Deep Edge-Aware Filters

Li Xu XULI@SENSETIME.COM
Jimmy SJ. Ren RENSIJIE@SENSETIME.COM
Qiong Yan YANQIONG@SENSETIME.COM
SenseTime Group Limited

Renjie Liao RJLIAO@CSE.CUHK.EDU.HK
Jiaya Jia LEOJIA@CSE.CUHK.EDU.HK
The Chinese University of Hong Kong

Abstract

There are many edge-aware filters varying in their construction forms and filtering properties. It seems impossible to uniformly represent and accelerate them in a single framework. We made the attempt to learn a big and important family of edge-aware operators from data. Our method is based on a deep convolutional neural network with a gradient domain training procedure, which gives rise to a powerful tool to approximate various filters without knowing the original models and implementation details. The only difference among these operators in our system becomes merely the learned parameters. Our system enables fast approximation for complex edge-aware filters and achieves up to 200x acceleration, regardless of their originally very different implementation. Fast speed can also be achieved when creating new effects using spatially varying filter or filter combination, bearing out the effectiveness of our deep edge-aware filters.

1. Introduction

Filters are fundamental building blocks for various computer vision tasks, among which edge-aware filters are of special importance due to their faithfulness to image structures. Different filtering approaches were proposed in literature to tackle texture removal (Subr et al., 2009; Xu et al., 2012), salient edge enhancement (Osher & Rudin, 1990), image flattening and cartoon denoise (Xu et al., 2011). It is, however, still an open question to bridge, not to mention to unify, these essentially different filtering approaches. It results in the common practice of implementing and accelerating these techniques regarding individual properties using distinct algorithms.

For instance, the well-trodden bilateral filter (Tomasi & Manduchi, 1998) has many accelerated versions (Durand & Dorsey, 2002; Paris & Durand, 2006; Weiss, 2006; Chen et al., 2007; Porikli, 2008; Yang et al., 2009; 2010). Images with multiple channels can be smoothed using the acceleration of high dimensional Gaussian filters (Adams et al., 2009; 2010; Gastal & Oliveira, 2011; 2012). Steps in these acceleration techniques cannot be used interchangeably. Further, new methods emerging every year (Farbman et al., 2008; Xu et al., 2011; Paris et al., 2011; Xu et al., 2012) greatly expand solution diversity. Efforts have also been put into understanding the nature, where connection between global and local approaches was established in (Elad, 2002; Durand & Dorsey, 2002; He et al., 2013). These techniques can be referred to as edge-aware filters.

In this paper, we initiate an attempt to implement various edge-aware filters within a single framework. The difference to previous approaches is that we construct a unified neural network architecture to approximate many types of filtering effects. It thus enlists tremendous practical benefit in implementation and acceleration where one segment of codes in programming can impact many filtering approaches.

We note that deep neural network (Krizhevsky et al., 2012) was applied to denoise (Burger et al., 2012; Xie et al., 2012; Agostinelli et al., 2013), rain drop removal (Eigen et al., 2013), super resolution (Dong et al., 2014), and image deconvolution (Xu et al., 2014) before. But it is still not trivial to model and include many general filters, which typically involve very large spatial support for creating necessary smoothing effect.

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

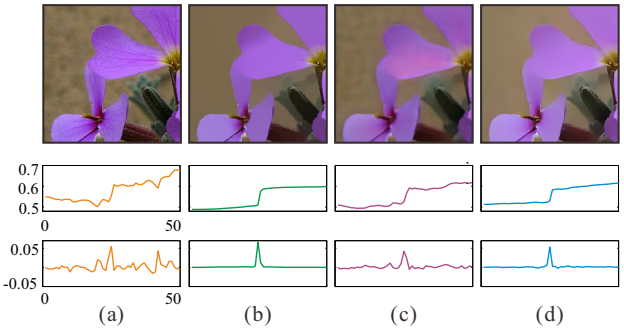


Figure 1. Learning operations using color- and gradient-domain constraints. (a) Input image. (b) L_0 smoothing effect to learn. (c) Learning result on image color. (d) Learning result on image gradient.

Our work contributes in multiple ways. First, we build a practical and complete system to learn and reproduce various filtering effects. The tractability is in part due to the proposed gradient-domain learning procedure with optimized image reconstruction, which captures the commonness of edge-aware operators by enforcing constraints on edges. Second, the resulting deep edge-aware filters are with linear complexity and run at a constant time even for filters that are completely different in their original implementation. Finally, various new effects can be easily created by combining or adapting original filters in our unified framework. On graphics processing unit (GPU), our implementation achieves up to 200x acceleration for several filters and yields state-of-the-art speed for most filters.

2. Our Approach

The input color image and edge-aware filtering operators are denoted as I and $\mathcal{L}(I)$ respectively. $\mathcal{L}(I)$ could be a nonlinear process and operates locally or globally. Our goal is to approximate $\mathcal{L}(I)$ by a unified feed-forward process $\mathcal{F}_{\mathbf{W}}(I)$ for any input image I. Here $\mathcal{F}$ denotes the network architecture and $\mathbf{W}$ represents network parameters, controlling the behavior of the feed-forward process. One simple strategy is to train the network by directly minimizing the summed color square errors

$$\|\mathcal{F}_{\mathbf{W}}(I) - \mathcal{L}(I)\|^2. \quad (1)$$

It however does not satisfyingly approximate a few edge-preserving operators. One example is shown in Fig. 1, where (a) is the input image and (b) is the smoothing effect we approximate, generated by L_0 smoothing (Xu et al., 2011). (c) shows the result with the network trained using the color square difference. The training details will be provided later. Obviously, (c) is more blurred than necessary and contains unwanted details. The 1D scan lines of the region are shown in the second row with their gradients in the third row.

2.1. Gradient Constraints and Objective

According to above finding, our method does not simply adopt Eq. (1), but rather modifies it to the gradient domain process, as illustrated in the pipeline of Fig. (2). We show our result using the exactly the same training and testing process in Fig. 1(d) for comparison. It is visually much better than (c) in terms of reproducing the L_0 smoothing effect.

Quantitatively, the mean square error (MSE) of (c) along the scanline in gradient domain is $2.6E-5$ while that of (d) is $0.6E-5$, complying with human perception to perceive contrast better than absolution color values. This gradient domain MSE is also sensitive to the change on sharp edges, which is a favored property in edge-preserving filtering.

With this understanding, we define our objective function on ∇I instead of I. Because most edge-aware operators can produce the same effect even if we rotate the input image by 90 degrees, we train the network only on one direction of gradients and let the horizontal and vertical gradients share the weights. We denote by ∂I the gradients.

Now, given D training image pairs $(I_0, \mathcal{L}(I_0))$, $(I_1, \mathcal{L}(I_1))$, $\cdots$, $(I_{D-1}, \mathcal{L}(I_{D-1}))$ exhibiting the ideal filtering effect, our objective is to train a neural network minimizing

$$\frac{1}{D}\sum_i \left\{ \frac{1}{2}\|\mathcal{F}_{\mathbf{W}}(\partial I_i) - \partial\mathcal{L}(I_i)\|^2 + \lambda\Phi(\mathcal{F}_{\mathbf{W}}(\partial I_i)) \right\}, \quad (2)$$

where $\{\partial I_i, \partial\mathcal{L}(I_i)\}$ is one training example pair in gradient domain. We also incorporate a sparse regularization term $\Phi(\mathcal{F}_{\mathbf{W}}(\partial I_i))$ to enforce sparsity on gradients. Note that this is not for constraining the neural network weight $\mathbf{W}$, but rather to enforce one common property in edge-preserving smoothing to suppress color change and favor strong edges. Empirically, this term helps generate decent weights initialization in the neural network. $\Phi(z) = (z^2+\epsilon^2)^{1/2}$ in Eq. (2) is the Charbonnier penalty function, approaching $|z|$ but differentiable at 0. λ is the regularization weight.

2.2. Network $\mathcal{F}_{\mathbf{W}}(\cdot)$

The choice of convolutional neural network (CNN) architecture is based on the fact that weights sharing allows for relatively larger interactive range than other fully connected structures such as Denoising Autoencoders (DAE). More importantly, several existing acceleration approaches for edge-aware filters map each 2D image into a higher dimensional space for acceleration by Gaussian convolutions. It partly explains how edge-preserving operators can be accomplished by the convolutional structure.

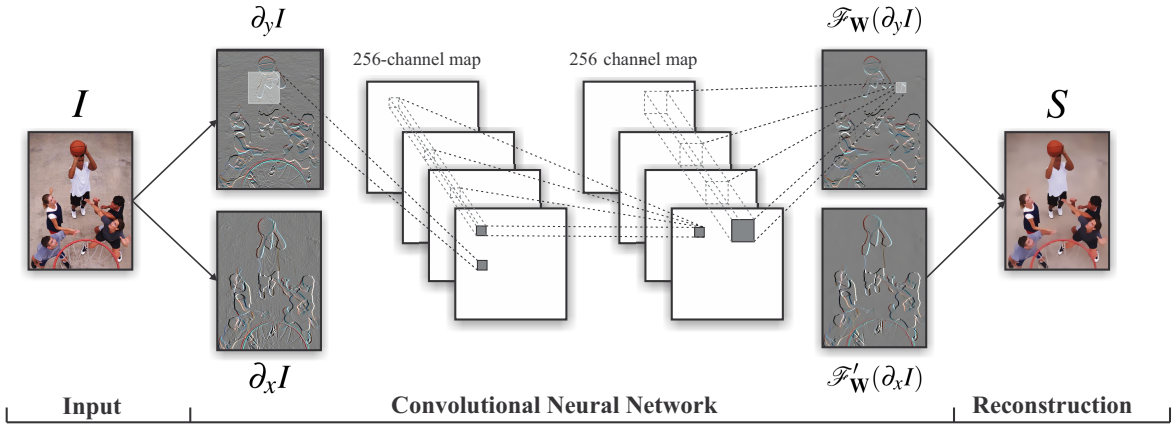


Figure 2. A unified learning pipeline for various edge-aware filtering techniques. The main building blocks are a 3-layer deep convolutional neural network and an optimized image reconstruction process.

Algorithm 1 Deep Edge-Aware Filters

1: **input:** one million image patches $\{I_i\}$, learning rate η, regularization parameter λ;
2: **initialization:** $\{\mathbf{W}^n\} \leftarrow N(0,1)$, $\{b^n\} \leftarrow \mathbf{0}$;
3: **for** each image patch **do**
4: apply $\mathcal{L}()$ to patch I_i;
5: compute gradients $\partial_x I_i$, $\partial_y I_i$, $\partial_x \mathcal{L}(I_i)$, $\partial_y \mathcal{L}(I_i)$; rotate $\partial_x I_i$, $\partial_x \mathcal{L}(I_i)$ to generate a sample;
6: update the CNN weights using backward-propagation (Eq. (6));
7: learning rate decay $\eta \leftarrow 0.001/(1 + i \cdot 1E - 7)$;
8: **end for**
9: **output:** optimized weights $\mathbf{W}$.

Our convolutional network structure can be expressed as

$$\mathcal{F}_{\mathbf{W}}(\partial I) = \mathbf{W}^n * \mathcal{F}^{n-1}(\partial I) + b^n, \quad n = 3 \tag{3}$$

$$\mathcal{F}^n(\partial I) = \sigma(\mathbf{W}^n * \mathcal{F}^{n-1}(\partial I) + b^n), \; n = 1, 2 \tag{4}$$

$$\mathcal{F}^0(\partial I) = \partial I. \tag{5}$$

n in this expression indexes layers, it ranges from 0 (bottom layer) to 3 (top layer), as our CNN contains two hidden layers for convolution generation. For the bottom layer with index 0, it is simply the input gradient.

In each intermediate layer, Eq. (4) denotes a convolution process for the nodes in the network regarding its neighbors. $\mathbf{W}^n$ here is the convolution kernel written in vector form and $\mathcal{F}^{n-1}(\partial I)$ denotes all nodes in this layer. By convention, $*$ is used to indicate the network connected in a convolution way, or typically referred to as weights sharing. b^n is the bias or perturbation vector. Nonlinearity is allowed with the hyperbolic tangent $\sigma(\cdot)$. The top layer with $\mathcal{F}_{\mathbf{W}}(I)$ in Eq. (3) generates the final output from the network, i.e., the filtering result.

The network is illustrated in Fig. 2. The input image ∂I is of size $p \times q \times 3$ for 3 color channels. $p \times q$ is the spatial resolution. In our constructed network, the first hidden layer $\mathcal{F}^1(\partial I)$ is generated by applying k different kernels in 3 dimensions to the zero layer input after convolution and nonlinear truncation, resulting in a k-channel image map in $\mathcal{F}^1(\partial I)$. This process, explained intuitively, maps each local color patch into a k-dimensional pixel vector by convolution. The operations are to get pixels within each local patch, re-map the detail, and put them into a vector.

The second hidden layer $\mathcal{F}^2(\partial I)$ is generated on output $\mathcal{F}^1(\partial I)$ from the first layer by applying an $1 \times 1 \times k$ filter, as shown in Fig. 2. This process produces weighted average of the processed pixel vector and performs the "smoothing" operation. The final result is obtained by further applying three 3D filters to $\mathcal{F}^2(\partial I)$ for restoring the sharp edges, corresponding to "edge-aware" processing. We do not add the hyperbolic tangent to the final layer.

When working in gradient domain, the fixed kernel size is sufficient when approximating filters with large spatial influence, leading to a constant time implementation. In our implementation, the convolution kernel is of the size 16×16 and $k = 256$.

3. Training

We use the stochastic gradient descent (SGD) to minimize the energy function Eq. (2). We randomly collect one million 64×64 patches from high resolution natural image data obtained from flickr and their smoothed versions as training samples. They contain enough structure variation for successful CNN training in multiple layers. More patches do not improve the results much in our extensive experiments.

Given one patch I_i, we first generate $\{\mathcal{L}(I_i)\}$. Gradient operators are then applied to get ∂I_i and $\partial \mathcal{L}(I_i)$. In each

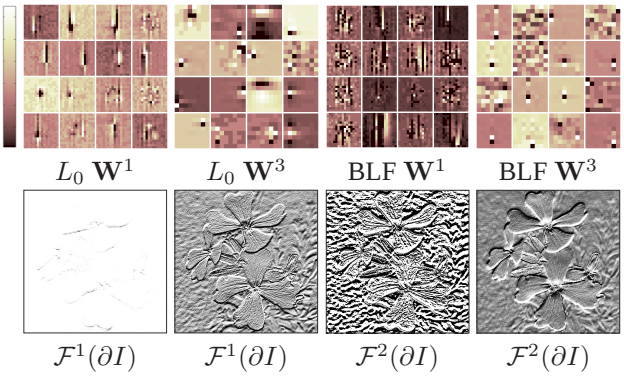


Figure 3. Visualization of intermediate results from hidden layers. The top row shows the selected weights trained for L_0 smoothing and bilateral filter. The bottom row shows the hidden layer activation for the bilateral filter.

step of training for one sample, the update process is

$$W_{t+1} = W_t - \eta \left\{ (\mathcal{F}_{\mathbf{W}}(\partial I_i) - \partial \mathcal{L}(I_i))^T + \lambda (\mathcal{F}^2_{\mathbf{W}}(\partial I_i) + \epsilon^2)^{-1/2} \mathcal{F}^T_{\mathbf{W}}(\partial I_i) \right\} \frac{\partial \mathcal{F}_{\mathbf{W}}(\partial I_i)}{\partial W}, \quad (6)$$

where η is the learning rate, setting to 0.001 with decay during the training process. The gradients are further back-propagated through $\partial \mathcal{F}_{\mathbf{W}}(\partial I_i) \partial W^{-1}$. The steps of our training procedure are provided in Algorithm 1.

To understand this process, we visualize in Fig. 3 the weights and intermediate results from hidden layers. The first row shows the trained weights $\mathbf{W}^1$ and $\mathbf{W}^3$ for two filtering methods. $\mathbf{W}^1$ contains mostly vertical structures, in accordance to the direction of input gradients. The output of the hidden layer $\mathcal{F}^1(\partial I)$ for bilateral filter is also visualized in the second row, which contains noisy structure that represents details at differen locations. $\mathcal{F}^2(\partial I)$ looks more blurry than $\mathcal{F}^1(\partial I)$. It actually plays the role to smooth details by canceling out them from different channels. The main edges are not that sharp in $\mathcal{F}^2(\partial I)$. They are enhanced adaptively in W^3 with respect to image content.

4. Testing

After neural network training, we apply our system to new images for producing the learned effect. This process needs a final step in image reconstruction from gradient to color domains. The common solution is to solve the Poisson equation (Pérez et al., 2003) for direct gradient integration. But our experiments show this may not produce visually compelling results because the gradients are possibly not integrable given them in two directions computed independently. Therefore, solving the Poisson equation may lead to large errors and cause color shift. One example is shown in Fig. 4. With this finding, we resort to another optimization method for final image reconstruction.

(a) (b) (c)

Figure 4. Poisson blending in (c) makes color change too much from the input (a). Our image reconstruction in (b) does not have this problem.

4.1. Image Reconstruction

We denote by S our final output gradient maps. Our image reconstruction is to also consider the structural information in the input image to guide smoothing in gradient domain. We thus introduce two terms putting together as

$$\|S - I\|^2 + \beta \left\{ \|\partial_x S - \mathcal{F}'_{\mathbf{W}}(\partial_x I)\|^2 + \|\partial_y S - \mathcal{F}_{\mathbf{W}}(\partial_y I)\|^2 \right\}, \quad (7)$$

where $\|S - I\|^2$ is the color confidence to use the input image to guide smoothed image reconstruction. The second term is the common one to use our gradient results. β is a parameter balancing the two loss functions. Contrary to Poisson integration that relies purely on the computed gradients with a boundary condition, our reconstruction finds a balance between original color and computed gradients, which is especially useful when gradients are not integrable. The optimal β is filter-dependent. We determine this value using a round of simple search, described later.

The energy in Eq. (7) is quadratic w.r.t. image S. When using forward difference to compute the partial derivative, the minimization process leads to a linear system with a sparse five point Laplacian. We use preconditioned conjugate gradient (PCG) to speed up the sparse linear system with the incomplete Cholesky factorization preconditioner (Szeliski, 2006; Xu et al., 2012).

Theoretically, the reason that this reconstruction step works so well can be exhibited by showing an analogy to half-quadratic splitting optimization (HQSO), which has been used to solve complex L_1, L_p $(0 < p < 1)$, and L_0 norm optimization in computer vision.

In these problems, the originally complex energy function is decomposed into single-variable optimization and quadratic optimization where the latter is exactly in the form of Eq. (7). The only difference is that HQSO produces the reference gradients iteratively while our method uses a data-driven learning scheme to generate the reference gradients $\mathcal{F}_{\mathbf{W}}(\partial I)$. Given that the network structure is expressive enough, it is possible to learn the behavior of underlying procedure without iteration. This also accounts

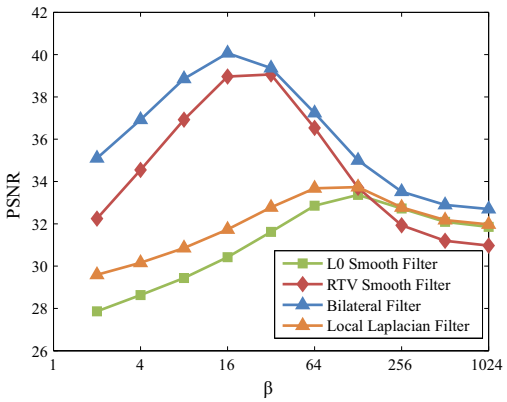


Figure 5. Varying β produces different-quality results.

for the capability of our method to approximate complex effects, such as L_0 smoothing.

4.2. Optimal β Finding

β is a relatively sensitive parameter because it corresponds to the suitable amount of color information in the input image to be taken into consideration. This value varies for different filters. We perform a greedy search for the optimal value.

The process is simple – we first set β to a group of values within $[1, 1E5]$ and apply them to 100 testing images. Peak signal-to-noise ratios (PSNRs) are recorded. Fig. 5 plots the curves for bilateral filter (Tomasi & Manduchi, 1998), local Laplacian (Paris et al., 2011), L_0 smoothing (Xu et al., 2011), and texture smoothing (Xu et al., 2012). The first two methods involve local filters and the latter two use global minimization. There are peaks located differently for these smoothing effects, indicating that different effects need their own best parameters. We choose β as the one producing the largest average PSNR for each method.

4.3. Applying Deep Edge-Aware Filters

Once β is obtained, we use $\mathcal{F}_{\mathbf{W}}(\cdot)$, together with β, to apply the learned Deep Edge-Aware Filters. It corresponds to the testing pass with the convolutional neural network. For a given image I, we first transform it into gradient domain and feed ∂I into the network to get the filtered gradients. The final image S is obtained by solving the reconstruction function (7) with its optimal β.

5. More Discussions

We discuss in this section parameter setting, choice of processed domain, and possible regularization.

(a) color domain (b) gradient domain

Figure 6. Comparison of color- and gradient-domain processing.

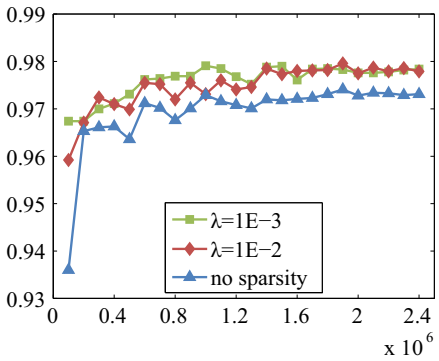


Figure 7. Effectiveness of sparsity regularization. x-axis: training samples; y-axis: SSIM scores.

Gradient vs. Color Fig. 6 shows a visual comparison of taking a color image and use its gradient maps in approximating L_0 smoothing respectively in our system, corresponding to the discussion in Section 2.1. The input image is shown in Fig. 4. The PSNRs are 28.71 and 34.32 respectively for color and gradient level process. The SSIMs (Wang et al., 2004) are 0.94 and 0.98. Higher PSNR and SSIM in gradient domain are observed for all data we experimented with. In implementation, the vertical and horizontal gradients are processed in a single pass by rotating the vertical gradients and stacking the two inputs.

Sparse Regularization We enforce sparsity in training the network in Eq. (2) to resist possible errors. We plot results in Fig. 7 to show evolution of the network over different training samples. x-axis values are the numbers of training patches. The SSIM (Wang et al., 2004) scores are obtained excluding the reconstruction step. For smoothing with strong sparsity, our regularization helps generate high SSIM values. More importantly, high SSIM results can already be obtained on only hundreds of image patches.

Adjusting Smoothing Strength Many edge-aware effects are controlled by smoothing strength parameters. To alter the behavior for one filter or method, we can train the network with all necessary parameter values. Alternatively, we also provide a faster approximation based on parameter sampling and interpolation. The intuition is that we obtain

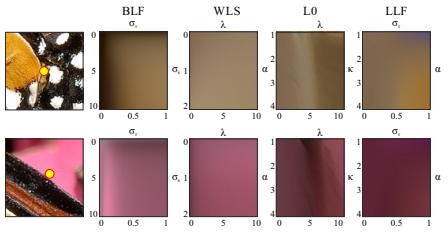


Figure 8. Parameter space for different methods.

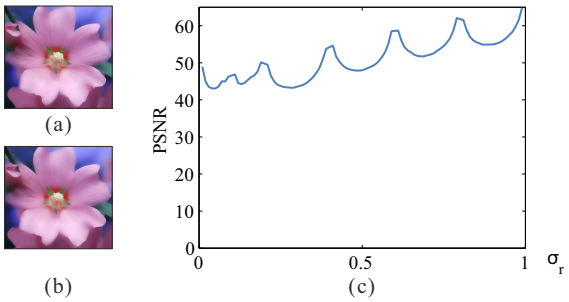


Figure 9. Parameter interpolation result. x-axis: range parameter σ_r; y-axis: PSNR scores.

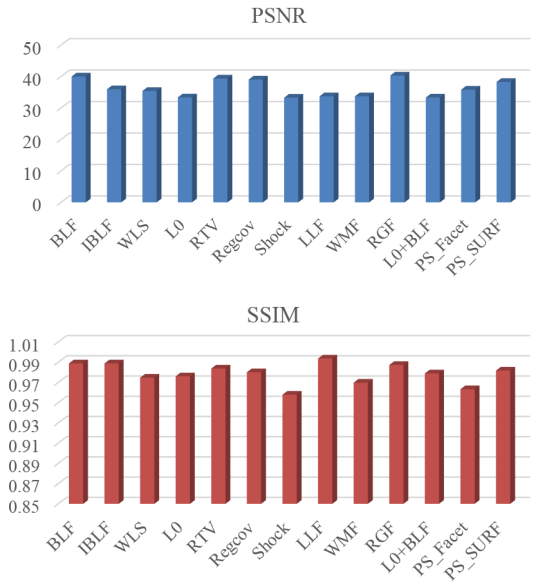


Figure 11. Average PSNRs and SSIMs of our approximation of various filtering techniques.

the required smoothing effect by interpolating results generated with similar parameters.

We use the gray-level co-occurrence matrices (GLCM) (Haralick et al., 1973) to measure pixel color change smoothness by varying parameter values, as visualized in Fig. 8. For example, in the first-column bilateral filter results, we show the color of one pixel (yellow dots in input images) by varying the two parameters σ_r and σ_s. It forms a 2D space shown in the first row. It is smooth, indicating steady pixel color change when varying parameter values. Quantitatively, the average GLCMs for bilateral filter, WLS, L_0 smoothing and local Laplacian filter are as large as 0.95, 0.96, 0.83, and 0.94 respectively. The high score manifests smooth variation in this space. For reference, the average GLCM for natural image patches is only 0.3.

Our interpolation works as follows. We sample smoothing parameters for neural network training. Then for a new image to be smoothed with parameter values different from all these samples, we generate a few smoothed images using the network trained for the closest parameter values. The final result is the bi-cubic interpolation of the corresponding color pixels in the nearby images. For filters with two controlling parameters, we interpolate the result using four color pixels in the 2D parameter space shown in Fig. 8. For filters with a single parameter, two nearby images are used.

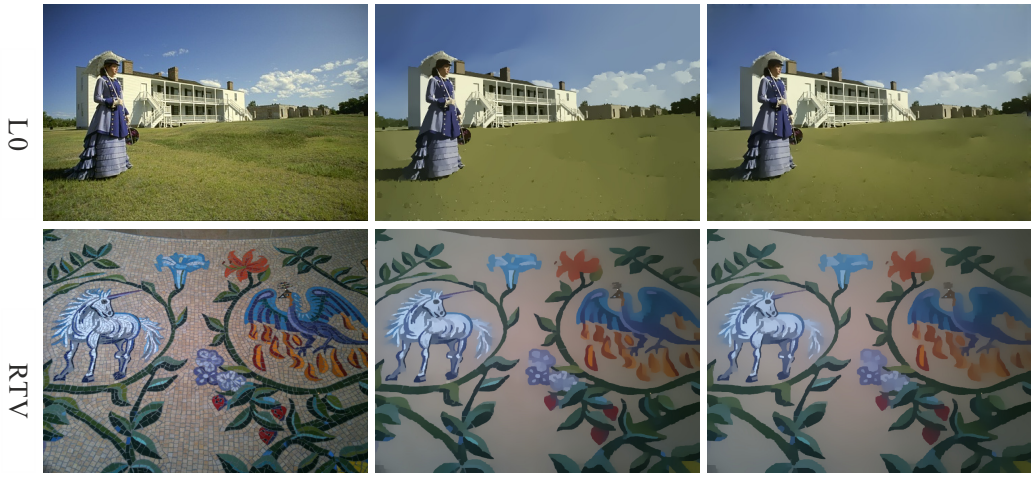
Fig. 9(a) shows a bilaterally filtered image with parameters $\sigma_s = 7$ and $\sigma_r = 0.35$. Our result by above interpolation is shown in Fig. 9(b) using the nearby trained networks for $\sigma_r = 0.2$ and $\sigma_r = 0.4$. For simplicity of illustration, we set σ_s to 7. The PSNR is as high as 46.17.

Fig. 9(c) plots PSNRs with training CNNs at $\sigma_r = 0.1, 0.2, 0.4, 0.6, 0.8, 1$. Results for other σ_r values are produced by interpolation. The PSNR slightly decreases when using interpolation, but is still within an acceptable range (> 30). It is also notable that PSNRs go up generally when increasing σ_r. It is because a larger σ_r makes bilateral filter more like Gaussian. When $\sigma_r = 1$, Gaussian smoothing can be exactly obtained using convolutional neural network.

Relation to Other Approach Yang et al. (Yang et al., 2010) proposed a learning based method to approximate bilateral filter. The major step is a combination of several image components, consisting of exponentiation of the original image and their Gaussian filtered versions. The combination mapping is trained using SVM. Intriguingly, if we set our $\mathcal{F}^2(I)$ layer to the much simplified image components this method used and the final layer to simply combining weights instead of networks, the two methods become similar. In fact compared to general learning methods, our system is much more powerful because we train not only the combining weights, but as well image maps from a deeper, more expressive network architecture.

6. Experiments and Applications

We use our method to simulate a number of practical operators, including but not limited to bilateral filter (BLF) (Paris & Durand, 2006), local Laplacian filter (LLF) (Paris et al., 2011), region covariance filter (RegCov) (Karacan et al., 2013), shock filter (Osher & Rudin, 1990), weighted least square (WLS) smoothing (Farbman et al.,

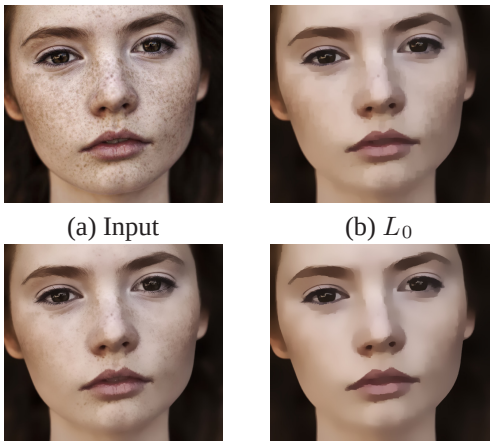


(a) input (b) smoothing results (c) ours

Figure 10. Visual comparison with other generative smoothing operators.

Resolution	QVGA	VGA	720p	1080p	2k
BLF Grid	0.11	0.41	0.98	2.65	3.03
IBLF	0.46	1.41	3.18	8.36	12.03
WLS	0.71	3.25	9.42	28.65	33.73
L0	0.36	1.60	4.35	11.89	15.07
RTV	1.22	6.26	16.26	42.59	48.25
RegCov	59.05	229.68	577.82	1386.95	1571.91
Shock	0.45	3.19	8.48	23.88	26.93
LLF	207.93	849.78	2174.92	5381.36	6130.44
WMF	0.94	3.54	4.98	14.32	15.41
RGF	0.35	1.38	3.42	9.02	10.31
Ours	0.23	0.83	2.11	5.78	6.65

Table 1. Running time for different resolution images on desktop PC (Intel i7 3.6GHz with 16GB RAM, Geforce GTX 780 Ti with 3GB memory). Running time is obtained on color images.



(a) Input (b) L_0 (c) BLF (d) Our learned filter combo

Figure 13. Filter combo effect illustration. Our method is capable of training a combination of filters without applying the network multiple times.

2008), L_0 smoothing (Xu et al., 2011), weighted median filter (Zhang et al., 2014b), rolling guidance filter (Zhang et al., 2014a), and RTV texture smoothing (Xu et al., 2012). They are representative as both local- and global- schemes are included and effect of smoothing, sharpening, and texture removal can be produced. Our implementation is based on the Matlab VCNN framework (Ren & Xu, 2015).[1]

6.1. Quantitative Evaluation

We quantitatively evaluate our method (Paris et al., 2011). The average PSNRs and SSIMs are plotted in Fig. 11. All are high to produce usable results. For bilateral filter, setting σ_r larger yields higher PSNRs. This is because the BLF degenerates to a Gaussian filter when σ_r is large. For fare comparison, we use $\sigma_r = 0.1$ to report PSNR.

Global optimization such as WLS smoothing does not incur problems in approximation, due primarily to the gradient learning scheme. Approximation of highly nonlinear weighted median filter and L_0 smoothing also yields reasonable results thanks to the nonlinearity in the deep convolutional neural network. Fig. 10 gives the visual comparison of the original filtering results and ours.

6.2. Filter Copycat

We learn edge-aware operators even without knowing any details in the original implementation. We approximate two effects generated by Adobe Photoshop. One is surface blur.

[1]Our implementation is available at the project webpage `http://lxu.me/projects/deaf`.

(a) inputs (b) Photoshop results (c) ours

Figure 12. Filter Copycat. We learn two operators from Adobe Photoshop and reproduce them on the new images in constant time.

Figure 14. Spatially varying filter is achieved using our neural network in one pass.

Our average PSNR and SSIM for this operator reach 40 and 0.97 respectively. Another effect is "facet", which turns an image into block-like clutters. Our corresponding average PSNR and SSIM are 35.8 and 0.96, which are also very good. Two image examples are shown in Fig. 12.

The deep edge-aware-filter takes constant time to generate different types of effect. We list running time statistics in Table 1. Our approximation is faster than almost all other filters expect the fast bilateral filter approximation using grid (Chen et al., 2007). It also achieves 200+ times acceleration for several new filtering effects (Paris et al., 2011; Karacan et al., 2013). The performance of all approaches is generated based on the authors' original implementation.

Among the filters we trained, an interesting group is the iterative bilateral filter (IBLF) (Fattal et al., 2007) and rolling guidance filter (Zhang et al., 2014a) that successively apply filter in the same image. We can approximate them perfectly without applying the network repeatedly. It naturally leads to our application of "filter combo" explained below.

6.3. Filter Combo

The combination of several filters may generate special effects. It was shown in (Xu et al., 2011) that combination of L_0 smoothing and bilateral filter can remove details and noise without introducing large artifacts. Instead of applying twice the network to achieve the special effect. We train the *filter combo* using the same network in one pass. Fig. 13 shows an input image (a), L_0 smoothing result (b), which does not remove details on the face, and (c) result of bilateral filtering. Our trained "filter combo" yields an average PSNR 35.26. The image result is shown in Fig. 13(d), preserving edges while removing many small-scale structures.

6.4. Spatially Varying Filtering

Since we process the image in a patch-wise fashion. We can apply different filtering effects to an image, still taking constant time. In this experiment, we reduce the patch size to 64×64 and gradually apply bilateral filter with different parameters. One result is shown in Fig. 14.

7. Concluding Remarks

We have presented a deep neural network based system to uniformly realize many edge-preserving smoothing and enhancing methods working originally either as filter or by global optimization. Our method does not need to know the original implementation as long as many input and output images are provided. After the training process, we can then apply very similar effects to new images in constant time.

The learning based system can only model deterministic procedures. Thus our method is not suitable to produce randomized filtering effect, which changes even with fixed input and system parameters.

References

Adams, Andrew, Gelfand, Natasha, Dolson, Jennifer, and Levoy, Marc. Gaussian kd-trees for fast high-dimensional filtering. *ACM Trans. Graph.*, 28(3), 2009.

Adams, Andrew, Baek, Jongmin, and Davis, Myers Abraham. Fast high-dimensional filtering using the permutohedral lattice. *Comput. Graph. Forum*, 29(2):753–762, 2010.

Agostinelli, Forest, Anderson, Michael R., and Lee, Honglak. Adaptive multi-column deep neural networks with application to robust image denoising. In *NIPS*, 2013.

Burger, Harold Christopher, Schuler, Christian J., and Harmeling, Stefan. Image denoising: Can plain neural networks compete with bm3d? In *CVPR*, 2012.

Chen, Jiawen, Paris, Sylvain, and Durand, Frédo. Real-time edge-aware image processing with the bilateral grid. *ACM Trans. Graph.*, 26(3):103, 2007.

Dong, Chao, Loy, Chen Change, He, Kaiming, , and Tang, Xiaoou. Learning a deep convolutional network for image super-resolution. In *ECCV*, 2014.

Durand, Frédo and Dorsey, Julie. Fast bilateral filtering for the display of high-dynamic-range images. *ACM Trans. Graph.*, 21(3):257–266, 2002.

Eigen, David, Krishnan, Dilip, and Fergus, Rob. Restoring an image taken through a window covered with dirt or rain. In *ICCV*, 2013.

Elad, Michael. On the origin of the bilateral filter and ways to improve it. *IEEE Transactions on Image Processing*, 11(10):1141–1151, 2002.

Farbman, Zeev, Fattal, Raanan, Lischinski, Dani, and Szeliski, Richard. Edge-preserving decompositions for multi-scale tone and detail manipulation. *ACM Trans. Graph.*, 27(3), 2008.

Fattal, Raanan, Agrawala, Maneesh, and Rusinkiewicz, Szymon. Multiscale shape and detail enhancement from multi-light image collections. *ACM Trans. Graph.*, 26 (3):51, 2007.

Gastal, Eduardo S. L. and Oliveira, Manuel M. Domain transform for edge-aware image and video processing. *ACM Trans. Graph.*, 30(4):69, 2011.

Gastal, Eduardo S. L. and Oliveira, Manuel M. Adaptive manifolds for real-time high-dimensional filtering. *ACM Trans. Graph.*, 31(4):33, 2012.

Haralick, Robert M., Shanmugam, K. Sam, and Dinstein, Its'hak. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, 3(6): 610–621, 1973.

He, Kaiming, Sun, Jian, and Tang, Xiaoou. Guided image filtering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(6): 1397–1409, 2013.

Karacan, Levent, Erdem, Erkut, and Erdem, Aykut. Structure-preserving image smoothing via region covariances. *ACM Trans. Graph.*, 32(6):176, 2013.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *NIPS*, pp. 1106–1114, 2012.

Osher, Stanley and Rudin, Leonid I. Feature-oriented image enhancement using shock filters. *SIAM Journal on Numerical Analysis*, 27(4):919–940, 1990.

Paris, Sylvain and Durand, Frédo. A fast approximation of the bilateral filter using a signal processing approach. In *ECCV (4)*, pp. 568–580, 2006.

Paris, Sylvain, Hasinoff, Samuel W., and Kautz, Jan. Local laplacian filters: edge-aware image processing with a laplacian pyramid. *ACM Trans. Graph.*, 30(4):68, 2011.

Pérez, Patrick, Gangnet, Michel, and Blake, Andrew. Poisson image editing. *ACM Trans. Graph.*, 22(3):313–318, 2003.

Porikli, Fatih. Constant time o(1) bilateral filtering. In *CVPR*, 2008.

Ren, Jimmy SJ. and Xu, Li. On vectorization of deep convolutional neural networks for vision tasks. In *AAAI*, 2015.

Subr, Kartic, Soler, Cyril, and Durand, Frédo. Edge-preserving multiscale image decomposition based on local extrema. *ACM Trans. Graph.*, 28(5), 2009.

Szeliski, Richard. Locally adapted hierarchical basis preconditioning. *ACM Trans. Graph.*, 25(3):1135–1143, 2006.

Tomasi, Carlo and Manduchi, Roberto. Bilateral filtering for gray and color images. In *ICCV*, pp. 839–846, 1998.

Wang, Zhou, Bovik, Alan C., Sheikh, Hamid R., and Simoncelli, Eero P. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

Weiss, Ben. Fast median and bilateral filtering. *ACM Trans. Graph.*, 25(3):519–526, 2006.

Xie, Junyuan, Xu, Linli, and Chen, Enhong. Image denoising and inpainting with deep neural networks. In *NIPS*, pp. 350–358, 2012.

Xu, Li, Lu, Cewu, Xu, Yi, and Jia, Jiaya. Image smoothing via l0 gradient minimization. *ACM Trans. Graph.*, 30(6), 2011.

Xu, Li, Yan, Qiong, Xia, Yang, and Jia, Jiaya. Structure extraction from texture via relative total variation. *ACM Trans. Graph.*, 31(6):139, 2012.

Xu, Li, Ren, Jimmy SJ., Liu, Ce, and Jia, Jiaya. Deep convolutional neural network for image deconvolution. In *NIPS*, 2014.

Yang, Qingxiong, Tan, Kar-Han, and Ahuja, Narendra. Real-time o(1) bilateral filtering. In *CVPR*, pp. 557–564, 2009.

Yang, Qingxiong, Wang, Shengnan, and Ahuja, Narendra. Svm for edge-preserving filtering. In *CVPR*, pp. 1775–1782, 2010.

Zhang, Qi, Shen, Xiaoyong, Xu, Li, and Jia, Jiaya. Rolling guidance filter. In *ECCV*, 2014a.

Zhang, Qi, Xu, Li, and Jia, Jiaya. 100+ times faster weighted median filter (wmf). In *CVPR*, 2014b.