2007) Evaluation of cost functions for stereo matching[C]. In: Proceedings of the IEEE conference on computer vision and pattern recognition. IEEE, pp 1–8
17. Hosni A, Bleyer M, Gelautz M et al (2009) Local stereo matching using geodesic support weights[C]. In: Proceedings of the 16th IEEE international conference on image processing. IEEE, pp 2093–2096
18. Hosni A, Rhemann C, Bleyer M et al (2013) Fast cost-volume filtering for visual correspondence and beyond[J]. IEEE Trans Pattern Anal Mach Intell 35(2):504–511

19. Hosni A, Bleyer M, Gelautz M (2013) Secrets of adaptive support weight techniques for local stereo matching [J]. Comput Vis Image Underst 117(6):620–632
20. Kanade T, Okutomi M (1994) A stereo matching algorithm with an adaptive window: theory and experiment [J]. IEEE Trans Pattern Anal Mach Intell 16(9):920–932
21. Kao CC (2017) Stereoscopic image generation with depth image based rendering[J]. Multimed Tools Appl 76(11):12981–12999
22. Kappes JH, Andres B, Hamprecht FA et al (2013) A comparative study of modern inference techniques for discrete energy minimization problems[C]. In: Proceedings of the IEEE conference on computer vision and pattern recognition. IEEE, pp 1328–1335
23. Kolmogorov V, Zabin R (2004) What energy functions can be minimized via graph cuts?[J]. IEEE Trans Pattern Anal Mach Intell 26(2):147–159
24. Lan X, Ma AJ, Yuen PC (2014) Multi-cue visual tracking using robust feature-level fusion based on joint sparse representation[C]. In: Proceedings of the IEEE conference on computer vision and pattern recognition. IEEE, pp 1194–1201
25. Lan X, Ma AJ, Yuen PC et al (2015) Joint sparse representation and robust feature-level fusion for multi-cue visual tracking[J]. IEEE Trans Image Process 24(12):5826–5841
26. Lan X, Zhang S, Yuen PC (2016) Robust joint discriminative feature learning for visual tracking[C]. In: Proceedings of the twenty-fifth international joint conference on artificial intelligence. AAAI, pp 3403–3410
27. Lan X, Yuen PC, Chellappa R (2017) Robust MIL-based feature template learning for object tracking[C]. In: Proceedings of the thirty-first AAAI conference on artificial intelligence. AAAI, pp 4118–4125
28. Lan X, Ye M, Zhang S et al (2018) Robust collaborative discriminative learning for RGB-infrared tracking[C]. In: Proceedings of the thirty-second AAAI conference on artificial intelligence, vol 7008. AAAI, p 7015
29. Lan X, Zhang S, Yuen PC et al (2018) Learning common and feature-specific patterns: a novel multiple-sparse-representation-based tracker[J]. IEEE Trans Image Process 27(4):2022–2037
30. Liu Y, Nie L, Han L et al (2015) Action2Activity: recognizing complex activities from sensor data[C]. In: Proceedings of the twenty-fourth international joint conference on artificial intelligence. AAAI, pp 1617–1623
31. Liu L, Cheng L, Liu Y et al (2016) Recognizing complex activities by a probabilistic interval-based model[C]. In: Proceedings of the thirtieth AAAI conference on artificial intelligence. AAAI, pp 1266–1272
32. Liu Y, Nie L, Liu L et al (2016) From action to activity: sensor-based activity recognition[J]. Neurocomputing 181:108–115
33. Liu Y, Zhang L, Nie L et al (2016) Fortune teller: predicting your career path[C]. In: Proceedings of the thirtieth AAAI conference on artificial intelligence. AAAI, pp 201–207
34. Mattoccia S, Giardino S, Gambini A (2009) Accurate and efficient cost aggregation strategy for stereo correspondence based on approximated joint bilateral filtering[C]. In: Proceedings of the Asian conference on computer vision. Springer, pp 371–382
35. Mei X, Sun X, Zhou M et al (2011) On building an accurate stereo matching system on graphics hardware[C]. In: Proceedings of the IEEE international conference on computer vision workshops. IEEE, pp 467–474
36. Mei X, Sun X, Dong W et al (2013) Segment-tree based cost aggregation for stereo matching[C]. In: Proceedings of the IEEE conference on computer vision and pattern recognition. IEEE, pp 313–320
37. Milanfar P (2013) A tour of modern image filtering: new insights and methods, both practical and theoretical[J]. IEEE Signal Process Mag 30(1):106–128
38. Richardt C, Orr D, Davies I et al (2010) Real-time spatiotemporal stereo matching using the dual-cross-bilateral grid[C]. In: Proceedings of the European conference on computer vision. Springer, pp 510–523
39. Scharstein D, Szeliski R (2002) A taxonomy and evaluation of dense two-frame stereo correspondence algorithms [J]. Int J Comput Vis 47(1):7–42
40. Scharstein D, Szeliski R (2002) Middlebury stereo vision website. [Online]. Available: http://vision.middlebury.edu/stereo/data. Accessed March 2018
41. Sengupta S, Greveson E, Shahrokni A et al (2013) Urban 3d semantic modelling using stereovision[C]. In: Proceedings of the IEEE international conference on robotics and automation. IEEE, pp 580–585
42. Sun J, Zheng NN, Shum HY (2003) Stereo matching using belief propagation [J]. IEEE Trans Pattern Anal Mach Intell 25(7):787–800
43. Tombari F, Mattoccia S, Stefano LD (2007) Segmentation-based adaptive support for accurate stereo correspondence [C]. In: Proceedings of the Pacific-rim symposium on image and video technology. Springer, pp 427–438

44. Tombari F, Mattoccia S, Stefano LD et al (2008) Classification and evaluation of cost aggregation methods for stereo correspondence [C]. In: Proceedings of the IEEE conference on computer vision and pattern recognition. IEEE, pp 1–8
45. Veksler O (2003) Fast variable window for stereo correspondence using integral images[C]. In: Proceedings of the IEEE conference on computer vision and pattern recognition. IEEE, pp 556–561
46. Veksler O (2005) Stereo correspondence by dynamic programming on a tree[C]. In: Proceedings of the IEEE conference on computer vision and pattern recognition, vol 2. IEEE, pp 384–390
47. Wang ZF, Zheng ZG (2008) A region based stereo matching algorithm using cooperative optimization[C]. In: Proceedings of the IEEE conference on computer vision and pattern recognition. IEEE, pp 1–8
48. Wu W, Zhu H, Zhang Q (2018) Epipolar rectification by singular value decomposition of essential matrix[J]. Multimed Tools Appl 77(12):15747–15771
49. Yamaguchi K, McAllester D, Urtasun R (2014) Efficient joint segmentation, occlusion labeling, stereo and flow estimation[C]. In: Proceedings of European conference on computer vision. Springer, pp 756–771
50. Yang Q (2012) A non-local cost aggregation method for stereo matching[C]. In: Proceedings of the IEEE conference on computer vision and pattern recognition. IEEE, pp 1402–1409
51. Yang Q (2015) Stereo matching using tree filtering [J]. IEEE Trans Pattern Anal Mach Intell 37(4):834–846
52. Yang Q, Wang L, Yang R et al (2009) Stereo matching with color-weighted correlation, hierarchical belief propagation, and occlusion handling [J]. IEEE Trans Pattern Anal Mach Intell 31(3):492–504
53. Yoon K, Kweon I (2006) Adaptive support-weight approach for correspondence search [J]. IEEE Trans Pattern Anal Mach Intell 28(4):650–656
54. Zabih R, Woodfill J (1994) Non-parametric local transforms for computing visual correspondence[C]. In: Proceedings of the European conference on computer vision. Springer, pp 151–158
55. Zhang K, Lu J, Lafruit G (2009) Cross-based local stereo matching using orthogonal integral images [J]. IEEE Trans Circuits Syst Video Technol 19(7):1073–1079
56. Zhang K, Fang Y, Min D et al (2014) Cross-scale cost aggregation for stereo matching[C]. In: Proceedings of the IEEE conference on computer vision and pattern recognition. IEEE, pp 1590–1597
57. Zhang C, Li Z, Cheng Y et al (2015) Meshstereo: a global stereo model with mesh alignment regularization for view interpolation[C]. In: Proceedings of the IEEE international conference on computer vision. IEEE, pp 2057–2065

Wenhuan Wu received his M.S. degree in computer application technology from Nanchang Hangkong University, Nanchang, China, in 2009. He is currently a Ph.D. candidate at School of Automation and Information Engineering, Xi'an University of Technology, Xi'an, China. And also he is a lecturer in Hubei University of Automotive Technology, Shiyan, China. His research interests include computer vision, pattern recognition and image processing.

E-mail: wuwenhuan5@163.com

Hong Zhu received her B.Sc. degree in 1984 from Xiamen University, received her M.Sc. degree in 1991 from Xi'an University of Technology, and received her Ph.D. degree in 1999 from University of FuKui, Japan. Now she is a professor and doctoral supervision in Xi'an University of Technology. Her main research interests include digital image processing, intelligent video surveillance and pattern recognition.

E-mail: zhuhong@xaut.edu.cn

Qian Zhang received her Ph.D. degree from Department of Computer Science in Kyungwon University of Korea, in 2010. She currently does her research in Shandong University as a post doctor. And also she is an associate professor of Taishan University, China. Her research interests include image processing and computer vision.

E-mail: aazhqg@hotmail.com