Multipoint Filtering with Local Polynomial Approximation and Range Guidance

Xiao Tan[*†], Changming Sun[†] and Tuan D. Pham[‡]
[*]The University of New South Wales, Canberra, ACT 2600, Australia
[†]CSIRO Computational Informatics, North Ryde, NSW 1670, Australia
[‡]The University of Aizu, Fukushima, Japan
{tanxchong@gmail.com} {changming.sun@csiro.au} {tdpham@u-aizu.ac.jp}

Abstract

This paper presents a novel guided image filtering method using multipoint local polynomial approximation (LPA) with range guidance. In our method, the LPA is extended from a pointwise model into a multipoint model for reliable filtering and better preserving image spatial variation which usually contains the essential information in the input image. In addition, we develop a scheme with constant computational complexity (invariant to the size of filtering kernel) for generating a spatial adaptive support region around a point. By using the hybrid of the local polynomial model and color/intensity based range guidance, the proposed method not only preserves edges but also does a much better job in preserving spatial variation than existing popular filtering methods. Our method proves to be effective in a number of applications: depth image upsampling, joint image denoising, details enhancement, and image abstraction. Experimental results show that our method produces better results than state-of-the-art methods and it is also computationally efficient.

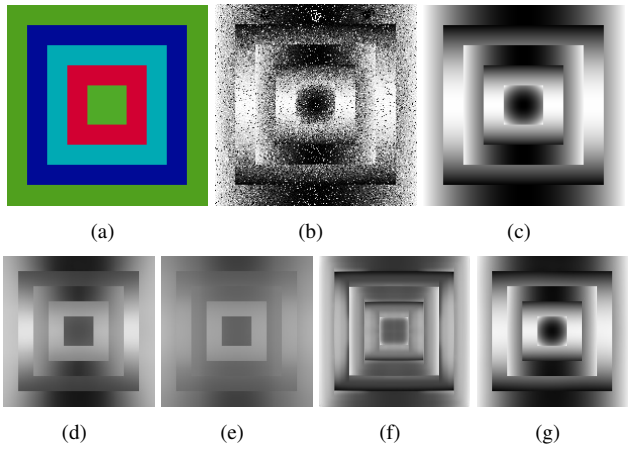


Figure 1. Conventional filtering methods that use range guidance only are prone to smooth the spatial variation (see text for details). All results are obtained with a fixed size of filtering kernel $r = 40$. (a) The guidance image I. (b) The input image P. (c) The reference of (b). (d) The result obtained by BF [1]. (e) The result obtained by GF [2]. (f) The result obtained by CLMF [11]. (g) The result obtained by our filtering method.

1. Introduction

Joint image filtering [1] or guided image filtering [2] has been introduced for image smoothing, structure transformation and other purposes in recent studies. Compared with traditional image filtering process, this operation not only keeps the desired propriety of preserving edges but also considers additional information from a given guidance image. This attribute makes it a powerful method in many computer vision and graphics applications including stereo matching [3, 4], optical flow estimation [4], colorization [5], and saliency detection [6].

A well known joint filtering method is based on the bilateral filter (BF) as proposed in [1]. In this method, the output at a point is a weighted average over the support region around the point. However, this BF based joint image filtering is computationally expensive and therefore many schemes are proposed to accelerate it for fast implementation such as [7, 8]. Another problem of the joint BF is its gradient reversal artifacts as discussed in [9]. Recently, the guided filter (GF) proposed in [2] is shown to be superior to BF for its computational efficiency and the better gradient preserving property. However, the squared window used in the joint BF and GF degrades the performance, since the fixed support region lacks spatial adaptivity which leads to the bleeding artifacts at boundaries in applications such as depth upsampling [10].

Many methods are proposed to achieve spatial adaptivity such as the anisotropic estimator which proved effective for image denoising [12]. Despite of the effectiveness, the arbitrary shape of support regions in these methods makes fast implementation impossible. To enable a fast aggregation over the adaptive support regions, a study in [11] ex-

tended the idea behind the GF and proposed a generalized multipoint framework (named CLMF) which introduces the cross based aggregation method and the order flexibility of local range models. In [11], the support arms are calculated at each point, and the data is then aggregated by using the integral image scheme [13]. CLMF is reported to be superior to BF and GF owing to its sophisticated support region calculation technique which gains the spatial adaptivity of the support region. However, the support region calculation method adopted in CLMF has limitations. First, although the data aggregation can be carried out within $O(1)$ complexity, the complexity of calculation for support arms at each point depends on the maximum length of arms. This will slow down the algorithm particularly for large support arms. Second, the support region depends on the order of aggregation (aggregating horizontally then vertically or vertically then horizontally). This leads to unreliable estimation particularly for points near concave regions. For achieving fast support region calculation and producing reliable filtering results, we propose a new support region generation and aggregation method.

In the filtering process, one would expect that noises are to be suppressed, while the spatial variation which usually contains the image information is to be preserved. However, a common problem existing in all conventional joint/guided filters is that they are prone to the change of spatial variation by smoothing it, despite of being good at suppressing noises. For example, GF and CLMF only consider the range information in the guidance image regardless of the position of the point in the image. This formulation is very likely to oversmooth the input image in regions where the value of the guidance image are similar. This artifact is notably observed for large filter kernels. Let us look at an example as shown in Fig. 1. In this case, the guidance image contains concentric squares of constant colors. The input image to be filtered varies smoothly in each square and is contaminated by noises. This case is frequently found in a number of graphics studies. For example, in depth image denoising, the guidance image or color image may contain an object whose surface is uniformly colored while the depth of the surface may form a curve. Under the assumption that the filtered image is locally linear to the guidance image, the value of the point in the filtered image becomes similar to each other once they have a similar value in the guidance image regardless of the spatial position of the points. Therefore, the range based multipoint filters using this assumption such as GF and CLMF tend to oversmooth the regions corresponding to the homogenous region in the guidance image. However, this formulation is obviously not working for the case as illustrated in Fig. 1. The pointwise filter (e.g., the joint BF) also has this oversmoothing artifact because the weighted summation tends to smooth values over the support region. The oversmoothing artifact becomes much more severe as the size of the filter kernel increases.

We show in this paper that local polynomial approximation [14] (LPA) can be introduced into joint/guided image filtering for better preserving spatial variation. As demonstrated in [12], multipoint filtering methods are usually more robust than pointwise based method. We therefore extend the original pointwise LPA method into a multipoint LPA (MPLA) method for achieving robust filtering. Finally, we show that our filtering process can be carried out efficiently by using the integral image scheme.

2. Algorithm Overview and Notations

Suppose we have a guided image I and an input image P. The proposed filtering method contains the following steps: generating an adaptive support region Ω_p for each point $p \in P$, modeling regression between P and I over each Ω_p, finally aggregating values from the multiple estimates in Ω_p for p and outputting a filtered image Q.

Adaptive support regions are generated by firstly segmenting the guided image into non-overlapping regions. Then a support region around point $p \in P$ is determined by four support arms which extend to the region boundaries or reach the maximum arm size r. A similarly strategy is used in [11] and [15]. We will show in Section 4 an improved scheme for overcoming the shortcoming introduced by the asymmetrical treatment of the horizontal and the vertical line segments [11, 15].

Denote x_p and y_p the image coordinates of point p. For the support region around each point, we carry out data regularization on P according to a model between P and I. The regularization is based on both the range guidance and local polynomial approximation. As the support regions are overlapping with each other, this process will result in multiple estimates q_k ($k \in \Omega_p$) for each point p. The final estimate q to point p is calculated by aggregating or fusing the multiple estimates q_k.

3. Filtering with LPA and Range Guidance

3.1. Multipoint LPA

To overcome the shortcoming of the conventional range based guided filtering methods, we incorporate the local polynomial approximation in regularizing the value of points in P. This process is explained using a 1D signal in Fig. 2. Note that the horizontal coordinate is the spatial position of points rather than the color or intensity values in the guidance image. In Fig. 2, the black lines are signals to be filtered; the red lines denote signals within a support region; and the green lines denote regularized signal by using polynomials with different orders.

For a given point p and its support region Ω_p, we use a 2D polynomial function to regularize the data in the support region at each point k ($k \in \Omega_p$). Denote the

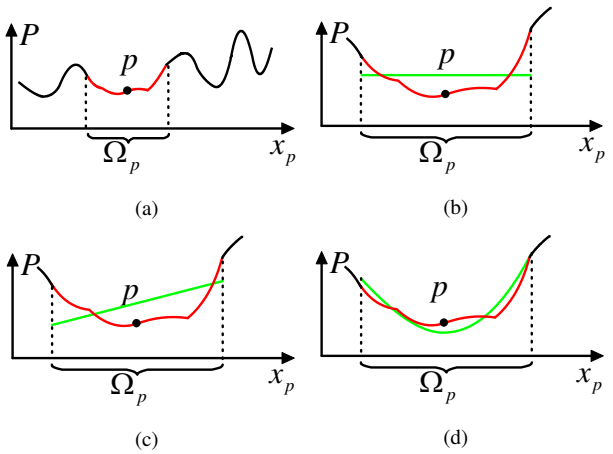


Figure 2. Illustration for local polynomial approximation at a point of interest p in the 1D case. (a) The signal to be filtered P and the support region Ω_p of p. (b) Approximation results by using constant values or zero order polynomials. (c) Approximation results by using first order polynomials.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

obtained m-order polynomial regularization results in the support region of point 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:

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

where x and y are the spatial coordinates. The multi-point local polynomial approximation by using the "m" order polynomial is named as MLPA-m in this paper. Denote the coordinate nomials of point p as 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 point p from regularization results around different k (k: $p \in \Omega_k$):

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

The final estimate for point p is given as a weighted average over all these estimates:

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

where w_k is the weight associated with $\mathcal{P}_k^m(p)$ to be discussed later in Section 5. As suggested in [11], it is advantageous to use an approximation of Eq. (3) as the output of the MLPA for computational efficiency and data access patterns:

$$Q_{\mathcal{P}}(p) = \frac{\sum_{k\in\Omega_p} w_k \mathcal{P}_k^m(p)}{\sum_{k\in\Omega_p} w_k} \tag{4}$$

3.2. MLPA with Range Guidance

To achieve edge-preserving filtering, we incorporate range guidance into the MLPA based on a local linear model (LLM) as used in [2]. The range guidance is the intensity signal for gray scale images or the RGB channels for color images. Suppose that we have obtained the output of LPA for all points: $Q_{\mathcal{P}}$, we use the range information as the guidance to generate a signal $Q_{\mathcal{R}}$ which takes into account the difference between the final output Q and $P - Q_{\mathcal{P}}$. Therefore, Q is the summation of $Q_{\mathcal{P}}$ and $Q_{\mathcal{R}}$ which minimizes the difference between Q and $P - Q_{\mathcal{P}}$ while maintaining the LLM:

$$Q = Q_{\mathcal{P}} + Q_{\mathcal{R}} \tag{5}$$

Similar to the weighted averaging scheme as used in Eq. (4), we calculate $Q_{\mathcal{R}}$ at point p by

$$Q_{\mathcal{R}}(p) = \frac{\sum_{k\in\Omega_p} w_k \left(\boldsymbol{\beta}_k \boldsymbol{I}_p + \gamma_k^1\right)}{\sum_{k\in\Omega_p} w_k} \tag{6}$$

where $\boldsymbol{\beta}_k$ and γ_k^1 are LLM coefficients of the support region around point k; $\boldsymbol{I}_p$ is the intensity value for gray scale images or the RGB vector for color images; the weights w_k are identical to those in Eq. (4). Readers are encouraged to refer to [2, 11, 16] for more details about LLM with range guidance.

Define a hybrid guidance vector of $\mathbf{S}_p^m$ and $\boldsymbol{I}_p$ by $\mathbf{G}_p^m = \begin{bmatrix} \mathbf{S}_p^m \\ \boldsymbol{I}(p) \end{bmatrix}$. Reorganize the coefficients of LPA and LLM and put them into one vector: $\boldsymbol{\theta}_k^m = [\boldsymbol{\alpha}_k^m, \boldsymbol{\beta}_k]$. When using color image as a 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 RGB channels respectively. Substituting Eq. (6) and Eq. (4) into Eq. (5), we obtain:

$$Q(p) = \frac{\sum_{k\in\Omega_p} w_k \left(\boldsymbol{\theta}_k^m \boldsymbol{G}_p^m + \gamma_k\right)}{\sum_{k\in\Omega_p} w_k} \tag{7}$$

where $\gamma_k = \gamma_k^1 + \gamma_k^0$. For estimating model parameters, $\boldsymbol{\theta}_k^m$, we extent the model used in [2, 11] into a weighted linear ridge regression model. Specifically, we minimize the following cost function in the support region of point k:

$$E(\boldsymbol{\theta}_k^m, \gamma_k) = \sum_{s\in\Omega_k} w_s' \Big(\left(\boldsymbol{\theta}_k^m \boldsymbol{G}_s^m + \gamma_k - P(s)\right)^2 + \varepsilon_s \|\boldsymbol{\alpha}_k\|^2 + \varepsilon_r \|\boldsymbol{\beta}_k\|^2 \Big) \tag{8}$$

where ε_s and ε_r are used for regularizing the coefficient of spatial guidance and range guidance respectively, and w_s',

to be discussed in Section 5, is the weight associated with point s. The solution to Eq. (8) is given by

$$\boldsymbol{\theta}_k^m = \left(\boldsymbol{\Sigma}_k\left(\boldsymbol{G}^m, P\right)\right)^{\mathrm{T}} \left(\boldsymbol{\Sigma}_k\left(\boldsymbol{G}^m, \boldsymbol{G}^m\right) + \boldsymbol{\varepsilon}\right)^{-1} \quad (9)$$

$$\gamma_k = \bar{P}_k - \boldsymbol{\theta}_k^m \boldsymbol{\mu}_k \quad (10)$$

Here $\boldsymbol{\mu}_k$ and $\bar{P}_k$ are the weighted mean of $\boldsymbol{G}^m$ and P in Ω_k; $\boldsymbol{\Sigma}_k\left(\boldsymbol{G}^m, \boldsymbol{G}^m\right)$ and $\boldsymbol{\Sigma}_k\left(\boldsymbol{G}^m, P\right)$ are the weighted covariance matrixes of $\boldsymbol{G}^m$ and of $\left(\boldsymbol{G}^m, P\right)$ in Ω_k respectively; $\boldsymbol{\varepsilon}$ is a diagonal matrix whose entries are ε_s or ε_r arranged in an order corresponding to entries in $\boldsymbol{G}^m$. The size of $\boldsymbol{\Sigma}_k$ depends on the order of the LPA and the number of channels in the range guidance. For example, the size of $\boldsymbol{\Sigma}_k$ is 5×5 when a color image (RGB image) is used as guidance and $m = 1$.

4. Adaptive Support Regions Computation

We improve the method as used in [11, 15] to define a support region for each point. This improved method calculates a support region within a constant computational complexity w.r.t. the size of the support region and it is also invariant to the aggregation order of the image integration. In this method, we start with a region segmentation to group points into homogenous regions. Then, for a point p, four support arms $\{h_p^0, h_p^1, h_p^2, h_p^3\}$ are decided by the boundaries of regions and the boundaries of a square window of size $(2r+1) \times (2r+1)$ centered at p. After that the spatial adaptive support region is defined implicitly by carrying out image integration sequentially.

4.1. Region Segmentation

For fast implementation, we employ the graph based segmentation method as proposed in [17]. A short review of this segmentation method in [17] is provided and followed by our improvement. Given the guidance image I, we construct a graph $G\left(V, E\right)$ with the vertices being points in I and edges corresponding to neighboring point pairs in I, then sort the edges into a non-decreasing edge weight order. The segmentation is conducted in a similar manner as Kruskals algorithm for constructing a minimum spanning tree (MST). The algorithm starts at a segmentation where each point represents a component and keeps merging two components until no two components can be merged. During the merging produce, two components C_1 and C_2 are merged if the weights of all connecting edges between them are smaller than their minimum internal difference which is defined as:

$$\begin{aligned} &M \cdot Int\left(C_1, C_2\right) \\ &= \min\left(Int\left(C_1\right) + \tau\left(C_1\right), Int\left(C_2\right) + \tau\left(C_2\right)\right) \end{aligned} \quad (11)$$

where $Int\left(C\right)$ is the internal difference being defined as the largest weight in the MST of the component C, $\tau\left(C\right)$ is a non-negative function of the size of the component C. Unlike the method in [17] where the size of C is directly used for defining $\tau\left(C\right)$, we define it as a function of the ratio between the size of the guided image $|I|$ and the size of the component $|C|$, so that $\tau\left(C\right)$ is robust to the change of the image size. Thus,

$$\tau\left(C\right) = k\frac{|I|}{|C|} \quad (12)$$

where k is a constant parameter. In the original graph based segmentation method [17], the weight of the edge is defined as the intensity difference between two points. However, this measurement does not always produce satisfactory segmentation results near region boundaries. We therefore define the weight as the product between the absolute intensity difference and the exponential of the gradient so that the weight become much lager at region boundaries. The gradient of I at point p is calculated as follows (for color images, we use its corresponding gray scale image to calculate the gradient):

$$\nabla\left(p\right) = \sqrt{\nabla_x^2\left(p\right) + \nabla_y^2\left(p\right)} \quad (13)$$

where ∇_x and ∇_y are the outputs of the Sobel filter on the horizonal and vertical directions respectively. The modified weight on the edge between p and q in the graph G is

$$w\left(p, q\right) = \left|\boldsymbol{I}_p - \boldsymbol{I}_q\right| \exp\left(\max_{i=p,q} \nabla\left(i\right)\right) \quad (14)$$

4.2. Support Arms Calculation

When the region segmentation is obtained, four support arms $\{h_p^0, h_p^1, h_p^2, h_p^3\}$ of a point p are produced accordingly. The support arms take p as the origin and extend to the segment boundaries or until the maximum length r is reached. Unlike the cross based aggregation methods in [11, 15] where the computational complexity is linear to the length of support arms, our new method computes the support arm in a constant computational complexity w.r.t. their lengths. The pseudocode of the algorithm for efficiently calculating the lengths of the horizontal support arms is given in Algorithm 1. Vertical arms are calculated in a similar manner.

Algorithm 1:

Input: the width W and the height H of the guided image I, a region segmentation result S of I, the maximum size of support arms r.

Output: horizontal support arms h^0 and h^2 for all points.

Set $y = 1$ and repeat the following steps until $y = H$:

0. Set the leftmost coordinate to 1, i.e., $x_{\min} := 1$ and set $x := 2$.

1. If $x \neq W$ and $S(x-1, y)=S(x, y)$ where $S(x, y)$ is the segment index of point (x, y), then set $x := x+1$ and repeat Step 1. Otherwise, set the rightmost coordinate to x, i.e., $x_{\max} := x$.

2. For points $p(i, j)$, where $j = y$ and $i = x_{\min}, ..., x_{\max}$, calculate the horizontal arms by $h_p^0 := i - x_{\min}$, and $h_p^2 := x_{\max} - i$. If $x \neq W$, then set $x_{\min} := x$, and $x := x+1$. Go to Step 1.

3. For points $p(i, j)$, where $j = y$ and $i = 1, ..., W$, set horizontal arms to be equal to r if they are larger than r, set the arms to be the shorter length, i.e., $h_p^0 := h_p^2 := \min\left(h_p^0, h_p^2, r\right)$. If $y = H$, then go to Step 4. Otherwise, set $y := y+1$ and go to Step 0.

4. Output the horizontal arms for all points.

In Step 3, we follow the idea in [11] to enforce balanced arms in order to prevent the gradient reversal artifacts. It is useful in some applications such as detail enhancement and joint denoising. However, we found that they degrade the performance in some other applications such as depth image enhancement where we expect a sharp depth jump at boundaries. Therefore we do not use balanced arm for these applications.

4.3. Aggregation Order Invariant Support Region

In the cross based aggregation methods [11, 15], the order of performing horizontal or vertical aggregation determines the support regions. Support regions can be quite different particularly for points near the concave regions. As shown in Fig. 3, when the aggregation is performed first in the horizontal direction then the vertical direction, the data from points colored green is fused; however, when the aggregation is performed vertically then horizontally, the points colored in blue are aggregated. Denote point p's support region when aggregating horizontal data first by Ω_p^H, and denote its support region when aggregating vertical data first by Ω_p^V. In our new formulation, the support region is the union of these two: $\Omega_p = \Omega_p^H \cup \Omega_p^V$. The support region in the original cross based aggregation methods [11, 15] is Ω_p^H.

5. Implementation Issues and Running Time

Similar to [2], our algorithm requires a weighted summation of range data $\boldsymbol{I}_p$ and coordinate nomials $\mathbf{S}_p^m$ over Ω_p as given in Eq. (7), Eq. (9), and Eq. (10). However, there is no general method to achieve this for arbitrary weights within a constant computation complexity. Fortunately, a group of exclusive weights makes the constant computation complexity possible. Define weights in Eq. (8) for calculating

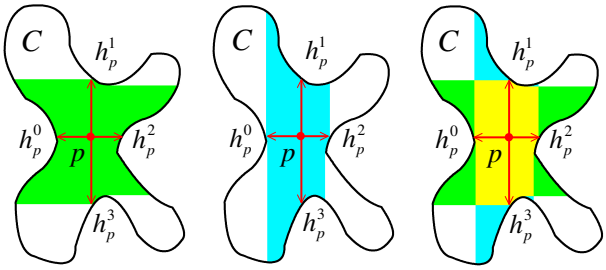


Figure 3. Support regions of a point p in the region component C. (a) Support region Ω_p^H used in the original cross based aggregation method, which is defined implicitly by performing horizontal aggregation first. (b) Ω_p^V is the support region when performing vertical aggregation first. (c) Order invariant support region: $\Omega_p = \Omega_p^H \cup \Omega_p^V$ where the region colored in yellow is $\Omega_p^H \cap \Omega_p^V$.

Eq. (9), and Eq. (10) by

$$w'_s = \begin{cases} 1 & \text{if } s \in \left(\Omega_p^H - \Omega_p^V\right) \cup \left(\Omega_p^V - \Omega_p^H\right) \\ 2 & \text{if } s \in \Omega_p^H \cap \Omega_p^V \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

where the notation of point k in Eq. (8) is changed into p. The weighted summation over Ω_p with the weights defined in Eq. (15) can be computed by adding the summation over Ω_p^V and Ω_p^H respectively:

$$\begin{aligned} \sum_{s \in \Omega_p} w'_s P(s) &= 2 \sum_{s \in \Omega_p^H \cap \Omega_p^V} P(s) + \sum_{s \in \Omega_p^H - \Omega_p^V} P(s) + \sum_{s \in \Omega_p^V - \Omega_p^H} P(s) \\ &= \sum_{s \in \Omega_p^H} P(s) + \sum_{s \in \Omega_p^V} P(s) \end{aligned} \tag{16}$$

As presented in [15], the summation of data over Ω_p^H can be efficiently computed with a complexity of $O(N)$ for N points by using integral image technique [13] which is carried out sequentially along horizonal and vertical directions. The summation of data over Ω_p^V can also be computed in a similar way by switching the order of performing image integration. Weights in Eq. (7) are similarly defined as

$$w_k = \begin{cases} |\Omega_k| & \text{if } k \in \left(\Omega_p^H - \Omega_p^V\right) \cup \left(\Omega_p^V - \Omega_p^H\right) \\ 2|\Omega_k| & \text{if } k \in \Omega_p^H \cap \Omega_p^V \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

Here we use the number of sampling points to encourage a stable estimate as suggested in [11]. Another reason of defining weights as given in Eq. (15) and Eq. (17) is that points in $\Omega_p^H \cap \Omega_p^V$ are much closer to p than those in $\Omega_p^H - \Omega_p^V$ or $\Omega_p^V - \Omega_p^H$ (see Fig. 3). Therefore, points in $\Omega_p^H \cap \Omega_p^V$ are assigned to higher weights than points in $\Omega_p^H - \Omega_p^V$ and points in $\Omega_p^V - \Omega_p^H$.

The computational cost of our algorithm consists of two parts, one is the cost for the support region calculation, the

Table 1. Comparison of running time on color image filtering.

Methods	Time (ms) $r = 9$	Time (ms) $r = 100$
Guided Image Filtering [2]	820	820
CLMF-0 [11]	490	1560
CLMF-1 [11]	910	2150
MLPA-0	1100	1100
MLPA-1	2350	2350
MLPA-2	5250	5250

other is the cost for data aggregation in the filtering process. As discussed above, the complexity for filtering is $O(N)$ for an image with N points. Algorithm 1 takes $8 \times N$ operations (comparison or +/- operations) for generating support arms. Unlike the adaptive support region calculation method in [11] whose complexity of calculating support arms is linear to r, complexity of our method is constant w.r.t. windows size r due to the efficient support arms calculation method as described in Algorithm 1. Although the $O(N \log N)$ complexity of MST based segmentation is higher than $O(N)$ complexity of the other building blocks of our method, it does not degrade our filter with respect to running time. The reason is: similarly to GF, the running time of our filter is mainly dominated by inverting a symmetric matrix whose order is depended on the order of LPA and the form of I. For example, when the order of LPA is 1 and I is a gray scale image, we need to invert a 3×3 matrix. This process is computed in about 30 operations, and $\log N$ is usually smaller than 30 in most cases. The algorithm is implemented on a 32bits PC with a 3.0 GHz CPU and 4GB RAM with C++ without using any SIMD instructions. The comparison of running time for filtering 1000000 points with color guidance is given in Table 1. In MLPA-m, we invert a 3×3 matrix which is similar to that of GF and therefore the running time is close to GF. In MLPA-1 and MLPA-2, the matrix is increased to 5×5 and 8×8 respectively and causes the increase of the running time. The running time of the MLPA-m method with the order m higher than 2 is not tested in our experiments; but it can be predicted according to the complexity of inverting a matrix w.r.t. the matrix size. The running time of computing adaptive support regions is about 150 ms.

6. Applications and Experiments

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

6.1. Depth Image Enhancement

Depth image enhancement includes the depth image upsampling and depth image denoising. The enhancement can be carried out by adopting joint filtering methods. The quantitative comparison for the results from different filters,

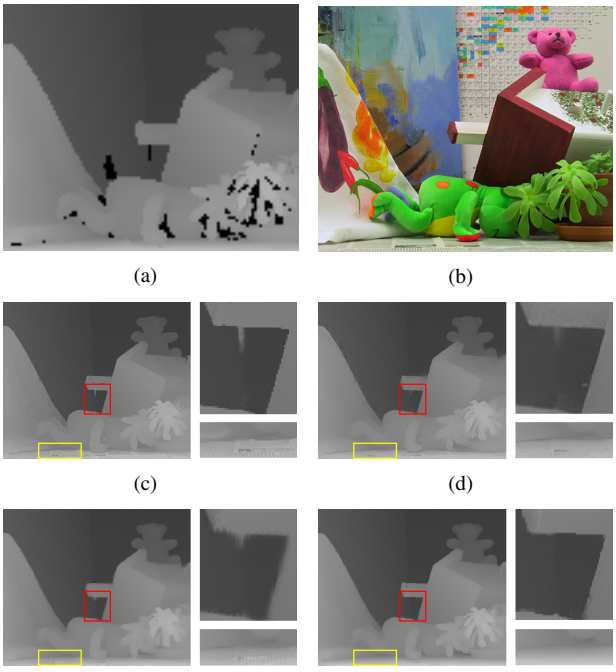


Figure 4. Depth upsampling results with $r = 9$. (a) and (b) A low resolution depth image and its high resolution color image. (c) The results obtained by BF ($\sigma_s = 30, \sigma_r = 20/255$). (d) The results obtained by GF ($\varepsilon = 0.05^2$). (e) The results obtained by CLMF ($\varepsilon = 0.05^2, \tau = 20/255$). (f) The results by MLPA-1 filter ($\varepsilon_r = 0.05^2, \varepsilon_s = 0.001^2, k = 0.005/255$). These parameters are kept constants for all datasets in Table 2.

BF, GF, CLMF, and MLPA-1 is shown in Table 2 where the error threshold is 1 pixel (see [18] for details). To demonstrate the advantage of using the order invariant support region, we perform MPLA-1 with fixed parameters on points in Ω_p^H. These results are listed as MPLA-1(Ω_p^H). In the upsampling experiments, the upsampling factor is 4 and the missing data in the depth image is colored in black. In the denoising experiments, a white noise with $\sigma = 10$ is added to all points in the depth image. Compared with BF, GF, and CLMF, our MPLA-1 provides a much more accurate high resolution depth image particularly for preserving slant and curved surface and suppressing the bleeding artifacts [10]. Fig. 4 and Fig. 5 show the visual comparison between the results from different filters where the parameters are carefully tuned so that best results are presented.

Table 2. Error percentage for depth image upsampling among results from different filters on Middlebury datasets [18].

Datasets	BF	GC	CMLF	MLPA-1	MLPA-1(Ω_p^H)
Tsukuba	5.35	7.03	5.67	**4.21**	4.52
Venus	1.29	1.66	1.19	**0.72**	0.77
Teddy	7.21	8.92	6.55	**3.06**	3.28
Cones	7.59	10.4	7.82	**5.27**	5.46

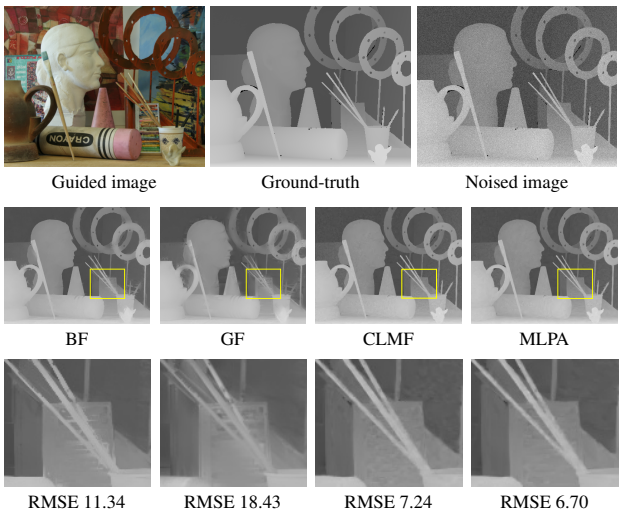


Figure 5. Depth denoising results with $r = 9$. Parameters are set to those in upsampling experiments except $\tau = 5/255$ in CLMF and $k = 0.001/255$ in MLPA for the bleeding artifacts suppression purpose.

6.2. Flash/no-flash Denoising

In this application, a no-flash image is to be denoised based on the guidance from a given flash version of the same scene as described in [1]. Thanks to the adaptive support region and local polynomial approximation schemes, our filter not only preserves the spatial variation around the shadow regions and the light reflecting regions very well but also takes the advantage of large filter kernels to suppress noises as much as possible in smooth regions such as the surface of the sofa in Fig. 6.

6.3. Other Applications

Our MLPA filter can also be used for other graphics or computer vision applications, such as detail enhancement and image abstraction [19]. Similarly to GF [2], MLPA avoids the gradient reversal artifacts which may appear in detail enhancement. Compared with GF, our MLPA filter better preserves spatial variation of the original image in the base layer; however the spatial variation in the filtering results from GF is smoothed. It means that the spatial variation will be less boosted in our method compared with the results from GF. Therefore, the gradient of plant leaves in Fig. 7 is less enhanced in our results. The image abstraction results provided by our MLPA-1 are given in Fig. 8.

6.4. Parameters Effects

In our MLPA, the parameter k controls the spatial adaptivity. The larger the k becomes, the more neighboring points will be considered for the model regression. Generally, a large k value will weaken the spatial adaptivity and lead to undesirable results such as the bleeding artifact in depth denoising/upsampling. The spatial adaptivity will be totally lost when setting k to an extremely large value. In this case, our filtering will perform on squared windows (see Fig. 6 (e)). For the purpose of better preserving spatial variation, parameter ε_s is set to a small value or to 0. The larger the ε_s is, the more smoothing effect spatial variation will suffer.

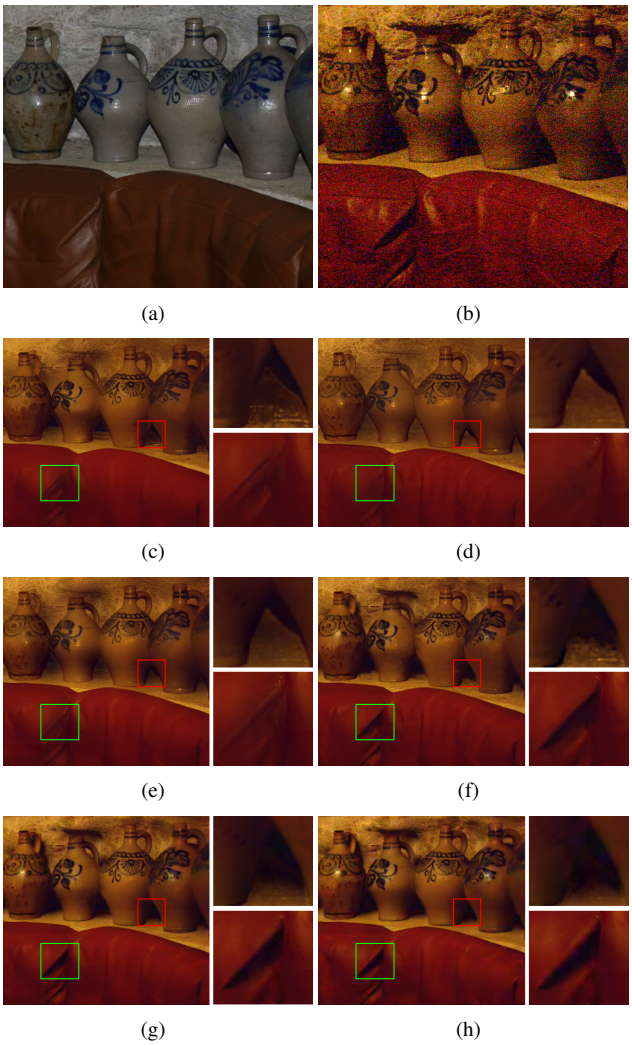
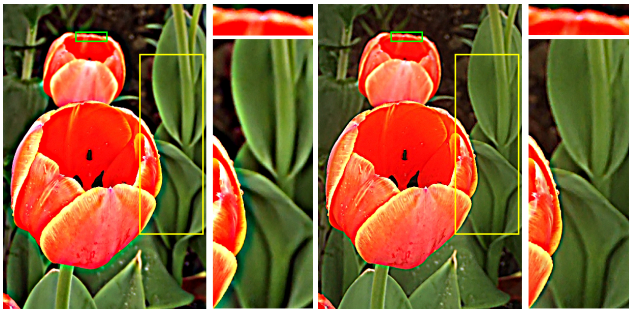


Figure 6. Result of the flash/no flash denoising. (a) and (b) No flash image with noises as filtering inputs and the flash image as guidance. (c) and (d) The denoised images by using the joint bilateral filter ($\sigma_s = 30, \sigma_r = 20/255$) and the guided filter ($\varepsilon = 0.02^2$) respectively. (e) The result obtained by using MLPA-1 filter with square windows. (f), (g) and (h) Results obtained by MLPA-0, MLPA-1, and MLPA-2 respectively using adaptive support regions with parameters: $\varepsilon_r = 0.02^2, \varepsilon_s = 0, k = 0.007/255$.

7. Conclusions

In this paper, we developed an image filtering method which incorporates the LPA with range guidance into a mul-



(a) (b)

Figure 7. Results of detail enhancement ($r = 16$, details are boosted by 5 times). (a) The result by guided filter ($\varepsilon = 1.0$). (b) The result by MLPA-1 ($\varepsilon_r = 1.0, \varepsilon_s = 0.005^2, k = 0.1/255$).

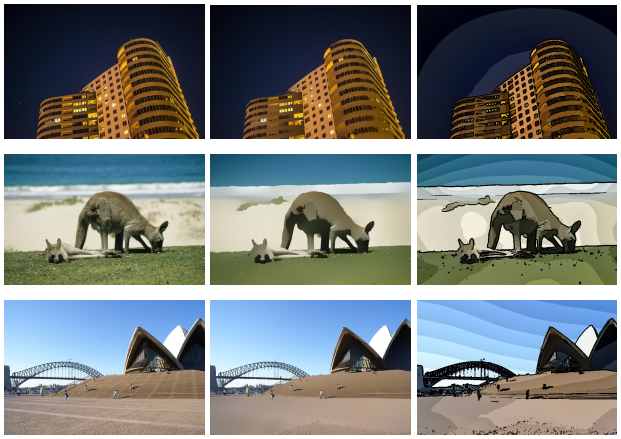


Figure 8. Results of image abstraction using MLPA-1 with constant parameters. First column is the original image. Middle column is the filtered image. Last column is the abstracted image after quantization with edges.

tipoint estimation framework for the first time. The proposed hybrid guided image filtering method has the desired property of better persevering the spatial variation in the input image. We proposed an efficient method for computing the adaptive support regions in constant computational complexity and developed a new scheme for the symmetrical treatment on the horizontal and vertical line segments. Experiments on a number of applications verify the effectiveness of the proposed method and show its unique property over the existing guided/joint image filtering methods. The comparisons with the state-of-the-art methods indicate that our MLPA filter is very competitive. Our further work will focus on developing new local models for reducing the computational cost when using high order polynomial 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 Redmond for providing testing images.

References

[1] G. Petschnigg, R. Szeliski, M. Agrawala, M. Cohen, H. Hoppe, and K. Toyama, "Digital photography with flash and no-flash image pairs," *ACM TOG*, vol. 23, no. 3, pp. 664–672, 2004.

[2] K. He, J. Sun, and X. Tang, "Guided image filtering," in *ECCV*, 2010, pp. 1–14.

[3] L. De-Maeztu, S. Mattoccia, A. Villanueva, and R. Cabeza, "Linear stereo matching," in *ICCV*, 2011, pp. 1708–1715.

[4] C. Rhemann, A. Hosni, M. Bleyer, C. Rother, and M. Gelautz, "Fast cost-volume filtering for visual correspondence and beyond," in *CVPR*, 2011, pp. 3017–3024.

[5] A. Levin, D. Lischinski, and Y. Weiss, "Colorization using optimization," *ACM TOG*, vol. 23, no. 3, pp. 689–694, 2004.

[6] Y. Ding, J. Xiao, and J. Yu, "Importance filtering for image retargeting," in *CVPR*, 2011, pp. 89–96.

[7] Q. Yang, K.-H. Tan, and N. Ahuja, "Real-time O(1) bilateral filtering," in *CVPR*, 2009, pp. 557–564.

[8] F. Porikli, "Constant time O (1) bilateral filtering," in *CVPR*, 2008, pp. 1–8.

[9] Z. Farbman, R. Fattal, D. Lischinski, and R. Szeliski, "Edge-preserving decompositions for multi-scale tone and detail manipulation," *ACM TOG*, vol. 27, no. 3, pp. 1–10, 2008.

[10] J. Park, H. Kim, Y.-W. Tai, M. S. Brown, and I. Kweon, "High quality depth map upsampling for 3D-TOF cameras," in *ICCV*, 2011, pp. 1623–1630.

[11] J. Lu, K. Shi, D. Min, L. Lin, and M. N. Do, "Cross-based local multipoint filtering," in *CVPR*, 2012, pp. 430–437.

[12] V. Katkovnik, A. Foi, K. Egiazarian, and J. Astola, "From local kernel to nonlocal multiple-model image denoising," *IJCV*, vol. 86, no. 1, pp. 1–32, 2010.

[13] F. C. Crow, "Summed-area tables for texture mapping," *ACM TOG*, vol. 18, no. 3, pp. 207–212, 1984.

[14] J. Fan, *Local Polynomial Modelling and its Applications*. CRC Press, 1996, vol. 66.

[15] K. Zhang, J. Lu, and G. Lafruit, "Cross-based local stereo matching using orthogonal integral images," *IEEE TCSVT*, vol. 19, no. 7, pp. 1073–1079, 2009.

[16] A. Levin, D. Lischinski, and Y. Weiss, "A closed-form solution to natural image matting," *IEEE TPAMI*, vol. 30, no. 2, pp. 228–242, 2008.

[17] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *IJCV*, vol. 59, no. 2, pp. 167–181, 2004.

[18] http://vision.middlebury.edu/stereo/, 2013.

[19] H. Winnemöller, S. C. Olsen, and B. Gooch, "Real-time video abstraction," *ACM TOG*, vol. 25, no. 3, pp. 1221–1226, 2006.