HHS Public Access

Author manuscript

*Pattern Recognit*. Author manuscript; available in PMC 2016 July 01.

Published in final edited form as:

*Pattern Recognit*. 2015 July 1; 48(7): 2269–2278. doi:10.1016/j.patcog.2015.01.002.

# Cross-trees, Edge and Superpixel Priors-based Cost aggregation for Stereo matching

**Feiyang Cheng**[a], **Hong Zhang**[a], **Mingui Sun**[b], and **Ding Yuan**[a,*]

Feiyang Cheng: chfybuaa@gmail.com; Hong Zhang: dmrzhang@buaa.edu.cn; Mingui Sun: drsun@pitt.edu; Ding Yuan: dyuan@buaa.edu.cn

[a]Image Research Center, Beihang University, Beijing, China

[b]Department of Neurosurgery, University of Pittsburgh, Pittsburgh, USA

## Abstract

In this paper, we propose a novel *cross-trees* structure to perform the nonlocal cost aggregation strategy, and the *cross-trees* structure consists of a *horizontal-tree* and a *vertical-tree*. Compared to other spanning trees, the significant superiorities of the *cross-trees* are that the trees' constructions are efficient and the trees are exactly unique since the constructions are independent on any local or global property of the image itself. Additionally, two different priors: *edge prior* and *superpixel prior*, are proposed to tackle the false cost aggregations which cross the depth boundaries. Hence, our method contains two different algorithms in terms of *cross-trees+prior*. By traversing the two crossed trees successively, a fast non-local cost aggregation algorithm is performed twice to compute the aggregated cost volume. Performance evaluation on the 27 Middlebury data sets shows that both our algorithms outperform the other two tree-based non-local methods, namely *minimum spanning tree* (MST) and *segment-tree* (ST).

### Keywords

stereo matching; cost aggregation; image filtering; spanning trees

## 1. Introduction

Dense two-frame stereo matching has been extensively investigated for decades as a traditional low-level vision task, since it is crucial for many applications such as 3D reconstruction [1, 2], image-based rendering [3, 4] and anonymous driving *etc* [5]. According to the analysis and taxonomy scheme proposed in [6], stereo matching algorithms can be categorized into two groups: local algorithms and global algorithms. Stereo matching algorithms are often implemented following a subset of the four steps or all:

1. Cost function/cost volume estimation



*Correspondence author, Phone & Fax:+86-10-82338991.

2. Cost aggregation within a support region
3. Disparity computation/optimization
4. Disparity refinement

Global algorithms usually make explicit smoothness assumptions, and minimize a predefined energy function to obtain optimal results [7, 8, 9]. Despite of the reliable matching results obtained, global algorithms are often time-consuming. All local algorithms compute the matching cost (step 1) firstly and then perform the cost aggregation (step 2) to get a locally optimized cost volume [6, 10, 11, 12, 13, 14]. We mainly focus on efficient and effective local and non-local methods in this paper, and the readers are referred to a recent study for a comprehensive study of the global methods [15].

To find a correspondence (x, x′), the problem of the local methods can be concluded as a comparison of the similarity of two local patches which around x and x′ respectively. The similarity of the two patches are computed by aggregating the costs of the pixels within the patches. Hence, the cost aggregation (step 2) procedure has important impacts on the accuracy and the efficiency of a local algorithm. The cost aggregation of a pixel in traditional local algorithms is usually performed by averaging the costs of the pixel itself and all its neighboring pixels. Here, the implicit assumption is that all the pixels which lie in a special local support region have similar disparities, as shown in Fig.1(a). Such local methods suffer from well known “edge fatten” effect once the local support regions cover the depth boundaries. The problem can be explained in the context of image filtering. For instance, the box filter always blurs the edges of an image during the image denoising procedure. Hence, the problem of the cost aggregation step is how to choose optimal local support regions for each pixel. Various researches have been conducted to estimate optimal support regions for the cost aggregation, such as *various window-based* methods [10, 11] and *adaptive support weights* (ASW) methods (also known as *local filtering-based* methods) [12, 13, 14] which have state of the art performance in last years. However, the selected support regions of the ASW methods are often limited in a pre-defined window of fixed size. Due to this reason, this kind of methods cannot work well for the stereo images with large planar surfaces.

Recently, Yang proposed a non-local cost aggregation method based on a MST [16]. As shown in Fig.1(b), a pixel is able to get supports from all the other pixels of the image through unique paths on the tree structure. Different from aforementioned *various window-based* methods and ASW methods, the cost aggregation was performed over the whole image for each pixel to establish non-local optimized results. M.Xing proposed a *segment-tree* structure to perform the non-local cost aggregation strategy [17]. These work proved that the non-local cost aggregation methods outperform the local ones much more.

Hence, this paper mainly focuses on the non-local cost aggregation procedure by comparing different tree construction techniques [16, 17]. Section 2 is an overview of the previous work of the cost aggregation procedure. We briefly review the workflow of the non-local framework and the non-local cost aggregation algorithm in Section 3.1 and then introduce the *cross-trees* and the two priors in Section 3.2. A discussion of the strategies for

constructing different tree structures is also provided in Section 3.2. Experimental results and performance evaluations are shown in Section 4. A detail analysis of the short points of the tree-based non-local cost aggregation is given in Section 5. Finally, we draw the conclusions and discuss the future work in Section 6.

## 2. Previous work

The cost aggregation procedure can be considered as two sub-problems:(1) how to estimate the optimal support regions; (2) how to aggregate the matching costs of the pixels within the estimated support regions (usually, in terms of support weights). We review the related work in this section according to the two sub-problems above.

### 2.1. Various window-based methods

Most early local methods aimed at estimating various windows for different pixels, from adaptive [10, 11] to shiftable windows [18]. The optimal windows were often selected based on certain local properties to avoid covering disparity discontinuities. Fusiello *et al* developed a multiple window approach by performing cost aggregation in nine different window models and chose the window with the smallest aggregated cost as the optimal window. However, limited window models are not sufficient to represent support regions with arbitrary shapes and sizes. Some researchers proposed to use *cross-based structures* to represent various support regions and developed several competitive algorithms [19, 20].

Most of these methods considered the sub-problem (1) only, and gave all the pixels the same weight. It means that these methods were developed based on a simple smoothness assumption that all the pixels within the same support region have a constant disparity. Such methods may result in over-smooth disparity slices within smooth curved surfaces.

### 2.2. Adaptive support weights methods

One main resolution is adaptive support weights (ASW) strategy, in which weighted supports decide whether the neighboring pixels contribute more or less to the center pixel [12, 13]. The support weights that adapt according to similarity and proximity to the central pixel of the predefined large support window actually control the real aggregation region and power. However, computing support weights iteratively for each central pixel is a time-consuming task. Many researchers indicated that such strategy can be approximately re-implemented by using local filter such as bilateral filter [21] and guided filter [14, 22]. In this way, both the accuracy and the efficiency of the ASW algorithms have been improved. Hereto, the idea that the edge-preserving filters can be employed to aggregate the matching costs and to preserve the depth edges simultaneously becomes clear gradually. However, all the *local filtering-based* methods still establish locally optimized results within predefined windows which have a fixed size. Detailed comparisons and discussions of the *local filtering-based* methods can be found in two recent reviews [23, 24].

### 2.3. Tree-based non-local methods

As mentioned above, a MST-based non-local cost aggregation method was proposed recently [16]. In the same non-local framework, X.Mei *et al*. proposed to employ a ST

instead of a MST to optimize the non-local cost aggregation procedure [17]. Conceptually, they segmented the image at first and then constructed a sub-tree (i.e., a sub-MST) for each segment. Finally, a ST was constructed by linking these sub-trees for the non-local cost aggregation. The main idea, using the segment prior to avoid connecting two pixels which locate at the different sides of a segment boundary (i.e., potential depth boundaries), is similar with other segment-based stereo matching methods [25, 26]. Both the MST method and the ST method outperform the local methods in aggregation accuracy [14, 21].

Actually, constructing a tree or a graph to improve the optimization procedure is not new in many global methods such as *Dynamic Programming* (DP) and *Loop Belief Propagation* (LBP). Veksler firstly employed DP on a tree instead of a scanline to enforce vertical consistency to establish truly global optimization results [27]. Cheng Lei *et al*. improved the tree-based DP method by using a novel tree structure which they called region-tree [28]. Zitnick *et al*. formulated a new MRF for global optimization by over-segmenting the images [4].

In conclusion, for both the non-local cost aggregation and the global optimization which are based on a tree structure, the critical problem is the construction of the tree. In this paper, we focus on the non-local cost aggregation methods only. Hence, we describe the problem in the context of the non-local cost aggregation. To construct a tree from a graph, the important edges need to be preserved and the edges crossing the depth boundaries must be removed. By an important edge, we mean an edge whose two nodes are more likely to have the same disparity. Local criterions (i.e., color difference [16], distance [27] etc.), non-local properties (i.e., segmentation [17], over-segmentation [4] etc.) or a joint version of the two can be used to decide whether an edge should be selected to construct the tree. If an edge is selected, its two nodes are connected directly and they can get supports from each other during the cost aggregation procedure. Moreover, two nodes which are not connected directly by an edge can also get supports from each other through a unique path on the tree; see details in Fig. 1(b). Once the tree is constructed, the support weight between a pair of pixels is decided by the distance between them on the tree.

## 3. Algorithm

Our work is directly motivated by the two non-local cost aggregation methods recently [16, 17] as well as the *local filtering-based* methods [14, 21] and a simple tree-based DP optimization method [29]. We employ two crossed trees, namely *horizontal-tree* and *vertical-tree*, and perform the nonlocal cost aggregation algorithm proposed in [16] successively to get the aggregated cost volume. For convenience, we use the *cross-trees* to denote the two crossed trees in the residual part of this paper. The constructions of the *cross-trees* are simple, efficient and especially independent on any local or non-local properties. We make explicit smoothness assumption in our method by truncating the distance (i.e., color difference) between two neighboring pixels. However, performing the non-local cost aggregation on such trees directly will blur the depth boundaries. Hence, two different priors, namely *edge prior* and *superpixel prior*, are proposed to prevent the false smoothing at the depth boundaries. Based on the two different priors, two different algorithms are proposed respectively and each one has competitive performance compared to other non-

local cost aggregation methods. By performing the non-local cost aggregation strategy on different trees, MST, ST and our method all have competitive rankings on the Middlebury website compared to the state of the art local cost aggregation methods.

In this section, we review the non-local framework at first and then propose our methods.

### 3.1. Non-local cost aggregation

The non-local framework for initial estimation of disparity maps can be concluded in four steps as following:

1. Cost function/cost volume estimation
2. Tree construction
3. Non-local cost aggregation on the constructed tree
4. Disparity estimation(WTA)

We mainly contribute to step 2 in this work, and Yang has proposed an efficient algorithm for step 3 which we will brief review in the remained part of this section [16]. Steps 1 and 4 are commonly used techniques in the context of stereo matching.

In the non-local framework, the reference image $I$ is represented as a 4-connected and undirected graph $G = (V, E)$. Here, a pixel in $I$ corresponds to a node in $V$ and a pair of neighboring pixels are connected by an edge in $E$ with the edge weight computed by using (1):

$$w(s,r)=w(r,s)=|I_s - I_r| \quad (1)$$

Here, $s$ and $r$ are two neighboring pixels in $I$, $I_s$ and $I_r$ represent the intensity of $s$ and $r$ respectively. Veksler proposed a MIDDT tree by considering distance transform additionally to estimate the edge weights [27]. However, using sophisticated similarity costs contribute to improving the tree construction while at the price of increasing computational cost. Note that using various matching cost functions to compute the edge weight is not appropriate here [30, 31, 32]. When we compare two pixels within the stereo images respectively, the assumption is that the support regions of the two pixels are also similar if the two pixels are the same point in 3D space. In this case, we can use various strategies to compare the two pixels by utilizing their neighboring pixels within the local support regions. It is totally reasonable. However, when we compare the two neighboring pixels in the same image to decide whether they can support each other, we assume that the two neighboring pixels which have similar color are likely to have similar disparity. Here, there is no assumption that the two pixels' neighboring pixels are also similar because the image textures are not repeated everywhere.

For the color images, the edge weight is the maximum $w(s, r)$ value estimated from three R-G-B channels. Using the max color difference on each channel can guarantee that the two neighboring pixels which have similar intensity on all the three channels give high support

weights to each other. Using the max rather than other possibilities, such as the mean, is an empirical choice according to the experiments.

A subset *T* of *E* will be extracted to construct a tree where a unique path *P*(*p*, *q*) can be find between any pair of pixels (*p*, *q*) in *I*. The distance *D*(*p*, *q*) = *D*(*q*, *p*) between *p* and *q* is defined as the sum of the edge weights along *P*(*p*, *q*). Thus the weighted support *S*(*p*, *q*) which *p* and *q* give to each other is defined as (2) with a user-specified parameter $\sigma$:

$$S(p,q)=S(q,p)=\exp(-\frac{D(p,q)}{\sigma}) \quad (2)$$

For the disparity level *d*, let $C_d(p)$ denote the matching cost of *p*. It is straightforward to compute the final aggregated cost $C_d^A(p)$ of *p* by using (3):

$$C_d^A(p)=\sum_{q\in I} S(p,q)C_d(q) \quad (3)$$

During the non-local cost aggregation, *p* gets weighted supports from all the pixels in *I*. It is different from the *local filtering-based* methods in which *p* gets weighted supports from its neighboring pixels within a local support region only. Usually, the local support region is a predefined square window.

Computing $C_d(p)$ for each pixel *p* by using (3) iteratively is really time-consuming. Yang proved that the non-local cost aggregation can be accomplished in exactly linear time [16]. By performing a leaf-to-root and a root-to-leaf cost aggregation on the tree structure successively, the final aggregated cost for each node can be estimated (see Fig.2 for details).

In the leaf-to-root pass, the intermediate aggregated cost $C_d^{A\uparrow}(q)$ of each node *q* can be computed by using (4):

$$C_d^{A\uparrow}(q)=C_d(q)+\sum_{Par(q')=q} S(q,q')C_d^{A\uparrow}(q') \quad (4)$$

Here, $Par(q')$ denotes the collection of all the children nodes of *q*. Note that the final cost aggregation $C_d^A(p)$ equals to $C_d^{A\uparrow}(p)$ for the root node *p* in Fig.2. Then for each leaf node *q*, the final aggregated cost $C_d^A(q)$ can be computed in the root-to-leaf pass as following:

$$C_d^A(q)=S(Par(q),q)\cdot C_d^A(Par(q))+[1-S^2(q,Par(q))]\cdot C_d^{A\uparrow}(q) \quad (5)$$

For each disparity level, the computational complexity of the cost aggregation is $O(n)$, where *n* is the number of the pixels of the reference image. Hence, the total computational complexity of the non-local cost aggregation algorithm is $O(n \cdot l)$, where *l* is the disparity range. To refine the established disparity maps, a non-local post-processing technique based on the tree is also proposed in [16].

### 3.2. Cross-trees and Priors

Our work exactly follows the non-local framework, and the significant difference is that we employ a different *cross-trees* structure. Moreover, two different algorithms are proposed due to the two different priors (*edge prior* and *superpixel prior*) incorporated into the non-local framework. The weights of the edges in *G* are also redefined according to the pixel similarity and the priors.

**3.2.1. Cross-trees for non-local cost aggregation—**The problem comes back to the construction of the tree. Yang constructed a MST which has the minimum sum of the edge weights among all the spanning trees of *G* [16]. The intuition is that an edge is less likely to cross the depth boundaries if its two nodes (two neighboring pixels in *I*) have higher pixel similarity. For the MST method, there are two problems need to be emphasized on: 1) the tree structure is not unique since there are a lot of edges which have the same weight; 2) the edges which lie in highly textured regions are less likely to be selected though neighboring color variation enhances the pixels' distinction. Hence, only employing simple local property is not enough for constructing a unique optimal tree. Non-local segment prior was additionally employed in [17] to enhance the tree construction procedure. Though accuracy improvement was achieved, it involved the segmentation task which is especially hard to tackle in some cases. Despite of the image segmentation task itself, constructing a sub-MST in each segment may still suffer from the problems of the MST construction if the segments are large.

Motivated by the problems discussed above, our tree construction aims at being independent on the reference image (i.e., not related to pixel similarity), but consistent with the fact that disparity maps are mostly spatially smooth. Hence, we make explicit smoothness assumption as global methods during the tree construction procedure, and then incorporate priors into the nonlocal framework to preserve the depth boundaries.

Instead of the pixel similarity, the proximity is referred only to construct the *cross-trees*. The structures of the *cross-trees* are shown in Fig.3. We perform the non-local cost aggregation algorithm on the two crossed trees successively. The final cost aggregation process for a pixel *p* is illustrated in Fig.4. To sum up, we perform a ‘window-based’ cost volume filtering in a non-local way. Different from the *local filtering-based* methods, the support ‘window’ for each pixel is set to be the whole reference image.

As mentioned above, we make explicit smoothness assumption by assigning each edge a truncated weight as (6) at first, and then preserve the depth boundaries by incorporating a prior into the non-local framework later.

$$w(s,r)=\min(|I_s - I_r|, \tau) \quad (6)$$

Here, $\tau$ is the threshold of the color difference of two neighboring pixels *s* and *r*. Truncated weights of the edges are corresponding to our global smoothness assumption. Note that the proximity is implicitly employed here since the distance is accumulated along a path on the

tree. In the other words, the further the two pixels in Euclidean space, the further the two pixels on the tree. Fig.5 shows the consistency of the two different spatial measurements.

**3.2.2. Edge prior and Superpixel prior—**Any pair of neighboring pixels is connected on the *cross-trees* with a truncated edge weight, and the non-local cost aggregation will blur the depth boundaries because we assume disparity smoothness everywhere. Due to this reason, appropriate priors must be employed to indicate the potential locations of the depth boundaries. One of the commonly used priors is the *edge prior*, as shown in Fig.6(a). However, only a small part of the color edges belongs to the true depth boundaries and assuming all the color edges as the depth boundaries will degrade the cost aggregation within highly textured regions. Motivated by the categorization scheme proposed in [33, 34], we propose to employ the *superpixel prior* in this work. By taking advantage of the over-segmentation results, many false depth boundaries are removed as shown in Fig.6(b).

Someone may wonder why we employ over-segmentation instead of segmentation. We use the segmentation results as a prior to estimate the support regions during performing the cost aggregation. As the units of local support regions, the segments need to be compact and regular, especially in highly textured regions. From this perspective, segmentation algorithms cannot work as well as over-segmentation algorithms. Additionally, the depth boundaries are fully connected in this way since the superpixels are compact and regular. In this paper, we utilize the SLIC algorithm [35] which have competitive performance on adherence to potential disparity boundaries than other superpixel algorithms [36, 37, 38]. We have tried more complex constraints such as the segmentation and the global edge constraint (*GEC*) proposed in [33, 34], but the improvement was not obvious.

In our work, the weights of the edges are dependent on the pixel similarity (color difference) and the prior we employed. The weight of an edge is rewritten as following:

$$w(s,r)=\begin{cases} |I_s - I_r|, if\, e(s,r) \cap the\ prior \\ \min\left(|I_s - I_r|, \tau\right), others \end{cases} \quad (7)$$

Here, $e(s, r) \cap$ *the prior* means that the edge between *s* and *r* crosses the prior. $\tau$ is the threshold of the edge weights if the edges do not cross the prior. $\tau$ is set to be 6/255 in all our experiments to enforce the cost aggregation between pixels at the same side of the prior. In this way, we aggregate the costs within planar surfaces and preserve the depth boundaries when the prior exists.

Fig.7 shows how the prior selects optimal support regions for the center pixel *p*. For an edge $e(q_3 \rightarrow q_2)$ on a path $P(q_3, p)$, its weight is defined as the color difference of the two neighboring pixels since it crosses the proposed prior. In this case, the distance from $q_3$ to *p* will be raised sharply if the color difference of $q3$ and $q2$ is large. Hence, further pixels such as $q_3$ will contribute much less to the center pixel *p*. By cutting the cost aggregation flow on different paths, our method can estimate support regions with arbitrary shapes and sizes, which is a challenge problem for traditional *window-based* local methods.

To a certain extent, our work is also motivated by the *cross-based structures* methods [19, 20]. Instead of estimating the *cross-based structures* locally, we use the priors to locate potential local support regions' boundaries and then estimate the support regions as cross-structures on the horizontal and the vertical scanlines implicitly by cutting the cost aggregation flow as shown in Fig.7.

Note that for the *superpixel prior*, there are over-segment boundaries even between neighboring pixels within low-textured regions. Such over-segment boundaries will not terminate the cost aggregations from further pixels on the path since the edge weights (equal to pixel similarity as defined in (7)) are still low.

### 3.3. Computational complexity

For all the tree-based non-local methods, we compare the computational complexities of the tree construction and the non-local cost aggregation only because the other two steps are exactly the same for all the methods in the non-local framework.

For an $N \times N$ image, let $n$ denote the number of the pixels in $I$ and $m = O(n)$ denote the number of the edges. For both the MST and the ST constructions, the computational complexity of the edge weights estimation is $O(m)$. The computational complexity is $O(log_2 m + 2m log_2 n + n)$ for the MST construction, and $O(m + 2m\alpha(m))$ for the ST1 construction. Here, $\alpha(m)$ can be approximated as a small constant for practical applications according to [17]. For ST2, the tree construction time is doubled compared to ST1 because ST1 need to be performed at first to provide initial depth results. Our *cross-trees* structure needs $n - N$ *subtraction* operations for constructing each tree and the total computational complexity is $O(2(n - N))$.

The computational complexity of the non-local cost aggregation ($O(n \cdot l)$) is doubled in our algorithms and ST2 since we both need to process two trees successively. We use Cross_E to denote our *cross-trees+edge prior*-based algorithm and Cross_Sp to denote our *cross-trees+superpixel prior*-based algorithm. Table 1 shows the comparison of the computational complexities of the entire five algorithms. For Cross_E and Cross_Sp, additional computations are needed to detect the color edges and to establish superpixels respectively. The exact execution time of each algorithm will be provided in the experiments (Section 4).

## 4. Experimental results

The performance of our method and the other two non-local cost aggregation methods (MST and ST) is compared in this section. Both qualitative evaluations and disparity maps are provided. Our method contain two algorithms: Cross_E and Cross_Sp. The ST method also contains two algorithms: initial ST-based algorithm ST1 and depth map enhanced ST-based algorithm ST2.

### Code

The results of ST1 and ST2 are estimated by using the C++ code which has been published by the paper authors [17]. We implement our algorithms and re-implement MST in the same code framework. Hence, all the functions are the same in the five algorithms except the

functions of the tree construction and some pre-processing steps. This is an equitable way for comparing the cost aggregation accuracy and the running time of the algorithms.

### Data sets

We use 27 Middlebury data sets [39], including the four commonly used data sets (*Tsukuba*, *Venus*, *Teddy* and *Cones*), to give a more reliable evaluation of the performance of the five algorithms. Moreover, we test MST, ST1 and Cross_E on many stereo video data sets from the Microsoft i2i dataset [40].

### Parameter setting

For all the test data sets, the parameter of both our algorithms is constant: $\sigma = 0.05$. $\sigma$ is set empirically to permit a pixel to get weighted supports from other pixels on the tree, and to cut the cost aggregation flow on a path at where it crosses the prior we employed.

For both our algorithms, the priors define the support region for a pixel $p$ and $\sigma$ controls the support weights of $p$'s neighboring pixels within the estimated support region. A larger $\sigma$ means that the cost of pixel $p$ will be smoothed powerfully, and such parameter setting may result in over-smooth disparity maps. On the contrary, a smaller $\sigma$ means a less powerful filtering of the cost of pixel $p$ which will lead to noisy disparity maps.

For the Cross_Sp algorithm, the number of superpixels $n_{sp}$ is set to be $n/(10 \times 10)$. Here, we suppose that the stereo images can be over-segmented into superpixels with an approximate size $10 \times 10$. In the other words, we suppose that the superpixels with an approximate size $10 \times 10$ will not cover the depth boundaries under this constraint for the test stereo images. A $10 \times 10$ superpixel, within highly textured regions of the reference image, usually has a clear distinction to find reliable corresponding points in the other image for the pixels within it.

However, $n_{sp}$ should be changed to a larger value to segment the stereo images into smaller patches (superpixels) for stereo images which contain tiny elements (less than $10 \times 10$ pixels). In this way, the superpixels will not cover the tiny elements' boundaries while degrade the distinctiveness. Hence, $n_{sp}$ should be set as a trade-off between details preservation and superpixels' distinctiveness for matching.

For the MST, ST1 and ST2 algorithms, the parameters are set to be the same with the settings in the original papers. The AD-Gradient cost function proposed in [14] is used for all the five algorithms to compute the cost volume.

Table 2 shows the quality evaluation of the initial disparity maps established by using the five algorithms. We firstly perform the non-local cost aggregation algorithm on the different trees and then establish the disparity maps with the WTA strategy. To compare the aggregation accuracy, no post-processing is performed in this step and only the non-occluded regions are evaluated. In Table 2, the normal numbers are the percentages of the bad pixels in the disparity maps with error threshold 1 and the subscript numbers are the relative rank in each row. ST1 and ST2 outperform MST for most of the test data sets as declared in the authors' paper [17]. The initial disparity maps established by using the five

algorithms are shown in Fig.8. Both our algorithms perform better than other methods within planar surfaces for the reason that the truncated edge weights enforce strong cost aggregations within these regions where no prior is detected or the color difference is low.

Among the 27 test data sets, Cross_E ranks 1 for 8 stereo pairs, Cross_Sp ranks 1 for 10 stereo pairs and ST2 ranks 1 for 9 stereo pairs. The average rank and the average error are computed in the last two rows of Table 2. Both our algorithms are competitive and rank 1 and 2 respectively. Cross_Sp has both the best overall accuracy and the best overall rank.

For the four standard Middlebury data sets, we test the five algorithms on a PC platform with Core i5 3.00GHz CPU-8GB Memory-64 bit OS. The average execution time for each algorithm is: MST-0.44s, ST1-0.49s, ST2-0.97s, Cross_E-0.49s, and Cross_Sp-0.69s. Cross_E and ST1 have the similar efficiency with MST. Cross_Sp is slower than Cross_E due to the superpixel algorithm we employed [35]. ST2 is the slowest algorithm since it requires an initial estimation of the depth map and an enhanced construction of the tree.

We use the Cross_Sp algorithm to establish the initial disparity maps for both the left and the right images of a stereo pair. After the crosschecking, we employ a constant time weighted median filter (WMF) as the post-processing technique [41]. The final disparity maps of the four standard Middlebury data sets are shown in Fig.9. Table 3 is the performance evaluation on the Middlebury test bed. Compared to the results which other authors submitted to the Middlebury website, our method ranks 2 among the four methods. GF had been the best local method when the original paper was published [14]. Actually, the gaps between the average error rates of the algorithms are really narrow and different post-processing techniques have significant effect on the final results. In this paper, we mainly focus on the cost aggregation accuracy (i.e., the initial disparity maps without post-processing).

Additional results of the *Illkay* and the *Simon* stereo video data sets from the Microsoft i2i dataset are shown in Fig.8. We only compare the MST, ST1 and Cross_E algorithms for video processing because they are much faster than Cross_Sp and ST2 and have the similar efficiency. Cross_E establishes the most smooth disparity maps within the foregrounds and has the best performance near the foregrounds' depth boundaries.

## 5. Discussion

A challenge problem of local stereo matching methods is locally repeated texture. In such case, a pixel of the reference image can often find several correspondents in the other image because local methods measure the similarity of a correspondence locally. In this section, we show the performance of the non-local algorithms on three challenge test data sets: *Midd1*, *Midd2* and *Plastic*. For all the images, there is large repeated texture in the backgrounds or foregrounds. As shown in Table 4, the average error rates of the algorithms are all over 30%. The bad results mainly locate in repeated texture regions as shown in Fig.11.

Actually, all tree-based non-local methods are proved to be still establishing locally optimized results within large planar surfaces due to the Gaussian kernel of the support weights. As we have discussed in Section 3, the distance between two pixels increases

progressively along a path. Fig.12 shows that a leaf node contributes to the root node only when the distance between them is smaller than a value which is related to the parameter $\sigma$. Hence for a given $\sigma$, the maximum size of the support regions is limited. It means that 'non-local' methods still establish 'local' optimized results when there are large planar surfaces with repeated texture.

## 6. Conclusions and Future Work

We proposed a novel *cross-trees* structure for the non-local cost aggregation. Based on two different priors, two algorithms are implemented to address the stereo matching problem. Compared to the other two top-ranked non-local cost aggregation methods, both our algorithms have better overall rank and overall accuracy for initial disparity map estimation. Performance evaluation on the Middlebury test bed also shows that our method is competitive. The future work should emphasize on constructing more effective trees to resolve the challenge problem as we discussed above. New ideas for performing the non-local cost aggregation are needed too.

## Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

## Acknowledgments

This work was supported by the National Natural Science Foundation of China Grant No.61272351 and the National Institutes of Health Grant No.R01CA165255 and R21CA172864 of the United States.

## References

1. Pollefeys M, Nistér D, Frahm, et al. Detailed real-time urban 3d reconstruction from video. International Journal of Computer Vision. 2008; 78(2-3):143–167.
2. Geiger, A.; Ziegler, J.; Stiller, C. Intelligent Vehicles Symposium. IEEE; 2011. Stereoscan: Dense 3d reconstruction in real-time; p. 963-968.
3. Fehn C. Depth-image-based rendering (dibr), compression, and transmission for a new approach on 3d-tv. Electronic Imaging, International Society for Optics and Photonics. 2004:93–104.
4. Zitnick CL, Kang SB. Stereo for image-based rendering using image over-segmentation. International Journal of Computer Vision. 2007; 75(1):49–65.
5. Sengupta, S.; Greveson, E.; Shahrokni, A.; Torr, PH. International Conference on Robotics and Automation. IEEE; 2013. Urban 3d semantic modelling using stereo vision; p. 580-585.
6. Scharstein D, Szeliski R. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. International Journal of Computer Vision. 2002; 47(1):7–42.
7. Boykov Y, Veksler O, Zabih R. Fast approximate energy minimization via graph cuts. Pattern Analysis and Machine Intelligence. 2001; 23(11):1222–1239.
8. Kolmogorov V, Zabin R. What energy functions can be minimized via graph cuts? Pattern Analysis and Machine Intelligence. 2004; 26(2):147–159.
9. Yang, Q.; Wang, L.; Ahuja, N. Computer Vision and Pattern Recognition. IEEE; 2010. A constant-space belief propagation algorithm for stereo matching; p. 1458-1465.
10. Kanade T, Okutomi M. A stereo matching algorithm with an adaptive window: Theory and experiment. Pattern Analysis and Machine Intelligence. 1994; 16(9):920–932.
11. Veksler, O. Computer Vision and Pattern Recognition. Vol. 1. IEEE; 2003. Fast variable window for stereo correspondence using integral images; p. I-556.

12. Yoon KJ, Kweon IS. Adaptive support-weight approach for correspondence search. Pattern Analysis and Machine Intelligence. 2006; 28(4):650–656.
13. Hosni, A.; Bleyer, M.; Gelautz, M.; Rhemann, C. International Conference on Image Processing. IEEE; 2009. Local stereo matching using geodesic support weights; p. 2093-2096.
14. Rhemann, C.; Hosni, A.; Bleyer, M., et al. Computer Vision and Pattern Recognition. IEEE; 2011. Fast cost-volume filtering for visual correspondence and beyond; p. 3017-3024.
15. Kappes, JH.; Andres, B.; Hamprecht, FA., et al. Computer Vision and Pattern Recognition. IEEE; 2013. A comparative study of modern inference techniques for discrete energy minimization problems; p. 1328-1335.
16. Yang, Q. Computer Vision and Pattern Recognition. IEEE; 2012. A non-local cost aggregation method for stereo matching; p. 1402-1409.
17. Mei, X.; Sun, X.; Dong, W., et al. Computer Vision and Pattern Recognition. IEEE; 2013. Segment-tree based cost aggregation for stereo matching; p. 313-320.
18. Okutomi M, Katayama Y, Oka S. A simple stereo algorithm to recover precise object boundaries and smooth surfaces. International Journal of Computer Vision. 2002; 47(1-3):261–273.
19. Zhang K, Lu J, Lafruit G. Cross-based local stereo matching using orthogonal integral images. Circuits and Systems for Video Technology. 2009; 19(7):1073–1079.
20. Mei, X.; Sun, X.; Zhou, M., et al. International Conference on Computer Vision Workshops. IEEE; 2011. On building an accurate stereo matching system on graphics hardware; p. 467-474.
21. Richardt, C.; Orr, D.; Davies, I., et al. European Conference on Computer Vision. Springer; 2010. Real-time spatiotemporal stereo matching using the dual-cross-bilateral grid; p. 510-523.
22. He, K.; Sun, J.; Tang, X. European Conference on Computer Vision. Springer; 2010. Guided image filtering; p. 1-14.
23. Hosni, A.; Gelautz, M.; Bleyer, M. Pattern Recognition. Elsevier; 2012. Accuracy-efficiency evaluation of adaptive support weight techniques for local stereo matching; p. 337-346.
24. Hosni A, Bleyer M, Gelautz M. Secrets of adaptive support weight techniques for local stereo matching. Computer Vision and Image Understanding. 2013; 117(6):620–632.
25. Klaus, A.; Sormann, M.; Karner, K. International Conference on Pattern Recognition. Vol. 3. IEEE; 2006. Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure; p. 15-18.
26. Bleyer, M.; Rother, C.; Kohli, P., et al. Computer Vision and Pattern Recognition. IEEE; 2011. Object stereojoint stereo matching and object segmentation; p. 3081-3088.
27. Veksler, O. Computer Vision and Pattern Recognition. Vol. 2. IEEE; 2005. Stereo correspondence by dynamic programming on a tree; p. 384-390.
28. Lei, C.; Selzer, J.; Yang, YH. Computer Vision and Pattern Recognition. Vol. 2. IEEE; 2006. Region-tree based stereo using dynamic programming optimization; p. 2378-2385.
29. Bleyer M, Gelautz M. Simple but effective tree structures for dynamic programming-based stereo matching. VISAPP. 2008:415–422.
30. Chambon S, Crouzil A. Similarity measures for image matching despite occlusions in stereo vision. Pattern Recognition. 2011; 44(9):2063–2075.
31. Aschwanden P, Guggenbuhl W. Experimental results from a comparative study on correlation-type registration algorithms. Robust computer vision. 1992:268–289.
32. Brown MZ, Burschka D, Hager GD. Advances in computational stereo. Pattern Analysis and Machine Intelligence. 2003; 25(8):993–1008.
33. Cheng F, Zhang H, Yuan D, Sun M. Stereo matching by using the global edge constraint. Neurocomputing. 2014; 131:217–226.
34. Zhang, H.; Cheng, F.; Yuan, D.; Li, Y.; Sun, M. International Conference on Pattern Recognition. IEEE; 2012. Stereo matching with global edge constraint and graph cuts; p. 372-375.
35. Achanta R, Shaji A, Smith K, et al. Slic superpixels compared to state-of-the-art superpixel methods. Pattern Analysis and Machine Intelligence. 2012; 34(11):2274–2282.
36. Levinshtein A, Stere A, Kutulakos KN, Fleet DJ, Dickinson SJ, Siddiqi K. Turbopixels: Fast superpixels using geometric flows. Pattern Analysis and Machine Intelligence. 2009; 31(12):2290–2297.

37. Moore, AP.; Prince, S.; Warrell, J.; Mohammed, U.; Jones, G. Computer Vision and Pattern Recognition. IEEE; 2008. Superpixel lattices; p. 1-8.
38. Moore, AP.; Prince, SJ.; Warrell, J.; Mohammed, U.; Jones, G. International Conference on Computer Vision. IEEE; 2009. Scene shape priors for superpixel segmentation; p. 771-778.
39. Middlebury steteo matching test bed. 2002. http://vision.middlebury.edu/stereo/eval/
40. Microsoft i2i dataset. 2007. http://research.microsoft.com/en-us/projects/i2i/data.aspx
41. Ma Z, He K, Wei Y, et al. Constant time weighted median filtering for stereo matching and beyond. International Conference on Computer Vision. 2013

## Biographies

**Feiyang Cheng** received the B.S. degree in bio-medical engineering from Tianjin Medical University, China in 2010. He is currently a Ph.D. candidate in Image Research Center, School of Astronautics at Beihang University, China. His research interests include stereo vision, image analysis and computational photograph.

**Hong Zhang** received her Ph.D. degree from Beijing Institute of Technology, China in 2002. She is currently a Professor of Beihang University. She was at the University of Pittsburgh as a visiting scholar from 2007 to 2008. Her research interests include activity recognition, image indexing, object detection and stereo vision.

**Mingui Sun** received his Ph.D. degree in 1989 from the University of Pittsburgh, USA where he is currently a Professor of Neurosurgery, Electrical and Computer engineering, and Bioengineering. His current research interests include advanced biomedical electronic devices, biomedical signal and data processing systems for diet and physical activity assessment.

**Ding Yuan** received her M.Phil. degree and Ph.D. from the Chinese University of Hong Kong in 2004 and 2008 respectively. She is now an assistant professor of Image Research Center at Beihang University, China. Her research interests include stereo vision, 3D reconstruction, camera calibration, and camera's ego-motion estimation.

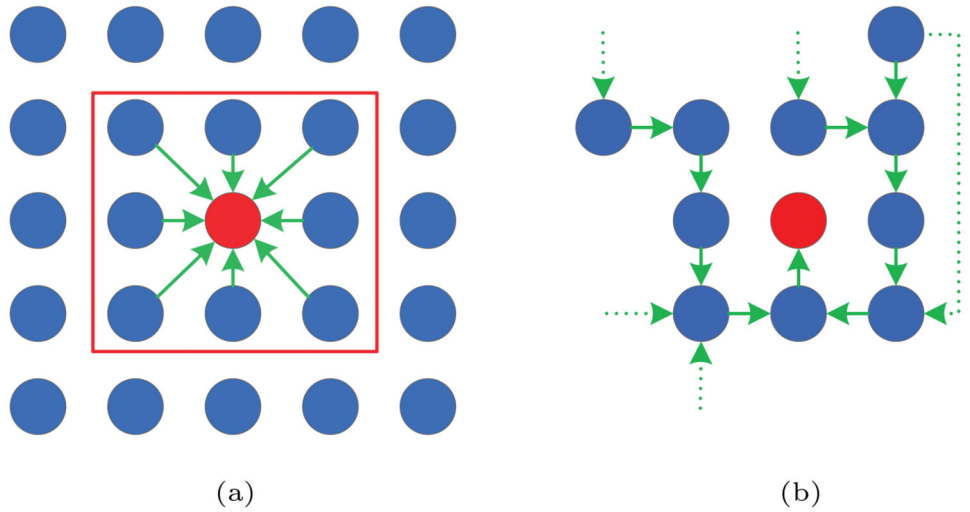


**Figure 1.**

Cost aggregation of the center pixel. (a) Local support region within the square frame: the center pixel gets supports only from its neighboring pixels. (b) Tree: an unique path can be found between the center pixel and each pixel of the image. The dot line denotes that many pixels on the path are not shown here.

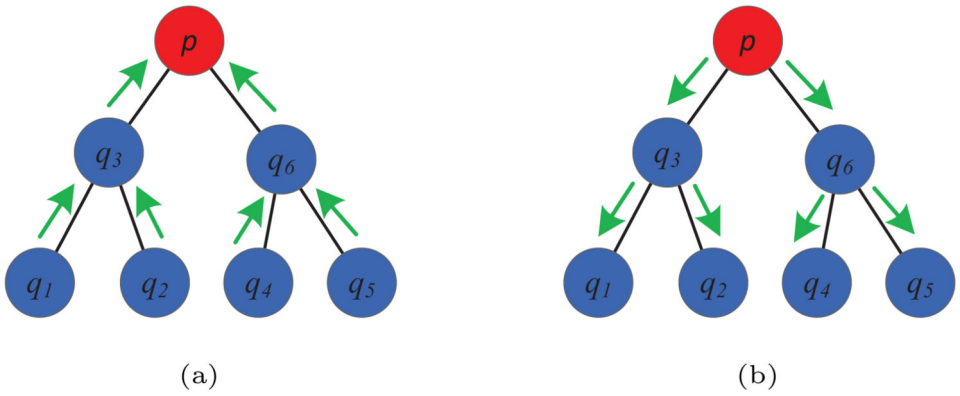


**Figure 2.**

Two-pass non-local cost aggregation. Here, the pixel *p* is treated as the root node of the tree. (a) leaf-to-root pass: The intermediate aggregated cost of each pixel is computed in a straightforward way. (b) root-to-leaf pass: The final aggregated cost for each pixel can be computed by using the results of (a).

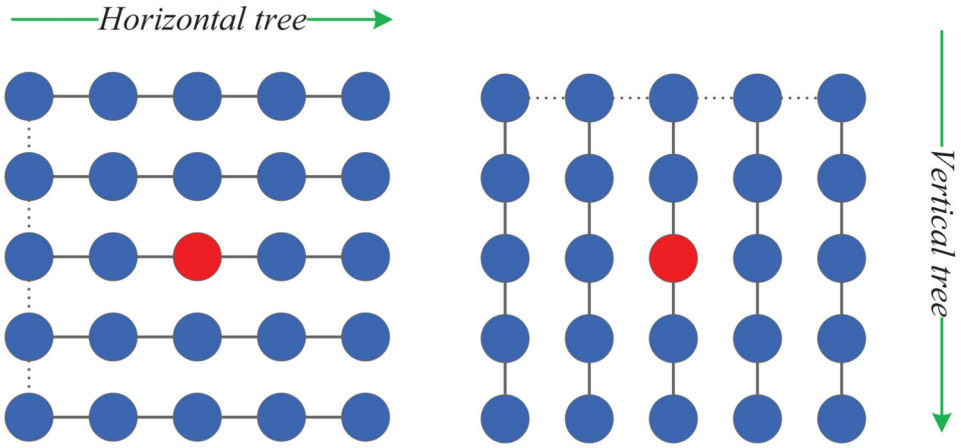


**Figure 3.**

Illustration of the proposed *cross-trees* structure. Left: *horizontal-tree*; right: *vertical-tree*. The edges denoted as dot lines are the extra edges we add to connect different rows or columns for constructing a tree. The weights of these edges are set to be a large number to prevent cost aggregation between different rows or columns in practical applications.

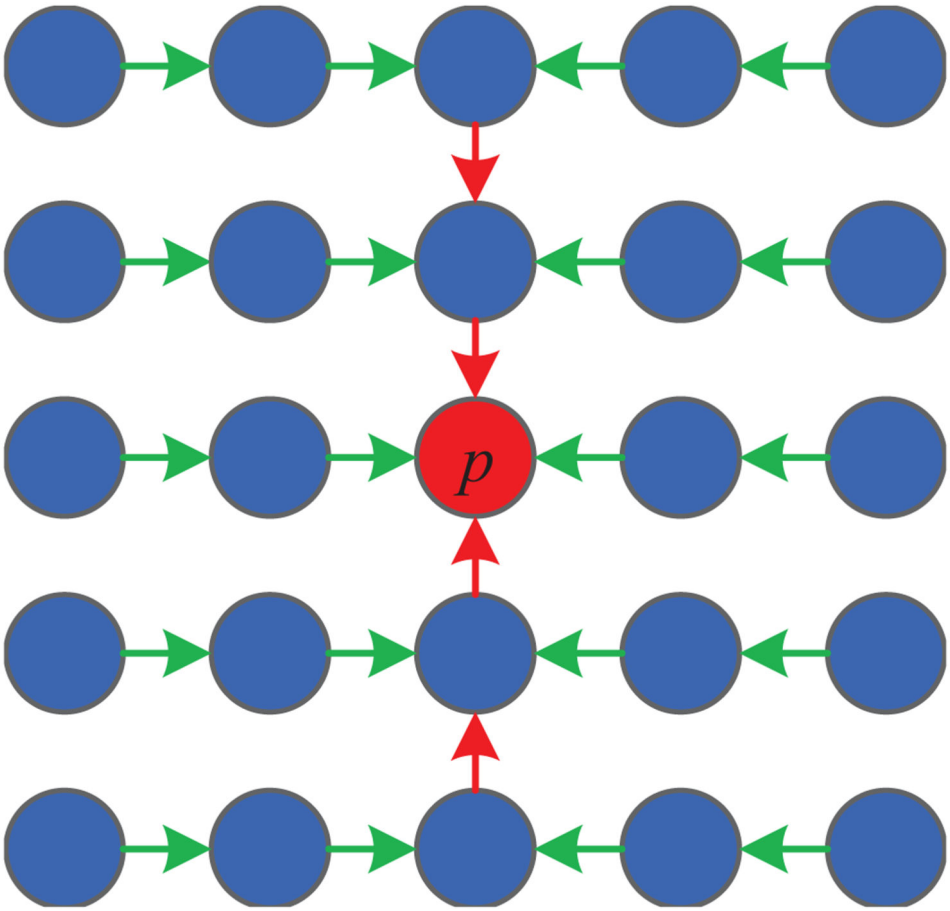


**Figure 4.**

Sequential non-local cost aggregation of a pixel *p*. The horizontal arrows denote the non-local cost aggregation on the *horizontal-tree* and the vertical arrows denote the non-local cost aggregation on the *vertical-tree*.

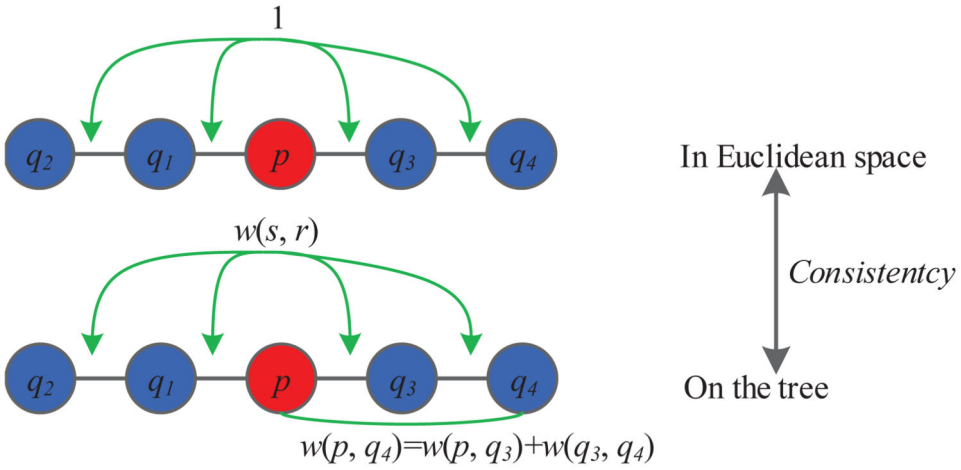


**Figure 5.**
The proximity in Euclidean space and the distance on the tree.


**Figure 6.**

Priors incorporated into the non-local framework. Left: *edge prior*; Right: *superpixel prior*. The edges are detected by using *Canny* edge detector. The superpixels are estimated by using the SLIC algorithm [35].

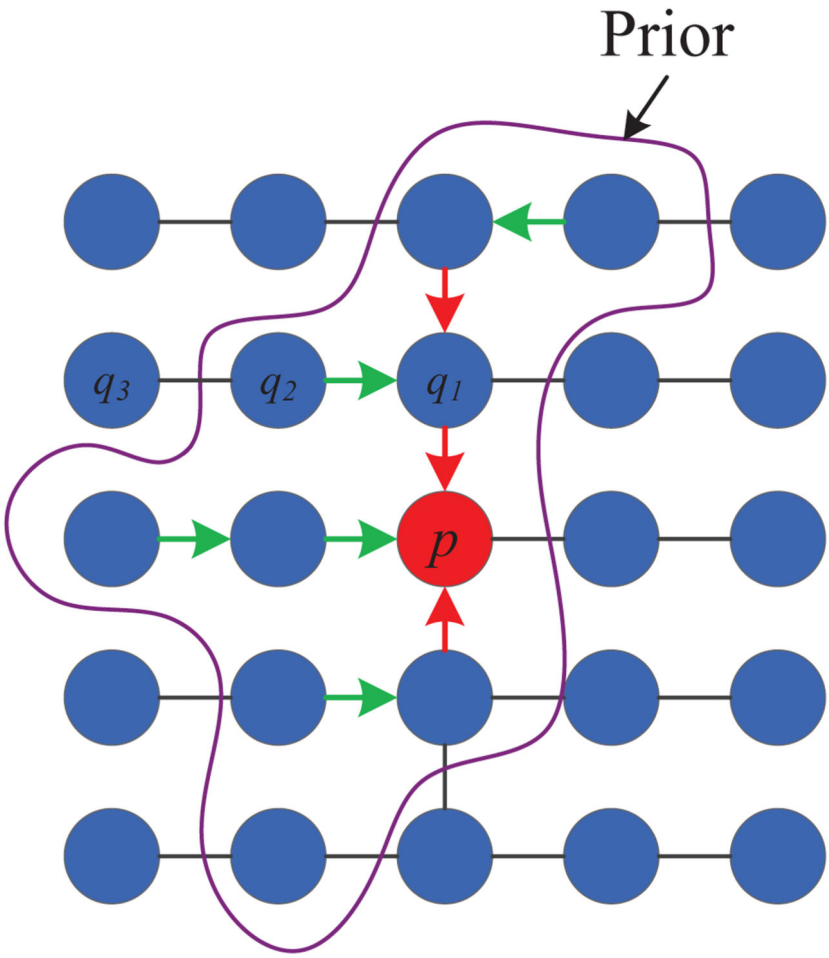


**Figure 7.**

Non-local cost aggregation for pixel *p* on the *cross-trees* structure with a prior. Green arrows represent cost aggregation flow on horizontal scanlines and red arrows represent cost aggregation flow on the vertical scanlines. A *cross-based structure* which consists of these color arrows is the support region of pixel *p*.

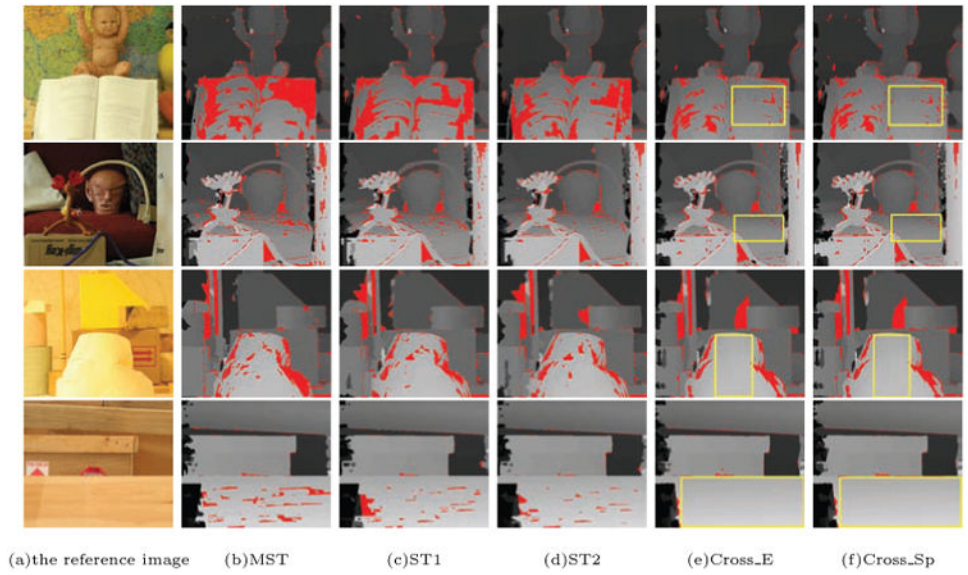


**Figure 8.**

The initial disparity maps without post-processing. The 1$^{st}$ row: *Baby2*; The 2$^{nd}$ row: *Reindeer*; The 3$^{rd}$ row: *Lampshade1*; The *4*$^{th}$ row: *Wood2*. The bad pixels in the disparity maps are marked red. See the regions of (e) and (f) within the yellow rectangular boxes, both our algorithms establish more smooth disparity maps within planar surfaces. Details are best viewed in the electronic version.

**Figure 9.**

Experimental results of the Middlebury data sets. The 1$^{st}$ row: the reference images; The 2$^{nd}$ row: the disparity maps established with ST2. The 3$^{rd}$ row: the disparity maps established with Cross_Sp+WMF. The *4$^{th}$* row: the disparity maps established with GF. The *5$^{th}$* row: the disparity maps established with MST.

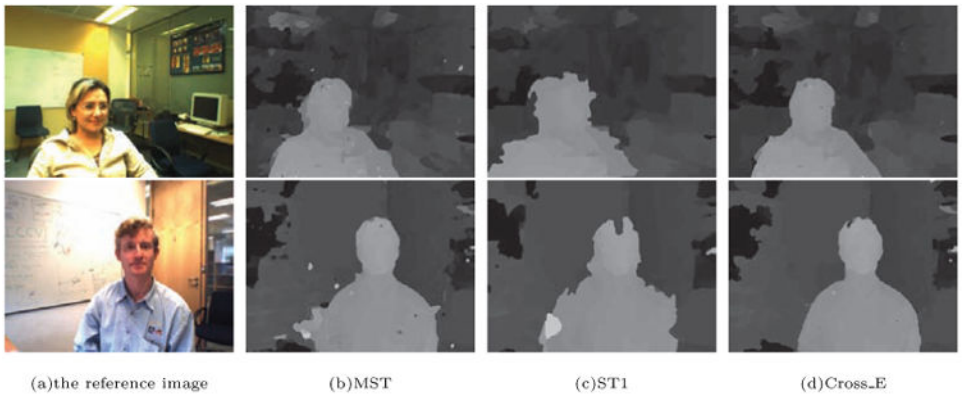


**Figure 10.**

For the stereo video data sets, the disparity maps of a frame are established by using the MST, ST1 and Cross_E algorithms respectively. The 1$^{st}$ row: *Ilkay*; The 2$^{nd}$ row: *Simon*. Cross_E performs much better than the other two algorithms in the foregrounds. The non-local post-processing technique proposed in [12] is employed in all the three algorithms.

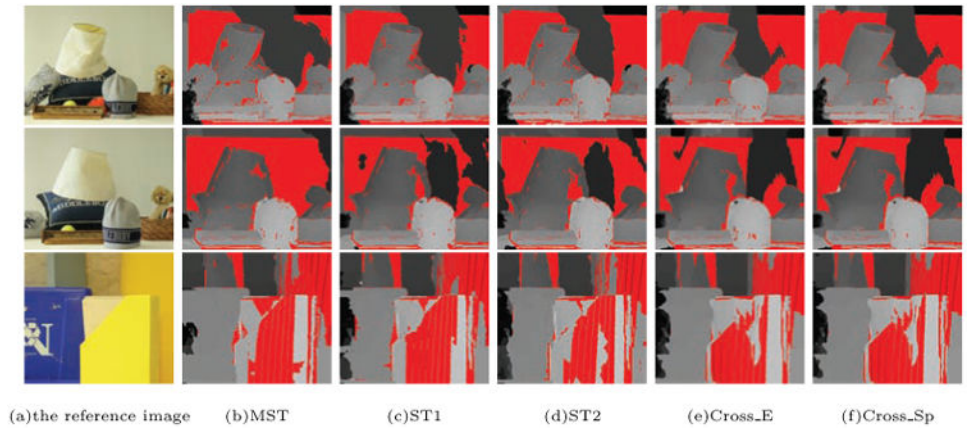


**Figure 11.**

The initial disparity maps without post-processing. The bad pixels in the disparity maps are marked red (see details in electronic version). The 1$^{st}$ row: *Midd1*; The 2$^{nd}$ row: *Midd2*; The 3$^{rd}$ row: *Plastic*.

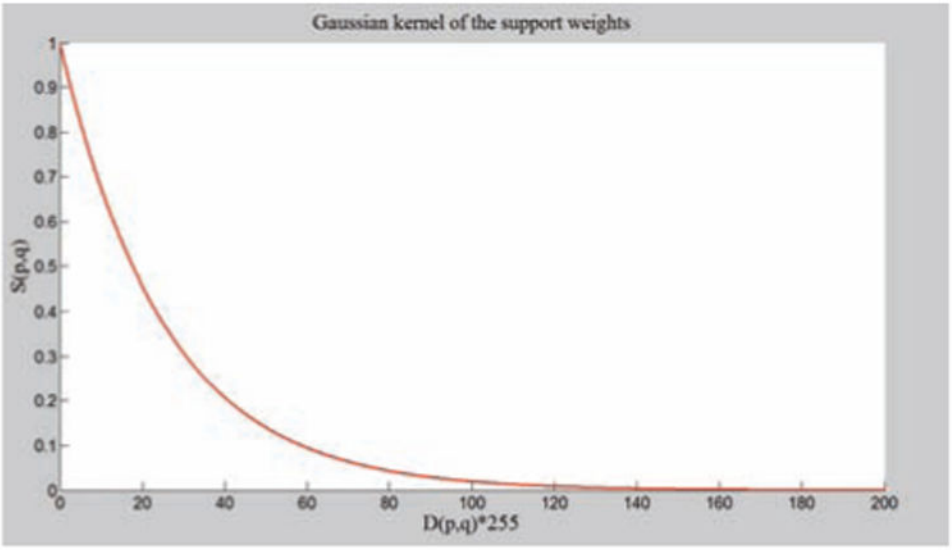


**Figure 12.**

Gaussian kernel of the support weights, $\sigma = 0.1$. The horizontal axis shows the un-normalized distance between two nodes $p$ and $q$ on the constructed tree, and the vertical axis shows the values of support weights. As we can see, the root node can only get effective support weights from leaf nodes which are near enough (for example, $D(p, q) < 80$, $S(p, q) > 0.05$).

**Table 1**

Computational complexities of the tree construction and the non-local cost aggregation.

| Methods | Computational Complexity | |
|---|---|---|
| | Tree Construction | Non-local Cost aggregation |
| MST | $O(m + log_2 m + 2m log_2 n + n)$ | $O(n \cdot l)$ |
| ST1 | $O(2(m + m\alpha(m)))$ | $O(n \cdot l)$ |
| ST2 | $O(4(m + m\alpha(m)))$ | $O(2(n \cdot l))$ |
| Cross_E | $O(2(n - N))$ | $O(2(n \cdot l))$ |
| Cross_Sp | $O(2(n - N))$ | $O(2(n \cdot l))$ |

**Table 2**

Performance evaluation of the cost aggregation accuracy.

| Data | MST | ST1 | ST2 | Cross_E | Cross_Sp |
|---|---|---|---|---|---|
| Tsukuba | $2.26_5$ | $1.85_2$ | $1.83_1$ | $2.23_4$ | $2.14_3$ |
| Venus | $0.69_4$ | $0.64_3$ | $0.42_1$ | $0.71_5$ | $0.60_2$ |
| Teddy | $7.28_2$ | $7.67_4$ | $7.17_1$ | $7.82_5$ | $7.65_3$ |
| Cones | $3.82_4$ | $3.55_3$ | $3.22_1$ | $3.92_5$ | $3.23_2$ |
| Flowerpots | $16.64_5$ | $14.91_4$ | $13.85_1$ | $14.36_2$ | $14.42_3$ |
| Baby1 | $5.57_5$ | $4.52_2$ | $4.26_1$ | $4.63_3$ | $4.63_3$ |
| Baby 2 | $18.95_5$ | $15.20_4$ | $14.29_3$ | $6.33_2$ | $6.13_1$ |
| Baby 3 | $5.13_3$ | $5.07_2$ | $4.89_1$ | $5.23_4$ | $5.27_5$ |
| Art | $10.62_5$ | $10.52_4$ | $10.08_3$ | $8.58_1$ | $8.58_1$ |
| Aloe | $4.67_5$ | $4.40_4$ | $4.37_3$ | $4.02_2$ | $3.52_1$ |
| Books | $9.79_5$ | $9.41_4$ | $8.87_3$ | $8.33_2$ | $8.02_1$ |
| Cloth1 | $0.57_5$ | $0.41_3$ | $0.46_4$ | $0.29_2$ | $0.15_1$ |
| Cloth2 | $3.62_5$ | $3.36_4$ | $2.97_3$ | $1.82_1$ | $1.83_2$ |
| Cloth3 | $2.24_5$ | $1.57_2$ | $1.82_4$ | $1.68_3$ | $1.22_1$ |
| Cloth4 | $1.37_5$ | $1.27_4$ | $1.18_3$ | $0.93_2$ | $0.77_1$ |
| Dolls | $5.88_5$ | $5.17_3$ | $5.50_4$ | $4.43_1$ | $4.46_2$ |
| Lampshade1 | $11.57_5$ | $10.31_1$ | $10.40_2$ | $10.42_3$ | $10.45_4$ |
| Lampshade2 | $14.74_2$ | $21.65_4$ | $21.74_5$ | $14.72_1$ | $14.76_3$ |
| Laundry | $13.56_2$ | $13.83_4$ | $12.05_1$ | $13.91_5$ | $13.70_3$ |
| Moebius | $8.39_5$ | $8.19_4$ | $7.75_1$ | $8.09_3$ | $7.89_2$ |
| Wood1 | $5.14_5$ | $5.08_4$ | $4.19_3$ | $3.71_2$ | $3.65_1$ |
| Wood2 | $3.99_5$ | $3.06_4$ | $2.05_3$ | $0.80_1$ | $0.83_2$ |
| Bowling1 | $22.9_5$ | $18.88_4$ | $16.59_3$ | $15.51_1$ | $15.52_2$ |
| Bowling2 | $12.87_5$ | $11.13_4$ | $10.07_3$ | $8.79_2$ | $8.73_1$ |
| Rocks 1 | $2.60_5$ | $2.34_3$ | $2.41_4$ | $1.48_1$ | $1.61_2$ |

| Data | MST | ST1 | ST2 | Cross_E | Cross_Sp |
|---|---|---|---|---|---|
| Rocks2 | $1.98_5$ | $1.61_3$ | $1.74_4$ | $1.27_1$ | $1.35_2$ |
| Reindeer | $9.15_5$ | $7.73_4$ | $6.51_3$ | $5.67_2$ | $5.39_1$ |
| **Avg. Error** | $7.63_5$ | $7.16_4$ | $6.69_3$ | $\mathbf{5.91_2}$ | $\mathbf{5.80_1}$ |
| **Avg. Rank** | $4.52_5$ | $3.37_4$ | $2.52_3$ | $\mathbf{2.44_2}$ | $\mathbf{2.04_1}$ |

**Table 3**

Performance evaluation on the Middlebury stereo test bed.

| Algorithms | Avg.Rank | Tsukuba | | | Venus | | | Teddy | | | Cones | | | Avg.Error |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | nonocc | all | disc | nonocc | all | disc | nonocc | all | disc | nonocc | all | disc | |
| ST2[17] | **34.5** | **1.25** | **1.68** | **6.69** | **0.20** | **0.30** | **1.77** | **6.00** | 11.9 | 15.0 | 2.77 | 8.82 | 7.81 | **5.35** |
| Cross_Sp+WMF | 41.2 | 1.68 | 1.99 | 7.82 | 0.22 | 0.32 | 2.84 | 6.23 | 11.7 | 14.8 | **2.52** | **7.71** | **7.50** | 5.44 |
| GF[14] | 42.2 | 1.51 | 1.85 | 7.61 | **0.20** | 0.39 | 2.42 | 6.16 | 11.8 | 16.0 | 2.71 | 8.24 | 7.66 | 5.55 |
| MST[16] | 45.4 | 1.47 | 1.85 | 7.88 | 0.25 | 0.42 | 2.60 | 6.01 | **11.6** | **14.3** | 2.87 | 8.45 | 8.10 | 5.48 |

**Table 4**

Performance evaluation on three challenge data sets. The numbers are the percentages of bad pixels in the non-occluded regions of the disparity maps with error threshold 1.

| Data | MST | ST1 | ST2 | Cross_E | Cross_Sp |
|---|---|---|---|---|---|
| Midd1 | 26.05 | 31.37 | 34.37 | 35.66 | 35.41 |
| Midd2 | 45.14 | 28.67 | 35.45 | 31.68 | 31.71 |
| Plastic | 39.76 | 40.82 | 35.84 | 36.87 | 36.71 |
| **Avg.Error** | 36.98 | 33.62 | 35.22 | 34.74 | 34.61 |