Constant Time Weighted Median Filtering for Stereo Matching and Beyond

Ziyang Ma[1]* Kaiming He[2] Yichen Wei[2] Jian Sun[2] Enhua Wu[1]
[1]University of Chinese Academy of Sciences &
State Key Laboratory of Computer Science, Institute of Software, CAS
[2]Microsoft Research Asia

Abstract

Despite the continuous advances in local stereo matching for years, most efforts are on developing robust cost computation and aggregation methods. Little attention has been seriously paid to the disparity refinement.

In this work, we study weighted median filtering for disparity refinement. We discover that with this refinement, even the simple box filter aggregation achieves comparable accuracy with various sophisticated aggregation methods (with the same refinement). This is due to the nice weighted median filtering properties of removing outlier error while respecting edges/structures. This reveals that the previously overlooked refinement can be at least as crucial as aggregation. We also develop the first constant time algorithm for the previously time-consuming weighted median filter. This makes the simple combination "box aggregation + weighted median" an attractive solution in practice for both speed and accuracy.

As a byproduct, the fast weighted median filtering unleashes its potential in other applications that were hampered by high complexities. We show its superiority in various applications such as depth upsampling, clip-art JPEG artifact removal, and image stylization.

1. Introduction

Since the proposition of a stereo framework in a seminal paper [24] a decade ago, most stereo matching methods are following the four-step pipeline: 1) matching cost computation; 2) cost aggregation; 3) disparity optimization; 4) disparity refinement.

Most previous studies are on the first three steps of this framework. Global stereo methods optimize all disparities of all pixels simultaneously, and mostly focus on optimization techniques (step 3). They generates good results but are often time-consuming. By contrast, local stereo methods estimate the disparity of a pixel using simple "Winner-Take-All" in step 3. To harness this simplicity while improving the accuracy, local methods have focused on robust cost computation (step 1) [2, 13, 8] and edge-aware cost aggregation (step 2) [33, 34, 22, 17, 30].

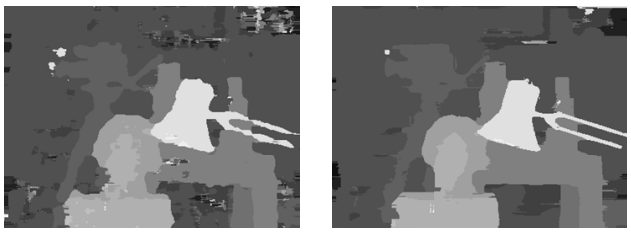
But the impact of *disparity refinement* (step 4) has attracted far less attention in the literature. Traditional practices involve left-right consistency check (*e.g.*, [4]), hole filling (*e.g.*, [2]), and (unweighted) median filtering (*e.g.*, [18]). Recently, Rhemann *et al*. [22] adopt weighted median filtering with bilateral weights [26] to refine local aggregation results. But the high complexity of this filter becomes the timing bottleneck and sacrifices the speed of fast local aggregation[1]. Yang [30] refines the non-local aggregation results by another non-local cost aggregation. In the related field of optical flow, Sun *et al*. [25] discover that the weighted median filtering is a crucial post-processing in each iteration of optimizing the flows.

In this paper, we present a constant time weighted median filter for local stereo refinement. Our method provides new insights for both research and practices of local stereo methods. In terms of research, we discover that with our technique, the refined results of simple box filter cost aggregation [24] can be *comparable* with the refined results of various sophisticated cost aggregation (*e.g.*, [34, 22, 30]). The *comparability* presents on error statistics (detailed in Fig. 4) and also on the superiority of the refined box-filter aggregation results in a fair number of individual cases (detailed in Fig. 5). This is because of the nice weighted median filtering properties of removing outliers while respecting edges/structures. Our discovery indicates that the largely overlooked disparity refinement (step 4) can be at least as crucial as the other three steps.

In practice, this discovery leads to a fast and high-quality stereo solution - simple box aggregation followed by our constant time weighted median refinement. Fig. 1 shows this solution on the benchmark pair "Tsukuba" [24]. In this example, with our refinement, the box aggregation and the more complex guided aggregation [10, 22] are comparably accurate (1.66 vs. 1.62), but the former is much faster

*This work was done when Ziyang Ma was an intern at MSRA.

[1]Actually, in [22] the weighted median filter is only performed near detected depth edges. Even so, its CPU implementation is very slow and dominates the time of the whole algorithm. In [22] it is based on GPU.



(a) box agg. + LR check + hole-filling 46ms, Err. 4.68

(b) guided agg. + LR check + hole-filling 246ms, Err. 2.08

(c) weighted median filtering of (a) **134ms**, Err. 1.66

(d) weighted median filtering of (b) 334ms, Err. **1.62**

Figure 1. With our weighted median filtering refinement, simple box-filter aggregation [24] can be comparably good as sophisticated aggregation [22] using guided filters [10]. (a) Box-filter aggregation followed by left-right check [4] and hole-filling [2]. (b) Guided-filter aggregation followed by left-right check and hole-filling. (c) Our weighted median filtering refinement on (a). (d) Our weighted median filtering refinement on (c). The error is evaluated on bad pixel percentage. The running time is reported on a single-core CPU with C++ implementation.

(134ms vs. 334ms).

The above discussion is made possible by our fast *constant time* algorithm for weighted median filtering. This algorithm is driven by the recent progress on fast median filtering [20, 3, 15], fast algorithms [7, 19, 21, 31] for bilateral filtering [26], and other fast edge-aware filtering [10, 9]. We show that (in Sec. 2) the above fast median/bilateral filtering algorithms can be unified in a framework of high-dimensional filtering. This framework reveals that the (unweighted) median filter can be implemented as a box filter in a high-dimensional space. Thus we can replace this box filter with constant time edge-aware filters [10, 9] for weighted median filtering. Our algorithm is simple and easy to implement.

As a byproduct, our fast algorithm for weighted median filtering allows us to treat it as a general image filter and study its behaviors in various applications. In this paper, we show its high-quality results in depth upsampling, clip-art JPEG artifact removal, and image stylization. We believe our algorithm is potential for many other applications due to its simplicity and nice properties.

In sum, our paper makes these main contributions: (1) We discover that the disparity refinement (step 4) can be at least as important as the other steps in stereo matching. The combination of simple box aggregation and our weighted median refinement achieves good accuracy and very fast speed. (2) We present the first constant time algorithm for weighted median filtering. It provides a practically fast solution to stereo matching and various other applications.

We publish the Matlab code of the constant time weighted median filter on our website[2].

2. Constant Time Weighted Median Filtering

We first introduce our constant time algorithm for weighted median filtering. Then we demonstrate its effect for stereo refinement (Sec. 3) and other applications (Sec. 4)

2.1. Median Filtering

The (unweighted) median filter [14] has been long considered as a way of removing "outliers" like salt-and-pepper noise. This filter replaces the value of a pixel with the median of its neighbors. For discrete signals (*e.g.*, disparity/intensities), this median can be computed from a histogram $h(\mathbf{x}, \cdot)$ that calculates the population around the position $\mathbf{x} = (x, y)$:

$$h(\mathbf{x}, i) = \sum_{\mathbf{x}' \in \mathcal{N}(\mathbf{x})} \delta(V(\mathbf{x}') - i). \tag{1}$$

Here, $\mathcal{N}(\mathbf{x})$ is a local window (usually a box) near $\mathbf{x}$, V is the pixel value, i is the discrete bin index, and $\delta(\cdot)$ is the Kronecker delta function: $\delta(\cdot)$ is 1 when the argument is 0, and is 0 otherwise. It is straightforward to pick the median value through accumulating this histogram.

2.2. Weighted Median Filtering

The unweighted median filter treats each neighbor equally, and may lead to morphological artifacts like rounding sharp corners and removing thin structures (*e.g.*, Fig. 2(c)). To address this problem, the weighted median filter [25, 22] has been introduced. The pixels are weighted in the local histograms:

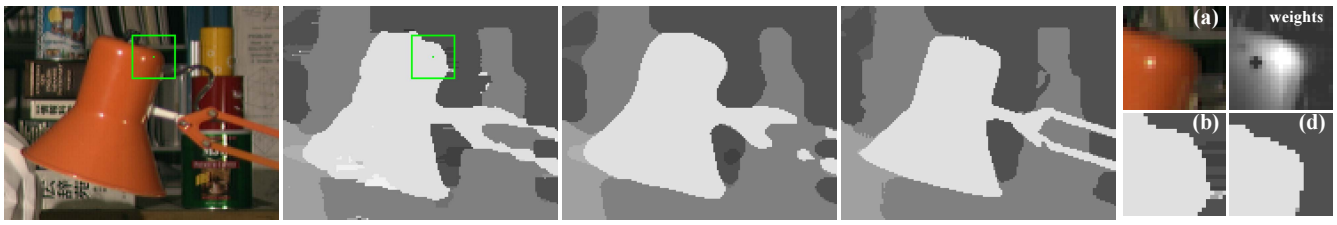
$$h(\mathbf{x}, i) = \sum_{\mathbf{x}' \in \mathcal{N}(\mathbf{x})} w(\mathbf{x}, \mathbf{x}') \delta(V(\mathbf{x}') - i). \tag{2}$$

Here the weight $w(\mathbf{x}, \mathbf{x}')$ depends on an image I that can be different from V, *e.g.*, the left image in stereo. In [25, 22] w is the bilateral weight [26] that suppresses the pixels with different color from the center pixel. As in the unweighted case, the median value is obtained by accumulating this histogram.

2.3. A Constant Time Algorithm

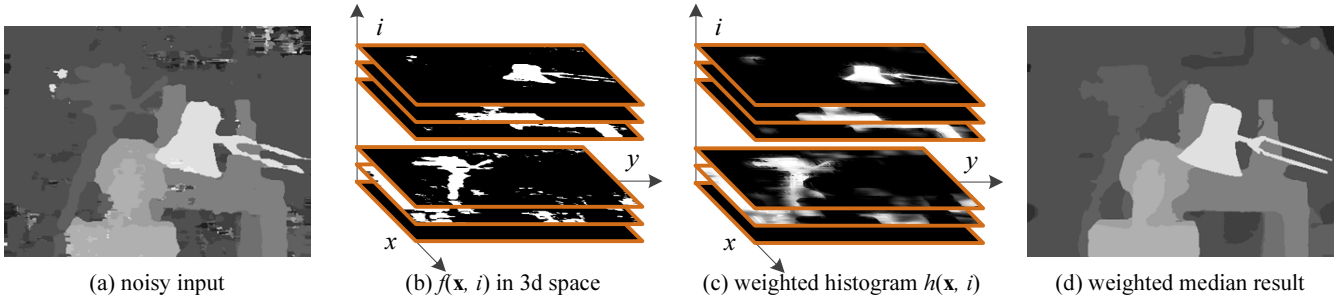
A brute-force implementation of Eqn.(2) can be time-consuming: its complexity is $O(r^2)$ per pixel with the support radius r. This high complexity largely limits the applications and studies of this filter. Next we provide a constant

[2] research.microsoft.com/en-us/um/people/kahe/



(a) left image (b) box agg.+LR check+hole-filling (c) median filtering of (b) (d) weighted median filtering of (b) (e) zoom-in

Figure 2. Unweighted and weighted median filtering. (a) The left image of a stereo pair. (b) The result of box aggregation + left-right check + hole-filling. (c) The unweighted median filtering result of (b). (d) The weighted median filtering result of (b). On the right we show a zoom-in region and a filtering kernel.



(a) noisy input (b) $f(\mathbf{x}, i)$ in 3d space (c) weighted histogram $h(\mathbf{x}, i)$ (d) weighted median result

Figure 3. Constant time weighted median filtering. (a): the map to be filtered (V in Eqn.(2)). (b): the 3-dimensional signal $f(\mathbf{x}, i)$ in Eqn.(3). (c): A guided filter is performed on $f(\mathbf{x}, i)$ for each fixed i. (d) The median value is taken from the histogram of each pixel and gives the final result.

time algorithm (*i.e.*, $O(1)$ per pixel) for weighted median filtering.

We first discuss the unweighted case Eqn.(1). We consider the argument $(\mathbf{x}, i)$ as 3D coordinates where $\mathbf{x}$ represents the 2D spatial coordinates and i represents a 1D *range* coordinate (disparity/intensity). Define a signal $f(\mathbf{x}, i)$ in this 3-dimensional space:

$$f(\mathbf{x}, i) \triangleq \delta(V(\mathbf{x}) - i). \tag{3}$$

Then the computation of the unweighted histogram Eqn.(1) is essentially a 2D box filtering of f in the spatial domain:

$$h(\mathbf{x}, i) = \sum_{\mathbf{x}' \in \mathcal{N}(\mathbf{x})} b(\mathbf{x}, \mathbf{x}') f(\mathbf{x}', i), \tag{4}$$

where b is a box kernel. The computation of (4) can be simply done by performing a 2D box filter on $f(\mathbf{x}, i)$ for each fixed i. Because box filtering can be efficiently performed using integral images [5, 27] or moving sums in $O(1)$ time, the unweighted median filter is $O(1)$ time. The existing $O(1)$ median filtering algorithms [3, 20] can be viewed as an implementation of this formulation.

For weighted median filtering, we can simply replace the box filter $b(\mathbf{x}, \mathbf{x}')$ with any other edge-aware weight $w(\mathbf{x}, \mathbf{x}')$, *e.g.*, the bilateral filter [26], the guided filter [10], or the domain transform filter [9]. To compute the weighted histogram Eqn.(2), we only need to perform the specified edge-aware filter on $f(\mathbf{x}, i)$ for each fixed i. If the edge-aware filter is $O(1)$ time, the resulting weighted median filter is $O(1)$ time. Fig. 3 illustrates our algorithm.

Choosing Filter Weights Our constant time weighted median filter is compatible with various weights - it only requires the corresponding edge-aware filter to be constant time. In this paper, we consider the following $O(1)$ time edge-aware filters: the $O(1)$ time solutions [21, 31] to the bilateral filter [26], the guided filter [10], and the domain transform filter [9]. The $O(1)$ bilateral filter [21, 31] is much slower than the other two, unless the range is aggressively quantized. But the quantization impacts the quality. On the contrary, both the guided filter and the domain transform filter require no quantization/coarsening. We notice that the domain transform filter is more suitable for image smoothing, but less so for preserving detailed structures [11]. We also find the accuracy of using the domain transform weights for stereo refinement is not as good as using the guided filter weights. For these reasons, throughout this paper we use the guided filter weights as the weights for median filtering. The guided filter has edge-aware kernels similar to the bilateral filter [10] (Fig. 2 right).

2.4. Relations to Previous Methods

We show that various fast algorithms for bilateral filtering [7, 19, 21, 31] and median filtering [14, 3, 20, 15] can be unified in the same framework of high-dimensional filtering. This provides more insights on the relationship between our method and existing ones.

We consider a more general *3-dimensional* filtering for-

	method	spatial	range	order
Bilateral	Durand & Dorsey, *SIGGRAPH* '02 [7]	FFT Gaussian	conv.	range ⇒ spatial
	Paris & Durand, *ECCV* '06 [19]	FFT Gaussian	FFT Gaussian	simultaneously
	Porikli, *CVPR* '08 [21]	$O(1)$ box	conv.	spatial ⇒ range
	Yang *et al*., *CVPR* '09 [31]	$O(1)$ Gaussian [6]	conv.	range ⇒ spatial
Median	Huang *et al*., *TASSP* '79 [14]	$O(r)$ box	median	spatial ⇒ range
	Perreault & Hébert, *TIP* '07 [20]	$O(1)$ box	median	
	Kass & Solomon, *SIGGRAPH* '10 [15]	$O(1)$ Gaussian [6]	conv. + median	
	Our weighted median	$O(1)$ **guided** [10]	median	

Table 1. A summary of various fast bilateral/median filtering algorithms. The 'spatial/range' columns show the spatial/range operations applied. The 'order' column shows the order of these operations, *e.g*., 'range ⇒ spatial' means the range operation goes first. In the 'range' column, 'conv.' means arbitrary convolutions (usually Gaussian), and 'median' means taking the median from the histogram of each pixel.

mulation:

$$h(\mathbf{x}, i) = \sum_{\mathbf{x}' \in \mathcal{N}(\mathbf{x})} \sum_{i' \in \mathcal{N}(i)} k(\mathbf{x}, \mathbf{x}', i, i') f(\mathbf{x}', i'). \quad (5)$$

If the 3-dimensional kernel $k(\mathbf{x}, \mathbf{x}', i, i') = w(\mathbf{x}, \mathbf{x}')\delta(i - i')$, Eqn.(5) becomes the computation of a histogram; if $k(\mathbf{x}, \mathbf{x}', i, i') = k_s(\mathbf{x} - \mathbf{x}')k_r(i - i')$ for some spatial/range kernels k_s and k_s (often Gaussian), it has been shown [19] that Eqn.(5) is exactly the bilateral filter (plus some proper manipulations, namely, *division* and *slicing* [19]).

The 3-dimensional filtering in Eqn.(5) is *linear*, *separable*, and *commutative* (between spatial/range). We show that various existing fast algorithms for bilateral/median filtering can all be viewed in this formulation (though they can be originally derived from other ways). They mainly differ in the linear filtering operations and the computation orders (due to the separability and commutativity). We roughly summarize various fast algorithms for bilateral/median filtering in Table 1[3].

Strictly speaking, the $O(1)$ complexity in all the above methods involves a constant scale which is the number of the discrete values (*e.g*., the disparity/intensity range). But in conventional median/bilateral filtering literatures [3, 20, 21, 31], the argument in the complexity $O(\cdot)$ is often considered as the kernel radius r. So this constant has been ignored in the discussion.

2.5. Filter Properties

The weighted median filter inherits some desired properties from both median filtering and edge-aware averaging filtering (the guided filter).

The weighted median filter is capable of removing "outlier" noise as a median filter. This is particularly desired for refining local stereo aggregation results, which may generate erroneous disparity values in arbitrary ranges (see Fig. 1(a,b) and Fig. 2(b)). The edge-aware *averaging* filters like [26, 10, 9] are not suitable for this noise (in plane regions, this is analogous to box/Gaussian filers vs. traditional median filters).

On the other hand, the weighted median filter is edge-aware. This is in contrast to the unweighted median filter. Due to the edge-aware weights that suppress the impact of the pixels in different colors (Fig. 2 (e)), the weighted median filter is capable of capturing the strong edges, sharp corners, and thin structures from the image I (see Fig. 2(c)). The unweighted median filter does not have this good property and produces morphological artifacts (Fig. 2(b)). Actually, the unweighted median filter is *blind* to the image I and can merely access the noisy map V.

3. Experiments on Stereo Refinement

Thanks to the constant time algorithm for weighted median filtering, it is feasible for us to study its performance for refining local stereo results.

3.1. Experimental Settings

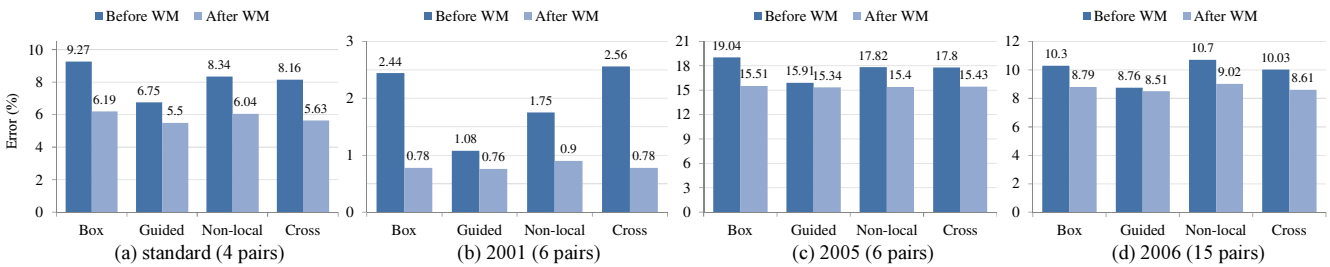
For fair comparisons and for clearly understanding the refinement step, in all experiments throughout this paper we strictly follow a local stereo matching pipeline [22] described as below:

(1): *cost computation*. We use exactly the same cost as [22]. It is a blending of truncated color difference and truncated gradient difference.

(2): *cost aggregation*. We use the cost aggregation as [22]. We have tested four kinds of $O(1)$ time aggregation: the box filter, the guided filter [10] (adopted in [22]), the variable cross filter [34], and the non-local filter [30]. For box filters we use a 9x9 support. The parameters of all other filters are as recommended in these papers.

(3): *disparity optimization*. We simply choose the disparity with the minimal aggregated cost for each pixel (Winner-Take-All).

[3]For a clearer summarization, in Table 1 we have ignored some operations applied for individual methods, including division, slicing, downsampling/quantizing/coarsening, and upsampling.



(a) standard (4 pairs) (b) 2001 (6 pairs) (c) 2005 (6 pairs) (d) 2006 (15 pairs)

Figure 4. Error rates on four Middlebury datasets before and after WM, using four different aggregation filters.

(4-i): *disparity refinement* (i). We do left-right check and simple hole-filling as in [22].

(4-ii): *disparity refinement* (ii). On the resulting disparity map of (4-i), we apply the weighted median (**WM**) filter. The parameters of the guided filter used in this WM are fixed as $\epsilon = 0.01^2$, $r = \max(wid, hei)/40$. A 3x3 unweighted median filter is finally applied to remove a few spikes.

This pipeline follows the public Matlab code[4] provided by [22]. The main modifications are that we apply our WM filter for refinement, and we investigate four filters in cost aggregation. The purpose of using this pipeline is for a better understanding of the roles of aggregation *vs*. refinement. It is possible to improve the quality by, *e.g.*, using advanced costs, fine-tuning the filter parameters, incorporating median filters for intermediate-processing, or improving the simple hole-filling. But these are not our focus and not considered in this paper.

All the experiments are implemented in C++ and run on a PC with 8G RAM and an Intel Xeon 2.83GHz CPU using a single thread. In this implementation, the weighted median filter takes about 60ms per mega-pixel per disparity (see [11] for more technical details about implementing the guided filter).

Our experiments are on the Middlebury stereo benchmark [1]. We evaluate all the pairs that have ground-truths available: the standard 4 pairs, the 2001 set (6 pairs), the 2005 set (6 pairs), and the 2006 set (21 pairs) [23, 12]. We consider the error metric as the percentage of bad pixels with error threshold 1.

We notice that local methods are less suitable for textureless images. The error of these images would dominate the error statistics. To remove this bias, we first run the method of [22] on all image pairs and ignore those with error rate $> 20\%$ (6 pairs in 2006) in the remaining experiments. This leaves us in total 31 pairs for evaluation.

3.2. Results

Fig. 4 shows the error rates before and after weighted median filtering. In each dataset we test four different filters for cost aggregation. We find that before WM, the error rates of different aggregation methods are significant different. Before WM, the guided filter aggregation is the most competitive in all four datasets, whereas the box filter aggregation is generally poor. But we discover that after WM the error rates of different aggregation methods are very comparable (especially in the 2001, 2005, and 2006 sets). Typically, the box filter aggregation with WM can be almost as good as the other sophisticated aggregation methods with WM. The comparability not only presents on the statistics (Fig. 4), but also on the fact that the simple "box aggregation + WM" is more accurate than the sophisticated "guided aggregation + WM" in a number of cases (12 out of all 31 pairs). Fig. 5 shows some examples. This discovery reveals that the disparity refinement step is at least as important as the cost aggregation.

Analysis The above discovery can be explained by the capability of "outlier" removal of the WM. Before WM, the box filter aggregation produces more outliers than the other sophisticated aggregation methods (see Fig. 1(a,b) and Fig. 5(b,c)). These outliers can be reliably removed by median filtering in a sufficiently large support, if only the "inliers" in this large support are dominant. But a large box support leads to morphological artifacts (Fig. 2(c)) and often degrades the results. This issue is nicely addressed by median filtering in a large *edge-aware* support (Fig. 2(d)). It can remove more outliers by a large support without introducing morphological artifacts. So even though the box filter aggregation produces more outliers, they can still be removed (unless in the support the outliers are dominant).

3.3. A Fast Practical Solution

The above discovery suggests a very fast solution in practice - combining the box filter aggregation with WM refinement. This solution harnesses the fast speed of box filter aggregation, and almost does not sacrifice accuracy due to the WM. Fig. 6 shows the error rates vs. the CPU running time. It is clear that the "box aggregation + WM" is a compelling choice, considering both speed and accuracy.

GPU implementation The box aggregation and the WM can both be easily implemented on GPU. Because our WM uses guided filtering weights and the algorithm of the guid-

[4] www.ims.tuwien.ac.at/research/costFilter/

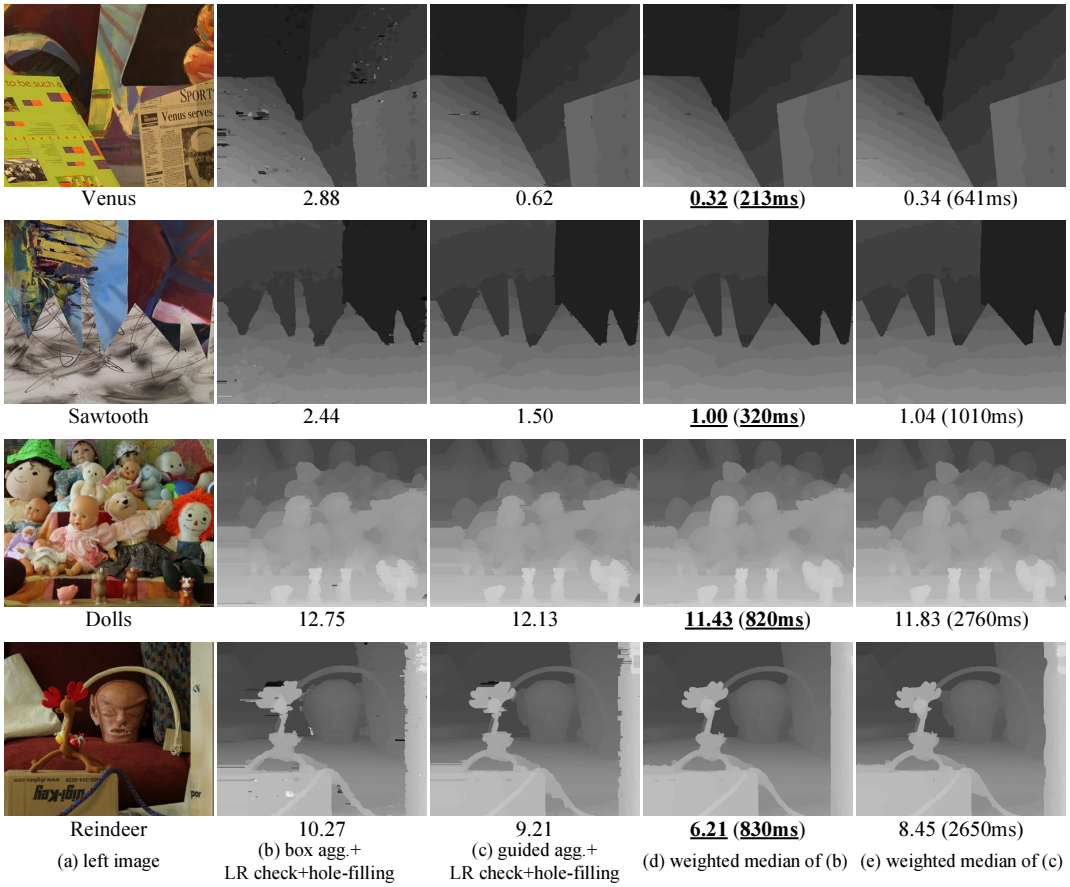


Figure 5. Several benchmark examples where the simple "box aggregation + WM" *outperforms* the complex "guided aggregation + WM". For each image we show the error rate and the running time.

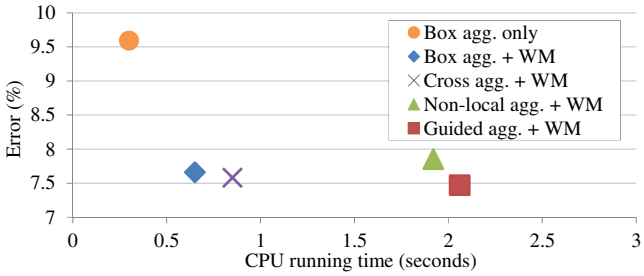


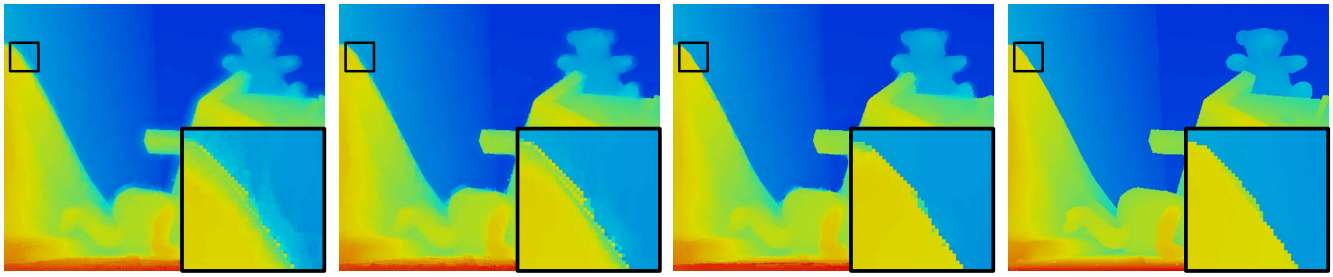
Figure 6. Error rates and CPU running time of four different aggregation methods. The results are averaged on all 31 pairs of all four datasets.

Ours and previous work	Avg. error	GPU time
NonLocalFilter [30]	5.48†	n/a
Guided agg. + **WM**	5.50	**54ms**
CostFilter [22]	5.55†	65ms†
Cross agg. + **WM**	5.63	n/a
Non-local agg. + **WM**	6.04	n/a
Box agg. + **WM**	6.19	**22ms**
VariableCross [34]	7.60†	n/a

Table 2. The error rates and GPU running time on the standard four pairs in Middlebury benchmark. (†These numbers are reported by original papers. Their pipelines can be different from ours, so the error can be different even using the same cost aggregation.)

ed filter is a series of box filters (see [10]), our WM is naturally GPU-friendly. We have implemented "box aggregation + WM" and "guided aggregation + WM" on a GeForce GTX580 GPU (512 CUDA cores, 1.5GB VRAM). Table 2 shows the GPU time and error rates in the standard four pairs. We are not aware of any GPU implementation of the non-local aggregation [30] and cross aggregation [34], so their performance on GPU remains unknown.

To the best of our knowledge, the previous fastest GPU implementation is in [22]. It takes on average 65ms in the standard four pairs (reported on GeForce GTX480, 480 CUDA cores, 1.5GB VRAM). Table 2 shows that our GPU implementation of "box aggregation + WM" takes 22ms, about 3x faster than [22]. Considering the error rates in Table 2 and Fig. 4 & 6, we believe the "box aggregation + WM" is a very fast practical solution (in both CPU and GPU) that slightly trades off accuracy.



(a) Joint Bilateral, Err. 14.43 (b) Depth SR, Err. 10.19 (c) Ours, Err. **8.58** (d) Ground truth

Figure 7. A visual example of depth upsampling (8×), displayed using colormaps. (a) Joint bilateral filter [16]. (b) Depth super-resolution (SR) [32]. (c) Ours. (d) Ground-truth.

4. More Applications

The weighted median filter is not only applicable for stereo refinement. It is a general purpose filter that is potential in various applications. In this section we demonstrate its power on depth upsampling, clip-art JPEG artifact removal, and image stylization.

4.1. Depth Upsampling

The problem is to upsample a low resolution depth image with the guidance of a registered high resolution color image [16, 32]. This is a practical problem for stereo matching in high resolution images. The method in [16] uses a joint bilateral filter, which is an edge-aware *averaging* filter. The method in [32] uses robust truncated costs and a post quadratic interpolation. We compare both methods with ours WM. In our method, we simply upscale the low resolution depth image to the high resolution using bilinear interpolation, and apply the WM.

We evaluate on the 2×, 4× and 8× downsampled images of the four standard Middlebury pairs [1] with ground-truth high resolution depth available. Table 3 shows the quantitative results. It is clear that our simple method outperforms the two competitors in all cases. Fig. 7 shows a visual example. We see that the WM avoids "halos" near depth edges and better preserves the edge profiles.

Algorithm	Tsukuba	Venus	Teddy	Cones
Bilinear	6.40	3.22	13.55	16.63
[16]	5.15	2.54	14.43	16.23
[32]	4.53	1.25	10.19	11.53
Our method	**4.35**	**1.09**	**8.58**	**9.34**

Table 3. Bad pixel percentage with error threshold 1 for 8× depth upsampling on the standard four Middlebury pairs [1].

4.2. Clip-Art JPEG Artifact Removal

The JPEG compression may lead to artifacts near step edges due to the frequency quantization. This is an annoying problem typically for clip-art cartoon images as in Fig. 8 (a,b). Such artifacts are not Gaussian-alike noise. Even worse, they are near strong edges that are challenging for most edge-aware filters. A recent state-of-the-art solution [28] optimizes a complex L_0-regularized energy to address this problem.

We compare our WM with the method of [28] (Fig. 8). Despite the simplicity, the WM better recovers the piecewise constant colors and the strong step edges, with almost no smoothing near edges. We observed such improvements on all test images on the website of [28] (see supplementary materials).

4.3. Image Stylization

The capability of removing outliers while preserving edges/structures is suitable for removing textures and generating piecewise constant images. This is particularly favored in applications like image stylization or image abstraction. In Fig. 9 we show a comparison with a recent sophisticated method in [29]. We find the simple WM can produce compelling results.

5. Conclusion

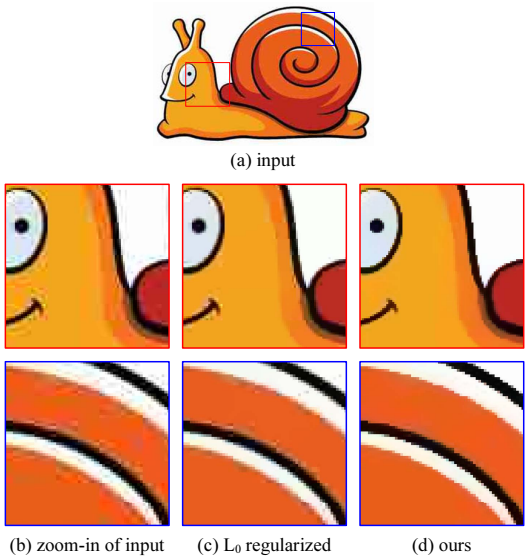
We have proposed the first constant time weighted median filtering algorithm for stereo refinement and other applications. For stereo, we discover that the refinement can be as important as other steps. We expect this work can attract more attention on stereo refinement, in company with the recent intensive efforts on aggregation [33, 34, 22, 30]. For general image filtering, we believe the simple weighted median filtering is very potential in various applications. We will develop more applications in the future.

References

[1] http://vision.middlebury.edu/stereo.

[2] S. Birchfield and C. Tomasi. A pixel dissimilarity measure that is insensitive to image sampling. *TPAMI*, pages 401–406, 1998.

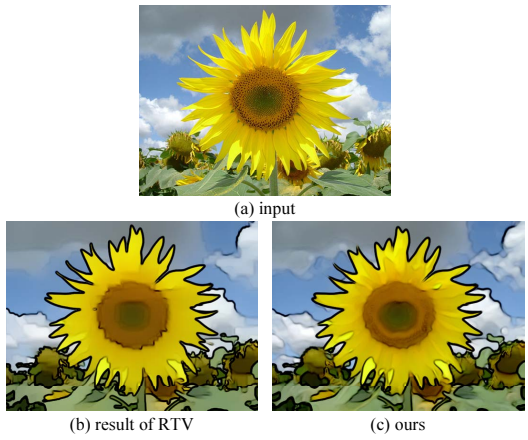
[3] D. Cline, K. B. White, and P. K. Egbert. Fast 8-bit median filtering based on separability. In *ICIP*, 2007.



(a) input

(b) zoom-in of input (c) L_0 regularized (d) ours

Figure 8. Clip-art JPEG compression artifact removal result. (a) Input. (b) Zoom-in of the JPEG artifacts. (c) The L_0 regularized method [28]. (d) Ours.



(a) input

(b) result of RTV (c) ours

Figure 9. Image stylization. (a) Input. (b) Results of Relative Total Variation [29]. (c) Ours. The edges are imposed as in [28].

[4] S. D. Cochran and G. Medioni. 3-d surface description from binocular stereo. *TPAMI*, pages 981–994, 1992.

[5] F. C. Crow. Summed-area tables for texture mapping. In *SIGGRAPH*, pages 207–212, 1984.

[6] R. Deriche. Recursively implementing the gaussian and its derivatives, 1993.

[7] F. Durand and J. Dorsey. Fast bilateral filtering for the display of high-dynamic-range images. In *SIGGRAPH*, pages 257–266, 2002.

[8] N. Einecke and J. Eggert. A two-stage correlation method for stereoscopic depth estimation. In *International Conference on Digital Image Computing: Techniques and Applications*, pages 227–234, 2010.

[9] E. S. Gastal and M. M. Oliveira. Domain transform for edge-aware image and video processing. In *SIGGRAPH*, page 69, 2011.

[10] K. He, J. Sun, and X. Tang. Guided image filtering. In *ECCV*, 2010.

[11] K. He, J. Sun, and X. Tang. Guided image filtering. *TPAMI*, 2013.

[12] H. Hirschmuller and D. Scharstein. Evaluation of cost functions for stereo matching. In *CVPR*, 2007.

[13] H. Hirschmuller and D. Scharstein. Evaluation of stereo matching costs on images with radiometric differences. *TPAMI*, pages 1582–1599, 2009.

[14] T. Huang, G. Yang, and G. Tang. A fast two-dimensional median filtering algorithm. *IEEE Transactions on Acoustics, Speech and Signal Processing*, pages 13–18, 1979.

[15] M. Kass and J. Solomon. Smoothed local histogram filters. In *SIGGRAPH*, page 100, 2010.

[16] J. Kopf, M. Cohen, D. Lischinski, and M. Uyttendaele. Joint bilateral upsampling. In *SIGGRAPH*, page 96, 2007.

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

[18] K. Mühlmann, D. Maier, J. Hesser, and R. Männer. Calculating dense disparity maps from color stereo images, an efficient implementation. *IJCV*, pages 79–88, 2002.

[19] S. Paris and F. Durand. A fast approximation of the bilateral filter using a signal processing approach. In *ECCV*, 2006.

[20] S. Perreault and P. Hébert. Median filtering in constant time. *TIP*, pages 2389–2394, 2007.

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

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

[23] D. Scharstein and C. Pal. Learning conditional random fields for stereo. In *CVPR*, 2007.

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

[25] D. Sun, S. Roth, and M. Black. Secrets of optical flow estimation and their principles. In *CVPR*, 2010.

[26] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *ICCV*, 1998.

[27] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001.

[28] L. Xu, C. Lu, Y. Xu, and J. Jia. Image smoothing via L0 gradient minimization. In *SIGGRAPH Asia*, page 174, 2011.

[29] L. Xu, Q. Yan, Y. Xia, and J. Jia. Structure extraction from texture via relative total variation. In *SIGGRAPH Asia*, page 139, 2012.

[30] Q. Yang. A non-local cost aggregation method for stereo matching. In *CVPR*, 2012.

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

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

[33] K. Yoon and I. Kweon. Adaptive support-weight approach for correspondence search. *TPAMI*, pages 650–656, 2006.

[34] K. Zhang, J. Lu, and G. Lafruit. Cross-based local stereo matching using orthogonal integral images. *TCSVT*, pages 1073–1079, 2009.