Edge-aware Filtering with Local Polynomial Approximation and Rectangle based Weighting

Xiao Tan, Changming Sun, and Tuan D. Pham

***Abstract*—This paper presents a novel method for performing guided image filtering using local polynomial approximation (LPA) with range guidance. In our method, the LPA is introduced into a multipoint framework for reliable model regression and better preservation on image spatial variation which usually contains the essential information in the input image. In addition, we develop a weighting scheme which has the spatial flexibility during the filtering process. All components in our method are efficiently implemented and a constant computation complexity is achieved. Compared with conventional filtering methods, our method provides clearer boundaries and performs especially better in recovering spatial variation from noisy images. We conduct a number of experiments for different applications: depth image upsampling, joint image denoising, details enhancement, and image abstraction. Both quantitative and qualitative comparisons demonstrate that our method outperforms state-of-the-art methods.**

***Index Terms*—Edge-aware filtering, Local polynomial regression, Rectangular weights, Computer vision, Depth enhancement**

I. Introduction

Image filtering is a technique to remove noises or defects in an image so as to improve image quality. Over several decades, a number of methods have been proposed in the field, such as local averaging based methods [1], [2], sparse and low rank representation based methods [3], learning based approaches [4], [5], and those based on kernel methods [6], [7]. Readers are referred to a more comprehensive study on the taxonomy of image denoising techniques and an extensive comparison among these techniques [8]. As to image filtering methods, it is important that the method does not damage any vital information such as the size or the shape of objects in the image. To this end, a filtering method which preserves edges/object boundaries in the image is often welcomed, which is also known as the edge-aware filtering method.

In addition to the single image filtering techniques, joint image filtering [9] or guided image filtering [2] has also been introduced for image smoothing, structure transformation, and other purposes in recent studies. Comparing with the single image filtering methods, this filtering operation can not only keep the desired property of edge-awareness but can also incorporate additional information from a given guidance image. It therefore becomes a powerful method in many computer vision and graphics applications including stereo matching [10], [11], [12], [13], optical flow estimation [11], colorization [14], and saliency detection [15]. By taking the original image as the guidance image, the guided image filtering can also be used as single image smoothing which is widely used in image denoising [16] and multiscale detail enhancement [17]. Recently, depth senors drew public attention among the cybernetics community. Many works have been carried out in the community such as human activity detection [18] and 3D modeling [19]. A comprehensive review of depth sensor based computer vision applications can be found in [20]. Unfortunately, the raw depth map obtained from the sensor is usually not satisfactory. As a result, depth map enhancement is required after obtaining the raw depth map. The joint image filtering method is an effective and efficient one for achieving depth enhancement.

The filtering methods can be categorized into averaging methods and non-averaging methods. An averaging filtering method is characterized by outputting the weighted average value of a specified region from the filtering image. Non-averaging filtering methods may have various formulations. A well-known non-averaging method is the weighted median filtering [21]. In [22], a constant weighted median filtering method is proposed by using histogram and an integral image scheme [23], [24]. Some other methods are proposed by minimizing a global cost function such as filtering by L_0 minimization [25] and total-variation filtering [26]. A collaborative filtering approach is proposed by using sequential grouping and 3D transformation [27]. Compared with these non-averaging methods, the averaging filtering methods are much more efficient and also achieve good results in practice. In this paper, we focus on the averaging based filtering methods.

A good implementation of using averaging based filtering methods for addressing the joint filtering problem is achieved in [9] where a bilateral filter (BF) [1] is adopted. This method will output a weighted average value where the weights are computed by Gaussian kernels of both spatial and color or intensity values over the support region around the pixel of interest. However, this BF based joint image filtering is computationally expensive and therefore many methods are proposed to accelerate it for fast implementation such as those in [28], [29], [30], [31], [32]. All these methods rely on quantization techniques. However, as discussed in [2], such quantization based methods do not perform well. Another problem of the joint BF is that it may introduce gradient

Xiao Tan is with the Department of Computer Science, the University of Hong Kong, Pokfulam Rd, Hong Kong.
E-mail: tanxchong@gmail.com

Changming Sun is with CSIRO Digital Productivity Flagship, PO Box 52, North Ryde, NSW 1670, Australia.
E-mail: changming.sun@csiro.au

Tuan D. Pham is with Aizu Research Cluster for Medical Engineering and Informatics, The University of Aizu, Fukushima 965-8580, Japan.
E-mail: tdpham@u-aizu.ac.jp

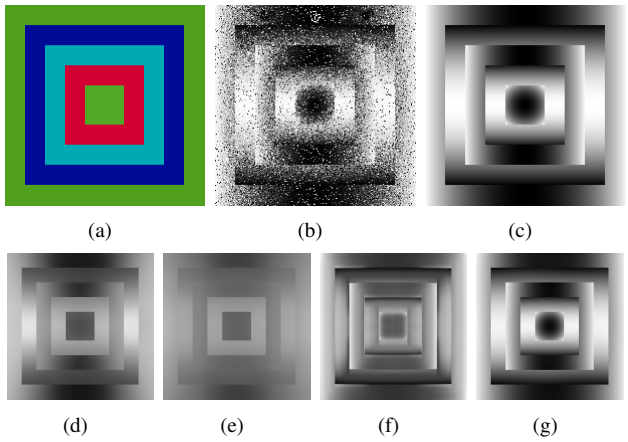

Fig. 1. Our method performs much better in preserving the spatial variation of the input image than existing filtering methods that use range guidance only (see text for details). All results are obtained with a fixed size of filtering kernel $r = 40$. (a) The guidance image I. (b) The input image P. (c) The reference of (b). (d) The result obtained by BF [9]. (e) The result obtained by GF [2]. (f) The result obtained by CLMF [36]. (g) The result obtained by our filter.

reversal artifacts as discussed in [33]. In recent years, local linear model (LLM) has drawn the wide attention from the community for its edge-awareness property [2], [34]. LLM assumes the filtering values are locally linear to the values in the guidance image. The guided filter (GF) [2] based on LLM is shown to be superior to BF based joint image filtering methods for its computational efficiency and a better gradient preservation property. Both BF and GF are averaging based filtering processes according to [2]. Although joint BF and GF are effective and efficient, they do not always provide desirable results because the constant support regions (squared windows) used in their methods lack spatial adaptivity. This will lead to artifacts in some applications, for example the bleeding artifact [35] at object boundaries in depth upsampling.

Many methods are proposed to achieve the spatial adaptivity such as the anisotropic estimator which is proved to be effective for image denoising [37]. However, any arbitrary shape of support regions in this method makes fast implementation impossible. To enable a fast aggregation over the adaptive support regions, a study in [36] extended the idea behind the GF and proposed a generalized multipoint filtering framework named cross-based local multipoint filtering (CLMF) which introduces the cross based aggregation method [38] and extends the LLM into a more general model. In [36], a support region is precalculated at each pixel, and the filtering process is then carried out within this region. CLMF is reported to be superior to BF and GF because the support region helps reduce the chance of incorrectly aggregating values of pixels in unrelated regions. However, the support region calculation method adopted in CLMF has limitations. First, although the aggregation can be carried out with $O(1)$ time, the complexity for calculating for support arms at each pixel depends on the maximum length of the filtering kernel r. This slows down the algorithm in the case of large r. Second, a hard decision is made on the support region by thresholding: some pixels inside the support region are totally ignored when carrying out aggregation.

In the filtering process, one would expect that noise is to be suppressed, and the spatial variation which usually contains the important information in the image is to be preserved. However, a common problem for all conventional joint or guided filters is that they do not perform well in preserving the spatial variation despite of being good at suppressing noises. For example, GF and CLMF consider the color similarity between pixels and the spatial distance between pixels when calculating weights, but they do not formulate the variation of pixel values with respect to the spatial position. As a result, it is very likely for the methods to change the filtering image by greatly suppressing the variation in regions where the values of the guidance image are similar. This artifact becomes particularly severe for large filter kernels. Let us look at an example as shown in Figure 1. In this case, the guidance image contains concentric squares of constant colors. The intensities in each square vary with some noises in the input image. This case is quite common in a number of graphics studies. For example, guided depth image denoising where the guidance image or color image contains an object of a single color whose curved surface contains noises in the depth image. Let us see what the filters, which are based on the local linear model, will do for this case. Under the assumption that the filtering output is locally linear to the guidance image, the values of pixels in the filtering output is assigned to be similar once these pixels have similar values in the guidance image regardless of the spatial position of the pixels. Therefore, the local linear model based filters such as GF and CLMF tend to oversmooth regions where values in the guidance image are close to each other. However, this formulation is obviously not working for the case as illustrated in Figure 1. The pointwise filtering methods such as the joint BF also have this oversmoothing artifact because it pushes the value of a pixel towards the mean or weighted mean value within its support region regardless the spatial relationship of the values of pixels. The oversmoothing effect becomes even more severe as the size of the filtering kernel increases.

For the sake of better modeling on spatial variation, a new filtering method is developed by combining LPA and LLM into the popular multipoint filtering framework [36]. We also proposed a spatially adaptive scheme by weighting pixels in the aggregation step for better edge-aware filtering. Combining these two techniques, our method achieves very good quantitative and qualitative results in a number of applications. In the aspect of algorithm complexity, our filtering method is a constant time filtering method where the computational complexity is constant with regard to the size of the filtering kernels, which means our method works for any large filtering kennels without increasing the running time.

Our method has two main differences to the original GF method [2]. (1) We for the first time introduce LPA into the guided image filtering for modeling on the spatial variation, which is not considered in [2]. (2) We extend the least square model in [2] to a weighted least square model where the weight is used to achieve spatial adaptivity for better edge preservation. Our earlier work [39] utilizes on a segmentation

based hard thresholding method to determine the weights in the weighted least square model. Unfortunately, any incorrect segmentation will introduce mis-leading information and hence degrade the performance of the filtering method. In this paper, we therefore develop a soft weighting scheme, and as will be shown in the experimental results section, by adopting this soft weighting scheme, our method outperforms the method in [39] in many applications. In summary, this paper presents a new guided image filtering method based upon the following three contributions:

1. We combined LPA and LLM into a multipoint filtering framework to achieve both spatial variation preservation and edge-aware filtering.
2. We developed a new weighting scheme with a constant time implementation to achieve spatial adaptive filtering.
3. We evaluated the proposed method in a variety of vision and graphics applications, and also compared it with many latest state-of-the-art filtering methods.

II. Local Multipoint filtering with LPA and Range Guidance

A. Algorithm Overview and Notations

The guidance image is denoted by I and the input image is denoted by P. In our method, Ω_p, the support region of a pixel $p \in P$, is a squared window centered at p with the size of $(2r+1) \times (2r+1)$. Denote x_p and y_p as the image coordinates of pixel p. The model regression is carried out within Ω_p and the calculated model is applied to all pixels in Ω_p. The model is based on both the range guidance and the local polynomial model. As the support regions are overlapping with each other, each pixel k in Ω_p will give an estimate q_k for p. The final estimate q to pixel p is calculated by aggregating or fusing the multiple estimates q_k.

The proposed filtering method contains the following steps: calculating the color difference between pixels of all 4-connected pairs, carrying out weighted model regression between P and I over each Ω_p, and producing the final output at p by aggregating the weighted values from the multiple estimates in Ω_p.

B. Multipoint LPA

To overcome the shortcoming of the original range guided filters, we employ the local polynomial approximation for modeling on spatial variation of pixel values of P over Ω_p. This process is explained by using a 1D signal as shown in Figure 2. Note that the horizontal coordinate is the spatial position of pixels rather than the color or intensity values in the guidance image. In Figure 2, the black lines are input signals, the red lines denote signals within a support region, and the green lines denote the results by using local polynomial regressions with polynomial functions of different orders.

For a given pixel p and its support region Ω_p, we assume that the values in Ω_p can be represented by a 2D polynomial function. This is reasonable in many cases, for example, for the depth value in a local region of a depth map or the intensity of pixels on a curved surface under a single light source. Denote the m-order polynomial within the support region of pixel k by $\mathcal{P}_k^m$ whose coefficients are $\boldsymbol{\alpha}_k^m = \left[\alpha_k^{01}, \alpha_k^{10}, \cdots, \alpha_k^{ij}, \cdots, \alpha_k^{0m}\right]$ and γ_k^0. The coefficient of nominal $x^i y^j$ is α_k^{ij}. The 2D polynomial model with zero-order ($m=0$), the first-order ($m=1$), and the second-order ($m=2$) are given as follows

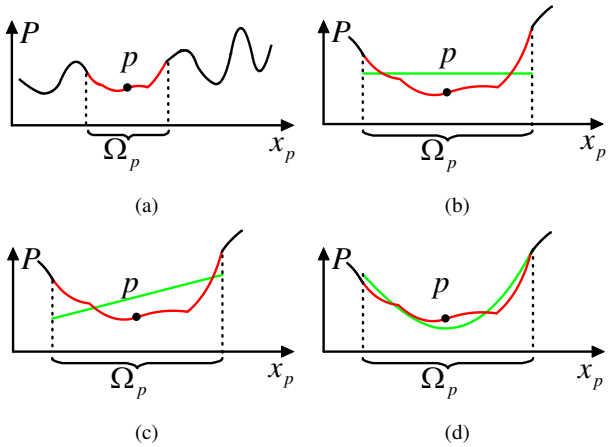

Fig. 2. Illustration for local polynomial approximation at a pixel of interest p in the 1D case. (a) The input signal P and the support region Ω_p of p. (b) Approximation results by using constant values or zero order polynomials. (c) Approximation results by using first order polynomials. (d) Approximation results by using second order polynomials.

$$\mathcal{P}_k^m = \begin{cases} \gamma_k^0 & m=0 \\ \gamma_k^0 + \alpha_k^{10} x + \alpha_k^{01} y & m=1 \\ \gamma_k^0 + \alpha_k^{10} x + \alpha_k^{01} y & \\ \quad + \alpha_k^{20} x^2 + \alpha_k^{11} xy + \alpha_k^{02} y^2 & m=2 \end{cases} \tag{1}$$

where x and y are the spatial coordinates. The multipoint local polynomial approximation with order "m" is named as MLPA-m in this paper. Denote the nominal of spatial coordinates of pixel p by a vector $\mathbf{S}_p^m = \left[x_p, y_p, \cdots, {x_p}^i {y_p}^j, \cdots, y_p^m\right]^{\mathrm{T}}$. As the support regions are overlapped with each other, we will have multiple estimates, $\mathcal{P}_k^m$, for pixel p from different k (k: $k \in \Omega_p$)

$$\mathcal{P}_k^m(p) = \boldsymbol{\alpha}_k^m \mathbf{S}_p^m + \gamma_k^0 \tag{2}$$

The final estimate for pixel p is given by a weighted averaging over all these estimates:

$$Q_{\mathcal{P}}(p) = \frac{\sum_{k:p\in\Omega_k} w_{pk} \mathcal{P}_k^m(p)}{\sum_{k:p\in\Omega_k} w_{pk}} \tag{3}$$

where w_{pk}, the weight of support from k to p, will be discussed later in Section III.

C. Multipoint LPA with Range Guidance

Since LPA only formulates smooth variation, this model does not preserve edges. Fortunately, LLM is able to preserve edges and its formulation has a similar form with LPA. Therefore, we combine LLM with LPA to achieve edge-aware filtering. Thus, the local model in Eq. (2) becomes

$$P_k^m(p) = \alpha_k^m \mathbf{S}_p^m + \gamma_k^0 + \beta_k \mathbf{I}_p + \gamma_k^1 \tag{4}$$

where the range guidance coefficients, $\boldsymbol{\beta}_k$ and γ_k^1, are computed from the support region around pixel k based on LLM; the range guidance, $\boldsymbol{I}_p$, is the intensity value for gray scale images or the RGB vector for color images. Define a hybrid guidance vector of $\mathbf{S}_p^m$ and $\boldsymbol{I}_p$ by $\mathbf{G}_p^m = \begin{bmatrix} \mathbf{S}_p^m \\ \boldsymbol{I}(p) \end{bmatrix}$. Reorganizing the coefficients of LPA and LLM and putting them into one vector: $\boldsymbol{\theta}_k^m = [\boldsymbol{\alpha}_k^m, \boldsymbol{\beta}_k]$, we rewrite Eq. (4) as

$$P_k^m(p) = \boldsymbol{\theta}_k^m \boldsymbol{G}_p^m + \gamma_k \tag{5}$$

where $\gamma_k = \gamma_k^1 + \gamma_k^0$. Similarly to Eq. (3), the final output is given by the weighted average:

$$Q(p) = \frac{\sum_{k\in\Omega_p} w_{pk}\left(\boldsymbol{\theta}_k^m \boldsymbol{G}_p^m + \gamma_k\right)}{\sum_{k\in\Omega_p} w_{pk}} \tag{6}$$

When using a color image as the guidance, the last three entries of $\mathbf{G}_p^m$ are the RGB channels and the last three entries of $\boldsymbol{\theta}_k^m$ are coefficients associated with the RGB channels respectively. For estimating model parameters $\boldsymbol{\theta}_k^m$, we use a weighted linear ridge regression model. Specifically, we minimize the following cost function in the support region of pixel k

$$E(\boldsymbol{\theta}_k^m, \gamma_k) = \sum_{s\in\Omega_k} w_{ks}\Big(\left(\boldsymbol{\theta}_k^m \boldsymbol{G}_s^m + \gamma_k - P(s)\right)^2 + \varepsilon_s \|\boldsymbol{\alpha}_k\|^2 + \varepsilon_r \|\boldsymbol{\beta}_k\|^2\Big) \tag{7}$$

where ε_s and ε_r are the regulation factors or weightings for spatial coordinates and range guidance respectively. The solution to Eq. (7) is given by

$$\boldsymbol{\theta}_k^m = \Big(\boldsymbol{\Sigma}_k(\boldsymbol{G}^m, P)\Big)^{\mathrm{T}} \Big(\boldsymbol{\Sigma}_k(\boldsymbol{G}^m, \boldsymbol{G}^m) + \boldsymbol{\varepsilon}\Big)^{-1} \tag{8}$$

$$\gamma_k = \bar{P}_k - \boldsymbol{\theta}_k^m \boldsymbol{\mu}_k \tag{9}$$

Here $\boldsymbol{\mu}_k$ and $\bar{P}_k$ are the weighted mean of $\boldsymbol{G}^m$ and P in Ω_k; $\boldsymbol{\Sigma}_k(\boldsymbol{G}^m, \boldsymbol{G}^m)$ and $\boldsymbol{\Sigma}_k(\boldsymbol{G}^m, P)$ are the weighted covariance matrices of $(\boldsymbol{G}^m, \boldsymbol{G}^m)$ and $(\boldsymbol{G}^m, P)$ in Ω_k respectively; $\boldsymbol{\varepsilon}$ is a diagonal matrix whose entries are ε_s or ε_r arranged in the order corresponding to entries in $\boldsymbol{G}^m$. The proof of Eq. (8) and Eq. (9) is given in the appendix. They depend on the order of the LPA and the number of channels in the range guidance.

III. Rectangle Based Weights

In this section, we discuss the weights as mentioned in Section II, w_{pk} and w_{ks}. Taking p as the pixel of interest, we use w_{pk} to encourage higher supports from the pixels near p and having a similar color with p. Here, we propose a new weighting method based on a rectangle which takes p and k as two opposite vertexes. w_{ks} is defined in the same way by taking k as the pixel of interest.

For a pixel k we consider two paths which source from k and sink at p. One path, $\mathcal{HP}_k$, goes vertically then horizontally; the other path, $\mathcal{VP}_k$, goes horizontally then vertically (see Figure 3). The weight (we assume that the guidance image is a color image) between neighboring two pixels p_1 and p_2 along one of the two paths ($\mathcal{HP}_k$ and $\mathcal{VP}_k$) is defined by

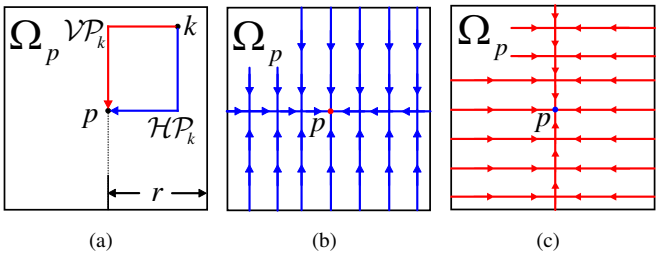

Fig. 3. Illustration of calculating the rectangular weights. (a) Two paths for calculating the rectangular weights. The blue line is path $\mathcal{HP}_k$ and the red line is path $\mathcal{VP}_k$. (b, c) Aggregating within Ω_p along $\mathcal{HP}_k$ and $\mathcal{VP}_k$.

$$\mathrm{dis}(p_1, p_2) = \frac{1}{3}\sum_{c=\{r,g,b\}} |I_c(p_1) - I_c(p_2)| \tag{10}$$

The weight between p and k is then given by

$$w_{pk} = \exp\left(-\frac{1}{\sigma_w}\sum_{p_1,p_2\in\mathcal{HP}_k} \mathrm{dis}(p_1, p_2)\right) + \exp\left(-\frac{1}{\sigma_w}\sum_{p_1,p_2\in\mathcal{VP}_k} \mathrm{dis}(p_1, p_2)\right) \tag{11}$$

where σ_w is the parameter controlling how the distance value is measured from the color difference. As the weight between two pixels is defined by a rectangle which takes these two pixels as vertexes, we call it "rectangular weights" (RWs). The value of RWs decreases when the edge of the rectangle intersects with color boundaries. Therefore, the weighted model regression in Eq. (7) and the weighted averaging in Eq. (6) do not have the problem of aggregating values across unconnected regions (see Figure 5).

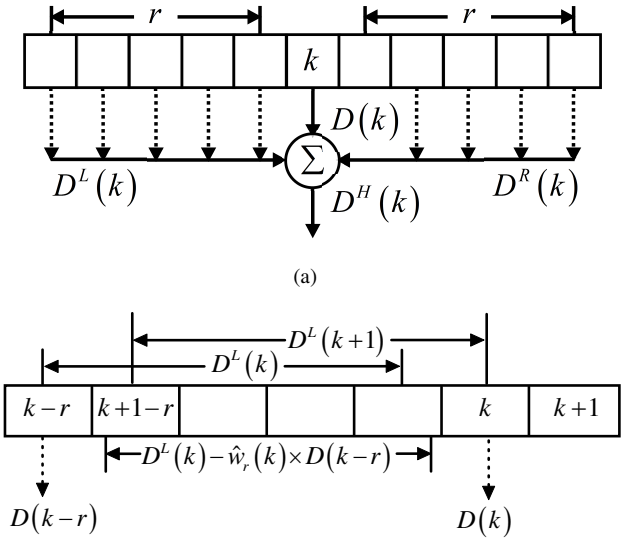

Fig. 4. Weighted aggregation along the horizontal direction. (a) The aggregated value is the summation of $D^L(k)$, $D^R(k)$, and $D(k)$. (b) Updating rules for computing $D^L(k)$ by using Eqs. (15) and (16).

Eq. (6) and Eq. (7) require the computing of a weighted mean value and weighted covariance matrix. Suppose I is a gray scale image, the weighted covariance value between the guidance image I and the filtering image P over a support region Ω_p is computed by

$$\begin{aligned} Var_p\left(I,P\right) &= \frac{1}{\sum\limits_{k\in\Omega_p} w_{pk}} \sum_{k\in\Omega_p} w_{pk}\left(I\left(k\right)-\bar{I}\right)\left(P\left(k\right)-\bar{P}\right) \\ &= \frac{1}{\sum\limits_{k\in\Omega_p} w_{pk}} \sum_{k\in\Omega_p} w_{pk} I\left(k\right) P\left(k\right) - \bar{I}\bar{P} \end{aligned} \tag{12}$$

where $\bar{I}$ and $\bar{P}$, the weighted average values over Ω_p, are computed from the weighted aggregation results divided by the weighted aggregation results of an all ones image. The weighted aggregation of $I\left(k\right)P\left(k\right)$ are computed in a similar manner from an image where the value of pixel k is the product of $I\left(k\right)$ and $P\left(k\right)$. Therefore, it is straightforward that for given weights, if we could compute the weighted aggregation of a specified image which can be an image with all ones, I, P or the product of I and P in a constant time with respect to the size of the support region. The computational complexity of our filtering method is constant. To simply the notation, we refer to the image where the weighted aggregation is applied as image D. We show that the computational complexity for the weighted aggregation over Ω_p is constant with respect to the size of Ω_p. The computational efficiency origins from the distance measurement between k and p as defined in Eq. (11). This cumulative formulation enables us to use the weighted moving sum method to efficiently compute the weights. The weighted moving sum method reuses the computed weighted summation values to compute the summation value at the current pixel. The weighted aggregation over a specified image D for p using RWs can be rewritten as the addition of two terms:

$$\begin{aligned} \sum_{k\in\Omega_p} w_{pk} D\left(k\right) = &\sum_{k\in\Omega_p} \prod_{p_1,p_2\in\mathcal{HP}_k} \exp\left(-\frac{1}{\sigma_w}\mathrm{dis}\left(p_1,p_2\right)\right) D\left(k\right) \\ &+ \sum_{k\in\Omega_p} \prod_{p_1,p_2\in\mathcal{VP}_k} \exp\left(-\frac{1}{\sigma_w}\mathrm{dis}\left(p_1,p_2\right)\right) D\left(k\right) \end{aligned} \tag{13}$$

where $D\left(k\right)$ is the value of the image being aggregated at pixel k. In the first term, the value is aggregated along $\mathcal{HP}_k$ where the aggregation is carried out vertically then horizontally. In the second term, the value is aggregated along $\mathcal{VP}_k$ where the aggregation is carried out horizontally then vertically. The horizontal aggregation fuses the weighted summation results from the left and right sides of a pixel, and the vertical aggregation adds the weighted value from the top and the bottom of a pixel. The aggregated value at pixel k along the horizontal direction, $D^H\left(k\right)$, is the summation of three terms

$$D^H\left(k\right) = D\left(k\right) + D^L\left(k\right) + D^R\left(k\right) \tag{14}$$

where $D^L\left(k\right)$ and $D^R\left(k\right)$ are the values aggregated from the left side and the right side of k (see Figure 4(a)). The calculation of $D^L\left(k\right)$ is carried out by recursively updating $D^L\left(k\right)$ and $\hat{w}_r\left(k\right)$, the weight of the pixel at a distance of r to k (see Figure 4(b)). $\hat{w}_r\left(k\right)$ is initialized to 1 and $D^L\left(k\right)$ is initialized to the value of D at the left most pixel along the horizontal line. The updating rules are

$$\hat{w}_r\left(k+1\right) = \hat{w}_r\left(k\right) \times \frac{\exp\left(-\frac{1}{\sigma_w}\mathrm{dis}\left(k,k+1\right)\right)}{\exp\left(-\frac{1}{\sigma_w}\mathrm{dis}\left(k-r,k-r+1\right)\right)} \tag{15}$$

and

$$\begin{aligned} D^L\left(k+1\right) = &\left(D^L\left(k\right) - \hat{w}_r\left(k\right)\times D\left(k-r\right) + D\left(k\right)\right) \\ &\times \exp\left(-\frac{1}{\sigma_w}\mathrm{dis}\left(k,k+1\right)\right) \end{aligned} \tag{16}$$

After updating $\hat{w}_r\left(k\right)$ and $D^L\left(k\right)$ from the left most pixel to the right most pixel, $D^L\left(k\right)$ for all k along the horizontal line will be obtained. $D^R\left(k\right)$ is calculated by reversing the updating direction, i.e., from the right most pixel to the left most pixel. The vertical aggregation is carried out in a similar manner along the vertical direction. Each of the two terms in Eq. (13) is computed by recursively performing weighted aggregation along horizontal and vertical directions. The final weighted aggregation is obtained by adding up the two terms. Algorithm 1 shows this process.

In [40], the similarity between two neighboring pixels is used to calculate the distance between two neighboring pixels in the transformed domain. Based on the distance, a recursive filtering method was proposed to achieve edge-aware filtering in the transformed domain. A similar scheme is adopted in recursive bilateral filtering [41] where the cumulative color similarity is used to define the range weight. Both methods filter a 2D image globally over the whole image by recursively applying 1D image filtering along horizontal and vertical directions. Our RWs also use the color similarity between two

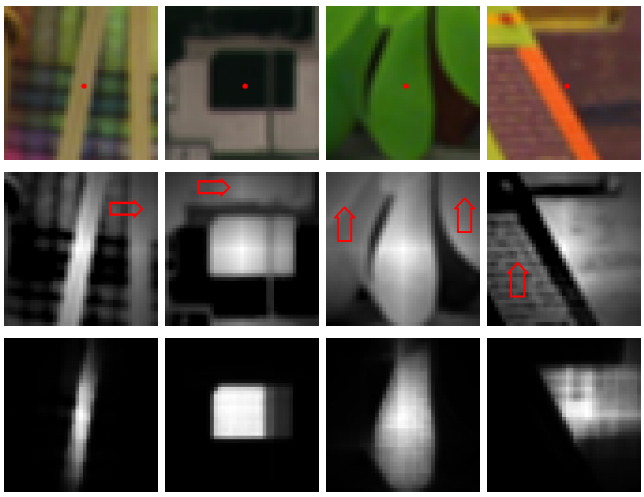

Fig. 5. Filter kernels. First row: Different image patches. The pixel of interest is colored in red. Second row: Guided filtering kernels. Red arrows denote the pixels which are assigned with undesired high weights. Last row: Our RWs. By using RWs, pixels with high weights are distributed in a homogenous region around the pixel of interest.

Algorithm 1:
Input: The guidance image I, the image to be aggregated D, parameters σ_w and r.
Output: Image of rectangularly weighted summation.

1. Calculate $\mathrm{dis}\,(p_1, p_2)$ for all 4-connected pairs: (p_1, p_2). Carry out weighted aggregation on D by recursively calculating Eq. (15) and Eq. (16) along horizontal direction to obtain D^H.
2. Carry out weighted aggregation on D^H by recursively calculating Eq. (15) and Eq. (16) along vertical direction and obtain $D^{\mathcal{VP}}$.
3. Carry out weighted aggregation on D by recursively calculating Eq. (15) and Eq. (16) along vertical direction to obtain D^V.
4. Carry out weighted aggregation on D^V by recursively calculating Eq. (15) and Eq. (16) along horizontal direction and obtain $D^{\mathcal{HP}}$.
5. Output the image $D^{\mathcal{VP}} + D^{\mathcal{HP}}$.

neighboring pixels, but formulate the weight in a different way where the image filtering is carried out in a closed form without using iterations. Besides, unlike the other two methods, our method uses only the pixels within a support window rather than all pixels, which provides users with more flexibility in choosing a desired support region. However, it is still easy to use all pixels when filtering by setting r large enough to cover the whole image. Comparing with CLMF, our method uses a soft weighting method to achieve spatial adaptivity rather than the hard thresholding method as used in CLMF. This modification provides better quantitative results in some specific applications such as depth enhancement which will be shown in Section V.

To summarize the proposed filtering method, we provide the pseudocode in Algorithm 2.

IV. Computational Complexity and Running Time

The computational cost of our algorithm consists of two parts, one is the time for weighted aggregation, the other is the time for model regression. The computational complexity of both parts is constant with respect to r, the size of the window. Therefore the computational complexity of our method is $O\,(1)$. The weighted summation which is required when computing Eq. (6), Eq. (8), and Eq. (9) takes 13 operations (addition, substraction, mutilation, and division). The cost for model regression mainly depends on the matrix inversion process. In MLPA-0, we invert a 3×3 matrix which is similar to that of GF and therefore their running time are similar. In MLPA-1 and MLPA-2, the matrix is increased to 5×5 and 8×8 respectively and leads to the increase of running time. The running time of the MLPA-m methods where "m" is larger than 2 is not tested in our experiments; but it can be predicted according to the complexity of inverting a matrix with respect to the matrix size. We test the computation time using a 32 bits PC with a 3.0 GHz CPU and 4 GB RAM in C++ without using any SIMD instructions. The comparison on running time for filtering approximately 10 million pixels with color guidance is given in Table I. It is worth mentioning

Algorithm 2:
Input: The guidance image I, the image to be filtered P, parameters σ_w and r.
Output: Filtering results Q.

1. Covariance matrix transformation:
   For all $k \in P$
   Decompose all entries of matrix $\boldsymbol{\Sigma}_k\,(\boldsymbol{G}^m, \boldsymbol{G}^m)$ and all entries of vector $\boldsymbol{\Sigma}_k\,(\boldsymbol{G}^m, P)$ into the form of weighted aggregation using Eq. (12).
   End for
2. Compute covariance matrices:
   Compute $\boldsymbol{\Sigma}\,(\boldsymbol{G}^m, \boldsymbol{G}^m)$ and $\boldsymbol{\Sigma}\,(\boldsymbol{G}^m, P)$ according to the results of step 1 by using Algorithm 1.
3. Compute parameters:
   For all $k \in \mathcal{P}$
   Compute $\boldsymbol{\theta}_k^m$ and γ_k using Eq. (8) and Eq. (9).
   End for
4. Build parameter images:
   Build parameter images, and each image has one entry of $\boldsymbol{\theta}^m$ or γ.
5. Compute the filtering results:
   5.1 Compute the numerator of Eq. (6) by using Algorithm 1 on parameter images and the results are then multiplied by $\boldsymbol{G}^m$.
   5.2 Compute the denominator of Eq. (6) by using Algorithm 1 on an all ones image.
6. Output results:
   Compute the final results by using Eq. (6) based on the results from step 5.

here that since the local linear model depends on the guidance image, the coefficients of the LLM and LPA combined model as defined in Eq. (4) cannot be computed through a single convolution which is a neatest approach to compute the original LPA [42].

TABLE I
Running time of different filtering methods.

Methods	Time (s)	
	$r = 9$	$r = 100$
Guided Image Filtering [2]	0.82	0.82
CLMF0 [36]	0.49	1.56
CLMF1 [36]	0.91	2.15
MLPA-0	0.89	0.89
MLPA-1	1.93	1.93
MLPA-2	5.05	5.05

V. Applications and Experiments

In this section, we show the performance of the proposed MLPA filter on a variety of computer vision and graphics applications.

A. Depth Image Enhancement

Depth image enhancement includes the depth image upsampling and depth image denoising. This enhancement can be carried out by adopting a joint filtering and using the color

TABLE II
DEPTH MAP UPSAMPLING RESULTS (IN MAD) ON THE MIDDLEBURY DATASETS AT FOUR SUBSAMPLING RATES.

	Art				Books				Moebius				Reindeer				Laundry				Dolls			
	×2	×4	×8	×16	×2	×4	×8	×16	×2	×4	×8	×16	×2	×4	×8	×16	×2	×4	×8	×16	×2	×4	×8	×16
BF	0.57	0.89	1.24	2.93	0.63	0.79	0.56	1.09	0.24	0.31	0.47	1.09	0.32	0.53	0.72	1.26	0.34	0.45	0.72	1.38	0.23	0.33	0.55	1.09
GF	0.63	1.02	1.71	3.46	0.21	0.32	0.54	1.14	0.23	0.37	0.58	1.16	0.43	0.53	0.87	1.83	0.38	0.52	0.96	1.91	0.28	0.37	0.56	1.13
RBF	0.83	0.93	1.14	1.89	0.63	0.67	0.74	0.96	0.64	0.68	0.79	1.11	0.66	0.71	0.83	1.18	0.69	0.74	0.85	1.20	0.61	0.67	0.78	1.05
CLMF0	0.44	0.76	1.23	2.94	0.15	0.28	0.54	1.03	0.17	0.29	0.56	1.03	0.34	0.51	0.84	1.51	0.31	0.52	0.81	1.63	0.25	0.33	0.62	1.02
CLMF1	0.44	0.76	1.44	2.87	0.14	0.28	0.51	1.20	0.19	0.31	0.51	0.86	0.35	0.47	0.74	1.32	0.29	0.43	0.78	1.69	0.27	0.32	0.61	1.04
MF0+RWs	0.49	0.72	1.21	2.39	0.14	0.26	0.46	0.97	0.20	0.29	0.53	0.91	0.31	0.48	0.62	1.02	0.27	0.41	0.63	1.37	0.22	0.33	0.58	0.94
MF1+RWs	0.49	0.72	1.37	2.39	0.15	0.27	0.49	1.07	0.24	0.34	0.57	0.82	0.39	0.41	0.67	1.03	0.29	0.38	0.67	1.43	0.26	0.33	0.60	0.99
JGF	0.29	0.47	0.78	1.54	0.15	0.24	0.43	0.81	0.15	0.25	0.46	0.80	0.23	0.38	0.64	1.09	0.21	0.36	0.64	1.20	0.19	0.33	0.59	1.06
ARM	0.18	0.49	0.64	2.01	0.12	0.22	0.37	0.77	0.10	0.22	0.41	0.79	0.22	0.40	0.58	1.00	0.20	0.34	0.53	1.12	0.21	0.34	0.50	0.82
JSDF	0.49	1.17	2.62	5.28	0.25	0.50	1.06	2.29	0.30	0.60	1.21	2.43	0.37	0.76	1.44	3.04	0.32	0.64	1.39	3.26	0.47	0.91	1.48	2.78
MPLA-1	0.49	0.64	1.18	2.00	0.16	0.26	0.42	0.75	0.19	0.28	0.51	0.80	0.26	0.40	0.59	0.96	0.25	0.38	0.59	1.25	0.20	0.31	0.47	0.77
MPLA-2	0.41	0.58	0.85	1.59	0.14	0.22	0.36	0.64	0.22	0.33	0.50	0.86	0.22	0.33	0.50	0.86	0.20	0.30	0.45	0.86	0.18	0.27	0.43	0.75
MPLA-2($\mathcal{HP}$)	0.41	0.58	0.87	1.64	0.14	0.22	0.36	0.65	0.22	0.33	0.50	0.89	0.26	0.40	0.59	0.96	0.20	0.30	0.46	0.90	0.18	0.28	0.44	0.76

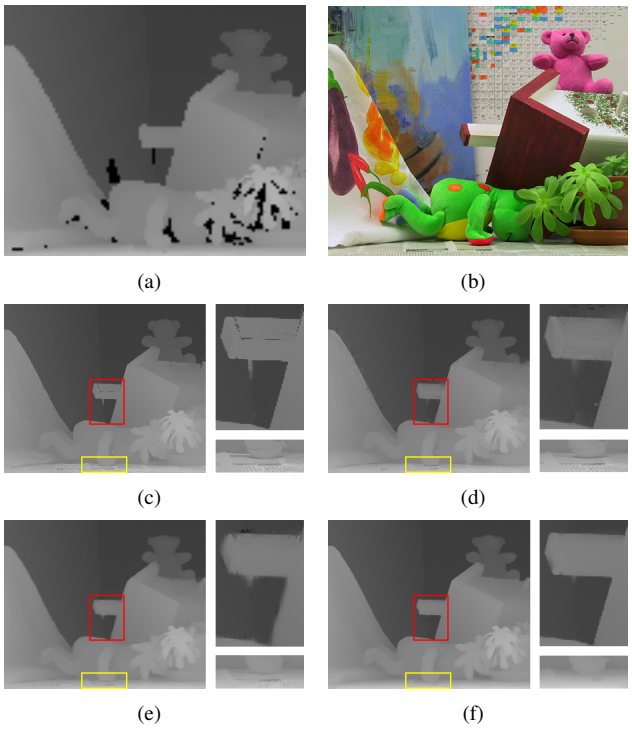

Fig. 6. Depth upsampling results with $r = 9$. (a) A low resolution depth image. Black pixels are void. (b) High resolution color image. (c) The results obtained by BF ($\sigma_s = 30, \sigma_r = 20/255$). (d) The results obtained by GF ($\varepsilon = 0.05^2$). (e) The results obtained by CLMF ($\varepsilon = 0.05^2, \tau = 20/255$). (f) The results by MLPA-1 ($\varepsilon_r = 0.1^2, \varepsilon_s = 0, \sigma_w = 40/255$).

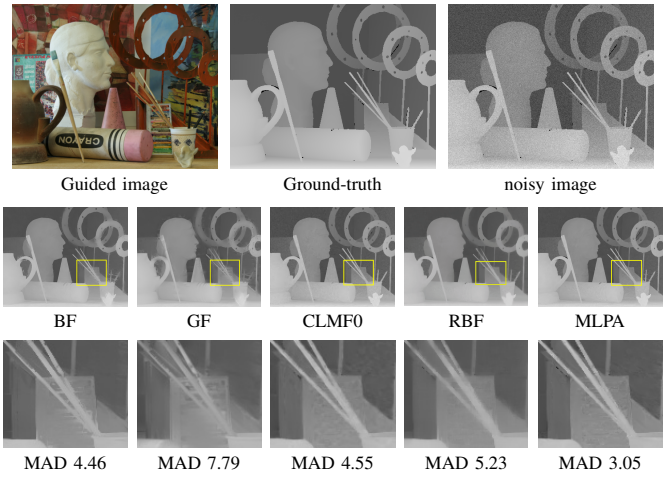

Fig. 7. Depth denoising results with $r = 9$. Parameters used are the same as those in upsampling experiments except $\tau = 5/255$ in CLMF0, and $\sigma_w = 20/255$ in MLPA for suppressing bleeding artifacts.

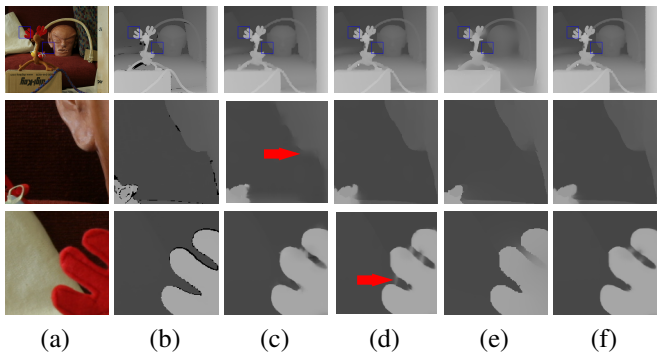

Fig. 8. Depth image upsampling results at upsamping rate 16. (a) Color images. (b) Groundtruth depth map. The dark pixels are void. (c) Results from MLPA-1 where boundaries are fattened. (d) Results from MLPA-2 ($\mathcal{HP}$) where bleeding artifacts happen in the concave region. (e) Results from JSDF [43]. (f) Results from MLPA-2.

image as the guidance image. The experiments are carried out on the Middlebury datasets [44] where the color images with approximately 1300×1200 pixels are provided along with the ground truth disparity maps indicating the depth of pixels in the color image. The first depth upsampling experiment is carried out at a upsampling rate of 4 on four standard testing images from the Middlebury datasets, and the methods to be compared with are the most popular filtering methods: BF, GF, and CLMF. The results are shown in Table III where the error threshold is 1 pixel (see [44] for details). We examined the performance of using path $\mathcal{HP}_k$ alone for defining weights rather than using both paths $\mathcal{HP}_k$ and $\mathcal{VP}_k$, and the results are listed as "MLPA-1($\mathcal{HP}$)". The results from the segmentation based MLPA method [39], "MLPA(Seg)", are also compared. To understand why our method performs well especially in preserving object surface, let us look at the bottom part (the regions in the yellow square) of the "Teddy" image (Figure 6) where the dark letters are printed on the slant newspapers. In this case, the local linear model that assumes the depth is linear to color values tends to assign a higher depth value to the pixels on newspaper than to those of dark letters. This model does not consider depth variation and fails to correctly recover the slant surface. However, our MLPA method produces good results in this region by using a plane to approximate the depth of the newspapers. Because the RWs are used in the

model regression and estimation step, the impact of the depth estimation value from unconnected regions is weak. Therefore, our MLPA provides much clearer boundaries. The MLPA outperforms MLPA(Seg) due to its soft weighting scheme, which will be further verified in later experiments.

To extensively compare with other state-of-the-art methods and show the effectiveness of our method, we conducted more experiments on the Middlebury datasets on the depth enhancement task. Here, our method is compared with seven methods: recursive bilateral filtering (RBF) [41], auto-regressive model upsampling (ARM), joint upsampling (JGF), BF, GF, CLMF and joint guidance filtering (JSDF) [43]. We also considered the performance of replacing the cross based region selection and weighting scheme in CLMF with our soft weighting scheme, which is listed as MF0+RWs and MF1+RWs. In this experiment, the performance with a single path version of MLPA-2 was evaluated for demonstrating the benefit of using two paths weighting scheme. Following the work in [45], we used the mean absolute difference (MAD) to measure the quality and the results are reported in Table II.

Although the performance of our method is not as good as ARM, a global optimization based method, our method achieves the best performance among all constant filtering methods in depth enhancement tasks. We would like to highlight that our MLPA-2 method has the best performance in all algorithms for large upsampling rates. As shown in Table II where MLPA-2 ($\mathcal{HP}_k$) and MLAP-2 achieve similar quantitative results, these two methods produce visually similar results as well. The difference between the two methods origins from treating the x/y axis symmetrically or not. This symmetrical treatment helps propagating the weight correctly especially at concave regions, and hence has a better performance but with a higher computational cost. Using two paths or one path is a tradeoff between accuracy and efficiency. The advantage of MLPA-2 over MLPA-1 comes from adopting a more flexible model to approximate the spatial variation, which provides clear boundaries and better suppression on bleeding artifacts. The qualitative comparison among MLPA-1, MLPA-2, and MLPA-2($\mathcal{HP}_k$) are provided in Figure 8 alongside with the results from JSDF [43].

In the depth denoising experiments, white noises are added to pixels in the depth image. Figure 7 shows the visual comparison on the results where the parameters are carefully tuned so that best results are presented. The experiments on depth denoising is conducted and the results are shown in Table IV.

TABLE III
ERROR PERCENTAGE OF DEPTH IMAGE UPSAMPLING RESULTS BY DIFFERENT FILTERS ON THE MIDDLEBURY DATASETS [44].

Datasets	BF	GF	CMLF	MLPA-1($\mathcal{HP}$)	MLPA(Seg)	MLPA-1
Tsukuba	5.35	7.03	5.67	4.83	**4.21**	4.24
Venus	1.29	1.66	1.19	0.48	0.72	**0.47**
Teddy	7.21	8.92	6.55	2.42	3.06	**2.34**
Cones	7.59	10.4	7.82	4.08	5.27	**4.06**

TABLE IV
DEPTH MAP DENOISING RESULTS (IN MAD) ON THE MIDDLEBURY DATASETS.

	Art	Books	Moebius	Reindeer	Laundry	Dolls
BF	1.52	1.07	1.20	1.25	0.84	0.94
GF	1.71	1.12	1.25	1.32	0.89	1.13
RBF	1.51	1.06	1.12	1.19	1.15	1.11
CLMF0	1.41	1.06	1.14	1.17	1.01	1.29
CLMF1	1.42	1.10	1.18	1.17	1.00	1.32
MLPA	1.31	1.21	0.82	0.89	0.78	0.79

B. Flash/No-Flash Denoising

Flash/no-flash denoising is for denoising a no-flash image based on a given guidance of the flash version of the same scene as described in [9]. RWs helps our MLPA work with a large filtering kennel, r, without over-smoothing the shadow regions and the specular reflection regions. Thanks to RWs, MLPA does not suffer from the misleading segmentation results which may degrade the performance of our previous segmentation based method MLPA(Seg) [39] (see Figure 10). In this application, we find that the results from the conventional filtering methods suffer from JPEG compression artifacts particularly at places where the blocky quality of the guidance image is obviously noticed. However, by using the first or second order polynomial approximation, our method can help suppress this JPEG compression artifacts. Note that the JPEG compression artifacts on the shadow regions and on the surface of porcelains are greatly suppressed by our method and the results at these regions are much better than conventional methods (see Figure 9). This benefit comes from using the polynomial approximation to model the spatial variation of the pixel values.

C. Graphics Applications

Our MLPA filter can also be used for graphics applications, such as detail enhancement and image abstraction [46]. Compared with GF, we find that our MLPA method is able to enhance details which are lost in the results from GF (see Figure 11). We now explain this through the example in Figure 11 where the dark pixels denoted by the yellow arrows are lost in the result from GF. As the filtering input is the guidance image itself in a detail enhancement task, the effect of the filtering is the pushing of the value of a pixel towards the mean value (for GF filtering) or the weighted mean value (for MLPA filtering) of the pixels around it. The color of the dark pixels between the flower petals will be pushed towards a darker/greener color, because the mean value within these parts is darker/greener than the dark pixels due to the existence of leaves in the background. As a result, these dark pixels are pushed towards lighter/reder together with other parts on the petals. This causes the lost of these dark pixels in the results. Thanks to RWs, the contribution of the background pixels to the weighted mean value is weak and the color of the dark pixels will be pushed towards a lighter/reder color. This makes the value of these pixels being pushed towards a different direction of other pixels on the petals. Consequently, these dark pixels are well preserved in the results. The illustration

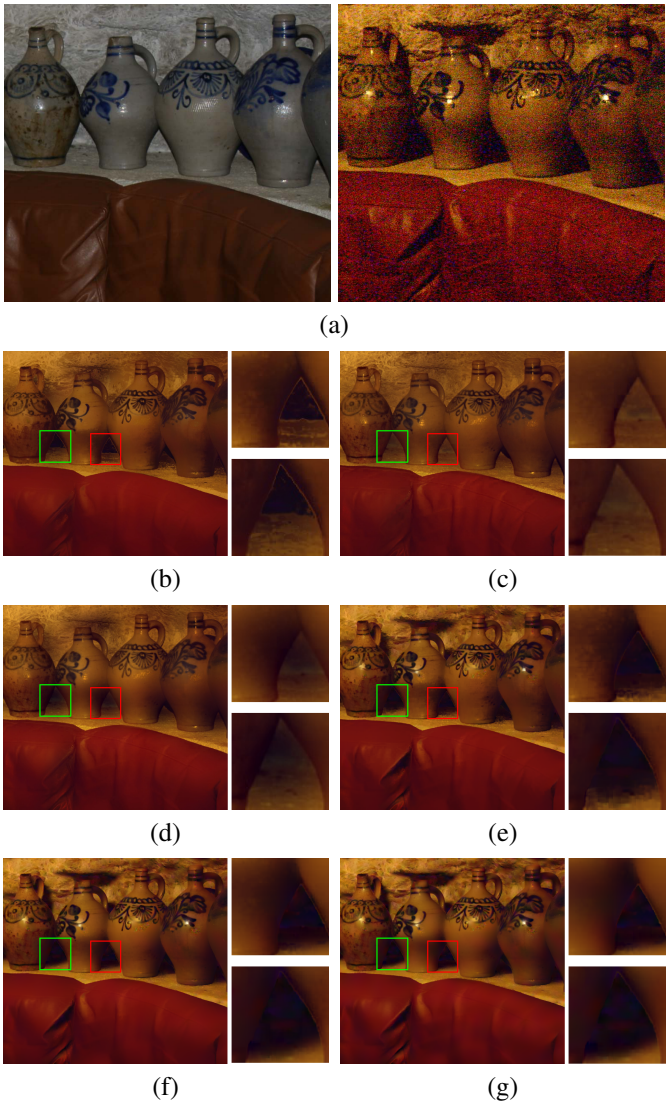

Fig. 9. Result of the flash/no flash denoising. (a) Flash guidance image (top) and the noised no flash image (bottom). (b)-(c) Filtering outputs by using the joint bilateral filter ($\sigma_s = 30, \sigma_r = 20/255$) and the guided filter ($\varepsilon = 0.02^2$) respectively. (d) Result obtained by using MLPA-1 with non-adaptive support regions ($\varepsilon_r = 0.02^2, \varepsilon_s = 0, \sigma_w = \infty$). (e)-(g) Results obtained by MLPA-0, MLPA-1, and MLPA-2 respectively with adaptive support regions ($\varepsilon_r = 0.02^2, \varepsilon_s = 0, \sigma_w = 30/255$). Our method recovers the spatial variation from noised signal very well (better viewed on a digital display).

of this reason is shown in Figure 11 (d). The weighted mean and mean value are denoted by horizontal dash lines. The values of the dark pixels, the pixels on the flower, and the pixels on the background are denoted in pink, red, and green respectively. The circles represent the values before filtering, and the triangles denote the values after filtering. The dark arrows near the circles indicate the direction of the pixel intensity change.

In image abstraction, an image is filtered recursively followed by a quantification procedure, then edges are extracted and added to the quantified image. The image abstraction results provided by our MLPA-1 filter are given in Figure 12.

VI. Discussions

To better understand why MLPA outperforms conventional

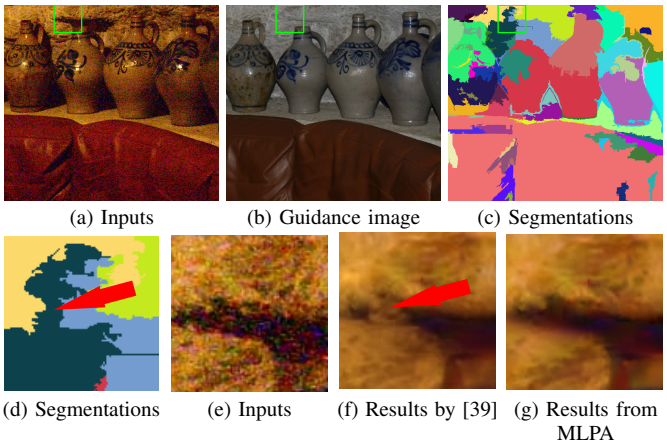

Fig. 10. Filtering results by MLPA(Seg) [39] (f) and our current MLPA method (g). MLPA(Seg) breaks the dark region in (e) due to the misleading segmentation results in (d). The results in (g) do not have this problem. Readers are referred to [39] for full scale results.

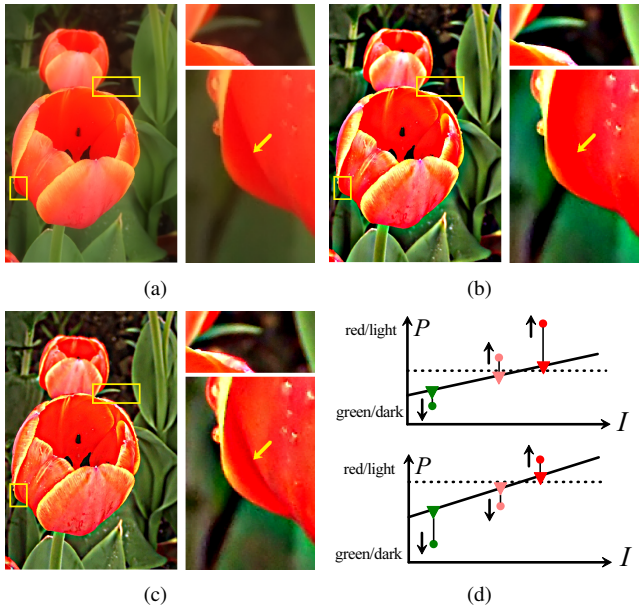

Fig. 11. Results of detail enhancement ($r = 30$, details are boosted by 5 times). Better viewed on a digital display. (a) Image to be enhanced. (b) The result by guided filter [2] ($\varepsilon = 0.3$). (c) The result by MLPA-1 ($\varepsilon_r = 0.3, \varepsilon_s = 0, \sigma_w = 100/255$). Note that details are lost in the results from [2]. (d) The value of the dark pixels between petals is pushed towards different directions by GF (top graph) and by MLPA (bottom graph).

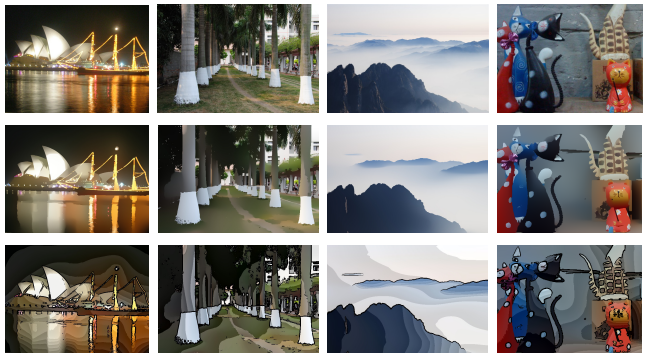

Fig. 12. Results of image abstraction using MLPA-1. First row is the original image. Middle row is the filtering output. Last row is the abstracted image with edges after quantization.

TABLE V
COMPARISON OF DIFFERENT FILTERS.

Filters	Filtering model	Spatial variation preserving	Edge-aware	Adaptivity in aggregation
BF [1]	Weighted averaging	N	Y	N
GF [2]	Linear model on range	N	Y	N
CLMF [36]	Linear model on range	N	Y	Hard thresholding
MLPA(Seg)	Linear model range+LPA	Y	Y	Hard thresholding
MLPA	Linear model range+LPA	Y	Y	Soft decision

filtering methods which do not consider spatial variation (or why preserving spatial variation is important), let us look at an example as shown in Figure 13. In this example, we intend to repair the depth map (a) from a given guidance image (b). A large r ($= 150$) is used so that the large damaged region can be covered. MLPA-0 and MLPA-1 outperform GF [2], because MLPA-0 and MLPA-1 employ RWs which help prevent aggregating contributions from unconnected regions. First order polynomials in MLPA-1 implicitly use slant planes for approximating local depth values, which are more suitable than the frontal planes with MLPA-0. Therefore, MLPA-1 provides better results than MLPA-0.

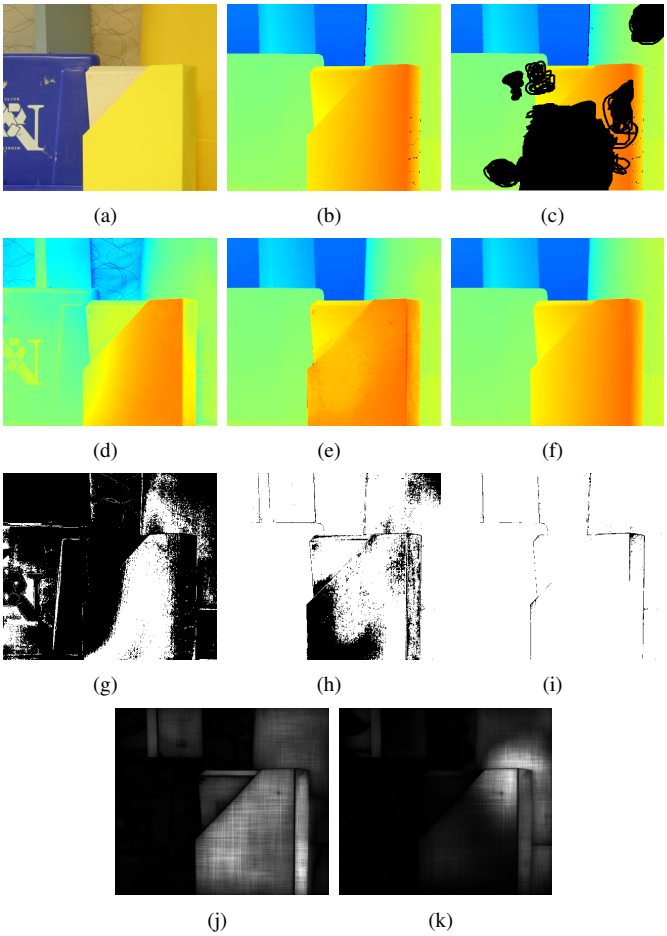

Fig. 13. Results of depth completion. (a) Guidance image. (b) Groundtruth depth map. (c) Damaged depth image. (d)-(f) Results from GF, MLPA-0, and MLPA-1. Without considering spatial variation, GF and MLPA-0 fail to provide accurate results. (g)-(i) Error maps (pixels with depth error larger than 1 pixel are colored in dark). (j) and (k) Norm of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ at each pixel (higher intensity indicates a higher value). LPA models at the slant surface, and LMM takes effect around weak boundaries.

Another interesting question is how LPA, LLM and RWs interplay to realize edge-aware filtering while preserving the spatial variation. Generally speaking, LLM and RWs help realize edge-aware filtering, and LPA is for preserving the spatial variability. In regions where the value of filtering input varies smoothly over spatial coordinates rather than the range value of the guidance image, such as the surface at the bottom of the "Teddy" image or the surface of the plastic box in Figure 13(a), LPA models the spatial variability. At places around strong edges where the values of the input image and the guidance image both change dramatically within a small region, the LLM coefficients are pulled down to zero. The reason is that contributions cannot be propagated cross regions at these places owing to the RWs. However, for places around weak edges where the values of the input image change dramatically but the guidance image does not change very much, LLM takes effect for preserving edges in the input image. When the LPA models the spatial variability well (when the squared term in Eq. (7) is small), the LLM coefficients are pulled down to zero. Because the coefficients of both the LPA model and the LLM model are computed together, our method is able to adaptively choose either the LPA model or the LLM model or both of them. Although the example shown in Figure 13 contains comparatively simple structures, this example emphasizes the effectiveness of introducing LPA into depth enhancement tasks. Not surprisingly, if the depth information is totally lost in a large region containing complex structure, it will be difficult for any method to correctly reconstruct this region. In our MLPA, the parameter σ_w controls the spatial adaptivity which helps to preserve edges. The larger the σ_w becomes, the more equally neighboring points will be considered during model regression. Generally, a large σ_w value will weaken the spatial adaptivity and may lead to undesirable results such as the bleeding or fatten effect. When setting σ_w to an extremely large value, the spatial adaptivity will be totally lost. In this case, our method is carried out with non-adaptive windows as shown in Figure 9 (d) where the dark regions denoted by squares are brightened due to the fatten artifact. For the purpose of better preservation on spatial variation, parameter ε_s is usually set to a small value or to 0. It is seen from Eq. (7) that increasing ε_s will push $\boldsymbol{\alpha}_k$ towards a zero vector, and hence suppress the spatial variation. For example, when ε_s is set to an extremely large value, it will force $\boldsymbol{\alpha}_k$ to be a zero vector, leading to a filtering method, MLPA-0, without considering spatial variation. In the results of this method (as shown in Figure 13 (e)), the slant surface on the yellow plastic box is not well recovered due to the lack of modeling on spatial variation.

For comparison, we summarize the characteristics of our MLPA and other filtering methods including BF, GF, CLMF, and MLPA(Seg) in Table V. On the aspect of filtering model, our MLPA utilizes a polynomial to model the spatial variation and employ local linear model over the range guidance to preserve edges; while all other filters do not consider the spatial variation. Therefore, our filter does a much better job in preserving the spatial variation than the others. In the models of GF or CLMF, the filtering output are treated to be locally linear to the value of the guidance image or to be locally constant over all neighboring pixels. All other models can be regarded as specific instances of our model in which no spatial variation is modeled (0-order polynomials), or equivalently setting $\varepsilon_s = 0$. In the regression and the estimation steps, CLMF, MLPA(Seg), and MLPA methods have the adaptability of selecting related pixels around the pixel of interest during model regression. However, the soft strategy by weighting the value of a pixel in our method has a better performance than the hard thresholding method as used in CLMF and segmentation based method as adopted in MLPA(Seg). To achieve good results, constant BF requires large quantizing levels. Therefore the constant BF is usually slower than other methods. Our method is fast when using first or second order polynomials, but the running time increases very quickly for higher order polynomials.

However, since the RWs take into account the pixel connectivity information, thus it is not suitable for filtering textured regions. An example is presented in Figure 14. Visual comparison shows unlike filtering methods which ignore the spatial relationships, the proposed method does not provide enough suppression to noises in texture regions. A possible solution proposed in [41] is pre-filtering the guidance image with a median filter so that the large oscillation in the image gradient can be removed. We used our MLPA as a cost volume filtering method for solving multi-labeling tasks such as interactive segmentation and stereo matching. Unfortunately, we found that adding the local polynomial term does not improve the result and sometimes even degrades the performance. It is possible that because the cost value which implicitly indicates the labelling likelihood usually keeps unchanged within regions and changes rapidly at region boundaries. As a result, being good at formulating smooth variation within regions, the local polynomial model therefore does not work well in formulating such cost value variation.

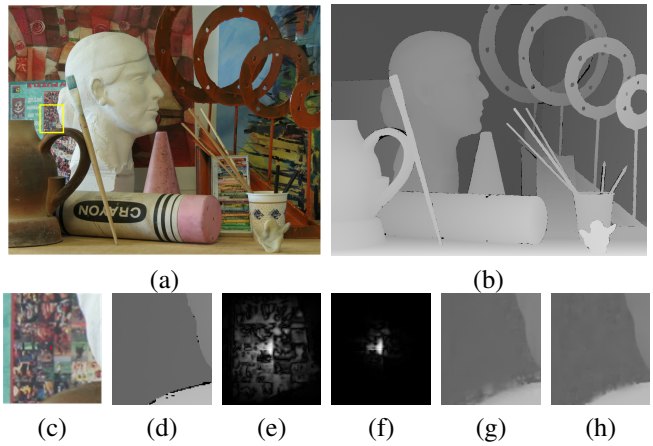

Fig. 14. A failure case where noises in textured regions are not fully suppressed. (a) Guidance image. (b) Groundtruth depth map of (a). (c) Close-up view of the region indicated by the red square. (d) Groundtruth depth map of (c). (e) Filtering kernel of [2] for the red pixel in (c). (f) Filtering kernel of MLPA-0 for the red pixel in (c). (g) Results from [2]. (h) Results from MPLA-0.

VII. Conclusions

This paper presented a new image filtering method which incorporates the LPA with range guidance into a multipoint estimation framework for the first time. The proposed hybrid guided image filtering method has the desired property of better preserving the spatial variation in an input image. We also proposed a weighting scheme with an efficient implementation in constant time to adaptively aggregate values from connected regions. Experiments on a number of applications verify the effectiveness of the proposed method and show its unique property over the existing guided or joint image filtering methods. The comparisons with the state-of-the-art methods also indicate that our MLPA filter is very competitive. Our further work will focus on developing new local models for reducing the computational cost for higher order approximations. In addition, we also plan to investigate the performance of our filter with GPUs acceleration.

Acknowledgment

The authors would like to thank Dr. Kaiming He at Microsoft Research Asia and Prof. Hugues Hoppe at Microsoft Research for providing testing images, and anonymous reviewers for their very constructive comments. A preliminary version of this work appeared in CVPR2014 [39].

References

[1] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *International Conference on Computer Vision*, 1998, pp. 839–846.
[2] K. He, J. Sun, and X. Tang, "Guided image filtering," in *European Conference on Computer Vision*, 2010, pp. 1–14.
[3] Y. Peng, J. Suo, Q. Dai, and W. Xu, "Reweighted low-rank matrix recovery and its application in image restoration," *IEEE Transactions on Cybernetics*, vol. 44, no. 12, pp. 2418–2430, 2014.
[4] R. Yan, L. Shao, and Y. Liu, "Nonlocal hierarchical dictionary learning using wavelets for image denoising," *IEEE Transactions on Image Processing*, vol. 22, no. 12, pp. 4689–4698, 2013.
[5] R. Yan, L. Shao, L. Liu, and Y. Liu, "Natural image denoising using evolved local adaptive filters," *Signal Processing*, vol. 103, pp. 36–44, 2014.
[6] Z. Liu, S. Xu, C. P. Chen, Y. Zhang, X. Chen, and Y. Wang, "A three-domain fuzzy support vector regression for image denoising and experimental studies," *IEEE Transactions on Cybernetics*, vol. 44, no. 4, pp. 516–525, 2014.
[7] H. Zhang, J. Yang, Y. Zhang, and T. S. Huang, "Image and video restorations via nonlocal kernel regression," *IEEE Transactions on Cybernetics*, vol. 43, no. 3, pp. 1035–1046, 2013.
[8] L. Shao, R. Yan, X. Li, and Y. Liu, "From heuristic optimization to dictionary learning: a review and comprehensive comparison of image denoising algorithms," *IEEE Transactions on Cybernetics*, vol. 44, no. 7, pp. 1001–1013, 2014.
[9] G. Petschnigg, R. Szeliski, M. Agrawala, M. Cohen, H. Hoppe, and K. Toyama, "Digital photography with flash and no-flash image pairs," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 664–672, 2004.
[10] L. De-Maeztu, S. Mattoccia, A. Villanueva, and R. Cabeza, "Linear stereo matching," in *International Conference on Computer Vision*, 2011, pp. 1708–1715.

[11] C. Rhemann, A. Hosni, M. Bleyer, C. Rother, and M. Gelautz, "Fast cost-volume filtering for visual correspondence and beyond," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 3017–3024.
[12] S. Zhu, L. Zhang, and H. Jin, "A locally linear regression model for boundary preserving regularization in stereo matching," in *European Conference on Computer Vision*, 2012, pp. 101–115.
[13] K. J. Yoon and I. S. Kweon, "Adaptive support-weight approach for correspondence search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 650–656, 2006.
[14] A. Levin, D. Lischinski, and Y. Weiss, "Colorization using optimization," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 689–694, 2004.
[15] Y. Ding, J. Xiao, and J. Yu, "Importance filtering for image retargeting," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 89–96.
[16] C. Liu, W. T. Freeman, R. Szeliski, and S. B. Kang, "Noise estimation from a single image," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, 2006, pp. 901–908.
[17] R. Fattal, M. Agrawala, and S. Rusinkiewicz, "Multiscale shape and detail enhancement from multi-light image collections." *ACM Trans. Graph.*, vol. 26, no. 3, pp. 51–59, 2007.
[18] B. Ni, Y. Pei, P. Moulin, and S. Yan, "Multilevel depth and image fusion for human activity detection," *IEEE Transactions on Cybernetics*, vol. 43, no. 5, pp. 1383–1394, 2013.
[19] M. Camplani, T. Mantecon, and L. Salgado, "Depth-color fusion strategy for 3-d scene modeling with kinect," *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 1560–1571, 2013.
[20] J. Han, L. Shao, D. Xu, and J. Shotton, "Enhanced computer vision with microsoft kinect sensor: A review," *IEEE Transactions on Cybernetics*, vol. 43, no. 5, pp. 1318–1334, 2013.
[21] D. Brownrigg, "The weighted median filter," *Communications of the ACM*, vol. 27, no. 8, pp. 807–818, 1984.
[22] Z. Ma, K. He, Y. Wei, J. Sun, and E. Wu, "Constant time weighted median filtering for stereo matching and beyond," in *International Conference on Computer Vision*, December 2013, pp. 49–56.
[23] F. C. Crow, "Summed-area tables for texture mapping," in *ACM SIGGRAPH Computer Graphics*, vol. 18, no. 3, 1984, pp. 207–212.
[24] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, 2001, pp. 1–8.
[25] L. Xu, C. Lu, Y. Xu, and J. Jia, "Image smoothing via L_0 gradient minimization," *ACM Transactions on Graphics*, vol. 30, no. 6, pp. 252–265, 2011.
[26] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, no. 1, pp. 259–268, 1992.
[27] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering," *IEEE Transactions on Image Processing*, vol. 16, no. 8, pp. 2080–2095, 2007.
[28] S. Paris and F. Durand, "A fast approximation of the bilateral filter using a signal processing approach," in *European Conference on Computer Vision*, 2006, pp. 568–580.
[29] F. Porikli, "Constant time O(1) bilateral filtering," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.
[30] A. Adams, N. Gelfand, J. Dolson, and M. Levoy, "Gaussian KD-trees for fast high-dimensional filtering," *ACM Transactions on Graphics*, vol. 28, no. 3, pp. 1–12, 2009.
[31] Q. Yang, K.-H. Tan, and N. Ahuja, "Real-time O(1) bilateral filtering," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 557–564.
[32] Q. Yang, S. Wang, and N. Ahuja, "SVM for edge-preserving filtering," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 1775–1782.
[33] Z. Farbman, R. Fattal, D. Lischinski, and R. Szeliski, "Edge-preserving decompositions for multi-scale tone and detail manipulation," *ACM Transactions on Graphics*, vol. 27, no. 3, pp. 1–10, 2008.
[34] A. Levin, D. Lischinski, and Y. Weiss, "A closed-form solution to natural image matting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 228–242, 2008.
[35] J. Park, H. Kim, Y.-W. Tai, M. S. Brown, and I. Kweon, "High quality depth map upsampling for 3D-TOF cameras," in *International Conference on Computer Vision*, 2011, pp. 1623–1630.
[36] J. Lu, K. Shi, D. Min, L. Lin, and M. N. Do, "Cross-based local multipoint filtering," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 430–437.
[37] V. Katkovnik, A. Foi, K. Egiazarian, and J. Astola, "From local kernel to nonlocal multiple-model image denoising," *International Journal of Computer Vision*, vol. 86, no. 1, pp. 1–32, 2010.
[38] K. Zhang, J. Lu, and G. Lafruit, "Cross-based local stereo matching using orthogonal integral images," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 7, pp. 1073–1079, 2009.
[39] X. Tan, C. Sun, and T. D. Pham, "Multipoint filtering with local polynomial approximation and range guidance," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, 2014, pp. 1–8.
[40] E. S. L. Gastal and M. M. Oliveira, "Domain transform for edge-aware image and video processing," *ACM Transactions on Graphics*, vol. 30, no. 4, pp. 69:1–69:12, 2011.
[41] Q. Yang, "Recursive bilateral filtering," in *European Conference on Computer Vision*, 2012, pp. 399–413.
[42] K. Vladimir, "Multiresolution local polynomial regression: A new approach to pointwise spatial adaptation," *Digital Signal Processing*, vol. 16, no. 5, pp. 73–116, 2005.
[43] B. Ham, M. Cho, and J. Ponce, "Robust image filtering using joint static and dynamic guidance," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.
[44] *http://www.vision.middlebury.edu/stereo/*, 2013.
[45] J. Yang, X. Ye, K. Li, C. Hou, and Y. Wang, "Color-guided depth recovery from RGB-D data using an adaptive autoregressive model," *IEEE Transactions on Image Processing*, vol. 23, no. 8, pp. 3443–3458, 2014.
[46] H. Winnemöller, S. C. Olsen, and B. Gooch, "Real-time video abstraction," *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 1221–1226, 2006.
[47] Z. Xu, "On the second-order statistics of the weighted sample covariance matrix," *IEEE Transactions on Signal Processing*, vol. 51, no. 2, pp. 527–534, 2003.

Xiao Tan received his Ph.D. degree in the area of computer vision from the University of New South Wales, Sydney in 2014. His research interest includes computer vision, pattern recognition, and image processing. He is now working in the HongKong Univeristy as a research fellow. He has contributed several papers on prestigious conferences and journals such as CVPR, ECCV and Pattern Recognition. He served as reviewers for Image and Vision Computing and IEEE Transactions on Circuits and Systems for Video Technology.

Changming Sun received the PhD degree in the area of computer vision from Imperial College London in 1992. Then, he joined CSIRO Computational Informatics, Australia, where he is currently a principal research scientist carrying out research and working on applied projects. His research interests include computer vision, image analysis, and pattern recognition. He has served on the program/organizing committees of various international conferences. He is an Associate Editor for EURASIP Journal on Image and Video Processing, a SpringerOne journal. Dr. Sun is a member of the Australian Pattern Recognition Society.

Tuan D. Pham received his PhD degree in 1995 from the University of New South Wales, Sydney, Australia. He is Professor of Biomedical Engineering, The Institute of Technology, Linkoping University, Sweden. Prior to this current position, he held positions as Professor and Leader of the Aizu Research Cluster for Medical Engineering and Informatics at The University of Aizu, Japan; and Group Leader of Bioinformatics Research at The University of New South Wales, Canberra, Australia. His current research interests include image processing, pattern recognition, chaos and nonlinear dynamics applied to biology and medicine.

APPENDIX
DERIVATION OF THE MODEL COEFFICIENTS

The cost function for calculating model coefficients is defined as

$$E\left(\boldsymbol{\theta}_k^m,\gamma_k\right)=\sum_{s\in\Omega_k}w'_s\Big(\left(\boldsymbol{\theta}_k^m\boldsymbol{G}_s^m+\gamma_k-P\left(s\right)\right)^2 +\varepsilon_s\|\boldsymbol{\alpha}_k\|^2+\varepsilon_r\|\boldsymbol{\beta}_k\|^2\Big) \tag{17}$$

Define the following matrix notations: $\boldsymbol{G}$ is a matrix with the sth column being $\boldsymbol{G}_s^m$. $\boldsymbol{W}$ is a diagonal matrix whose sth entry is w'_s. $\boldsymbol{P}$ and $\boldsymbol{\gamma}$ are $1\times|\Omega_k|$ vectors with the sth value being $P\left(s\right)$ and γ_k, respectively. $\boldsymbol{\varepsilon}$ is a diagonal matrix where the entries are ε_s or ε_r arranged in an order corresponding to entries in $\boldsymbol{G}_s^m$. The partial derivative of this function with respect to γ_k and $\boldsymbol{\theta}_k^m$ is given by

$$\frac{\partial E}{\partial\gamma_k}=2\sum_{s\in\Omega_k}w'_s\left(\boldsymbol{\theta}_k^m\boldsymbol{G}_s^m+\gamma_k-P\left(s\right)\right) \tag{18}$$

$$\frac{\partial E}{\partial\boldsymbol{\theta}_k^m}=2\Big(\boldsymbol{\theta}_k^m\boldsymbol{G}\boldsymbol{W}\boldsymbol{G}^{\mathrm{T}}+\sum_{s\in\Omega_k}w'_s\boldsymbol{\theta}_k^m\boldsymbol{\varepsilon} -(\boldsymbol{P}-\boldsymbol{\gamma})\boldsymbol{W}\boldsymbol{G}^{\mathrm{T}}\Big) \tag{19}$$

As E is in a quadratic from, its minimum value can be found by solving: $\frac{\partial E}{\partial\gamma_k}=0$ and $\frac{\partial E}{\partial\boldsymbol{\theta}_k^m}=0$. We set the derivative of Eq. (18) to 0 and solve for γ_k

$$\begin{aligned}\gamma_k&=\frac{1}{\sum\limits_{s\in\Omega_k}w'_s}\sum_{s\in\Omega_k}w'_sP\left(s\right)-\boldsymbol{\theta}_k^m\frac{1}{\sum\limits_{s\in\Omega_k}w'_s}\sum_{s\in\Omega_k}w'_s\boldsymbol{G}_s^m\\&=\bar{P}_k-\boldsymbol{\theta}_k^m\boldsymbol{\mu}_k\end{aligned} \tag{20}$$

where $\boldsymbol{\mu}_k$ and $\bar{P}_k$ are the weighted mean of $\boldsymbol{G}_s^m$ and P in Ω_k. By substituting Eq. (20) into Eq. (19) and setting it to 0, we solve for $\boldsymbol{\theta}_k^m$

$$\boldsymbol{\theta}_k^m=\left(\boldsymbol{P}-\left[\bar{P}_k\right]\right)\boldsymbol{W}\boldsymbol{G}^{\mathrm{T}}\Big(\boldsymbol{G}\boldsymbol{W}\boldsymbol{G}^{\mathrm{T}} -\boldsymbol{\mu}_k\boldsymbol{U}\boldsymbol{W}\boldsymbol{G}^{\mathrm{T}}+\sum_{s\in\Omega_k}w'_s\varepsilon\Big)^{-1} \tag{21}$$

where $\left[\bar{P}_k\right]$ and $\boldsymbol{U}$ are $1\times|\Omega_k|$ vectors where all entries are equal to $\bar{P}_k$ and 1 respectively. Denote the weighted covariance matrix of $\boldsymbol{G}^m$ by $\boldsymbol{\Sigma}_k\left(\boldsymbol{G}^m,\boldsymbol{G}^m\right)$ and the weighted covariance matrix between $\boldsymbol{G}^m$ and P by $\boldsymbol{\Sigma}_k\left(\boldsymbol{G}^m,P\right)$. According to [47], for non-negative weights the weighted covariance matrix can be calculated by

$$\boldsymbol{\Sigma}_k\left(\boldsymbol{G}^m,\boldsymbol{G}^m\right)=\frac{\left(\boldsymbol{G}-\boldsymbol{\mu}_k\boldsymbol{U}\right)\boldsymbol{W}\left(\boldsymbol{G}-\boldsymbol{\mu}_k\boldsymbol{U}\right)^{\mathrm{T}}}{\sum\limits_{s\in\Omega_k}w'_s} \tag{22}$$

$$\boldsymbol{\Sigma}_k\left(\boldsymbol{G}^m,P\right)=\frac{\left(\boldsymbol{P}-\left[\bar{P}_k\right]\right)\boldsymbol{W}\left(\boldsymbol{G}-\boldsymbol{\mu}_k\boldsymbol{U}\right)^{\mathrm{T}}}{\sum\limits_{s\in\Omega_k}w'_s} \tag{23}$$

Then Eq. (21) can be rewritten as

$$\begin{aligned}\boldsymbol{\theta}_k^m&=\Big(\frac{1}{\sum\limits_{s\in\Omega_k}w'_s}\left(\boldsymbol{P}-\left[\bar{P}_k\right]\right)\boldsymbol{W}\boldsymbol{G}^{\mathrm{T}}\Big)\Big(\boldsymbol{\Sigma}_k\left(\boldsymbol{G}^m,\boldsymbol{G}^m\right)+\boldsymbol{\varepsilon}\Big)^{-1}\\&=\Big(\frac{1}{\sum\limits_{s\in\Omega_k}w'_s}\sum_{s\in\Omega_k}w'_sP\left(s\right)\boldsymbol{G}_s^m-\bar{P}_k\boldsymbol{\mu}_k\Big)^{\mathrm{T}}\Big(\boldsymbol{\Sigma}_k\left(\boldsymbol{G}^m,\boldsymbol{G}^m\right)+\boldsymbol{\varepsilon}\Big)^{-1}\\&=\Big(\boldsymbol{\Sigma}_k\left(\boldsymbol{G}^m,P\right)\Big)^{\mathrm{T}}\Big(\boldsymbol{\Sigma}_k\left(\boldsymbol{G}^m,\boldsymbol{G}^m\right)+\boldsymbol{\varepsilon}\Big)^{-1}\end{aligned} \tag{24}$$

Eq. (20) and Eq. (24) are the expressions of the model coefficients.