applied optics

3D cost aggregation with multiple minimum spanning trees for stereo matching

Lincheng Li,[1,*] Xin Yu,[2] Shunli Zhang,[3] Xiaolin Zhao,[4] and Li Zhang[1]

[1]Department of Electronic Engineering, Tsinghua University, Beijing 100084, China
[2]College of Engineering and Computer Science, Australian National University, Canberra 2601, Australia
[3]School of Software Engineering, Beijing Jiaotong University, Beijing 100044, China
[4]School of Aeronautics and Astronautics Engineering, Airforce Engineering University, Xi'an 710038, China
*Corresponding author: lilc11@mails.tsinghua.edu.cn

Received 6 February 2017; revised 17 March 2017; accepted 21 March 2017; posted 24 March 2017 (Doc. ID 286217); published 14 April 2017

Cost aggregation is one of the key steps in the stereo matching problem. In order to improve aggregation accuracy, we propose a cost-aggregation method that can embed minimum spanning tree (MST)-based support region filtering into PatchMatch 3D label search rather than aggregating on fixed size patches. However, directly combining PatchMatch label search and MST filtering is not straightforward, due to the extremely high complexity. Thus, we develop multiple MST structures for cost aggregation on plenty of 3D labels, and design the tree-level random search strategy to find possible 3D labels of each pixel. Extensive experiments show that our method reaches higher accuracy than the other existing cost-aggregation and global-optimization methods such as the 1D MST, the PatchMatch and the PatchMatch Filter, and currently ranks first on the Middlebury 3.0 benchmark.

OCIS codes: (330.1400) Vision - binocular and stereopsis; (100.2000) Digital image processing; (100.6890) Three-dimensional image processing; (110.3010) Image reconstruction techniques.

https://doi.org/10.1364/AO.56.003411

1. INTRODUCTION

Stereo matching has long been one of the core problems in computer vision [1,2]. According to the categories in [3], most stereo-matching algorithms take the following four steps or a subset of them: matching cost computation, cost (support) aggregation, disparity computation/optimization, and disparity refinement. Among the four steps, cost aggregation is one of the crucial steps for most local and global algorithms [4–6].

Traditional 1D cost-aggregation methods [7,8] implicitly assume that all pixels inside the support region have the same disparity, which is not true in most cases. For more accurate modeling, instead of assigning a discrete disparity to each pixel, 3D cost-aggregation methods [9,10] assign three continuous values to each pixel, representing the disparity and the surface normal direction simultaneously. The support region of a pixel in 3D cost aggregation is properly decided by the disparity and the normal direction together. Hence, 3D cost-aggregation methods are more reasonable and achieve better results, especially at subpixel accuracy. Meanwhile, 3D cost aggregation also faces the challenge that it is impossible to traverse the infinite 3D label space $\mathbb{R}^3$ to find the best one. Early methods perform oversegmentation and generate limited 3D labels at the superpixel level [11–13], but the number of labels is insufficient in most cases.

A group of methods borrows the idea of PatchMatch [14–16] to enrich the candidate label set. Starting from assigning a random 3D label to each pixel, PatchMatch-based methods iteratively perform spatial propagation between neighboring pixels and random refinement at each pixel to find the 3D labels with lower aggregated costs. Benefiting from the random search strategy, the approximate best 3D labels of most pixels are directly located with high efficiency. Although PatchMatch-based methods can solve the problem of searching 3D labels well, the support region of each 3D label is a fixed small patch due to high computational complexity. This may lead to degradation at some challenging image regions, e.g., low-texture surfaces and thin objects.

On the other hand, tree-based support region filtering [17–19] avoids the locality of fixed size patches in 1D disparity cases. A pixel-level minimum spanning tree (MST) is first constructed from the reference image grid; then the matching costs are aggregated along the tree edges from all pixels in the image. Since the support region of each pixel is the entire image, MST-based cost aggregation outperforms the fixed size adaptive weight cost aggregation that is used by PatchMatch in 1D disparity cases.

However, it is not straightforward to combine MST-based support region filtering and the 3D label search strategy of PatchMatch due to high computational complexity. Given the

maximum disparity size D and the number of pixels N, the computational complexity of the naive combination is at least $O(N^2 \log D)$, which is unacceptable in most applications. For instance, for high-resolution images with the number of pixels N larger than 10^6, the computational complexity is at least 10^{13}.

In this paper, we propose what we believe is a novel cost-aggregation method that successfully weaves together MST-based support region filtering and PatchMatch label search with low computational complexity. For cost aggregation with a given 3D label, we propose the MST-based tree filtering that works on 3D labels for better support regions. The aggregated costs with the same 3D label at all pixels in the MST can be calculated simultaneously with reusable intermediate results. To reduce the huge computational complexity caused by the large amount of 3D labels, we introduce a multiple MST structure instead of one single MST for aggregated cost computation. The multiple MSTs are constructed based on the tree sizes and maximum edge weights, so that the aggregated cost computation on each smaller MST is a good approximation of that on the entire image. With similar accuracy at each pixel, both unnecessary aggregation for distant pixels and redundant support regions are avoided. To efficiently construct the possible 3D labels of each pixel, we generalize the PatchMatch random search strategy to apply on the multiple MSTs. Possible labels are iteratively random refined at each MST and propagated between neighboring MSTs. Although the 3D label search is performed at tree level, the best 3D label with the lowest aggregated cost is assigned at each pixel independently. As a result, our method reduces the computational complexity from $O(N^2 \log D)$ to $O(N \log D)$, which is one of the lowest among related methods.

Figure 1 shows an example of the support regions of different cost-aggregation methods in 3D space. Existing MST-based cost aggregation locates the proper support region sizes for pixel P and Q in the reference image, but the fronto-parallel support regions in 3D space do not align with actual object surfaces. PatchMatch-based cost aggregation finds the correct support region positions and directions in 3D space, but the size of support regions is fixed and not optimal. In contrast, by weaving together MST-based support region divisions and PatchMatch label search strategy, our cost-aggregation method (denoted as 3D MST) finds both the proper support region size and the best 3D position, thereby achieving the best stereo-matching accuracy.

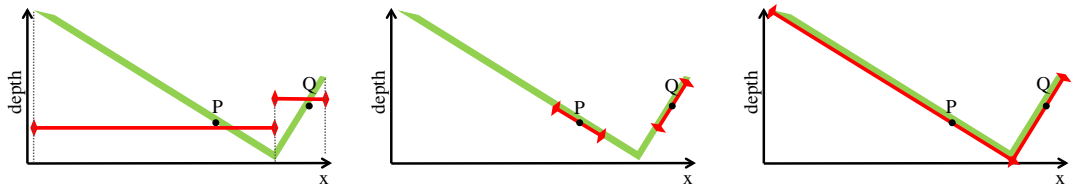
Our contributions are mainly fourfold:

- We propose an efficient 3D label cost-aggregation method with MST-based filtering and random label search.
- We propose the 3D label tree filtering to efficiently aggregate costs on the 3D MST.
- We introduce the multiple MST structure to further accelerate the 3D label cost aggregation.
- We propose the MST-based label search strategy that generalizes the original PatchMatch random search.

2. RELATED WORKS

PatchMatch [14] was first proposed for approximate nearest-neighbor matching. The random search strategy was invented to locate the 2D displacement of each pixel from the huge label space. PatchMatch Stereo (PM) [15] employs the random sampling and spatial propagation strategy for 3D label search in the stereo-matching problem. For each label, adaptive support weight [20] is used to aggregate costs in local windows. However, the computational complexity grows huge quickly for high-resolution images or large window sizes, and the fixed-size windows cannot handle noisy and low-texture regions well. To speed up the original algorithm, PatchMatch Filter (PMF) [16] employs the edge-aware filtering to remove the redundant computation of adaptive weight cost aggregation, but the failure at low-texture regions is not well solved even with global information from the image pyramid. To compensate for the locality of fixed-size windows at challenging regions, later methods added expensive global optimization with belief propagation [21,22] or graph cut [23,24] to enforce spatial smoothness. Due to the infinite possible 3D label space, the global-optimization methods on 3D labels are much more complicated than those on 1D disparities. Instead of using the fixed-size small-support region, our method derives the support region from the MST, which is large enough to overcome the matching ambiguity in most cases. Hereby, our method achieves better performance at low-texture regions, and avoids the expensive global optimization step. Furthermore, compared to the brute force summation in each fixed-size window independently, the simultaneous computation at all pixels by tree filtering is much faster.

Yang *et al.* [17] first introduced the MST structure for non-local 1D cost aggregation, and designed tree filtering to reduce the complexity. To regularize the shape of the MST, Mei *et al.* [18] employed graph-based segmentation and proposed a two-step segment tree as a better MST structure. For better spatial consistency, Vu *et al.* [19] constructed two MSTs in pixel level and superpixel level, respectively, and combined the two costs to estimate the final disparity. However, the above methods only apply MST-based cost aggregation on integer disparities instead of 3D labels, and therefore have less accurate support regions in 3D and lower subpixel accuracy. To get rid of the



(a) 1D MST (b) PatchMatch (c) 3D MST

Fig. 1. Support regions of different cost-aggregation methods in 3D space (top view). (a) Support regions of 1D MST. (b) Support regions of PatchMatch. (c) Support regions of 3D MST. P and Q are two points on the green object surface. Red segments are the 3D positions of the corresponding support regions of different methods for P and Q.

inaccurate fronto-parallel assumption, Krishna *et al.* [25] generated 3D labels from segmentation and performed MST-based cost aggregation on them, but the small 3D label set is far from sufficient due to high computational complexity. Psota *et al.* [26] gave up cost aggregation and employed MST structure for global optimization by message passing, but the accuracy was still restricted by the integer disparity labels. In contrast, benefiting from the advanced 3D labels, our method achieves better accuracy than the 1D MST-based methods. Meanwhile, our method generates and tests sufficient possible 3D labels for each pixel with low computational complexity. As a result, our method outperforms all existing MST-based methods in stereo-matching accuracy.

3. PROPOSED METHOD

Given a pair of input images, the goal of our method is to assign a 3D label $l = (a, b, c)^T \in \mathbb{R}^3$ to each pixel $p = (p_x, p_y)$ of the reference image. The disparity of the pixel is then computed by $d_p = ap_x + bp_y + c$. We wish to find the 3D label with the minimum aggregated cost for every pixel. The 3D cost-aggregation strategy and the 3D label search strategy are discussed in the following subsections.

A. 3D Cost Aggregation with MST

As shown in Fig. 2, the reference image is regarded as a grid undirected graph. Each pixel is a node, and each four-neighboring pixel pair is connected with an edge. The weight of the edge connecting pixel p and q is defined as

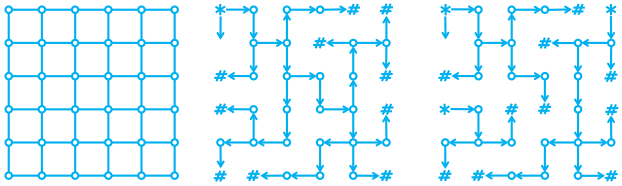
$$w(p, q) = w(q, p) = \|I(p) - I(q)\|_1, \tag{1}$$

where $\|I(p) - I(q)\|_1$ is the l_1 distance between the colors of pixel p and q in the reference image. With the preceding edge weights, an MST is extracted from the image grid. In the constructed tree structure, for any two nonadjacent pixels p and q, the edge weight $w(p, q)$ between them is defined as the sum of edge weights along their path. Correspondingly, in the cost-aggregation process along the MST, the support weight between any two pixels p and q is defined as

$$S(p, q) = S(q, p) = \exp(-w(p, q)/\gamma), \tag{2}$$

where parameter γ controls the support weight strength.

On the MST structure, instead of aggregating costs with 1D disparities, we explore the MST-based cost aggregation with 3D labels. For pixel p with 3D label l, the aggregated cost is computed by



(a) Pixel grid (b) MST (c) Multiple MSTs

Fig. 2. Visual illustration of the multiple MST structures in a 6 × 6 image. (a) Original image grid. The small circles stand for pixels; (b) single MST constructed from (a). "∗" stands for the root node and "#" stands for leaf nodes. Arrows show the order from the root node to the leaf nodes; (c) multiple MSTs constructed from (a).

$$C^A(p, l) = \sum_{q \in T} S(q, p) \min(C(q, l), \tau_{\text{sim}}), \tag{3}$$

where costs of all pixels q in the tree T are aggregated with different support weights. Parameter τ_{sim} is introduced to truncate pixel costs for more robustness. $C(q, l)$ is the computed pixel similarity cost between pixel q and the corresponding pixel in the other view with disparity $aq_x + bq_y + c$. Many similarity costs can be embedded here. We employ the state-of-the-art similarity cost predicted by a neural network [27], written as

$$C(q, l) = C_{\text{CNN}}(R^L(q_x, q_y), R^R(q_x - (aq_x + bq_y + c), q_y). \tag{4}$$

C_{CNN} is the predicted similarity cost between the 11 × 11 image patch R^L centered at pixel q of the left view and the image patch R^R centered at the corresponding pixel at the right view.

The brute force computation of Eq. (3) scans the entire tree to compute the aggregated cost for every pixel with every 3D label, which has an extremely high complexity. Instead, we propose the two-step tree filtering strategy inspired by [17]. The computational complexity is reduced by reusing intermediate results.

Given a 3D label l, in the first step, we scan the MST from the leaf nodes to the root node, and calculate the temporal aggregated costs $C^{A\uparrow}(p, l)$ by

$$C^{A\uparrow}(p, l) = C(p, l) + \sum_{q \in Ch(p)} S(q, p) C^{A\uparrow}(q, l), \tag{5}$$

where $Ch(p)$ is the set of children nodes of pixel p. At the leaf nodes, $C^{A\uparrow}(q, l)$ is set to $C(q, l)$; then $C^{A\uparrow}$ at the other nodes can be calculated once their children are all visited. In the second step, the final aggregated costs are computed from the root node to the leaf nodes by

$$\begin{aligned} C^A(p, l) &= C^{A\uparrow}(p, l) + S(Pa(p), p)[C^A(Pa(p), l) \\ &\quad - S(Pa(p), p) C^{A\uparrow}(p, l)] \\ &= S(Pa(p), p) C^A(Pa(p), l) \\ &\quad + [1 - S(Pa(p), p)^2] C^{A\uparrow}(p, l), \end{aligned} \tag{6}$$

where $Pa(p)$ is the parent node of pixel p.

After the above leaf-to-root scan and root-to-leaf scan, each pixel is visited only twice, while the aggregated costs of all pixels along the entire MST are correctly obtained. The efficiency of the 1D MST tree filtering is maintained in the 3D label case here.

B. Multiple MST Structures

Even with the employment of the two-step tree filtering strategy, the computational complexity is still too high when the number of 3D labels is huge. The redundant computation exists in two aspects in the 3D label case. First, given a 3D label l, it is unnecessary to simultaneously compute $C^A(p, l)$ in Eq. (3) for all pixels of the image. Unlike the 1D disparity case, two distant pixels are not quite likely to share the same 3D label, since the long distance enlarges a tiny error in normal direction into a huge error in depth. Second, it is unnecessary to scan the entire image to compute $C^A(p, l)$ for pixel p. Once $S(q, p)$ is small enough, pixels on the far side of q in the tree can be

ignored, as their support weights are sure to be smaller than $S(q, p)$.

To remove the above redundant computations, we perform 3D cost aggregation on multiple smaller trees instead of one single tree for faster speed and similar accuracy. If the multiple tree structure is well designed, on one hand, the simultaneously aggregated costs calculation with the same 3D label is only applied on pixels inside the smaller tree; on the other hand, the aggregated cost along the smaller tree is a good approximation of that along the single MST.

We employ the strategy of [28] to construct the multiple tree structure. On initialization, each pixel is regarded as a tree. All edges $w(p, q)$ of the pixel grid are then visited in weight-increasing order. The two trees connected by $w(p, q)$ are merged if

$$w(p, q) \le \min(\mathrm{Int}(T_p) + \tau(T_p), \mathrm{Int}(T_q) + \tau(T_q)). \quad (7)$$

$\mathrm{Int}(T)$ in Eq. (7) controls the maximum edge weight in tree T, and is defined as

$$\mathrm{Int}(T) = \max_{e \in T}(w(e)). \quad (8)$$

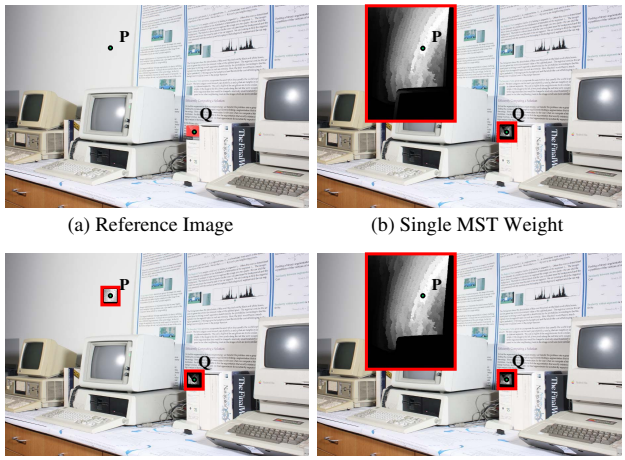
$\tau(T)$ in Eq. (7) controls the size of tree T, and is defined as

$$\tau(T) = \lambda / |T|, \quad (9)$$

where λ is the balancing parameter, and $|T|$ is the size of tree T. By the sequential visit of all edges, pixels belonging to different trees are connected into K clusters. For each tree, although selecting different pixels as the root node leads to different parent-child structures and different computation orders, the final aggregated costs are always the same. Hence, we directly select the upper-left pixel of each T_i as the root node of that tree. Given the K root nodes, the multiple tree structure $\{T_i\}$ of K trees is constructed, as shown in Fig. 2. Each tree T_i is the MST of the corresponding pixel region. After the construction of $\{T_i\}$, cost aggregation of 3D labels is carried out in each tree T_i, respectively, by Eqs. (5) and (6), instead of in the single MST of the entire image.

Although the multiple MST construction strategy is first proposed for image segmentation in [28], it meets the requirements of our 3D label cost aggregation well. In the cost-aggregation process, when the path is long, the support weights behind a relatively large weight edge are more likely to be so small that it is fine to abandon the connected pixels. Correspondingly, during the tree construction process, when a tree grows larger, it is less likely to accept a new edge with relatively large weight. As a result, the multiple MST structure turns out to be a good approximation of the single MST when computing Eq. (3). To the contrary, common segmentation algorithms like SLIC [29] or Meanshift [30] do not have similar properties, as they rely on color and coordinate distance instead of MST path distance for clustering. In their segmentation results, pixels in different superpixels are quite likely to have a short distance in the single MST.

Figure 3 shows the support regions and weights of two pixels as an example. The fixed window size of PatchMatch is too small to avoid matching ambiguity for pixel P in a low texture region, even though it is set to the best window size (61×61) for overall performance. In contrast, the support region generated by MST-based filtering is huge for pixel P and is appropriate for pixel Q, and therefore is superior to the fixed-size windows. As an approximation of the single MST, our multiple MST structure generates similar or even slightly better support regions and weights in both pixels, but has much lower computational complexity.



(a) Reference Image (b) Single MST Weight

(c) PatchMatch Weight (d) Multiple MST Weight

Fig. 3. Support regions and weights of different cost-aggregation methods at pixel P and Q in "Vintage." Support regions and weights for P and Q are shown in the red rectangles and are overlaid on the reference image in (b)–(d). Note that the support regions of (b) and (d) are not limited to the red rectangles. We ignore outer regions with tiny weights for better display.

C. Random Search Strategy

Although cost aggregation with multiple MSTs is more efficient, it is impossible to traverse the infinite 3D label space to find the best label. An intuition is that pixels in the same tree or nearby trees are more likely to share the same label. Inspired by the idea of PatchMatch, we design a random search strategy to find and test possible best labels from the infinite label space.

On initialization, we assign a random 3D label l_i to each tree T_i. In order to generate uniformly distributed random labels, we first generate a random unit normal vector $n_0 = (n_x, n_y, n_z)^T$ and a random disparity $d_0 \in [0, D]$ for each tree T_i, where D is the known maximum disparity of the image. We then pick a random pixel p in T_i. The 3D label $l_i = (a_i, b_i, c_i)^T$ is computed by $a_i = -n_x/n_z$, $b_i = -n_y/n_z$, $c_i = (n_x p_x + n_y p_y + n_z d_0)/n_z$. The current labels of all pixels in T_i are set to l_i. After the random initialization, the spatial propagation step and the random refinement step are iteratively carried out to update the 3D label of each pixel.

In the spatial propagation step, each tree is processed sequentially in scan order. When processing tree T_i, we randomly choose one pixel q from each of its neighbor trees. Two trees are defined as neighbors if and only if two pixels from each of them are connected on the four-neighbor pixel grid. The current 3D labels of these pixels consist of the candidate label set P_i, which is then tested on T_i. Aggregated costs of each new label $l_q \in P_i$ are calculated by Eqs. (5) and (6) for all pixels in tree T_i. The current label l_p of any pixel p is replaced with the new label l_q if the aggregated cost $C^A(p, l_q) < C^A(p, l_p)$.

In the random refinement step, we add decreasing random values to current best labels. For each tree T_i, we pick a random pixel p again, and acquire the current best label. l_p. l_p is first converted back to the normal and disparity representation by $d_0 = a_p p_x + b_p p_y + c_p$, $n_z = 1/(a_p^2 + b_p^2 + 1)^{1/2}$, $n_x = -a_p n_z$, $n_y = -b_p n_z$; then we iteratively add three random values in $[-\Delta_n^{\max}, \Delta_n^{\max}]$ to n_0, and one random value in $[-\Delta_d^{\max}, \Delta_d^{\max}]$ to d_0. We set $\Delta_n^{\max} = 1$, $\Delta_d^{\max} = D/2$ in the first iteration, and reduce them by half after each iteration. The iteration stops when $\Delta_d^{\max} \le 0.1$. The corresponding new labels consist the refined label set R_i. Similar to the spatial propagation step, we perform cost aggregation for the labels in R_i on T_i, and update the current best labels of the pixels if their new costs are lower.

The overall algorithm is summarized in Algorithm 1. Note that instead of being a simplified presentation of the 3D surfaces, our multiple tree structure is only designed to speed up the cost-aggregation process. Although the random search strategy generates and tests new 3D labels in tree level, the best labels are updated at each pixel individually. Hence, the algorithm does not suffer from imperfect tree construction, in contrast to the segment-based methods that often suffer from inaccurate segmentation.

Algorithm 1: ***Pipeline of 3D MST Algorithm***

Input: A pair of stereo images I and I'
Output: The best 3D labels of each pixel
1: Construct multiple MST structures $\{T_i\}$.
2: Generate a random 3D label l_i to each T_i.
3: **While** max iteration time is not reached **do**
4: **for all** T_i **do**
5: Construct P_i from neighbor trees
6: **for all** $l \in P_i$ **do**
7: Aggregate costs by Eqs. (5) and (6).
8: Update pixel labels with lower costs.
9: Construct R_i from a random pixel label in T_i
10: **for all** $l \in R_i$ **do**
11: Aggregate costs by Eqs. (5) and (6).
12: Update pixel labels with lower costs.

D. Postprocessing and Complexity

To obtain the final result, we run our algorithm on both left and right views independently; then the custom left-right consistency check is employed to find inconsistent pixels with $|d_p^L - d_{p'}^R| > 1$. These pixels are refilled with the left or right nearest consistent 3D label that corresponds to the smaller disparity. No other postprocessing is conducted.

Given the input images of size N and the level of disparities D, the time and space complexity can be derived as follows. The construction of the multiple MST structures takes $O(N\alpha(N))$ time, where $\alpha(N)$ is the inverse Ackermann function and is less than 5 for any practical N, as discussed in [28]. For each MST T_i with the size N_i, a total number of $O(\log D)$ labels are tested for cost aggregation in each iteration, and the expected number of iterations for convergence is a small number approximately independent to N [14]. Hence, the complexity for the single MST is $O(N_i \log D)$, and the overall time complexity for an image pair is $\sum_i O(N_i \log D) = O(N \log D)$.

Table 1. Time and Space Complexity of Different Methods

	1D MST	PM	PMF	3D MST
Time	$O(ND)$	$O(NW \log D)$	$O(N \log D)$	$O(N \log D)$
Memory	$O(ND)$	$O(N)$	$O(N)$	$O(N)$

The storage of the multiple MST structures takes $O(N)$ memory. When filtering with 3D labels, each T_i takes $O(N_i)$ to store the current best labels and aggregated costs. As a result, the overall memory cost is $O(N)$ for our 3D cost-aggregation method.

Table 1 shows the time and space complexity of our method, together with 1D MST [17], PM [15], and PMF [16]. W is the size of the support window for PM, which is set to 61×61 in our experiments. Recently PMF removed W in the time complexity as the key contribution. While combining PatchMatch 3D label search and MST-based support region filtering for better accuracy, our method still has the lowest computational complexity.

4. EXPERIMENTS

A. Parameter Setting

We evaluate our method against the Middlebury 3.0 benchmark [31], consisting of 30 high-resolution (about 3000×2000) image pairs, the Middlebury 2006 data set [32] consisting of 21 low-resolution (about 450×350) image pairs, together with the KITTI 2015 benchmark [33] consisting of 200 image pairs. First, we evaluate our ranks on the official Middlebury 3.0 benchmark to compare with other top-performing methods. Next, we compare our method with related MST-based and PatchMatch-based cost-aggregation methods on the Middlebury 2006 data set. Then, we evaluate our method against the KITTI 2015 benchmark to show the robustness of handling outdoor scenes. Finally, we analyze the sensitivity to some key parameters and running time.

We run the experiments on a PC equipped with an i7-6700K 4.0GHz CPU, 40G memory, and a GTX TITAN X graphics card. When computing the pixel costs, we set the truncation threshold of CNN similarity cost τ_{sim} to 0.5. Since the image size of Middlebury 3.0 is about 40 times larger than that of the Middlebury 2006 data set, two parameters are set to different values on the two data sets. The parameter γ to control the support weights is set to 50 on the Middlebury 3.0 benchmark and 8 for the Middlebury 2006 data set, while the parameter λ to control the tree sizes is set to 10,000 and 3000, respectively. We iterate 50 times for propagation and refinement. The parameters stay constant in each data set.

B. Evaluation Based on Middlebury 3.0 Benchmark

The overall performance of our method is tested against the Middlebury 3.0 benchmark. As the upgrade version of Middlebury 2.0 benchmark (including Tsukuba, Venus, Teddy, and Cones), it consists of 30 high-resolution image pairs with different challenging conditions. To prevent excessive parameter tuning, the 30 image pairs are divided into the training set and the test set evenly with ground truth disparities of the test set hidden. Results of the test set are only allowed to be

uploaded one time for online evaluation. Although our method is fast enough to run at full-resolution images on the benchmark, we test our method at half resolution (about 1500 × 1000) for fair comparison, which is the same as all other top-performing algorithms.

Table 2 shows the ranks of the top 10 methods [34,35] on the test set under the default criterion "bad 2.0." Since all the top-performing methods use half-resolution images as inputs, while the evaluation is carried out with full-resolution ground truth, "bad 2.0" is equivalent to error threshold 1 pixel for the listed methods. The final ranks are based on the weighted average error rates of the 15 image pairs. Our method (denoted as 3D MST) currently ranks first on the table. As a simple cost-aggregation method without global optimization, our method not only outperforms all other cost-aggregation methods, but also outperforms all global- optimization methods. Table 3 shows the results on the test set with criterion "bad 1.0," which is equivalent to error threshold 0.5 pixel for the listed methods. Our method still ranks first. The results demonstrate that our method obtains both the best pixel level accuracy and the best subpixel level accuracy, and is robust against different challenging situations. Besides the two error thresholds, our method also ranks first under almost all other criteria in both the test set and the training set. More detailed results can be viewed on the benchmark website [36].

Figure 4 shows the comparison of results from our method and that of MC-CNN [27] and TMAP [26]. MC-CNN uses the same pixel similarity cost as ours, while TMAP employs MST for global optimization. Compared to the two methods, our cost-aggregation strategy works much better on the challenging textureless regions, e.g., desk and chair surfaces in the second row and left wall in the last row, which most existing global methods cannot handle well.

C. Evaluation Based on Middlebury 2006 Data Set

We then compare our method with some related MST-based and PatchMatch-based cost-aggregation methods, including 1D MST [17], ST [18], PatchMatch [15], SPM-BP [22] and PMF [16]. Since the parameters and strategies of the above methods are well tuned for low-resolution images, it is unfair to test them directly against the Middlebury 3.0 benchmark. Besides, the Middlebury 2.0 benchmark is almost solved and no longer active. As a result, we run the corresponding experiments on the Middlebury 2006 data set, which consists of 21 low-resolution image pairs.

Table 4 shows the error rates of 1D MST, ST, PatchMatch, SPM-BP, and our method on all image pairs of the Middlebury 2006 data set, with an error threshold of 1.0 pixel on the non-occluded regions. To avoid the influence of different matching costs, we replace the original similarity cost of 1D MST with our neural network predicted cost, and denote the method as "MST + CNN." The source code of ST is provided by the authors. SPM-BP offers the executable file for comparison, while PatchMatch is our own implementation. Our method obtains the lowest error rates on 20 of the 21 image pairs, and has the lowest average error rate. On the image pairs that contain large

Table 2. Top 10 Algorithms on the Test Set of Middlebury 3.0 Benchmark under Default Criterion "bad 2.0" (Equivalent to Error Threshold 1 pixel) by the Time of Submission[a]

Name	Avg	Austr	AustrP	Bicyc2	Class	ClassE	Compu	Crusa	CrusaP	Djemb	DjembL	Hoops	Livgrm	Nkuba	Plants	Stairs
3D MST	**5.92**	3.71	2.78	**4.75**	2.72	7.36	**4.28**	**3.44**	**3.76**	**2.35**	12.6	11.5	8.56	14.0	5.35	8.87
PMSC	6.71	**3.46**	**2.68**	6.19	**2.54**	**6.92**	4.54	3.96	4.04	2.37	13.1	12.3	12.2	16.2	5.88	10.8
LW-CNN	7.04	4.65	3.95	5.30	2.63	11.2	5.41	4.32	4.22	2.43	12.2	13.4	13.6	14.8	**4.72**	12.0
MeshStereoExt	7.08	4.41	3.98	5.40	3.17	10.0	6.23	4.62	4.77	3.49	12.7	12.4	10.4	14.5	7.80	8.85
APAP-stereo	7.26	5.43	4.91	5.11	5.17	21.6	6.99	4.31	4.23	3.24	14.3	**9.78**	**7.32**	13.4	6.30	**8.46**
NTDE	7.44	5.72	4.36	5.92	2.83	10.4	5.71	5.30	5.54	2.40	13.5	14.1	12.6	13.9	6.39	12.2
MC-CNN-acrt	8.08	5.59	4.55	5.96	2.83	11.4	5.81	8.32	8.89	2.71	16.3	14.1	13.2	13.0	6.40	11.1
MC-CNN + RBS	8.42	6.05	5.16	6.24	3.27	11.1	6.36	8.87	9.83	3.21	15.1	15.9	12.8	13.5	7.04	9.99
SNP-RSM	8.75	5.46	4.85	6.50	3.37	10.4	7.31	8.73	9.37	3.58	14.3	14.7	14.9	**12.8**	10.1	10.8
MC-CNN_layout	8.94	5.53	5.63	5.06	3.59	12.6	7.23	7.53	8.86	5.79	23.0	13.6	15.0	14.7	5.85	10.4

[a]Best results are shown in bold. Average errors are listed in the second column.

Table 3. Top 10 Algorithms on the Test Set of Middlebury 3.0 Benchmark under Criterion "bad 1.0" (equivalent to Error Threshold 0.5 Pixel) by the Time of Submission[a]

Name	Avg	Austr	AustrP	Bicyc2	Class	ClassE	Compu	Crusa	CrusaP	Djemb	DjembL	Hoops	Livgrm	Nkuba	Plants	Stairs
3D MST	**14.5**	8.97	7.03	10.0	9.18	23.0	**11.1**	**13.7**	14.9	8.53	24.2	23.2	**17.1**	28.0	14.1	**15.7**
PMSC	14.8	**7.92**	5.88	10.6	8.99	22.6	12.1	13.8	14.6	8.17	23.9	24.0	19.8	27.6	15.2	18.6
LW-CNN	14.9	8.47	6.89	**9.47**	8.15	27.6	12.8	16.6	16.0	5.88	22.0	25.9	21.4	22.7	**12.8**	24.4
MeshStereoExt	15.6	9.29	7.97	11.9	9.96	27.2	14.4	13.9	15.0	10.1	24.5	23.9	17.2	26.1	17.7	17.0
NTDE	16.2	12.7	8.51	10.1	7.78	26.1	13.2	20.7	20.4	**5.43**	24.3	26.2	20.3	23.4	15.7	23.7
MC-CNN-acrt	17.1	10.4	8.35	10.1	**7.71**	23.4	13.6	25.0	25.9	6.80	24.8	29.0	21.8	22.5	17.1	22.0
MC-CNN_layout	17.8	13.7	10.9	10.9	8.22	26.3	15.6	20.2	22.9	10.2	31.3	30.8	22.3	24.1	15.5	20.4
MC-CNN-fst	18.0	14.3	8.51	12.3	11.0	29.4	16.3	22.5	20.8	6.83	29.1	30.4	22.8	24.1	17.3	22.1
SNP-RSM	18.0	10.8	8.94	11.4	8.39	24.7	15.7	24.5	25.9	8.47	24.4	29.9	23.1	**22.5**	20.6	21.9
MC-CNN+RBS	18.1	11.5	9.89	10.7	8.89	24.4	14.1	27.1	28.2	7.81	24.8	30.6	22.0	22.8	17.8	21.6

[a]Best results are shown in bold. Average errors are listed in the second column.

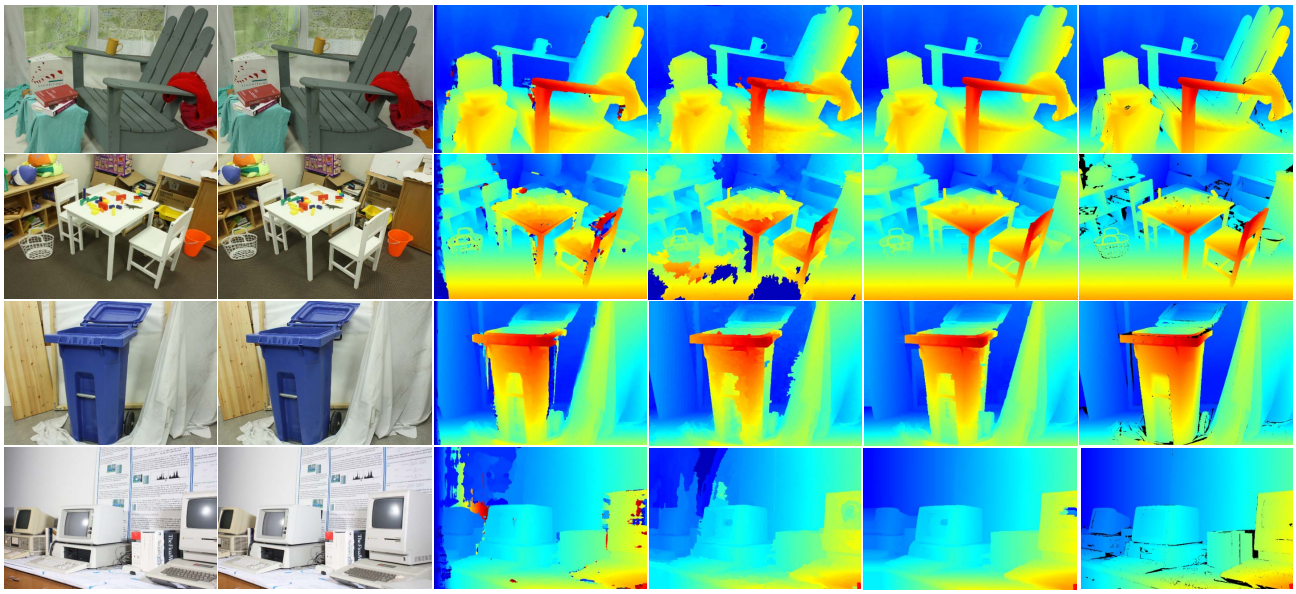


Fig. 4. Comparison of disparity maps generated by MC-CNN, TMAP, 3D MST, together with the ground truth.

textureless regions, the error rates of our method are much lower than that of MST-based and PatchMatch-based cost-aggregation methods. When aggregating on the same neural network predicted matching cost, our method generates much more accurate results than that of MST + CNN. Even though SPM-BP employs global optimization to regularize results of PatchMatch-based cost aggregation, their results are still less accurate than ours without global optimization.

Figure 5 shows some results of our method, together with 1D MST, PatchMatch, and PMF. Error pixels with an error threshold of 1.0 pixel are marked in red. Results of PMF are provided by the authors. Error pixels of our method are much less than those of the other methods, especially in textureless regions, e.g., the book in the first row and the ball in the second row. Note that the last row contains two objects that clearly demonstrate the key advantage of our method. The yellow object is a fronto-parallel plane with no texture at all. MST-based cost aggregation can handle it well, while PatchMatch-based methods do not have large enough support regions to find the correct disparities. Even though PMF uses global information to deal with textureless regions by employing the image pyramid, it still fails on part of that object. The white object, in contrast, contains slanted surfaces with weak texture. 1D MST fails on the slanted parts because of the implicit fronto-parallel assumption, while PatchMatch-based methods handle it well with the help of 3D labels. By extending MST-based cost aggregation into 3D label space, our method works perfectly on both objects and has the best performance on the entire data set.

Table 4. Error Rates with Threshold 1.0 Pixel on Nonoccluded Regions of Middlebury 2006 Data Set[a]

Image	MST + CNN	ST	PM	SPM-BP	3D MST
Aloe	5.12	5.03	3.78	6.75	**1.88**
Baby1	9.32	4.45	2.15	3.27	**1.17**
Baby2	17.6	16.9	2.87	3.97	**0.88**
Baby3	4.96	4.54	3.22	3.82	**1.75**
Bowling1	17.1	16.5	5.30	12.1	**1.24**
Bowling2	9.37	10.6	2.74	5.27	**1.56**
Cloth1	0.53	0.68	0.89	1.17	**0.14**
Cloth2	4.58	4.30	2.81	4.52	**0.76**
Cloth3	2.08	2.54	1.91	2.15	**0.80**
Cloth4	1.58	1.70	1.78	2.20	**0.61**
Flowerpots	12.0	12.5	5.56	8.80	**2.74**
Lampshade1	9.07	10.9	9.70	8.67	**1.07**
Lampshade2	9.62	16.2	11.1	17.2	**1.03**
Midd1	6.72	29.1	37.1	37.4	**7.46**
Midd2	6.45	31.3	34.9	33.2	**6.62**
Monopoly	**4.37**	24.9	19.6	32.7	6.15
Plastic	29.4	39.3	42.2	35.2	**5.05**
Rocks1	3.04	2.96	2.35	2.60	**0.84**
Rocks2	2.17	2.38	1.64	1.68	**0.85**
Wood1	10.7	4.66	0.87	4.19	**0.68**
Wood2	1.50	2.38	0.56	**0.49**	0.50
Average	7.97	11.61	9.19	10.83	**2.08**

[a]Best results are shown in bold.

D. Evaluation Based on KITTI 2015 Benchmark

We then evaluate our method based on the KITTI 2015 benchmark. In contrast to the clean input images of the Middlebury 3.0 benchmark, the images of the KITTI 2015 benchmark are much noisier. Furthermore, while the Middlebury 3.0 benchmark evaluates subpixel accuracy on complex scene structures, the KITTI 2015 benchmark uses a loose error threshold of three pixels to evaluate results on simple scene structures. As a result, currently few methods achieve top ranks on both the Middlebury 3.0 benchmark and the KITTI 2015 benchmark simultaneously. Compared to the winner-takes-all cost-aggregation methods, the global-optimization methods with heavy spatial smoothing are more advanced under the evaluation criterion of KITTI. However, although not designed for solving this kind of problem,

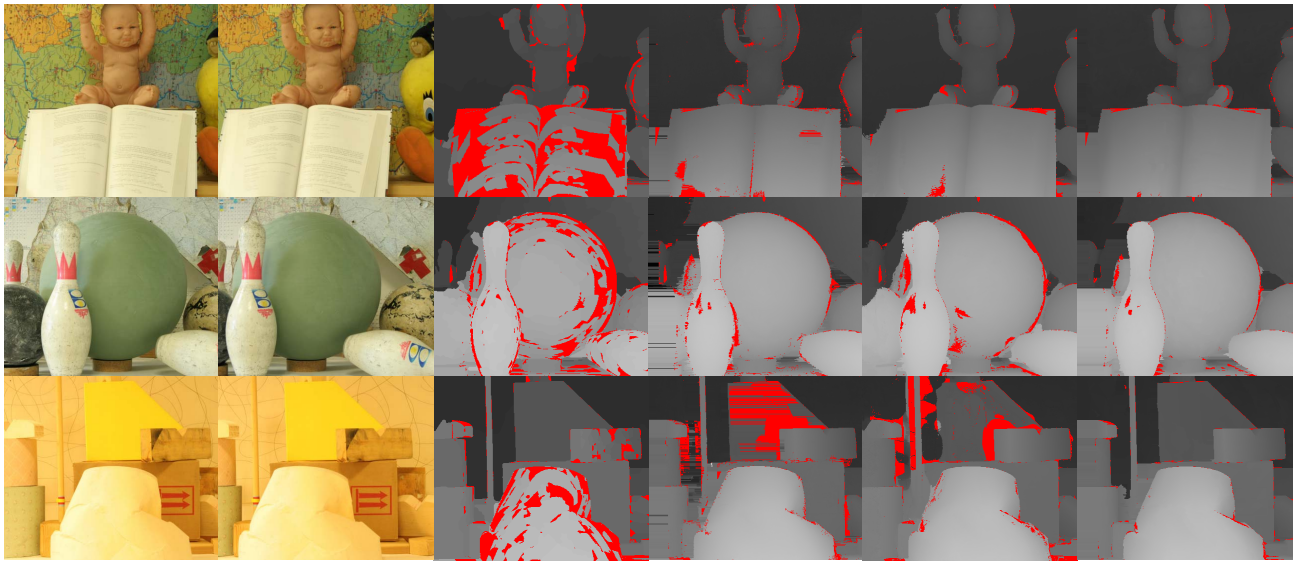


Fig. 5. Comparison of disparity maps generated by different methods. (a) Left input image. (b) Right input image. (c) Result of MST + CNN. (d) Result of PatchMatch. (e) Result of PMF. (f) Result of 3DMST. Error pixels are marked in red.

our method still performs comparably well against the state-of-the-art methods on the benchmark.

Table 5 shows the error rates of all methods that also perform well on the Middlebury benchmark [37,38], together with other cost-aggregation methods. Our method ranks second among them, and outperforms all other cost-aggregation methods. "D1-all" is the error rate of all pixels that have ground truth with an error threshold of three pixels, while "D1-bg" and "D1-fg" are error rates of foreground and background pixels.

Figure 6 compares the results of ours and MC-CNN, which enforces spatial smoothness heavily and obtains state-of-the-art performance against the KITTI benchmark. As our method computes the disparity of each pixel independently without global regularization, our results are noisier but have better detail information. Although our average error rate is slightly higher due to the absence of spatial regularization, our method recovers better details of the further objects, e.g., distant cars, trees, and pillars.

E. Parameter and Time Analysis

We then evaluate the performance of our method with different settings of some key parameters. Parameter γ controls the support weights for cost aggregation. Figure 7 shows the error rates when setting γ to different values for some randomly chosen high-resolution image pairs (about 1500 × 1000) from the Middlebury 3.0 benchmark and low-resolution image pairs (about 450 × 350) from the Middlebury 2006 data set, respectively. Our method is insensitive to γ in a large region in both data sets. On the other hand, for different resolution images that have a similar object scale, the optimal support region sizes should be different. Hence, the optimal γ values of the two data sets are different in our experiments.

Parameter λ controls the tree sizes of the multiple MSTs. When λ is too small, the multiple MST structures are not a good approximation of the single MST, while when λ is too large, the number of 3D labels tested for each pixel is not sufficient. Figure 8 shows the error rates of different λ on the two data sets. Our method is insensitive to λ in a large region in both cases. Meanwhile, as the image sizes of the Middlebury 3.0 benchmark are about 10 times larger than those of the Middlebury 2006 data set, the best tree sizes and λ values are different in the two data sets.

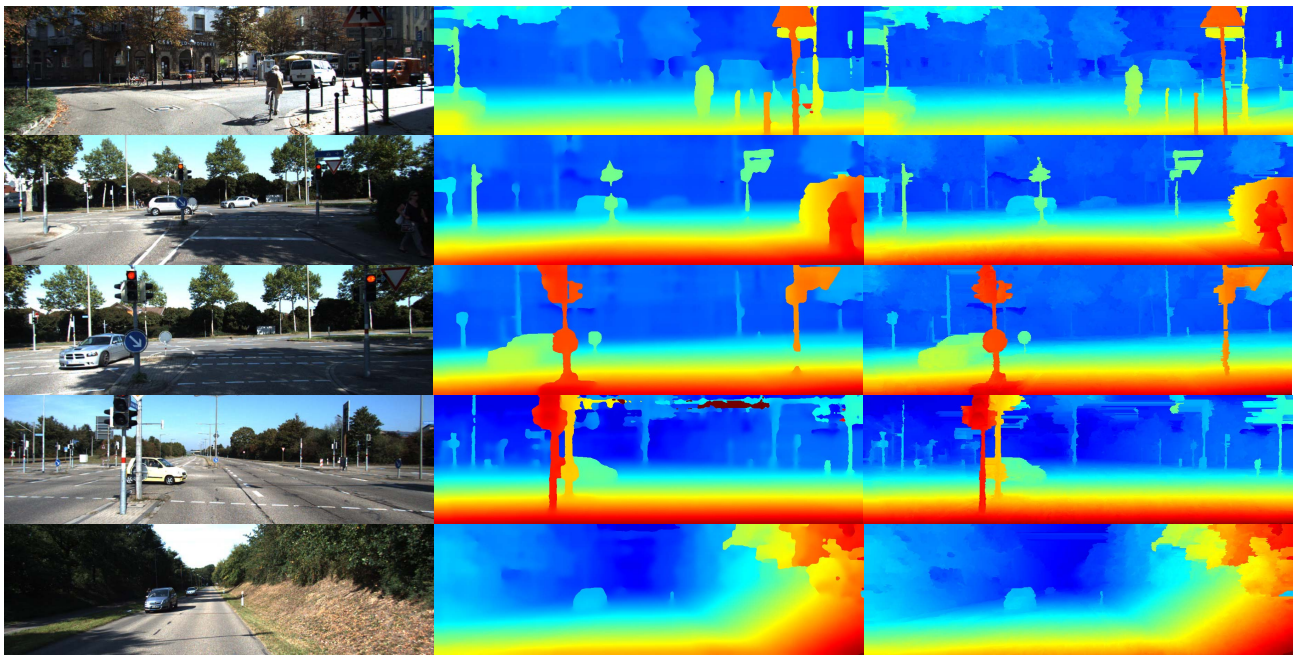
Figure 9 shows the raw disparity maps of "Adiron" after different iteration times as an example. Starting from random initialization, the error regions decrease rapidly during the first few iterations. After the first 10 iterations, the succeeding raw disparity maps are visually indifferent. In the latter iterations, only a tiny number of error pixels are improved, leading to slightly lower error rates.

In terms of running time, for the high-resolution images (about 1500 × 1000) in the Middlebury 3.0 benchmark, our cost-aggregation method takes about 160 s to generate a pair of disparity maps for both views. For the low-resolution images (about 450 × 350) in the Middlebury 2006 data set, our method takes about 25 s to process one image pair. For comparison, our implementation of PatchMatch without parallel acceleration takes more than 10 min to process one

Table 5. Error Rates with Threshold 3 Pixels on the Test Set of KITTI 2015 Benchmark[a]

Method	D1-bg	D1-fg	D1-all
MC-CNN-acrt	**2.89**	**8.88**	**3.89**
DMST	3.36	13.03	4.97
LPU	3.55	12.30	5.01
MDP	4.19	11.25	5.36
MeshStereo	5.82	21.21	8.38
CostFilter	17.53	22.88	18.42
MST	45.83	38.22	44.57

[a]Best results are shown in bold.



(a) Reference Image (b) MC-CNN (c) 3D MST

Fig. 6. Comparison of disparity maps generated by MC-CNN and the proposed method.

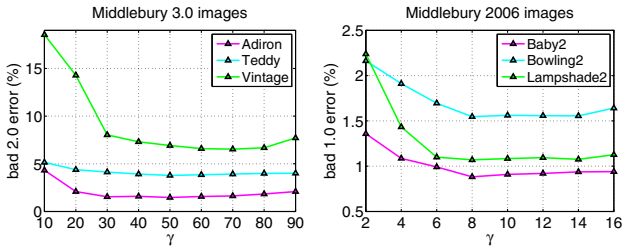


Fig. 7. Sensitivity analysis of γ on the Middlebury 3.0 benchmark and the Middlebury 2006 data set. The image sizes in the left figure are 10 times larger than those in the right figure.

low-resolution image pair, and PMF as the accelerated version of PatchMatch reports 20 s. Our method is one of the fastest among the methods that employ the random search strategy of PatchMatch.

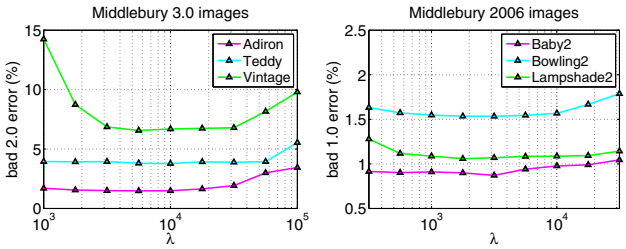


Fig. 8. Sensitivity analysis of λ on the Middlebury 3.0 benchmark and the Middlebury 2006 data set. The image sizes in the left figure are 10 times larger than those in the right figure. Note that the x axis is represented in log coordinates.

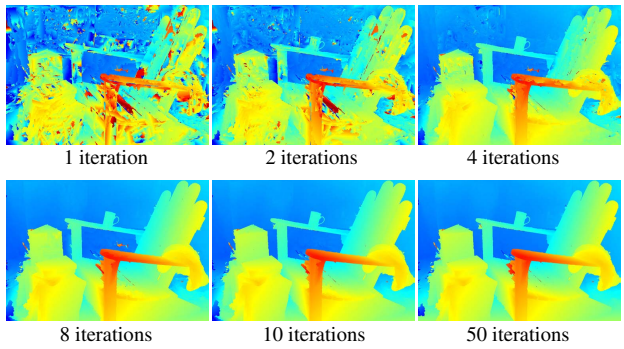


Fig. 9. Raw disparity maps of "Adiron" after different iteration times (without postprocessing).

5. CONCLUSION

We present a novel cost-aggregation method for the stereo-matching problem that works on 3D labels. We propose the MST-based tree filtering that works on 3D labels for better support regions. To reduce redundant computation, we introduce the multiple MST structure as an approximation of the single MST. In order to search possible 3D labels for each pixel, we design the spatial propagation and random refinement strategy on the multiple MST structures. Experiments have shown that our method outperforms all existing MST-based and PatchMatch-based methods in different data sets, and currently ranks first on the Middlebury 3.0 benchmark. As a simple cost-aggregation method, our method obtains better accuracy than all global optimization methods. In the future, we will try to add a global optimization step onto the current

cost-aggregation method for better spatial consistency, and design the parallel structure for real-time acceleration.

Funding. National Natural Science Foundation of China (NSFC) (61132007, 61503405, 61601021, U1533132).

Acknowledgment. We would like to thank the anonymous reviewers for their valuable advice.

REFERENCES

1. F. Gu, H. Zhao, X. Zhou, J. Li, P. Bu, and Z. Zhao, "Photometric invariant stereo matching method," Opt. Express **23**, 31779–31792 (2015).
2. Y. Geng, Y. Zhao, and H. Chen, "Stereo matching based on adaptive support-weight approach in RGB vector space," Appl. Opt. **51**, 3538–3545 (2012).
3. D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," Int. J. Comput. Vis. **47**, 7–42 (2002).
4. K. Zhang, J. Lu, and G. Lafruit, "Cross-based local stereo matching using orthogonal integral images," IEEE Trans. Circuits Syst. Video Technol. **19**, 1073–1079 (2009).
5. N. Guo, X. Sang, D. Chen, P. Wang, S. Xie, X. Yu, B. Yan, and C. Yu, "Automatic parameter estimation based on the degree of texture overlapping in accurate cost-aggregation stereo matching for three-dimensional video display," Appl. Opt. **54**, 8678–8685 (2015).
6. Y. Xu, Y. Zhao, and M. Ji, "Local stereo matching with adaptive shape support window based cost aggregation," Appl. Opt. **53**, 6885–6892 (2014).
7. F. Tombari, S. Mattoccia, L. Di Stefano, and E. Addimanda, "Classification and evaluation of cost aggregation methods for stereo correspondence," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2008), pp. 1–8.
8. M. Galar, A. Jurio, C. Lopez-Molina, D. Paternain, J. Sanz, and H. Bustince, "Aggregation functions to combine RGB color channels in stereo matching," Opt. Express **21**, 1247–1257 (2013).
9. C. Zhang, Z. Li, Y. Cheng, R. Cai, H. Chao, and Y. Rui, "Meshstereo: a global stereo model with mesh alignment regularization for view interpolation," in *Proceedings of the IEEE International Conference on Computer Vision* (IEEE, 2015), pp. 2057–2065.
10. K. Yamaguchi, D. McAllester, and R. Urtasun, "Efficient joint segmentation, occlusion labeling, stereo and flow estimation," in *Proceedings of the European Conference on Computer Vision* (Springer, 2014), pp. 756–771.
11. A. Klaus, M. Sormann, and K. Karner, "Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure," in *Proceedings of the IEEE International Conference on Pattern Recognition* (IEEE, 2006), Vol. **3**, pp. 15–18.
12. M. Bleyer, C. Rother, and P. Kohli, "Surface stereo with soft segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2010), pp. 1570–1577.
13. C. Olsson, J. Ulén, and Y. Boykov, "In defense of 3D-label stereo," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2013), pp. 1730–1737.
14. C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman, "PatchMatch: a randomized correspondence algorithm for structural image editing," ACM Trans. Graph. **28**, 24 (2009).
15. M. Bleyer, C. Rhemann, and C. Rother, "PatchMatch stereo–stereo matching with slanted support windows," in *Proceedings of the British Machine Vision Conference* (BMVA, 2011), pp. 1–11.
16. J. Lu, Y. Li, H. Yang, D. Min, W. Eng, and M. Do, "PatchMatch filter: edge-aware filtering meets randomized search for visual correspondence," IEEE Trans. Pattern Anal. Mach. Intell. (to be published).
17. Q. Yang, "A non-local cost aggregation method for stereo matching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2012), pp. 1402–1409.
18. X. Mei, X. Sun, W. Dong, H. Wang, and X. Zhang, "Segment-tree based cost aggregation for stereo matching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2013), pp. 313–320.
19. D. T. Vu, B. Chidester, H. Yang, M. N. Do, and J. Lu, "Efficient hybrid tree-based stereo matching with applications to postcapture image refocusing," IEEE Trans. Image Process. **23**, 3428–3442 (2014).
20. K.-J. Yoon and I. S. Kweon, "Adaptive support-weight approach for correspondence search," IEEE Trans. Pattern Anal. Mach. Intell. **28**, 650–656 (2006).
21. F. Besse, C. Rother, A. Fitzgibbon, and J. Kautz, "PMBP: PatchMatch belief propagation for correspondence field estimation," Int. J. Comput. Vis. **110**, 2–13 (2014).
22. Y. Li, D. Min, M. S. Brown, M. N. Do, and J. Lu, "SPM-BP: sped-up PatchMatch belief propagation for continuous MRFS," in *Proceedings of the IEEE International Conference on Computer Vision* (IEEE, 2015), pp. 4006–4014.
23. T. Taniai, Y. Matsushita, and T. Naemura, "Graph cut based continuous stereo matching using locally shared labels," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2014), pp. 1613–1620.
24. L. Li, S. Zhang, X. Yu, and L. Zhang, "PMSC: PatchMatch-based superpixel cut for accurate stereo matching," IEEE Trans. Circuits Syst. Video Technol. (to be published).
25. V. Muninder, U. Soumik, and A. Krishna, "Robust segment-based stereo using cost aggregation," in *Proceedings of the British Machine Vision Conference* (BMVA, 2014), pp. 1–11.
26. E. T. Psota, J. Kowalczuk, M. Mittek, and L. C. Perez, "Map disparity estimation using hidden Markov trees," in *Proceedings of the IEEE International Conference on Computer Vision* (IEEE, 2015), pp. 2219–2227.
27. J. Zbontar and Y. LeCun, "Computing the stereo matching cost with a convolutional neural network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2015), pp. 1592–1599.
28. P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," Int. J. Comput. Vis. **59**, 167–181 (2004).
29. R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," IEEE Trans. Pattern Anal. Mach. Intell. **34**, 2274–2282 (2012).
30. D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," IEEE Trans. Pattern Anal. Mach. Intell. **24**, 603–619 (2002).
31. D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nešić, X. Wang, and P. Westling, "High-resolution stereo datasets with subpixel-accurate ground truth," in *Proceedings of the German Conference on Pattern Recognition* (Springer, 2014), pp. 31–42.
32. H. Hirschmüller and D. Scharstein, "Evaluation of cost functions for stereo matching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2007), pp. 1–8.
33. M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2015), pp. 3061–3070.
34. K.-R. Kim and C.-S. Kim, "Adaptive smoothness constraints for efficient stereo matching using texture and edge information," in *Proceedings of the IEEE International Conference on Image Processing* (IEEE, 2016), pp. 3429–3433.
35. J. T. Barron and B. Poole, "The fast bilateral solver," in *Proceedings of the European Conference on Computer Vision* (Springer, 2016), pp. 617–632.
36. H. H. D. Scharstein and R. Szeliski, "Middlebury stereo evaluation. Version 3," http://vision.middlebury.edu/stereo/eval3/.
37. A. Hosni, C. Rhemann, M. Bleyer, C. Rother, and M. Gelautz, "Fast cost-volume filtering for visual correspondence and beyond," IEEE Trans. Pattern Anal. Mach. Intell. **35**, 504–511 (2013).
38. A. Li, D. Chen, Y. Liu, and Z. Yuan, "Coordinating multiple disparity proposals for stereo computation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2016), pp. 4022–4030.