Fast Multi-Layer Laplacian Enhancement

Hossein Talebi and Peyman Milanfar

***Abstract*— A novel, fast and practical way of enhancing images is introduced in this paper. Our approach builds on Laplacian operators of well-known edge-aware kernels, such as bilateral and nonlocal means, and extends these filter's capabilities to perform more effective and fast image smoothing, sharpening and tone manipulation. We propose an approximation of the Laplacian, which does not require normalization of the kernel weights. Multiple Laplacians of the affinity weights endow our method with progressive detail decomposition of the input image from fine to coarse scale. These image components are blended by a structure mask, which avoids noise/artifact magnification or detail loss in the output image. Contributions of the proposed method to existing image editing tools are: (1) Low computational and memory requirements, making it appropriate for mobile device implementations (e.g. as a finish step in a camera pipeline), (2) A range of filtering applications from detail enhancement to denoising with only a few control parameters, enabling the user to apply a combination of various (and even opposite) filtering effects.**

***Index Terms*— Image Enhancement, Image Editing, Image sharpening, Local Tone mapping, Image smoothing**

I. Introduction

Recently, edge-preserving image operators have been widely used in image enhancement applications. These filters allow separate processing of texture and piecewise smooth components of the image. Given that the main structure (edges) of the images are preserved by these edge-aware filters, applying an appropriate nonlinearity on the texture component results in local contrast enhancement and tonal adjustment [1]–[9]. However, when using these methods, the default assumption is that undesired perturbations, such as noise or compression artifacts are removed beforehand. In practical imaging scenarios, boosting a detail image layer can result in noise and artifact magnification, limiting applications of the existing detail enhancement algorithms. This issue is mitigated in our proposed method by employing a new blending strategy, which smoothes regions containing noise while sharpening significant image details. Our experiments demonstrate that the proposed method can be effective in improving details and local contrast of images, whilst efficiently handling mildly degraded cases (examples of the proposed method's applications are illustrated in Fig. 1). The existing relevant literature is reviewed next.

A. Related Work

Linear unsharp masking (UM) is perhaps the simplest algorithm for enhancing the edge and detail information of an image. Linear UM is a high-pass filter, which sharpens high frequency content of images, yet magnifies noise and produces undesirable distortions, such as halo artifacts. Polesel et al. [10] proposed adaptive unsharp mask to improve on the classic UM. This method measures local image gradient to adaptively apply the UM filter on details, and leave flat regions unchanged. Constrained unsharp mask [11] is another alternative, which combines a denoised and a sharpened version of the input image. Overall, the linear smoothing filter employed at the core of these methods can restrict their performances. Replacing the linear operator with a data-dependent (nonlinear) smoother diminishes this issue.

The Bilateral filter is possibly the most widely used edge-aware filter in image processing and computer graphics [12]. Similarity of neighbor pixels is measured by bilateral range filter, avoiding averaging across principal edges. Durand and Dorsey [13] exploit application of the bilateral filter in contrast reduction of high dynamic range images. A multi-scale implementation of the bilateral filter for progressive detail extraction is explored in [14]. Variations of bilateral filter can also be used for sharpening [15], creating cartoon effects [4], image editing [16] and abstraction [17]. Although bilateral filter outperforms linear smoothers, it still lacks robustness in some applications such as denoising.

Nonlocal means filter (NLM) works similarly to the bilateral kernel, except that photometric similarity of neighboring pixels is determined by measuring patch distances [18]–[20]. NLM weights better handle noise and other image distortions compared to bilateral kernel, yet offer competitive smoothing properties. Choudhury et al. [21] propose a multi-scale sharpness enhancement scheme based on NLM weights. A noise suppression step is performed first, and then different detail layers are extracted by recomputing NLM weights several times with various smoothing parameters. However, multiple realizations of the NLM filter impose a high computational complexity on this algorithm. NLM affinity weights were also used for various image editing tasks, such as tone manipulation and edit propagation in [22]. In that work, the global affinity matrix is approximated by its leading eigenvectors, enabling different filtering effects by polynomial mapping of the filter eigenvalues. More recently, differences of NLM smoothers are used to sharpen mildly blurred images [23]. Overall, global filtering parameters, filter weight computation and memory storage may limit application of these methods.

More nonlinear filters have been introduced in the past few years. A progressive coarsening operator based on a weighted least square optimization is proposed in [6]. Subr et al. [7] introduced a new image decomposition method by smoothing large image oscillations and preserving edges. Gaussian pyramids are also used for edge-aware filtering in local Laplacian framework [3], [8], where each detail layer is mapped by a specific function, resulting in tonal enhancement of the reconstructed image. The domain transform paradigm proposed by Gastal et al. [24] formulates the nonlinear image smoothing as

H. Talebi and P. Milanfar are with Google Inc., Mountain View, USA, Email: {htalebi, milanfar}@google.com.

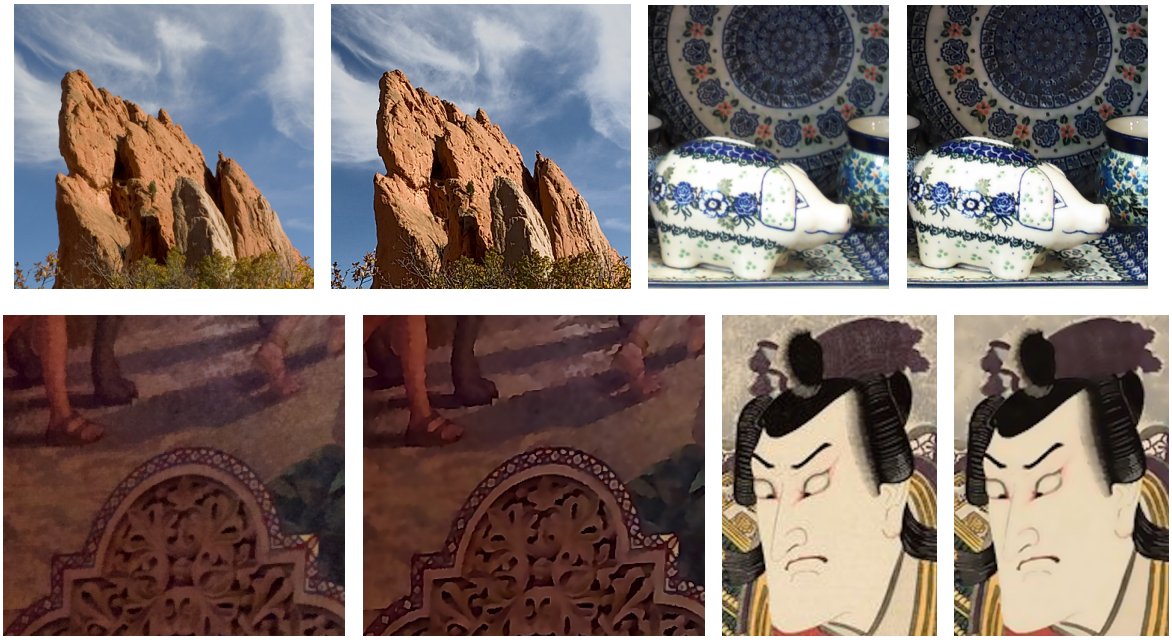


Fig. 1. Examples of detail enhancement using our method. Each image pair shows the input (left) and enhanced image (right). Top row: Application of our method for sharpening and local contrast enhancement. Bottom row: Application of our method for simultaneous artifact/noise removal and sharpening.

a few iterations of one dimensional filtering. The main edges are detected by image gradient and preserved in the filtering process. Another gradient-based smoother is introduced in [5] where image structure and texture are distinguished by means of local covariance. Edge preserving operators are also practically viable by guided image filtering [1], [2]. Image smoothing while constraining the number of non-zero gradients is another edge-aware filtering technique [25]. This approach removes low-amplitude structures by progressively reducing the number of non-zero gradients. Similar to the framework in [25], an L_1 energy minimization method for image smoothing is proposed in [26]. The energy cost includes local variations and global sparsity terms, and minimizing it results in flattening details. Our intention in this paper is not to introduce yet another nonlinear smoother. In fact, the base smoothing filter upon which the rest of the presented framework is constructed can be any of the existing filters mentioned above.

In addition to edge preserving filters, other contrast enhancement techniques based on the retinex theory [27] have been developed in the past few years [28], [29]. Retinex theory explains how humans can see colors consistently in spite of difference in light levels. Inspired by this theorem, several enhancement algorithms have been proposed recently [29]–[31]. Although these techniques are quite efficient and produce compelling results, noise magnification while brightening dark pixels remains challenging.

B. Contributions

The Laplacian operator of the local affinity matrix is at the core of our algorithm. The Laplacian operators can be computed for any smoothing operator, yet we develop our method based on the NLM kernel which is quite resilient to noise. Contributions of this work to the current image enhancement literature are:

- *A novel filtering approach using normalization-free filters*: Affinity weights are conventionally normalized and applied on the image to preserve the local brightness. In this paper we propose an efficient approximation of the normalized affinities to provide a computationally simplified *un-normalized* filtering paradigm.
- *Detail manipulation in the presence of mild image distortions*: Instead of applying noise/artifact suppression as a pre-filtering stage (which imposes extra complexity to the framework and may remove image details), our approach naturally handles these degradations. More specifically, fine detail boosting is replaced by smoothing when dealing with noisy regions. Fig. 2 demonstrates an example of simultaneous smoothing and sharpening using our proposed method.
- *Substantial complexity reduction of nonlinear multi-scale decomposition*: We propose a simple, yet effective way to compute the multi-scale detail decomposition by approximating affinity weights. Typical nonlinear multi-scale decomposition relies on successive computation of the filter weights on the input image. Given the exponential affinities of the NLM and bilateral kernels, we precompute the image-dependent filter weights only once and produce different versions of the filter by simple direct product of the weights. Significant speed up is observed by this strategy.

The proposed method allows real-time detail manipulation and enhancement such as examples shown in Fig. 3. The rest of the paper is organized as follows. In Section II, a detailed explanation of the proposed method is described. Next, in Section III, applications of our algorithm are exemplified. We

Fig. 2. Application of our method for simultaneous artifact/noise removal and sharpening. The input image is of size 1856×2528 and average running time for producing this result on an Intel Xeon CPU @ 3.5 GHz is about 0.2 seconds.

also provide details of our implementation along with running time comparisons. Finally, this paper is concluded in Section IV.

II. Proposed scheme

Our enhancement algorithm is broadly illustrated in Fig. 4. The input image is filtered by different affinity-based operators (built on the NLM weights [18]) to produce the image detail layers. Each layer is mapped by a nonlinear function to boost or suppress the associated detail and then, the manipulated layers are blended through a structure mask. The proposed scheme has parameters of the mapping functions as its filtering knobs, which control the filter's behavior by altering it from smoothing to sharpening and from tone enhancement to tone compression. Our processing is principally in the YUV domain, by filtering luma and leaving chroma unaltered. For completeness, we also illustrate our method for filtering RGB color channels separately (Fig. 3). In this section, first the affinity filters are reviewed and then the normalization-free filter weights are discussed. We elaborate on the details of the proposed filtering scheme, and finally our mapping functions and blending strategy are discussed.

A. Nonlinear edge-aware filters

The general construction of an edge-aware filter begins by specifying a symmetric positive semi-definite (PSD) kernel $k_{ij} \geq 0$ that measures the similarity, or affinity, between individual or groups of pixels. This affinity can be measured as a function of both the distance between the spatial variables (denoted by $\mathbf{x}$), but more importantly, also using the gray or color value (denoted by $\mathbf{y}$). While the results of this paper extend to any filter with a PSD kernel, some popular examples commonly used in the image processing, computer vision, and graphics literature are as follows:

a) Bilateral (BL) [12], [32]: This filter takes into account both the spatial *and* value distances between two pixels, generally in a separable fashion. For BL we have:

$$k_{ij} = \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{h_x}\right) \exp\left(\frac{-(y_i - y_j)^2}{h_y}\right) \qquad (1)$$

As seen in the overall exponent, the similarity metric here is a weighted Euclidean distance between the concatenated vectors $(\mathbf{x}_i, y_i)$ and $(\mathbf{x}_j, y_j)$.

b) Nonlocal Means (NLM) [18], [20]: The NLM kernel is a generalization of the bilateral kernel in which the value distance term is measured patch-wise instead of point-wise:

$$k_{ij} = \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{h_x}\right) \exp\left(\frac{-\|\mathbf{y}_i - \mathbf{y}_j\|^2}{h_y}\right), \qquad (2)$$

where $\mathbf{y}_i$ and $\mathbf{y}_j$ refer now to *subsets* of samples (i.e. patches) in $\mathbf{y}$.

These affinities are not used directly to filter the images, but instead in order to maintain the local average brightness, they are normalized so that the resulting weights pointing to each pixel sum to one. More specifically,

$$w_{ij} = \frac{k_{ij}}{\sum_{j=1}^{n} k_{ij}}, \qquad (3)$$

where each element of the filtered signal $\mathbf{z}$ is then given by

$$z_i = \sum_{j=1}^{n} w_{ij}\, y_j. \qquad (4)$$

(a) Input (b) Smoothed (c) Enhanced (effect 1) (d) Enhanced (effect 2)

Fig. 3. Example of our method applied for (b) detail smoothing, (c) detail enhancement in luma channel, and (d) detail enhancement in RGB channels. The input image is of size 700×458 and average running time for producing effect 1 is about 0.015 second.

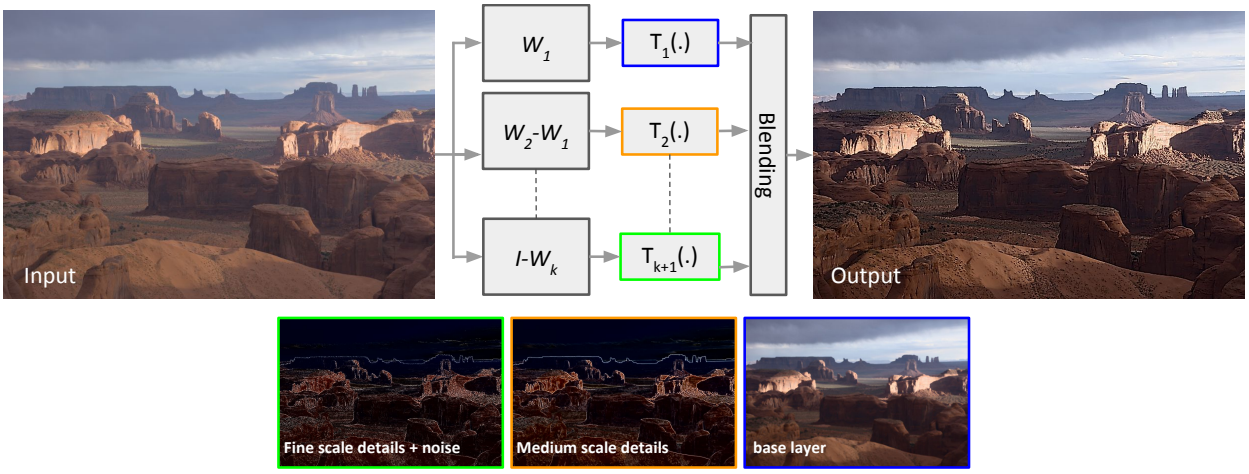


Fig. 4. The proposed pipeline: The input image is fed to multiple affinity based Laplacians obtained from NLM kernel ($\mathbf{W}_l$) to produce various detail layers. The filter operators are $\{\mathbf{W}_1, \mathbf{W}_2 - \mathbf{W}_1, ..., \mathbf{W}_k - \mathbf{W}_{k-1}, \mathbf{I} - \mathbf{W}_k\}$ and the smoothest image layer is obtained from $\mathbf{W}_1$ (Given the smoothing parameter of $\mathbf{W}_l$ as h_l, for every $1 \leq l < k$ we have $h_l > h_{l+1}$). Detail layers are mapped by nonlinear s-curves ($T_l(.)$) and blended by a structure mask to produce the enhanced image.

It is worth noting that the denominator in (3) can be computed by simply applying the filter (without normalization) to an image of all 1's.

In matrix notation, the collection of the weights used to produce the i-th output pixel is the vector $[w_{i1}, \cdots, w_{in}]$; and this can in turn be placed as the i-th row of a filter matrix $\mathbf{W}$ so that

$$\mathbf{z} = \mathbf{W}\mathbf{y}. \tag{5}$$

We note again that due to the normalization of the weights, the rows of the matrix $\mathbf{W}$ sum to one, That is, for each $1 \leq i \leq n$,

$$\sum_{j=1}^{n} w_{ij} = 1. \tag{6}$$

Viewed another way, the filter matrix $\mathbf{W}$ is a *normalized* version of the symmetric positive definite affinity matrix $\mathbf{K}$ constructed from the *un-normalized* affinities k_{ij}, $1 \leq i, j, \leq n$. As a result, $\mathbf{W}$ can be written as a product of two matrices

$$\mathbf{W} = \mathbf{D}^{-1}\mathbf{K}, \tag{7}$$

where $\mathbf{D}$ is a diagonal matrix with diagonal elements $[\mathbf{D}]_{ii} = \sum_{j=1}^{n} k_{ij} = d_i$.

B. Motivation

In our multiscale scheme, the filtered image $\mathbf{z}$ is expressed as a linear combination of k detail layers and one smooth layer:

$$\mathbf{z} = \beta_1 \, \mathbf{y}_{smooth} + \beta_2 \, \mathbf{y}_{detail_1} + \cdots + \beta_{k+1} \, \mathbf{y}_{detail_k} \tag{8}$$

where β_i denotes the boosting or shrinking factor associated with each layer. A nonlinear filter $\mathbf{W}$ can be used to realize this decomposition scheme as:

$$\begin{aligned} \mathbf{z} \;\; = \;\; & \beta_1 \mathbf{W}^k \mathbf{y} + \beta_2 (\mathbf{I} - \mathbf{W}) \mathbf{W}^{k-1} \mathbf{y} + \cdots \\ & + \beta_k (\mathbf{I} - \mathbf{W}) \mathbf{W} \mathbf{y} + \beta_{k+1} (\mathbf{I} - \mathbf{W}) \mathbf{y} \end{aligned} \tag{9}$$

in which $\mathbf{W}^k\mathbf{y}$ represents the smooth layer obtained from k diffusion iterations of $\mathbf{W}$ filter. The remaining terms consist of $k-1$ "band-pass" and one "high-pass" filters that decompose the input image into various detail layers (This image

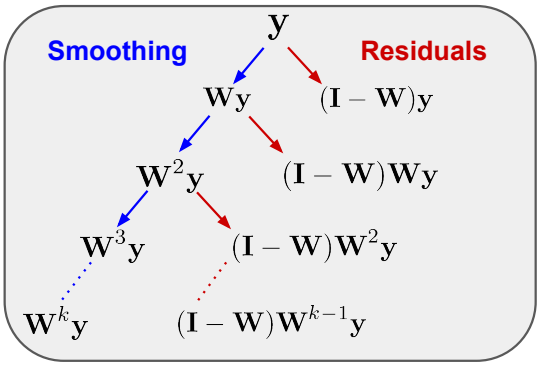


Fig. 5. The proposed image decomposition framework. Input image $\mathbf{y}$ is decomposed into smoothed $\mathbf{W}\mathbf{y}$ and residual $(\mathbf{I}-\mathbf{W})\mathbf{y}$ components. k iterations of this process on the smoothed image leads to k residual layers.

decomposition scheme is shown in Fig. 5). As shown in [22], the multiscale filtering in (9) corresponds to a polynomial mapping of the filter eigenvalues. Although this interpretation provides a flexible global framework for affinity based filtering in the spectral domain, complexity of the eigen-decomposition approximation remains relatively high.

Our current paper's motivation is to expedite the local computation of the diffusion filtering in (9). More explicitly, given the filter $\mathbf{W}$ of size $n \times n$, the expensive matrix multiplication of the diffusion process in computing $\mathbf{W}^k$ ($\mathcal{O}(kn^3)$) is replaced by recomputation of the affinity weights using a larger smoothing parameter h_k. As it will be addressed in this work, we only compute the affinity weights once and reuse the filter weights to efficiently reevaluate the affinity kernel weights. This leads to a quadratic filter computation complexity of $\mathcal{O}(kn^2)$. In what follows, the normalization-free filter is discussed first, and then, our multiscale enhancement scheme is described in more details.

C. The Normalization-free Filter

To avoid the normalization in (7), we will replace the filter $\mathbf{W}$ with an approximation $\widehat{\mathbf{W}}$ that only involves $\mathbf{D}$ rather than its inverse. More specifically,

$$\widehat{\mathbf{W}} = \mathbf{I} + \alpha(\mathbf{K} - \mathbf{D}). \tag{10}$$

Why is this a good idea? In what follows, we will motivate and derive this approximation from first principles, while also providing an analytically sound and numerically tractable choice for the scalar $\alpha > 0$ that gives the best approximation to $\mathbf{W}$ in the least-squares sense. Before doing so, it is worth noting some of the key properties and advantages of this approximate filter which are evident from the above expression (10).

- Regardless of the value of α, the rows of $\widehat{\mathbf{W}}$ always sum to one. That is, like its counterpart $\mathbf{W}$ constructed with $\mathbf{D}^{-1}$, the approximation $\widehat{\mathbf{W}}$, constructed with only $\mathbf{D}$, is automatically normalized. This can be easily seen by multiplying $\widehat{\mathbf{W}}$ on the right by a vector of ones, and observing that it returns the same vector back regardless of α.
- While the filter $\mathbf{W}$ is not symmetric due to the multiplicative normalization (see Eq. 7), the approximate $\widehat{\mathbf{W}}$ is always symmetric, again regardless of α. The advantages of having a symmetric filter matrix are many, as documented in the recent work [33].
- The PSD affinity matrix $\mathbf{K}$ is typically also non-negative *valued*, leading to filter weights in $\mathbf{W}$ which are also in turn non-negative valued. The elements in $\widehat{\mathbf{W}}$ however, can be negative valued due to the term $\mathbf{K}-\mathbf{D}$. This means that the behavior of the approximate filter may differ from its reference value, and must be carefully studied and controlled. We will do this next.

To derive the approximation, we first note that the standard filter can be written as:

$$\mathbf{W} = \mathbf{I} + \mathbf{D}^{-1}(\mathbf{K} - \mathbf{D}) \tag{11}$$

Comparing this form to the one presented earlier in (10), we note that the approximation is replacing the matrix inverse (on the right hand side) with a scalar multiple of the identity:

$$\mathbf{D}^{-1} \approx \alpha\mathbf{I} \tag{12}$$

As an illustration, an image containing the normalization terms d_i (which comprise the diagonal elements of $\mathbf{D}$) for the photo in Fig. 7, are shown in Fig. 6. The proposal, as we elaborate below, is to replace these many normalization constants in (11) with a single constant.

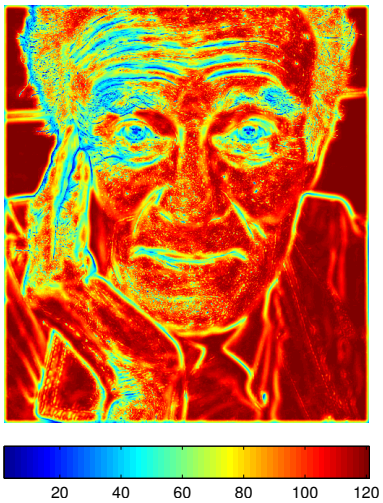


Fig. 6. Values of d_i for the old man photo. Large values shown in red indicate pixels that have many "nearest neighbors" in the metric implied by the bilateral kernel. Weights were computed over 11×11 windows (i.e. $m = 121$)

The justification for this approximation is a Taylor series in terms of $\mathbf{D}$ for the filter matrix. In particular, let's consider the first few terms in the series around a nominal $\mathbf{D}_0$:

$$\mathbf{D}^{-1}\mathbf{K} \approx \mathbf{I} + \mathbf{D}_0^{-1}(\mathbf{K} - \mathbf{D}) - \mathbf{D}_0^{-2}(\mathbf{D} - \mathbf{D}_0)(\mathbf{K} - \mathbf{D}) \tag{13}$$

The series expresses the filter as a perturbation of the identity, where the second and third terms are linear and quadratic in $\mathbf{D}$. For simplicity, we can elect to retain only the linear term, arriving at the approximation

$$\mathbf{D}^{-1}\mathbf{K} \approx \mathbf{I} + \mathbf{D}_0^{-1}(\mathbf{K} - \mathbf{D}). \tag{14}$$

Letting $\mathbf{D}_0 = \alpha^{-1}\mathbf{I}$, we arrive at the suggested approximation in (10).

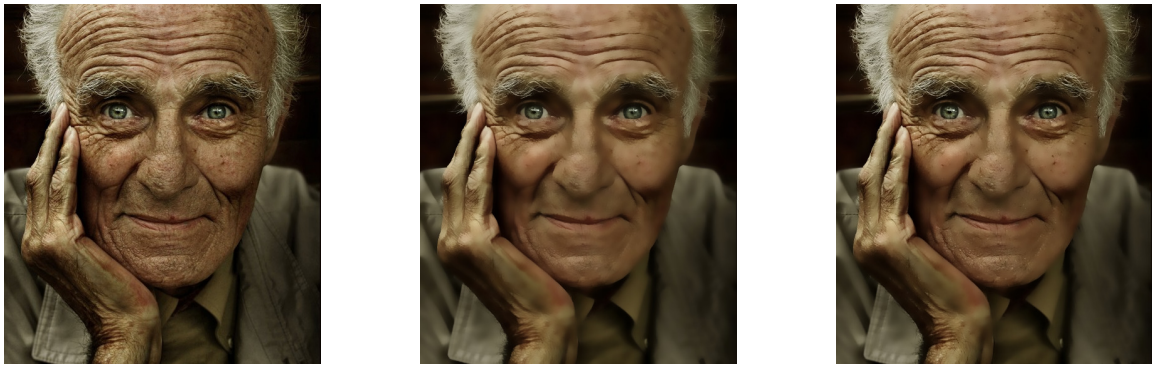


Fig. 7. (Left) Input $\mathbf{y}$; (Center) exact BL filter $\mathbf{Wz}$, and (right) approximate BL filter $\widehat{\mathbf{W}}\mathbf{z}$.

Choosing the best α: A direct approach to optimizing the value of the parameter α is to minimize the following cost function using the matrix *Frobenius* norm:

$$\min_{\alpha} \|\mathbf{W} - \widehat{\mathbf{W}}(\alpha)\|^2 \tag{15}$$

We can write the above difference as

$$J(\alpha) = \|\mathbf{W} - \widehat{\mathbf{W}}(\alpha)\|^2 = \|\mathbf{D}^{-1}\mathbf{K} - \mathbf{I} - \alpha(\mathbf{K} - \mathbf{D})\|^2 \tag{16}$$

This is a quadratic function in α. Upon differentiating and setting to zero, we are led to the global minimum solution:

$$\widehat{\alpha} = \frac{\text{tr}(\mathbf{K}\mathbf{D}^{-1}\mathbf{K}) - 2\text{tr}(\mathbf{K}) + \text{tr}(\mathbf{D})}{\text{tr}(\mathbf{K}^2) - 2\text{tr}(\mathbf{K}\mathbf{D}) + \text{tr}(\mathbf{D}^2)} \tag{17}$$

For sufficiently large m, where m is the size of the widnow over which filter weights are calculated, the terms $\text{tr}(\mathbf{D})$ and $\text{tr}(\mathbf{D}^2)$ dominate the numerator and the denominator, respectively. Hence,

$$\widehat{\alpha} \approx \frac{\text{tr}(\mathbf{D})}{\text{tr}(\mathbf{D}^2)} = \frac{s_1}{s_2}, \tag{18}$$

where

$$s_1 = \sum_{i=1}^{n} d_i, \quad \text{and} \quad s_2 = \sum_{i=1}^{n} d_i^2 \tag{19}$$

This ratio is in fact bounded as $\frac{1}{mn} \leq \frac{s_1}{s_2} \leq \frac{1}{d}$, which for large n justifies a further approximation:

$$\widehat{\alpha} \approx \frac{1}{\overline{d}} \tag{20}$$

where $\overline{d} = \text{mean}(d_i)$ (the upper bound comes from the arithmetic-geometric mean inequality [34]) . Effect of this approximation on local variance is addressed in Appendix A. Properties of the normalization-free filter are further discussed in [35] [1]. Next, our affinity-based multiscale image enhancement framework is explained.

[1]Note that our proposed enhancement scheme is not dependent on the normalization-free weights, yet, using this technique can further simplify our method and result in a speed up.

D. Proposed Filtering Scheme

Our proposed filtering scheme (Fig. 4) has the following form:

$$\begin{aligned} \mathbf{z} \;=\; & T_1\left(\mathbf{W}_1\mathbf{y}\right) + T_2\left((\mathbf{W}_2 - \mathbf{W}_1)\mathbf{y}\right) + \cdots \\ & + T_k\left((\mathbf{W}_k - \mathbf{W}_{k-1})\mathbf{y}\right) + T_{k+1}\left((\mathbf{I} - \mathbf{W}_k)\mathbf{y}\right) \end{aligned} \tag{21}$$

where $\mathbf{W}_l$ denotes the (normalized or normalization-free) filter weights with smoothing parameter h_l and $T_l(.)$ is a scalar point-wise mapping function applied on each layer. It is worth mentioning that the generic filtering scheme in (21) includes (9) as a special case where $T_l(t) = \beta_l t$ and $\mathbf{W}_l = \mathbf{W}^{k-l+1}$. Each term in (21) is a filter difference applied on the input image $\mathbf{y}$ and mapped through $T_l(.)$. The proposed filter consists of one high-pass term $(\mathbf{I} - \mathbf{W}_k)$, $k-1$ band-pass terms $(\mathbf{W}_{l+1} - \mathbf{W}_l)$ and one low-pass term $(\mathbf{W}_1\mathbf{y})$. Apparently the filtering behavior is determined by mapping functions $T_l(.)$ which can boost or suppress each signal layer.

An example of the proposed filter is shown in Fig. 8. The NLM affinity weights $\mathbf{W}_1$ and $\mathbf{W}_2$ are computed for the center pixel in the texture patch with different smoothing parameters. The output filter is obtained by linear mapping functions as $T_1(t) = T_3(t) = t$ and $T_2(t) = 5t$. As can be seen, the output filter is a band-pass filter with both negative and positive weights.

Fig. 9 illustrates application of the proposed filtering scheme in (21). First, the degraded input image is decomposed into smooth and detail layers. Then, each layer is mapped by a function $T_l(.)$, and finally, all the layers are blended using a structure mask to produce the output image. As can be seen, image details are recovered in the band-pass layer and blended into the smooth layer, while the compression artifacts are suppressed. In the following, the filtering steps in (21) are explained in more depth.

1) Laplacian Interpretation: Given a linear mapper as $T_l(t) = \beta_l t$, Eq. 21 can be rewritten as:

$$\mathbf{z} = \beta_1\mathbf{y} + (\beta_1 - \beta_2)\mathbf{L}_1\mathbf{y} + \cdots + (\beta_k - \beta_{k+1})\mathbf{L}_k\mathbf{y} \tag{22}$$

in which $\mathbf{L}_l$ represents the *random walk* Laplacian $\mathbf{W}_l - \mathbf{I}$ [36] . This basically is the input image added to a linear combination of the Laplacian-filtered images. Another interpretation of the proposed filter can be described by *un-normalized* graph Laplacian [36]. As shown in Sec. II-C, the normalized filter

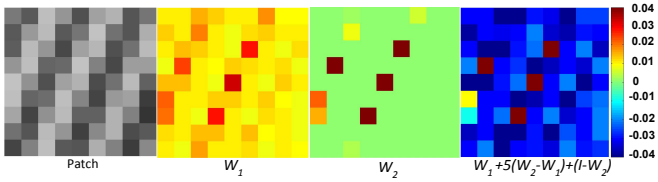


Fig. 8. An example of our proposed filter built on the two low-pass filters $\mathbf{W}_1$ and $\mathbf{W}_2$. The filter weights are shown for the center pixel of the 9×9 texture patch. $\mathbf{W}_1$ and $\mathbf{W}_2$ contain positive affinity weights, yet the combination of their Laplacians may produce negative filter values.

can be approximated as $\mathbf{W}_l \approx \mathbf{I} + \alpha_l(\mathbf{K}_l - \mathbf{D}_l) = \mathbf{I} + \alpha_l \mathcal{L}_l$. Then, Eq. 22 can be expressed in terms of un-normalized Laplacians as:

$$\mathbf{z} = \beta_1 \mathbf{y} + (\beta_1 - \beta_2)\alpha_1 \mathcal{L}_1 \mathbf{y} + \cdots + (\beta_k - \beta_{k+1})\alpha_k \mathcal{L}_k \mathbf{y} \quad (23)$$

where α_l are used in the normalization approximation. Next, we address the multiple computations of the affinity weights in (21) .

2) Multiple Affinity Weight Computation: The represented filtering scheme in (21) requires multiple computations of the edge-aware weights $\mathbf{W}_l$ for $l = 1, \cdots, k$. Ideally, an appropriately tuned filter based on (21) needs the affinity kernels to be evaluated by different smoothing parameters. This leads to a significant slow down of the algorithm's running time. This is due to the multiple evaluations of the $\exp(.)$ function. Our proposed solution to address this issue is an element-wise product of the kernel weights as:

$$\begin{aligned} \mathbf{W}_{l+1} \quad &= \quad \mathbf{D}_{l+1}^{-1}\mathbf{K}_{l+1} \quad (\text{or } \mathbf{W}_{l+1} = \alpha_{l+1}(\mathbf{K}_{l+1} - \mathbf{D}_{l+1})) \\ &\text{with} \quad \mathbf{K}_{l+1} = \mathbf{K}_l \odot \mathbf{K}_l \end{aligned} \quad (24)$$

where $\mathbf{W}_1$ is computed explicitly, l varies from 2 to $k-1$, and $\odot$ denotes the element-wise Hadamard product. Given the exponential affinities of BL or NLM, (24) leads to a set of filters defined by smoothing parameters as $h_{l+1} = h_l/2$ (both h_x and h_y in (1) and (2) will be divided by 2). In practice, we can start with a large h_1 and successively compute multiple versions of the filter using (24).

An example of the element-wise weight multiplication is shown in Fig. 10. Starting with $\mathbf{W}_1$, multiple filter weights from $\mathbf{W}_2$ to $\mathbf{W}_7$ are computed. It's worth pointing out that the variable bandwidths of these weights allow a more flexible evaluation of the proposed filtering scheme in (21).

3) Mapping Functions: The detail manipulation of the proposed algorithm strictly depends on the mapping functions $T_l(.)$. Having the input image decomposed into multiple detail layers, there are several ways to manipulate image texture and edges. The linear mapping discussed earlier is the simplest way of manipulating image details. Although the linear mapper has the interesting Laplacian interpretation, its main restrictive issue is the over-sharpening (-smoothing) of the detail content. In other words, a properly tuned detail mapping operator should treat details based on their respective local gradient magnitude. Recently, nonlinear detail manipulation has been successfully used for this task [6], [8]. Our choice is a nonlinear mapping function, specifically the *sigmoid* function:

$$T(t) = 1/(1 + \exp(-at)) \quad (25)$$

Our mapping operators derived from sigmoid function are demonstrated in Fig. 11 (appropriate shifting and scaling is applied on the sigmoid function). Application of the s-curve mapper on the detail and base layers leads to sharpness and tonal enhancement, respectively. On the other hand, the inverse s-curve can suppress details and compress the image contrast. Given the generic sigmoid function in (25), our mapping operator has two tuning parameters for each image layer. Parameter a determines the strength of the mapping operator. The other control parameter of the mapping function is its width (illustrated in the left and right plots of Fig. 11). The width parameter can prevent generation of halo and over-sharpening artifacts around large gradient edges. It also allows mid-tone contrast enhancement without suppressing details in dark or bright regions. Another possible mapping function is the combination of gamma correction with an s-curve for enhancing dark and bright details while boosting mid-tone details (shown in the middle plot of Fig. 11). It is worth mentioning that these mapping functions can be computed in advance as look up tables and used at run time.

Examples of applying our mapping functions are shown in Fig. 12. The detail layers of both enhanced images are fed to the same s-curves, and the base (smooth) layers are fed to s-curve (effect 1) and inverse s-curves (effect 2) mappers. As can be seen, details are enhanced in both cases, with effect 1 offering higher contrast and effect 2 representing relatively lower tonal range.

4) Structure Mask: Detail enhancement and artifact magnification are inseparable. Conventionally it is preferable to boost strong image structure with high signal-to-noise (SNR) and keep the noisy regions unaltered. This requires a mechanism to detect the image structure and somehow distinguish it from other areas. Edge detection provides a rough structure mask by detecting image irregularities. However, artifacts also are prone to be recognized as image details in a gradient map. One might argue that a pre-filtered image could possibly result in a more stable edge detection; yet, this approach could lead to extra complexity in the overall framework.

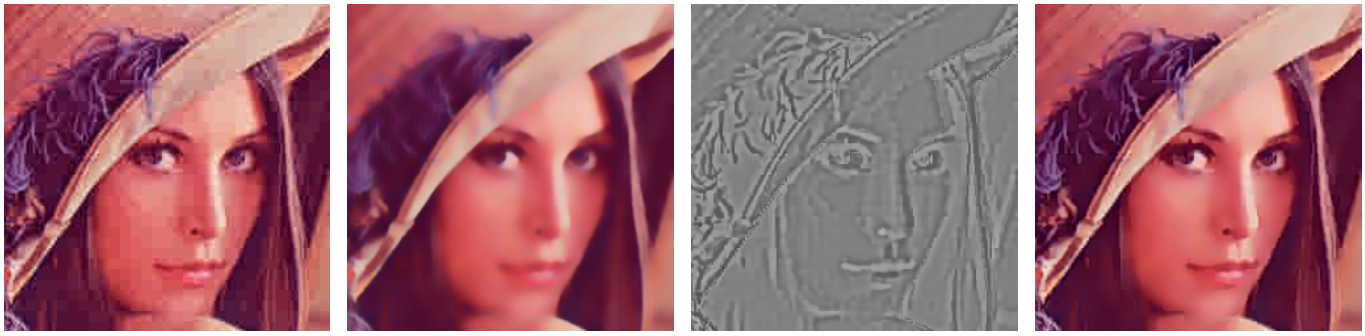
Interestingly, we have observed that the sum of the affinity degrees $[d_1, \cdots, d_p]$ (in a p-pixel neighborhood) conveys useful information about the image structure (see Fig.6). A pixel located on an edge or textured region has relatively low weight sum compared to a pixel in a flat area. A soft structure mask for i-th pixel can be defined as:

$$\mathbf{m}_i = 1 - d_i/p \quad (26)$$

where d_i denotes sum of the kernel weights associated with the i-th pixel and $\mathbf{m}_i$ takes values in $[0, 1]$. Examples of this structure mask are demonstrated in Fig. 13. Blending results using these masks are shown in Fig. 2 and Fig. 16. The detail layers of our image decomposition scheme are modulated by these masks to attenuate any possible noise and artifact boosting:

$$\begin{aligned} \mathbf{z} \quad = \quad & T_1(\mathbf{W}_1\mathbf{y}) + \mathbf{M}T_2((\mathbf{W}_2 - \mathbf{W}_1)\mathbf{y}) + \cdots \\ & + \mathbf{M}T_k((\mathbf{W}_k - \mathbf{W}_{k-1})\mathbf{y}) + \mathbf{M}T_{k+1}((\mathbf{I} - \mathbf{W}_k)\mathbf{y}) \end{aligned}$$

where $\mathbf{M}$ is a diagonal matrix representing the structure mask with values in [0, 1]. The detail layers of our image



(a) Input (b) $T_1\,(\mathbf{W}_1\mathbf{y})$ (c) $T_2\,((\mathbf{W}_2-\mathbf{W}_1)\mathbf{y})$ (d) $T_1\,(\mathbf{W}_1\mathbf{y})+T_2\,((\mathbf{W}_2-\mathbf{W}_1)\mathbf{y})$

Fig. 9. Removing compression artifacts using our proposed method in Eq. 21. (a) JPEG compressed image, (b) The base layer image smoothed by filter $\mathbf{W}_1$, (c) The luma detail layer obtained from the band-pass filter $\mathbf{W}_2-\mathbf{W}_1$, (d) Blended output $\mathbf{z}$. In this example, the baseline kernel is NLM [18], the mapping functions $T_l(.)$ are s-curves (see Sec. II-D3) and layer blending is based on a structure mask (see Sec. II-D4).

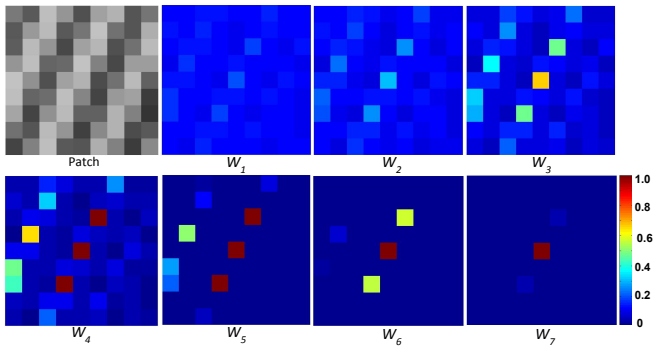


Fig. 10. NLM filters produced by element-wise multiplication as $\mathbf{W}_{l+1}=\mathbf{D}_{l+1}^{-1}\mathbf{K}_{l+1}$ where $\mathbf{K}_{l+1}=\mathbf{K}_l\odot\mathbf{K}_l$ and with $\mathbf{W}_1$ computed explicitly. The filter weights are shown for the center pixel of the 9×9 texture patch.

decomposition scheme are modulated by these masks to attenuate any possible noise and artifact boosting. Fig. 2 shows smoothing of the artifacts and sharpening of the details as a result of applying the structure mask. It is worth noting that the structure mask costs almost no additional computation, given that the kernel weights are already computed. Also, the structure mask is moderately robust to noise, because (1) it includes many summed weights, and (2) NLM kernel weights measure similarity between patches.

III. Experimental Results

Enhancement applications of the proposed filtering method are demonstrated through some examples in this section. NLM is our choice of affinity kernel without the spatial term given in (2), and filter weights are computed in a 5×5 neighborhood window. The patch size is 3×3, and the smoothing parameter h_y is set as 0.7 for pixel values in $[0,1]$. Three decomposition layers are selected based on Fig. 4 (i. e. $k=2$), meaning that NLM weights are computed once ($\mathbf{W}_1$), and used in the element-wise weight multiplication to form the second filter ($\mathbf{W}_2$).

There are three main applications for our method. First, detail smoothing (shown in Fig. 14 and Fig. 15); second, sharpening mildly blurred images (shown in Fig. 16 and Fig. 17), and finally, detail enhancement in noisy/artifacted images (shown in Fig. 18, Fig. 19 and Fig. 20). Mapping functions $T_l(.)$ are tuned specifically for each application to produce the best results.

Our multiscale decomposition allows smoothening fine details while preserving medium and coarse scale details. Our method is compared to the guided edge-aware filter [1] in Fig. 14 and Fig. 15. These results are obtained by removing the fine scale detail layer and mapping the medium scale layer (see Fig. 4) by an s-curve of width 0.2 and $a=10$. As can be seen, in contrast to the guided filter, our result is less blurry.

Out-of-focus blur is another common problem in mobile imaging. Objects typically lose sharpness and local contrast in a mildly blurred scene (see input photo in Fig. 16). Our filtering framework can effectively enhance these images (see output photo in Fig. 16). Parameters of the s-curve functions in each scale are: $a=20$ and width of 0.66 for the fine scale detail layer, $a=50$ and width of 0.33 for the medium scale detail layer, and $a=6$ with width of 0.75 for the base layer. Comparison of the proposed method with other techniques is demonstrated in Fig. 17. The adaptive unsharp masking [10] and Farbman's detail enhancement [6] tend to boost the image sharpness and noise together. Our result demonstrates better local contrast with no noise magnification or detail loss.

Noise is an inevitable part of any imaging pipeline. We also used our method for enhancing images corrupted by real noise and other artifacts (see input images in Fig. 9, Fig. 18, Fig. 19, Fig. 20, Fig. 22 and Fig. 21). To better handle noise in the input image, the fine scale detail is suppressed in our image decomposition and the base and medium scale layers are boosted. The mapping parameters to achieve this effect are: $a=10$ and width of 1 for the fine scale detail layer (inverse s-curve), $a=60$ and width of 0.45 for the medium scale detail layer (s-curve), and $a=5$ and width of 0.75 for the base layer (s-curve). Fig. 18-21 show examples of noisy/artifacted images enhanced by different methods. Overall, visual comparisons indicate superiority of the proposed algorithm when dealing with degraded images.

Our C++ implementation is tested on an Intel Xeon CPU @ 3.5 GHz with 32 MB memory. Complexity of the proposed algorithm is linearly dependent on the filter size. Running time of our method is reported for some test images in Table I. Examples shown in this paper are mostly based on 5×5 NLM

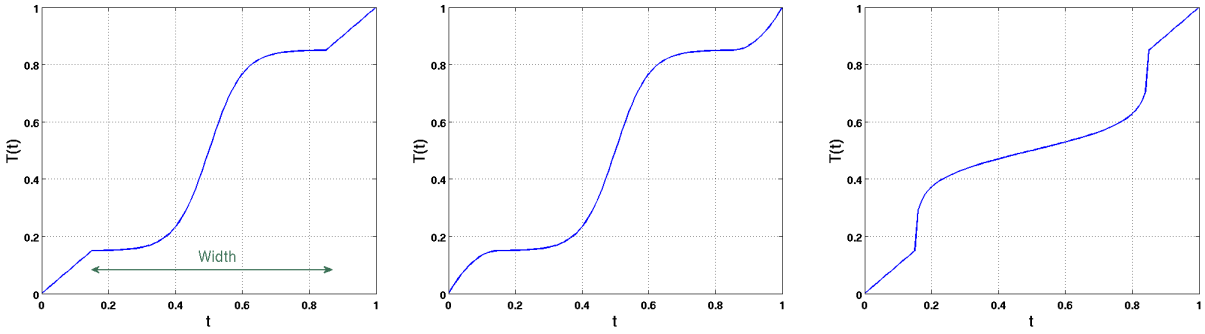


Fig. 11. Left: s-curve used for detail/tonal enhancement. Middle: s-curve combined with gamma correction for enhancing mid-tone contrast and boosting dark/bright details. Right: inverse s-curve used for smoothing and tone compression.

(a) Input (b) Enhanced (effect 1) (c) Enhanced (effect 2)

Fig. 12. Example of our method applied for detail enhancement. The same s-curve functions were applied on the detail layers for both output images. The base layer image of effect 1 and effect 2 are fed to s-curve and inverse s-curve, respectively. Effect 1 represents contrast enhancement and effect 2 shows tonal compression. The input image is of size 1028×926 and running time for producing the enhanced images is about 0.031 second.

Fig. 13. Structure masks used for blending of the detail layers. These masks are shown for images in Fig. 2 and Fig. 16.

filters, leading to an average speed of 21 MP/sec. Given available implementations of the other enhancement techniques, our method is significantly faster. For instance, processing an image of size 0.5 Mega pixel takes 0.03, 0.91, 3.2, 30.5, 12.7 seconds for [1], [31], [6], [10], and [11] , respectively. Our implementation takes less than 0.025 seconds to enhance the same image. We also tested our algorithm without weight normalization and weight re-computation approximations to measure the overall saved time. Our experiments suggest that these approximations lower the running time by 15-20%.

TABLE I
AVERAGE RUNNING TIME (SECONDS) OF THE PROPOSED ALGORITHM COMPUTED FOR NLM KERNEL OF DIFFERENT SIZES. SIZE OF THE TEST IMAGES ALONG WITH THE NEIGHBORHOOD WINDOW SIZES ARE SHOWN IN THE FIRST ROW AND COLUMN OF THE TABLE.

	0.4 MP	**1 MP**	**3 MP**	**12 MP**
3×3	0.014	0.019	0.034	0.143
5×5	0.022	0.045	0.105	0.575
7×7	0.040	0.075	0.223	1.363
9×9	0.078	0.152	0.473	2.623

IV. CONCLUSION

We introduced a new multiscale image enhancement algorithm to improve on the existing edge-aware filters. Our multiscale decomposition scheme provides a fast detail manipulation paradigm with a minor complexity added to the computation of the baseline kernel. Combination of the detail layers with a structure mask produces state-of-the-art image enhancement results, addressing shortcomings of the existing algorithms. This proposed work is implemented for NLM filter weights; however, it can be easily extended to other edge-aware filters.

(a) Input (b) [1] (c) Ours

Fig. 14. Edge-aware smoothing using our method compared to the result from guided image filtering [1]. For nearly the same running time budget (0.08 second), our method better flattens the fine details and avoids blurring the piecewise smooth output.

(a) Input (b) [1] (c) Ours

Fig. 15. Edge-aware smoothing using our method compared to the result from guided image filtering [1]. For nearly the same running time budget (0.06 second), our method better flattens the fine details and avoids blurring the piecewise smooth output.

APPENDIX A
EFFECT OF APPROXIMATION ON LOCAL VARIANCE

We expect that the approximate filter should affect the variance of the output pixels. Here we characterize this effect. Recall the pixel-wise expressions for the exact and approximate filter, respectively:

$$z_i = \sum_{j=1}^{n} w_{ij}\, y_j, \qquad \widehat{z}_i = \sum_{j=1}^{n} \widehat{w}_{ij}\, y_j \tag{27}$$

The variance in the output pixel in terms of the variance in the input pixel is given by the sum-squared of the filter weights. That is,

$$\text{var}(z_i) = \left(\sum_{j=1}^{n} w_{ij}^2 \right) \text{var}(y_i) = \nu_i \ \text{var}(y_i) \tag{28}$$

$$\text{var}(\widehat{z}_i) = \left(\sum_{j=1}^{n} \widehat{w}_{ij}^2 \right) \text{var}(y_i) = \widehat{\nu}_i \ \text{var}(y_i) \tag{29}$$

It is of interest to establish a relationship between the factors ν_i and $\widehat{\nu}_i$. We proceed as follows:

$$\begin{aligned} \widehat{\nu}_i &= \widehat{\mathbf{w}}_i^T \widehat{\mathbf{w}}_i \\ &= \left(\delta_i + \alpha^2 d_i^2 \left(\mathbf{w}_i - \delta_i\right)\right)^T \left(\delta_i + \alpha^2 d_i^2 \left(\mathbf{w}_i - \delta_i\right)\right) \\ &= \alpha^2 d_i^2 \nu_i + (\alpha^2 d_i^2 - 2\alpha(1+\alpha) d_i + 1 + 2\alpha) \end{aligned} \tag{30}$$

where δ_i is the shifted Dirac delta vector $[0, \cdots, 0, 1, 0, \cdots, 0]$, with subscript i indicating that the value 1 occurs in the i-th position. The two variance factors are linearly related when α is small:

$$\widehat{\nu}_i \approx \rho_i^2 \ \nu_i + (\rho_i - 1)^2 \tag{31}$$

where $\rho_i = \alpha d_i$. The contour plot in Fig. 23, shows the values of $\widehat{\nu}_i$ as a function of ρ_i and ν_i. Also, for the specific approximation pertaining to (18), we note that $d_i = \mathcal{O}(m)$ where m is the size of the window over which filter weights are calculated. For instance, in the case of Fig. 7, $m = 11 \times 11 = 121$. Given n pixels in the image, $\text{tr}(\mathbf{D}) = \mathcal{O}(mn)$. In the meantime,

(a) Input (b) Enhanced details

Fig. 16. Example of our method applied for detail enhancement. The input image is of size 1289×1029 and running time for producing the enhanced image is about 0.04 second.

(a) Input (b) [10] (c) [11] (d) [6] (e) Ours

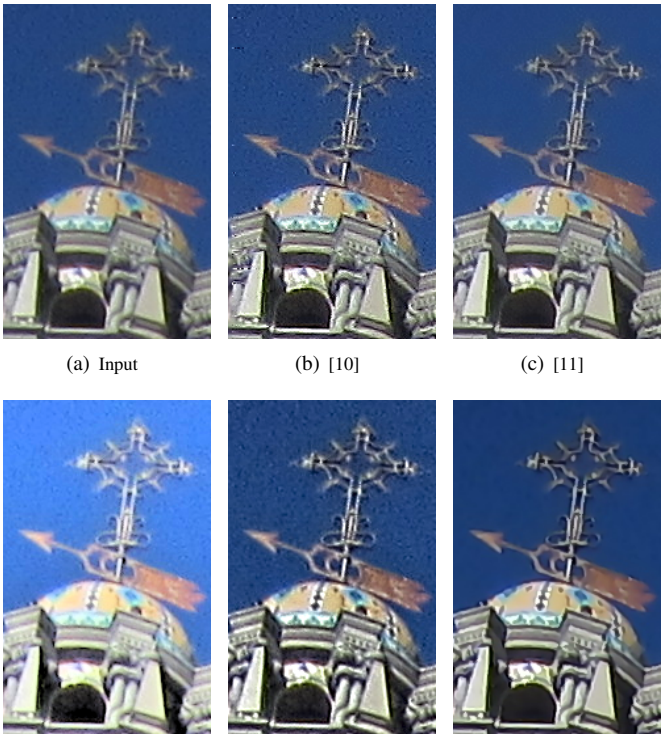
Fig. 17. Comparing existing detail enhancement methods with our proposed algorithm.

$\text{tr}(\mathbf{K}) = \mathcal{O}(n)$, $\text{tr}(\mathbf{K}\mathbf{D}^{-1}\mathbf{K}) = \mathcal{O}(n/m)$, $\text{tr}(\mathbf{K}^2) = \mathcal{O}(n^2)$, $\text{tr}(\mathbf{K}\mathbf{D}) = \mathcal{O}(mn^2)$, and $\text{tr}(\mathbf{D}^2) = \mathcal{O}(m^2n^2)$. So for sufficiently large m (typically larger than 5×5), the terms $\text{tr}(\mathbf{D})$ and $\text{tr}(\mathbf{D}^2)$ dominate the numerator and denominator as claimed.

REFERENCES

[1] K. He, J. Sun, and X. Tang, "Guided image filtering," *IEEE Trans. on Pattern Anal. Mach. Intell.*, vol. 35, no. 6, pp. 1397–1409, June 2013. 1, 2, 8, 9, 10, 13

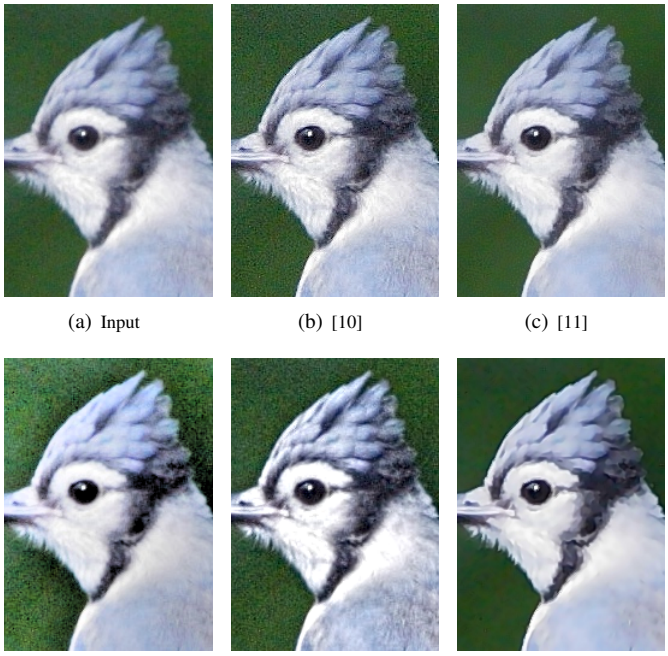
[2] Z. Li, J. Zheng, Z. Zhu, W. Yao, and S. Wu, "Weighted guided image filtering," *IEEE Trans. on Image Proc.*, vol. 24, no. 1, pp. 120–129, January 2015. 1, 2



(a) Input (b) [10] (c) [11]

(d) [31] (e) [6] (f) Ours

Fig. 18. Comparing existing detail enhancement methods with our proposed algorithm. Running times (in seconds): (b) 1.86, (c) 0.69, (d) 0.05, (e) 0.19, (f) 0.002.



(a) Input (b) [10] (c) [11]

(d) [31] (e) [6] (f) Ours

Fig. 19. Comparing existing detail enhancement methods with our proposed algorithm. Running times (in seconds): (b) 2.83, (c) 1.16, (d) 0.08, (e) 0.031, (f) 0.003.

[3] M. Aubry, S. Paris, S. W. Hasinoff, J. Kautz, and F. Durand, "Fast local Laplacian filters: Theory and applications," *ACM Trans. on Graph.*, vol. 33, no. 167, August 2014. 1

[4] H. Cho, H. Lee, H. Kang, and S. Lee, "Bilateral texture filtering," *ACM Trans. on Graph.*, vol. 33, no. 4, July 2014. 1

[5] L. Karacan, E. Erdem, and A. Erdem, "Structure-preserving image smoothing via region covariances," *ACM Trans. on Graph.*, vol. 32, no. 176, November 2013. 1, 2

[6] Z. Farbman, R. Fattal, D. Lischinski, and R. Szeliski, "Edge-preserving decompositions for multi-scale tone and detail manipulation," *ACM Trans. on Graph.*, vol. 27, no. 3, August 2008. 1, 7, 8, 9, 11, 12, 13

[7] K. Subr, C. Soler, and F. Durand, "Edge-preserving multiscale image decomposition based on local extrema," *ACM Trans. on Graph.*, vol. 28, no. 5, 2009. 1

[8] S. Paris, S. W. Hasinoff, and J. Kautz, "Local Laplacian filters: edge-aware image processing with a Laplacian pyramid," *ACM Trans. on Graph.*, vol. 30, no. 4, 2011. 1, 7

[9] Z. Farbman, R. Fattal, and D. Lischinski, "Diffusion maps for edge-aware image editing," *ACM Trans. on Graph.*, vol. 29, no. 6, December 2010. 1

[10] A. Polesel, G. Ramponi, and V. J. Mathews, "Image enhancement via adaptive unsharp masking," *IEEE Trans. on Image Proc.*, vol. 9, pp. 505–510, 2000. 1, 8, 9, 11, 12, 13

[11] R. C. Bilcu and M. Vehvilainen, "Constrained unsharp masking for image enhancement," *Proc. 3rd Int. Conf. on Image and Signal Proc.*, pp. 10–19, 2008. 1, 9, 11, 12, 13

[12] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," *ICCV, Bombay, India*, pp. 836–846, January 1998. 1, 3

[13] F. Durand and J. Dorsey, "Fast bilateral filtering for the display of high-dynamic-range images," *ACM Trans. on Graph.*, vol. 21, no. 3, July 2002. 1

[14] R. Fattal and S. Agrawala, M. Rusinkiewicz, "Multiscale shape and detail enhancement from multi-light image collections," *ACM Trans. on Graph.*, vol. 26, no. 3, 2007. 1

[15] B. Zhang and J. P. Allebach, "Adaptive bilateral filter for sharpness enhancement and noise removal," *IEEE Trans. on Image Proc.*, vol. 17, pp. 664–678, 2008. 1

[16] J. Chen, S. Paris, and F. Durand, "Real-time edge-aware image processing with the bilateral grid," *ACM Trans. on Graph.*, vol. 26, no. 3, 2007. 1

[17] H. Winnemöller, S. C. Olsen, and B. Gooch, "Real-time video abstraction," *ACM Trans. on Graph.*, vol. 25, no. 3, pp. 1221–1226, 2006. 1

[18] A. Buades, B. Coll, and J. M. Morel, "A review of image denoising algorithms, with a new one," *Multiscale Modeling and Simulation (SIAM interdisciplinary journal)*, vol. 4, no. 2, pp. 490–530, 2005. 1, 3, 8

[19] ——, "A non-local algorithm for image denoising," *IEEE Conf. on Computer Vision and Pattern Recog.*, pp. 60–65, 2005. 1

[20] C. Kervrann and J. Boulanger, "Optimal spatial adaptation for patch-based image denoising," *IEEE Trans. on Image Proc.*, vol. 15, no. 10, pp. 2866–2878, October 2006. 1, 3

[21] A. Choudhury and G. G. Medioni, "Perceptually motivated automatic sharpness enhancement using hierarchy of non-local means," *Poc. IEEE Int. Conf. Comput. Vis. Workshops*, pp. 730–737, Nov. 2011. 1

[22] H. Talebi and P. Milanfar, "Nonlocal image editing," *IEEE Trans. on Image Proc.*, vol. 23, no. 10, pp. 4460–4473, October 2014. 1, 5

[23] A. Kheradmand and P. Milanfar, "Nonlinear structure-aware image sharpening with difference of smoothing operators," *Frontiers in ICT, Computer Image Analysis*, vol. 2, no. 22, 2015. 1

[24] E. S. L. Gastal and M. M. Oliveira, "Domain transform for edge-aware image and video processing," *ACM Trans. on Graph.*, vol. 30, no. 4, pp. 69:1–69:12, 2011. 1

[25] L. Xu, C. Lu, Y. Xu, and J. Jia, "Image smoothing via L_0 gradient minimization," *ACM Trans. on Graph.*, vol. 30, no. 5, 2011. 2

[26] S. Bi, X. Han, and Y. Yu, "An L_1 image transform for edge-preserving smoothing and scene-level intrinsic decomposition," *ACM Trans. on Graph.*, vol. 34, no. 4, 2015. 2

[27] E. H. Land and J. J. McCann., "Lightness and retinex theory," *JOSA*, vol. 61, no. 1, pp. 1–11, 1971. 2

[28] D. J. Jobson, Z. Rahman, and G. A. Woodell, "A multiscale retinex for bridging the gap between color images and the human observation of scenes," *IEEE Trans. on Image Proc.*, vol. 6, no. 7, pp. 965–976, 1997. 2

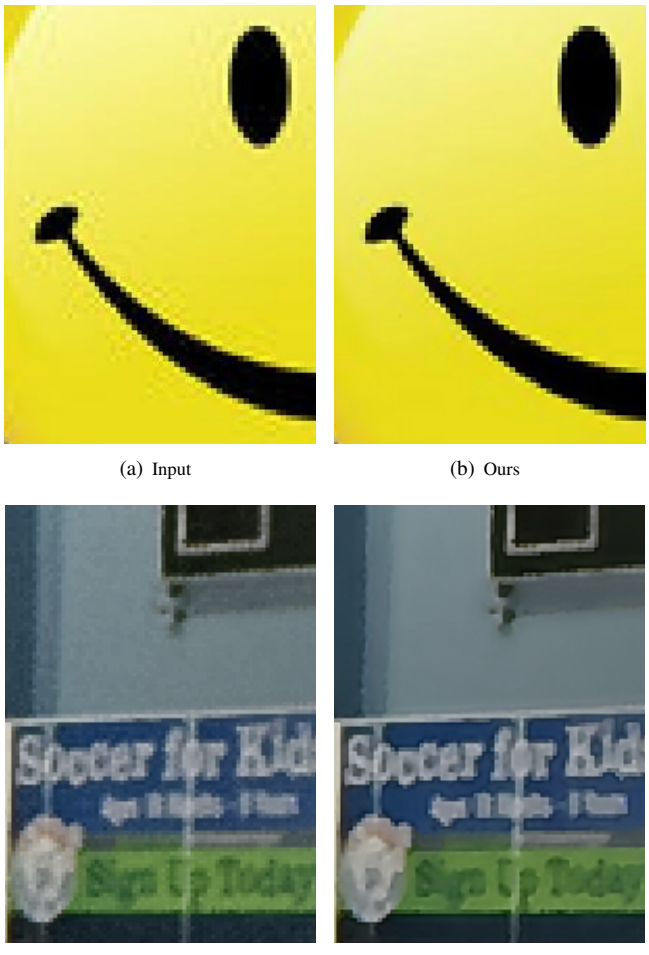
[29] R. Kimmel, M. Elad, D. Shaked, R. Keshet, and I. Sobel, "A variational framework for retinex," *International Journal of computer vision*, vol. 52, no. 1, pp. 7–23, 2003. 2

[30] J. M. Morel, A. B. Petro, and C. Sbert, "A PDE formalization of retinex theory," *IEEE Trans. on Image Proc.*, vol. 19, no. 11, pp. 2825–2837, 2010. 2

[31] A. Petro, C. Sbert, and J. M. Morel, "Multiscale retinex," *Image Processing On Line*, pp. 71–88, 2014. 2, 9, 12

(a) Input (b) [10] (c) [11] (d) [6] (e) Ours

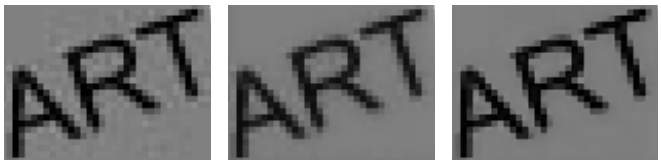
Fig. 20. Comparing existing detail enhancement methods with our proposed algorithm.



(a) Input (b) Ours

(c) Input (d) Ours

Fig. 21. Removing compression artifacts using our method. Filters are applied in RGB domain and are computed in an 11×11 neighborhood window.



(a) Input (b) [1] (c) Ours

Fig. 22. Removing compression artifacts using (b) guided filter [1] and (c) our method. Both filters are applied in RGB domain and are computed in an 11×11 neighborhood window. For the same running time budget, our result is superior to [1].

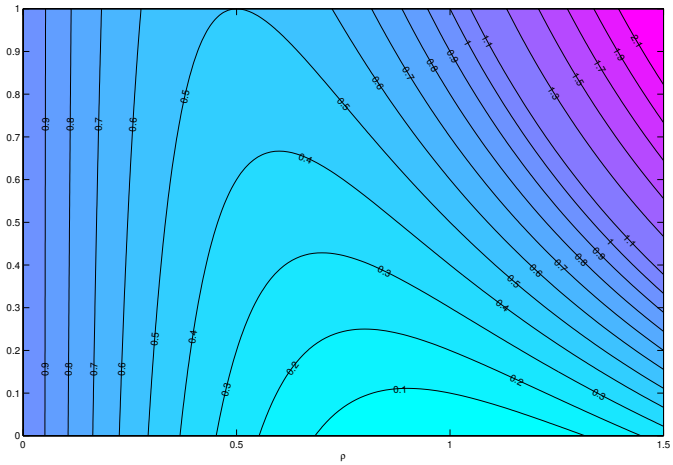


Fig. 23. Values of $\widehat{\nu}^2$ as a function of ρ (horizontal axis) and ν^2 (vertical axis)

[32] M. Elad, "On the origin of the bilateral filter and ways to improve it," *IEEE Trans. on Image Proc.*, vol. 11, no. 10, pp. 1141–1150, October 2002. 3

[33] P. Milanfar, "Symmetrizing smoothing filters," *SIAM Journal on Imaging Sciences*, vol. 6, no. 1, pp. 263–284, 2013. 5

[34] J. M. Steele, *THE CAUCHY–SCHWARZ MASTER CLASS*. Cambride University Press, 2004. 6

[35] P. Milanfar and H. Talebi, "A new class of image filters without normalization," *to appear in ICIP*, 2016. 6

[36] P. Milanfar, "A tour of modern image filtering," *IEEE Signal Proc. Magazine*, vol. 30, no. 1, pp. 106–128, 2013. 6