Fast Global Image Smoothing Based on Weighted Least Squares

Dongbo Min, *Member, IEEE*, Sunghwan Choi, *Student Member, IEEE*, Jiangbo Lu, *Member, IEEE*, Bumsub Ham, *Member, IEEE*, Kwanghoon Sohn, *Senior Member, IEEE*, and Minh N. Do, *Fellow, IEEE*

***Abstract*—This paper presents an efficient technique for performing spatially inhomogeneous edge-preserving image smoothing, called *fast global smoother*. Focusing on sparse Laplacian matrices consisting of a data term and a prior term (typically defined using four or eight neighbors for 2D image), our approach efficiently solves such global objective functions. Specifically, we approximate the solution of the memory- and computation-intensive large linear system, defined over a d-dimensional spatial domain, by solving a sequence of 1D sub-systems. Our separable implementation enables applying a linear-time tridiagonal matrix algorithm to solve d three-point Laplacian matrices iteratively. Our approach combines the best of two paradigms, i.e., efficient edge-preserving filters and optimization-based smoothing. Our method has a comparable runtime to the fast edge-preserving filters, but its global optimization formulation overcomes many limitations of the local filtering approaches. Our method also achieves high-quality results as the state-of-the-art optimization-based techniques, but runs about 10 to 30 times faster. Besides, considering the flexibility in defining an objective function, we further propose generalized fast algorithms that perform L_γ norm smoothing ($0 < \gamma < 2$) and support an aggregated (robust) data term for handling imprecise data constraints. We demonstrate the effectiveness and efficiency of our techniques in a range of image processing and computer graphics applications.**

***Index Terms*—Edge-preserving smoothing (EPS), weighted least squares (WLS), fast global smoother (FGS), iterative reweighted least squares (IRLS), aggregated data constraint, imprecise input.**

I. Introduction

Many applications in image processing and computer graphics often require decomposing an image into a piecewise smooth base layer and a detail layer. The base layer captures the main structural information, while the detail layer contains the residual smaller scale details in the image. These layered signals can be manipulated and/or recombined in various ways to match different application goals. Over the last decades, several edge-preserving smoothing (EPS) methods have been proposed.

At a high level, EPS methods can be classified into two groups. The first group consists of the edge-preserving (EP) filters that explicitly compute a filtering output as a weighted average, sometimes in an iterative way. The early work in this class includes the anisotropic diffusion [1] and the bilateral filter [2]. Recent intensive efforts have led to several efficient techniques [3]–[7] to accelerate the bilateral filter, and also quite a few fast filtering approaches based on different theories and computational models [8]–[12]. Though with varying smoothing performance and application constraints, these EP filters are typically efficient, often giving a linear-time complexity dependent on the number of image pixels only. However, as will be elaborated later, a common limitation of these essentially *local* filters is that they cannot completely resolve the ambiguity regarding whether or not to smooth certain edges. In addition, most of them are not directly applicable to several advanced image editing tasks.

The second class of existing EPS methods are based on *global* optimization formulations [13]–[17]. They seek to find a globally optimal solution to an objective function usually involving a data constraint term and a prior smoothness term. Thanks to solving such objective functions globally in a principled manner, the optimization-based approaches often achieve the state-of-the-art results in a range of image processing and computer graphics tasks, overcoming the limitation (e.g. halo artifacts) of the explicit EP filters. However, this outperformance is achieved at a much increased price of computational cost, mainly arising from solving a large linear system. Even with the recent endeavor in developing acceleration techniques [18], [19], the optimization-based methods are typically still an order of magnitude slower than the efficient EP filters.

In this paper, we present a fast technique that performs spatially inhomogeneous edge-preserving smoothing, called *fast global smoother* (FGS). Similar to [14], [15], our approach aims to optimize a global objective function defined with a data constraint and a smoothness prior, but we propose an efficient alternative to the previous time-consuming large linear system solvers [19]. Specifically, we approximate the solution of an original linear system with an inhomogeneous Laplacian matrix, defined over a d-dimensional spatial domain, by solving a sequence of 1D global optimization-based linear sub-systems. Such a decomposition scheme allows one to leverage a highly efficient tridiagonal matrix algorithm [20] in a cascaded fashion iteratively. As a result, our algorithm has a runtime complexity linear to the number of image pixels only.

D. Min and J. Lu are with the Advanced Digital Sciences Center, Singapore (e-mail: dongbo@adsc.com.sg; jiangbo.lu@adsc.com.sg). (*Corresponding author: J. Lu.*)

S. Choi and K. Sohn are with the School of Electrical and Electronic Engineering, Yonsei University, Seoul, South Korea (e-mail: shch@yonsei.ac.kr; khsohn@yonsei.ac.kr).

B. Ham is with WILLOW team at INRIA (e-mail: bumsub.ham@inria.fr).

M. N. Do is with the University of Illinois at Urbana-Champaign, Urbana, Illinois, USA (e-mail: minhdo@illinois.edu).

This work was supported by a research grant from the Human Sixth Sense Program (HSSP), Advanced Digital Sciences Center from Singapores Agency for Science, Technology, and Research (A*STAR).

Table I compares our method with the state-of-the-art EPS approaches, regarding smoothing properties. We chose two EP filters, the guided filter (GF) [8] and the domain transform (DT) [9], and one optimization-based smoothing technique, the weighted least squares (WLS) method [15]. We measure the runtime for filtering a 1M pixel RGB image on a single CPU core. In DT, the recursive filter (RF) is used, which is the fastest local filter. Our method has a comparable runtime to the efficient EP filters, but the optimization formulation of our approach overcomes the limitations of previous filters in terms of smoothing quality, i.e. producing no halos (see Figure 1). It achieves high-quality results as the state-of-the-art optimization-based techniques, but runs about 30 times faster. More specifically, our method employs only a small (fixed) number of arithmetic operations (5 multiplications and 1 division) at every pixel for 1D signal smoothing. Besides, considering the flexibility in defining an objective function, we further propose generalized fast algorithms that perform L_γ norm smoothing and support an aggregated (robust) data term for handling imprecise data constraints. Note that such capabilities are unattainable for local EP filters.

We demonstrate the effectiveness and efficiency of our techniques in a range of applications, where the sparse Laplacian matrix is defined e.g. using a four- or eight-neighbor smoothness term on a 2D image, including image smoothing, multi-scale detail manipulation [15], structure extraction from texture [17], sparse data interpolation [13], imprecise edit propagation [21], and depth upsampling [22].

The main contributions of this work are summarized in the following.

- We present a fast $O(N)$ edge-preserving image smoothing method, where N represents an image size, by approximating the solution of an original linear system with a series of 1D global linear sub-systems.
- We propose new computational tools for efficiently performing L_γ norm smoothing and supporting an aggregated (robust) data term, which are not feasible in existing filtering approaches.

TABLE I
COMPARISON WITH OTHER APPROACHES; TWO LOCAL FILTERS, THE GUIDED FILTER (GF) [8] AND THE DOMAIN TRANSFORM (DT) [9], AND ONE OPTIMIZATION BASED SMOOTHING APPROACH, THE WEIGHTED LEAST SQUARES (WLS) METHOD [15]. FOR MORE DETAILS, PLEASE REFER TO THE TEXT IN SECTION I. (N.A.: NOT AVAILABLE)

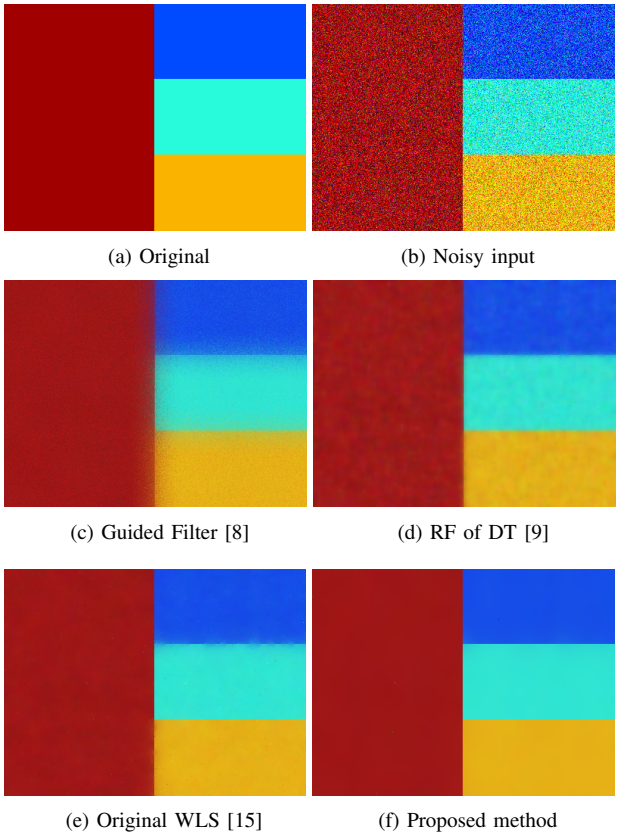
Properties	**GF** [8]	**DT** [9]	**WLS** [15]	**Ours**
Runtime efficiency [Sec. V.C]	0.15s	0.05s	3.3s	0.10s
Smoothing quality [Sec. V.A]	halo	halo	no halo	no halo
L_γ norm smoothing $(0 < \gamma < 2)$ [Sec. IV.D]	N.A.	N.A	N.A.	Yes
Using aggregated data [Sec. IV.E]	N.A.	N.A.	N.A.	Yes

II. RELATED WORK

Among many edge-preserving filters, the bilateral filter (BF) [2] is the most popular one. The BF computes a filtered output with a weighted average of neighboring pixels, by taking into account spatial and range (color) distances in the bilateral kernel. It is, however, computationally expensive due to its nonlinearity, when a large kernel is employed. Recently, several methods have been proposed for accelerating the BF [3]–[6]. These methods have a linear-time complexity with the image size only, called $O(N)$ filters where N is the number of image pixels, but typically compromise the smoothing quality by using quantization or downsampling on a bilateral grid. Adams *et al.* proposed a fast BF method on a permutohedral lattice [7], but it is still an order of magnitude slower than other fast algorithms due to a complicate data access pattern. Fattal [12] introduced a new type of fast filter based on edge-avoiding wavelets (EAW). This multiscale framework enables a very fast computation, but constrains the size of the filtering kernel to powers of two. More recently, several remarkable $O(N)$ edge-preserving filters have been proposed, e.g. with a local linear model [8], [11], a domain transform [9], or a recursive data propagation [10], [23]. Many researchers have demonstrated the effectiveness of these methods in such applications as detail/tone manipulation, high dynamic range (HDR) compression, colorization, and interactive segmentation.

As an alternative approach of image smoothing, Farbman *et al.* proposed to perform the edge-preserving smoothing using the WLS framework [15], consisting of a data term and a prior term that is based on a weighted L_2 norm. The filtered output is obtained by solving a large linear system with a sparse Laplacian matrix, representing an affinity function defined by the given input image. It was shown that this optimization-based approach achieves an excellent smoothing quality with no halo artifact, which commonly appears even in the state-of-the-art local filters [9], [25]. It is worth noting that despite their weak smoothing quality, BF-like local filters [2], [26] can be more desirable in some applications, e.g. recoloring between disjoint elements. The WLS-based optimization has also been applied into image colorization [13] and tone mapping [14] using sparse user scribbles on a grayscale image. Unlike the local filtering approaches [2], [8], [9], the optimization-based methods are directly applicable to many tasks beyond image smoothing, which can be modeled as a linear system with an inhomogeneous Laplacian matrix, including structure extraction [17], gradient domain processing [16], and Poisson blending [27]. For instance, Bhat *et al.* presented a generalized optimization framework for exploring gradient domain solutions in various image and video processing tasks [16]. Xu *et al.* formulated a structure-texture decomposition problem with a relative total variation measure and solved a series of linear equation systems [17].

These optimization-based methods usually obtain the results using sparse matrix solvers which have been considerably advanced using multi-level and multigrid preconditioning [18], [19]. Despite recent significant progress, in case of image smoothing, all the existing solvers are still an order of magnitude slower than the state-of-the-art local filters [8], [9]. Please refer to [19] for detailed analysis of the runtime efficiency and



(a) Original (b) Noisy input

(c) Guided Filter [8] (d) RF of DT [9]

(e) Original WLS [15] (f) Proposed method

Fig. 1. Quality comparison of image smoothing results: (a) Noise-free image (a typical test image used in [15]), (b) Input with Gaussian noise, (c) Guided Filter (GF) [8] with $\epsilon = 0.5^2$ and a radius $r = 18$, (d) Recursive Filter (RF) of Domain Transform (DT) [9] with $\sigma_r = 2.5$ and $\sigma_s = 30.0$, (e) Original WLS [15] with $\sigma_c = 0.12$ and $\lambda = 30.0^2$, and (f) Proposed method with $\sigma_c = 0.1$ and $\lambda = 30.0^2$. Our **global** smoother outperforms the two state-of-the-art **local** methods (GF and DT), especially around edges (best viewed in an electric version). The results of (c)(d)(e) were obtained by the author-provided software (implemented in MATLAB). Note that for true color filtering, all the results were obtained by considering RGB channels simultaneously. Our runtime efficiency is very comparable to that of (c)(d) the state-of-the-art fast local filtering methods and also much faster (about **30**×) than that of (e), although our method is based on a global solver. For an objective evaluation, we measured the structural similarity (SSIM) [24] between the results (c)(d)(e)(f) and the noise-free image (a): (b) 0.0385, (c) 0.3391, (d) 0.5259, (e) 0.5357, and (f) 0.5660. Please refer to Section V for more performance analysis.

accuracy of these solvers.

III. Background

Low-level vision problems are often formulated as minimizing an energy function comprising the data and prior terms, and their solutions are obtained by solving a linear system either once or iteratively according to the norm used to define the energy function. We start with a basic formulation for the edge-preserving image smoothing task which serves as a simple example to provide some intuition.

In image smoothing, given an input image f and a guidance image g, a desired output u is obtained by minimizing the following weighted least squares (WLS) energy function [14], [15], [28]:

$$J(u) = \sum_p \left((u_p - f_p)^2 + \lambda \sum_{q \in \mathcal{N}(p)} w_{p,q}(g)(u_p - u_q)^2 \right). \quad (1)$$

Let f_p and g_p: $\Omega_D \rightarrow \mathbb{R}^3$ (or $\mathbb{R}$) denote a function with a discrete spatial domain $\Omega_D = \{p = (x, y) | 0 \leq x < W, 0 \leq y < H\} \subset \mathbb{R}^2$. f_p and g_p represent a color (or grayscale) value of (x, y). g can be defined with the input image f or another guidance signal aligned to the input according to applications (for image filtering, $g = f$). Here, we define $S = H \times W$ as an image size. $\mathcal{N}(p)$ is a set of neighbors (typically, 4) of a pixel p, and λ controls the balance between the two terms. Increasing λ results in more smoothing in the output u. The smoothness constraint is adaptively enforced using the spatially varying weighting function $w_{p,q}(g)$ defined with g. $w_{p,q}$ represents a similarity between two pixels p and q, and for example can be defined as follows:

$$w_{p,q}(g) = \exp(- \parallel g_p - g_q \parallel / \sigma_c), \quad (2)$$

where σ_c is a range parameter.

By setting the gradient of $J(u)$ defined as in (1) to zero, the minimizer u is obtained by solving a linear system based on a large sparse matrix:

$$(\mathbf{I} + \lambda \mathbf{A})\mathbf{u} = \mathbf{f}, \quad (3)$$

where $\mathbf{u}$ and $\mathbf{f}$ denote $S \times 1$ column vectors containing color values of u and f, respectively. $\mathbf{I}$ denotes an identity matrix. A spatially-varying Laplacian matrix $\mathbf{A}$ with a size of $S \times S$ is defined in a way similar to the random walk approach [29] as follows:

$$\mathbf{A}(m, n) = \begin{cases} \sum_{l \in \mathcal{N}(m)} w_{m,l}(g) & n = m \\ -w_{m,n}(g) & n \in \mathcal{N}(m) \\ 0 & \text{otherwise} \end{cases}, \quad (4)$$

where m and n represent a scalar index corresponding to a pixel p, i.e., $m, n \in \{0, \cdots, S-1\}$. Here, $\mathbf{A}$ is a five-point sparse matrix including diagonal elements. The final smoothing result $\mathbf{u}$ is then written as follows:

$$\mathbf{u}(m) = ((\mathbf{I} + \lambda \mathbf{A})^{-1} \mathbf{f})(m). \quad (5)$$

Figure 1 compares image smoothing results of the local filtering methods, the WLS method, and our global smoother. As expected, the WLS method [15] outperforms the state-of-the-art edge-aware *local* filtering algorithms such as the DT [9] and the GF [8]. Our fast *global* smoother also achieves a much better smoothing quality than the *local* methods, with a comparable $O(N)$ complexity to these local filters. Our method will be detailed in the next section.

IV. Method

In this section, we first present an efficient alternative of seeking the solution of an objective function defined on weighted L_2 norm (1) by decomposing it into each spatial dimension and solving the matrix with a sequence of 1D fast solvers. Then, this approach is extended into more general cases, by solving objective functions defined on weighted L_γ norm ($0 < \gamma < 2$) or using an aggregated data term, which cannot be achievable in the existing EP filters. The flexibility and efficiency of our approach enable a significant acceleration of a range of applications, which typically require solving a large linear system.

A. 1D Fast Global Smoother

First, we consider the WLS energy function defined with a 1D horizontal input signal f^h and a 1D guide signal g^h along the x dimension ($x = 0, \cdots, W-1$). The WLS energy function for the 1D signal becomes:

$$\sum_x \left((u_x^h - f_x^h)^2 + \lambda_t \sum_{i \in \mathcal{N}_h(x)} w_{x,i}(g^h)(u_x^h - u_i^h)^2 \right), \quad (6)$$

where $\mathcal{N}_h(x)$ is a set of two neighbors for x (i.e., $x-1$ and $x+1$). The 1D smoothing parameter λ_t is defined to distinguish it with that of original WLS formulation (3), and will be detailed in Section IV-B. The 1D output solution u^h that minimizes the above equation is written as a linear system:

$$(\mathbf{I}_h + \lambda_t \mathbf{A}_h)\mathbf{u}_h = \mathbf{f}_h. \quad (7)$$

$\mathbf{I}_h$ is an identity matrix with a size of $W \times W$. $\mathbf{u}_h$ and $\mathbf{f}_h$ represent the vector notations of u^h and f^h, respectively. $\mathbf{A}_h$ is a three-point Laplacian matrix with a size of $W \times W$. The linear system in (7) can be written as follows:

$$\begin{bmatrix} b_0 & c_0 & 0 & \cdots & 0 \\ \ddots & \ddots & \ddots & 0 & 0 \\ 0 & a_x & b_x & c_x & 0 \\ 0 & 0 & \ddots & \ddots & \ddots \\ 0 & \cdots & 0 & a_{W-1} & b_{W-1} \end{bmatrix} \begin{bmatrix} u_0^h \\ \vdots \\ u_x^h \\ \vdots \\ u_{W-1}^h \end{bmatrix} = \begin{bmatrix} f_0^h \\ \vdots \\ f_x^h \\ \vdots \\ f_{W-1}^h \end{bmatrix}$$

with boundary conditions $a_0 = 0$ and $c_{W-1} = 0$. Here, u_x^h and f_x^h are the x^{th} elements of $\mathbf{u}_h$ and $\mathbf{f}_h$. a_x, b_x, and c_x represent three nonzero elements in the x^{th} row of $(\mathbf{I}_h + \lambda_t \mathbf{A}_h)$, which can be written as

$$\begin{aligned} a_x &= \lambda_t \mathbf{A}_h(x, x-1) = -\lambda_t w_{x,x-1}, \\ b_x &= 1 + \lambda_t \mathbf{A}_h(x, x) = 1 + \lambda_t (w_{x,x-1} + w_{x,x+1}), \\ c_x &= \lambda_t \mathbf{A}_h(x, x+1) = -\lambda_t w_{x,x+1}. \end{aligned}$$

In fact, solving (7) becomes much easier than solving (3), as the three-point Laplacian matrix $\mathbf{A}_h$ becomes a tridiagonal matrix, whose nonzero elements exist only in the diagonal, the left and right diagonals. Such a matrix has an exact (non-approximate) solution obtained using the Gaussian elimination algorithm with a $O(N)$ complexity (here $N = W$) [20]. Our solution for (7) is an exact minimum of the given energy function (6) defined on the 1D dimension, and more importantly the 1D solver is very fast.

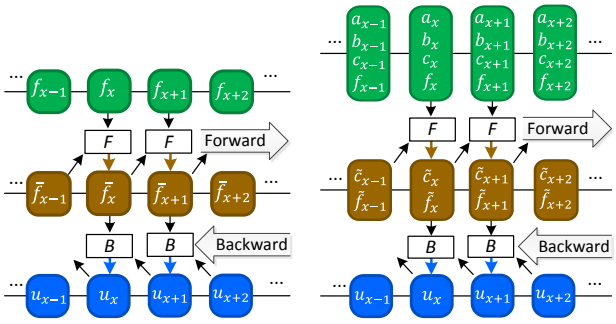
The solution $\mathbf{u}_h$ for the 1D signal in (7) is obtained in a recursive manner. Intermediate outputs $\tilde{c}_x$ and $\tilde{f}_x^h$ are recursively computed along a forward direction as follows:

$$\begin{aligned} \tilde{c}_x &= c_x / (b_x - \tilde{c}_{x-1} a_x) \\ \tilde{f}_x^h &= (f_x^h - \tilde{f}_{x-1}^h a_x) / (b_x - \tilde{c}_{x-1} a_x) \end{aligned}, \quad (x = 1, \cdots, W-1) \quad (8)$$

with $\tilde{c}_0 = c_0 / b_0$ and $\tilde{f}_0^h = f_0^h / b_0$. Then, an output u_x^h is recursively obtained along a backward direction as

$$u_x^h = \tilde{f}_x^h - \tilde{c}_x u_{x+1}^h, \quad (x = W-2, \cdots, 0) \quad (9)$$

with $u_{W-1}^h = \tilde{f}_{W-1}^h$.



(a) Local EP recursive filters (b) Global smoother

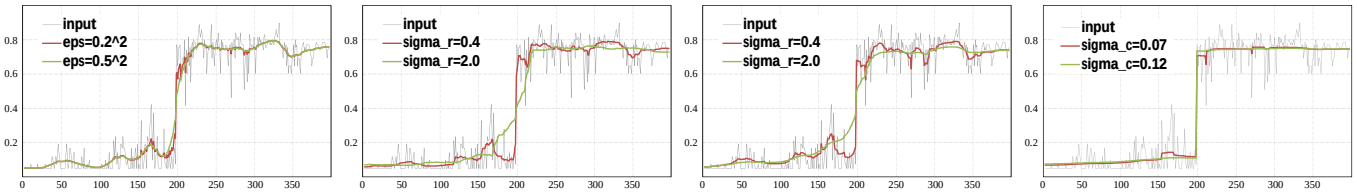
Fig. 2. Recursive data propagation on a 1D signal f of (a) local EP recursive filters [9], [10], [23] and (b) our 1D global smoother. 'F' and 'B' denote simple arithmetic operations. Although the arithmetic operations used in both methods are different, they share a similar data propagation scheme; the output u is recursively computed along forward and backward directions. Specifically, intermediate results (a) $\bar{f}_x$ and (b) $\tilde{c}_x$ and $\tilde{f}_x$ are recursively computed using (28) and (8), respectively, in the forward operation ('F'). Then, the output u_x is obtained using (29) for (a) and (9) for (b) in the backward operation ('B'). Our global smoother, however, achieves a much better smoothing quality than the local EP filters, also with a comparable runtime.

Such a computation procedure is conceptually similar to that of some $O(N)$ separable image filtering algorithms such as the recursive filter (RF) of the DT [9] and the recursive bilateral filter [10], [23] (see Appendix A for more details). Figure 2 illustrates the data propagation schemes of these local EP recursive filters and our global smoother for 1D signals. Our global scheme achieves a much better smoothing quality than those efficient EP filters (e.g. no halos), as shown in Figure 1, yet with a comparable runtime, since our solution yields an exact minimum for the 1D energy function (6).

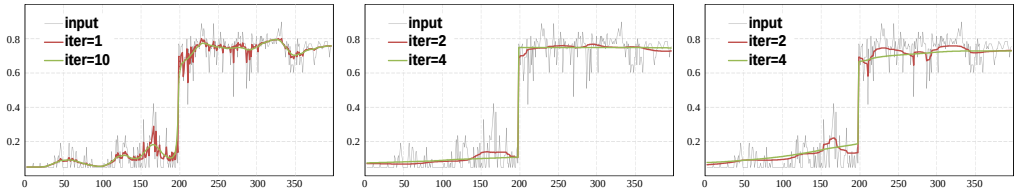
Figure 3 compares the smoothing quality on 1D signal. The author-provided MATLAB codes of existing EP filters were slightly modified, so that they work on 1D signal. The best results were obtained by tuning the parameters of the GF and DT methods. All these methods suffer from less flattening and/or halo artifacts, but our method produces excellent results with sharp edges preserved. We further investigate the performance of these EP filters in Figure 3(e)-(g), when applied with multiple iterations. Here, the weight kernels are recomputed with intermediate results every iteration, and the range kernel parameters were set relatively smaller to avoid over-smoothing during iterations, $\epsilon = 0.1^2$ and $\sigma_r = 0.2$. Even after multiple iterations, the GF and the RF of the DT still have a difficulty in flattening the signal and/or produces halo artifacts. In contrast, the normalized convolution (NC) filter of Figure 3 (f) smoothes similar regions while preserving relevant edges very well, which is consistent with what was reported in [9]. Such results, however, are achievable only by iterating the NC filter even for 1D signal, and estimating the total number of iterations becomes even more challenging when it comes to a multi-dimensional signal (e.g. 2D image). Indeed, the GF-like filtering shown in Figure 3 (e) may be desirable in some applications where a locality-preserving smoothing is needed, e.g. matting.

B. Separable Approximate Algorithm

Now, we consider a method for efficiently smoothing a 2D signal using our 1D solver. In general, solving a linear system



(a) GF ($iter = 1$) (b) NC of DT ($iter = 1$) (c) RF of DT ($iter = 1$) (d) Ours ($iter = 1$)



(e) GF with iter. (ϵ=0.1^2) (f) NC of DT with iter. (σ_r=0.2) (g) RF of DT with iter. (σ_r=0.2)

Fig. 3. Results on 1D signal with varying parameters. To obtain the best results for the GF [8] and DT [9] methods, the parameters were tuned experimentally: (a) GF with a radius $r = 9$ and varying ϵ and (b)(c) NC and RF of DT with $\sigma_s = 50$ and varying σ_r. In our method, λ was set to 50.0^2. In (a)-(d), the smoothing operations were applied once (iteration number=1) for fair comparison. The EP filters (a)-(c) suffer from less flattening and/or halo artifacts, while our method achieves an excellent smoothing result in terms of signal abstraction and halo. To further investigate the smoothing quality of the EP filters, we iterated them in (e)-(g). It should be noted that the weight kernels used were re-computed every iteration, since keeping initial weight kernels (obtained from a given input) during iterations does not smooth the signal out very well. The range parameters were set relatively smaller to avoid over-smoothing during iterations, $\epsilon = 0.1^2$ and $\sigma_r = 0.2$. After 4 iterations, the NC filter produced the smoothed results while preserving strong step edges, which is consistent with what was reported in [9]. Such results, however, are achievable only by iterating the NC filter even for 1D signal.

Algorithm 1 : Separable global smoother for 2D image smoothing

Target Operation: $\mathbf{u} = (\mathbf{I} + \lambda\mathbf{A})^{-1}\mathbf{f}$
Input: 2D image $f(x, y)$; $\mathbf{f}$ ($S \times 1$ vector)
Input: 2D guide image $g(x, y)$; $\mathbf{g}$ ($S \times 1$ vector)
($g = f$ for image filtering, $g \neq f$ for joint filtering)
Output: 2D image $u(x, y)$; $\mathbf{u}$ ($S \times 1$ vector)
Parameters and Notations
T: iteration num., S=HW: image size
λ: smoothing parameter
$\mathbf{A}_h$ (or $\mathbf{A}_v$): $W \times W$ (or $H \times H$) three-point Laplacian matrix
f^h, g^h, u^h: 1D hor. signal of f, g, u; $\mathbf{f}_h$, $\mathbf{g}_h$, $\mathbf{u}_h$ ($W \times 1$ vector)
f^v, g^v, u^v: 1D ver. signal of f, g, u; $\mathbf{f}_v$, $\mathbf{g}_v$, $\mathbf{u}_v$ ($H \times 1$ vector)

Algorithm
Initialize $\mathbf{u} \leftarrow \mathbf{f}$
for $t = 1 : T$ **do**
  compute λ_t using (12)
  for $y = 0 : H - 1$ **do**
    $f^h(x) \leftarrow u(x, y)$ for all $x = 0, \cdots, W - 1$
    if (image filtering), **then** $g^h(x) \leftarrow f^h(x)$ for all x
    compute $w_{x,i}$ using g^h for $i \in \mathcal{N}_h(x)$
    build a tridiagonal matrix $\mathbf{A}_h$
    solve $(\mathbf{I}_h + \lambda_t\mathbf{A}_h)\mathbf{u}_h = \mathbf{f}_h$ using (8) and (9)
    $u(x, y) \leftarrow u^h(x)$ for all $x = 0, \cdots, W - 1$
  end
  for $x = 0 : W - 1$ **do**
    $f^v(y) \leftarrow u(x, y)$ for all $y = 0, \cdots, H - 1$
    if (image filtering), **then** $g^v(y) \leftarrow f^v(y)$ for all y
    compute $w_{y,j}$ using g^v for $j \in \mathcal{N}_v(y)$
    build a tridiagonal matrix $\mathbf{A}_v$
    solve $(\mathbf{I}_v + \lambda_t\mathbf{A}_v)\mathbf{u}_v = \mathbf{f}_v$ using (8) and (9)
    $u(x, y) \leftarrow u^v(y)$ for all $y = 0, \cdots, H - 1$
  end
end

defined with more neighbors in a 2D image is usually computationally demanding, even with recent sparse matrix solvers [19]. Thus, focusing on many low-level vision tasks where a four-neighbor $\mathcal{N}_4$ (or an eight-neighbor $\mathcal{N}_8$) smoothness constraint is often used to define a prior term, we decompose a 2D spatial domain along each spatial dimension, so that our 1D solver is directly applicable to the decomposed 1D signals.

The most common approach of smoothing a multidimensional signal in a separable manner is to sequentially apply the 1D solver along each dimension of the signal [30], [31]. For a given 2D image, the 1D solver is iteratively applied along the rows and the columns of the image. In the 2D smoothing context, a single iteration consisting of horizontal and vertical 1D solvers is not enough to propagate information across edges, leading to the 'streaking artifact' which is common in separable algorithms [9], [10]. Therefore, we perform 2D smoothing by applying sequential 1D global smoothing operations for a multiple number of iterations. In this scheme, the amount of spatial smoothing is progressively reduced by adjusting λ_t in (7), considering that intermediate results are coarsened during iterations. This strategy helps reduce streaking artifacts which may be caused by the separable smoothing algorithm.

To control the total amount of spatial smoothing in applying the sequential 1D smoothing operations, we first analyze a response function of (7), i.e., $(\mathbf{I}_h + \lambda_t\mathbf{A}_h)^{-1}$. Since our objective is to adjust the amount of spatial smoothing in the separable algorithm, the analysis starts with the case of a linear smoothing. When the weight function in (2) is employed, we can rewrite three nonzero elements of $(\mathbf{I}_h + \lambda_t\mathbf{A}_h)^{-1}$ as $a_x = c_x = -\lambda_t$ and $b_x = 1 + 2\lambda_t$. The frequency response of this homogeneous matrix is given by $\mathbb{R}(\omega) = 1/(1 + \lambda_t\omega^2)$ [32]. Its impulse response function $r(x)$ is then written as

$$r(x) = \frac{1}{2\sqrt{\lambda_t}} e^{-\frac{|x|}{\sqrt{\lambda_t}}}, \tag{10}$$

with a standard deviation $\sigma_R = \sqrt{2\lambda_t}$. The linear smoothing is performed in a way of $u_h(x) = r(x) * f_h(x)$, where $*$ represents a convolution operator. The total amount of spatial smoothing is defined by adding variances of the spatial kernel

used for each iteration. Here, it should be noted that applying a linear smoothing with the exponential kernel multiple times approximates the Gaussian (not exponential) linear smoothing. If the total amount of spatial smoothing is defined as a variance λ of the Gaussian kernel, i.e., $1/\sqrt{2\pi\lambda}\exp(-x^2/2\lambda)$, and the standard deviation of (10) is progressively reduced by half through iterations, the following equation can be induced where the variance λ of the Gaussian kernel matches the sum of the variance of the exponential kernel used at each iteration.

$$\sum_{t=1}^{T} 2\lambda_t = \sum_{t=1}^{T} \left(\frac{1}{2}\right)^{2t} 2\lambda_1 = \lambda \tag{11}$$

where λ_t is the smoothing parameter used in the t^{th} iteration, and T represents the total number of iterations. From this equation, λ_t is obtained as follows:

$$\lambda_t = \frac{3}{2}\frac{4^{T-t}}{4^T-1}\lambda. \tag{12}$$

The separable algorithm in case of 2D image smoothing is summarized in Algorithm 1. When the smoothing parameter λ (representing the total amount of spatial smoothing) and the total number of iterations T are given as inputs, the separable 1D smoothing operations are performed along the horizontal and vertical directions with λ_t computed at the t^{th} iteration. As mentioned above, this sequential operations approximate the spatial kernel with the Gaussian function, different from original WLS formulation (3), but we found the difference of the smoothing results from using the approximated Gaussian kernel to be marginal.

We analyze the parameter used in our 2D separable method described in Algorithm 1. In order to decide the typical number of iterations (T), we adopted a similar method to what used in the DT method [9]. The smoothing result obtained when $T = 15$ is assumed to be streaking-artifact free, and the structural similarity (SSIM) index [24] is measured by comparing this to the result obtained after T iterations. Using various σ_c ($0.008 \sim 0.1$) and λ ($3.0^2 \sim 60.0^2$), the FGS was performed for 30 natural images with diversified contents. For each T with varying σ_c, minimum SSIM values were computed among 30 images and all λ. We found that our smoothing achieved converged results after three iterations. For instance, the estimated SSIM values range from 0.982 to 0.995 after 3 iterations for all σ_c tested. In our work, the total number of iterations T is set to 3. Figure 4 shows visual results with varying T. The parameters were set as $\sigma_c = 0.03$ and $\lambda = 30.0^2$. The smoothing result of our FGS method with $T = 3$ is almost similar to that of $T = 15$.

C. Extension to Sparse Data Interpolation

The separable global smoother is extended into a sparse data interpolation scenario, where the input data is sparse, such as image colorization [13], recoloring [9], and interactive image segmentation [33]. Given a guide (color or grayscale) image g and a sparse input (e.g. user scribbles) f, the energy function can be expressed as:

$$\sum_p \left(h_p(u_p - f_p)^2 + \lambda \sum_{q\in\mathcal{N}_4(p)} w_{p,q}(g)(u_p - u_q)^2 \right), \tag{13}$$

where h_p is an index function, which is 1 for valid (constraint) pixels, 0 otherwise. In the colorization, the guide image is a grayscale image. The output u minimizing this function is obtained with the following linear system:

$$(\mathbf{H} + \lambda\mathbf{A})\mathbf{u} = \mathbf{H}\mathbf{f}, \tag{14}$$

where $\mathbf{H}$ is a diagonal matrix whose elements are 1 for valid pixels and 0 otherwise. By solving the above linear system, the sparse data f is adaptively propagated by considering the structure of the guidance image g.

A straightforward approach of solving (14) using our method is to decompose the energy function (13) into x and y axes as in (6). However, such decomposition leads to unstable results due to the sparse input data f. Unlike the image filtering task using a dense input, the decomposed 1D signal used in the sparse data interpolation may sometimes have no valid data, resulting in a null output.

Similar to [9], [34], we instead smooth both the input signal $\mathbf{f}$ and the index vector $\mathbf{h}$. Let $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{u}}_h$ denote the smoothed outputs of $\mathbf{f}$ and $\mathbf{h}$ obtained using (5). A final solution $\mathbf{u}$ is then computed as $\tilde{\mathbf{u}}./\tilde{\mathbf{u}}_h$, where '$./$' represents an element-wise division operator. Intuitively, $\tilde{\mathbf{u}}_h$ represents how much the sparse input $\mathbf{f}$ is propagated to each pixel. Formally, the solution can be written as

$$\mathbf{u}(m) = \frac{((\mathbf{I} + \lambda\mathbf{A})^{-1}\mathbf{f})(m)}{((\mathbf{I} + \lambda\mathbf{A})^{-1}\mathbf{h})(m)}, \tag{15}$$

where m represents a scalar index corresponding to a 2D pixel p, i.e., $m \in \{0, \cdots, S-1\}$. The inversion of the sparse matrix is performed using Algorithm 1. Therefore, hereafter, we will always explain our solver based on a variant of (1) that assumes a dense input. Namely, the solution of a dense input is computed by applying Algorithm 1 once, while that of a sparse input is obtained by applying Algorithm 1 twice. Table II summarizes the proposed solvers according to the type of the data term, including (19) using an aggregated data term, which is explained in Section IV.E.

TABLE II
PROPOSED SOLVERS ACCORDING TO THE TYPE OF INPUT DATA.

Solution $\mathbf{u}(m)$	Dense input	Sparse input
Per-pixel data in (1)	$((\mathbf{I} + \lambda\mathbf{A})^{-1}\mathbf{f})(m)$	$\frac{((\mathbf{I}+\lambda\mathbf{A})^{-1}\mathbf{f})(m)}{((\mathbf{I}+\lambda\mathbf{A})^{-1}\mathbf{h})(m)}$
Aggregated data in (19)	$((\mathbf{D} + \lambda\mathbf{A})^{-1}\mathbf{C}\mathbf{f})(m)$	$\frac{((\mathbf{D}+\lambda\mathbf{A})^{-1}\mathbf{C}\mathbf{f})(m)}{((\mathbf{D}+\lambda\mathbf{A})^{-1}\mathbf{C}\mathbf{h})(m)}$

D. Extension to IRLS

The proposed global smoother can also be used to optimize an objective function defined on the L_γ norm with the iterative re-weighted least squares (IRLS) algorithm [35]. Specifically, we can define the following energy function with $0 < \gamma < 2$:

$$J_\gamma(u) = \sum_p \left((u_p - f_p)^2 + \lambda \sum_{q\in\mathcal{N}_4(p)} w_{p,q}(g)|u_p - u_q|^\gamma \right). \tag{16}$$

(a) Input (b) $T = 1$ (c) $T = 2$ (d) $T = 3$ (e) $T = 15$

Fig. 4. Effect of the total number of iterations T: $\sigma_c = 0.03$ and $\lambda = 30.0^2$. The smoothing result with $T = 3$ is almost similar to that of $T = 15$. In the results with $T = 1$ and 2, the streaking artifact is still visible (see the boundary of 'shadow'). These results are best viewed in their original resolution.

This non-convex function is iteratively minimized with the weight of the prior term adjusted with $u^{(k)}$, which is an intermediate estimate at the k^{th} iteration:

$$w_{p,q}(g)|u_p - u_q|^\gamma \approx \frac{w_{p,q}(g)}{|u_p^{(k)} - u_q^{(k)}|^{2-\gamma} + \kappa}(u_p - u_q)^2 = \phi_{p,q}(g, u^{(k)})(u_p - u_q)^2, \quad (17)$$

where κ is introduced for numerical stability. For each iteration, the weight $\phi_{p,q}(g, u^{(k)})$ updated with the current estimate $u^{(k)}$ is used to obtain a new estimate $u^{(k+1)}$ by minimizing the following energy function:

$$J_\gamma^{(k+1)}(u) = \sum_p \left((u_p - f_p)^2 + \lambda \sum_{q \in \mathcal{N}_4(p)} \phi_{p,q}(g, u^{(k)})(u_p - u_q)^2 \right). \quad (18)$$

Thus, minimizing the energy function in (16) finally becomes equivalent to solving a series of linear equation systems $(\mathbf{I} + \lambda \mathbf{A}^{(k)})\mathbf{u}^{(k+1)} = \mathbf{f}$ $(k = 0, \cdots, K-1)$, where $\mathbf{A}^{(k)}$ is a Laplacian matrix computed with the weight $\phi(g, u^{(k)})$ at the k^{th} iteration. Note that K represents the *external* iteration number used to minimize (16), each of which is solved by using Algorithm 1 with T iterations. This extension can be used in a range of applications, where the prior term is defined on the L_γ norm. For instance, in [17], the prior term is defined in a form of total variation (L_1 norm), and the objective function is optimized using the preconditioned conjugate gradient (PCG) [19]. We can easily replace the solver with our approach, achieving a much better runtime efficiency (see results in Section VI.B).

E. Using Aggregated Data Term

The energy functions (1) and (13) enforce a hard constraint on the data term. In other words, the data constraint is exactly imposed with respect to a given observation through an un-weighted per-pixel penalty function. This assumption, however, may be violated in many applications when there exists an imprecise input data, e.g. incorrect user scribbles in image colorization. Interestingly, it was shown that a more robust data constraint using an aggregated input term leads to a better quality in the image denoising algorithm [36]. In a similar context, An and Pellacini proposed to use the aggregated data term for handling the erroneous input data in image editing applications [21]. However, this method based on a dense affinity matrix is extremely slow to solve. By considering a soft data aggregation constraint, we re-define the energy function as follows:

$$\sum_p \left(\sum_{r \in \mathcal{N}_D} c_{p,r}(g)(u_p - f_r)^2 + \lambda \sum_{q \in \mathcal{N}_4} w_{p,q}(g)(u_p - u_q)^2 \right), \quad (19)$$

where $\mathcal{N}_D$ (of a $L \times L$ size) represents a set of neighbors used to aggregate the input data f. Note that $\mathcal{N}_D$ is not limited to $\mathcal{N}_4$ neighbors (used to define the prior term), but usually using more neighbors for the aggregation is recommended for ensuring a large support. Here, $c_{p,r}$ is defined using a bilateral kernel, e.g. $\exp\big(- \parallel p - r \parallel^2 / 2\sigma_{ds} - \parallel g_p - g_r \parallel^2 / 2\sigma_{dc}\big)$, and it decides the contribution of neighboring data constraints.

The linear system formulation is then obtained through a similar derivation as follows:

$$(\mathbf{D} + \lambda \mathbf{A})\mathbf{u} = \mathbf{Cf}, \quad (20)$$

where $\mathbf{A}$ is defined with $\mathcal{N}_4$ as in (4). $\mathbf{D}$ is a diagonal matrix whose diagonal values are the sum of weights $\sum_r c_{p,r}$ defined inside the kernel window $r \in \mathcal{N}_D(p)$. $\mathbf{C}$ is a kernel matrix whose nonzero elements are given by weights $c_{p,r}$, and $\mathbf{Cf}$ represents an unnormalized bilateral sum of the input signal f.

$$\mathbf{D}(m,n) = \begin{cases} \sum_{l \in \mathcal{N}_D(m)} c_{m,l} & m = n \\ 0 & \text{otherwise} \end{cases}, \quad (21)$$

$$\mathbf{C}(m,n) = \begin{cases} 0 & \text{otherwise} \\ c_{m,n} & n \in \mathcal{N}_D(m) \end{cases}. \quad (22)$$

All the sparse matrices used in this paper are constructed from two sources: one captures the data term ($\mathbf{I}$ or $\mathbf{D}$), and the other is from the smoothness term ($\mathbf{A}$). The linear system (3) using the per-pixel data term always assigns a constant reliability ($\mathbf{I}$) to all pixels. In contrast, (20) adaptively considers the reliability metric specified by $\mathbf{D}$, which is a sum of weights inside the kernel $\mathcal{N}_D$. By leveraging such reliability, the linear system (19) becomes more robust against errors that may exist in the input data. In summary, the solutions of dense or sparse inputs using the aggregated data term are written as

$$\mathbf{u}(m) = ((\mathbf{D} + \lambda \mathbf{A})^{-1}\mathbf{Cf})(m), \quad (23)$$

$$\mathbf{u}(m) = \frac{((\mathbf{D} + \lambda \mathbf{A})^{-1}\mathbf{Cf})(m)}{((\mathbf{D} + \lambda \mathbf{A})^{-1}\mathbf{Ch})(m)}. \quad (24)$$

The sum of bilateral weights $\mathbf{D}$ and the unnormalized bilateral sum $\mathbf{Cf}$ can be efficiently computed using recent $O(N)$ image filtering algorithms [6], [10], which yield the two terms separately. It is worth noting that some $O(N)$ filters [8], [9] are not feasible to calculating $\mathbf{D}$ and $\mathbf{Cf}$, since they

directly produce a filtered output (i.e., $\mathbf{D}^{-1}\mathbf{Cf}$). In this work, we utilize the recursive BF (RBF) [10] (see (14) and (22) in [10]), but any other kind of fast filters yielding $\mathbf{D}$ and $\mathbf{Cf}$ can also be employed. We compute $\mathbf{D}$ and $\mathbf{Cf}$ using a 2D RBF with one iteration. A filtering result of the 2D RBF is adjusted by two parameters: a range parameter σ_{dc} and a spatial parameter α (ranging 0∼1), which plays a role similar to σ_{ds}.

F. Aggregated Data Term vs. Multiple Filtering

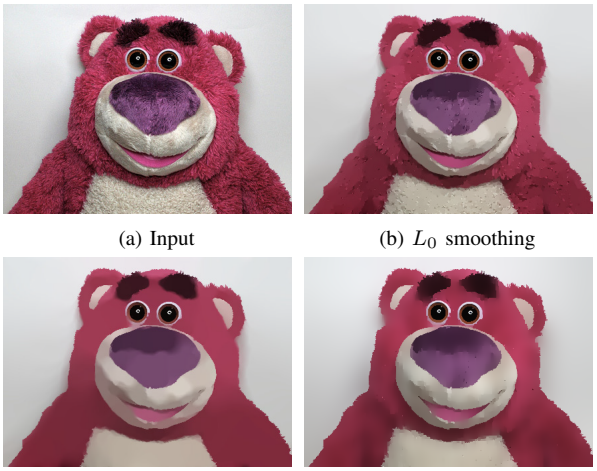
It was discussed in [21] that applying nonlinear local filtering multiple times may provide similar capability of dealing with erroneous input data, e.g. sequentially performing the joint BF (JBF) [37] about 50 times. Specifically, by setting λ to 0, the solution of (20) becomes the BF (i.e., $\mathbf{D}^{-1}\mathbf{Cf}$). Thus, performing the BF multiple times may help suppress the artifacts incurred by erroneous input data to some extent. However, as already pointed out in [21], it is very challenging to find an optimal number of iterations for the local filters (e.g. BF). More importantly, such local filter based approaches do not handle imprecise inputs effectively. Figure 12 shows the results of image colorization using imprecise user scribbles, when iteratively applying the edge-preserving filters such as the brute-force JBF [37], the GF [8], and the RF of the DT [9]. The color bleeding artifacts incurred by the imprecise inputs seem to be reduced in some results, but the simple iterative local smoothing strategy does not remove the artifacts effectively.

V. Performance

We first compare the smoothing performance of our method applied to image filtering against that of state-of-the-art fast $O(N)$ edge-preserving filters: DT [9] and GF [8]. Other existing $O(N)$ filters such as the permutohedral lattice BF [7] and the constant time BF using a bilateral grid [5], [6] are not compared, since they are relatively slow [7] or rely on a space-color sampling method [5], [6] that may degenerate a filtering quality. The WLS-based method [15] is also compared together. In our method, the default parameters for the total number of iterations T is set to 3, unless specified otherwise.

A. Smoothing Quality

As already shown in Figure 1, our smoother outperforms existing fast $O(N)$ local filters in terms of image abstraction and edge-preserving capability, e.g. no halo artifact. For all the results of the existing methods, we used the source code provided by the authors; GF[1], DT[2], and WLS[3]. For our method, $w_{p,q}$ is defined with (2). We also compare the smoothing quality on a natural image in Figure 5. As pointed out in [25], halo artifacts on color boundaries are unavoidable for local filters. They also suffered from less image abstraction in strong textures. Using larger ϵ or σ_r in the GF and DT methods leads to more smoothing, but they still fail to smooth out strong textures, and often lose small details (see thin 'flag' in the top of the center area). As stated in [9], the NC works better than the RF in terms of image abstraction, but it still has the less flattening issue in strong textures and has a trade-off between halo artifacts and image abstraction. Similar smoothing results were shown in Figure 6. The GF [8] does not smoothen regions near high contrast edges or high variance regions (e.g. marked by arrows in Figure 6(b)) due to its inherent limitation that may arise from a multipoint aggregation based on a box window. In contrast, our method produces excellent smoothing results with no blurring on the object boundaries. While the local filters adaptively determine the amount of smoothing by considering local neighbors only, our global smoother globally distributes an intensity profile by taking into account an overall energy cost [25].



(a) Input (b) L_0 smoothing (c) BF+L_0 smoothing (d) Ours

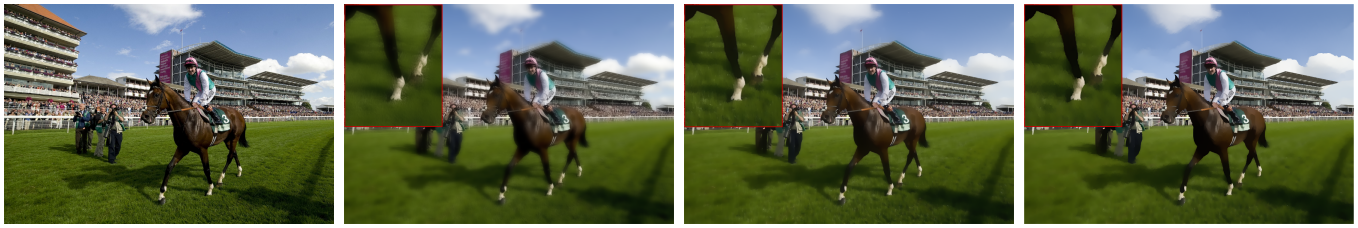
Fig. 7. More results of image smoothing using (b) L_0 smoothing [38], (c) BF + L_0 smoothing, and (d) our method (5) with $\sigma_c = 0.04$ and $\lambda = 20.0^2$. Note that the result of (c) was obtained by first applying the BF, which suppresses noise-like structures, followed by their L_0 smoothing [38]. Our smoother using (5) suppresses the noise-like structures with no pre-processing step such as the BF.

More results for image smoothing are compared with L_0 smoothing [38] in Figure 7. We applied the smoothing to a 'bear' image containing many noise-like structures. It was explained in [38] that the L_0 smoothing method can be complementary to existing local filtering methods such as the BF [2] to smooth out noise-like structures in Figure 7 (a), since the L_0 smoothing method does not use a spatial averaging. In contrast, our smoother using (5) shows a similar smoothing quality to (c) BF+L_0 smoothing in suppressing such noise-like structures, without performing such a pre-filtering step (e.g. the BF). Basically, the WLS based edge-preserving smoothing is equivalent to applying $(\mathbf{I}+\lambda\mathbf{A})^{-1}$ to filter an input image f, and each row of this matrix can be seen as a smoothing kernel with a large support varying according to λ and σ_c (or β) [15]. Thus, through this adaptive averaging process, our fast global smoother handles such noise-like structures better than the L_0 smoothing, yet more efficiently.

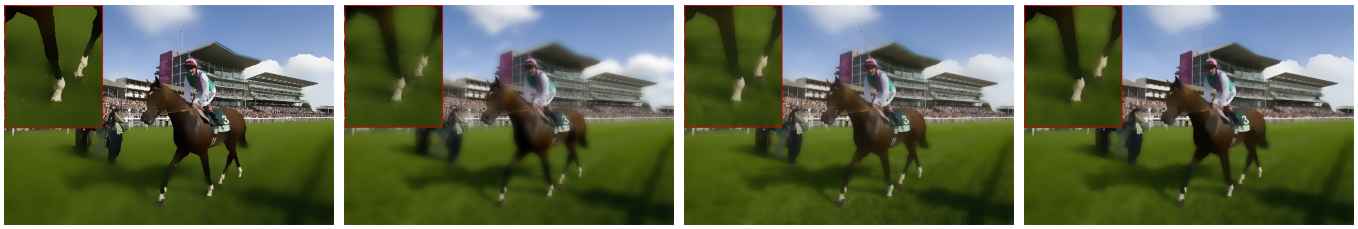
B. Smoothing with $\mathcal{N}_4$ and $\mathcal{N}_8$ neighbors

Our solver is applicable to solving the Laplacian matrix defined with both $\mathcal{N}_4$ and $\mathcal{N}_8$ neighbors. In case of using $\mathcal{N}_8$

[1]http://research.microsoft.com/en-us/um/people/kahe/eccv10/
[2]http://www.inf.ufrgs.br/∼eslgastal/DomainTransform/
[3]http://www.cs.huji.ac.il/∼danix/epd/



(a) Input (b) GF ($r = 7$, $\epsilon = 0.1^2$) (c) RF of DT ($\sigma_s = 50.0$, $\sigma_r = 0.5$) (d) NC of DT ($\sigma_s = 50.0$, $\sigma_r = 0.5$)



(e) Ours ($\lambda = 25.0^2$, $\sigma_c = 0.035$) (f) GF ($r = 11$, $\epsilon = 0.2^2$) (g) RF of DT ($\sigma_s = 50.0$, $\sigma_r = 1.0$) (h) NC of DT ($\sigma_s = 50.0$, $\sigma_r = 1.0$)

Fig. 5. Image smoothing results: the GF [8] with (b) a radius $r = 7$ and $\epsilon = 0.1^2$ (f) $r = 11$ and $\epsilon = 0.2^2$, the recursive filter (RF) of the DT [9] with (c) $\sigma_s = 50.0$ and $\sigma_r = 0.5$ (g) $\sigma_s = 50.0$ and $\sigma_r = 1.0$, the normalized convolution (NC) of the DT [9] with (d) $\sigma_s = 50.0$ and $\sigma_r = 0.5$ (h) $\sigma_s = 50.0$ and $\sigma_r = 1.0$, and (e) our method with $\sigma_c = 0.035$ and $\lambda = 25.0^2$. The local filters produced halo artifacts and/or suffered from less image abstraction in strong textures. Also, using larger ϵ or σ_r leads to more smoothing in these local filters, but they still fail to smooth out strong textures, and often lose small details (see thin 'flag' in the top of the center area). In contrast, our method outperforms these local filters in terms of image abstraction and preserves such thin details very well. For better visualization, please see the close-up of an electronic version.

(a) Input (b) GF (c) RF of DT (d) NC of DT (e) Ours

Fig. 6. Image smoothing results: (a) a 'lamp' image, (b) GF [8] with a radius $r = 15$ and $\epsilon = 0.3^2$, (c) RF of DT (from author's web) [9], (d) NC of DT with $\sigma_r = 0.7$ and $\sigma_s = 40.0$, and (e) our method with ($\sigma_c = 0.03$) and $\lambda = 20.0^2$. We applied a similar amount of smoothing for all the methods. These results are best viewed in their original resolution.

neighbors, we additionally iterate diagonal and anti-diagonal passes in Algorithm 1. Specifically, we apply 1D solvers in a clock-wise direction. Figure 8 compares the smoothing results with $\mathcal{N}_4$ and $\mathcal{N}_8$ neighbors. In order to apply the same amount of smoothing, we set the total number of iterations T to 4 and 2 for $\mathcal{N}_4$ and $\mathcal{N}_8$, respectively. Namely, in the filtering with $\mathcal{N}_8$ neighbors, the number of horizontal, anti-diagonal, vertical, and diagonal passes is all set to 2. The smoothing results are very similar to each other, except weak diagonal edges with smoothly-varying gradients (e.g. 'cloud'). Also, the memory access time for including diagonal and anti-diagonal passes would become longer. In this work, all the results are obtained using $\mathcal{N}_4$ neighbors, similar to existing separable algorithms.

C. Computational Efficiency

We report the timing results measured on a PC with a 3.4 GHz CPU and 8 GB of memory. For comparison, by referring to the author-provided software (implemented in MATLAB), we implemented the RF of DT method (the fastest local filter) in C, and measured the runtime. The runtimes of the RF of DT and our method (also implemented in C) for filtering a 1M pixel RGB image on a single CPU core are 0.05 and 0.10 seconds, respectively. Three iterations were used in both methods. Note that the original DT paper [9] reported that filtering a 1M pixel RGB image using three iterations takes 0.06 second (on 2.8GHz CPU). In [25], the runtime of GF method was reported as 0.15 second for a 1M pixel RGB image. To report the best runtime of the WLS method [15] using sparse matrix solvers, we arithmetically calculated the runtime with the timing results reported in the recent work, e.g. HSC of Table 2 in [19]. They reported that the runtime of the 'EPD sharpening' operation for 5 scales (i.e., applying the smoothing four times) on a 4.2M pixel RGB image is 55.7 second, and thus the runtime for filtering a 1M pixel RGB image once arithmetically is about 3.3 second. This runtime analysis might vary depending on the degree of code optimization and the hardware used in the experiments, but our global smoother is much faster (over 30×) than the WLS method using the recent matrix solver [19]. It is also very comparable even to the state-of-the-art fast local filters,

(a) Result with $\mathcal{N}_4$ after 4 iter. (b) Result with $\mathcal{N}_8$ after 2 iter.

Fig. 8. Image smoothing results with (a) $\mathcal{N}_4$ and (b) $\mathcal{N}_8$ neighbors. For weak diagonal edges with smoothly-varying gradients (e.g. 'cloud'), our smoother (5) using $\mathcal{N}_8$ neighbors has a tendency to slightly better preserve diagonal edges.

(a) Input (b) WLS method (c) Ours

Fig. 9. Results of multi-scale detail manipulation: using (b) WLS method with $\beta = 1.2$ and $\lambda = 0.8$, and (c) our method (5) with $\beta = 1.2$ and $\lambda = 0.8$. Here, a weighting kernel $w_{p,q}(g) = \left(|z_p - z_q|^\beta + \delta\right)^{-1}$ is used, where z is the log-luminance channel of a guidance color image g. Thus, λ was set differently from that of other image smoothing results (e.g. Figs. 6 and 7), where a weighting kernel (2) is used.

although our method is based on a global solver.

By counting the number of arithmetic operations, we can also estimate the computational complexity of our method. For instance, the RF of the DT [9] uses 2 multiplications at every pixel for filtering a 1D signal, while our method using (8) and (9) employs 5 multiplications and 1 division for computing $\tilde{c}_x$, $\tilde{f}_x^h$ and u_x^h at every pixel. More specifically, $1/(b_x - \tilde{c}_{x-1}a_x)$ is first calculated and temporally saved to obtain $\tilde{c}_x$ and $\tilde{f}_x^h$. Thus, after considering the cost for other operations and memory access, the timing results (0.05 vs. 0.10 seconds) measured by our optimized code are in agreement. Our code is publicly available[4].

VI. Applications

This section demonstrates a range of low-level vision and computer graphics applications, where a Laplacian matrix is defined with a small number of neighbors ($\mathcal{N}_4$, here), e.g. multi-scale detail manipulation [15], structure extraction from texture [17], image colorization [13], imprecise edit propagation [21], and depth upsampling [22].

A. Multi-scale Detail Manipulation

Our global smoother can be used to manipulate image details by decomposing an image into several scales and recombining the decomposed signals. Similar to [15], for a $(L+1)$-level decomposition with an input image $f = u^0$, we produce a set of progressively smoothed outputs u^l $(l = 0, \cdots, L)$ using our smoother (5), and construct several details layers as $d^l = u^l - u^{l+1}$. The results are obtained by manipulating the detail layers with a sigmoid curve used in [15]. Figure 9 compares the results ($L = 2$) using our smoother and the WLS method [15]. Here, a weight kernel used for image smoothing is $w_{p,q}(g) = \left(|z_p - z_q|^\beta + \delta\right)^{-1}$, where z is the log-luminance channel of a guidance color image g, β controls the sensitivity of the gradients, and δ is a small constant preventing division by zero [15]. Achieving similar results, our method is about $30\times$ faster than the WLS method.

B. Structure Extraction from Texture: L_γ smoothing

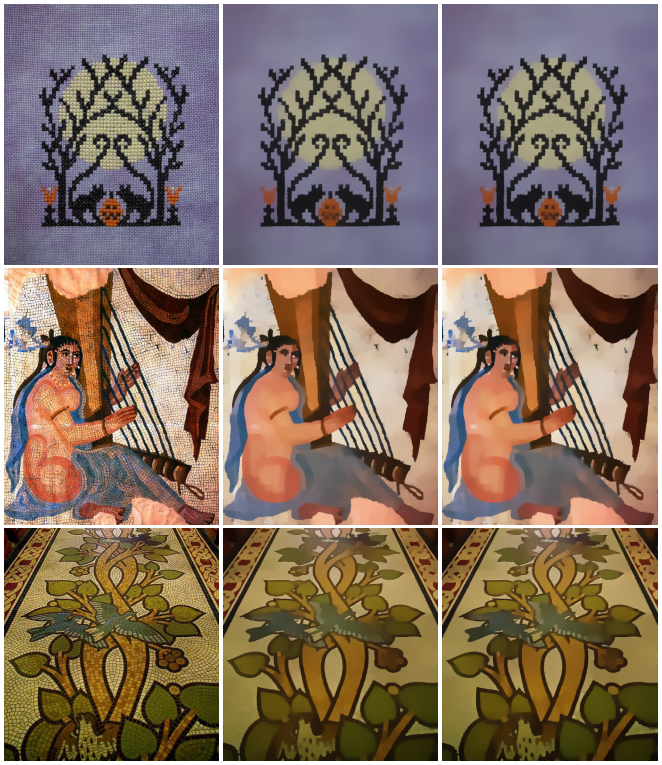
As mentioned in Table I, the proposed solver is applicable to many applications, some of which are not directly handled by local filtering algorithms. One of such applications is structure extraction from a highly-textured image, recently proposed by [17]. To extract meaningful structures from an image with complicate texture patterns, they proposed a new measure, called 'relative total variation' (RTV). The prior term is defined using a ratio of a pixel-wise windowed total variation (TV) and a windowed inherent variation. A new objective function is then formulated using the per-pixel fidelity term and the prior term defined with the RTV. Since this function employs the (relative) 'total variation' based on the L_1 norm, the authors proposed to minimize the function in a way similar to the IRLS algorithm, as explained in Section IV.D. Specifically, the objective function of (16) is written by setting γ to 1 and $w_{p,q}(g) = [G_\chi * \partial_{p,q} g + \epsilon_w]^{-1}$. $\partial_{p,q} g = g(p) - g(q)$ $(q \in \mathcal{N}_4(p))$ can be either x or y derivatives. ϵ_w is a small constant value to avoid division by zero, and G_χ is a Gaussian filter with a standard deviation χ and $*$ stands for a convolution operator.

Since a single iteration used in the IRLS is identical to solving a linear system with a five-point Laplacian matrix, our solver (5) can be seamlessly integrated in their minimizer. Figure 10 compares the results of the structure-texture decomposition implemented using the original method (b) and our smoother (c). The quality of these results is similar, but our solver is much faster. For instance, the two methods take about 3.7 and 0.6 seconds to process a 800×600 color image [17], respectively. It should be noted that these runtimes include all the processing units such as the RTV calculation which is relatively time-consuming. These results indicate that our solver can be useful in some tasks based on the L_γ norm $(0 < \gamma < 2$, usually using the IRLS procedure) as well as the weighted L_2 norm.

C. Sparse Data Interpolation (Colorization)

When a sparse input f is given along with a guidance image g, the sparse data is interpolated by referring to the structure of the guidance image. One of examples is image colorization using sparse color scribbles specified by a user. Similar to [13], we propagate the sparse color input f by smoothing it with the grayscale image g. Specifically, two chrominance channels U and V extracted from the input color scribbles are used as inputs f, and their interpolated outputs are then obtained

[4]https://sites.google.com/site/globalsmoothing/



(a) Input (b) RTV (c) Ours

Fig. 10. Results of structure-texture decomposition: (b) using the original method using a RTV measure [17] ($\sigma = 3$ and $\lambda = 0.015$) and (c) using our solver ($\sigma = 3$ and $\lambda = 0.019$). Here, σ ($\neq \sigma_c$) is a parameter for computing the RTV. We used the same L_1 penalty function as what was defined in [17]. Specifically, in (16), γ is set to 1 and the weighting function $w_{p,q}(g) = [G_\chi * \partial_{p,q} g + \epsilon_w]^{-1}$, where $\partial_{p,q} g = g(p) - g(q)$ ($q \in \mathcal{N}_4(p)$) can be either x or y derivatives and ϵ_w is a small constant value to avoid division by zero. Please refer to Section VI.B for more details. It should be noted that the balancing parameter λ used in the objective function can vary depending on application scenarios, since it should be set differently by considering the ratio of the data and prior terms used.

(a) Input (b) Levin et al. 2004 (c) Ours

Fig. 11. Results of image colorization: (b) using [13] and (c) using our solver ($\sigma_c = 0.01$ and $\lambda = 30.0^2$).

by following (15). A final colorized image is produced by converting the grayscale image g and the outputs U and V into the RGB color space. Figure 11 shows the results obtained with [13] and our global smoother.

D. Handling Imprecise (Sparse) Input using Aggregated Data

Next, we demonstrate how to handle imprecise inputs using our framework, while maintaining its computational advantage. For instance, when rough color scribbles are specified by a user, existing colorization methods assuming completely correct strokes may produce serious color artifacts. Figure 12 shows serious color bleeding artifacts (e.g. the 'neck', 'left hair' and 'bangs' regions of a 'girl' image) appeared in the colorization results obtained using our global smoother (15) (with a per-pixel data constraint) and other algorithms such as the JBF [37], the GF [8], the RF of the DT [9], and Levin's work [13]. It is worth noting that such artifacts are not effectively resolved even after applying the algorithms for a number of iterations (e.g. 10 to 50 times). In contrast, our method (24) using the aggregated data term achieves a better colorization result, justifying the effectiveness of the 'reliability' metrics defined by $\mathbf{D}$ of (21) and $\mathbf{Cf}$ of (22). For a stronger propagation, λ_t is fixed during iterations ($\lambda_t = \lambda$ for $t = 1, \cdots, T$), not being varied in a form of (12), and using a relatively large λ is recommended (e.g., 50.0^2).

The proposed method using (24) consists of three data aggregations ($\mathbf{D}$, $\mathbf{Cf}$, and $\mathbf{Ch}$) and two global smoothing operations (corresponding to numerator and denominator). We found that updating $\mathbf{Cf}$ and $\mathbf{Ch}$ in every (horizontal or vertical) iteration produces better results. $\mathbf{D}$ is computed only once, as it remains unchanged through iterations. As mentioned earlier, the data aggregation terms are computed using the 2D RBF [10], which takes about 0.035 seconds for filtering a 1M pixel three-channel image (using the author's public code). Here, an input $\mathbf{f}$ consists of two chrominance channels U and V, and $\mathbf{h}$ is a single-channel index function. Thus, the runtime for computing $\mathbf{D}$, $\mathbf{Cf}$, and $\mathbf{Ch}$ is about 0.225 second ($= 0.015 + 6 \times 0.035$). Our global smoothing operation is performed on a three-channel signal (corresponding to $\mathbf{f}$ and $\mathbf{h}$) with a runtime of 0.10 second. Totally, our method using (24) takes about 0.325 seconds ($= 0.225 + 0.10$) for colorization of a 1M pixel image ($T = 3$).

More results are provided in Figure 13. As expected, even when iteratively applying our smoother (15) with a per-pixel data constraint (10 iterations), the results still contain noticeable color artifacts, e.g. the upper part of a 'flower' image. The method employing a non-local smoothness constraint defined with dense neighbors [21] is expected to produce better results than our method (24). However, it is extremely time-consuming due to the densely defined prior term, and also there exists no fast $O(N)$ solution for the energy function defined with the dense prior term. Furthermore, even such a method [21] may often need a user to add more color scribbles in a progressive manner as in [13]. Our method is very fast thanks to the use of the fast global smoother with a linear complexity, and thus is able to yield the colorization results at an interactive rate, allowing the user to include additional inputs instantly. Compared to the existing colorization approaches based on the optimization [13] and filtering algorithms [8], [9], our solution achieves a much better quality with a comparable runtime to the filter based methods, meaning that the workload provided by a user can be reduced significantly.

E. Depth Upsampling

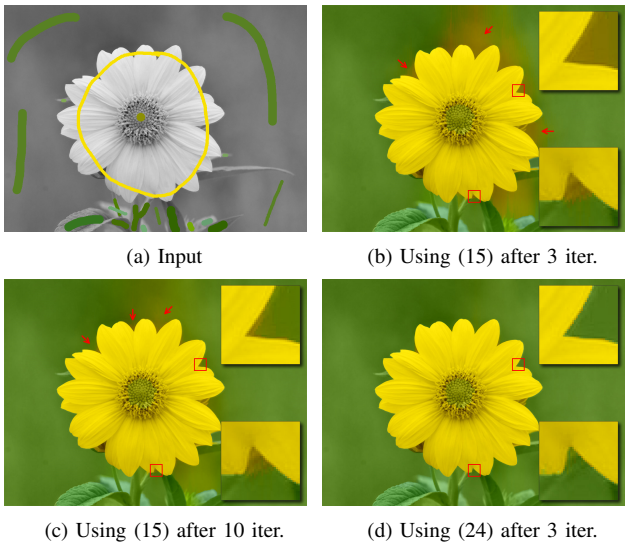
We also applied our global smoother to depth upsampling task [22], where a low-resolution depth map and its associated high-resolution color image are used as inputs. Its objective is to enhance the quality of the input low-resolution depth map by increasing its spatial resolution. We performed the experiments with ground truth depth maps from the Middlebury test

(a) Input (b) JBF (50 iter.) (c) GF (30 iter.) (d) RF of DT (50 iter.)

(e) Levin et al. 2004 (f) Ours with per-pixel data (3 iter.) (g) Ours with per-pixel data (10 iter.) (h) Ours with aggregated data (3 iter.)

(i) Enlarged image of (b) (j) Enlarged image of (e) (k) Enlarged image of (f) (l) Enlarged image of (g) (m) Enlarged image of (h)

Fig. 12. Visual comparison of colorization results for a 'girl' image with (a) imprecise color scribbles: (b) Joint BF (JBF) [37] (50 iter.) with a range parameter $\sigma_R = 0.008$, a spatial parameter $\sigma_S = 10.0$, and 31×31 window, (c) GF [8] (30 iter.) with $\epsilon = 0.07^2$ and a radius $r = 10$, (d) RF of the DT [9] (50 iter.) with $\sigma_r = 0.03$ and $\sigma_s = 300$, (e) Levin et al. 2004 [13], (f) Ours (15) using the per-pixel data term (3 iter.), (g) Ours (15) using the per-pixel data term (10 iter.), (h) Ours (24) with the aggregated data term (3 iter.). All the results using our global smoother used $\sigma_c = 0.008$ and $\lambda = 50.0^2$. For computing the aggregated data term, the 2D RBF [10] was performed with a range parameter $\sigma_{dc} = \sigma_c$, and a spatial parameter $\alpha = 0.05$. Please refer to the 'neck', 'left hair' and 'bangs' regions of 'girl' image. These results are best viewed in their original resolution.



(a) Input (b) Using (15) after 3 iter.

(c) Using (15) after 10 iter. (d) Using (24) after 3 iter.

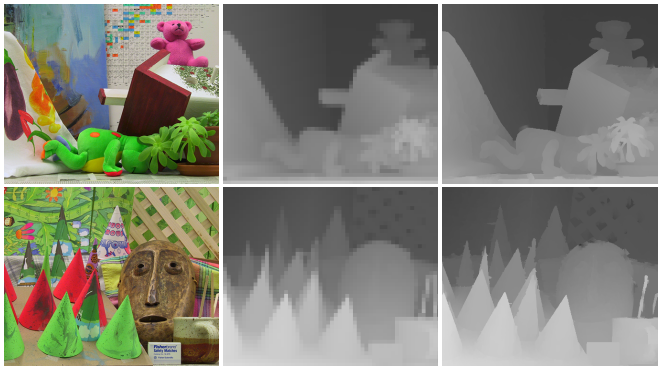
Fig. 13. Colorization results for a 'flower' images with (a) imprecise color scribbles: using (15) with the per-pixel data term after (b) 3 iterations and (c) 10 iterations, and (d) using (24) with the aggregated data term after 3 iterations. The parameters were set to $\sigma_c = 0.008$ and $\lambda = 50.0^2$. The parameters of the 2D RBF [10] were $\sigma_{dc} = \sigma_c$, and a spatial parameter $\alpha = 0.03$.

bed [39]; "Tsukuba", "Venus", "Teddy", and "Cone". Low-resolution depth maps are generated by downsampling the ground truth depth maps. The results upsampled by our global smoother are then compared with the ground truth depth maps for an objective evaluation. Similar to [22], the original depth values mapped into the color image coordinate only are used for the upsampling procedure, and thus the upsampling task becomes a sparse data interpolation. Specifically, the sparse depth input f is smoothed with the corresponding color image g by using (15).

Figure 14 shows the upsampled results for the Middlebury test images, when the upsampling ratio is 8 in each dimension. These results shows that the global smoother preserves sharp depth boundaries very well without producing halo artifacts. Table III shows the objective evaluation of the upsampling results. We compare our method with existing depth upsampling approaches; 2-D joint bilateral upsampling (2-D JBU) [37], 3-D JBU [40], and weighted mode filtering (WMF) [22]. The percentage (%) of bad matching pixels (where the absolute disparity error is greater than 1 pixel) was measured for 'all' (all pixels in the image) and 'disc' (the visible pixels near the occluded regions). Our method outperforms the existing approaches, particularly around the depth boundaries ('disc'). Regarding the runtime efficiency, we compare the WMF and our smoother only, since it was reported in [22] that the WMF runs faster than the 2-D JBU and the 3-D JBU. The runtimes of the WMF and our method are 0.48 and 0.02 seconds, respectively.

VII. Conclusions

This paper has presented an efficient edge-preserving smoothing method based on the WLS formulation, called *fast*



(a) Color image (b) Input depth (c) Upsampled depth

Fig. 14. Results of depth upsampling for "Teddy" and "Cone" images: (b) Input low-resolution depth maps downsampled with a factor of 8, (c) Upsampled results using our global smoother with $\sigma_c = 0.024$ and $\lambda = 30.0^2$. The upsampling ratio is 8 in each dimension. Please note that for better visualization of (b), the input depth maps are upsampled by a simple nearest neighbor (NN) method.

TABLE III
OBJECTIVE EVALUATION FOR DEPTH UPSAMPLING. WE MEASURED THE PERCENTAGE (%) OF BAD MATCHING PIXELS ON 'ALL' (ALL PIXELS IN THE IMAGE) AND 'DISC' (THE VISIBLE PIXELS NEAR THE OCCLUDED REGIONS) REGIONS. WE COMPARE OUR METHOD WITH 2-D JOINT BILATERAL UPSAMPLING (2-D JBU) [37], 3-D JBU [40], AND WEIGHTED MODE FILTERING (WMF) [22].

Algo.	*Tsukuba*		*Venus*		*Teddy*		*Cone*	
	all	disc	all	disc	all	disc	all	disc
Input	10.4	46.2	3.26	36.6	11.9	35.5	14.7	36.4
2-D JBU	9.04	40.4	2.04	22.1	14.0	37.6	14.7	34.8
3-D JBU	7.89	35.0	1.67	17.8	10.7	30.6	12.1	30.0
WMF	4.35	20.2	0.61	5.73	9.51	23.7	9.43	19.2
Ours	4.66	18.4	0.33	3.70	6.70	16.9	4.54	10.3

global smoother. The linear system with an inhomogeneous Laplacian matrix defined on e.g. a $\mathcal{N}_4$ (or $\mathcal{N}_8$) grid of a 2D image is decomposed as a sequence of 1D linear subsystems, enabling us to solve them very efficiently using a linear-time tridiagonal matrix algorithm. We showed through various experiments that our global smoother outperforms the state-of-the-art local filtering methods in terms of smoothing quality, yet with a very comparable runtime. Furthermore, our efficient and versatile computational tool was shown to be directly applicable to several advanced image editing tasks, which commonly employ a $\mathcal{N}_4$ (or $\mathcal{N}_8$) neighbor-based prior term. We demonstrated that in such applications, our solver is an efficient alternative to existing time-consuming large linear system solvers. Finally, our flexible formulation in defining data constraints has led to more robust image editing tools against imprecise inputs, while maintaining its runtime efficiency.

The proposed global smoother is feasible to many other computer vision and computer graphics applications as well. For instance, it was noted in [16] that in the gradient-domain smoothing frameworks, 95% of the total runtime is spent to optimize a least-square function defined with $\mathcal{N}_4$ neighbors, indicating that many gradient domain processing tasks can be significantly accelerated using our method. We also believe that our efficient, high-quality, and flexible computational tool may drive many (new) applications, where the heavy computational cost is a main bottleneck and/or the capability of robustly handling imprecise inputs in an efficient manner is required.

(a) Input (b) $T = 10$ iter. (c) $T = 1100$ iter.

Fig. 15. Limitation of the proposed global smoother in sparse data interpolation (colorization): When an input data is extremely sparse as in (a) (see red and yellow scribbles), a huge number of iterations is needed for ensuring efficient data propagation. Here, λ_t is fixed during iterations ($\lambda_t = \lambda$ for $t = 1, \cdots, T$), not being varied in a form of (12), so that the amount of spatial smoothing is adjusted depending on the number of iterations T. It should be noted that this kind of problem is common in all existing smoothing based approaches.

A. Limitations

Like other separable algorithms [9], our smoother is not rotationally invariant. In fact, the original WLS formulation is also not rotationally invariant, because the smoothness regularization is separately enforced for axis-aligned directions [15]. Also, as most sparse matrix solvers, our smoother is applicable only when the neighbors are defined under a specific smoothness constraint (e.g. $\mathcal{N}_4$ or $\mathcal{N}_8$ grids in case of a 2D signal). This design choice may also bring about problems in terms of data propagation. When an input data is extremely sparse in case of the sparse data interpolation, a huge number of iterations is required for ensuring sufficient data propagation. Figure 15 shows results of the image colorization, when user color scribbles are very sparse. Here, λ_t is fixed during iterations ($\lambda_t = \lambda$ for $t = 1, \cdots, T$), not being varied in a form of (12), so that the amount of spatial smoothing is adjusted depending on the number of iterations T. Applying $T = 1100$ iterations produces the satisfactory colorized result in Figure 15 (c). Actually, this limitation is generally common in all existing smoothing based approaches.

APPENDIX I
DATA PROPAGATION OF LOCAL EP FILTER

For a 1D input signal f_x^h, the forward recursion used in the RF of the DT [9] is performed as follows:

$$\bar{f}_x^h = (1-\alpha^{d_x}) f_x^h + \alpha^{d_x} \bar{f}_{x-1}^h, \qquad (x = 1, \cdots, W-1), \quad (25)$$

with a boundary condition $\bar{f}_0^h = f_0^h$. $\alpha \in [0, 1]$ is a spatial smoothing parameter, and α^{d_x} is a feedback coefficient controlling the amount of edge-preserving smoothing. $\bar{f}^h$ is an intermediate output, which will be used as an input in the backward recursion. d_x is defined as a distance between two samples f_x^h and f_{x-1}^h on the transformed domain [9], and it is computed as

$$d_x = 1 + \frac{\sigma_s}{\sigma_r} \| f_x^h - f_{x-1}^h \|, \quad (26)$$

where σ_s and σ_r are spatial and range parameters used in the DT [9]. They also set $\alpha = \exp\{-\sqrt{2}/\sigma_s\}$, so the feedback coefficient α^{d_x} is re-written as

$$\begin{aligned}\alpha^{d_x} &= \exp(-\sqrt{2}/\sigma_s)^{1+(\sigma_s/\sigma_r)\|f_x^h - f_{x-1}^h\|} \\ &= \alpha\exp(-\sqrt{2}\|f_x^h - f_{x-1}^h\|/\sigma_r) = \alpha w_{x,x-1}, \end{aligned} \quad (27)$$

where $w_{x,x-1} = \exp(-\sqrt{2}\|f_x^h - f_{x-1}^h\|/\sigma_r)$. Then, (25) is expressed as follows:

$$\bar{f}_x^h = (1-\alpha w_{x,x-1})f_x^h + \alpha w_{x,x-1}\bar{f}_{x-1}^h, \quad (x = 1, \cdots, W-1), \quad (28)$$

Similarly, the backward recursion of the RF is induced as

$$u_x^h = (1-\alpha w_{x,x+1})\bar{f}_x^h + \alpha w_{x,x+1}u_{x+1}^h, \quad (x = W-2, \cdots, 0), \quad (29)$$

with a boundary condition $u_{W-1}^h = \bar{f}_{W-1}^h$. The backward recursion uses the intermediate output $\bar{f}^h$ as an input, and computes the final output u^h. Note that the same recursion is also used in [23], although derived from different considerations.

The RBF also employs similar forward and backward recursions [10]:

$$\begin{aligned}\bar{f}_x^h &= (1-\alpha)f_x^h + \alpha w_{x,x-1}\bar{f}_{x-1}^h, \qquad (x = 1, \cdots, W-1), \\ \bar{f}_0^h &= (1-\alpha)f_0^h, \end{aligned} \quad (30)$$

$$\begin{aligned}u_x^h &= (1-\alpha)\bar{f}_x^h + \alpha w_{x,x+1}u_{x+1}^h, \qquad (x = W-2, \cdots, 0), \\ u_{W-1}^h &= (1-\alpha)\bar{f}_{W-1}^h. \end{aligned} \quad (31)$$

These recursive filters adaptively (and locally) updates the filtered output u^h using the weighting function $w_{x,x-1}$ (or $w_{x,x+1}$) through the forward (or backward) pass. Namely, the amount of smoothing is determined by considering only neighboring pixels at $x-1$ or $x+1$. Our global smoother performs the smoothing through similar forward and backward recursions, but it yields an exact minimum for an 1D energy function, so outperforms these local filters (e.g. no halo artifacts). Similar analysis on smoothing quality was also presented in [25], i.e. WLS-based method vs. local filters.

REFERENCES

[1] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 7, pp. 629–639, 1990.

[2] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *IEEE Int. Conf. on Computer Vision*, 1998, pp. 839–846.

[3] F. Durand and J. Dorsey, "Fast bilateral filtering for the display of high-dynamic-range images," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 257–266, 2002.

[4] F. Porikli, "Constant time O(1) bilateral filtering," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2008.

[5] Q. Yang, K.-H. Tan, and N. Ahuja, "Real-time O(1) bilateral filtering," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2009, pp. 557–564.

[6] S. Paris and F. Durand, "A fast approximation of the bilateral filter using a signal processing approach," *International Journal of Computer Vision*, vol. 81, no. 1, pp. 24–52, 2009.

[7] A. Adams, J. Baek, and M. A. Davis, "Fast high-dimensional filtering using the permutohedral lattice," *Comput. Graph. Forum*, vol. 29, no. 2, pp. 753–762, 2010.

[8] K. He, J. Sun, and X. Tang, "Guided image filtering," in *European Conf. on Computer Vision*, 2010, pp. 1–14.

[9] E. S. L. Gastal and M. M. Oliveira, "Domain transform for edge-aware image and video processing," *ACM Trans. Graph.*, vol. 30, no. 4, p. 69, 2011.

[10] Q. Yang, "Recursive bilateral filtering," in *European Conf. on Computer Vision*, 2012, pp. 399–413.

[11] J. Lu, K. Shi, D. Min, L. Lin, and M. N. Do, "Cross-based local multipoint filtering," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2012, pp. 430–437.

[12] R. Fattal, "Edge-avoiding wavelets and their applications," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 1–10, 2009.

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

[14] D. Lischinski, Z. Farbman, M. Uyttendaele, and R. Szeliski, "Interactive local adjustment of tonal values," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 646–653, 2006.

[15] Z. Farbman, R. Fattal, D. Lischinski, and R. Szeliski, "Edge-preserving decompositions for multi-scale tone and detail manipulation," *ACM Trans. Graph.*, vol. 27, no. 3, 2008.

[16] P. Bhat, C. L. Zitnick, M. F. Cohen, and B. Curless, "Gradientshop: A gradient-domain optimization framework for image and video filtering," *ACM Trans. Graph.*, vol. 29, no. 2, 2010.

[17] L. Xu, Q. Yan, Y. Xia, and J. Jia, "Structure extraction from texture via relative total variation," *ACM Trans. Graph.*, vol. 31, no. 6, 2012.

[18] I. Koutis, G. L. Miller, and D. Tolliver, "Combinatorial preconditioners and multilevel solvers for problems in computer vision and image processing," *Computer Vision and Image Understanding*, vol. 115, no. 12, pp. 1638–1646, 2011.

[19] D. Krishnan, R. Fattal, and R. Szeliski, "Efficient preconditioning of laplacian matrices for computer graphics," *ACM Trans. Graph.*, 2013.

[20] G. H. Golub and C. F. Van Loan, *Matrix computations (3rd ed.)*. Baltimore, MD, USA: Johns Hopkins University Press, 1996.

[21] X. An and F. Pellacini, "AppProp: all-pairs appearance-space edit propagation," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 40:1–40:9, 2008.

[22] D. Min, J. Lu, and M. N. Do, "Depth video enhancement based on weighted mode filtering," *IEEE Trans. on Image Processing*, vol. 21, no. 3, pp. 1176–1190, 2012.

[23] P. Thevenaz, D. Sage, and M. Unser, "Bi-exponential edge-preserving smoother," *IEEE Trans. on Image Processing*, vol. 21, no. 9, pp. 3924–3936, 2012.

[24] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[25] K. He, J. Sun, and X. Tang, "Guided image filtering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 6, pp. 1397–1409, 2013.

[26] J. Chen, S. Paris, and F. Durand, "Real-time edge-aware image processing with the bilateral grid," *ACM Trans. Graph.*, vol. 26, no. 3, p. 103, 2007.

[27] P. Perez, M. Gangnet, and A. Blake, "Poisson image editing," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 313–318, 2003.

[28] M. Elad, "On the origin of the bilateral filter and ways to improve it," *IEEE Trans. on Image Processing*, vol. 11, no. 10, pp. 1141–1151, 2002.

[29] L. Grady, "Random walks for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 11, pp. 1768–1783, 2006.

[30] T. Q. Pham and L. J. van Vliet, "Separable bilateral filtering for fast video preprocessing," in *IEEE Int. Conf. on Multimedia and Expo*, 2005, pp. 454–457.

[31] J. Weickert, B. M. ter Haar Romeny, and M. A. Viergever, "Efficient and reliable schemes for nonlinear diffusion filtering," *IEEE Trans. on Image Processing*, vol. 7, no. 3, pp. 398–410, 1998.

[32] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, 3rd ed. Prentice Hall Press, 2009.

[33] T. H. Kim, K. M. Lee, and S. U. Lee, "Generative image segmentation using random walks with restart," in *European Conf. on Computer Vision*, 2008, pp. 264–275.

[34] M. Lang, O. Wang, T. Aydin, A. Smolic, and M. Gross, "Practical temporal consistency for image-based graphics applications," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 34:1–34:8, Jul. 2012.

[35] R. Chartrand and W. Yin, "Iteratively reweighted algorithms for compressive sensing," in *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, 2008, pp. 3869–3872.

[36] L. Pizarro, P. Mrázek, S. Didas, S. Grewenig, and J. Weickert, "Generalised nonlocal image smoothing," *Int. Journal of Computer Vision*, vol. 90, no. 1, pp. 62–87, 2010.

[37] J. Kopf, M. F. Cohen, D. Lischinski, and M. Uyttendaele, "Joint bilateral upsampling," *ACM Trans. Graph.*, vol. 26, no. 3, 2007.

[38] L. Xu, C. Lu, Y. Xu, and J. Jia, "Image smoothing via l0 gradient minimization," *ACM Trans. Graph.*, vol. 30, no. 6, pp. 174:1–174:12, Dec. 2011.
[39] http://vision.middlebury.edu/stereo.
[40] Q. Yang, R. Yang, J. Davis, and D. Nistér, "Spatial-depth super resolution for range images," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2007.

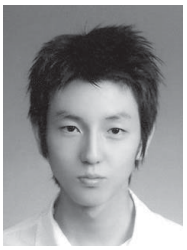
Dongbo Min (M'09) received the B.S., M.S. and Ph.D. degrees from School of Electrical and Electronic Engineering at Yonsei University, in 2003, 2005 and 2009, respectively. From 2009 to 2010, He worked with Mitsubishi Electric Research Laboratories (MERL) as a post-doctoral researcher, where he developed a prototype of 3D video system (3DTV). Since July 2010, he has been working with Advanced Digital Sciences Center (ADSC) in Singapore, which was jointly founded by University of Illinois at Urbana-Champaign (UIUC) and the Agency for Science, Technology and Research (A*STAR), a Singapore government agency. His research interests include computer vision, 2D/3D video processing, computational photography, augmented reality, and continuous/discrete optimization.

Sunghwan Choi (S'10) received the B.S. degree in electronic engineering and avionics from Korea Aerospace University, Gyeonggi-do, Korea, in 2009. He is currently pursuing the joint M.S. and Ph.D. degrees in electrical and electronic engineering with Yonsei University, Seoul, Korea. His current research interests include 3D image and video processing, computer vision, computational aspects of human vision, and image-based modeling and rendering.

Jiangbo Lu (M'09) received the B.S. and M.S. degrees in electrical engineering from Zhejiang University, Hangzhou, China, in 2000 and 2003, respectively, and the Ph.D. degree in electrical engineering, Katholieke Universiteit Leuven, Leuven, Belgium, in 2009.

He was with VIA-S3 Graphics, Shanghai, China, from 2003 to 2004, as a Graphics Processing Unit Architecture Design Engineer. In 2002 and 2005, he conducted visiting research at Microsoft Research Asia, Beijing, China. Since 2004, he has been with the Multimedia Group, Interuniversity Microelectronics Center, Leuven, Belgium, as a Ph.D. Researcher. Since 2009, he has been with the Advanced Digital Sciences Center, Singapore, which is a joint research center between the University of Illinois at Urbana-Champaign, Champaign, IL, USA, and the Agency for Science, Technology and Research, Singapore, where he is leading a few research projects. His research interests include computer vision, visual computing, image processing, video communication, interactive multimedia applications and systems, and efficient algorithms for various architectures.

Dr. Lu is an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY. He was a recipient of the 2012 TCSVT Best Associate Editor Award.



Bumsub Ham (M'13) received the B.S. and Ph.D degrees from School of Electrical and Electronic Engineering at Yonsei University in Seoul, Korea in 2008 and 2013, respectively. He is now a postdoctoral research fellow in WILLOW team at INRIA / Ecole Normale Superieure. He is working with Prof. Jean Ponce on computer vision and machine learning. He was the recipient of the Honor Prize in 17th Samsung Human-Tech Prize in 2011 and the Grand Prize in Qualcomm Innovation Fellowship in 2012. His current research interests include variational methods and geometric partial differential equations, both in theory and applications in computer vision and image processing, particularly regularization, stereo vision, super-resolution, and HDR imaging.

Kwanghoon Sohn (M'92-SM'12) received the B.E. degree in electronic engineering from Yonsei University, Seoul, Korea, in 1983, the MSEE degree in electrical engineering from University of Minnesota in 1985, and the Ph.D. degree in electrical and computer engineering from North Carolina State University in 1992. He was employed as a senior member of the research staff in the Satellite Communication Division at Electronics and Telecommunications Research Institute, Daejeon, Korea, from 1992 to 1993 and as a postdoctoral fellow at the MRI Center in the Medical School of Georgetown University in 1994. He was a visiting professor at Nanyang Technological University from 2002 to 2003. He is currently a professor in the School of Electrical and Electronic Engineering at Yonsei University. His research interests include three-dimensional image processing, computer vision and image communication. Dr. Sohn is a senior member of IEEE and a member of SPIE.

Minh N. Do (M'01-SM'07-F'14) was born in Vietnam in 1974. He received the B.Eng. degree in computer engineering from the University of Canberra, Canberra, ACT, Australia, in 1997, and the Dr. Sci. degree in communication systems from the Swiss Federal Institute of Technology Lausanne, Lausanne, Switzerland, in 2001.

He has been on the faculty with the University of Illinois at Urbana-Champaign, Champaign, IL, USA, since 2002, where he is currently a Professor with the Department of Electrical and Computer Engineering, and hold joint appointments with the Coordinated Science Laboratory, the Beckman Institute for Advanced Science and Technology, and the Department of Bioengineering. His research interests include image and multidimensional signal processing, wavelets and multiscale geometric analysis, computational imaging, augmented reality, and visual information representation.

Prof. Do was a recipient of the Silver Medal from the 32nd International Mathematical Olympiad in 1991, the University Medal from the University of Canberra in 1997, the Doctorate Award from the EPFL in 2001, the CAREER Award from the National Science Foundation in 2003, and the Young Author Best Paper Award from the IEEE in 2008. He was named a Beckman Fellow at the Center for Advanced Study, UIUC, in 2006, and received the Xerox Award for Faculty Research from the College of Engineering, UIUC, in 2007. He was a member of the IEEE Signal Processing Theory and Methods Technical Committee, the Image, Video, and Multidimensional Signal Processing Technical Committee, and an Associate Editor of the IEEE TRANSACTIONS ON IMAGE PROCESSING.