Robust Piecewise-Constant Smoothing: M-Smoother Revisited

Linchao Bao and Qingxiong Yang

`http://www.cs.cityu.edu.hk/~qiyang/`

Abstract—A robust estimator, namely M-smoother, for piecewise-constant smoothing is revisited in this paper. Starting from its generalized formulation, we propose a numerical scheme/framework for solving it via a series of weighted-average filtering (*e.g.*, box filtering, Gaussian filtering, bilateral filtering, and guided filtering). Because of the equivalence between M-smoother and local-histogram-based filters (such as median filter and mode filter), the proposed framework enables fast approximation of histogram filters via a number of box filtering or Gaussian filtering. In addition, high-quality piecewise-constant smoothing can be achieved via a number of bilateral filtering or guided filtering integrated in the proposed framework. Experiments on depth map denoising show the effectiveness of our framework.

Index Terms—Piecewise-constant smoothing, M-smoother, Edge-preserving filter, Bilateral filter, Guided filter

I. Introduction

Piecewise-constant smoothing serves as a fundamental tool in many image processing and low-level vision tasks. Originating from different background, a wide range of techniques are proposed to solve the problem, including anisotropic diffusion [23], bilateral filtering [26], robust estimation [5], etc. Their relations have been widely discussed in the literature [3], [28], [1], [9], [19] to benefit each other. For example, anisotropic diffusion is improved by exploiting new "edge-stopping" functions based on the well-studied influence functions from robust statistics [3]. We in this paper focus on exploring the reciprocity between robust estimation and weighted-average filters.

L. Bao is a Ph.D. student with the Department of Computer Science at the City University of Hong Kong, Hong Kong (e-mail: linchaobao@gmail.com).

Q. Yang is an Assistant Professor with the Department of Computer Science at the City University of Hong Kong, Hong Kong (e-mail: qiyang@cityu.edu.hk).

A binary executable demo with GUI that can reproduce most of the results in the manuscript can be accessed here: http://www.cs.cityu.edu.hk/~qiyang/publications/m-smoother/.

Specifically, We show that a robust estimator – the M-smoother [5] – can be reformulated as a series of weighted-average filtering followed by a winner-take-all operation. As a result, it can be much more efficiently approximated utilizing existing fast filtering algorithms. More importantly, existing weighted-average based edge-preserving filters, such as bilateral filter, guided filter [11], and cross-multi-point filter [16], can be largely "enhanced" to achieve high-quality piecewise-constant smoothing when integrated into the framework with well-studied robust loss functions.

The paper is organized as follows. In the next section we briefly review the preliminaries of weighted-average filters, local-histogram-based filters and M-smoother (note that local-histogram-based filters are closely related with M-smoother [19]). We present our framework in Sec. III. Sec. IV provides more discussion by exploiting the specific forms of the proposed framework. Finally, Sec. V concludes the paper.

II. Preliminaries and Notions

A. Weighted-average Filters

Throughout this paper, we use $\mathcal{F}(\cdot)$ to denote a filter operation performed on an image, that is, assuming I to be an input image and J the filtered image,

$$J = \mathcal{F}(I). \qquad (1)$$

A wide range of weighted-average filters can be expressed as, denoting $I_\mathbf{p}$ the color/intensity value at pixel $\mathbf{p}$ and $J_\mathbf{p}$ the filtered value at pixel $\mathbf{p}$,

$$J_\mathbf{p} = \sum_{\mathbf{q} \in \Omega_\mathbf{p}} w_{\mathbf{pq}}^{(T)} \cdot I_\mathbf{q}, \qquad (2)$$

where $\Omega_\mathbf{p}$ is the neighborhood of pixel $\mathbf{p}$ and the weighting function $w_{\mathbf{pq}}^{(T)}$ may or may not depend on a guidance image T. For example, when the

weighting function is a normalized Gaussian function depending on spatial distance between $\mathbf{p}$ and $\mathbf{q}$, the formulation is a Gaussian filter. When the color/intensity value of pixel $\mathbf{p}$ and $\mathbf{q}$ in guidance image T is taken into account, the formulation becomes the bilateral filter[1] [26]. See Table I for the weighted-average filters considered in this paper. Note that although the weighting function of the guided filter [11] seems a little complicated, it actually has a rather efficient algorithm (the complexity is about 6 times as much as that of a box filtering for single channel image).

TABLE I
WEIGHTED-AVERAGE FILTERS

Weighting function	Filter
$w_{\mathbf{pq}}^{(T)} = \frac{1}{W_{\mathbf{p}}}$	Box (BF)[†]
$w_{\mathbf{pq}}^{(T)} = \frac{1}{W_{\mathbf{p}}} G_\sigma(\|\mathbf{p} - \mathbf{q}\|)$	Gaussian (GF)[†]
$w_{\mathbf{pq}}^{(T)} = \frac{1}{W_{\mathbf{p}}} G_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|) G_{\sigma_r}(|T_{\mathbf{p}} - T_{\mathbf{q}}|)$	Bilateral (BLF)[†]
$w_{\mathbf{pq}}^{(T)} = \frac{1}{|\omega|^2} \sum_{k:(\mathbf{p},\mathbf{q}) \in \omega_k} (1 + \frac{(T_{\mathbf{p}} - \mu_k)(T_{\mathbf{q}} - \mu_k)}{\sigma_k^2 + \epsilon})$	Guided (GDF)[‡]

[†]: Notion $W_{\mathbf{p}}$ is normalization factor (summing all weights for pixel $\mathbf{p}$), $G(\cdot)$ is the Gaussian function.

[‡]: Notion ω_k denotes each window (with radius r and $|\omega|$ pixels) that covers both pixel $\mathbf{p}$ and $\mathbf{q}$, whose mean and variance are μ_k and σ_k, respectively. ϵ is a parameter.

The parameters for the four filters are as follows: box filter is controlled by the radius r of the box window; Gaussian filter is controlled by the parameter σ in Gaussian function; bilateral filter is controlled by parameter σ_s of spatial kernel and parameter σ_r of range kernel; guided filter is controlled by r and ϵ. The parameter "correspondence" between bilateral filter and guided filter is suggested in [11]: $\sigma_s \leftrightarrow r$ and $\sigma_r^2 \leftrightarrow \epsilon$. For a unified discussion, we further extend the "correspondence" to box filter and Gaussian filter: we also use σ_s to denote the parameter σ in Gaussian filter and control the parameter r in box filter[2]. In this way, we can use the two parameters σ_s and σ_r to discuss all the filters, following the convention of the bilateral filter [22] (σ_s is measured by pixel number and σ_r is a real number between 0 and 1 indicating a fraction of the whole intensity range $\mathcal{R}$).

[1]Bilateral filter is also referred to as σ-filter in [5] or nonlinear Gaussian filter in [28].

[2]To achieve the same amount of smoothing as Gaussian filter, the r in box filter is calculated by $r = \lfloor \sqrt{2}\sigma_s \rfloor$ empirically in our experiments [12] (it can be verified by experimental comparison using PSNR between box filtered image and Gaussian filtered image).

B. Local-histogram-based Filters

A family of local-histogram-based filters [27], [15] (e.g., median filter and mode filters) are to replace the color/intensity of each pixel with the color/intensity of neighboring majority pixels (e.g., using some certain robust statistics drawn out from local histogram). For example, median filtering is to replace each pixel value with the median of neighboring pixel values (if Gaussian weighted neighborhood is used, it is called *isotropic* median filtering [15]). The *closest-mode* filtering [15] (also referred to as *local mode* filtering in [27]) replaces each pixel with its closest mode, and the *dominant-mode* filtering [15] (similar to the *global mode* filtering in [27]) instead uses the mode having the largest population. Although such kind of filters can smooth out high contrast, fine-scale details, they often face a problem of serious deviation from the original edges (especially at corners), since local histogram completely ignores image geometric structures.

For convenience, in the rest of this paper, we use the term "local mode filter" and "global mode filter" in [27] to refer to filters whose local histograms are constructed within *hard* spatial windows, and use the term "closest-mode filter" and "dominant-mode filter" in [15] to refer to filters whose local histograms are constructed within Gaussian weighted *soft* spatial windows.

C. M-Smoother

From the statistical point of view, the simplest, non-robust estimation of the underlying image signal contaminated by zero-mean Gaussian noise is to estimate the intensity value of each pixel by minimizing the sum of squared residual errors (L_2 norm) within local window

$$J_{\mathbf{P}} = \underset{\theta}{\operatorname{argmin}} \sum_{\mathbf{q} \in \Omega_{\mathbf{P}}} (\theta - I_{\mathbf{q}})^2. \qquad (3)$$

Solving the optimization problem yields exactly the box filter. The problem of such least-squares estimation is that it is very sensitive to outliers and thus pixels at the two sides of an edge will affect each other. As a result, edges will get blurred.

In order to increase *robustness* and reject outliers, the quadratic function in the above formulation need to be replaced with more tolerant function that gives less penalty to outliers. For example, using

the absolute error function instead of the quadratic function, we obtain (L_1 norm)

$$J_{\mathbf{p}} = \underset{\theta}{\operatorname{argmin}} \sum_{\mathbf{q} \in \Omega_{\mathbf{p}}} |\theta - I_{\mathbf{q}}|, \qquad (4)$$

whose solution yields exactly the median filter [28].

More generally, the above formulation can be extended to the *M-smoother* [5][3] (originated from the *M-estimator* in robust statistics [13], [10])

$$J_{\mathbf{p}} = \underset{\theta}{\operatorname{argmin}} \sum_{\mathbf{q} \in \Omega_{\mathbf{p}}} \rho(\theta - I_{\mathbf{q}}), \qquad (5)$$

where $\rho(\cdot)$ is a *loss function* (also referred to as *ρ-function* or *error norm*). For robustness, the loss function should not grow too rapidly, hence lessening the influence of outliers. Its derivative function is a good tool for studying the influence of outliers, which is often referred to as *influence function* or *ψ-function* in robust statistics [13], [10]. In order to preserve sharp edges in images, a *redescending* influence function ($|\psi(x)| \to 0$ as $|x| \to \infty$) is often preferred [5], [3] (we hereafter refer to the loss function whose influence function is redescending as *redescending-influence loss function*). Table II shows several pairs of the loss function and the corresponding influence function. More robust loss functions can be found in [2]. Note the parameter σ in Table II is used to control the influence scale [3]. In this paper, we associate the scale parameter σ in loss function with the range parameter σ_r in weighted-average filters (let $\sigma = \sigma_r \cdot \mathcal{R}$), as basically both of them are to control the "robustness" [8].

Additionally, the M-smoother can be extended [5] to take into account spatial weights in local window. The formulation further becomes
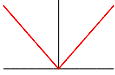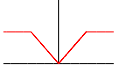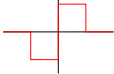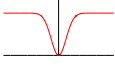
$$J_{\mathbf{p}} = \underset{\theta}{\operatorname{argmin}} \sum_{\mathbf{q} \in \Omega_{\mathbf{p}}} \rho(\theta - I_{\mathbf{q}}) \cdot G(\|\mathbf{p} - \mathbf{q}\|), \quad (6)$$

where $G(\cdot)$ is a Gaussian weighting function.

The strategy of employing a robust loss function to reject outliers is analogous to the range weighting function in bilateral filter [28], [8]. Durand and Dorsey [8] noticed this and propose to improve bilateral filter by replacing the Gaussian weighting function with a superior function – *Tukey's biweight* – whose influence function is more conform to

---

[3]The M-smoother in [5] means to find *local* minima that is closest to the original input pixel value when minimizing the objective function. We in this paper do not mean to find local minima, but rather to find the *global* minima of the objective function.

TABLE II
LOSS FUNCTION AND INFLUENCE FUNCTION

Loss function†	Influence function†
L_1 norm: $\rho(x) = |x|$	
Truncated L_1 norm: $\rho(x,\sigma) = \begin{cases} |x|, & \text{if } |x| \leq \sigma \\ \sigma, & \text{otherwise} \end{cases}$	
Negative Gauss: $\rho(x,\sigma) = 1 - e^{-\frac{x^2}{(0.64\sigma)^2}}$	
Tukey's biweight: $\rho(x,\sigma) = \begin{cases} \frac{x^2}{\sigma^2} - \frac{x^4}{\sigma^4} + \frac{x^6}{3\sigma^6}, & \text{if } |x| \leq \sigma \\ \frac{1}{3}, & \text{otherwise} \end{cases}$	
Geman-Reynolds: $\rho(x,\sigma) = \frac{-\sigma}{\sigma+|x|}$	

†: Influence function is the derivative of loss function. Note that the last four loss functions are *redescending-influence loss functions* (RILF).

the redescending principle. Note that although the correspondence between M-smoother and bilateral filter can be established in this way, their output are usually much different from each other, since the bilateral filter can only be viewed as one step towards finding *local* minima of the objective function of M-smoother using iterative methods [28], [19]. Simply increasing iterations of bilateral filtering still cannot correctly lead to the output of M-smoother, as the image is transformed after each step [28].

Actually, the global minima of M-smoother with a negative Gaussian loss function is similar to that of a global mode filter [27] or dominant-mode filter, and the local minima of M-smoother corresponds to the local mode filter or closest-mode filter [15]. Please refer to [19] for a detailed discussion.

III. OUR FILTERING FRAMEWORK

In this section, we generalize the M-smoother and propose a numerical scheme to solve it via a series of weighted-average filtering. The numerical scheme is indeed our proposed filtering framework, the specific forms of which will be exploited in next section.

A. Generalized M-Smoother

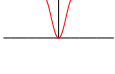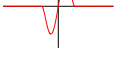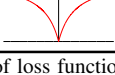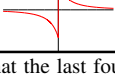We extend the weighting function of M-smoother in Eq. (6) into a more general form:

$$J_{\mathbf{p}} = \operatorname*{argmin}_{\theta} \sum_{\mathbf{q} \in \Omega_{\mathbf{p}}} \rho(\theta - I_{\mathbf{q}}) \cdot w_{\mathbf{pq}}^{(T)}, \qquad (7)$$

where the $w_{\mathbf{pq}}^{(T)}$ is the weight of pixel $\mathbf{q}$ contributing to $\mathbf{p}$ and T can be either a guidance image (see Table I) or the input image I itself. Note that our formulation actually combines the *implicit* piecewise-constant model (from robust estimator) and the *explicit* weighting scheme (from weighted-average filters). If the weighting function is the edge-preserving weighting from bilateral filter or guided filter, our formulation can achieve better edge preservation in piecewise-constant smoothing than traditional M-smoother due to the explicit weighting scheme (see Sec. IV-B for details).

B. Solve via a Filtering Framework

Let Θ denote the set of all possible output pixel values of the smoother, for a given $\theta \in \Theta$, we define $D(\theta)$ as a *cost image* where each pixel $\mathbf{p}$ is computed from input image I as follows

$$[D(\theta)]_{\mathbf{p}} = \rho(\theta - I_{\mathbf{p}}). \qquad (8)$$

Then we can reformulate Eq. (7) into

$$J_{\mathbf{p}} = \operatorname*{argmin}_{\theta \in \Theta} [\mathcal{F}(D(\theta))]_{\mathbf{p}}, \qquad (9)$$

where $\mathcal{F}(\cdot)$ is a weighted-average filter (see Eq. (2)). Assuming set Θ is finite, the formulation essentially means the filtering on all possible cost images $\{D(\theta) \mid \theta \in \Theta\}$ is first performed and then an argmin operation at each pixel $\mathbf{p}$ is individually applied according to the filtered results at $\mathbf{p}$. The process is known as *cost-volume filtering framework* [24] in the context of discrete labeling problem like stereo matching.

The key insight of the above reformulation is that the filtering on cost image allows us to apply fast filtering algorithms for efficient computation of the generalized M-smoother. With the constant-time complexity (per input pixel) filtering algorithms[4] [6], [7], [29], [4], [11], we are able to compute the weighted averaging in Eq. (7) in constant time

---

[4]In the literature, "constant time (per pixel)" is also referred to as "linear time (in pixel number)". We follow the way of "constant time" in [4].

Algorithm 1 Approximate Algorithm

Input: image I, number of samples n, filter $\mathcal{F}$.
Output: smoothed image J.
————————**Algorithm Start**————————
calculate evenly distributed samples $\hat{\Theta}$
for each sample θ in $\hat{\Theta}$ **do**
    (1) compute cost image $D(\theta) = \rho(\theta - I)$;
    (2) filtering the cost image to get $\mathcal{F}(D(\theta))$;
end for
for each pixel $\mathbf{p}$ **do**
    (1) compute $\hat{J}_{\mathbf{p}}$ using Eq. (10);
    (2) compute output $J_{\mathbf{p}}$ using Eq. (11);
end for
————————-**Algorithm End**————————-

---

regardless of the size of the neighborhood Ω_p at each pixel p. Note the argmin step is performed at each pixel individually, thus its computation can be ignored comparing to the filtering step. Let $|\Theta|$ denote the size of set Θ, the computation of Eq. (9) is mainly the $|\Theta|$ filtering on cost images.

We in this paper mainly target on 8-bit image, which is the most commonly used format in practice (for 24-bit or 32-bit color image, we separately process each of the RGB channels). Thus set Θ contains 256 integers in $[0, 255]$, *i.e.*, $|\Theta| = 256$. We will show how to further reduce the number of filtering in next section.

C. Approximate Algorithm

An effective strategy to reduce the number of filtering required in the framework is through uniformly sampling the set Θ. The idea is that, filtering is only performed for samples rather than all possible values in Θ, and the output for each pixel is approximated using the samples according to the filtered results (see Algorithm 1). Specifically, let the sampling set be $\hat{\Theta}$, which has n samples evenly distributed in Θ, we first find out the best θ among these samples at each pixel $\mathbf{p}$

$$\hat{J}_{\mathbf{p}} = \operatorname*{argmin}_{\theta \in \hat{\Theta}} [\mathcal{F}(D(\theta))]_{\mathbf{p}}. \qquad (10)$$

Since the following operation is performed at each pixel $\mathbf{p}$ individually, for simplicity, we use θ_0 to denote $\hat{J}_{\mathbf{p}}$ and $f(\theta)$ to denote the filtered pixel value $[\mathcal{F}(D(\theta))]_{\mathbf{p}}$ for a given θ. For pixel $\mathbf{p}$, suppose θ_+ and θ_- are the two closest samples near θ_0 in $\hat{\Theta}$, then the output value of pixel $\mathbf{p}$ can be approximated by
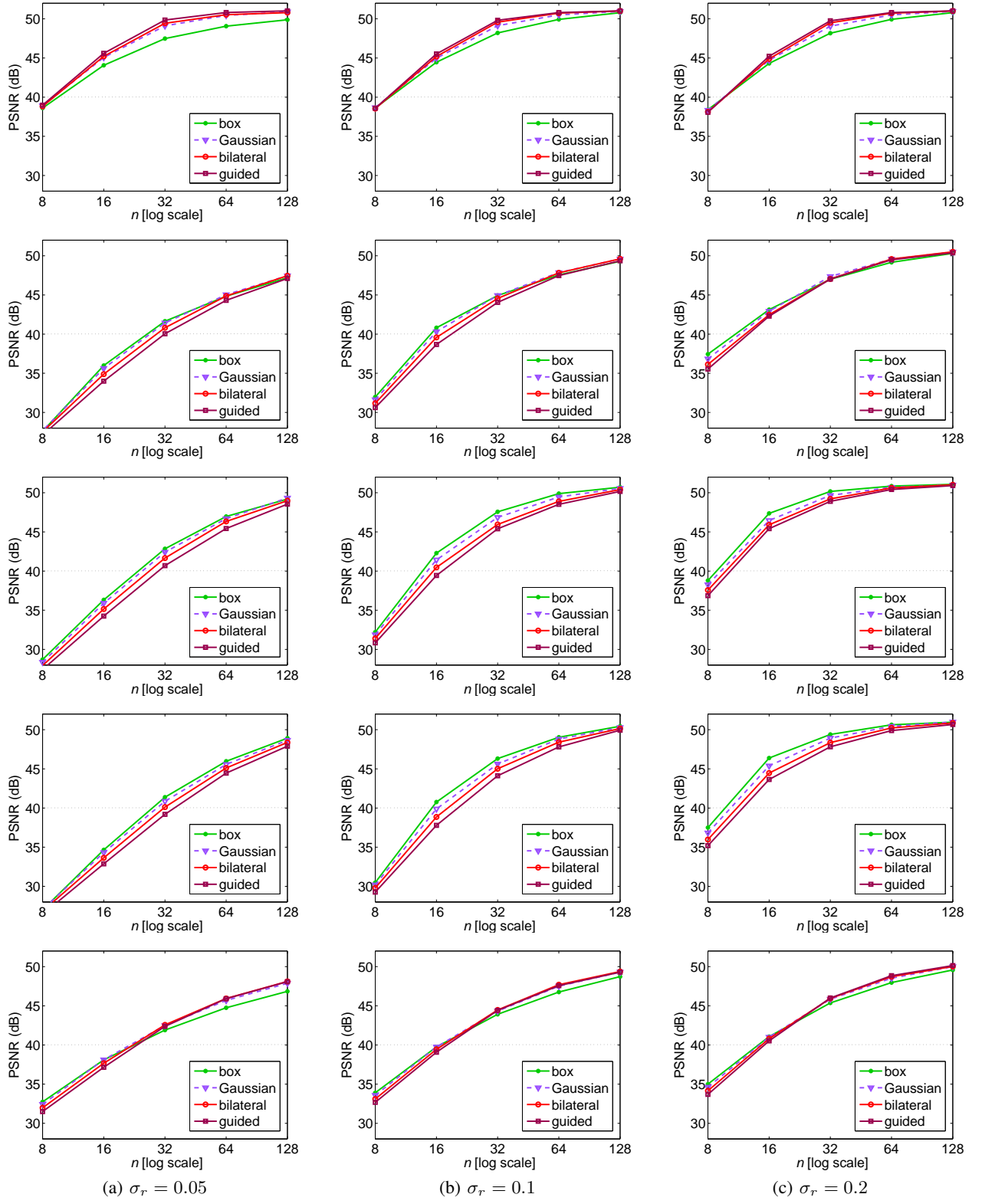
Fig. 1. PSNR accuracy of the approximate algorithm. The loss functions are (from top to bottom): L_1 norm, truncated L_1 norm, negative Gauss, Tukey's biweight, and Geman-Reynolds. It is suggested that PSNR value above 40dB often corresponds to almost invisible differences between two images [21].

fitting a parabolic curve [30] using the three samples and their corresponding filtered values,

$$J_{\mathbf{p}} = \theta_0 - \frac{(\theta_+ - \theta_-)(f(\theta_+) - f(\theta_-))}{4(f(\theta_+) + f(\theta_-) - 2f(\theta_0))}. \quad (11)$$

Note that the parabolic fitting is a closed-form approximation by assuming the cost function follows a parabolic curve near the bottom (near $\hat{J}_{\mathbf{p}}$). The actual cost function cannot be analytically solved without considering the pixel distribution of each pixel's neighborhood due to the data-dependent average (see Appendix for the derivation). Although a more precise way to solve it is to perform further sampling and filtering near $\hat{J}_{\mathbf{p}}$, we find that the closed-form approximation is much faster and usually accurate enough in practice. We will show this in next section.

D. Experimental Validation

As commonly adopted in developing approximate algorithms for bilateral filter [21], we also use the peak signal-to-noise ratio (**PSNR**) metric to measure the approximate accuracy. Higher PSNR value between the approximate and exact results means more accurate approximation (it is suggested that PSNR value above 40dB often corresponds to almost invisible differences between two images [21]). We test the approximate algorithm for four filters listed in Table I and five loss functions listed in Table II (in total 20 pairs of combination) in the following sampled parameter settings, respectively: $\sigma_s \in \{2, 4, 8, 16\}$, $\sigma_r \in \{0.05, 0.1, 0.2, 0.4\}$, and $n \in \{8, 16, 32, 64, 128\}$. The 8 test images are from Paris's bilateral filtering dataset [20] (color images are converted to grayscale): *dome*, *dragon*, *greekdome*, *housecorner*, *polin*, *swamp*, *tulip*, and *turtle*. The reference image for calculating PSNR is obtained from Eq. (9) by enumerating all possible output values within Θ (*i.e.*, integer values in $[0, 255]$ for 8-bit image).

According to our observation, we find that the PSNR value is not sensitive to the spatial parameter σ_s. Thus we only plot the PSNR results according to different σ_r for each loss function and filter in Fig. 1. That is, for a specific pair of loss function and filter, PSNR from different images with different σ_s but a same σ_r are averaged together for the plot (results for $\sigma_r = 0.4$ are not shown since the PSNR are commonly high). The results show that

TABLE III
APPROXIMATING HISTOGRAM FILTERS

	$\mathcal{F}$ is box filter	$\mathcal{F}$ is Gaussian filter
L_1 norm	median filter	isotropic median filter [15]
RILF[†]	global mode filter[‡] [27]	dominant-mode filter[‡] [15]

†: RILF stands for redescending-influence loss functions.

‡: The global mode filter uses hard spatial window to construct local histogram, while the dominant-mode filter uses Gaussian weighted window to construct local histogram.

Timing (per mega-pixel) with $n = 16$ (RILF is truncated L_1 norm):

filter type	CPU[†]		GPU[†]	
	ours	[15]	ours	[15]
median	150 ms	–	5 ms	–
isotropic median	230 ms	833 ms	6 ms	166 ms
global mode	150 ms	–	5 ms	–
dominant-mode	230 ms	2777 ms	6 ms	332 ms

†: The timing of [15] is reproduced from the paper. It is reported on Intel 2.83 GHz Xeon E5440 CPU and NVIDIA Quadro FX 770M GPU. Note that our algorithm for dominant-mode filter only needs half as many Gaussian filtering as [15] and is much simpler to implement (the algorithm in [15] requires $2n$ Gaussian filtering for computing integrals and derivatives of histogram).

(a) Input   (b) Ours ($\mathcal{F}$ is BF)   (c) Ours ($\mathcal{F}$ is GF)

Fig. 2. Example result of our approximate global mode filter ($\mathcal{F}$ is box filter) and dominant-mode filter ($\mathcal{F}$ is Gaussian filter).

the approximate accuracy for each loss function is sensitive to σ_r (except the L_1 norm which does not have a scale parameter). Commonly speaking, the larger value of σ_r, the smaller n required for high accuracy (PSNR above 40dB). For example, for $\sigma_r = 0.05$ (first row), $n = 32$ can commonly make the PSNR above 40dB, while for $\sigma_r = 0.1$ (second row), $n = 16$ is often enough.

The above experimental results imply that in practice we can safely use a n much smaller than 256 (for 8-bit image). Generally, for situations where parameter σ_r is not too small (e.g., ≥ 0.1), we can use $n = 16$ to get accurately approximated results (n can be further reduced for larger σ_r).

IV. EXPLOITING SPECIFIC FORMS

We in this section demonstrate our contribution by exploiting the specific forms of the generalized M-smoother and our filtering framework. Since during our previous experiments, we find that the smoothing effects of the four redescending-influence loss functions are actually visually similar
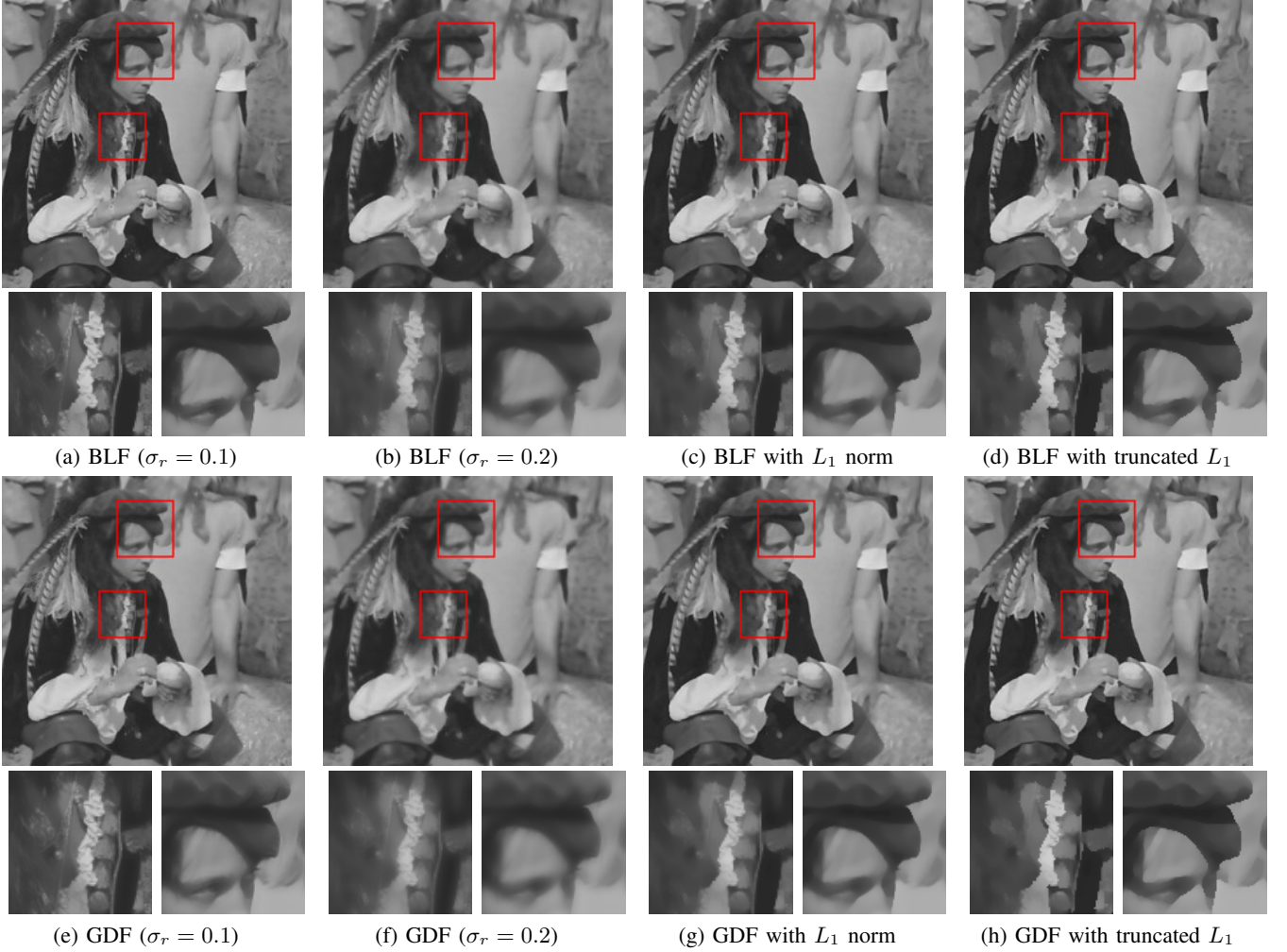
(a) BLF ($\sigma_r = 0.1$)   (b) BLF ($\sigma_r = 0.2$)   (c) BLF with L_1 norm   (d) BLF with truncated L_1

(e) GDF ($\sigma_r = 0.1$)   (f) GDF ($\sigma_r = 0.2$)   (g) GDF with L_1 norm   (h) GDF with truncated L_1

Fig. 3. Piecewise-constant smoothing with proposed framework. Spatial parameter is $\sigma_s = 3$ for all results. The σ_r in our framework is 0.2. The bilateral filter and guided filter with $\sigma_r = 0.1$ cannot smooth out fine details (see left close-up window), while with larger parameter $\sigma_r = 0.2$ they may blur major edges (see right close-up window).
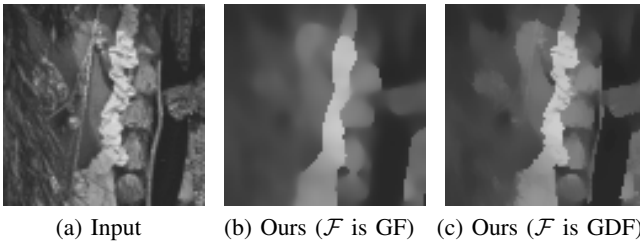

(a) Input   (b) Ours ($\mathcal{F}$ is GF)   (c) Ours ($\mathcal{F}$ is GDF)

Fig. 4. Comparison between Gaussian filter and guided filter as $\mathcal{F}$ (with truncated L_1 norm, $\sigma_s = 3$, $\sigma_r = 0.2$). The comparison shows that, when $\mathcal{F}$ is an edge-preserving filter (GDF), the framework can better preserve edges, while on the other hand, when $\mathcal{F}$ is a linear filter (GF), it can achieve stronger smoothing but may cause deviations from input edges.

(a) Input   (b) BLF   (c) GDF

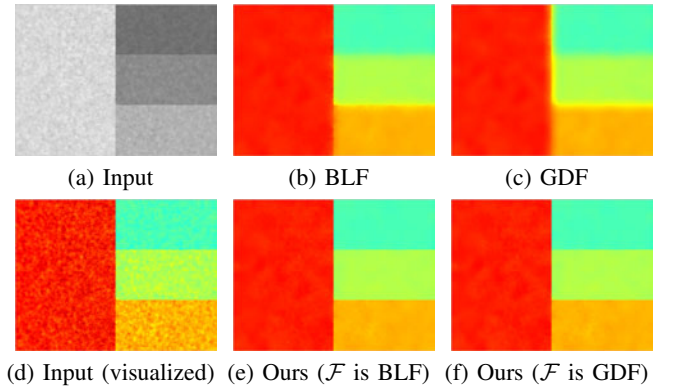(d) Input (visualized)   (e) Ours ($\mathcal{F}$ is BLF)   (f) Ours ($\mathcal{F}$ is GDF)

Fig. 5. Smoothing of a synthetic grayscale noisy image. The colored visualized display is shown for clarity. The parameters used in BLF and GDF are $\sigma_s = 10$, $\sigma_r = 0.15$. The loss function in our framework is truncated L_1 norm.

to each other (but different from the L_1 norm). We will use one of the last four loss functions as a representative when demonstrating the filtering effects in this section.
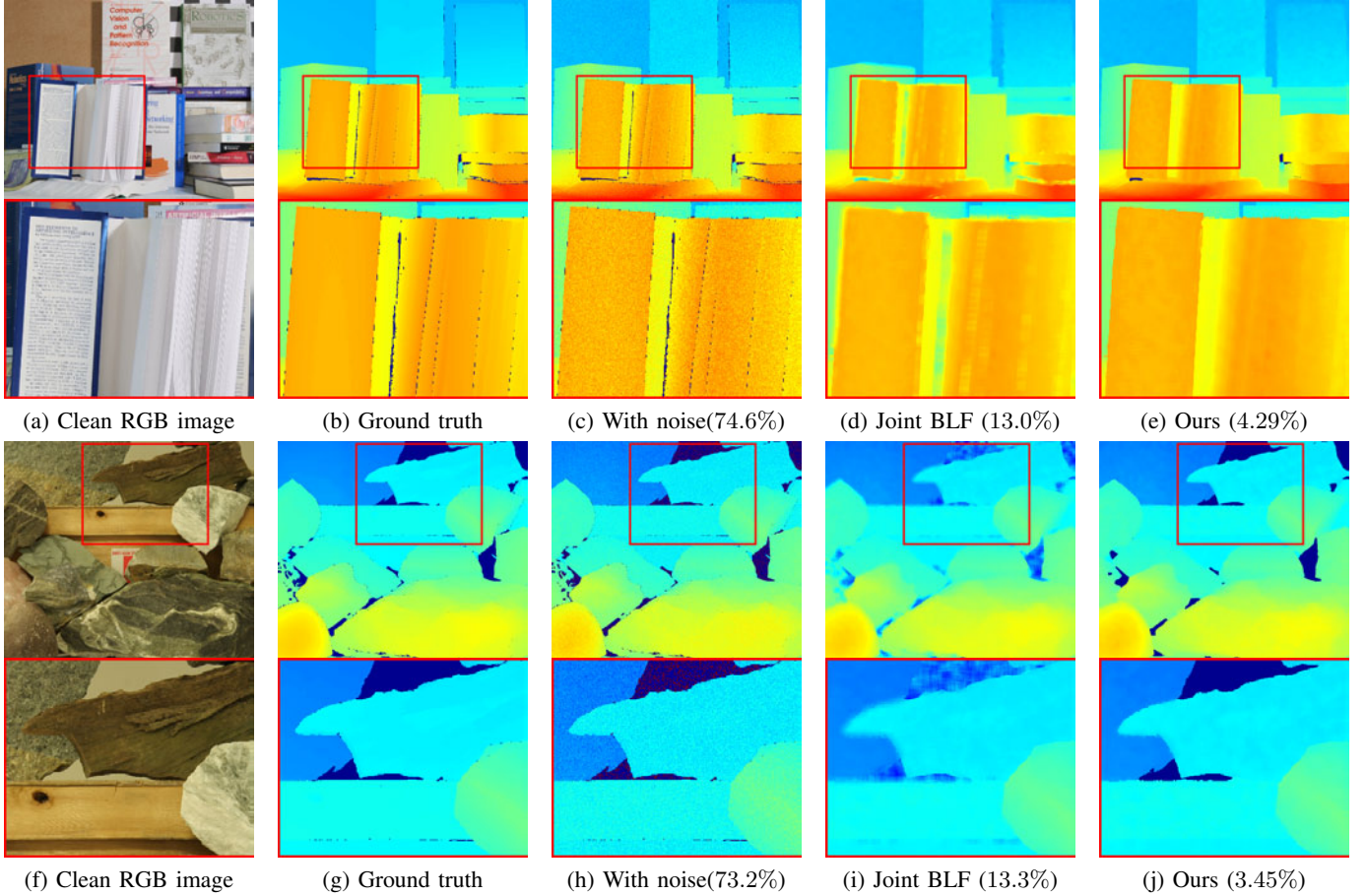
Fig. 6.   Joint filtering for disparity map denoising. From left to right: clean RGB images, ground-truth disparity maps, disparity maps deteriorated with Gaussian noise, denoised disparity maps using joint bilateral filter, denoised disparity maps using our enhanced joint bilateral filter with truncated L_1 norm as loss function. The parameters are $\sigma_s = 5$, $\sigma_r = 0.1$. The percentage shown under subfigures is the the percentage of bad estimated pixels: we adopted the methodology used in [25]: if the disparity error of a pixel is larger than 1, it is treated as a bad pixel. Notice that denoising on natural/textured images [31] or based on complicated noise model [14] is out of the scope of this paper.

A. Fast Algorithms for Histogram Filters

As discussed in Sec. II-C, traditional M-smoother is closely related to histogram filters. Thus the proposed framework with $\mathcal{F}$ being box filter or Gaussian filter can serve as fast approximation for histogram filters. Table III shows the correspondence between proposed framework and histogram filters.

Note that both box filter and Gaussian filter can be implemented in constant time complexity (per input pixel) [6], [7]. In our implementation, box filter takes 5 milliseconds per mega-pixel (**ms/Mp**) on CPU and 0.25 ms/Mp on GPU, while the Gaussian filter takes 12 ms/Mp on CPU and 0.3 ms/Mp on

GPU[5]. The running time of our approximated local-histogram-based filters is shown in Table III. Fig. 2 shows an example result of our approximate global mode filter and dominant-mode filter.

B. Piecewise-constant Smoothing

When the operator $\mathcal{F}$ in Eq. (9) is bilateral filter or guided filter, the proposed framework is actually a *weighted median filter* [17] (with L_1 norm) or a *weighted mode filter* [18] (with a redescending-influence loss function). The framework plays a role for "enhancing" the edge-preserving ability of $\mathcal{F}$ to achieve piecewise-constant smoothing, due to

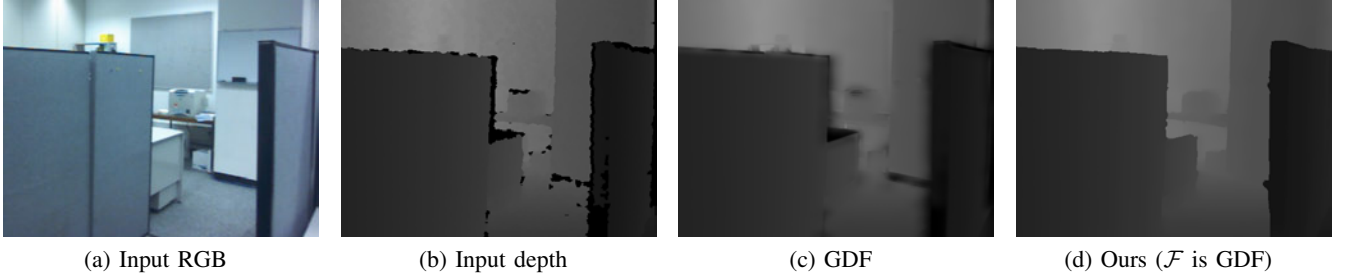| (a) Input RGB | (b) Input depth | (c) GDF | (d) Ours ($\mathcal{F}$ is GDF) |

Fig. 7. Joint filtering on depth map obtained by Microsoft Kinect camera. The parameters are $\sigma_s = 10$, $\sigma_r = 0.1$. The loss function in our framework is truncated L_1 norm. The invalid regions in the depth map (black holes) can be neatly fixed by our enhanced joint filtering (the value of invalid pixels is treated as zero during the filtering).

the derivation of the M-smoother from piecewise-constant model [5]. Fig. 5 shows such an example on a synthetic image. Fig. 3 shows an example on a natural image. Note that although the loss function in our framework works like the range weighting function in bilateral filter, yet adding another range weighting kernel into bilateral filter (i.e., changing σ_r in Gaussian range kernel) cannot yield our smoother. Also, performing bilateral filtering in an iterative manner cannot achieve the same results as our smoother (as discussed in Sec. II-C). Compared with the histogram filters (e.g., when $\mathcal{F}$ is box filter or Gaussian filter in our framework), the new smoother can better preserve edges (recall that local histogram completely ignores the color/intensity value of center pixel). Fig. 4 gives an illustration.

Additionally, both the (joint) bilateral filtering and guided filtering can use another image [18], rather than the input image itself, as guidance image. This gives the filtering more flexibility (hereafter referred to as *joint filtering*) and makes it especially suitable for depth image restoration. For example, the enhanced joint filtering can be employed to reduce the noise of depth image, which can be acquired by commercial cameras but is commonly noisy, using clean RGB image that can be simultaneously acquired with depth image as guidance image. Fig. 6 shows two experimental results of the joint bilateral filtering on disparity maps (disparity is inversely proportional to depth). The numerical comparison shows the effectiveness of our approach. Fig. 7 shows a real world example of the denoising on depth image obtained by Microsoft Kinect camera. Note that the invalid regions in the original depth map (black holes) can be neatly fixed by our approach. Similarly, the enhanced joint filtering can also serve as a tool for refining disparity

maps produced by existing stereo matching algorithms [25]. Fig. 8 shows a quantitative comparison of the disparity map refinement on the Middlebury benchmark [25] (we use three basic and efficient stereo matching algorithms to produce low-quality disparity maps), using original joint filtering and our enhanced joint filtering, respectively. Compared to the original joint filtering, our approach can produce cleaner and sharper disparity maps. With a CUDA implementation, our enhanced joint filtering can achieve real-time performance on GPU (e.g., the enhanced guided filtering with $n = 16$ takes about 30 ms/Mp on our GPU).

V. CONCLUDING REMARKS

We have presented a filtering framework for achieve piecewise-constant smoothing. The proposed framework is derived by solving a generalized M-smoother and can be implemented very efficiently on modern many-core processors utilizing parallelism. We demonstrate the effectiveness of the proposed framework for fast approximation of local-histogram-based filters and enhancing existing edge-preserving filters. An unsolved problem is how to further reduce the number of filtering required in the framework, for example, by automatically selecting the sampling intensity levels to minimize the quantization error. We intend to investigate this problem in the future.

APPENDIX
DERIVATION OF PARABOLIC FITTING APPROXIMATION

Let us first consider the simplest case Eq. (4) (box filter with L_1 norm loss function). Since neighboring pixels tend to be similar to each other, we assume the pixel values near $\mathbf{p}$ follow a uniform distribution

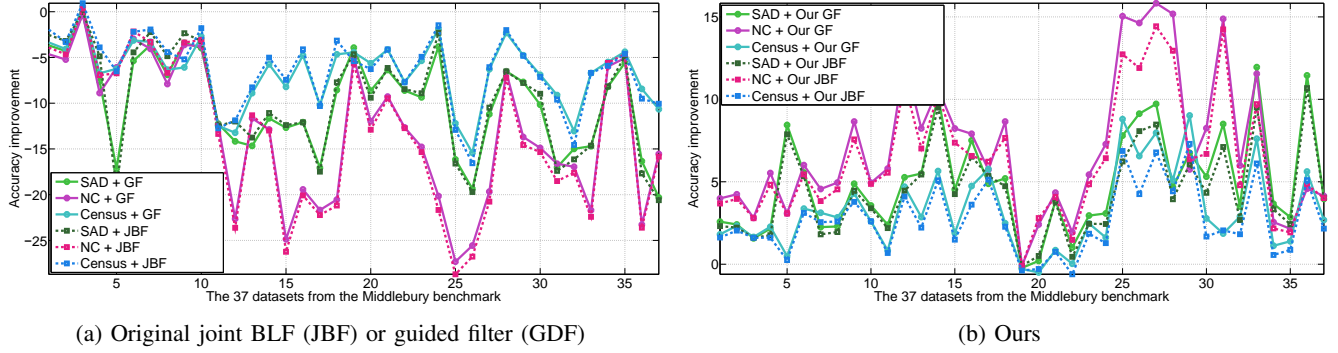(a) Original joint BLF (JBF) or guided filter (GDF)

(b) Ours

Fig. 8. Quantitative comparison of disparity map refinement by original joint filtering and our enhanced joint filtering (accuracy is measured by percentage of bad pixels [25]). The three stereo matching algorithms, i.e., Sum of Absolute Differences (SAD), Normalized Correlation (NC), Census Transform, can be computed very efficiently but the quality of the produced disparity map is low. Filtering the disparity map using joint bilateral filter or guided filter with input RGB image as guidance image, the quality of disparity map can not be improved. By contrast, our enhanced joint bilateral filtering or guided filtering (with truncated L_1 norm loss function) can commonly make the quality of the disparity map improved. An example of the visual comparison is provided in Fig. 9.

(a) Original disparity maps (b) JBF (σ_s=10,σ_r=0.1) (c) Ours ($\mathcal{F}$ is JBF) (d) GDF (σ_s=10,σ_r=0.1) (e) Ours ($\mathcal{F}$ is GDF)
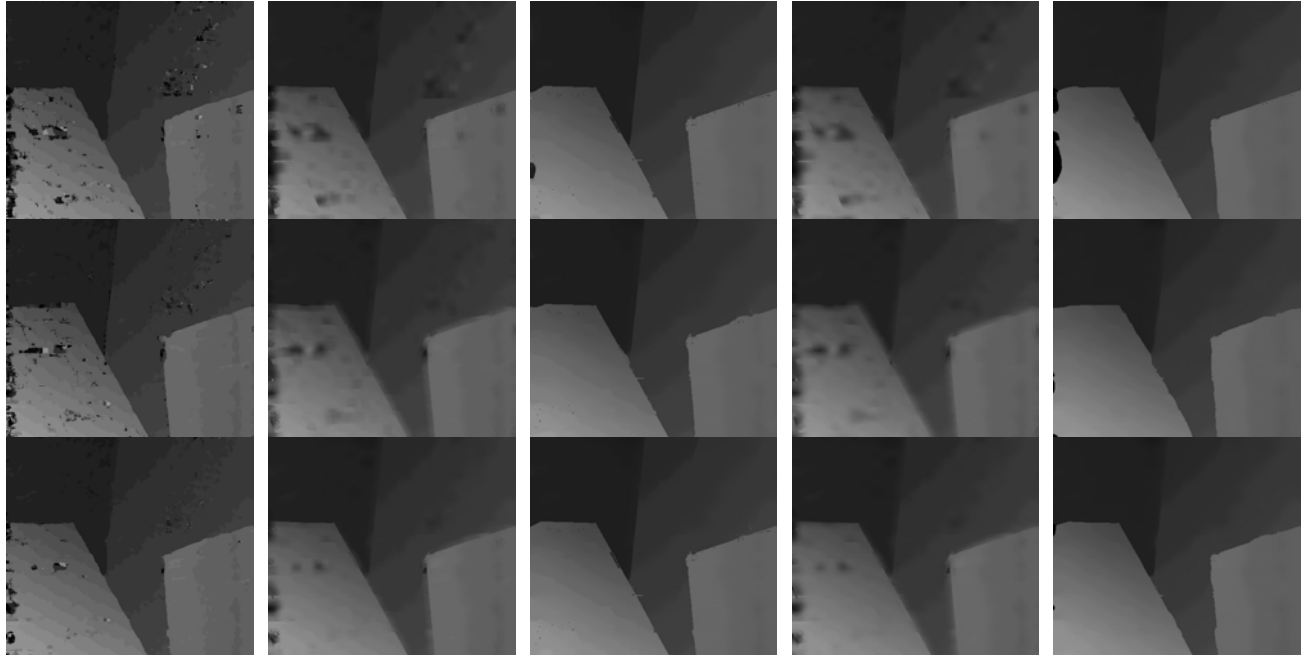
Fig. 9. Visual comparison of disparity map refinement for *Venus* dataset from Middlebury benchmark [25]. The three rows correspond to the three algorithms in Fig. 8. Loss function in our framework is truncated L_1 norm.

between a and b ($0 < a < b < 255$). Then the cost function of Eq. (4) can be rewritten as an integral

$$E(\theta) = \sum_{\mathbf{q} \in \Omega_{\mathbf{P}}} |\theta - I_{\mathbf{q}}| \approx \int_a^b |\theta - x| \mathrm{d}x$$

$$= \begin{cases} \frac{1}{2}[(\theta - b)^2 - (\theta - a)^2)], & \text{if } \theta < a \\ \frac{1}{2}[(\theta - a)^2 + (\theta - b)^2)], & \text{if } a \leqslant \theta \leqslant b \\ \frac{1}{2}[(\theta - a)^2 - (\theta - b)^2)], & \text{if } \theta > b \end{cases}$$

The function is a "cup-shaped" continuous function, with two linear segments in the ends and a quadratic

segment in the middle. Thus in our approximate algorithm, we can fit a parabola near the bottom of the "cup-shaped" function to find the minima. The derivation can be generalized to other loss functions in Table II (although their resulting cost function will not be parabolic curve, in our algorithm we use parabolic fitting to approximate all the cases for simplicity).

Notice that the above derivation is based on a much simplified problem (uniform distributed pixel

values near **p**). In fact, because the cost function is data-dependent (depending on the neighboring pixels near **p**), the actual cost function cannot be analytically solved. Besides, the complicated weighting schemes other than box filter (see Table I) will add more complexity to the problem. Thus we conducted a thorough experimental validation in Section III-D. The experiments show that the approximate algorithm works well in practice.

REFERENCES

[1] D. Barash. Fundamental relationship between bilateral filtering, adaptive smoothing, and the nonlinear diffusion equation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(6):844–847, 2002.

[2] M. Black and A. Rangarajan. On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *Int. J. Comput. Vis.*, 19(1):57–91, 1996.

[3] M. Black, G. Sapiro, D. Marimont, and D. Heeger. Robust anisotropic diffusion. *IEEE Trans. Image Process.*, 7(3):421–432, 1998.

[4] K. Chaudhury, D. Sage, and M. Unser. Fast o(1) bilateral filtering using trigonometric range kernels. *IEEE Trans. Image Process.*, 20(12):3376–3382, Dec 2011.

[5] C. Chu, I. Glad, F. Godtliebsen, and J. Marron. Edge-preserving smoothers for image processing. *Journal of the American Statistical Association*, 93(442):526–541, 1998.

[6] F. Crow. Summed-area tables for texture mapping. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 18(3):207–212, 1984.

[7] R. Deriche. Recursively implementing the gaussian and its derivatives. In *ICIP 1992*, pages 263–267, 1992.

[8] F. Durand and J. Dorsey. Fast bilateral filtering for the display of high-dynamic-range images. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 21(3):257–266, 2002.

[9] M. Elad. On the origin of the bilateral filter and ways to improve it. *IEEE Trans. Image Process.*, 11(10):1141–1151, 2002.

[10] F. Hampel, E. Ronchetti, P. Rousseeuw, and W. Stahel. *Robust statistics: the approach based on influence functions*. Wiley, 1986.

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

[12] K. He, J. Sun, and X. Tang. Guided image filtering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(6):1397–1409, 2013.

[13] P. Huber. *Robust statistics*. Wiley, 1981.

[14] J. Jiang, L. Zhang, and J. Yang. Mixed noise removal by weighted encoding with sparse nonlocal regularization. *IEEE Trans. Image Process.*, 23(6):2651–2662, June 2014.

[15] M. Kass and J. Solomon. Smoothed local histogram filters. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 29(4):100:1–100:10, 2010.

[16] J. Lu, K. Shi, D. Min, L. Lin, and M. N. Do. Cross-based local multipoint filtering. In *CVPR*, pages 430–437. IEEE, 2012.

[17] Z. Ma, K. He, Y. Wei, J. Sun, and E. Wu. Constant time weighted median filtering for stereo matching and beyond. In *ICCV*, pages 49–56. IEEE, 2013.

[18] D. Min, J. Lu, and M. N. Do. Depth video enhancement based on weighted mode filtering. *IEEE Trans. Image Process.*, 21(3):1176–1190, 2012.

[19] P. Mrázek, J. Weickert, and A. Bruhn. On robust estimation and smoothing with spatial and tonal kernels. *Geometric Properties for Incomplete Data*, pages 335–352, 2006.

[20] S. Paris. http://people.csail.mit.edu/sparis/bf/#code, 2013. [Online; accessed 2-Feb-2013].

[21] S. Paris and F. Durand. A fast approximation of the bilateral filter using a signal processing approach. *Int. J. Comput. Vis.*, 81(1):24–52, 2009.

[22] S. Paris, P. Kornprobst, and J. Tumblin. *Bilateral filtering: Theory and applications*, volume 1. Now Publishers Inc, 2009.

[23] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(7):629–639, 1990.

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

[25] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vis.*, 47(1):7–42, 2002.

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

[27] J. Van de Weijer and R. Van den Boomgaard. Local mode filtering. In *CVPR 2001*, volume 2, pages II–428. IEEE, 2001.

[28] G. Winkler, V. Aurich, K. Hahn, A. Martin, and K. Rodenacker. Noise reduction in images: Some recent edge-preserving methods. *Pattern Recognition and Image Analysis*, 9(4):749–766, 1999.

[29] Q. Yang, K. Tan, and N. Ahuja. Real-time o (1) bilateral filtering. In *CVPR 2009*, pages 557–564. IEEE, 2009.

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

[31] W. Zuo, L. Zhang, C. Song, D. Zhang, and H. Gao. Gradient histogram estimation and preservation for texture enhanced image denoising. *IEEE Trans. Image Process.*, 23(6):2459–2472, June 2014.