Revisiting Deep Image Smoothing and Intrinsic Image Decomposition

Qingnan Fan[1] David Wipf[2] Gang Hua[2] Baoquan Chen[1]
[1]Shandong University [2]Microsoft Research Asia
fqnchina@gmail.com, {davidwip,ganghua}@microsoft.com, baoquan@sdu.edu.cn

Abstract

We propose an image smoothing approximation and intrinsic image decomposition method based on a modified convolutional neural network architecture applied directly to the original color image. Our network has a very large receptive field equipped with at least 20 convolutional layers and 8 residual units. When training such a deep model however, it is quite difficult to generate edge-preserving images without undesirable color differences. To overcome this obstacle, we apply both image gradient supervision and a channel-wise rescaling layer that computes a minimum mean-squared error color correction. Additionally, to enhance piece-wise constant effects for image smoothing, we append a domain transform filter with a predicted refined edge map. The resulting deep model, which can be trained end-to-end, directly learns edge-preserving smooth images and intrinsic decompositions without any special design or input scaling/size requirements. Moreover, our method shows much better numerical and visual results on both tasks and runs in comparable test time to existing deep methods.

1. Introduction

Image smoothing is a fundamental building block of many computer vision and graphics applications including texture removal, image enhancement, edge extraction, image abstraction, and beyond. The underlying goal of image smoothing, or edge preserving filtering, is to extract sparse salient structures like perceptually important edges and contours, while minimizing color differences in some input image of interest. A vast array of practical filtering algorithms have been proposed for this task using traditional image processing principles and energy functions that operate across both local and global image regions [5, 12, 13, 19, 26, 27, 31, 33, 34, 35]. However, often an expensive, impractical optimization problem must be solved per instance to produce the desired edge-aware filtered images [5, 19].

Recently, deep learning has demonstrated a remarkable ability to replace expensive optimization procedures with cheap computational modules that can be trained offline when given access to representative data, and later deployed at a fraction of the computational budget [15]. For example, given a sufficient training corpus obtained by passing input images through some preferred smoothing filter, we can attempt to learn a deep model that efficiently mimics the filter outputs [32]. Of course considerable care must be taken in designing such a network to reduce training data requirements and to ensure that the trained model is indeed capable of fast, robust inference on new images, and naive, black-box approaches are destined to fail. Importantly, a well-designed network can also serve as the foundation for more complex, higher-level vision tasks.

In this regard, one notable application of edge-preserving image smoothing is the computation of intrinsic image decompositions [23, 5], which are useful for material recognition, image-based resurfacing, and relighting, among other things. The intrinsic image model assumes an ideal diffuse environment, in which an input image $\boldsymbol{I}$ is the pixel-wise product of an albedo image $\boldsymbol{A}$ and a shading image $\boldsymbol{S}$, i.e.,

$$\boldsymbol{I} \approx \boldsymbol{A} \odot \boldsymbol{S}. \tag{1}$$

The albedo or reflectance image indicates how object surface materials reflect light, while shading accounts for illumination effects due to geometry, shadows, and inter-reflections. Obviously, estimating such a decomposition is a fundamentally ill-posed problem as there exist infinitely many feasible decompositions. Fortunately though, prior information, often instantiated via specially tailored image smoothing filters or energy terms, allows us to constrain the space of feasible solutions [2, 4, 6, 9, 28, 36]. For example, the albedo image will usually be nearly piece-wise constant, with a finite number of levels reflecting a discrete set of materials and boundaries common to natural scenes. In contrast, the shading image is often assumed to be grayscale, and is more likely to contain smooth gradations quantified by small gradients except at locations with cast shadows or abrupt changes in scene geometry [23].

Naturally, given access to ground truth intrinsic image decompositions, deep networks provide a data-driven can-

didate for solving this ill-posed inverse problem with fewer heuristic or hand-crafted assumptions. In this paper, we propose a flexible deep neural network architecture that is simultaneously capable of handling wide-ranging, generic image smoothing tasks, where the primary goal is an efficient implementation of existing edge-aware filtering effects, as well as intrinsic image decompositions, where now an augmented objective supports improved accuracy on a higher-level vision task. Our fully trainable, end-to-end network relies on three important ingredients:

Large Flexible Receptive Fields: Edge-aware filtering requires contextual information across a relatively wide image region, especially in the context of an end-to-end system. Additionally, accurate intrinsic image decompositions must account for the fact that distant pixels with the same reflectance on a large surface could have significant color differences due to the accumulation of smoothly-varying soft cast shadows [5]. Hence large receptive fields are essential for both tasks, and we choose to actualize them with maximal flexibility via a fully convolutional subnetwork with many layers.

Scale Factors and Gradient Supervision: For image smoothing, the input and output color images should be highly correlated, but unfortunately the color may be attenuated during many forward convolutions in a deep network such as ours. To address this problem in the context of superresolution, [20] adds a skip connection directly from input to output images, which is not effective for image smoothing in our experience. Instead, we include a rescaling layer for each channel that minimizes MSE color differences and naturally allows gradient backpropagation for training purposes. We also observe that supervising gradients of predicted images using an auxiliary objective helps preserve salient edges; however, this is notably different than direct operation in the gradient domain akin to most existing methods.

Domain Transform: To further enhance piece-wise constant effects favored by sparsity-based filtering methods [31, 5], we also leverage a domain transform [14, 8]. Practically speaking, in our system this can be viewed as a series of final, parameterless network layers that implicitly filter color information across the image in a warped domain. This warping is determined by edge maps derived from a preliminary smoothed image, which emerges from the lower convolutional and scaling layers described previously. Although the domain transform layers themselves do not actually increase the overall network capacity per se, they nonetheless alter the space of candidate filters and influence which image regions should be smoothed or not.

Overall, we propose an end-to-end system that can efficiently approximate a variety of computationally intensive image smoothing filters as well as infer related intrinsic image decompositions. For each of these tasks, our method demonstrates state-of-the-art performance both visually and numerically. We differentiate our design choices from existing methods in the next section.

2. Relationship with Existing Models

Image Smoothing: Recently, there have been multiple attempts [32, 25] to imitate the behavior of a large, representative family of image smoothing filters using a single, data-driven approximation framework capable of accelerated inference speeds. For example, [32] proposed a 3-layer network for first predicting prominent gradients, followed by a separate post-processing step involving filter-specific tuning parameters to reconstruct the smoothed image estimate. A potential limitation however, is that gradient or salient structure prediction errors can easily propagate to surrounding areas in the pixel domain during the reconstruction step, reducing visual quality. Moreover, practical image smoothing filters favor varying degrees of edge preservation, spanning the gamut from blurry to piece-wise constant images. Purely gradient-based processing tends to struggle more generating images on the blurry end of this scale, where continuous color transitions may be required. For these reasons, our approach operates entirely in the original color image domain, a gradient supervision term notwithstanding.

A second approach from [25] involves a hybrid network composed of several convolutional layers followed by a set of recurrent network layers. In brief, the CNN portion of the model takes the original image as input and computes a set of weights that are presumably reflective of salient image structures. The recurrent network layers then take these weights, as well as the original image, and act as a form of data-driven guided filter to produce the desired smoothing effect. To a certain degree, our approach can be viewed as the antithesis of this model. Whereas we explicitly parameterize an image model and only later refine it using gradient supervision and a parameterless domain transform modulated via a small subnetwork, the method from [25] fully parameterizes the weights of a recurrent guided filter array, which acts much like a domain transform via the analysis from [8], but then applies these filters only to multi-scale versions of the original raw images. A downside of the later strategy is that it requires downsampling the original images to multiple scales, and at least in its present form, will not work with image sizes that are not multiples of 16 (provided code actually on works with 256×256 images). Additionally, any mismatch between the computed weights and legitimate salient edge structure will necessarily propagate to wider regions in the predicted output image since the recurrent filters have no mechanism to compensate.

Intrinsic Image Decomposition: To move beyond the traditional filtering approaches mentioned in Section 1 to data-driven deep models capable of producing scene-

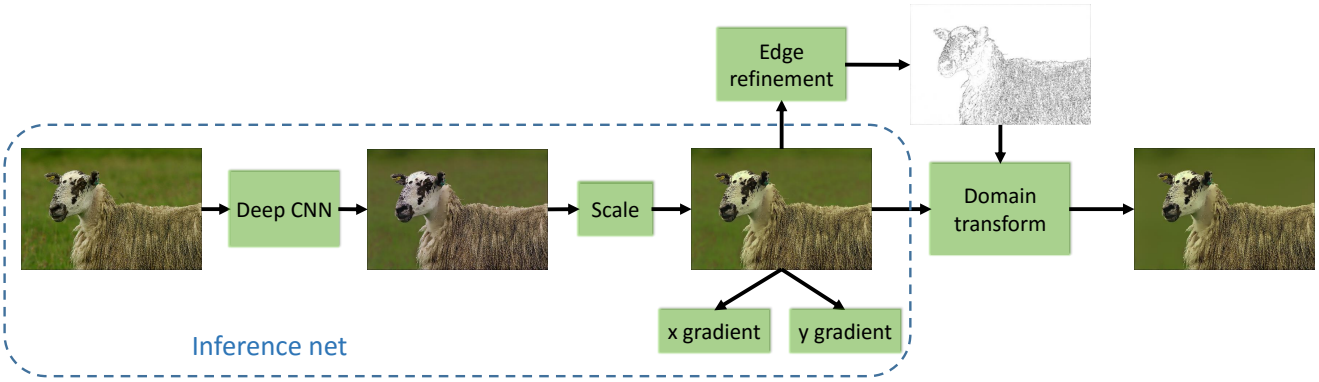

Figure 1. **Image Smoothing Network Structure**. Input image is fed into the inference network, channel-wise rescaling layer, edge refinement and domain transform to predict the final smooth images.

level intrinsic image decompositions, we require images with known ground truth. The MPI-Sintel intrinsic images benchmark [7] provides an important step in this direction, and two existing deep network pipelines have been built upon it. First, [30] learns a two-scale convolutional network to directly predict both albedo and shading images. However, the specific architecture, which closely resembles that from [10] developed for predicting depth and surface normals, involves intermediate feature maps at 1/32 scale such that significant detail information may be compromised. A second more recent method from [21] trains a joint model to learn depth maps and intrinsic images with activations coupled across iterations. Unlike our approach, the resulting pipeline operates in the gradient domain and requires separate post-processing steps, as well as additional guidance from labeled depth images.

3. Proposed Model

Our proposed model consists of three parts: an inference net, an edge refinement step, and a domain transform. We will now detail each component in turn.

3.1. Inference Net

Unlike previous work that primarily attempts to explicitly predict sparse image gradients [32, 21] or related weight maps [25], the primary parameterized workhorse of our model is an inference net designed to directly predict filtered images in an end-to-end manner. This network contains 20 convolutional neural network (CNN) layers followed by a channel-wise rescaling layer and two gradient computation layers. These are designed as follows:

CNN Layers: Of the 20 CNN layers, the middle 18 are all of the equivalent size 64×64×3×3, meaning that for each of 64 output feature maps, it filters 3×3 spatial regions of all 64 input feature maps. In contrast, the first layer takes the input image and represents it as 64 feature maps, while the last layer transforms the multi-channel representation back to the original image space (3 color channels). In both cases the kernel sizes are also 3×3. Additionally, each convolutional layer is followed by batch normalization (BN) and ReLU layers except for the last one. Finally, to accelerate the training process, we retrofit the middle 16 convolutions with 8 modified residual units [18] (see the supplementary file for details of these modifications).

As observed in [32], a fully convolutional network trained directly in the original image domain may tend to generate very blurry images and unwanted details, while gradient domain supervision is more sensitive to the changes in sharp edges that comply with human perception of contrast more than absolute color values. But we argue that this limitation of the former is largely due to unsufficient receptive field size and flexibility. After all, image smoothing and intrinsic image decomposition techniques determine the color values of each pixel in the context of a large surrounding area. Especially for albedo images, small color difference between nearby pixels can accumulate, producing significant differences between distant pixels with complex connecting pathways even when they share the same reflectance. This renders the estimation problem quite difficult if we are restricted to either small or insufficiently parameterized receptive fields. Therefore, instead of using 3 convolutional layers of large kernel size (16×16, 1×1 and 8×8) as in [32], we adopt 20 convolutional layers with 3×3 kernels, which allows the learned regression function to more accurately account for distant contextual information [29].

Channel-wise Rescaling Layer: Although image smoothing attempts to remove small gradients, it should still generally preserve a high degree of correlation between input and output colors. With so many convolutional layers however, such correlations may eventually weaken without some additional source of long-term memory that reflects the preservation of the original color schema. For this purpose, we include an additional channel-wise rescaling layer as follows. Let $\boldsymbol{P}_{1_c}$ denote the output of the 20 layer CNN module associated with color channel c, while $\boldsymbol{I}_c$ is the cor-

responding input image. We then compute

$$\alpha_c = \arg\min_{\alpha_c} \|\boldsymbol{I}_c - \alpha_c \boldsymbol{P}_{1_c}\|_2 = \frac{\boldsymbol{P}_{1_c} \cdot \boldsymbol{I}_c}{\boldsymbol{P}_{1_c} \cdot \boldsymbol{P}_{1_c}} \quad (2)$$

via a dedicated scaling layer (where $\cdot$ indicates a dot product) and output the updated image prediction $\boldsymbol{P}_{2_c} = \alpha_c \boldsymbol{P}_{1_c}$. As a small caveat, we also threshold values of α_c outside of the range $[0.7, 1.3]$ during training when the predicted images may still be radically different from $\boldsymbol{I}_c$. Note that network gradients are easily backpropagated through such a layer. Also for testing, we discard any such threshold. A related idea has been incorporated into a classical, optimization-based intrinsic image decomposition [16], but this represents a decidedly different context from our CNN model. In any event, a channel-wise rescaling layer is a simple but very effective addition to our pipeline, and we find that it reduces negative color difference effects by a wide margin.

Gradient Supervision: Most image smoothing algorithms seek to more-or-less achieve a piece-wise constant result. Certainly the albedo component of an intrinsic image decomposition problem is well-approximated by such an effect. But smoothing a large region into a single color is a challenging task for convolution since the kernel weights are shared over the whole feature map, but local color values can vary dramatically across regions. To help mitigate this problem, we supervise the x and y gradients $\nabla_x \boldsymbol{P}_2$,$\nabla_y \boldsymbol{P}_2$ from the color-normalized images $\boldsymbol{P}_2$ (across all color channels) using a standard auxillary MSE loss function.

3.2. Edge Refinement and Domain Transform

The images generated by the inference net described above already demonstrate excellent numerical performance. But we observe that some modest color changes still exist in regions that piece-wise constant filters like [5, 13, 31, 33] are able to smooth away. Inspired by [14, 8], we address this lingering issue via a domain transform guided by refined edge estimates. The domain transform is an edge-preserving processing step that admits efficient implementation via separable 1-D recursive filtering layers applied across rows and columns in an image. Two inputs are required: the raw input signal to be filtered, which corresponds with our predicted image $\boldsymbol{P}_2$ from the previous layer, and a domain transform density map that differentiates which regions to filter and which to leave unaltered.

This mapping is produced by a small edge refinement subnetwork in our pipeline. The core idea originates from [14], where the density map is simply the original input image gradient. Later [8] experimented with more sophisticated predicted edge maps applied in conjunction with image segmentation scores. However, their pipeline addresses a completely different task, and the embedded, application-specific edge map requires the concatenation of features culled from intermediate CNN layers. In contrast, since we already generate reasonable edge-preserving images from our base inference net, we do not require this degree of sophistication. Hence our proposed edge refinement subnetwork operates as follows. Further technical details about how to implement domain transforms within a neural network architecture, including gradient backpropagation steps, can be found in [8].

The edge refinement module consists of two stages. The first computes, for every point in the input $\boldsymbol{P}_2$, the averaged absolute differences between each of its four neighboring pixels, which is then further averaged across all color channels to produce a single value. The resulting map is next refined using a very shallow network containing 3 convolutional layers. The feature maps for the first two convolution layers have 64 channels. The kernel size for the 1st and 3rd layer is 15×15, zero padded with 7 pixels on each side to maintain spatial dimensions, while the kernel size for the 2nd layer is 1×1, which computes a weighted average of processed pixel vectors for the smoothing operation. The first two convolution layers are followed with ReLU units, while the third is not.

3.3. Modifications for Intrinsic Images

Finally we conclude this section by describing special accommodations for intrinsic image decompositions. Here the inference net is split into two branches after the middle convolutional layer in order to predict albedo and shading images simultaneously. We also observe that (1) can be leveraged as a useful prior for intrinsic image decomposition, enforcing that the estimated decomposition must recombine to approximate the original image $\boldsymbol{I}$ (to the extent that the diffusive, Lambertian model is accurate). Therefore we include an additional term that penalizes deviations of the estimated $\boldsymbol{A} \odot \boldsymbol{S}$ from $\boldsymbol{I}$, which helps to balance the error between albedo and shading images.

Additionally, intrinsic image decompositions often require potentially larger receptive fields since accumulated soft shadows sharing the same albedo could cover a wide region. Consequently we extend the depth to 42 convolutional layers and downsample intermediate features maps by $1/2$ which saves half the training time. And because albedo and shading images need not have a very close color similarity to the input image, we can remove the color normalization layer for training an intrinsic image decomposition network.

4. Training Details

We begin with the image smoothing network. The embedded inference net requires supervision on 4 outputs: the predicted image $\boldsymbol{P}_1$ directly from the CNN, the color-normalized image $\boldsymbol{P}_2$, and computed gradients in x and

y directions $\nabla_x \boldsymbol{P}_2$,$\nabla_y \boldsymbol{P}_2$. With the $*$ symbol denoting ground truth, the inference net objective terms become

$$\begin{aligned}\boldsymbol{l}_1(\theta) &= \boldsymbol{w}_1||\boldsymbol{P}_1 - \boldsymbol{P}_1^*||_2^2 + \boldsymbol{w}_2||\boldsymbol{P}_2 - \boldsymbol{P}_2^*||_2^2 \\ &+ \boldsymbol{w}_3(||\nabla_x \boldsymbol{P}_2 - \nabla_x \boldsymbol{P}_2^*||_2^2 + ||\nabla_y \boldsymbol{P}_2 - \nabla_y \boldsymbol{P}_2^*||_2^2) \quad (3)\end{aligned}$$

where θ denotes the parameter set of the network layers and ℓ_2 norm penalties are chosen since they favor high peak signal-to-noise ratios (PSNR), a common evaluation criteria for image estimation problems. After training inference net in isolation, we then learn an independent network for edge refinement, whose output is denoted $\boldsymbol{E}$ with corresponding loss function $||\boldsymbol{E}-\boldsymbol{E}^*||_2^2$. Finally, we jointly train the whole network using

$$\boldsymbol{L}(\theta) = \boldsymbol{l}_1(\theta) + \boldsymbol{w}_3||\boldsymbol{E} - \boldsymbol{E}^*||_2^2 + \boldsymbol{w}_4||\boldsymbol{P}_3 - \boldsymbol{P}_3^*||_2^2, \quad (4)$$

where $\boldsymbol{P}_3$ is the output of the domain transform, and noting that, as a gradient proxy we retain the same weighting parameter for $\boldsymbol{E}$ as used in (3) for coordinate-wise gradients.

For the intrinsic image network, the albedo and shading branches share the same loss function as introduced above, referred to as $\boldsymbol{L_a}(\theta)$ and $\boldsymbol{L_s}(\theta)$, except for the removal of supervision on the color-normalized image $\boldsymbol{P}_2$ and with analogous gradient supervision now demoted to $\boldsymbol{P}_1$. We also include supervision that reflects the constraint from (1), giving the final cost

$$\boldsymbol{L}(\theta) = \boldsymbol{L_a}(\theta) + \boldsymbol{L_s}(\theta) + \boldsymbol{w}_5||\boldsymbol{A}_1 \odot \boldsymbol{S}_1 - \boldsymbol{A}^* \odot \boldsymbol{S}^*||_2^2 \quad (5)$$

where $\boldsymbol{A}_1$ and $\boldsymbol{S}_1$ are the estimates obtained from the inference net analogous to $\boldsymbol{P}_1$.

We implement all networks using the Torch framework using mini-batch gradient descent, with batch size fixed at 16. The energy function weights from $\boldsymbol{w}_1$ to $\boldsymbol{w}_5$ are empirically determined as 0.2, 0.1, 0.35, 0.2, and $0.2/255$ respectively. Note that these weights are used for all smoothing filters and all intrinsic image decompositions, and hence are quite robust; manual tuning in an application-specific manner is not required. Convolutional layers are initialized using the methods from [17]. To train inference net, we use ADAM [22] to accelerate convergence. The learning rate is initially set to 0.01 and then reduced to 0.001 to generate further improvement. For edge refinement, we use simple stochastic gradient descent [24] with learning rate set to 0.01. And for fine-tuning the whole network together, we again use ADAM with a small learning rate of $1e$-5.

Finally, computation of the forward and backward passes follows standard rules through the inference net; however, propagating information through the edge computation step at the front end of the edge refinement layer requires non-standard updates. These are detailed in the supplementary file.

5. Experimental Results

This section showcases the numerical and visual advantages of our approach. Further experiments and thorough ablation studes are deferred to the supplementary file.

5.1. Image Smoothing

Dataset: As a low-level vision task, many corpora of natural images are suitable for training and evaluating smoothing algorithms. Here we use the public PASCAL VOC dataset [11] developed for visual object category recognition and detection, which contains around 17,000 images in a wide range of viewing conditions. These images were collected from the the flickr photo-sharing website, the same data source used by [32] with which we compare. We randomly crop images at a size of 224×224 pixels for a single image to generate about 17,000 patches. To evaluate our performance, we randomly picked 100 images to form our VOC test split. We also randomly pick 100 images from the BSDS dataset [1] for testing, since as we confirmed from the authors of [32], these data generate comparable results from training with their method on either BSDS or self-collected flickr images. However, they also trained slow filters on BSDS data which leads to a potential overfitting empirical advantage on our test data.

Comparisons: We approximate 9 different image smoothing filters including the bilateral filter (BLF) [27], iterative bilateral filter (IBLF) [13], L_0 smoothing (L0) [31], rolling guidance filter (RGF) [34], RTV texture smoothing (RTV) [33], weighted least square smoothing (WLS) [12], weighted median filter (WMF) [35], fast local Laplacian filter (FastLLF) [26] and L_1 smoothing (L1) [5]. Compared to 7 public models released by [32], we show much better results on both VOC and BSDS datasets in Tables 1. Likewise, Table 2 presents comparisons with the very recent work from [25]. However, their released code at the time of this writing, which is built upon a multiscale decomposition, only seems to operate on 256×256 images. Even though we do not apply any special tuning for this size, or train on multiscale images as in 2, we still achieve better re-

	PSNR		SSIM	
	Liu16	Ours	Liu16	Ours
L0	32.26	33.97	0.958	0.980
RGF	38.29	37.29	0.983	0.986
WLS	38.64	38.15	0.986	0.987
WMF	33.29	36.68	0.951	0.982
Ave.	35.64	36.52	0.966	0.984

Table 2. Comparison with recent work Liu16 [25] on image smoothing task. Experiments are conducted using their four released trained filters on PASCAL VOC dataset.

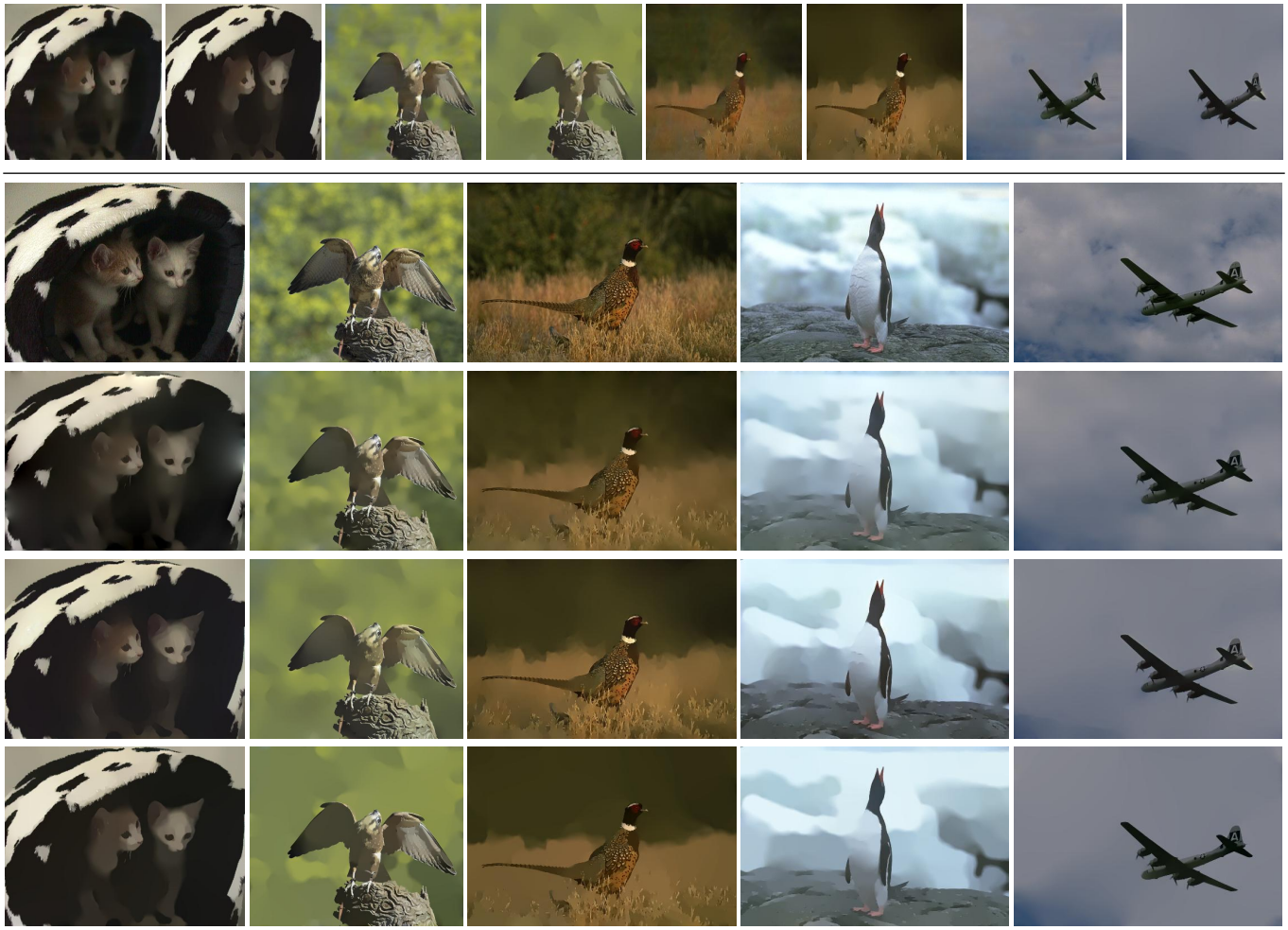

Figure 2. **Qualitative Comparison on Image Smoothing Task**. *Top*: Pairs of filtered results by [25] (left) and ours (right). *Bottom*: From top to bottom row we show input image, results by [32], ours and ground truth. All the results are trained on L0 filter.

			BLF	IBLF	L0	RGF	RTV	WLS	WMF	FastLLF	L1	Ave.
VOC	PSNR	Xu15	35.02	32.97	31.66	32.49	35.68	33.92	29.62			32.62
		Ours	38.34	35.07	33.56	38.62	36.02	38.03	35.75	31.87	32.43	36.48
	SSIM	Xu15	0.976	0.962	0.966	0.950	0.974	0.963	0.960			0.964
		Ours	0.987	0.980	0.981	0.987	0.983	0.985	0.980	0.979	0.950	0.983
BSDS	PSNR	Xu15	35.31	33.57	31.13	31.88	35.64	35.78	30.35			33.38
		Ours	36.89	35.07	32.98	37.70	36.34	35.89	35.65	31.57	32.18	35.79
	SSIM	Xu15	0.975	0.966	0.964	0.951	0.975	0.977	0.966			0.967
		Ours	0.986	0.981	0.976	0.986	0.985	0.984	0.983	0.981	0.953	0.984

Table 1. **Quantitative Comparison on Image Smoothing Task**. We report PSNR and SSIM error metrics (larger is better) on 9 different smoothing filters compared with Xu15 [32]. The average value is computed among the front 7 filters. Our results outperform our competitor by a large margin on both PASCAL VOC and BSDS test dataset.

sults on average in their setting. And our consistently higher SSIM values indicates that our learned filters are quite effective in generating perceptually good images. Additionally, we note that WMF can generate relatively blurry images compared to other filters with the parameterization used in these experiments taken from [32], and yet our network han-

		MSE			LMSE			DSSIM		
		albedo	shading	average	albedo	shading	average	albedo	shading	average
image split	Retinex [16]	0.0606	0.0727	0.0667	0.0366	0.0419	0.0393	0.2270	0.2400	0.2335
	Barron *et al.* [3]	0.0420	0.0436	0.0428	0.0298	0.0264	0.0281	0.2100	0.2060	0.2080
	Chen *et al.* [2]	0.0307	0.0277	0.0292	0.0185	0.0190	0.0188	0.1960	0.1650	0.1805
	MSCR [30]	0.0100	0.0092	0.0096	0.0083	0.0085	0.0084	0.2014	0.1505	0.1760
	JCNF [21]	0.0070	0.0090	0.0070	0.0060	0.0070	0.0070	0.0920	0.1010	0.0970
	Ours (*RegenInput*)	0.0038	0.0042	0.0040	0.0028	0.0031	0.0029	0.0758	0.0659	0.0708
	Ours (*RegenShade*)	0.0044	0.0046	0.0045	0.0033	0.0035	0.0034	0.1017	0.0733	0.0875
scene split	MSCR [30]	0.0201	0.0224	0.0213	0.0131	0.0148	0.0139	0.2073	0.1594	0.1833
	Ours (*RegenInput*)	0.0129	0.0171	0.0150	0.0083	0.0118	0.0100	0.1285	0.1252	0.1268
	Ours (*RegenShade*)	0.0151	0.0165	0.0158	0.0090	0.0106	0.0098	0.1555	0.1236	0.1395

Table 3. **Quantitative Comparison on MPI-Sintel Benchmark**. We evaluate our results on three standard error rates of intrinsic images on MPI-Sintel dataset. Our models trained on two resynthesized datasets outperform previous results by a large margin.

dles this case quite well in both tables relative to other approaches. This is likely because previous methods more-or-less directly rely on sparse structures and may have trouble reproducing the softer transitions required by this WMF filter.

We next display some visual comparisons in Figure 2 using the L0 filter, which produces effects more on the piecewise constant end of the smoothing spectrum. Here we observe that both [32] and [25] have some trouble reproducing constant regions, and [25] has difficulty with erroneous gradient estimates propagating from image boundaries (see cat image). Finally, from a computational standpoint, our model requires 0.21s to process one QVGA image and 0.75s to process one VGA image, which is marginally faster than [32], and amenable to many practical applications.

5.2. Intrinsic Image Decomposition

Dataset: We follow two recent state-of-the-art deep learning based methods [30, 21] and evaluate our algorithm on the MPI-Sintel dataset [7] that facilates scene-level quantitative comparisons. It consists of 890 images from 18 scenes with 50 frames each. Due to limited images in this dataset, we randomly crop 10 different patches of size 300×300 from one image to generate 8900 patches. Like [30], we use two-fold cross validation to obtain all 890 test results with two trained models. We evaluate our results on both a *scene split*, where half the scenes are used for training and the other half for testing, and an *image split*, where all 890 images are randomly separated into two parts. Clearly the *scene split* is less susceptible to inflated results from overfitting, since images in the same sequence has a big overlap with each other while images in different scenes do not.

As noted by [9, 30], the ground truth "clean pass", albedo and shading images in MPI-Sintel dataset are rendered independently. Their shading images are rendered with uniform grey albedo on all objects, which is actually not realistic, since rendered shading does not have inter-reflections between different surfaces. To better adhere to the most basic intrinsic image assumption from (1), we regenerate input images with multiplication of albedo and shading as *RegenInput* and regenerate shading images with division between input and albedo as *RegenShade* to train models. Testing results are also conducted on these two datasets.

Comparison: To quantitatively evaluate the performance, we use three criteria, mean-squared error (MSE), local mean-squared error (LMSE), and the dissimilarity version of the structural similarity index (DSSIM) as proposed in [2]. In Table 3, our model trained on the two regenerated datasets shows much better results. Note that the very recent work JCNF [21] benefits from training on additional depth map data, and as a gradient domain method needs to execute a separate post-processing step to reconstruct images. MSCR [30] also benefits from additional training data from the MIT Intrinsic Images dataset [16].

Since shading images in the MPI-Sintel dataset are rendered independently of the input image, they will contain information that is not included in the input. Therefore training on the original dataset can never approach the ground truth, regardless of the method. We trained our model with the original dataset and found that the lower bound of the shading image is relatively higher than the ones trained on our regenerated dataset as expected, reflecting theoretically unobservable shadow information which cannot be modeled. However, the small albedo errors remained at the small values reported in the table.

We show 4 groups of qualitative results trained on the *scene split* in Figure 3. For a better visual comparison of intrinsic images, we display images based on the *RegenInput* model not the *RegenShade* since it is supervised to learn the same intrinsic images as MSCR [30]. Note that we are able to predict much sharper and relatively high-quality results even though we have only around 500 training images (and no outside training information as with other methods), and

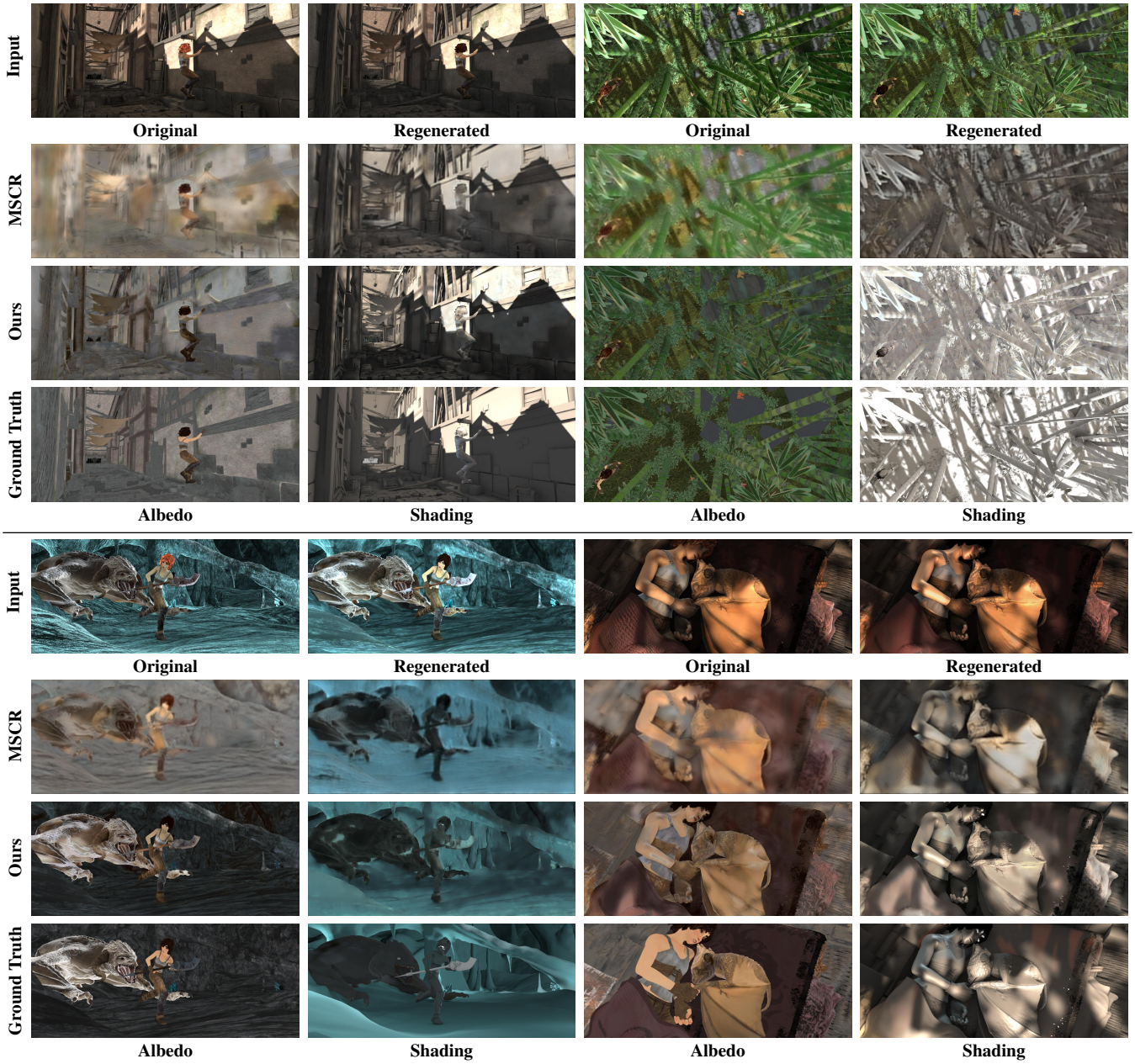

Figure 3. **Qualitative Comparison on MPI-Sintel Benchmark**. The visual results are evaluated on the model trained on *scene split*. Compared to state-of-the-art method MSCR [30], we show much sharper effects and better performance.

training and testing data do not have any overlap. Notice also that in the original ground truth input images, the color of the girl's hair is not reflected in either albedo or shading. The ground truth shading image on the bottom right also contains some highlights not included in the original input.

6. Conclusion

This paper proposes a deep network for image smoothing and intrinsic image decomposition tasks. Design features include a deep CNN with large receptive fields operating in the original color domain, a color normalization step, gradient supervision, and a domain transform for refining final image estimates. The resulting pipeline produces edge-preserving smooth images or intrinsic decompositions without any preprossing, postprocessing, application-specific tuning parameters, or special requirements of input image size. We have also numerically demonstrated state-of-the-art performance on important representative benchmarks.

References

[1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *PAMI*, 33(5):898–916, 2011. 5

[2] J. T. Barron and J. Malik. Intrinsic scene properties from a single rgb-d image. *CVPR*, 2013. 1, 7

[3] J. T. Barron and J. Malik. Shape, illumination, and reflectance from shading. *PAMI*, 37(8):1670–1687, 2015. 7

[4] S. Bell, K. Bala, and N. Snavely. Intrinsic images in the wild. *ACM Trans. on Graphics (SIGGRAPH)*, 33(4), 2014. 1

[5] S. Bi, X. Han, and Y. Yu. An L_1 image transform for edge-preserving smoothing and scene-level intrinsic decomposition. *ACM Transactions on Graphics*, 34(4):78, 2015. 1, 2, 4, 5

[6] N. Bonneel, K. Sunkavalli, J. Tompkin, D. Sun, S. Paris, and H. Pfister. Interactive intrinsic video editing. *ACM Transactions on Graphics (TOG)*, 33(6):197, 2014. 1

[7] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), editor, *ECCV*, Part IV, LNCS 7577, pages 611–625. Springer-Verlag, Oct. 2012. 3, 7

[8] L.-C. Chen, J. T. Barron, G. Papandreou, K. Murphy, and A. L. Yuille. Semantic image segmentation with task-specific edge detection using cnns and a discriminatively trained domain transform. In *CVPR*, 2016. 2, 4

[9] Q. Chen and V. Koltun. A simple model for intrinsic image decomposition with depth cues. In *ICCV*, 2013. 1, 7

[10] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, pages 2650–2658, 2015. 3

[11] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338, June 2010. 5

[12] Z. Farbman, R. Fattal, D. Lischinski, and R. Szeliski. Edge-preserving decompositions for multi-scale tone and detail manipulation. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2008)*, 27(3), Aug. 2008. 1, 5

[13] R. Fattal, M. Agrawala, and S. Rusinkiewicz. Multiscale shape and detail enhancement from multi-light image collections. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 26(3), Aug. 2007. 1, 4, 5

[14] E. S. Gastal and M. M. Oliveira. Domain transform for edge-aware image and video processing. In *ACM Transactions on Graphics (TOG)*, volume 30, page 69. ACM, 2011. 2, 4

[15] K. Gregor and Y. LeCun. Learning fast approximations of sparse coding. In *ICML*, pages 399–406, 2010. 1

[16] R. Grosse, M. K. Johnson, E. H. Adelson, and W. T. Freeman. Ground truth dataset and baseline evaluations for intrinsic image algorithms. In *ICCV*, pages 2335–2342. IEEE, 2009. 4, 7

[17] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, pages 1026–1034, 2015. 5

[18] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. *ECCV*, 2016. 3

[19] L. Karacan, E. Erdem, and A. Erdem. Structure-preserving image smoothing via region covariances. *ACM Trans. Graph.*, 32(6):176:1–176:11, 2013. 1

[20] J. Kim, J. K. Lee, and K. M. Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR*, 2016. 2

[21] S. Kim, K. Park, K. Sohn, and S. Lin. Unified depth prediction and intrinsic image decomposition from a single image via joint convolutional neural fields. In *ECCV*, 2016. 3, 7

[22] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *ICLR*, 2015. 5

[23] E. H. Land and J. J. McCann. Lightness and retinex theory. *J. Opt. Soc. Am.*, pages 1–11, 1971. 1

[24] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 5

[25] S. Liu, J. Pan, and M.-H. Yang. Learning recursive filters for low-level vision via a hybrid neural network. In *ECCV*, 2016. 2, 3, 5, 6, 7

[26] S. H. J. K. M. Aubry, S. Paris and F. Durand. Fast local laplacian filters: Theory and applications. *ACM Transactions on Graphics*, 2014. 1, 5

[27] S. Paris and F. Durand. A fast approximation of the bilateral filter using a signal processing approach. In *ECCV*, pages 568–580, 2006. 1, 5

[28] L. Shen and C. Yeo. Intrinsic images decomposition using a local and global sparse representation of reflectance. In *CVPR*, pages 697–704. IEEE, 2011. 1

[29] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 3

[30] M. M. Takuya Narihira and S. X. Yu. Direct intrinsics: Learning albedo-shading decomposition by convolutional regression. In *ICCV*, 2015. 3, 7, 8

[31] L. Xu, C. Lu, Y. Xu, and J. Jia. Image smoothing via L_0 gradient minimization. In *SIGGRAPH Asia*, 2011. 1, 2, 4, 5

[32] L. Xu, J. S. Ren, Q. Yan, R. Liao, and J. Jia. Deep edge-aware filters. In *ICML*, pages 1669–1678, 2015. 1, 2, 3, 5, 6, 7

[33] L. Xu, Q. Yan, Y. Xia, and J. Jia. Structure extraction from texture via natural variation measure. *ACM Transactions on Graphics (SIGGRAPH Asia)*, 2012. 1, 4, 5

[34] Q. Zhang, X. Shen, L. Xu, and J. Jia. Rolling guidance filter. In *ECCV*, pages 815–830, 2014. 1, 5

[35] Q. Zhang, L. Xu, and J. Jia. 100+ times faster weighted median filter. In *CVPR*, 2014. 1, 5

[36] Q. Zhao, P. Tan, Q. Dai, L. Shen, E. Wu, and S. Lin. A closed-form solution to retinex with nonlocal texture constraints. *PAMI*, 34(7):1437–1444, 2012. 1

Supplementary File: Revisiting Deep Image Smoothing and Intrinsic Image Decomposition

1 Outline

This document contains several technical details and experiments that could not be included in the main paper because of space considerations. Regarding the technical specifications, we first describe the modified residual unit we applied to our inference net and the forward and backward passes that form the front end of the edge refinement network. Later we present additional experiments, including an ablation study whereby we remove various parts of our pipeline and record relative performance, as well as more results on both intrinsic image decompositions and image smoothing.

2 Residual Unit Details

We applied a slightly modified residual unit [1, 2], which we found accelerated the training process and lead to modest improvement in the final performance in most cases. The structure of this unit shown in Figure 1, where the non-linearity represents a varient of ReLU [5] given by

$$f(x) = \begin{cases} 2x & \text{if } x \geqslant 0, \\ x & \text{otherwise.} \end{cases} \tag{1}$$

Also, BN refers to standard batch normalization. We defer a detailed evaluation of various residual units in the context of image smoothing to a future publication.

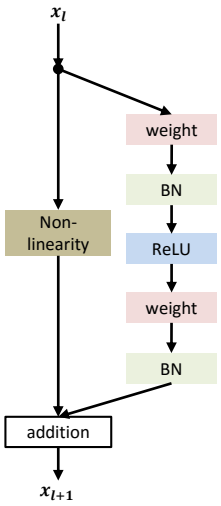

Figure 1: Modified residual unit.

3 Forward and Backward Passes for Edge Refinement Network

Here we introduce the forward and backward propagations through the edge computation step at the front end of the edge refinement network. Let us denote the input and output for this module as $\boldsymbol{x}$ and $\boldsymbol{y}$, then the forward and backward propagations become

$$\begin{aligned} \boldsymbol{y}_{i,j} = \frac{1}{2} \sum_{c} \big(& |\boldsymbol{x}_{i,j,c} - \boldsymbol{x}_{i-1,j,c}| + |\boldsymbol{x}_{i,j,c} - \boldsymbol{x}_{i+1,j,c}| \\ & + |\boldsymbol{x}_{i,j,c} - \boldsymbol{x}_{i,j-1,c}| + |\boldsymbol{x}_{i,j,c} - \boldsymbol{x}_{i,j+1,c}| \big) \end{aligned} \tag{2}$$

$$\frac{\partial L}{\partial \boldsymbol{x}_{i,j,c}} = \frac{1}{2}\Big[\frac{\partial L}{\partial \boldsymbol{y}_{i,j}}(\boldsymbol{h}_{i-1,j,c}+\boldsymbol{h}_{i+1,j,c}+\boldsymbol{h}_{i,j-1,c}+\boldsymbol{h}_{i,j+1,c}) - \big(\frac{\partial L}{\partial \boldsymbol{y}_{i-1,j}}\boldsymbol{h}_{i-1,j,c} + \frac{\partial L}{\partial \boldsymbol{y}_{i+1,j}}\boldsymbol{h}_{i+1,j,c} + \frac{\partial L}{\partial \boldsymbol{y}_{i,j-1}}\boldsymbol{h}_{i,j-1,c} + \frac{\partial L}{\partial \boldsymbol{y}_{i,j+1}}\boldsymbol{h}_{i,j+1,c}\big)\Big] \quad (3)$$

$$\boldsymbol{h}_{i+m,j+n,c} = \begin{cases} 1 & \text{if } \boldsymbol{x}_{i,j,c} \geqslant \boldsymbol{x}_{i+m,j+n,c}, \\ -1 & \text{if } \boldsymbol{x}_{i,j,c} < \boldsymbol{x}_{i+m,j+n,c}. \end{cases}$$
$$\boldsymbol{m} \in \{1,-1\} \text{ if } \boldsymbol{n} = 0 \text{ and } \boldsymbol{n} \in \{1,-1\} \text{ if } \boldsymbol{m} = 0 \quad (4)$$

respectively, where $\boldsymbol{i}$,$\boldsymbol{j}$ are image coordinates and $\boldsymbol{c}$ is the color channel index. The gradients will be back propagated to the previous inference network for jointly training the whole pipeline. Initially $\frac{\partial L}{\partial \boldsymbol{x}_{i,j,c}}$ is set to 0 and $\frac{\partial L}{\partial \boldsymbol{y}_{i,j}}$ receives value from the subsequent layer.

4 Self-Comparison Study

To better understand how each component of our network contributes to the final results, we evaluate several variants on test data in Table 1. In this table a "-" indicates that a particular feature has been removed, while "+" is analogously defined. Note also that "InferenceNet + DT" is equivalent to our full pipeline, where DT refers to the domain transform. Overall we observe that each component is essential for generating piece-wise constant images, which helps improve PSNR and SSIM by a substantial margin.

5 More Results on Intrinsic Image Decompositions

In Table 2 we reproduce results from the main paper, but with the inclusion of testing outcomes obtained when training on the original MPI-Sintel dataset. Again we are able to outperform previous methods by a significant margin, especially on the albedo images which are arguably more important for many applications. However, as mentioned in the main paper, the shading images from MPI-Sintel are rendered independently, and hence cannot properly characterize some inter-reflections and highlights that exist in the actual input images themselves. This explains why the shading error when training on the original dataset is a bit higher than when using the regenerated datasests (*RegenInput* and *RegenShade*), although it is still much lower than competing methods.

	MSE	PSNR	SSIM
InferenceNet - RU	71.12	30.00	0.961
InferenceNet - Scale	57.83	31.01	0.973
InferenceNet - Gradient	40.38	32.31	0.968
InferenceNet	38.01	32.61	0.973
InferenceNet + DT	30.42	33.56	0.981

Table 1: **Self-Comparison Study**. We justify the effectiveness of each component of our algorithm by removing "-" or adding "+" a single element each time: RU (residual units), Scale (rescaling layers), Gradient (gradient supervision), DT (domain transform). The experiments are performed using PASCAL VOC test data and training using the L0 filter.

Additionally, for more qualitative results and visualization of the different data types and issues, please refer to Figure 2. Here we see that the highlights in the left two ground truth shading images do not exist in the original input image, while the regenerated images do contain them. Moreover, we also observe that the original input images contain some textures or colors that do not exist in ground truth albedo images, like the orange hair of the girl in the bamboo scene, and the complex texture of the blanket in the left bottom image. These issues can impact performance, and are discussed further in the main paper.

6 More Results on Images Smoothing

We show more qualitative results in Figure 3, where the goal is to model the WMF filter with parameter setting from [8] that generates somewhat blurry images. Existing methods [4, 7], which primarily operate in the gradient domain (or weighted equivalent), may have difficulty mimicking this type of filtered response. In contrast, our approach handles a wide spectrum of image filters, from

		MSE			LMSE			DSSIM		
		albedo	shading	average	albedo	shading	average	albedo	shading	average
image split	MSCR [6]	0.0100	0.0092	0.0096	0.0083	0.0085	0.0084	0.2014	0.1505	0.1760
	JCNF [3]	0.0070	0.0090	0.0070	0.0060	0.0070	0.0070	0.0920	0.1010	0.0970
	Ours (*RegenInput*)	0.0038	0.0042	0.0040	0.0028	0.0031	0.0029	0.0758	0.0659	0.0708
	Ours (*RegenShade*)	0.0044	0.0046	0.0045	0.0033	0.0035	0.0034	0.1017	0.0733	0.0875
	Ours (*Original*)	0.0043	0.0057	0.0050	0.0032	0.0042	0.0037	0.1012	0.0827	0.0919
scene split	MSCR [6]	0.0201	0.0224	0.0213	0.0131	0.0148	0.0139	0.2073	0.1594	0.1833
	Ours (*RegenInput*)	0.0129	0.0171	0.0150	0.0083	0.0118	0.0100	0.1285	0.1252	0.1268
	Ours (*RegenShade*)	0.0151	0.0165	0.0158	0.0090	0.0106	0.0098	0.1555	0.1236	0.1395
	Ours (*Original*)	0.0158	0.0206	0.0182	0.0098	0.0135	0.0116	0.1652	0.1499	0.1575

Table 2: **Quantitative Comparison on MPI-Sintel Benchmark**. In addition to the regenerated datasets *RegenInput* and *RegenShade* described in the main paper, we trained our model on the *original* MPI-Sintel data and show all results for both *image split* and *scene split*.

those producing relatively blurry effects and some soft transitions, to canonical piece-wise constant outputs.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CVPR*, 2016.

[2] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. *ECCV*, 2016.

[3] S. Kim, K. Park, K. Sohn, and S. Lin. Unified depth prediction and intrinsic image decomposition from a single image via joint convolutional neural fields. In *ECCV*, 2016.

[4] S. Liu, J. Pan, and M.-H. Yang. Learning recursive filters for low-level vision via a hybrid neural network. In *ECCV*, 2016.

[5] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

[6] M. M. Takuya Narihira and S. X. Yu. Direct intrinsics: Learning albedo-shading decomposition by convolutional regression. In *ICCV*, 2015.

[7] L. Xu, J. S. Ren, Q. Yan, R. Liao, and J. Jia. Deep edge-aware filters. In *ICCV*, pages 1669–1678, 2015.

[8] Q. Zhang, L. Xu, and J. Jia. 100+ times faster weighted median filter. In *CVPR*, 2014.

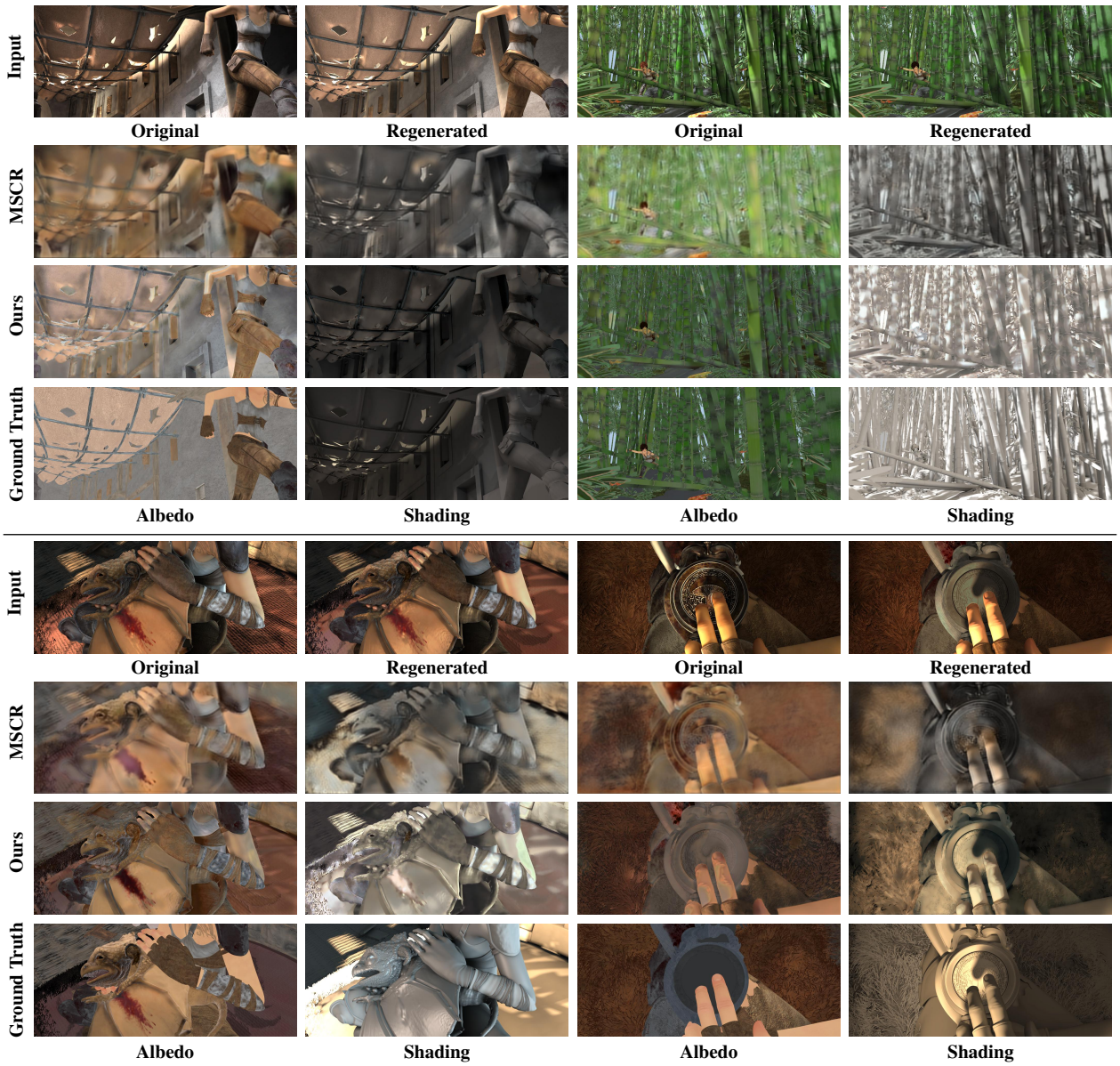

Figure 2: **Qualitative Comparison on MPI-Sintel Benchmark**. The visual results are evaluated on the model trained using *scene split* of *RegenInput* data. Compared to the state-of-the-art MSCR method [6], we show much sharper effects and better performance.

Figure 3: **Qualitative Comparison on Image Smoothing Task**. *Top*: Filtered results using the method from [4] (left), ours (middle), and ground truth (right). *Bottom*: From top to bottom row we show the input image, results from [7], ours, and ground truth. All the results are trained on the WMF filter with parameter settings from [7].