Image Smoothing via Unsupervised Learning

QINGNAN FAN*, Shandong University, Beijing Film Academy
JIAOLONG YANG, Microsoft Research Asia
DAVID WIPF, Microsoft Research Asia
BAOQUAN CHEN, Peking University, Shandong University
XIN TONG, Microsoft Research Asia

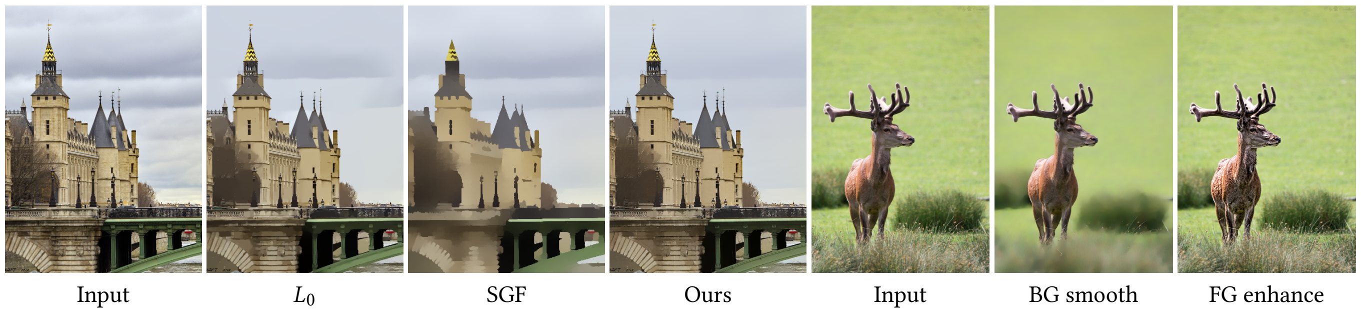

Fig. 1. Two applications of traditional edge-preserving image smoothing. *Left 4 panels*: Elimination of low-amplitude details while maintaining high-contrast edges using our method and representative traditional methods L_0 [Xu et al. 2011] and SGF [Zhang et al. 2015]. L_0 regularization has a strong flattening effect. However, the side effect is that some spurious edges arise in local regions with smooth gradations, such as those on the cloud. SGF is dedicated to elimination of fine-scale high-contrast details while preserving large-scale salient structures. However, semantically-meaningful information such as the architecture and flagpole can be over-smoothed. In contrast, our result exhibits a more appropriate, targeted balance between color flattening and salient edge preservation. *Right 3 panels*: Content-aware image manipulation. Using minimal modification of the guidance image in our proposed pipeline, we are able to implement background (BG) smoothing or foreground (FG) enhancement/exaggeration via a single deep network. More smoothing effects and applications can be found in Section 7. Photo courtesy of Flickr user Philippe Rouzet and Cloudtail the Snow Leopard.

Image smoothing represents a fundamental component of many disparate computer vision and graphics applications. In this paper, we present a unified unsupervised (label-free) learning framework that facilitates generating flexible and high-quality smoothing effects by directly learning from data using deep convolutional neural networks (CNNs). The heart of the design is the training signal as a novel energy function that includes an edge-preserving regularizer which helps maintain important yet potentially vulnerable image structures, and a spatially-adaptive L_p flattening criterion which imposes different forms of regularization onto different image regions for better smoothing quality. We implement a diverse set of image smoothing solutions employing the unified framework targeting various applications such as, image abstraction, pencil sketching, detail enhancement, texture removal and content-aware image manipulation, and obtain results comparable with or better than previous methods. Moreover, our method is extremely fast with a modern GPU (e.g, 200 fps for 1280×720 images).

CCS Concepts: • **Computing methodologies → Image manipulation**; *Computational photography*; *Neural networks*;

*Part of this work was done during Qingnan Fan's internship at MSRA.

Additional Key Words and Phrases: image smoothing, edge preservation, unsupervised learning

ACM Reference format:
Qingnan Fan, Jiaolong Yang, David Wipf, Baoquan Chen, and Xin Tong. 2018. Image Smoothing via Unsupervised Learning. *ACM Trans. Graph.* 37, 6, Article 259 (November 2018), 14 pages.
https://doi.org/10.1145/3272127.3275081

1 INTRODUCTION

The goal of image smoothing is to eliminate unimportant fine-scale details while maintaining primary image structures. This technique has a wide range of applications in computer vision and graphics, such as tone mapping, detail enhancement, and image abstraction.

Image smoothing has been extensively studied in the past. The early literature was dominated by filtering-based approaches [Chen et al. 2007; Fattal 2009; Gastal and Oliveira 2011; Paris and Durand 2006; Perona and Malik 1990; Tomasi 1998; Weiss 2006] due to their simplicity and efficiency. In recent years, smoothing algorithms based on global optimization have gained much popularity due to their superior smoothing results [Bi et al. 2015; Farbman et al. 2008; Liu et al. 2017; Min et al. 2014; Xu et al. 2011, 2012]. Despite the great improvements, however, their smoothing results are still not perfect, and no existing algorithm can serve as an image smoothing panacea for various applications. Moreover, these approaches are often very time-consuming. With the increasing power of modern

GPUs and the enormous growth of deep convolutional neural networks (CNNs), there is an emerging interest in employing CNNs as surrogate smoothers in lieu of the costly optimization-based approaches [Chen et al. 2017a; Fan et al. 2017; Liu et al. 2016; Xu et al. 2015]. These methods train CNNs in a fully supervised manner where the target outputs are generated by existing smoothing methods. While substantial speed-up can be achieved, they still produce (approximations of) extant smoothing effects.

In this work, we seek to generate flexible and superior smoothing effects by directly learning from data. We leverage a CNN to do so, such that our method not only features the learned smoothing effects that are more appealing, but also enjoys a fast speed. However, the desired smoothing results (ground-truth labels) for supervising the training are difficult to obtain. Dense manually labeling for a large volume of training images is costly and cumbersome. To circumvent this issue, we design the training signal as an energy function, similar to the optimization-based methods, and train our method in an *unsupervised*, label-free setting.

We carefully designed our energy function to achieve quality smoothing effects in a unified unsupervised-learning framework. First, to explicitly fortify important image structures that may be weakened by the flattening operator, we include a criterion that minimizes the masked quadratic difference between two guidance maps computed from the input image and the smoothed estimate respectively. The guidance maps are formulated as edge responses, and the masks are computed using simple edge detection heuristics and can be manually modified further if desired. Second, we identified that many previous methods apply a fixed L_p-norm flattening criterion across the entire image, which may be detrimental to the smoothing quality. We instead introduce a spatially-adaptive L_p flattening criterion whereby the specific value of p is varied across images in accordance with the guidance maps. Given that $p = 2$ tends to smooth out edges while $p \in (0, 1]$ largely preserves them, the guidance maps allow different image regions to receive different regularizations most appropriate for handling local structural conditions. Importantly, we can apply application-specific guidance maps which allow for the seamless implementation of multiple different flattening effects.

We test our method on edge-preserving smoothing and various applications including image abstraction, pencil sketching, detail magnification, texture removal and content-ware image manipulation to show its effectiveness and versatility. Broadly speaking, the contribution of this paper can be distilled as follows:

- We introduce an unsupervised learning framework for image smoothing. Unlike previous methods, we do not need ground-truth labels for training and we can jointly learn from any sufficiently diverse corpus of images to achieve the desired performance.
- We are able to implement multiple different image smoothing solutions in a single framework, and obtain results comparable with or better than previous methods. A novel image smoothing objective function is proposed to achieve this which is built upon a spatially-adaptive L_p flattening criterion and an edge-preserving regularizer.
- Our new method is based on a convolutional neural network and its computational footprint is far below most previous methods. For example, processing a 1280×720 image takes only 5ms on a modern GPU.

2 RELATED WORK

As a fundamental tool for many computer vision and graphics applications, image smoothing has been extensively studied in the past decades. The filtering based approaches, such as anisotropic diffusion [Perona and Malik 1990], bilateral filter [Tomasi 1998] and many others [Chen et al. 2007; Fattal 2009; Gastal and Oliveira 2011; Kass and Solomon 2010; Paris and Durand 2006; Weiss 2006] have been the dominating image smoothing solutions for a long time. The core idea for such methods lies in filtering each pixel with its local spatial neighbourhood, and these methods are usually very efficient.

Recently, algorithms using mathematical optimization for image smoothing tasks gain more popularity due to their robustness, flexibility and more importantly the superiority of the smoothing results. For example, [Farbman et al. 2008] proposed an edge-preserving operator in a weighted least square (WLS) optimization framework, which prevents the local image regions from being over-sharpened with an L_2 norm. Similar schemes have been achieved more efficiently by [Liu et al. 2017; Min et al. 2014]. These works are devoted to extracting and manipulating the image details using image smoothing for various applications such as detail enhancement, HDR tone mapping, *etc.*

On the other hand, [Xu et al. 2011] proposed a sparse gradient counting scheme in an optimization framework by minimizing the L_0 norm. The method of [Bi et al. 2015] aimed at producing almost ideally flattening images where sharp edges are also well preserved with L_1 norm. These methods are particularly well-suited for preserving or enhancing the sharp edges, and remove the low-amplitude details. They can be useful for some stylization effects or intrinsic image decompositions.

In the aforementioned applications, the image smoothing algorithms typically exploit only gradient magnitude as the main cue to discriminate primary image structures from details. [Xu et al. 2012] presented the specifically designed relative total variation measures to extract meaningful structure, and [Ham et al. 2015] fuses appropriate structures of static and dynamic guidance images into a non-convex regularizer. Their goal is to remove fine-scale repetitive textures where local gradient can still be significant, which can not be easily achieved by the aforementioned smoothing approaches.

This regularization idea can also be interpreted as an image prior formulated in a deep network [Ulyanov et al. 2018] or image denoising engine [Romano et al. 2017]. Discussions about the L_p-norm regularization can also be found in [Bach et al. 2012; Chung and Vese 2009; Prasath et al. 2015]. Interestingly, [Mrázek et al. 2006] observe that even the image filters can all be derived from minimization of a single energy functional with data and smoothness term.

Most of the optimization based approaches are time-consuming, as they typically require solving large-scale linear systems (or others). Therefore, recently some methods such as [Chen et al. 2017a; Fan et al. 2017; Gharbi et al. 2017; Liu et al. 2016; Xu et al. 2015] were proposed to speed up existing smoothing operators. These methods train a deep neural network using the ground-truth smoothed

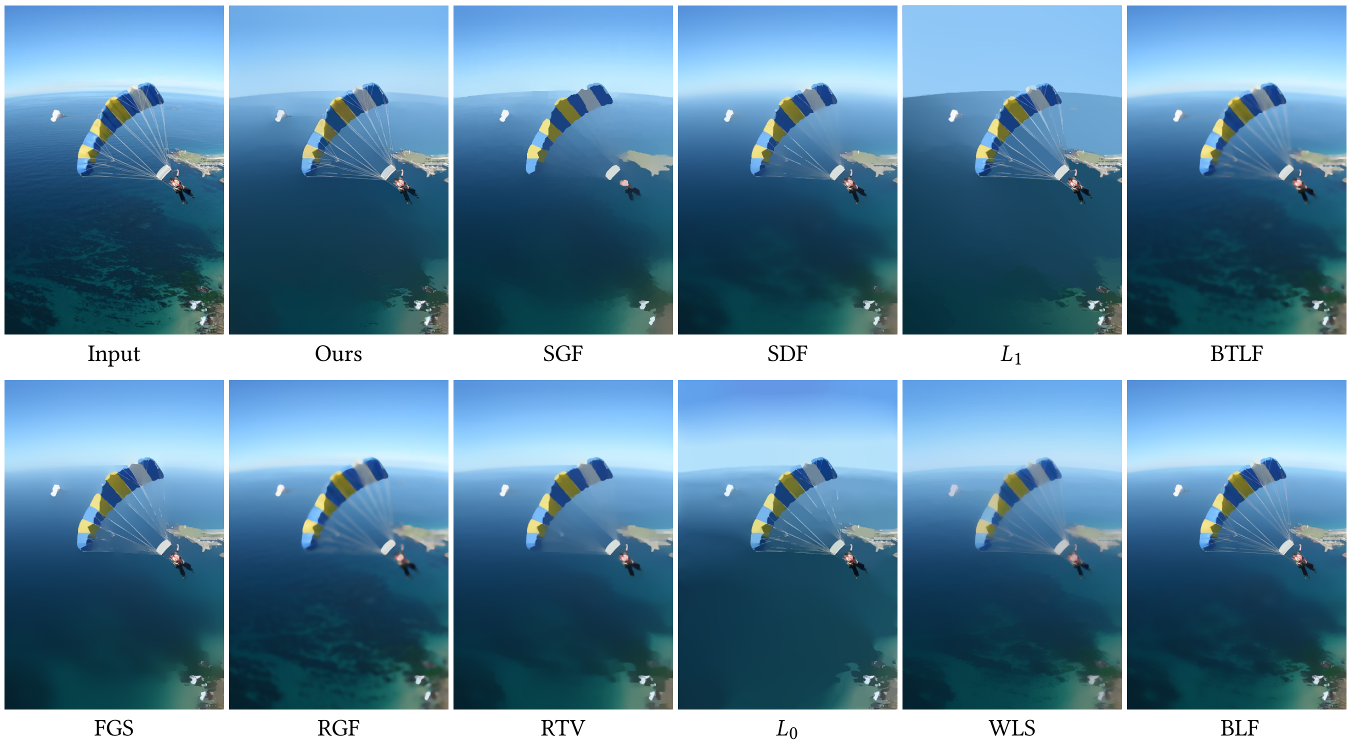

Fig. 2. Visual comparison between our method and previous image smoothing methods, abbreviated as SGF [Zhang et al. 2015], SDF [Ham et al. 2015], L_1 [Bi et al. 2015], BTLF [Cho et al. 2014], FGS [Min et al. 2014], RGF [Zhang et al. 2014], RTV [Xu et al. 2012], L_0 [Xu et al. 2011], WLS [Farbman et al. 2008] and BLF [Tomasi 1998]. Our smooth image is generated by depressing the low-amplitude details and preserve the high-contrast structures. It can be seen that in addition to achieving pleasing flattening effects, the slender ropes of the parachute are also maintained much better in our result than the others. **Zoom in** to see the details. Photo courtesy of Skydive Australia.

images generated by existing smoothing algorithms. In contrast, our neural network is trained by optimizing an objective function through deep neural network in an unsupervised fashion. Note these previous deep models and ours are fundamentally different in many fields: target goal, training data, essential algorithm logic, *etc.* Since they aimed at approximating traditional image smoothing algorithms, while ours creates some novel and unique smoothing effects, it makes our results not directly comparable to theirs by the quality of smooth images.

Deep learning has been applied to many image manipulation tasks [Chen et al. 2017,b, 2018; Fan et al. 2018; He et al. 2018]. But most previous work treat deep learning as a regression or classification tool. In this paper, we apply deep neural network as an optimization solution in a label-free setup.

3 APPROACH

In this section, we introduce our proposed formulation, including an edge-preserving criterion in Section 3.1 and a spatially adaptive L_p flattening criterion in Section 3.2 which account for structure preservation and detail elimination respectively. Later on we describe how deep learning is leveraged for optimizing the proposed objective in Section 3.3.

3.1 Objective Function Definition

Image smoothing aims at diminishing unimportant image details while maintaining primary image structures. To achieve this using energy minimization, our overall energy function for image smoothing is formulated as

$$\mathcal{E} = \mathcal{E}_d + \lambda_f \cdot \mathcal{E}_f + \lambda_e \cdot \mathcal{E}_e, \tag{1}$$

where $\mathcal{E}_d$ is the data term, $\mathcal{E}_f$ is the regularization term and $\mathcal{E}_e$ is the edge-preserving term. λ_f and λ_e are constant balancing weights.

The data term minimizes the difference between the input image and the smoothed image to ensure structure similarity. Denoting the input image by I and the output image by T, both in RGB color space, a simple data term can be defined as

$$\mathcal{E}_d = \frac{1}{N}\sum_{i=1}^{N} ||T_i - I_i||_2^2, \tag{2}$$

where i denotes pixel index and N is the total pixel number.

Some important edges may be missed or weakened during the smoothing process since the goal of color flattening naturally conflicts with edge preserving to some extent. To address this issue, we propose an explicit edge-preserving criterion which preserves important edge pixels.

Before presenting this criterion, we first introduce the concept of *guidance image*, which is formulated as the edge response of an image in appearance. A simple form of edge response is the local gradient magnitude:

$$E_i(I) = \sum_{j \in \mathcal{N}(i)} |\sum_c (I_{i,c} - I_{j,c})| \tag{3}$$

where $\mathcal{N}(i)$ denotes the neighborhoods of point i and c denotes the color channel of the input image I. A similar guidance edge map of the output smooth image T can also be calculated as $E(T)$.

Our edge-preserving criterion is defined by minimizing the quadratic difference of their edge responses between the guidance edge images $E(I)$ and $E(T)$. Let B be an binary map where $B_i = 1$ indicates an important edge point and 0 otherwise, our edge-preserving term is defined as

$$\mathcal{E}_e = \frac{1}{N_e} \sum_{i=1}^{N} B_i \cdot ||E_i(T) - E_i(I)||_2^2 \tag{4}$$

where $N_e = \sum_{i=1}^{N} B_i$ is the total number of important edge points.

The definition of "important edges" is more subjective and varies across different applications. The ideal way to obtain binary maps B would be manual labeling with user preference. However, pixel-level manual labeling is rather labor-intensive. In this paper, we leverage a heuristic yet effective method to detect edges. Since this process is not our main contribution, we defer the detailed description of this edge detector to the supplemental material. A few examples of the detected major image structure are shown in Section 7.1. Also note that any previous advanced and sophisticated edge detection algorithm can all be certainly incorporated based on user preference.

Given sufficient training images with classified edge points, the deep network will implicitly learn the edge importance through minimizing the edge-preserving term and reflect such information in the smooth images. Figure 3 demonstrates an extremely difficult case of a parachute. As can be seen, without the edge-preserving criterion, some thin yet semantically important structures like the white rope are smoothed out in the output images. In contrast, the result optimized with our full criterions maintains these structures very well.

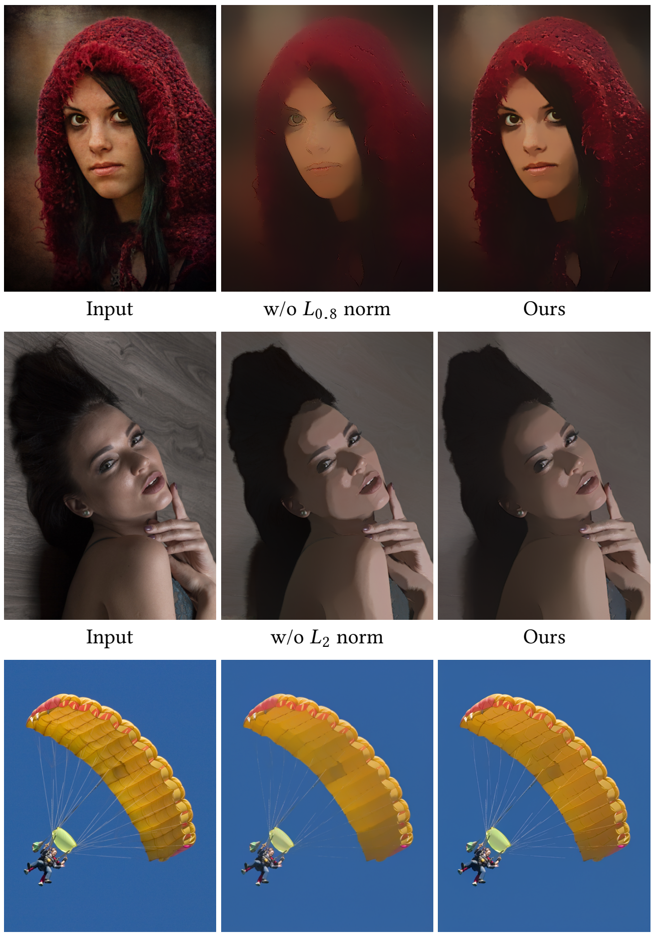

Fig. 3. The effectiveness of our proposed criterions. Note each individual part is essential to generate our visually-pleasing smooth results. **Zoom in** to see the details. Photo courtesy of Flickr user Michael Miller, Andre Wislez and Sheba_Also.

3.2 Dynamic Spatially-Variant L_p Flattening Criterion

We now present our new smoothness/flattening term with spatially-variant L_p norms on the image in order to gain better quality and more flexibility.

To remove unwanted image details, the smoothing or flattening term advocates smoothness for the solution by penalizing the color differences between adjacent pixels:

$$\mathcal{E}_f = \frac{1}{N} \sum_{i=1}^{N} \sum_{j \in \mathcal{N}_h(i)} w_{i,j} \cdot |T_i - T_j|^{p_i}, \tag{5}$$

where $\mathcal{N}_h(i)$ denotes the adjacent pixels of i in its $h \times h$ window, $w_{i,j}$ denotes the weight for the pixel pairs and $| \cdot |^p$ is an L_p norm[1].

The weight $w_{i,j}$ can be calculated from either color affinity or spatial affinity (or their combination), which are defined respectively as

$$w_{i,j}^r = \exp(-\frac{\sum_c (I_{i,c} - I_{j,c})^2}{2\sigma_r^2}), \tag{6}$$

$$w_{i,j}^s = \exp(-\frac{(x_i - x_j)^2 + (y_i - y_j)^2}{2\sigma_s^2}), \tag{7}$$

where σ_r and σ_s are the standard deviations for the Gaussian kernels computed in either color space or spatial space, c denotes image channel (in this paper we use the YUV color space to compute weights $w_{i,j}^r$ in Equation 6), and x, y denote pixel coordinates.

Determining the image regions for different L_p regularizers is not trivial. To help locate these regions in our algorithm, we leverage the guidance images to define the value of p_i and its corresponding weight for each image pixel as

$$p_i,\ w_{i,j} = \begin{cases} p^{large},\ w_{i,j}^s & \text{if } E_i(I) < c_1 \text{ and } E_i(T) - E_i(I) > c_2, \\ p^{small},\ w_{i,j}^r & \text{otherwise.} \end{cases} \tag{8}$$

[1]With slight abuse of terminology, we use L_p norm to refer to the L_p norm raised to the p-th power, *i.e.*, it will indicate $(| \cdot |^p + \cdots + | \cdot |^p)$ as opposed to $(| \cdot |^p + \cdots + | \cdot |^p)^{\frac{1}{p}}$.

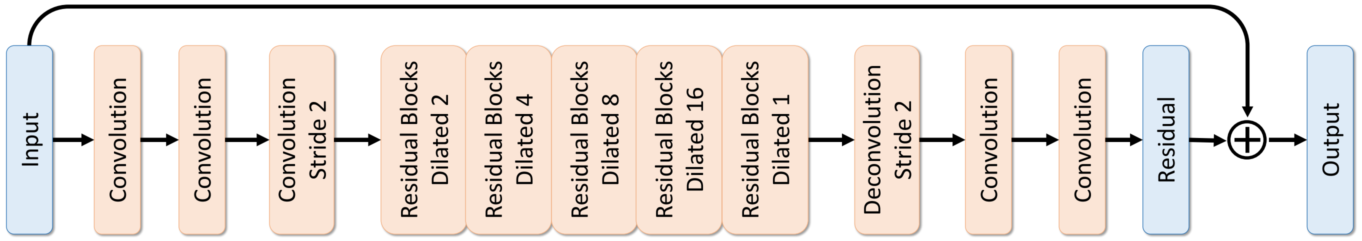
Input
Convolution
Convolution
Convolution Stride 2
Residual Blocks Dilated 2
Residual Blocks Dilated 4
Residual Blocks Dilated 8
Residual Blocks Dilated 16
Residual Blocks Dilated 1
Deconvolution Stride 2
Convolution
Convolution
Residual
Output

Fig. 4. The network structure used throughout this paper. Our network contains 26 convolution layers where the middle 20 layers are organized into residual blocks of dilated convolutions with exponentially increasing dilation factor (except for the last residual block) to enlarge the receptive field. The skip connection from the input to the output is employed such that the network learns to predict a residual image.

where p^{large} and p^{small} are two representative values for p, and c_1 and c_2 denote two positive thresholds. We set $p^{large} = 2$ and $p^{small} = 0.8$ throughout this paper. It can be seen that the p value distribution is not determined *a priori* with the input image, but is conditioned on the output image. We explain such a strategy in the following two points.

Suppressing artifacts caused by single regularizer. The intuition behind Equation 8 is that when we minimize the energy function, $L_{0.8}$ norm is applied until some over-sharpened spurious structures appear in the output image due to the piecewise constant effect caused by $L_{0.8}$ regularizer, at which time L_2 norm will be applied to suppress the artifact. These spurious structures are identified as the ones whose edge response of pixel i on the original image I is low (as characterized by $E_i(I) < c_1$) but is significantly heightened on the output image T (per $E_i(T) - E_i(i) > c_2$). In Figure 3, we demonstrate a few smooth results optimized with our objective function. As can be seen, without L_2 norm, it achieves strong smoothing effects but also yields staircasing artifacts on the lady's cheek and shoulder. On the other hand, without $L_{0.8}$ norm, the optimized image is very blurry and many important structures are not well preserved due to the L_2 regularizer. On the contrary, the results optimized with our proposed full criterions are much more visually pleasing.

Enabling different applications via specialized guidance images. Our guidance image and spatially-variant L_p flattening norm also enable us to achieve flexible smoothing effects for different applications. For example, if the goal is to remove a certain type of image structures like small-scale textures, we can simply eliminate all the edge points belonging to these textures in the guidance image $E(I)$ by setting their values to zero. This way, the edge responses $E(T)$ on these regions of the output image will always be larger, and L_2 norm will be applied to remove these textures. Later we show two such applications – texture removal (Section 7.3) and content-aware image manipulation (Section 7.4). All other results shown in this paper are obtained by raw guidance images computed via Equation 3.

Note that we adopt spatial affinity to calculate the weights w_s for regions with an L_2 norm, as it is more effective for edge suppression. Color affinity is utilized for $L_{0.8}$ norm regions for better flattening effect. Since L_2 and $L_{0.8}$ norm regularize the images differently, we amplifies the weight of L_2 norm with a scale scalar α for balance. We empirically determine these two p values, which represent the regularization for strong flattening and blurring effects in a general sense. They are replaceable with other alternatives.

It can be seen that our spatially variant L_p norm is not fixed, but *dynamically changing* in the iterative optimization (training) procedure based on the output image. Although we do not provide a theoretical proof of convergence, we have found empirically that such a procedure converges and the p value distribution stabilizes in the end. Note we observe that [Zhu et al. 2014] also employs data-guided sparsity in their work, but differently their regularization is static while ours is dynamically changed from the output image.

3.3 Deep Learning based Optimization

As the whole objective function is derivative to the optimized smooth image, we implement it as the loss layer in a deep learning framework. The loss function is optimized with gradient descent method through a deep neural network. The whole training process is in an unsupervised learning fashion with a large number of unlabeled natural images. The deep network implicitly learns the optimization procedure and once the network is trained, it only requires one forward pass through the deep neural network to predict the smooth image without further optimization steps.

Now we introduce architecture of our deep neural network which is used for minimizing the defined energy function. Inspired by the previous work [Yu and Koltun 2016] which enlarges the receptive field with dilated convolutions for semantic segmentation, and [Kim et al. 2016] which uses very deep convolutional neural network equipped with residual learning for super-resolution, we design a fully convolutional network (FCN) equipped with dilated convolution and skip connections for our task.

Basic structure description. Figure 4 is a schematic description of our FCN. The network contains 26 convolution layers, all of which use 3×3 convolution kernels and outputs 64 feature maps (except for the last one which produces a 3-channel image). All the convolution operations are followed by batch normalization [Ioffe and Szegedy 2015] and ReLU activation except for the last layer. The third conv layer downsamples the feature maps by half via using a stride of 2, and the third from last layer is a deconvolution (aka fractionally-strided convolution) layer recovering the original image size. The middle 20 conv layers are organized as 10 residual blocks [He et al. 2016]. A full description of the detailed network structure is presented in the supplemental material.

Large receptive field in dilated convolutions. As image smoothing requires contextual information across wide regions, we increase the receptive field of our FCN by using dilated convolution with *exponentially increasing dilation factors* except for the last residual block, similar to [Yu and Koltun 2016]. Specifically, any two consecutive residual blocks share one dilation factor, which is doubled in the next two residual blocks. This is an effective and efficient strategy to increase receptive field without sacrificing image resolution: with exponentially increasing dilation factors, all the points in an $n \times n$ grid can be reached from any location in logarithmic steps $log(n)$. Similar strategies have been used in parallel GPU implementation of some traditional algorithms, such as Voronoi diagram [Rong and Tan 2006] and PatchMatch [Fan et al. 2015].

Residual image learning. In the image smoothing task, the input and output images are highly correlated. In order to ease the learning, instead of directly predicting a smoothed image we predict a *residual image* and generate the final result via point-wise summation of the residual image and the raw input image. Such a residual image learning design avoids the color attenuation issue observed in previous works (*e.g.*, [Kim et al. 2016]).

4 IMPLEMENTATION DETAILS

Our FCN network and energy function are implemented in the Torch framework and optimized with mini-batch gradient descent. The batch size is set as 1. The network weights are randomly initialized using the method of [He et al. 2015]. The Adam [Kingma and Ba 2015] algorithm is used for training with the learning rate set as 0.01. We train the network for 30 epoches, which takes about 8 hours on an NVIDIA Geforce 1080 GPU.

Training and testing data. Since our network does not require ground-truth smooth image for training, any image can be used to train it. For better generalization to natural images, we use the PASCAL VOC dataset [Everingham et al. 2010] which contains about 17,000 images to train the network. These images were collected in the Flicker photo-sharing website and exhibit a wide range of scenes under a large variety of viewing conditions. We crop the images to the size of 224×224 to accelerate the training process without jeopardizing the smoothing quality. Once the network is trained, we run it on images outside of PASCAL VOC and evaluate the results.

Parameter specifics: The parameters in our proposed objective function are specified by default as follows: 0.1 (σ_r), 7 (σ_s), 1 (λ_f), 0.1 (λ_e), 5 (α), 20 (c_1), 10 (c_1), 21 (h). To achieve the optimal performance on each individual application, a small subset of parameters may be tweaked, which is discussed in Section 7. However, note that these parameters are only tuned based on the application type, not on any particular images. All the images in the same application shown in both the paper and supplemental material are generated by the same set of parameter values.

5 METHOD ANALYSIS AND DISCUSSION

In this section, we first compare the smooth images optimized with our deep learning solver and traditional numerical solvers, followed by the analysis of convergence of different optimizers and the potential reason why the deep learning solver achieves more visually pleasing results than the others for our problem.

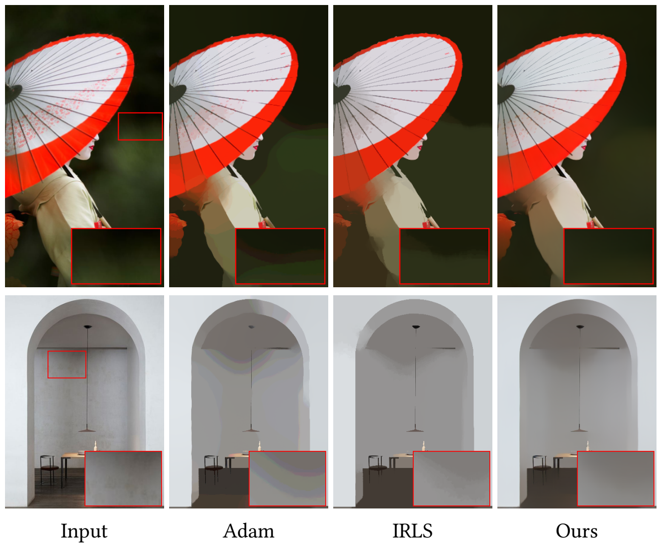

Fig. 5. Comparison between our learned deep neural network and traditional numerical solvers. Compared with the gradient descent optimizer Adam [Kingma and Ba 2015] and Iterative Reweighted Least Square (IRLS) [Holland and Welsch 1977], our results are visually more pleasing which do not have the spurious staircasing structures. Photo courtesy of Tumblr user gaaplite and LucidiPevere Studio.

5.1 Visual results of different optimizers

To verify the efficacy of our deep learning solver, as opposed to directly minimizing Equation 1 using a traditional optimization algorithm, we compare it against two popular representative approaches, Adam and IRLS. Adam [Kingma and Ba 2015] is a stochastic gradient descent-based optimization method that can be generically applied to nonconvex differentiable functions. Since our proposed objective is differentiable[2], Adam is a very straightforward approach for minimizing it, at least locally. Likewise, iterative reweighted least square (IRLS) [Holland and Welsch 1977] represents a classical tool for minimizing energy functions with non-quadratic forms. Several image smoothing papers [Min et al. 2014; Xu et al. 2012] also employ IRLS allied with objectives regularized by L_p norms ($0 < p \leq 1$).

To utilize IRLS for optimization, a tight quadratic upper bound of the energy has to be defined, which is trivial to accomplish for L_p terms as has been done in the past. However, the proposed non-quadratic edge-preserving criterion cannot be bounded in this way, making IRLS problematic. Therefore for the results reported in this section, the energy function is formed from only the data term and the spatially adaptive L_p flattening term for fair comparison across all three methods. Smooth images optimized by different solvers are shown in Figure 5. Note that Adam, as a gradient-based method requires 100 iterations to converge, while IRLS only requires

[2]Although L_p norms are not differentiable on a set of measure zero, in such cases subgradients can easily serve as a natural surrogate

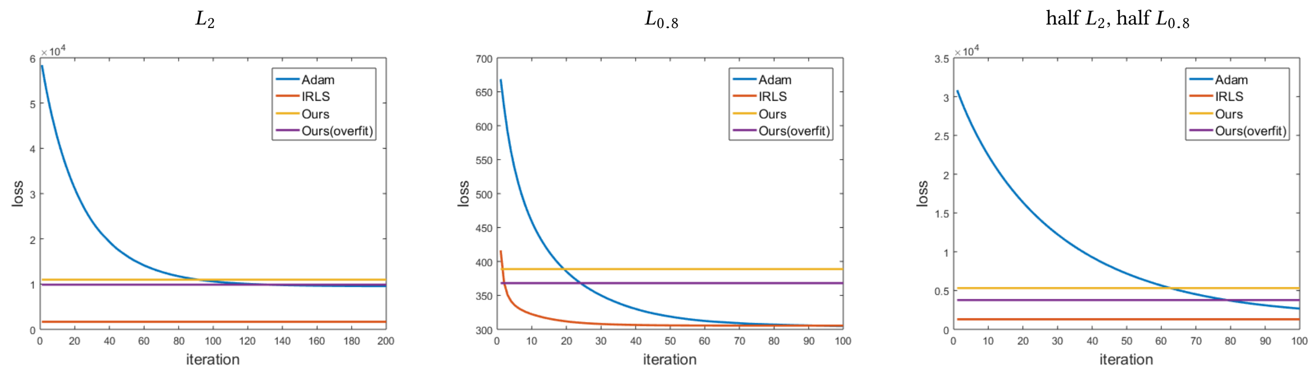

Fig. 6. The loss values from four optimization solvers (IRLS, Adam, our deep learning solver and the deep learning solver overfitted on a single image) for objective functions that contain either L_2 norm, $L_{0.8}$ norm or mixed of these two in the flattening criterion. Note the loss is computed in the evaluation stage; our deep learning solver is non-iterative thus the loss is constant (as represented by a horizontal line).

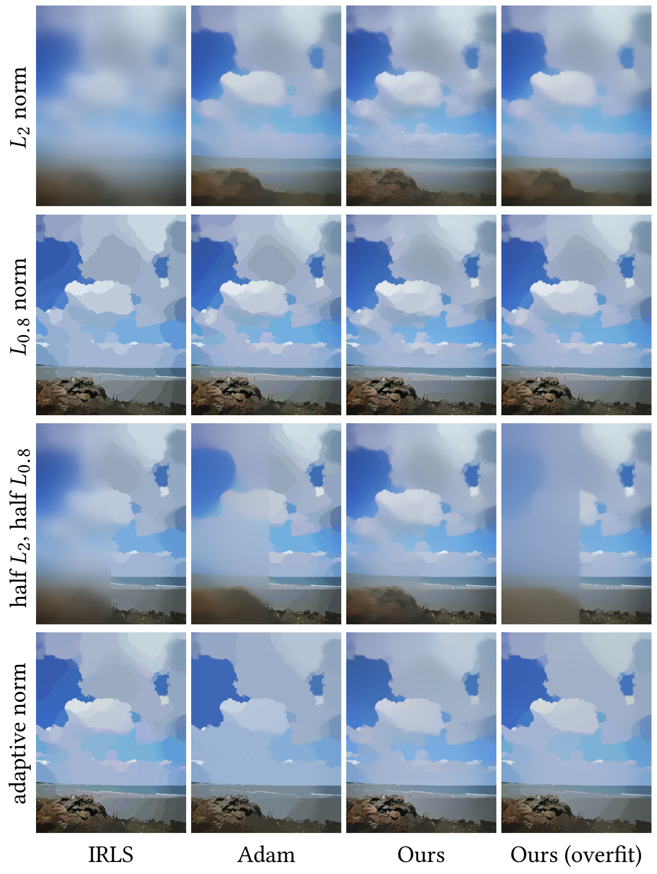

Fig. 7. Demonstration of the results optimized with L_2 norm, $L_{0.8}$ norm, half L_2 half $L_{0.8}$ and our adaptively-changed L_p norms. **Zoom in** to see the detailed difference. Photo courtesy of Flickr user Ryan L Collins.

about 10 iterations given that it applies second-order information in optimizing the quadratic upper bound.

In general, both the Adam and IRLS results are less satisfactory than our deep neural network solver. For example, with Adam some spurious stair-casing edges are still generated as undesirable visual artifacts. Likewise, for IRLS we observe over-sharpened side effects, although in places not quite as severe as with Adam. Also, the magnified local regions shown in the bottom of each figure display areas where the color intensity varies gradually, and both Adam and IRLS fail to smooth these areas well. Also, the magnified local region in each figure shows that both Adam and IRLS cannot well smooth the local regions where color varies gradually.

5.2 Performance analysis of different optimizers on fixed L_p distribution

Now we analyze the performance of these three optimizers by comparing their convergence curves. Note since our proposed objective function is adaptively changing based on the output images (per Equation 5 and 8), it is not intuitive for comparing the convergence trend of these different optimizers. Thus we test three representative loss functions with fixed L_p distribution map in $\mathcal{E}_f$, whose optimization difficulty is gradually increasing. They are the variants of our objective function, where we replace the adaptively changed regularizer with only L_2 norm, only $L_{0.8}$ norm, or fixed combination of these two. Following the previous ablation study, we disable the edge preserving term for IRLS. The loss values are averaged over 40 test images, and are shown in Figure 6. The corresponding visual results are shown in Figure 7. Note these loss curves do not illustrate the training process of our method. Instead, they are constant values computed on the testing images.

With the fixed L_2 norm in $\mathcal{E}_f$, the whole objective function becomes convex and quadratic. Thus IRLS is able to achieve the optimal results with one step. Adam is slightly better than our learned deep network. However, the smoothing results of both methods are relatively far from optimal. Accordingly to Figure 7, IRLS demonstrates the most blurry images regularized by L_2 norm. When the objective function contains $L_{0.8}$ norm in $\mathcal{E}_f$, it becomes nonconvex. From the middle loss figure, we can see IRLS and Adam achieves similar energy value in the end, and the deep learning solver does not obtain a loss value as low as theirs. Figure 7 shows that the smoothing results of all the methods appear to be more piece-wise constant. Finally, we demonstrate a case where the flattening criterion $\mathcal{E}_f$ contains L_2

norm in left half of the image and $L_{0.8}$ norm in right half. Different from our dynamically-changed L_p norm, the L_p distribution in this case is fixed for all different images and iterations. Figure 6 shows that IRLS achieves the lowest energy value[3], followed by Adam and deep learning solver.

To understand why the traditional optimizers achieve lower energy values on the above three objective functions, we first illustrate the workflow of both traditional numerical solvers and our deep learning solver in Figure 8. As can be seen, given a particular image, traditional numerical solvers work by iteratively optimizing this single image. In contrast, the deep learning solver *directly predicts their results from a one-forward-pass mapping, which is learned based on a large corpus of training data without any prior information on this specific image.* Thus traditional optimizers are more advantageous in achieving lower energy with fixed L_p distribution, as verified by the above three loss curves.

To further analyze the above hypothesis, we overfit our network by training on only one single image, and compare both the final loss and the visual results. This way, the deep learning solver works similarly to traditional solvers. As can be seen, our deep learning solver with overfitting is able to achieve much lower energy, especially in the case of L_2 norm where our overfitting results are almost identical to Adam. Likewise, Figure 7 shows that the visual result obtained with the overfitting solver are visually much closer to IRLS and Adam. For example, there is a clear separation line between the two regularized regions for the half L_2 half $L_{0.8}$ case, which is not present in the results of the original deep learning solver.

Note the loss functions defined in this subsection are used to analyze the performance of different solvers. They are not the actual loss function used for our image smoothing task.

5.3 Performance analysis of different optimizers on our dynamic L_p distribution

In our proposed edge flattening criterion, the L_p distribution is adaptively changing based on the output images. The whole objective function is highly complex and loss curve is not guaranteed to converge. Therefore, comparing the loss curves of the different solvers is less informative. The last row of Figure 7 shows an additional set of results obtained under our adaptively changing L_p norm. It can be observed the staircasing artifacts exist in the results of all IRLS, Adam, and the overfitting-based deep learning solver. In contrast, our deep learning solver generates very smooth results with no such artifacts.

Given each specific L_p distribution map in each iteration, the traditional numerical solvers still tend to "overfit" that unique distribution map for its optimal results, which accordingly results in some spurious edges that separate these different regularizations just like the case of half L_2 half $L_{0.8}$ norm. In contrast, the disadvantage of the deep learning solver exposed in Section 5.2 becomes an advantage in the presence of a dynamically changed L_p norm distribution. *Benefited from the large corpus of training data, the deep learning solver incorporates the learned implicit combination of L_2 and $L_{0.8}$ norm into the deep network and reflects such combination, instead of a fixed regularizer, into each pixel of the smoothed images.* It is able to generates more visually pleasing results, as shown in both Figure 5 and 7.

[3]To make the results more presentable, we slightly modified the objective function for IRLS and only show its final loss in this case.

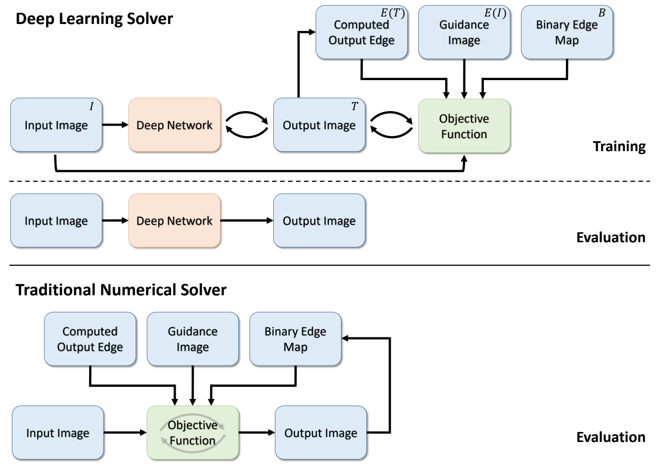

Fig. 8. Workflow of our deep learning solver compared to traditional numerical solvers when applied to the proposed objective function. Our method employs a neural network to optimize an objective function with *no ground truth labels* of smooth images. In the training stage, a few extra inputs are required to minimize the objective function; but once the network is train, the input image is only required to forward through the deep network once to predict smooth images in the evaluation stage. Regarding the traditional numerical solvers, given *each new image*, all the different inputs are required to iteratively optimize the objective function.

Therefore, we argue that what matters to solve the proposed objective function and obtain better smoothing result is the joint optimization over large corpus of images, instead of any particular image. In this specific problem, the deep learning solver plays a critical role. Note that although many empirical experiments have been conducted above, rigorous theoretical analysis is still lacking. Understanding and explaining deep neural networks are still open problems for the follow-up research.

6 EXPERIMENTAL RESULTS

In this section, we first conduct some ablation study to analyze the influence of the parameters and network structures to the results. Afterwards, we compare our results with previous methods in both visual quality and running time efficiency.

6.1 Ablation Study

Effect of parameter control. In out method, the main parameters to tune are λ_f and λ_e. Here we analyze the results of our network trained under different settings of these two parameters, and such a group of visual results are shown in Figure 9. As can be seen from the first row, altering weight of edge-preserving term λ_e influences the image structures. From the second row, tweaking the weight for flattening term λ_f controls the smoothness of predicted images. While enhancing the smoothness with a large λ_f, we observe gradually destructed structures, *e.g.* the ground tiles.

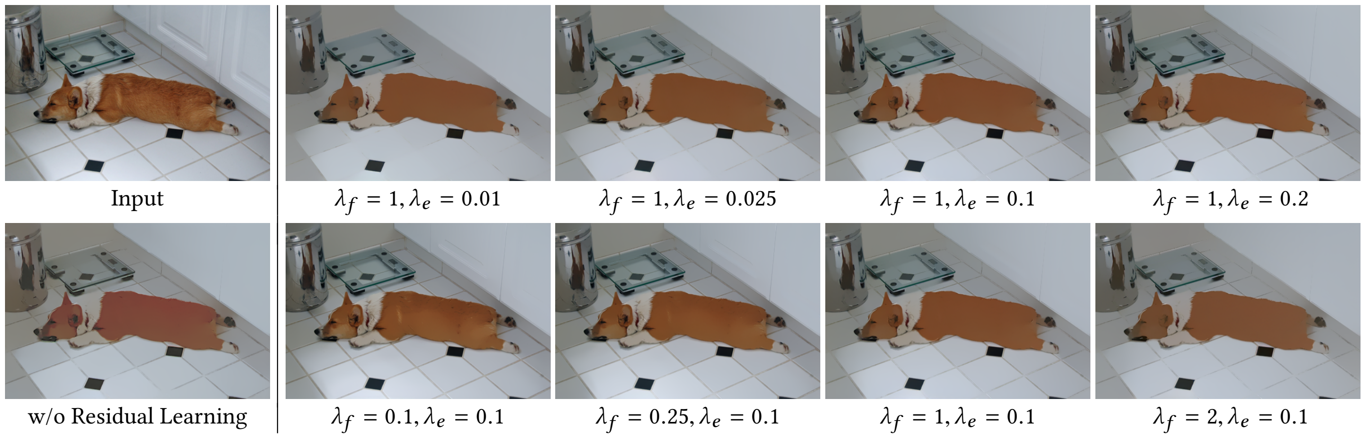
Input
$\lambda_f = 1, \lambda_e = 0.01$
$\lambda_f = 1, \lambda_e = 0.025$
$\lambda_f = 1, \lambda_e = 0.1$
$\lambda_f = 1, \lambda_e = 0.2$
w/o Residual Learning
$\lambda_f = 0.1, \lambda_e = 0.1$
$\lambda_f = 0.25, \lambda_e = 0.1$
$\lambda_f = 1, \lambda_e = 0.1$
$\lambda_f = 2, \lambda_e = 0.1$

Fig. 9. Effect of tuning parameters (Right) and ablation study for our residual learning (Left). The first row shows that with larger λ_e, we are able to maintain some important image structures. The second row shows that via adjusting λ_f, we can gradually change the smoothness strength. As can be seen on the left, the result predicted by the model without residual connection exhibits some noticeable color attenuation problems, which does not exist in our results with residual image learning. Photo courtesy of Flickr user Rachel Hinman.

	Traditional smoothing algorithms									Approximation CNN			Ours	
	SGF	BLF	RGF	Tree	L_0	RTV	WLS	SDF	L_1	Xu	Liu	Fan	GPU	CPU
QVGA (320×240)	0.05	0.03	0.22	0.05	0.17	0.41	0.70	4.99	32.18	0.23	0.07	0.008	0.003	0.010
VGA (640×480)	0.15	0.12	0.73	0.42	0.66	1.80	3.34	19.19	212.07	0.76	0.14	0.009	0.004	0.011
720p (1280×720)	0.25	0.34	1.87	2.08	2.43	5.74	13.26	66.14	904.36	2.16	0.33	0.010	0.005	0.012

Table 1. Running time comparison between different methods (in seconds). Our method is significantly faster than the traditional methods (SGF [Zhang et al. 2015], BLF [Tomasi 1998], RGF [Zhang et al. 2014], Tree Filter [Bao et al. 2014], L_0 [Xu et al. 2011], RTV [Xu et al. 2012], WLS [Farbman et al. 2008], SDF [Ham et al. 2015] and L_1 [Bi et al. 2015]), especially those based on optimization (L_0, RTV, WLS, SDF, L_1). It is also much faster than the recent methods [Fan et al. 2017; Liu et al. 2016; Xu et al. 2015] that train convolutional neural networks (CNN) as a regression tool to approximate traditional smoothing approaches. Due to lack of current GPU implementation of most traditional methods, their running time is evaluated under a modern CPU. The deep learning based methods are evaluated on GPU, meanwhile we also report the CPU time of our network structure with efficient multi-core implementation.

Effect of residual image learning. We analyze the importance of residual image learning by comparing the results with and without this component in our network. As shown in Figure 9, the smooth image generated without residual learning contains some obvious color degradation issues. It appears more orange compared to the raw input image. In contrast, the smooth images predicted with our complete network structure with residual image learning dose not have this issue, as shown in Figure 9. For the image smoothing task, the input and output image should be highly correlated. However, it can be difficult to maintain well the color information in the image after many convolution operations in a deep neural network like ours. Thus we propose to learn the residual image and combine it with input image to resolve this issue.

6.2 Comparison with Previous Methods

We compare the proposed method with previous ones in terms of the smoothing effect and speed. More comparisons on different applications can be found in Section 7.

Smoothing effect comparison. Figure 2 compares the smoothing results of our method and ten existing methods: [Zhang et al. 2015] (SGF), [Ham et al. 2015] (SDF), [Bi et al. 2015] (L_1), [Cho et al. 2014] (BTLF), [Min et al. 2014] (FGS), [Zhang et al. 2014] (RGF), [Xu et al. 2012] (RTV), [Xu et al. 2011] (L_0), [Farbman et al. 2008] (WLS) and [Tomasi 1998] (BLF). Note that these algorithms may be designed for different applications thus their goals may be slightly different. Compared to these methods under this difficult example, our method produced outstanding edge-preserving flattening result: it not only achieved pleasing flattening effects for regions of low amplitude (*e.g.* the sea), but also well preserved the high-contrast structures, especially the thin but salient edges (*e.g.* the ropes of the paraglider).

To our knowledge, there is no benchmark or dataset to quantitatively evaluate the performance of image smoothing algorithms. Visual perception is still the principal way for evaluation. To demonstrate the robustness of our method and its good performance for natural images with vastly different contents and capturing conditions, we present the visual results on over 100 images without any parameter tweaking for any particular images in the supplemental material.

Running time comparison. Table 1 compares the running time of our method and some previous methods, including both traditional image smoothing algorithms [Bao et al. 2014; Bi et al. 2015; Farbman et al. 2008; Ham et al. 2015; Tomasi 1998; Xu et al. 2011, 2012; Zhang

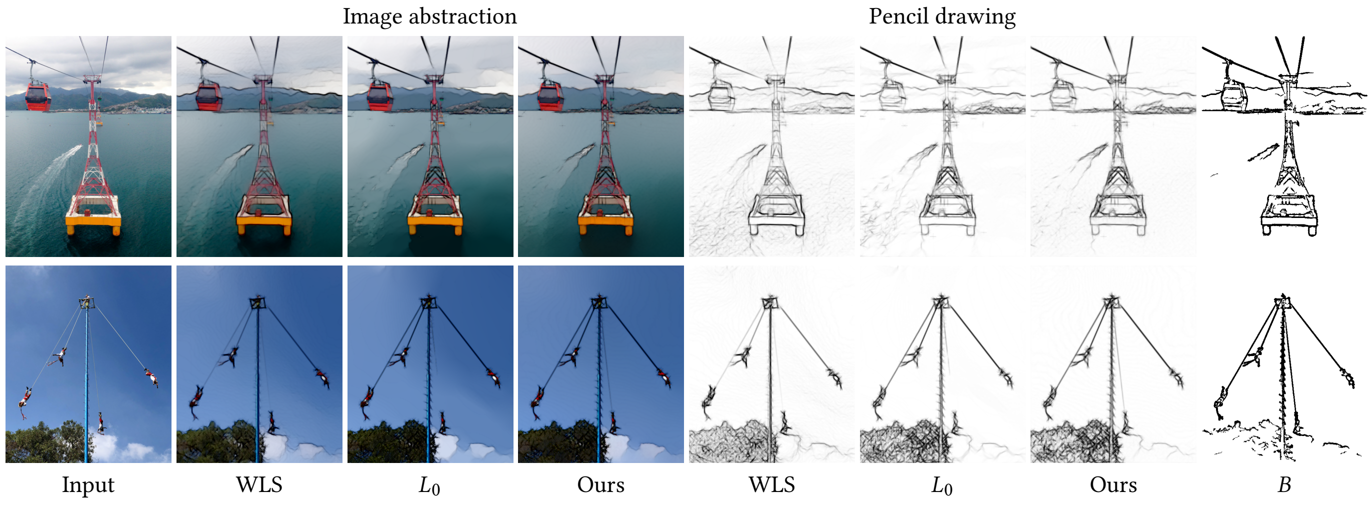

Fig. 10. Comparison of the image abstraction (Column 2-4) and pencil sketching (Column 5-7) results between our method and previous image smoothing methods WLS [Farbman et al. 2008] and L_0 [Xu et al. 2011]. With our learned edge-preserving image smoother, the slender but important image structures such as the cable wires and hanging ropes are well preserved, rendering the stylized images more visually pleasing. Moreover, we also demonstrate the binary edge map B detected by our heuristic detection method, which shows consistent image structure with our style image. Note that binary edge maps are only used in the objective function for training; they are not used in the test stage and are presented here only for comparison purpose. Photo courtesy of Flickr user gavindeas and MollySVH.

et al. 2015, 2014] and some recent methods [Fan et al. 2017; Liu et al. 2016; Xu et al. 2015] that apply neural networks to approximate the results of the existing smoothing algorithms. Traditional image smoothing methods are based on either filtering techniques [Bao et al. 2014; Ham et al. 2015; Tomasi 1998; Zhang et al. 2014] or mathematical optimization [Bi et al. 2015; Farbman et al. 2008; Ham et al. 2015; Xu et al. 2011, 2012]. While the latter category draws much attention in recent years and often produces quality results, the optimization procedure (*e.g.* solving large-scale linear systems iteratively) can be very time-consuming. For example, the state-of-the-art method of [Bi et al. 2015] takes about 15 minutes to process a 1280×720 image. Compared to these methods, ours runs significantly faster. It takes only a few milliseconds for a 1280×720 image at the aid of GPU. However, even on CPU, with efficient parallel implementation of our network structure[4], it still runs in at most tens of milliseconds, facilitating real-time applications.

Compared to the neural network approximators [Fan et al. 2017; Liu et al. 2016; Xu et al. 2015][5], our method not only generates novel, unique smoothing effects that allow better results in various applications (see Section 7), but also has a faster running speed. Note except for running a neural network, [Xu et al. 2015] also employs a post-processed optimization step and [Liu et al. 2016] leverages a recursive 1D filter, both of which slows down their whole algorithm.

7 APPLICATIONS

In this section, we demonstrate the effectiveness and flexibility of our image smoothing algorithm with a range of different applications including image abstraction, pencil sketching, detail magnification, texture removal and content-aware image manipulation. The tailored methods for these different applications mainly differ in the guidance edge maps used in training the network: the former three applications use the local gradient based edge map (per Equation 3) similar to the previous experiments, while the latter two modify them to achieve particular effects. For all the applications we use the images in the PASCAL VOC dataset [Everingham et al. 2010] for training.

7.1 Image Abstraction and Pencil Sketching

Edge-preserving image smoothing can be used to stylize imageries. For example, [Winnemöller et al. 2006] proposed to abstract imagery by simplifying the low-amplitude details and increasing the contrast of visually important structures with difference-of-Gaussian edges. Following previous works of [Farbman et al. 2008; Xu et al. 2011], we replace the iterative bilateral filter in [Winnemöller et al. 2006] with our learned edge-preserving image smoother, and decorate the extracted edges with random sketches in different directions to generate pencil drawing pictures. Furthermore, the smoothed images are combined with the pencil drawing picture to generate an abstract look [Lu et al. 2012].

Figure 10 presents the image abstraction and pencil sketching results of different methods on two examples. The flatting effects of our method and [Xu et al. 2011] are visually better than [Farbman et al. 2008] (*e.g.*, see the sea in the abstracted results of the first case). More importantly, our method clearly excelled at preserving important image structures, thanks to the proposed energy function which has an explicit edge-preserving constraint. For example, the hanging ropes in the second example can be clearly seen in our smoothing results, whereas they are not very well preserved by other

[4]Our CPU version is implemented in MXNet facilitated with the NNPACK module.
[5]The running time of [Fan et al. 2017] reported in their paper includes both generating images of particular sizes (per Table 1) and running the network. We excluded the former for a fair comparison, thus their numbers are lower than reported.

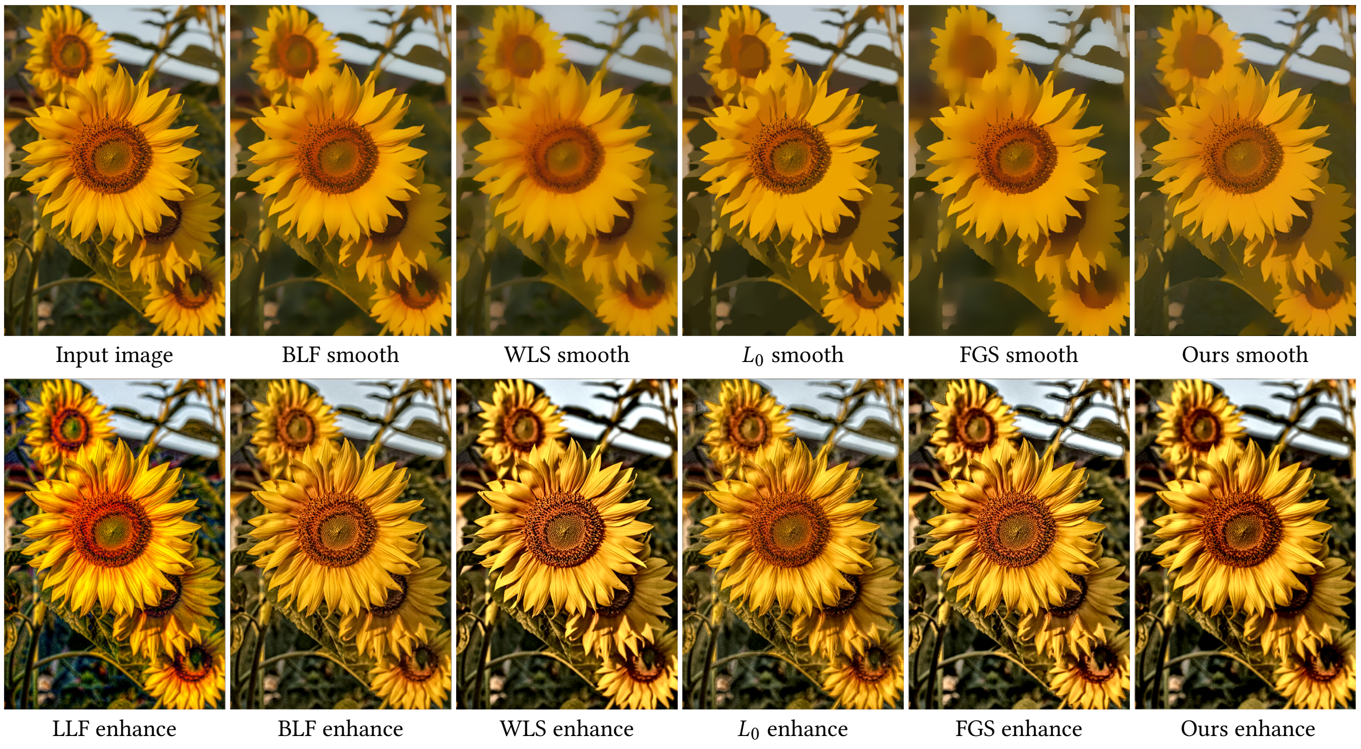

Fig. 11. Detail magnification results of our method compared with previous image smoothing algorithms LLF [Paris et al. 2011], BLF [Tomasi 1998], WLS [Farbman et al. 2008], L_0 [Xu et al. 2011] and FGS [Min et al. 2014]. In this example, the top row shows the smooth base layers obtained via image smoothing, while the bottom one demonstrates the enhancement results. As can be seen, our algorithm does not over-sharpen the image structures in the smooth image and achieves visually pleasing detail exaggeration effects. **Zoom in** to see the details. Photo courtesy of Flickr user Sheba_Also.

methods. Note that these structures are semantically meaningful, without which the images may look strange. The abstraction and pencil sketching results of our method are clearly more satisfactory.

Note to further overcome the over-sharpened effects, we expand the image region regularized by $L_{p^{large}}$ norm to its surrounding 7×7 pixel neighbourhood.

7.2 Detail Magnification

The effect of image detail magnification can be achieved by superposing a smooth base layer and an enhanced detail layer, the latter of which can be obtained by image smoothing algorithms. After extracting the smooth layer, the detail layer can be obtained as the difference between the original image and the smooth layer, which is then enhanced and added back to the smooth layer to generate the final result. An ideal smoothing algorithm for this task should neither blur nor over-sharpen the salient image structures [Farbman et al. 2008], as either operation can lead to "ringing" artifacts in the residual image, resulting in halo or gradient reversals in the detail-enhanced images. Developing such a smooth filter is challenging as it is difficult to determine the edges to preserve and diminish while avoiding to both over sharpen and smooth these edges.

Figure 11 presents the results on such example obtained by our method and previous ones [Farbman et al. 2008; Min et al. 2014; Paris et al. 2011; Tomasi 1998; Xu et al. 2011]. It can be observed that in the smoothed images the methods of [Min et al. 2014; Tomasi 1998; Xu et al. 2011] sharpened some edges that are blurry in the original images due to out of focus. As a result, conspicuous gradient reversal artifacts can be observed clearly on the top of enhanced images. In contrast, [Farbman et al. 2008; Paris et al. 2011] and our method produce better results without noticeable artifacts. Note that the method of [Farbman et al. 2008] applied L_2 regularizer over the entire image in their smoothness term to perfectly avoid over-sharpening the structures. In our approach, the edge-preserving term enforces a strong similarity between the major image structure of input and output images via minimizing their quadratic differences, preventing the edges from being significantly blurred or excessively sharpened. Moreover, the L_2 norm is also partially applied to the potentially over-sharpened regions to better avoid gradient reversal artifacts in the exaggerated image. As such, high-quality detail magnification results can be obtained as shown in Figure 11.

Note the gradient reversal artifacts are very likely to happen even if the smooth image is only slightly over-sharpened by either numerical analysis or visual perception. And such a case is almost unavoidable as for the edge-preserving filters that applies strong regularization, since it always tends to over-sharpen the image more or less. Therefore, we do not argue for the perfect detail exaggeration results, but we are able to outperform most previous algorithms that

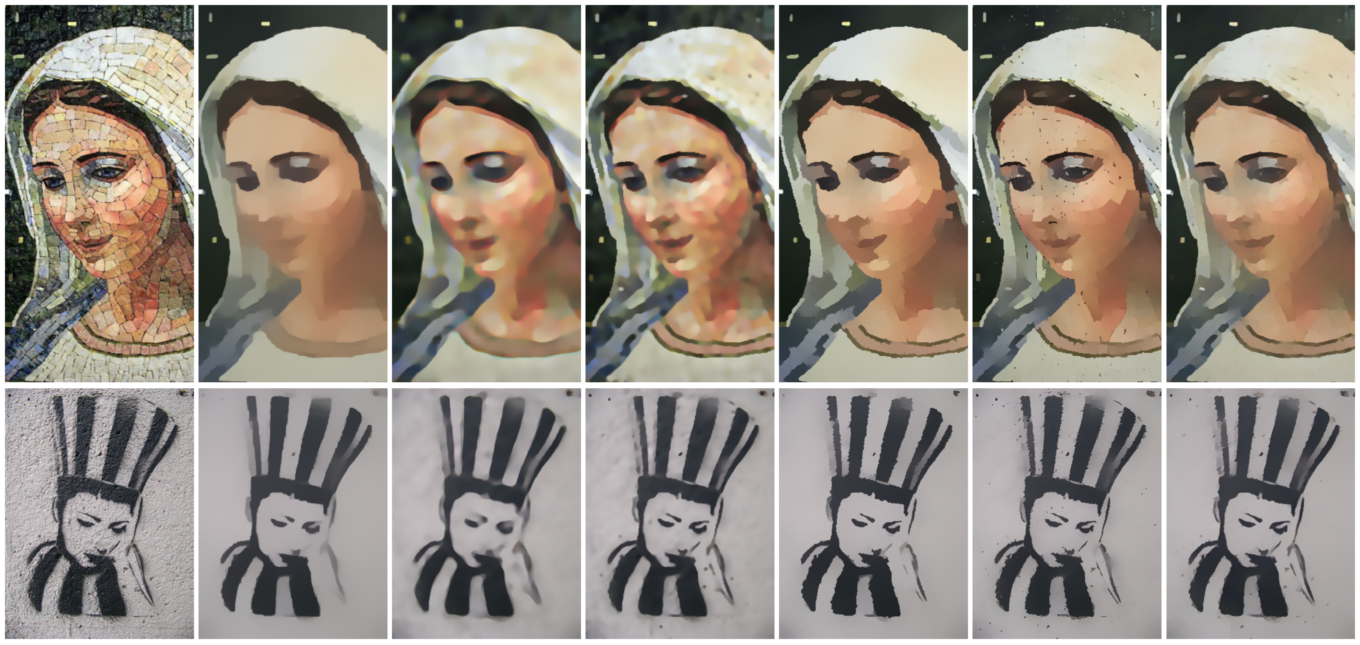

Fig. 12. Texture removal results of our method compared with state-of-the-art methods that address the texture removal problem: RTV [Xu et al. 2012], RGF [Zhang et al. 2014], BTLF [Cho et al. 2014], SGF [Zhang et al. 2015] and SDF [Ham et al. 2015]. Our method can effectively remove the texture patterns meanwhile well preserve the primary image structures. Photo courtesy of Emilie Baudrais and [Xu et al. 2012].

pursue strong smoothing effects [Min et al. 2014; Tomasi 1998; Xu et al. 2011] with only little effort of tweaking our objective function.

To avoid the gradient reversal effects that are usually caused by over-sharpening the smooth layer, we increase the p^{large} balance weight ($\alpha = 15$), and release the constraint on p selection ($c_1 = +\infty, c_2 = 0$).

7.3 Texture Removal

The texture removal task we consider here aims at removing the fine-scale repetitive patterns from primary image structures. In this task, the smoothing algorithms should be made scale-aware, as the textures to be removed may also have local gradients.

Our method can be easily tailored for this task. To grant a network the ability to distinguish fine-scale textures from primary image structures and smooth them out after training, we can simply set the edge responses of the texture points on the guidance image $E(I)$ to be zero. This way, the corresponding edge responses on the guidance map of the output image $E(T)$ will always be larger. Thus with slight modification on the constraint of Equation 8, a L_2 smoothness regularizer can be easily enforced on the texture regions, such that the network will learn to diminish them. The way to obtain the texture structure is elaborated in the supplemental material.

Figure 12 shows two examples that contain different types of texture patterns. We compare our results against state-of-the-art methods of [Cho et al. 2014; Ham et al. 2015; Xu et al. 2012; Zhang et al. 2015, 2014] that address the texture removal problem. It can be seen that both [Zhang et al. 2014] and [Cho et al. 2014] tends to blur some large-scale major structures, while the method of [Ham et al. 2015] produces some noisy structure boundaries. Compared to these methods, superior results are obtained by our method.

Since this task aims at diminishing textures that are very possibly locally salient, we enlarge the p^{large} weight ($\alpha = 20$) and limit the smooth region ($h = 5$).

7.4 Content-Aware Image Manipulation

Different from traditional methods, our proposed algorithm enables us to achieve content-aware image processing, *i.e.*, smoothing a particular category of objects in the image.

In this section, we use the image saliency cue to demonstrate content-aware image manipulations by our method. For example, the proposed objective function can be slightly modified to achieve background smoothing goal, which is smoothing out the background regions for highlighting the foreground (*i.e.*, the salient objects). To this end, we mask out the edge responses of the background regions in the guidance image $E(I)$ via the binary saliency masks obtained by recent salient object detection algorithm [Zhang et al. 2017]. By feeding the modified guidance image $E(I)$ to the proposed objective function, the L_2 norm regularizer can be applied on the background regions during training. Afterwards, we can even set the smoothing weights of foreground regions to relatively small values or even zero to keep the foreground unmodified. Figure 13 presents some example results from our method, from which we can see that our trained network is capable of implementing content-aware image smoothing very well.

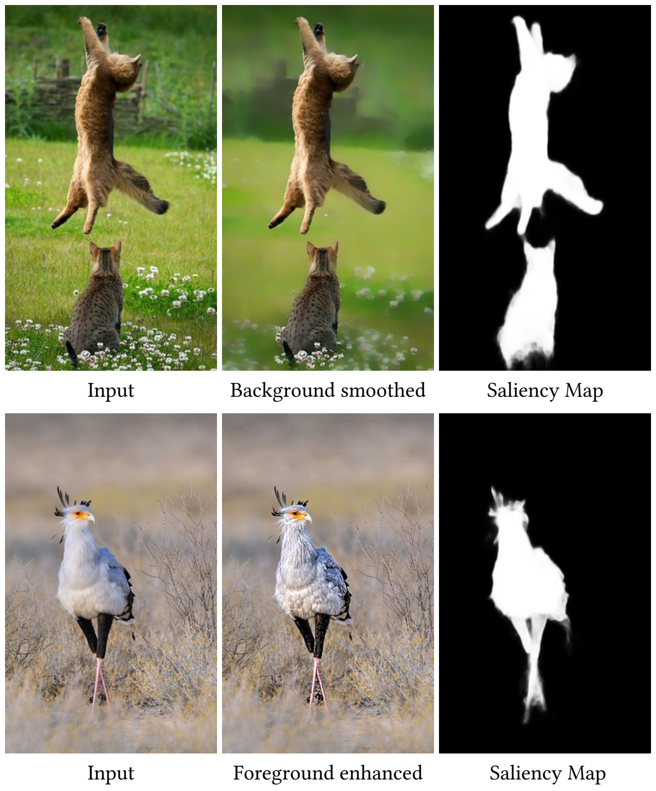

Fig. 13. Content-aware image manipulation with our method. The saliency maps generated by [Zhang et al. 2017] is only employed in the training stage for the optimization goal; they are not used in the test stage and are presented here for comparison purpose. Our method can implicitly learn saliency information and produce quality smoothing results accordingly. Photo courtesy of Lisa Beattie and Albert Froneman.

Alternatively, our algorithm are also able to smooth out foreground regions via a similar strategy, such that a foreground enhancement effect can be achieved via the approach described in Section 7.2. Figure 13 demonstrates very visually-pleasing exaggeration effect for the foreground objects via our approach.

Note in this application, the smoothness effects and saliency information are jointly learned within our network, while the latter information is reflected in the predicted smooth image. All these results are obtained solely by our trained network without any pre- or post- processing. We set h in Equation 5 as 5 to limit the smoothness only within either the foreground or background region.

8 CONCLUSION

In this paper, we have presented an unsupervised learning approach for the task of image smoothing. We introduced a novel image smoothing objective function built upon the mixture of a spatially-adaptive L_p flattening criterion and an edge-preserving regularizer. These criteria not only lead to state-of-the-art smoothing effects as demonstrated in our experiments, but also grant our method the flexibility to obtain different smoothing effects within a single framework. We have also shown that training a deep neural network on a large corpus of raw images without ground truth labels can adequately solve the underlying minimization problem and generate impressive results. Moreover, the end-to-end mapping from a single input image to its corresponding smoothed counterpart by the neural network can be computed efficiently on both GPU and CPU, and the experiments have shown that our method runs orders of magnitude faster than traditional methods. We foresee a wide range of applications that can benefit from our new pipeline.

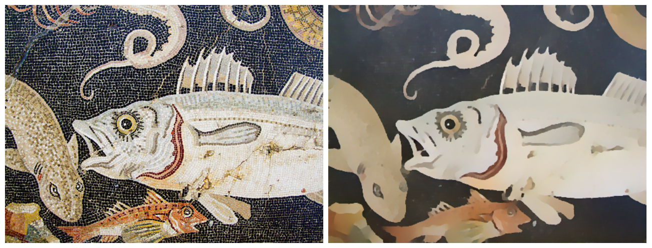

Fig. 14. A partial failure case of texture-removal. Our algorithm doesn't succeed in extracting some detailed textures perfectly such as the eyes of the two smaller fishes. Photo courtesy of Flickr user Chris Beckett.

8.1 Limitations and Future work

Our algorithm relies on some additional information to optimize the objective function during training, such as the detected structures or textures. Currently we employ some simple heuristic methods to detect the structures, and imperfect detection can influence the smoothing results. Figure 14 shows an example where our algorithm fails to extract some detailed textures perfectly. This issue can mitigated by introducing moderate effort of human interaction for refining the structure maps of the training data, or by synthesizing training images with separate textures and clear images similar to [Lu et al. 2018]. Developing more advanced detection algorithms is also one of our future works.

Due to the adaptively changed and spatially variant L_p flattening term and extra input information required for optimization, the optimization is complex and very challenging for traditional numerical solvers. To our knowledge, this is the first attempt of treating deep network as a numerical solver in the image smoothing field. In the future, we also would like to explore more complex and different tasks, such as multi-image or video processing.

ACKNOWLEDGEMENT

We would also like to acknowledge our research grants: National 973 Program (2015CB352501) and NSFC-ISF (61561146397).

REFERENCES

Francis Bach, Rodolphe Jenatton, Julien Mairal, Guillaume Obozinski, et al. 2012. Structured sparsity through convex optimization. *Statist. Sci.* 27, 4 (2012), 450–468.
Linchao Bao, Yibing Song, Qingxiong Yang, Hao Yuan, and Gang Wang. 2014. Tree filtering: Efficient structure-preserving smoothing with a minimum spanning tree. *IEEE Transactions on Image Processing* 23, 2 (2014), 555–569.
S. Bi, X. Han, and Y. Yu. 2015. An L_1 image transform for edge-preserving smoothing and scene-level intrinsic decomposition. *ACM Transactions on Graphics* 34, 4 (2015), 78.
Dongdong Chen, Jing Liao, Lu Yuan, Nenghai Yu, and Gang Hua. 2017. Coherent online video style transfer. In *Proc. Intl. Conf. Computer Vision (ICCV)*.

Dongdong Chen, Lu Yuan, Jing Liao, Nenghai Yu, and Gang Hua. 2017b. Stylebank: An explicit representation for neural image style transfer. In *Proc. CVPR*, Vol. 1. 4.

Dongdong Chen, Lu Yuan, Jing Liao, Nenghai Yu, and Gang Hua. 2018. Stereoscopic neural style transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 10.

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

Qifeng Chen, Jia Xu, and Vladlen Koltun. 2017a. Fast image processing with fully-convolutional networks. In *International Conference on Computer Vision*. 2497–2506.

Hojin Cho, Hyunjoon Lee, Henry Kang, and Seungyong Lee. 2014. Bilateral texture filtering. *ACM Transactions on Graphics* 33, 4 (2014), 128.

Ginmo Chung and Luminita A Vese. 2009. Image segmentation using a multilayer level-set approach. *Computing and Visualization in Science* 12, 6 (2009), 267–285.

Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. 2010. The PASCAL visual object classes (VOC) challenge. *International Journal of Computer Vision* 88, 2 (2010), 303–338.

Qingnan Fan, Jiaolong Yang, Gang Hua, Baoquan Chen, and David Wipf. 2017. A generic deep architecture for single image reflection removal and image smoothing. In *International Conference on Computer Vision*. 3238–3247.

Qingnan Fan, Jiaolong Yang, Gang Hua, Baoquan Chen, and David Wipf. 2018. Revisiting Deep Intrinsic Image Decompositions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Qingnan Fan, Fan Zhong, Dani Lischinski, Daniel Cohen-Or, and Baoquan Chen. 2015. JumpCut: non-successive mask transfer and interpolation for video cutout. *ACM Transactions on Graphics* 34, 6 (2015), 195.

Zeev Farbman, Raanan Fattal, Dani Lischinski, and Richard Szeliski. 2008. Edge-preserving decompositions for multi-scale tone and detail manipulation. *ACM Transactions on Graphics* 27, 3 (2008), 67.

Raanan Fattal. 2009. Edge-avoiding wavelets and their applications. *ACM Transactions on Graphics* 28, 3 (2009), 22.

Eduardo SL Gastal and Manuel M Oliveira. 2011. Domain transform for edge-aware image and video processing. In *ACM Transactions on Graphics*, Vol. 30. ACM, 69.

Michaël Gharbi, Jiawen Chen, Jonathan T Barron, Samuel W Hasinoff, and Frédo Durand. 2017. Deep bilateral learning for real-time image enhancement. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 118.

Bumsub Ham, Minsu Cho, and Jean Ponce. 2015. Robust image filtering using joint static and dynamic guidance. In *IEEE Conference on Computer Vision and Pattern Recognition*. 4823–4831.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *IEEE International Conference on Computer Vision*. 1026–1034.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.

Mingming He, Dongdong Chen, Jing Liao, Pedro V Sander, and Lu Yuan. 2018. Deep Exemplar-based Colorization. *ACM Transactions on Graphics (Proc. of Siggraph 2018)* (2018).

Paul W Holland and Roy E Welsch. 1977. Robust regression using iteratively reweighted least-squares. *Communications in Statistics-theory and Methods* 6, 9 (1977), 813–827.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*. 448–456.

Michael Kass and Justin Solomon. 2010. Smoothed local histogram filters. In *ACM Transactions on Graphics*, Vol. 29. 100.

Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. 2016. Accurate image super-resolution using very deep convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition*. 1646–1654.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *International Conference on Learning Representations* (2015).

Sifei Liu, Jinshan Pan, and Ming-Hsuan Yang. 2016. Learning recursive filters for low-level vision via a hybrid neural network. In *European Conference on Computer Vision*. 560–576.

Wei Liu, Xiaogang Chen, Chuanhua Shen, Zhi Liu, and Jie Yang. 2017. Semi-Global Weighted Least Squares in Image Filtering. In *IEEE International Conference on Computer Vision*.

Cewu Lu, Li Xu, and Jiaya Jia. 2012. Combining sketch and tone for pencil drawing production. In *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering*. Eurographics Association, 65–73.

Kaiyue Lu, Shaodi You, and Nick Barnes. 2018. Deep Texture and Structure Aware Filtering Network for Image Smoothing. EuropeanConferenceonComputerVision(ECCV)

Dongbo Min, Sunghwan Choi, Jiangbo Lu, Bumsub Ham, Kwanghoon Sohn, and Minh N Do. 2014. Fast global image smoothing based on weighted least squares. *IEEE Transactions on Image Processing* 23, 12 (2014), 5638–5653.

Pavel Mrázek, Joachim Weickert, and Andres Bruhn. 2006. On robust estimation and smoothing with spatial and tonal kernels. In *Geometric properties for incomplete data*. Springer, 335–352.

Sylvain Paris and Frédo Durand. 2006. A fast approximation of the bilateral filter using a signal processing approach. In *European Conference on Computer Vision*. 568–580.

Sylvain Paris, Samuel W Hasinoff, and Jan Kautz. 2011. Local Laplacian filters: Edge-aware image processing with a Laplacian pyramid. *ACM Trans. Graph.* 30, 4 (2011), 68–1.

Pietro Perona and Jitendra Malik. 1990. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12, 7 (1990), 629–639.

VB Surya Prasath, Dmitry Vorotnikov, Rengarajan Pelapur, Shani Jose, Guna Seetharaman, and Kannappan Palaniappan. 2015. Multiscale Tikhonov-total variation image restoration using spatially varying edge coherence exponent. *IEEE Transactions on Image Processing* 24, 12 (2015), 5220–5235.

Yaniv Romano, Michael Elad, and Peyman Milanfar. 2017. The little engine that could: Regularization by denoising (RED). *SIAM Journal on Imaging Sciences* 10, 4 (2017), 1804–1844.

Guodong Rong and Tiow-Seng Tan. 2006. Jump flooding in GPU with applications to Voronoi diagram and distance transform. In *The 2006 symposium on Interactive 3D graphics and games*. 109–116.

Carlo Tomasi. 1998. Bilateral filtering for gray and color images. In *International Conference on Computer Vision*. IEEE, 839–846.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. 2018. Deep Image Prior. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2018).

Ben Weiss. 2006. Fast median and bilateral filtering. *Acm Transactions on Graphics* 25, 3 (2006), 519–526.

Holger Winnemöller, Sven C Olsen, and Bruce Gooch. 2006. Real-time video abstraction. In *ACM Transactions On Graphics*, Vol. 25. 1221–1226.

Li Xu, Cewu Lu, Yi Xu, and Jiaya Jia. 2011. Image smoothing via L_0 gradient minimization. In *ACM Transactions on Graphics*, Vol. 30. 174.

Li Xu, Jimmy SJ. Ren, Qiong Yan, Renjie Liao, and Jiaya Jia. 2015. Deep edge-aware filters. In *International Conference on Machine Learning*. 1669–1678.

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

Fisher Yu and Vladlen Koltun. 2016. Multi-scale context aggregation by dilated convolutions. *International Conference on Learning Representations*.

Feihu Zhang, Longquan Dai, Shiming Xiang, and Xiaopeng Zhang. 2015. Segment graph based image filtering: fast structure-preserving smoothing. In *IEEE International Conference on Computer Vision*. 361–369.

Pingping Zhang, Dong Wang, Huchuan Lu, Hongyu Wang, and Xiang Ruan. 2017. Amulet: Aggregating multi-level convolutional features for salient object detection. In *International Conference on Computer Vision*. 202–211.

Qi Zhang, Xiaoyong Shen, Li Xu, and Jiaya Jia. 2014. Rolling Guidance Filter. In *European Conference on Computer Vision*. 815–830.

Feiyun Zhu, Ying Wang, Bin Fan, Shiming Xiang, Geofeng Meng, and Chunhong Pan. 2014. Spectral unmixing via data-guided sparsity. *IEEE Transactions on Image Processing* 23, 12 (2014), 5412–5427.