Segment-Tree based Cost Aggregation for Stereo Matching

Xing Mei[1], Xun Sun[2], Weiming Dong[1], Haitao Wang[2], Xiaopeng Zhang[1]
[1]NLPR, Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2]China Lab, Samsung Advanced Institute of Technology, Beijing, China
{xmei,wmdong,xpzhang}@nlpr.ia.ac.cn, {xunshine.sun,ht.wang}@samsung.com

Abstract

This paper presents a novel tree-based cost aggregation method for dense stereo matching. Instead of employing the minimum spanning tree (MST) and its variants, a new tree structure, "Segment-Tree", is proposed for non-local matching cost aggregation. Conceptually, the segment-tree is constructed in a three-step process: first, the pixels are grouped into a set of segments with the reference color or intensity image; second, a tree graph is created for each segment; and in the final step, these independent segment graphs are linked to form the segment-tree structure. In practice, this tree can be efficiently built in time nearly linear to the number of the image pixels. Compared to MST where the graph connectivity is determined with local edge weights, our method introduces some 'non-local' decision rules: the pixels in one perceptually consistent segment are more likely to share similar disparities, and therefore their connectivity within the segment should be first enforced in the tree construction process. The matching costs are then aggregated over the tree within two passes. Performance evaluation on 19 *Middlebury data sets shows that the proposed method is comparable to previous state-of-the-art aggregation methods in disparity accuracy and processing speed. Furthermore, the tree structure can be refined with the estimated disparities, which leads to consistent scene segmentation and significantly better aggregation results.*

1. Introduction

Dense two-frame stereo matching is one of the most extensively studied areas in computer vision. A stereo algorithm usually takes four steps [15]: matching cost computation, cost aggregation, disparity computation and disparity refinement. In step 1, the matching costs are initialized for each pixel at all possible disparity levels; in step 2, the costs are aggregated over each pixel's support region; in step 3, the disparities are computed with a local or global optimizer; and finally in step 4, the disparity results are refined with various post-processing techniques. We mainly focus on cost aggregation (step 2), which has great impact on the speed and accuracy of a stereo system. This step is required by all the local methods [14, 25], and it also turns out to be an important building block for many top-performing global algorithms [11, 24].

Most aggregation methods work by defining a local support window for each pixel and averaging the costs over the region, which are therefore closely related to image filtering techniques. While simple box/gaussian smoothing techniques can be adopted due to their high efficiency, they produce low-quality results with blurred depth boundaries [15]. Yoon and Kweon first proposed to filter the cost volume with a joint bilateral filter, which effectively preserves depth boundaries [25]. Since then, various edge-aware filtering techniques have been explored for cost aggregation, such as geodesic weight [7] and segment support [20]. He *et al.* presented a guided image filter with running time independent of kernel size [6]. This filter shows leading speed and accuracy performance in two recent stereo methods [3, 14].

Recently, Yang first proposed a non-local cost aggregation method [23]. Different from previous methods that rely on pixelwise support regions, Yang's method performs non-local cost aggregation over the image with a tree structure. The reference image is treated as a 4-connected, undirected planar graph: the nodes are image pixels, and the edges are all the edges between neighboring pixels. The tree is then constructed as a minimum spanning tree (MST) over this graph. By traversing the tree in two sequential passes (one from leaf to root, another one from root to leaf), each pixel receives proper contributions from all the other pixels in the image. Evaluation on the standard Middlebury benchmark [16] shows that this non-local method outperforms the guided image filter in aggregation accuracy. And it is computationally very efficient, with running time comparable to uniform box filtering.

Besides aggregation, MST and its variants have also been used as graphical models for many global stereo methods such as tree-based dynamic programming [10, 21] and graph cut on sparse graph [18]. MST is built by greedily choosing edges with small weights from the image graph.

The edge weights are determined with color differences, which are therefore sensitive to highly textured regions, image sampling and local noise. Edges with equal weights also lead to non unique tree structures [21].

In this paper, we propose a novel tree structure, Segment-Tree (ST), for non-local cost aggregation. We first divide the image graph into a set of coherent segments, then construct a sub-tree for each segment, and finally connect these sub-trees to build the ST structure.

Although MST has shown good performance in cost aggregation and other stereo work [21, 23], we believe ST is competitive for two reasons. First, MST is constructed with local edge weights. ST instead selects edges with both local edge weights and 'non-local' segment properties, which yields a more robust tree structure. Second and more importantly, ST incorporates the segmentation information into the cost aggregation step in a 'soft' way. On the one hand, it conforms to the assumption that the pixels in the same segment are more likely to share similar disparities. By enforcing tight connections for the pixels inside each segment, improved aggregation results and better depth boundaries can be expected. On the other hand, the aggregation weight between any two pixels in the same segment is determined by their geodesic distance over a sub-tree structure, which still allows large disparity variation inside the segment without a hard constraint. This is fundamentally different from previous region-based methods that employ the segmented regions as the basic matching units [8, 10, 22]. Ren proposed a similar idea for optical flow computation, which determines the weight between the pixels with pairwise affinities and avoids early pixel grouping [13].

We quantitatively evaluate the aggregation accuracy with ST, MST and guided image filter on 19 Middlebury data sets, including the standard benchmark. Experimental results show that ST is comparable to the guided filter in accuracy, and outperforms MST on most data sets. And even better results can be achieved if the tree structure is further updated with a color-depth joint segmentation.

A challenging problem with ST is that the construction process might be computationally expensive due to the segmentation operation. We show that, by extending a classic graph-based segmentation method [4], ST can be built in time nearly linear to the number of the image pixels. For the Middlebury data sets, ST-based aggregation method is about 11× faster than the guided image filter [14].

In summary, the contributions of this paper are:

- A novel tree structure for matching cost aggregation.
- An efficient graph-based tree construction algorithm.
- An effective tree structure refinement method.
- Quantitative evaluation of several recent aggregation methods on a number of stereo data sets.

2. Non-Local Cost Aggregation

In this section, we briefly review the non-local cost aggregation method. More details can be found in [23]. Our algorithm follows the same work flow, except that we employ a different tree structure.

In this problem, the reference color/intensity image I is represented as a connected, 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 pixel s and r, its weight is decided as follows:

$$w_e = w(s, r) = |I(s) - I(r)| \tag{1}$$

A tree T can then be constructed by selecting a subset of edges from E. Yang proposed to construct T as a minimum spanning tree (MST). The intuition behind this choice is that edges with small weights are less likely to cross the depth borders. For accurate aggregation, such edges should be selected during the tree construction process, which leads to a MST with the minimum sum of the edge weights. For any two pixels p and q, there is only one path connecting them in T, and their distance $D(p, q)$ is determined by the sum of the edge weights along the path.

Let $C_d(p)$ denote the matching cost for pixel p at disparity level d, the non-local aggregated cost $C_d^A(p)$ is computed as a weighted sum of C_d:

$$C_d^A(p) = \sum_{q \in I} S(p, q) C_d(q) \tag{2}$$

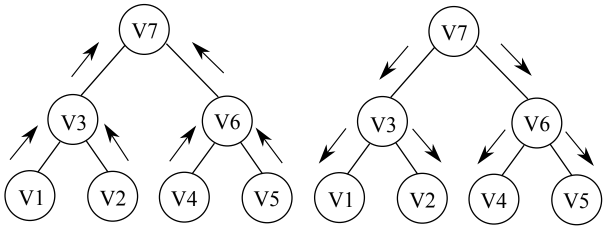
where q covers every pixel in image I. This is different from traditional aggregation methods, where q is limited in a local region around p. $S(p, q)$ is a weighting function, which denotes the contribution of pixel q to p in the sum. With the tree structure T, $S(p, q)$ is defined as follows:

$$S(p, q) = exp(-\frac{D(p, q)}{\sigma}) \tag{3}$$

where σ is a user-specified parameter for distance adjustment. σ is set to 0.1 in all our experiments. The pixels that are closer to p in T contribute more to the aggregated costs.

Aggregation needs to be performed for all the pixels at all disparity levels. A brute force implementation would be prohibitive for practical applications. Yang showed that the aggregated costs for all the pixels can be efficiently computed by traversing the tree structure T in two sequential passes. The aggregation process is illustrated in Figure 1. In the first pass (Figure 1 (a)), the tree is traced from the leaf nodes to the root node. For a pixel p, its cost values are not updated until all its children have been visited:

$$C_d^{A\uparrow}(p) = C_d(p) + \sum_{q \in Ch(p)} S(p, q) \cdot C_d^{A\uparrow}(q) \tag{4}$$



(a) 1st pass: from leaf to root (b) 2nd pass: from root to leaf

Figure 1. Two-pass cost aggregation on a tree structure.

where $C_d^{A\uparrow}$ denotes the the intermediate aggregated costs, and the set $Ch(p)$ contains the children of pixel p. After the first pass, the root node (node $V7$ in Figure 1(a)) receives the weighted costs from all the other nodes, while the rest receive the costs from their subtrees. Then in the second pass (Figure 1(b)), the tree is traversed from top to bottom. Starting from the root node, the aggregated costs are passed to the subtrees. For a pixel p, its final aggregated costs $C_d^A(p)$ are determined with its parent $Pr(p)$ as follows:

$$\begin{aligned} C_d^A(p) =& S(Pr(p), p) \cdot C_d^A(Pr(p)) \\ &+ (1 - S^2(Pr(p), p)) \cdot C_d^{A\uparrow}(p) \end{aligned} \tag{5}$$

The aggregation algorithm runs in very low computational complexity $O(n \cdot l)$, where n is the number of the image pixels, and l denotes the number of the disparity levels. Disparity results can then be computed with the aggregated cost volume C^A and a simple Winner-Take-All (WTA) strategy. A post-processing technique based on the same tree structure was also proposed in the original work [23].

3. Segment-Tree Construction

In this section we first propose a graph-based ST construction algorithm, then we present a simple but effective method to further enhance the tree structure, and lastly we discuss the computational complexity of the algorithm.

3.1. Graph-based Tree Construction

Conceptually, ST can be constructed with the reference image in a three-step process:

1. Image pixels are grouped into a set of segments.

2. A subtree is built for each segment.

3. All the subtrees are linked to produce the final tree.

Step 1 is a typical image segmentation problem, while step 2 and 3 enforce the connectivity inside and around each segment respectively. Technically, any robust segmentation algorithm can be employed in step 1, such as mean-shift clustering [1] and normalized cuts [17]. We instead focus on the graph-based segmentation method proposed by Felzenszwalb and Huttenlocher [4]. We show that this method can be extended to handle the three steps in a unified framework.

Algorithm 1 Graph-based Segment-Tree Construction

```
Input:
    Graph $G = (V, E)$, with $n$ vertices and $m$ edges. Each
    edge $e \in E$ has an associated weight $w_e$.
Output:
    Tree $T = (V, E')$, with $E' \subset E$.
 1: Sort $E$ such that $w_{e_1} \leq w_{e_2} \leq \ldots \leq w_{e_m}$.
 2: Initialize $E' \leftarrow \emptyset$.
 3: for each node $v_i \in V$ do
 4:     Initialize a tree $T_i = (V_i, E_i)$: $T_i \leftarrow \{v_i\}$, $E_i \leftarrow \emptyset$
 5: end for
 6: for each edge $e_j \in E$ do
 7:     Check the nodes $v_p$ and $v_q$ connected by $e_j$.
 8:     if $T_p \neq T_q$ and $e_j$ satisfies Equation (6) then
 9:         Merge $T_p, T_q$ into a new tree $T_{p,q} = (V_{p,q}, E_{p,q})$:
            $V_{p,q} \leftarrow V_p \cup V_q$, $E_{p,q} \leftarrow E_p \cup E_q \cup \{e_j\}$
10:         Update $E'$: $E' \leftarrow E' \cup \{e_j\}$
11:     end if
12: end for
13: Update $E$: $E \leftarrow E - E'$
14: for each edge $e_j \in E$ do
15:     Check the nodes $v_p$ and $v_q$ connected by $e_j$.
16:     if $T_p \neq T_q$ then
17:         Merge $T_p, T_q$ into a new tree $T_{p,q} = (V_{p,q}, E_{p,q})$:
            $V_{p,q} \leftarrow V_p \cup V_q$, $E_{p,q} \leftarrow E_p \cup E_q \cup \{e_j\}$
18:         Update $E'$: $E' \leftarrow E' \cup \{e_j\}$
19:     end if
20:     Break the for loop if $|E'| = |V| - 1$.
21: end for
22: return $T = (V, E')$
```

Our tree construction algorithm is listed in Algorithm 1. As described in Section 2, the reference image is treated as a graph $G = (V, E)$, and a subset of edges $E' \subset E$ is selected for the segment-tree $T = (V, E')$. The algorithm proceeds in three stages:

- **Initialization** (Line 1-5): The edges in E are sorted in a non-decreasing order according to the weights defined in Equation (1), and a subtree is created for each node in V. E' contains no edges.

- **Grouping** (Line 6-13): The subtrees are merged into bigger groups with a full scan of the edge set E. Let v_p and v_q denote the nodes connected by edge $e_j \in E$. If v_p and v_q belong to different subtrees, and the edge weight w_{e_j} satisfies a criterion proposed in [4], the subtrees T_p, T_q are merged into a new subtree $T_{p,q}$. At the same time, e_j is included in E'. The criterion, which considers the relative dissimilarity of the two subtrees, is expressed as follows:

$$w_{e_j} \leq \min(Int(T_p) + \frac{k}{|T_p|}, Int(T_q) + \frac{k}{|T_q|}) \quad (6)$$

where $Int(T_p)$ denotes the maximum edge weight in T_p, and k is a constant parameter. k is set to 1200 in all our experiments. After visiting each edge in E, each subtree corresponds to a visually consistent segment. The edges of these subtrees (already collected in E') are then removed from E.

- **Linking** (Line 14-21): Some more edges are selected from E to link the subtrees. We greedily search for these edges in another scan of E. If an edge connects two different subtrees, we merge the subtrees and include the edge in E'. The search stops when all the trees are finally merged into one component.

All three steps of the construction process are unified in one framework. The **Grouping** stage covers step 1 and 2, and generates a set of segments and their subtrees simultaneously. Furthermore, it can be proved that each subtree is a MST of the corresponding segment [4]. The **Linking** stage covers step 3, and selects edges with small weights to connect neighboring subtrees. In fact, if each segment is treated as a basic graph node, the **Linking** stage builds a MST for this segment graph [2]. Therefore, our method can be seen as a hierarchical algorithm: it creates a MST for the segment graph, and also maintains a MST over each segment. The quality of the segments is controlled by the merging criterion (Equation (6)). Detailed analysis on how Equation (6) helps to capture perceptually consistent, reasonably large regions can be found in [4].

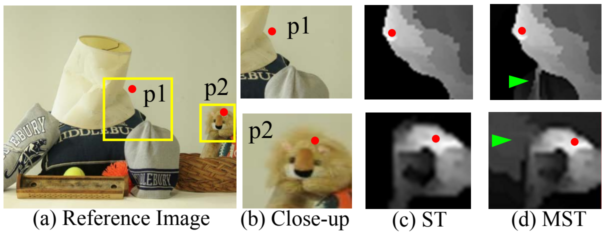


Figure 2. Support weights computed with ST and MST. Higher contributions are mapped to brighter intensity values. (a) Reference image and two selected pixels p_1 and p_2 (b) Close-up of p_1's and p_2's neighborhood (c) Support weights computed with ST (d) Support weights computed with MST. Details are best viewed in the electronic version. Parameter settings: $\sigma = 0.1, k = 1200$.

The advantages of the segment-tree are illustrated with an example image (from the Midd1 data set), as shown in Figure 2. We select two pixels p_1, p_2 from the reference image (marked as red dots), and calculate the support weights of the neighboring pixels with ST and MST respectively. Pixel p_1 belongs to the background, but lies closely to a foreground object which shares similar colors with the background in some local regions. Pixel p_2 lies on a highly textured object with fuzzy borders. For both cases, MST assigns wrong weights to some regions (indicated by green triangles), while ST successfully captures the object boundaries with a proper scene segmentation.

3.2. Enhancement with Color-Depth Segmentation

We further propose to enhance the tree structure with a second segmentation process, which employs both color and the estimated depth information. Our observation is that neighboring regions with different color distributions might still have similar disparities, and such regions should be merged for robust cost aggregation. Besides, it has been shown that improved scene segmentation results can be achieved when both cues are exploited as feature vectors [9, 12].

For joint segmentation, a rough disparity map D is computed with ST and non-local aggregation, as described in Section 2. Then all the edge weights are updated. For an edge e connecting pixel s and r, its weight is updated as follows:

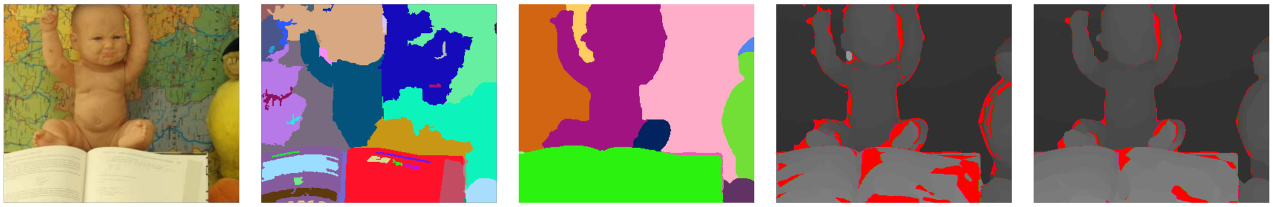
$$w_e = \lambda \frac{|I(s) - I(r)|}{\Delta_I} + (1 - \lambda) \frac{|D(s) - D(r)|}{\Delta_D} \quad (7)$$

where Δ_I and Δ_D are two constants which normalize I and D to the range $[0, 1]$, and $\lambda \in [0, 1]$ is a parameter for balancing the relative contributions of color and disparity. λ is set to 0.4 for all our experiments, with a little bias toward the estimated depth D. By re-running Algorithm 1 on the updated image graph, an enhanced segment-tree is constructed and serves as the final structure for cost aggregation and disparity estimation. Note that both the segmentation parameter k and the aggregation parameter σ are closely related to the edge weights. For enhanced ST, we employ a new set of parameters for all the data sets: $k_1 = 1200, \sigma_1 = 0.08$.

The benefits of the joint segmentation are illustrated in Figure 3. For this reference image, segmentation with only color information produces a number of small fragments and incorrectly connected regions (Figure 3(b)). By including an initial disparity map (Figure 3(d)), more consistent scene segmentation results can be achieved (Figure 3(d)), which in turn lead to much improved disparity estimation, especially around depth borders (Figure 3(e)).

3.3. Computational Complexity

Let n be the number of the image pixels, m be the number of the edges ($m = O(n)$), the computational complexity of Algorithm 1 is analyzed stage by stage. The **Initialization** stage requires a sorting step for the edge set. Since the edge weights are encoded with integers, this step can be done in $O(m)$ using counting sort. The **Grouping** stage and the **Linking** stage share similar computation



(a) Reference image (b) Color segmentation (c) Joint segmentation (d) Initial disparity Results (e) Final disparity results

Figure 3. Comparison of color segmentation and color-depth joint segmentation on the Baby2 data set. (a) Reference Image (b) Color segmentation results (c) Color-depth joint segmentation results (d) Initial disparity results computed with ST (e) Final disparity results computed with the enhanced ST. Parameter settings for (b) and (d): $\sigma = 0.1, k = 1200$. Parameter settings for (b) and (d): $\sigma_1 = 0.08, k_1 = 1200$. For (d) and (e), pixels with erroneous disparities are marked in red. Details are best viewed in the electronic version.

costs: they both perform a full scan of the edge set, and for each edge, *find* and *union* operations between different trees are required. These operations can be implemented using a disjoint-set forest with union by rank and path compression techniques [2]. Both stages takes $O(m\alpha(m))$ amortized running time, where $\alpha(m)$ can be approximated as a small constant for practical applications. To sum up, Algorithm 1 runs in amortized time nearly linear to n. Compared to the aggregation process ($O(n \cdot l)$), the ST construction process usually takes a very small part of the total running time, as shown in the experiments.

4. Experimental Results

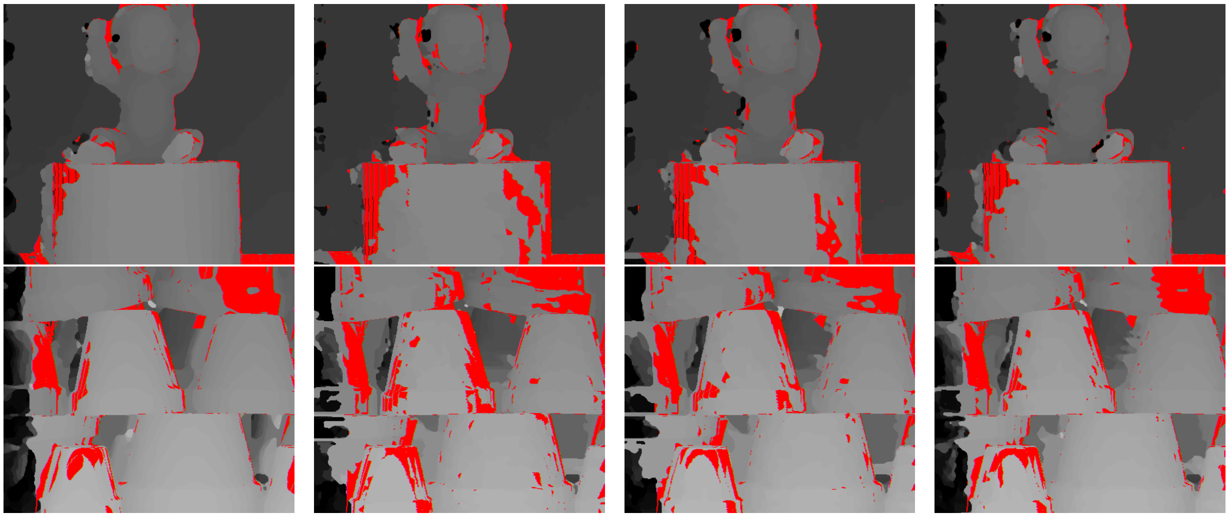
In this section, four cost aggregation methods are evaluated with various stereo data sets: aggregation with ST (denoted as ST-1), aggregation with enhanced ST (denoted as ST-2), aggregation with MST (denoted as MST) [23] and aggregation with guided image filter (denoted as GF) [14]. MST and GF have been reported with leading accuracy and efficiency on the standard Middlebury benchmark [16], and their source codes have been made available by the paper authors. An AD-Gradient measure defined in [14] is used as the matching cost term for the four methods. The parameters for ST-1 and ST-2 are kept constant for all the data sets: $\sigma = 0.1, k = 1200, \lambda = 0.4, \sigma_1 = 0.08, k_1 = 1200$. For MST and GF, their parameters follow the settings of the corresponding papers. Our test platform is a PC equipped with Core2 Quad 2.83GHz CPU and 2GB memories.

We first quantitatively evaluate the aggregation accuracy of the four methods. For each method, the disparity results are computed with the aggregated cost volume and a WTA local strategy. And no post-processing technique is employed. The disparity error rates in non-occlusion regions (non-occlusion errors) are used to evaluate the aggregation accuracy. Different from most previous methods which test the results only with four standard Middlebury data sets (Tsukuba, Venus, Teddy, Cones) [19, 14, 23], we include 15 more data sets for more reliable evaluation.

Data	GF [14]	MST [23]	ST-1	ST-2
Tsukuba	2.28_4	$\mathbf{1.71_1}$	1.89_3	1.84_2
Venus	0.91_4	0.64_2	0.76_3	$\mathbf{0.27_1}$
Teddy	8.30_4	7.14_2	7.55_3	$\mathbf{6.95_1}$
Cones	$\mathbf{2.90_1}$	3.89_4	3.64_3	3.50_2
Aloe	5.02_4	4.46_3	4.15_2	$\mathbf{3.65_1}$
Art	9.21_2	10.54_4	10.51_3	$\mathbf{8.58_1}$
Baby1	$\mathbf{3.71_1}$	8.89_4	7.37_3	4.43_2
Baby2	$\mathbf{4.44_1}$	13.53_4	11.28_3	10.05_2
Baby3	$\mathbf{4.92_1}$	6.37_4	5.36_3	5.19_2
Books	$\mathbf{8.27_1}$	10.10_4	9.05_3	8.40_2
Cloth2	2.64_2	3.61_4	3.15_3	$\mathbf{2.01_1}$
Cloth3	1.82_3	1.95_4	1.58_2	$\mathbf{1.43_1}$
Dolls	4.75_2	5.70_4	5.39_3	$\mathbf{4.42_1}$
Flowerpots	$\mathbf{13.19_1}$	19.21_4	15.73_3	13.31_2
Lampshade1	13.87_4	11.41_3	11.14_2	$\mathbf{9.30_1}$
Laundry	14.94_4	12.92_2	$\mathbf{12.70_1}$	13.69_3
Midd1	42.11_4	30.99_2	$\mathbf{24.92_1}$	31.57_3
Moebius	8.92_4	7.92_2	8.16_3	$\mathbf{7.86_1}$
Wood1	$\mathbf{4.13_1}$	10.13_4	9.51_3	5.81_2
Avg. Error	8.23_3	9.01_4	8.10_2	$\mathbf{7.59_1}$
Avg. Rank	2.53_2	3.21_4	2.63_3	$\mathbf{1.73_1}$

Table 1. Quantitative evaluation of four aggregation methods (GF [14], MST [23], ST-1, ST-2) on 19 Middlebury data sets with error threshold 1. The percentages of the erroneous pixels in non-occlusion regions are used to evaluate the aggregation accuracy of the methods. The subscripts represent the relative rank of the methods on the data sets. ST-1 and ST-2 show competitive performance. ST-1 outperforms MST/GF on 15/9 data sets respectively. ST-2 outperforms all the other methods with lowest average error rate and the highest average ranking. ST-2 produces the most accurate results for 9 data sets.

The quantitative evaluation results are presented in Table 1, which reveal some important characteristics about the performance of the four methods. First, MST does perform better than GF on the standard data sets, but it is less accurate than GF when more data sets are included in the evaluation. GF outperforms MST on 11 data sets, with a much



(a) GF (b) MST (c) ST-1 (d) ST-2

Figure 4. The WTA disparity results of Baby1 and Flowerpots data sets. Erroneous pixels in the non-occlusion regions are marked in red. (a) disparity results computed with GF (b) disparity results computed with MST (c) disparity results computed with ST-1 (d) disparity results computed with ST-2. Details are best viewed in the electronic version.

Algorithm	Avg. Rank	Avg. Error	Tsukuba			Venus			Teddy			Cones		
			nonocc	all	disc	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc
ST-2	32.0	5.35	**1.25**	**1.68**	**6.69**	0.20	**0.30**	1.77	**6.00**	11.9	15.0	2.77	8.82	7.81
ST-1	41.9	5.49	1.47	1.88	6.71	0.24	0.50	2.94	6.08	11.8	14.7	2.76	8.91	7.83
GF [14]	42.7	5.73	1.60	2.14	7.56	**0.10**	0.40	**1.28**	6.96	12.6	16.8	**2.66**	8.89	**7.67**
MST [23]	47.8	5.68	1.57	1.92	8.38	0.29	0.45	3.05	6.11	**11.6**	**14.6**	3.03	**8.60**	8.51

Table 2. Quantitative evaluation of four methods (GF [14], MST [23], ST-1, ST-2) on the standard Middlebury benchmark [16] with error threshold 1. The disparity results are refined with the same post-processing technique. The percentages of the erroneous pixels in non-occ./all/disc. regions are used to evaluate the performance of the method. ST-1 is slightly better than GF, while ST-2 outperforms the other methods with the best overall accuracy.

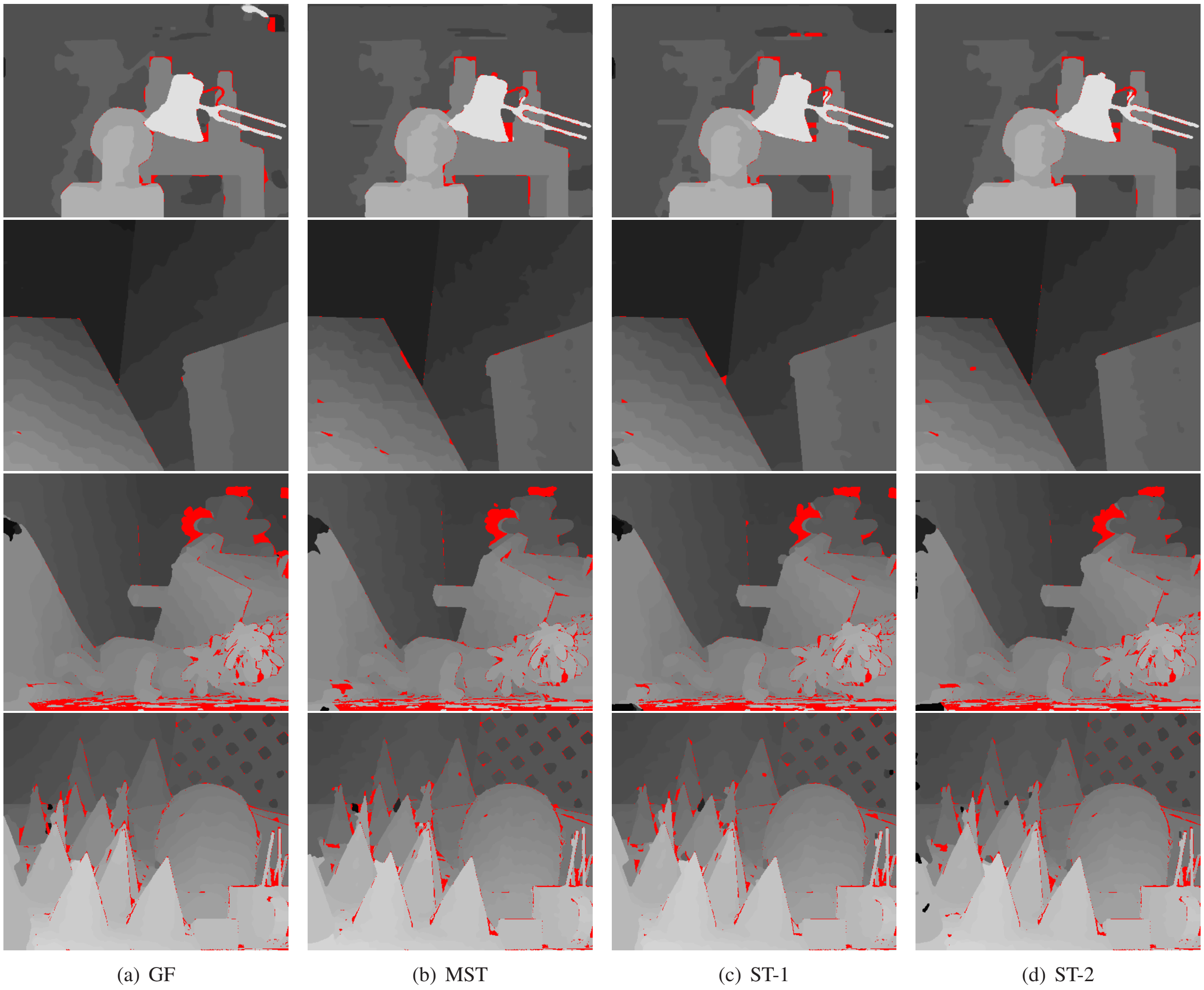
lower average error rate and a higher average ranking. Second, ST-1 is comparable to GF, with a better error rate but a slightly lower average ranking. ST-1 is consistently better than MST, outperforming MST on 15 data sets. Finally, ST-2 outperforms the other methods with the lowest error rate and the highest average ranking. It achieves the most accurate results for 9 data sets. The quantitative results of ST-1 and ST-2 show that the non-local aggregation method benefits greatly from the segmentation information.

For visual comparison, we present the WTA disparity results (without post-processing) of Baby1 and Flowerpots data sets in Figure 4. Compared to MST, ST-1 and ST-2 improve the results with the segmentation information, especially in large textureless regions and around depth boundaries. The complete disparity results can be found in the supplementary material.

We further evaluate the final disparity results of the four methods with the standard Middlebury benchmark [16]. For these data sets, more detailed evaluation on various regions can be performed. An efficient tree-based post-processing technique from [23] is applied to the four methods. The quantitative results are summarized in Table 2. With post-processing, GF performs better than MST, which is consistent with the report in [23]. ST-1 is slightly better than GF, while ST-2 outperforms the other methods with the best overall accuracy. The disparity results of the four methods are presented in Figure 5. Currently (April 2013), ST-2 ranks $15th$ on the Middlebury benchmark.

For the standard Middlebury data sets, the average running time of ST-1, ST-2, MST and GF (re-implemented in C++) are 0.35 seconds, 0.8 seconds, 0.31 seconds and 3.8 seconds respectively. ST-1 runs as efficiently as MST, and it is about $11\times$ faster than GF. ST-2 is about $2\times$ slower than ST-1, since it requires a rough estimation of the depth map and a second segmentation process. The average tree construction time for MST and ST are 21 milliseconds and 34 milliseconds respectively, which take less than 10% of the total running time.

Finally, we test GF, MST and ST-2 on two publicly available, real-world stereo video data sets: a *Book Arrival* se-



(a) GF

(b) MST

(c) ST-1

(d) ST-2

Figure 5. Final disparity results on the standard Middlebury benchmark [16]. Pixels with erroneous disparities are marked in red. (a) disparity results computed with GF (b) disparity results computed with MST (c) disparity results computed with ST-1 (d) disparity results computed with ST-2. Details are best viewed in the electronic version.

quence from FhG-HHI database and an *Ilkay* sequence from Microsoft i2i database. The snapshots for the two video sequences are presented in Figure 6. For both examples, ST-2 performs better than MST and GF with more accurate depth borders and less noise. See the supplementary material for better visual comparison of all the examples.

5. Conclusion

In this paper, we have presented a novel cost aggregation for stereo matching. Our approach is based on a new tree structure, which successfully integrates the segmentation information in a recently proposed non-local aggregation framework [23]. We have also proposed a fast tree construction algorithm and an effective method to update the tree structure. Preliminary results show that this method is very promising: it shows leading aggregation accuracy and speed performance for a number of Middlebury data sets. In the future, we hope to work on several issues. First, we would like to test our method with more challenging outdoor stereo data sets, such as the KITTI Vision Benchmark [5]. Second, we would like to test with various segmentation methods, since the performance of the algorithm is closely related to the segmentation results. Finally, we would like to extend this method to perform more general edge-preserving image processing tasks.

Acknowledgement

This work is funded by National Natural Science Foundation of China (Grant No. 61271430, 61172104, 61201402 and 61202324).

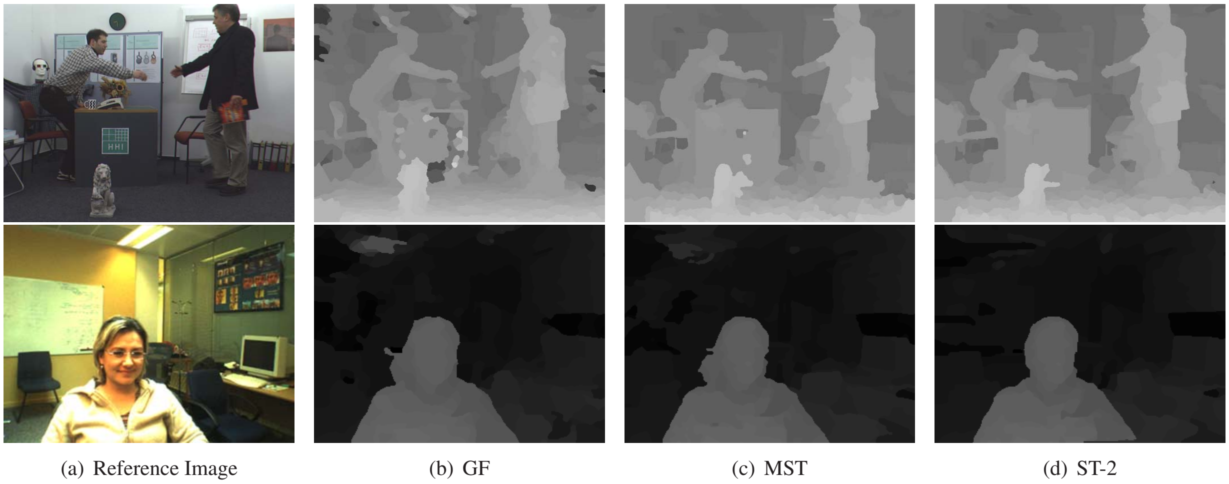


Figure 6. Snapshots of the *Book Arrival* and *Ilkay* stereo video sequences. (a) the reference frame (b) disparity results computed by GF (c) disparity results computed by MST (d) disparity results computed by ST-2. For both examples, ST-2 produces more accurate disparity results than MST and GF near depth borders.

References

[1] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *PAMI*, 24(5):603–619, 2002.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms, 3rd Edition*. MIT Press, 2009.

[3] L. De-Maeztu, S. Mattoccia, A. Villanueva, and R. Cabeza. Linear stereo matching. In *ICCV*, pages 1708–1715, 2011.

[4] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59(2):167–181, 2004.

[5] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, pages 3354–3361, 2012.

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

[7] A. Hosni, M. Bleyer, M. Gelautz, and C. Rhemann. Local stereo matching using geodesic support weights. In *ICIP*, pages 2093–2096, 2009.

[8] A. Klaus, M. Sormann, and K. Karner. Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. In *ICPR*, pages 15–18, 2006.

[9] V. Kolmogorov, A. Criminisi, A. Blake, G. Cross, and C. Rother. Bi-layer segmentation of binocular stereo video. In *CVPR*, pages 407–414, 2005.

[10] C. Lei, J. M. Selzer, and Y.-H. Yang. Region-tree based stereo using dynamic programming optimization. In *CVPR*, pages 2378–2385, 2006.

[11] X. Mei, X. Sun, M. Zhou, S. Jiao, H. Wang, and X. Zhang. On building an accurate stereo matching system on graphics hardware. In *ICCV Workshops*, pages 467–474, 2011.

[12] C. D. Mutto, P. Zanuttigh, and G. M. Cortelazzo. Fusion of geometry and color information for scene segmentation. *IJSTSP*, 6(5):505–521, 2012.

[13] X. Ren. Local grouping for optical flow. In *CVPR*, pages 1–8, 2008.

[14] 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.

[15] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 47(1-3):7–42, 2002.

[16] D. Scharstein and R. Szeliski. Middlebury stereo evaluation, 2012. http://vision.middlebury.edu/stereo/eval/.

[17] J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, 22(8):888–905, 2000.

[18] B. M. Smith, L. Zhang, and H. Jin. Stereo matching with nonparametric smoothness priors in feature space. In *CVPR*, pages 485–492, 2009.

[19] F. Tombari, S. Mattoccia, L. Stefano, and E. Addimanda. Classification and evaluation of cost aggregation methods for stereo correspondence. In *CVPR*, pages 1–8, 2008.

[20] F. Tombari, S. Mattoccia, and L. D. Stefano. Segmentation-based adaptive support for accurate stereo correspondence. In *PSIVT*, pages 427–438, 2007.

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

[22] Z. Wang and Z. Zheng. A region based stereo matching algorithm using cooperative optimization. In *CVPR*, pages 1–8, 2008.

[23] Q. Yang. A non-local cost aggregation method for stereo matching. In *CVPR*, pages 1402–1409, 2012.

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

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