A Non-Local Cost Aggregation Method for Stereo Matching

Qingxiong Yang
City University of Hong Kong
http://www.cs.cityu.edu.hk/~qiyang/

Abstract

*Matching cost aggregation is one of the oldest and still popular methods for stereo correspondence. While effective and efficient, cost aggregation methods typically aggregate the matching cost by summing/averaging over a user-specified, local support region. This is obviously only locally-optimal, and the computational complexity of the full-kernel implementation usually depends on the region size. In this paper, the cost aggregation problem is re-examined and a non-local solution is proposed. The matching cost values are aggregated adaptively based on pixel similarity on a tree structure derived from the stereo image pair to preserve depth edges. The nodes of this tree are all the image pixels, and the edges are all the edges between the nearest neighboring pixels. The similarity between any two pixels is decided by their shortest distance on the tree. The proposed method is non-local as every node receives supports from all other nodes on the tree. As can be expected, the proposed non-local solution outperforms all local cost aggregation methods on the standard (Middlebury) benchmark. Besides, it has great advantage in extremely low computational complexity: only a total of **2 addition/subtraction** operations and **3 multiplication** operations are required for each pixel at each disparity level. It is very close to the complexity of unnormalized box filtering using integral image which requires **6 addition/subtraction** operations. Unnormalized box filter is the fastest local cost aggregation method but blurs across depth edges. The proposed method was tested on a MacBook Air laptop computer with a 1.8 GHz Intel Core i7 CPU and 4 GB memory. The average runtime on the Middlebury data sets is about **90 milliseconds**, and is only about $1.25\times$ slower than unnormalized box filter. A non-local disparity refinement method is also proposed based on the non-local cost aggregation method.*

1. Introduction

Stereo correspondence has traditionally been, and continues to be, one of the most extensively researched topics in computer vision. Stereo algorithms generally perform (subsets of) the following four steps:

1. matching cost computation;
2. cost (support) aggregation;
3. disparity computation/optimization; and
4. disparity refinement.

局部算法需要做隐式平滑假设通过聚合支持

Scharstein and Szeliski [21] developed a taxonomy and categorization scheme for stereo algorithms, and separated different stereo algorithms into two broad classes: local and global algorithms. In a local algorithm, the disparity computation at a given image pixel depends only on image intensity/color values within a window. All local algorithms require cost aggregation (step 2), and usually make implicit smoothness assumptions by aggregating support. Global algorithms, on the other hand, make explicit smoothness assumptions and then solve an optimization problem. Such algorithms typically omit the cost aggregation step, but rather seek a disparity solution (step 3) that minimizes a global cost function. Popular global methods include dynamic programming [2, 23], belief propagation [13, 14, 15, 16] and graph cuts [3]. Unlike local algorithms, a global algorithm estimates the disparity at one pixel using the disparity estimates at all the other pixels.

全局算法做显式平滑假设，然后解决一个优化问题

全局算法的核心

Cost aggregation methods are traditionally performed *locally* by summing/averaging matching cost over windows with constant disparity. The most efficient local cost aggregation method is unnormalized box filtering which runs in linear time (relative to the number of image pixels) using integral image [24] (also known as a summed area table [8]) but blurs across depth edges. Yoon and Kweon [6] demonstrated that edge-aware filters like bilateral filter [22] are very effective for preserving depth edges and Yang *et al.* [5] used bilateral filter for depth superresolution. However, full-kernel implementation of the bilateral filter is slow.

A number of approximation methods have been developed to accelerate the bilateral filter, including Paris and Durand's fast bilateral filter [12], Porikli's O(1) bilateral filter [17] and Yang's real-time bilateral filters [25, 26]. These methods rely on quantization, and will degrade the performance as demonstrated in [18]. Paris and Durand's method

was implemented on graphics processing unit (GPU) and used for stereo matching. However, the reconstruction accuracy is much lower than the full-kernel implementation [6]. Recently, He *et al*. [19] proposed a new edge-aware filter called guided image filter. Unlike bilateral filter, its runtime is linear in the number of image pixels, and was demonstrated (in [7]) to outperform all the other local methods on Middlebury benchmark [20] both in terms of speed and accuracy.

All of the above cost aggregation methods greatly advance the state of stereo vision in the indicated ways, but they are adversely affected by the local nature of traditional window-based cost aggregation algorithms and are vulnerable to the lack of texture.

In this paper, the cost aggregation problem is re-examined and a non-local solution is proposed. Similar to previous edge-aware filters, a guidance image (typically the reference camera image) will be used to compute the pixel similarity for adaptive aggregation. However, the guidance image is treated as a connected, undirected graph in this paper. The vertices are all the image pixels and the edges are all the edges between the nearest neighboring pixels. The matching cost values are then aggregated adaptively based on pixel similarity in a way similar to bilateral filtering but on a minimum spanning tree (**MST**) derived from this graph. The similarity between any two vertices is decided by their shortest distance on the MST.

A great advantage of using MST is that it gives a more natural image pixel similarity measurement metric. It is more accurate than the previous methods so that every pixel in the image can correctly contribute to all the other pixels during cost aggregation. In contrast, local cost aggregation methods require a user-specified or automatically detected window, and only pixels inside this window provide supports. The proposed method is thus a *non-locally-optimal* solution for cost aggregation problem. It is theoretically better than traditional local solutions for low texture regions. This is straightforward as the window used by the local methods cannot guarantee to cover the whole low textured region. Quantitative evaluation on Middlebury benchmark [20] shows that the proposed method outperforms the guided image filter based cost aggregation method [7] which is the state-of-the-art local method (thus all local methods). The proposed method can be naturally extended to the time domain for enforcing temporal coherence without deteriorating the depth edges when all the video frames are considered.

比引导滤波还好

The proposed non-local method also offers the advantage of an extremely low computational complexity: only a total of 2 addition/subtraction operations and 3 multiplication operations are required for each pixel at each disparity level. It is very close to the computational complexity of unnormalized box filter which requires 6 addition/subtraction operations. Specifically, the average runtime of the proposed method is about *90* milliseconds for the Middlebury data sets on a MacBook Air laptop computer with a 1.8 GHz Intel Core i7 CPU and 4 GB memory, and is only about $1.25\times$ slower than unnormalized box filter[24] but $10\times$ faster than guided image filter based cost aggregation [7] which is the most efficient edge-aware local cost aggregation method.

A simple but effective non-local disparity refinement method is also proposed based on the non-local cost aggregation method. A new cost volume will be computed based on the current disparity estimates and the pixel membership (stable or unstable) obtained from mutual consistency check. The proposed non-local cost aggregation method is then applied to this cost volume, followed by a winner-take-all operation. This method is more robust and effective than the local disparity refinement method in [7] and about $70\times$ faster.

The main contribution of this paper is a novel cost aggregation method with the following advantages:

1. it is a *non-local* solution, which theoretically and experimentally outperforms local cost aggregation methods for low textured regions.
2. its computational complexity is extremely low: only a total of **2 addition/subtraction** operations and **3 multiplication** operations are required for each pixel at each disparity level.
3. it can be used for *non-local* disparity refinement, which is proved to be more robust and effective than the *local* disparity refinement method presented in [7] and about **70**$\times$ faster.

2. Non-Local Cost Aggregation

In this section, the bilateral filter is briefly introduced, and cost aggregation on a tree structure is proposed. A linear time algorithm is presented afterwards. The computational complexity is discussed finally.

2.1. Cost Aggregation using the Bilateral Filter

Edge-aware filters like bilateral filter are known to be effective for edge-preserving smoothing and have been demonstrated to be very effective for local matching cost aggregation. Let $C_d(p)$ denote the matching cost for pixel p at disparity level d, and $C_d^A(p)$ denote the aggregated cost. Using a guidance image I (which is typically the reference camera image), the bilateral filter can be used to compute the aggregated cost as follows

$$C_d^A(p) = \frac{\sum_q exp(-\frac{|p-q|}{\sigma_S}) exp(-\frac{|I(p)-I(q)|}{\sigma_R}) C_d(q)}{\sum_q exp(-\frac{|p-q|}{\sigma_S}) exp(-\frac{|I(p)-I(q)|}{\sigma_R})}, \quad (1)$$

where q is a pixel within the user-specified support region, and σ_S and σ_R are two constants used to adjust the spatial

similarity and the range (intensity/color) similarity respectively. If the guidance image I is different from the original matching cost, Eqn. (1) is a joint bilateral filter. The normalization step in Eqn. (1) is usually not required when the same guidance image I is used for all disparity levels:

$$C_d^A(p)=\sum_q exp(-\frac{|p-q|}{\sigma_S})exp(-\frac{|I(p)-I(q)|}{\sigma_R})C_d(q). \quad (2)$$

2.2. Cost Aggregation on a Tree Structure

Unlike previous cost aggregation methods, in this paper, the guidance image I is represented as a connected, undirected graph $G = (V, E)$, with weight function $w : E \rightarrow R$ mapping edges to real-valued weights. The vertices V are all the image pixels and the edges E are all the edges between the nearest neighboring pixels. The graph G is thus simply the standard $4-$connected grid, and is a planar graph. In this paper, the weight function w is actually the image gradient operation. Let s and r be a pair of neighboring pixels, the weight between s and r is

$$w(s, r) = w(r, s) = |I(s) - I(r)|.^1 \quad (3)$$

From G, a spanning tree can be computed by removing "unwanted" edges. Obviously, we do not want to aggregation across the depth edges which are typically also color/intensity edges. Edges with large weights thus will be removed during spanning tree construction. The result turns out to be the minimum spanning tree (**MST**) which connects all the vertices, and the sum of its weights is minimum out of all spanning trees.

It is straightforward to define the similarity between two image pixels using a MST. If the two pixels are close in a MST, then they are similar, vice versa. The distance between two nodes in a MST is the sum of weights of the connected edges (that is the shortest path) between the two nodes. Let $\mathcal{D}(p, q) = \mathcal{D}(q, p)$ denote the distance between p and q in the MST, and

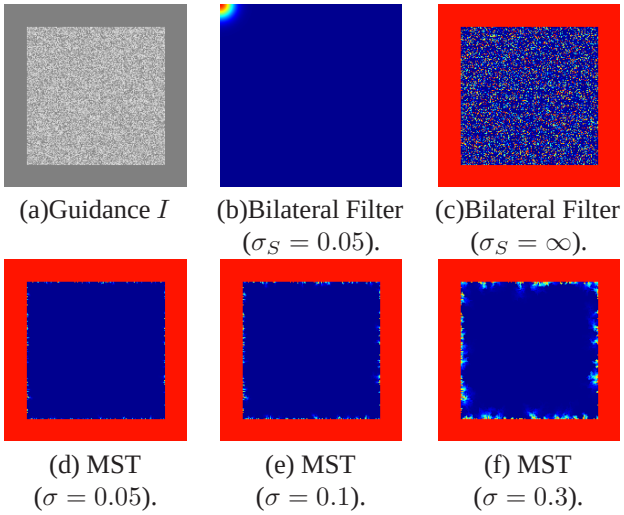
$$\mathcal{S}(p, q) = \mathcal{S}(q, p) = exp(-\frac{\mathcal{D}(p, q)}{\sigma}) \quad (4)$$

denote the similarity between p and q where σ is a constant used to adjust the similarity between two nodes, the joint bilateral filter in Eqn. (2) can then be directly extended to MST structure:

$$C_d^A(p) = \sum_q \mathcal{S}(p, q)C_d(q) = \sum_q exp(-\frac{\mathcal{D}(p, q)}{\sigma})C_d(q). \quad (5)$$

Note that aggregation using MST relaxes two ambiguities σ_S and σ_R in Eqn. (2) to a single ambiguity σ in Eqn. (5).

[1]For a color guidance image, the maximum $w(s, r)$ value computed separately from the three channels will be selected as correct.



(a)Guidance I (b)Bilateral Filter ($\sigma_S = 0.05$). (c)Bilateral Filter ($\sigma_S = \infty$). (d) MST ($\sigma = 0.05$). (e) MST ($\sigma = 0.1$). (f) MST ($\sigma = 0.3$).

Figure 1. Support weights computed from a synthetic image. (a) is the guidance image I, (b)-(f) are support weights received by the first image pixel (on the top left of I). The support weight in (b)-(f) decreases from 1 to 0 as the color changes from red to blue. (b)-(c) are computed using joint bilateral filter with $\sigma_R = 0.05$. σ_S is set to infinity in (c) so that every pixel contributes its support. (d)-(f) are support weights computed using the MST derived from (a) with $\sigma = 0.05, 0.1, 0.3$, respectively. Note that the support weights in (d)-(f) are more natural for cost aggregation than (b)-(c).

A great advantage of using MST is that it gives a more natural image pixel similarity measurement metric. It is more accurate than the previous methods so that every pixel in the image can correctly contribute to all the other pixels during cost aggregation. In contrast, previous cost aggregation methods require a user-specified or automatically detected window, and only pixels inside this window provide supports. This is equal to giving zero support weights to pixels outside the window. This window-based/local cost aggregation method is not natural and it is impossible to find a window that is optimal for different data sets. Fig. 1 shows one such failure case. Fig. 1 (a) is the guidance image I. The flat region in Fig. 1 (a) has uniform intensity, and it is natural that the support weights received by the first image pixel (on the top left of I) from this region are high. However, this is not true when the bilateral filter is used with a relative small window ($\sigma_S = 0.05$[2]) as shown in Fig. 1 (b). If a huge window ($\sigma_S = \infty$) is chosen to cover every pixel in I, the behavior of the bilateral filter will be correct in the flat region, but a relatively large amount of noises will be introduced by the texture region in Fig. 1 (a) as can be seen in Fig. 1 (c).

The use of a MST sheds some light on "non-local" cost aggregation. It automatically accepts similar pixels (close on MST) by assigning them relatively large support weight-

[2]Normalized image coordinate is used such that it resides in [0, 1]. Image intensity is also normalized such that it ranges from 0 to 1.

这话文艺

s and rejects dissimilar pixels (far way on MST) by giving them tiny support weights (that are very close to zero). Fig. 1 (d)-(f) present the support weights computed using MST derived from Fig. 1(a) with $\sigma = 0.05, 0.1, 0.3^2$, respectively. Note that for pixels that are far away from the flat region (e.g., pixels in the image center), their supports are always extremely low (shown in blur color) even when σ is relatively large. Meanwhile, the supports from the flat region are always high (shown in red color).

2.3. A Linear Time Exact Algorithm

Another great advantage of the proposed non-local method over local optimization is that the computational complexity is extremely low. The efficiency originates from the pixel similarity measurement metric defined by MST in Sec. 2.2. The advantage is that the similarity depends on the distance of the two nodes in MST which can be accumulated when MST is traced from the leaf nodes to the root node (as shown in Fig. 2 (a)).

Claim 1. *Let T_r denote a subtree of a node s and r denote the root node of T_r, then the supports node s received from this subtree is the summation of 1) the supports node s received from r and 2) $\mathcal{S}(s,r)$ times the supports node r received from its subtrees.*

Claim 1 is directly obtained from Eqn. (5) and the definition of spanning tree. Let v denote the nodes on r's subtrees, then the supports r received from its subtrees is $\sum_v \mathcal{S}(r,v)C_d(v)$, and the supports s received from T_r can be separated into two parts including 1) support from r: $\mathcal{S}(s,r)C_d(r)$; 2) supports from r's subtrees: $\sum_v \mathcal{S}(s,v)C_d(v) = \sum_v \mathcal{S}(s,r)\mathcal{S}(r,v)C_d(v) = \mathcal{S}(s,r)\sum_v \mathcal{S}(r,v)C_d(v) = \mathcal{S}(s,r)$ times the supports node r received from its subtrees.

Based on Claim 1, the matching cost is firstly aggregated from the leaf nodes towards root nodes as shown in Fig. 2 (a) where the numbers are the distance between the nodes and their parents, and v_4 is the root node. Let $C_d^{A\uparrow}$ denote the aggregated cost values and $P(v_c)$ denote parent of node v_c, then at each node $v \in V$,

$$C_d^{A\uparrow}(v) = C_d(v) + \sum_{P(v_c)=v} \mathcal{S}(v, v_c) \cdot C_d^{A\uparrow}(v_c). \quad (6)$$

Note that if node v is a leaf node (that has no child), then $C_d^{A\uparrow}(v) = C_d(v)$ according to Eqn. (6).

Claim 2. *Let T_r denote a subtree of a node s and r denote the root node of T_r, and let $C_d^A(s)$ denote all the supports received by node s on the tree, then the supports r received from nodes other than T_r is $\mathcal{S}(s,r) \cdot [C_d^A(s) - \mathcal{S}(r,s) \cdot C_d^{A\uparrow}(r)]$.*

After aggregation, $C_d^{A\uparrow} = C_d^A$ for the root node of MST which receives supports from all nodes on MST. The rest

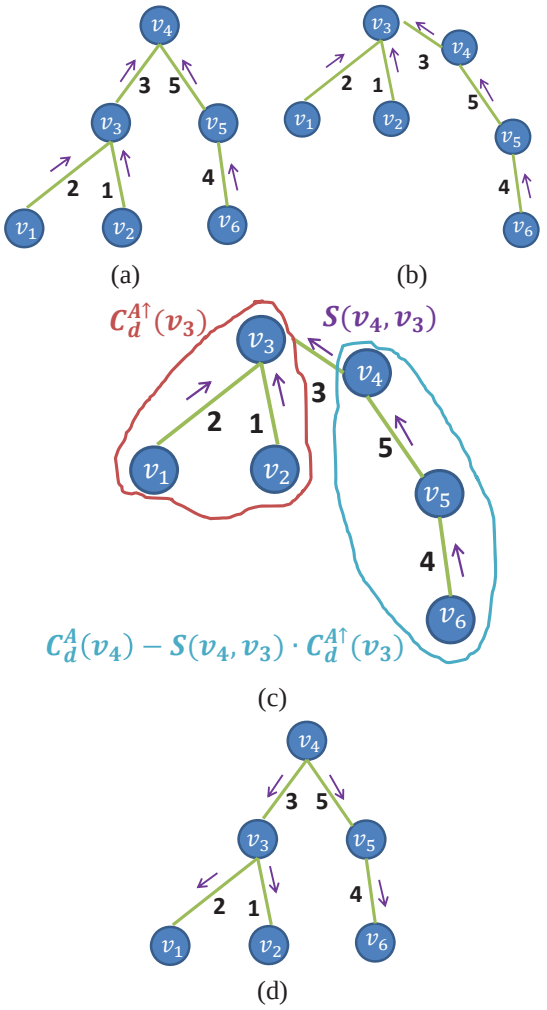


Figure 2. Cost aggregation on a MST.

only receive supports from its subtrees. The simplest solution is iteratively changing the root node and iteratively aggregating the cost as shown in Fig. 2 (b). This is obviously too slow, but it gives a clue. Note that in Fig. 2 (c), the aggregated cost value $C_d^{A\uparrow}(v_3)$ (note: grouped in red circle in Fig. 2 (c)) at node v_3 contains supports received from its subtrees (node v_1 and v_2) and itself, thus it is only necessary to aggregate the cost from the leaf nodes of v_3's other subtrees (note: grouped in blue circle in Fig. 2 (c)) according to Eqn. (6), which is actually $\mathcal{S}(v_4, v_3)$ times the summation of the supports node v_4 received from its subtrees (when v_3 is the root node) and itself according to Claim 1. Also note that the aggregated cost value $C_d^{A\uparrow}(v_4) = C_d^A(v_4)$ contains supports from all nodes if v_4 is the root node, thus $C_d^A(v_4) - \mathcal{S}(v_4, v_3) \cdot C_d^{A\uparrow}(v_3)$ (note: grouped in blue circle in Fig. 2 (c)) are actually all the supports v_4 received from nodes other than v_3 and its subtrees. Thus, $\mathcal{S}(v_4, v_3) \cdot [C_d^A(v_4) - \mathcal{S}(v_3, v_4) \cdot C_d^{A\uparrow}(v_3)]$ is actually

the supports received by node v_3 from v_4 and its subtrees as shown in Fig. 2 (c).

According to Claim 2, the aggregated cost value $C_d^A(v)$ for any node v on a MST can be obtained from its parent $P(v)$ as follows

$$\begin{aligned} C_d^A(v) &= C_d^{A\uparrow}(v) \\ &+\mathcal{S}(P(v),v)\cdot[C_d^A(P(v)) - \mathcal{S}(v,P(v))\cdot C_d^{A\uparrow}(v)], \\ &= \mathcal{S}(P(v),v)\cdot C_d^A(P(v)) + [1-\mathcal{S}^2(v,P(v))]\cdot C_d^{A\uparrow}(v). \end{aligned} \quad (7)$$

Because $C_d^A = C_d^{A\uparrow}$ for the root node of MST, the aggregated cost value at each node can be obtained using Eqn. (7) by tracing from the root node of MST towards its leaf nodes as shown in Fig. 2 (d). Hence, the whole cost aggregation process is separated into two steps: 1) aggregate the original matching cost C_d from leaf nodes towards root node using Eqn. (6) (shown in Fig. 2 (a)) and store it as $C_d^{A\uparrow}$; 2) aggregate $C_d^{A\uparrow}$ from root node towards leaf nodes using Eqn. (7) (shown in Fig. 2 (d)).

2.4. Computational Complexity

Note that in Eqn. (6) and (7), $\mathcal{S}(v,v_c)$, $\mathcal{S}(P(v),v)$ and $1-\mathcal{S}^2(v,P(v))$ only depend on the edges of MST and can be pre-computed, thus only a total of *2 addition/subtraction* operations and *3 multiplication* operations will be required for each node at each disparity level during cost aggregation. Unnormalized box filtering using integral image[24], on the other hand, requires 3 *addition/subtraction* operations for each pixel at each disparity level for computing the integral image, and 3 *addition/subtraction* operations on the integral image to compute the locally aggregated cost value, thus a total of *6 addition/subtraction* operations [8] is required at each pixel and each disparity level for local aggregation using integral image. Note that floating point addition takes $3-6$ clock cycles and multiplication takes $4-8$ clock cycles, depending on the microprocessor [9]. Hence, the speed of the proposed method is close to (at most $1.7\times$ slower than) unnormalized box filter, which is the most efficient cost aggregation method. The proposed non-local method and unnormalized box filter are tested on a MacBook Air laptop computer with a 1.8 GHz Intel Core i7 CPU and 4 GB memory. The average runtime of the four standard Middlebury data sets (including *Tsukuba*, *Venus*, *Teddy* and *Cones* data sets) is about *90* milliseconds using the proposed method, and the average runtime of unnormalized box filter is about *72* milliseconds. Note that the speed of the two methods is similar: the proposed method is only about $1.25\times$ slower. The average runtime of the guided image filter based cost aggregation method [7] (which is currently the most efficient edge-aware cost aggregation algorithm) was also computed using the same laptop computer, and is *960* milliseconds, which is about $10\times$ slower than the proposed method. The proposed method requires an extra step to compute the MST. Because the graph derived from the guidance image is a planar graph, the complexity of computing its MST is linear to the number of image pixels [4, 10]. Specifically, the runtime is about *7* milliseconds on average on the Middlebury data sets. Because MST is computed only once for every stereo pair and is independent of the number of disparity levels, the computation can be ignored usually, especially when the number of disparity levels is large.

3. Non-Local Disparity Refinement

In this section, a new disparity refinement method is proposed using the non-local cost aggregation method presented in Sec. 2. First, the non-local cost aggregation method is run in turn with both the left and the right image as reference images to obtain two corresponding disparity maps. This is done just to support a subsequent mutual consistency check (often called left right check) that divides all the pixels into stable or unstable pixels. A stable pixel should pass the mutual consistency check requiring that on the pixel grid that the left and right disparity maps are computed, they are perfectly consistent (same disparity value). Let D denote the left disparity map, a new cost value is computed for each pixel p at each disparity level d as

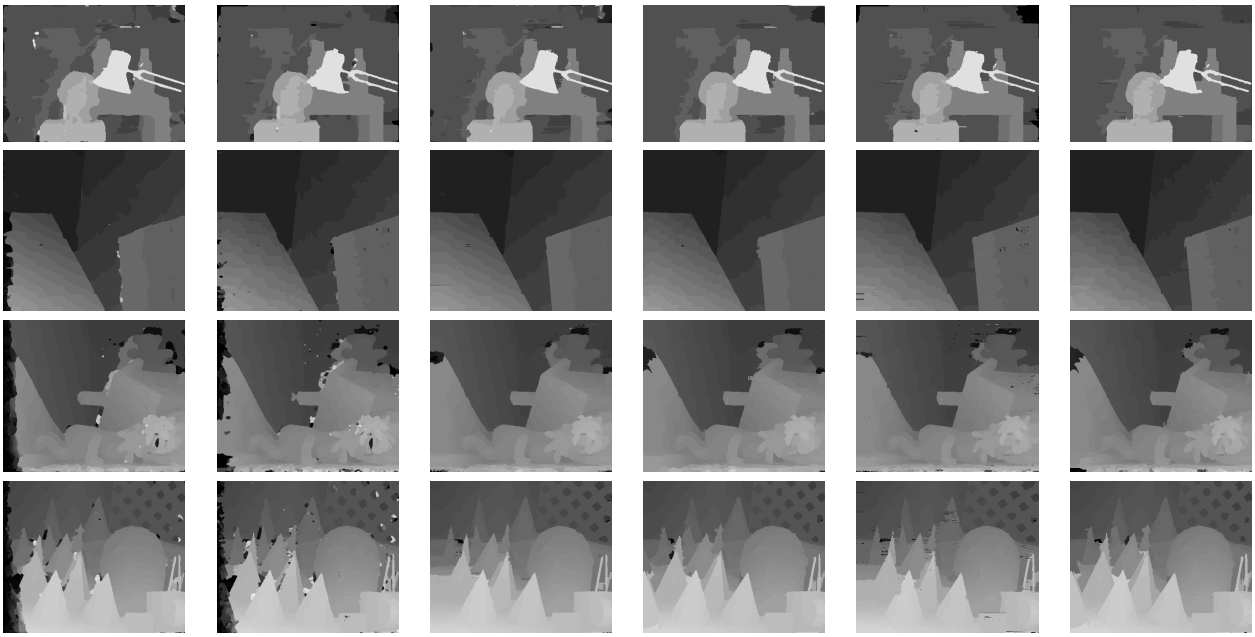
$$C_d^{new}(p) = \begin{cases} |d - D(p)| & p \text{ is stable and } D(p) > 0, \\ 0 & \text{else.} \end{cases} \quad (8)$$

Note that for all unstable pixels (due to occlusion, lack of texture, specularity, *etc*), the cost value $C_d^{new}(p)$ will be zero for all disparity levels, thus will completely depend on the stable pixels. If a winner-take-all selection is applied to this new cost volume, then the disparity values of the stable pixels will be the same as disparity map D, and the rest zero.

The non-local cost aggregation method presented in Sec. 2 is then employed to aggregate the new cost values $C_d^{new}(p)$ to propagate the disparity values from stable pixels to unstable pixels. The advantages of this method are

1. efficient. As summarized in Sec. 2.4, the computational complexity of this method is close to unnormalized box filter using integral image.
2. non-local. The size of unstable regions can be huge. Theoretically, it allows the total number of unstable pixels to be as large as the number of image pixels minus one. In this extreme case, the disparity values of the unstable pixels will be the same as the disparity of the stable pixel after refinement.

Note that this non-local property differentiates the proposed disparity refinement method from the others which usually use a local filter or color segmentation. For instance,



(a) *Local* guided image filter based cost aggregation[7] (**8.33%**).

(b) *Non-local* cost aggregation in Sec. 2 (**6.82%**).

(c) *Local* guided aggregation[7] + *local* refinement[7] (**5.82%**).

(d) *Local* guided aggregation[7] + *non-local* refinement (**5.58%**).

(e) *Non-local* aggregation + *local* refinement[7] (**6.56%**).

(f) *Non-local* aggregation + *non-local* refinement (**5.55%**).

Figure 3. Experimental results on the Middlebury data sets[20]. (a)-(b) are disparity maps obtained using the *local* guided image filter based cost aggregation method [7] and disparity maps obtained using the *non-local* cost aggregation method proposed in Sec. 2, respectively. The bold numbers under the images are the average errors (percentages of bad pixels) which shows that our method outperforms the guided filter [7]. The corresponding quantitative evaluation is summarized in Table 1. Note that the proposed method performs better around large low textured regions. For instance, see *Tsukuba* data set (1^{st} row), disparities of the low textured region between the head and the lamp in (b) is more accurate than (a). (c)-(f) present visual comparison of two disparity refinement methods with numerical comparison provided in Table 2. (c) and (e) are disparity maps obtained using the *local* weighted median filter base disparity refinement method in [7], and (d) and (f) are disparity maps obtained using the *non-local* disparity refinement method proposed in Sec. 3. (c)-(d) use the *local* guided image filter based cost aggregation method in [7] and (e)-(f) use the *non-local* cost aggregation method proposed in Sec. 2. Note: the reader is urged to view these images at full size on a video display, for details may be lost in hard copy.

[7] also presents a disparity refinement method for stereo matching. A median filter overlap the unstable regions is used to propagate the disparity from stable pixel to the unstable regions. The unstable region should thus be relative small. In contrast, the method proposed in this Section does not make any assumption about the size of the unstable region.

4. Experimental Results

In this section, results supporting the claim that the algorithm is currently the strongest cost aggregation method available on the Middlebury data sets[20] is reported. The guided image filter [19] has recently been proved to be the top performer [7] among all local methods on Middlebury benchmark [20]. As a result, demonstrating that the proposed non-local method outperforms this local filter[3] will be enough. The exact matching cost measurement method used in [7] is thus adopted. The guided image filter and the proposed non-local method are then used to aggregate the matching cost values. Finally, a winner-take-all operation is employed to obtain the disparity maps. The proposed non-local cost aggregation method has a single parameter σ, and is set to 0.1 throughout the paper because the optimal σ value for the Middlebury data sets [20] is 0.12 without disparity refinement (6.68% bad pixels on average) and 0.08 with disparity refinement (5.46% bad pixels on average).

The disparity maps of the Middlebury data sets computed using the guided image filter based cost aggregation method[7] and the proposed non-local cost aggregation

[3]The source code provided by the authors of [7] is used to obtain the disparity maps computed from the guided image filter.

Algorithm	Avg. Rank	*Tsukuba* nonocc	*Tsukuba* all	*Tsukuba* disc	*Venus* nonocc	*Venus* all	*Venus* disc	*Teddy* nonocc	*Teddy* all	*Teddy* disc	*Cones* nonocc	*Cones* all	*Cones* disc	**Avg. Error**
Fig. 3 (a): local guided image filter [7]	62.4	2.08	2.88	8.40	1.58	2.77	16.0	7.79	16.4	18.4	3.06	12.1	8.60	8.33
Fig. 3 (b): proposed **non-local** method	49.4	1.68	2.33	7.36	0.59	1.15	5.45	6.81	14.1	15.9	3.84	12.2	10.6	**6.82**

Table 1. Quantitative comparison of the *local* guided image filter based cost aggregation method in [7] and the proposed *non-local* cost aggregation methods on the Middlebury benchmark [20] with error threshold 1. The numbers in the last thirteen columns are the percentages of the pixels with incorrect disparities on different data sets and on average. As can be seen from the table, the performance of the proposed non-local cost aggregation method is higher than guided filter [7], which is currently the best edge-preserving local stereo method both in terms of speed and accuracy. Moreover, the proposed non-local cost aggregation method is about **10**$\times$ faster than [7] on average.

Algorithm	Avg. Rank	*Tsukuba* nonocc	*Tsukuba* all	*Tsukuba* disc	*Venus* nonocc	*Venus* all	*Venus* disc	*Teddy* nonocc	*Teddy* all	*Teddy* disc	*Cones* nonocc	*Cones* all	*Cones* disc	**Average Error**
Fig. 3 (c): local [7] + local [7]	32.0	1.87	2.23	8.07	0.28	0.46	2.75	6.79	12.3	16.4	2.75	8.16	7.82	5.82
Fig. 3 (d): local [7] + **non-local**	25.7	1.62	2.01	7.00	0.13	0.34	1.53	6.80	12.4	16.3	2.73	8.35	7.69	**5.58**
Fig. 3 (e): **non-local** + local [7]	45.4	2.33	2.64	9.44	0.50	0.61	3.93	6.77	11.9	15.9	4.20	9.23	11.3	6.56
Fig. 3 (f): **non-local** + **non-local**	28.6	1.56	1.91	8.25	0.28	0.42	2.72	5.99	11.5	14.0	3.00	8.53	8.47	**5.55**

Table 2. Numerical comparison of the *local* weighted median filter base disparity refinement method in [7] and the proposed *non-local* disparity refinement method on the Middlebury benchmark [20] with error threshold 1. The numbers in the last thirteen columns are the percentages of the pixels with incorrect disparities on different data sets and on average. As can be seen from the table, the proposed non-local disparity refinement method is more robust than [7] and the improvement is always higher than the weighted median filter base disparity refinement method in [7]. Moreover, the proposed non-local disparity refinement method is about **70**$\times$ faster than [7] on average.

method are presented in Fig. 3 (a)-(b). The corresponding quantitative evaluation is presented in Table 1, which shows that the proposed non-local cost aggregation method outperforms the local guided image filter. Besides, the speed of the proposed non-local cost aggregation method is about **10**$\times$ faster. Visual comparison in Fig. 3 (a)-(b) shows that the proposed non-local cost aggregation performs better around low textured regions. For instance, the low textured region between the head and the lamp in *Tsukuba* data set (first row of Fig. 3 (a)-(b)). However, the proposed method may be less accurate around highly-textured regions where stereo matching is expected to perform well (but may be not due to noises), thus the supports from the neighbors are relatively small according to the proposed method. Errors around highly-textured regions are mostly due to noises and will cause inconsistency between the left and right disparity maps. Mutual consistency check will classify these pixels into unstable pixels, and the proposed non-local disparity refinement method is expected to correct these errors.

Fig. 3 (c)-(f) visually compare the weighted median filter based disparity refinement method presented in [7] and the proposed non-local disparity refinement method. (c)-(d) used the same the guided image filter based cost aggregation method but different disparity refinement methods, and (e)-(f) used the proposed non-local cost aggregation method but different disparity refinement methods. As expected, most of the noises in Fig. 3 (b) are removed by the proposed non-local disparity refinement method as can be seen in Fig. 3 (f). However, these noises remain in Fig. 3 (e) which are obtained using the local disparity refinement method in [7]. Quantitative evaluation is summarized in Table 2, which shows that the proposed non-local disparity refinement method outperforms the local method in [7] for the two different cost aggregation methods. Table 2 also shows that the local disparity refinement method in [7] is not suitable for the proposed non-local cost aggregation method as the improvement is not evident. The average error is reduced by only 0.26% (from 6.82% to 6.56%) after disparity refinement, and the errors of the *Tsukuba* and *Cones* data sets increase. The average runtime of the local disparity refinement method in [7] is about 6.5 second on the Middlebury data sets. It is about **70**$\times$ slower than the proposed non-local disparity refinement method.

5. Conclusion

To avoid being adversely affected by the local nature of traditional window-based cost aggregation algorithms thus vulnerable to the lack of texture, this paper try to shed some light on developing non-local algorithms by suggesting aggregating cost on a tree structure. The proposed MST-based non-local solution has been demonstrated to outperform all local cost aggregation methods on Middlebury benchmark [20] both in terms of speed and accuracy. In the near future, redundance among the disparity search range [11] and parallel algorithms for the proposed non-local aggregation method [1] will be considered and more matching cost methods will be tested with the proposed method for a better understanding of its behavior.

References

[1] D. Bader and G. Cong. A fast, parallel spanning tree algorithm for symmetric multiprocessors (smps). *Journal of Parallel and Distributed Computing*, 65:994–1006, 2005.

[2] A. Bobick and S. Intille. Large occlusion stereo. *IJCV*, 33:181–200, 1999.

[3] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *PAMI*, 23(11):1222–1239, 2001.

[4] D. CHERITON and R. E. TARJAN. Finding minimum spanning trees. *SIAM Journal on Computing (SICOMP)*, pages 724–742, 1976.

[5] Q. Yang, R. Yang, J. Davis, and D. Nistér. Spatial-depth super resolution for range images. In *CVPR*, 2007.

[6] K.-J. Yoon and I.-S. Kweon. Adaptive support-weight approach for correspondence search. *PAMI*, 28(4):650–656, 2006.

[7] C.Rhemann, A. Hosni, M. Bleyer, C. Rother, and M. Gelautz. Fast cost-volume filtering for visual correspondence and beyond. In *CVPR*, pages 3017–3024, 2011.

[8] F. Crow. Summed-area tables for texture mapping. In *Siggraph*, 1984.

[9] A. Fog. *Optimizing software in C++: An optimization guide for Windows, Linux and Mac platforms*. 2011.

[10] T. Matsui. The minimum spanning tree problem on a planar graph. *Discrete Applied Mathematics*, pages 91–94, 1995.

[11] D. Min, J. Lu, and M. Do. A revisit to cost aggregation in stereo matching: How far can we reduce its computational redundancy? In *ICCV*, 2011.

[12] S. Paris and F. Durand. A fast approximation of the bilateral filter using a signal processing approach. *IJCV*, 81:24–52, January 2009.

[13] W. T. Freeman, E. Pasztor, and O. T. Carmichael. Learning low-level vision. *IJCV*, 40(1):25–47, 2000.

[14] J. Sun, N. Zheng, and H. Y. Shum. Stereo matching using belief propagation. *PAMI*, 25(7):787–800, 2003.

[15] Q. Yang, L. Wang, and N. Ahuja. A constant-space belief propagation algorithm for stereo matching. In *CVPR*, pages 1458–1465, 2010.

[16] Q. Yang, L. Wang, R. Yang, H. Stewenius, and D. Nister. Stereo matching with color-weighted correlation, hierachical belief propagation and occlusion handling. *PAMI*, 31(3):492–504, 2009.

[17] F. Porikli. Constant time o(1) bilateral filtering. In *CVPR*, 2008.

[18] C. Richardt, D. Orr, I. Davies, A. Criminisi, and N.Dodgson. Real-time spatiotemporal stereo matching using the dual-cross-bilateral grid. In *ECCV*, pages 510–523, 2010.

[19] K. He, J. Sun, and X. Tang. Guided image filtering. In *ECCV*, pages 1–14, 2010.

[20] D. Scharstein and R. Szeliski. Middlebury stereo evaluation.
http://vision.middlebury.edu/stereo/eval/.

[21] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 47:7–42, 2002.

[22] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *ICCV*, pages 839–846, 1998.

[23] O. Veksler. Stereo correspondence by dynamic programming on a tree. In *CVPR*, pages 384–390, 2005.

[24] P. Viola and M. Jones. Robust real-time face detection. volume 57, pages 137–154, 2003.

[25] Q. Yang, K.-H. Tan, and N. Ahuja. Real-time o(1) bilateral filtering. In *CVPR*, 2009.

[26] Q. Yang, S. Wang, and N. Ahuja. Svm for edge-preserving filtering. In *CVPR*, pages 775–1782, 2010.