Deep learning based stylization and smoothing for images and videos

Feida Zhu

Supervisor: Prof. Yizhou Yu

Department of Computer Science
The University of Hong Kong

This dissertation is submitted for the degree of
Doctor of Philosophy

July 2018

I would like to dedicate this thesis to my beloved family ...

Declaration

I declare that this thesis represents my own work, except where due acknowledgement is made, and that it has not been previously included in a thesis, dissertation or report submitted to this University or to any other institution for a degree, diploma or other qualifications.

Feida Zhu
July 2018

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my Ph.D supervisor, Prof. Yizhou Yu for his continuous support to my Ph.D study and research. His patient guidance and excellent advice encourage and help me a lot when I face difficulties. I appreciate his unwavering support in all the time of my research and writing of this thesis.

Part of my work in this thesis is collaborated with Prof. Lei Zhang from The Hong Kong Polytechnic University and Dr. Zhicheng Yan from Facebook Inc. I especially would like to thank them for their support and collaboration. Many thanks to Prof. Mingli Song and Prof. Zicheng Liao for their help and guidance when I visit Zhejiang University.

I am very grateful for the valuable discussions with my fellow labmates: Guanbin Li, Wei Zhang, Xiaoguang Han, Li Zhang, Zhen Li, Weifeng Ge, Haofeng Li, Chaowei Fang, Bingchen Gong, Sibei Yang, Yatong An, Kan Wu, Chuan Wang, Guangmei Jing, Weikai Chen, Congyi Zhang, Lei Yang, Nenglun Chen. I also would like to thank my friends Xiaolong Zhang, Yating Yue, Chunting Zhou, Cheng Wang, Yupeng Li, Wei Liu, Zejun Guo, Weisong Fang for the joyful extracurricular life we had in the last four years.

Last but not least, I would like to express deep appreciation to my families for their endless love and constant support that have made the hard times so much easier. Without them as my strong backing, I would not have go so far in my education.

Abstract

In recent years, with the prosperity of digital imaging devices and social network, tremendous photographs have been recorded and sharing photos through the social media has been quite popular. Intelligent techniques of image analysis and enhancement have been increasingly important. Many efforts from the research and industry community have been made to push the development of state-of-the-art algorithms of automatic image and video stylization and edge-preserving image smoothing. Images and video stylization strives to enhance unique themes with artistic color and tone adjustments. Edge-preserving image smoothing is to preserve major image structures, such as salient edges and contours, while eliminating insignificant details. This thesis consists of novel deep learning based image and video stylization algorithms and a benchmark for edge-preserving image smoothing.

For image and video stylization, mainstream photo enhancement softwares, such as Adobe Photoshop and Instagram, provide users with predefined styles. However, such photo adjustment tools lack a semantic understanding of image contents and the resulting global color transform limits the range of artistic styles it can represent. More advanced stylistic enhancement needs to apply distinct adjustments to various semantic regions. We propose a novel deep learning architecture for exemplar-based image stylization, which learns local enhancement styles from image pairs. Our deep learning architecture consists of fully convolutional networks (FCNs) for automatic semantics-aware feature extraction and fully connected neural layers for adjustment prediction. Image stylization can be efficiently accomplished with a single forward pass through our deep network. To extend our deep network from image stylization to video stylization, we exploit temporal superpixels (TSPs) to facilitate the transfer of artistic styles from image exemplars to videos. Experiments on a number of datasets for image stylization as well as a diverse set of video clips demonstrate the effectiveness of our deep learning architecture.

For edge-preserving image smoothing, it is a fundamental problem in image processing and low-level computer vision. At present, there are reasons that seriously hinder its further development. First, its performance evaluation remains subjective. Second, there does not exist widely accepted datasets for both evaluation and research. Third, most existing algorithms cannot perform well on a wide range of image contents using a single parameter

setting. To remove the aforementioned hurdles in performance evaluation and further advance the state of the art, we propose a benchmark for edge-preserving image smoothing. This benchmark includes an image dataset with “groundtruth" image smoothing results as well as baseline learning algorithms that produce models capable of generating reasonable edge-preserving smoothing results for a wide range of image contents. Our image dataset contains 500 training and testing images with a large number of representative visual object categories. The baseline methods in our benchmark are existing representative deep convolutional network architectures, on top of which we design novel loss functions well suited for edge-preserving image smoothing. The trained deep networks run faster than most state-of-the-art smoothing algorithms while the smoothing performance of our ResNet-based model outperforms such algorithms both qualitatively and quantitatively.

[488 words]

Table of contents

List of figures

List of tables

Chapter 1

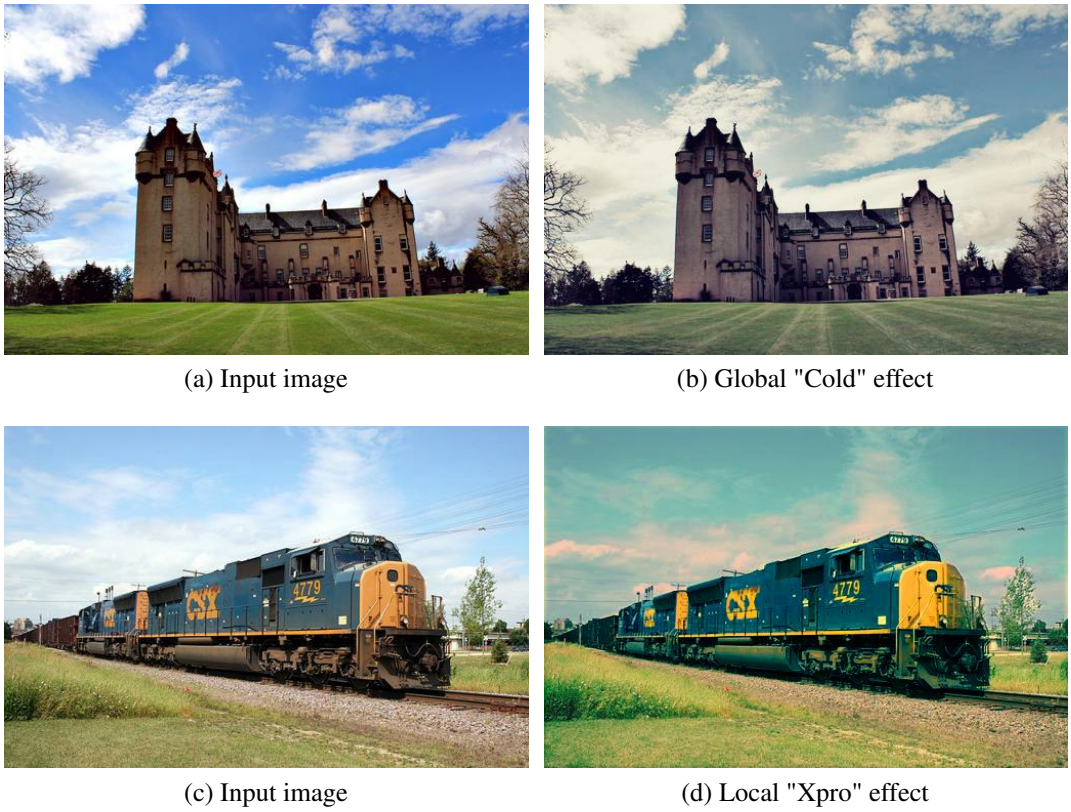
Introduction

1.1 Research Background and Motivation

With the prosperity of digital imaging devices and social network, it is popular that people take photos to record their life and share through social media. People often adjust the photos to make them look more stylistic or more pleasing before sharing through social media. Intelligent techniques of image analysis and enhancement have been increasingly important. Many efforts from the research and industry community have been made on automatic image and video stylization and edge-preserving image smoothing. Images and video stylization strives to enhance unique themes with artistic color and tone adjustments. Edge-preserving image smoothing is to preserve major image structures, such as salient edges and contours, while eliminating insignificant details. This thesis consists of novel deep learning based image and video stylization algorithms and a benchmark for edge-preserving image smoothing, aiming to push the development of state-of-the-art algorithms.

1.1.1 Image and Video Stylization

Image and video stylization strive to adjust the color and tone artistically to convey unique themes. Within Photoshop, some typical operations that photographer can make on the photo include adjustment of individual color curves, color blending, saturation adjustment, brightness/contrast manipulation and etc. For example, as shown in Figure 1.1 (a)(b), the *Cold* effect shared by an photo retouching enthusiast tends to make the photo look vintage by globally adjusting the color curves. In addition, some professional image editing softwares (such as Adobe Lightroom) and social mobile Apps (such as Instagram) provide users with predefined styles, which are often hand-crafted through a trial-and-error process. However, sophisticated stylization adjustments strive to present spatially varying effects according to



(a) Input image (b) Global "Cold" effect

(c) Input image (d) Local "Xpro" effect

Fig. 1.1 Image stylization examples. **Left**: Input image. **Right**: Manually enhanced by photographer. For local "Xpro" effect, distinct adjustments are applied to different semantic regions.

local statistics. Photographers often manually enhance images in a semantic-aware manner. For example, as shown in Figure Figure 1.1 (c)(d), a photographer utilizes the selection tools to isolate different semantic regions (human, building, train, etc.), which are then enhanced with distinct sets of adjustments. The property of applying locally distinct adjustments according to semantic regions enables more creativity and a broader range of visual styles, but is labor intensive at the same time. Data-driven approaches are capable of learning effects from examples, and then automatically apply learned styles on new-coming images. They replace time-consuming manual adjustments with automatic model learning. Global approaches [16, 72, 79, 11, 36] model the global color transform without considering local contexts. Recently, some local approaches [35, 43, 73] try to approximate complex spatially varying tone and color adjustments based on local low-level image statistics (e.g., color, intensity and histogram). Though these methods perform well on some examples, they may fail to learn semantic-aware styles when the context becomes complex since they didn't incorporate high-level contextual semantic information into their model.

Professional video editing softwares (Adobe After Effects, Nuke, etc.) offer a suite of adjustment operations (exposure/color correction, white balancing, sharpening, denoising, etc). However, to make pleasing local adjustments within specific spatiotemporal region, it needs to finely tune parameters and create masking layers with intensive user interaction, which are labor intensive processes. Bonneel *et al.* [10] propose to transfer the color palette of an example video to a novel input video. However, they still need to rely on users to provide foreground-background segmentation for estimating separate color transforms. The method proposed by Xue *et al.* [78] is solely dependent on low-level image statistics (luminance, hue and saturation), and is not able to support semantics-aware local adjustments.

Deep convolutional neural networks have shown great success in image classification [40, 65, 33]. The convolution layers can be regarded as feature extractor from low level to high level while the fully connected layers act like inference and decision making. Recently, fully convolutional networks [52, 13, 44] have proven to be efficient and powerful for image processing and visual understanding tasks, such as semantic image segmentation, contour detection and salient object detection, that need to generate high-resolution outputs. For our automatic image and video stylization task, generating high-resolution output is necessary. In this thesis, we propose a novel deep learning architecture to automatically learn stylistic enhancement from image exemplars. It consists of fully convolutional networks (FCNs) for feature extraction and fully connected layers for color transform prediction. The inner product of predicted color transform and input color basis is the predicted color after adjustment. In our deep network, feature maps with sufficiently large receptive fields are computed to represent semantic information. The fully connected layers are seamlessly integrated with the FCNs so that an input image can be enhanced with a single forward pass in our deep network.

Furthermore, our deep learning model trained on image exemplars can be extended to enhance videos with the learned artistic style. Compared with image stylization, some extra challenges need to be considered in video stylization such as the balance between spatial coherence and temporal coherence, computational efficiency. We utilize temporal superpixels (TSPs) [12], which are spatiotemporal regions consistently tracking image regions and object parts across frames. In a single frame, TSP looks like SLIC superpixel segmentation [1]. In our video stylization pipeline, a color transform is predicted for each TSP and then be applied to all pixels within the same TSP. Besides, semantic feature maps are only computed for a minimum number of video frames which intersect all TSPs to accelerate computation.

1.1.2 Image Smoothing

Edge-preserving image smoothing has been serving as a fundamental problem in many computer vision and graphics applications. It is important to preserve major image structures while eliminating insignificant details to effectively carry out image analysis and manipulation tasks, such as contour detection, image segmentation, content-aware image editing, etc. because real-world natural images often contain various trivial details or textures. Edge-

Input image

Bilateral Filter [71]

Rolling Guidance Filter (RGF) [86]

SD filter [31]

Tree Filtering [6]

Weighted Median Filter (WMF) [87]

L_0 smoothing [76]

Fast Global Smoother (FGS) [58]

L_1 smoothing [9]

Local Laplacian filter (LLF) [59]

Fig. 1.2 Image smoothing results produced by different edge-preserving image smoothing algorithms.

preserving image smoothing has received a great deal of attention and many algorithms with a diverse set of approaches have been proposed. Representative algorithms include Bilateral Filter [71], Anisotropic Diffusion (AD) [60], Weighted Least Square smoothing (WLS) [22], Edge-avoiding Wavelet (EAW) [23], Rolling Guidance Filter (RGF) [86], SD filter [31], L_0 smoothing [76], Fast Global Smoother (FGS) [58], Tree Filtering [6], Weighted Median Filter (WMF) [87], L_1 smoothing [9] and Local Laplacian filter (LLF) [59]. These algorithms generally aim to avoid edge blurring, halo artifacts, gradient reversal, and global intensity shift. Figure 1.2 shows an example and its smoothed results produced by different edge-preserving image smoothing algorithms.

Nevertheless, there exist three factors that hinder the further development of edge-preserving image smoothing algorithms. First, the evaluation of the algorithm performance is subjective. It's indeed hard to compare the quality of different smoothed images objectively. At present, the common way is to choose some examples and do visual comparison. User studies appear to be better than mere verbal justification. Second, any new edge-preserving smoothing algorithm is typically evaluated on a very small image collection. And even worth, there are no widely accepted image collections for this purpose. As a consequence, we can't tell whether the algorithm performs well on a wide range of images or just on few images. Third, the smoothing algorithms typically have tunable parameters. Different kinds of images may need different parameter setting for best performance. To the best of our knowledge, there do not exist smoothing algorithms that perform equally well on a wide range of image contents using a single parameter setting.

In this thesis, to overcome the aforementioned hurdles and further push the development of state-of-the-art smoothing algorithms, we propose a benchmark for edge-preserving image smoothing. The benchmark includes an image dataset of various image contents and the corresponding "groundtruth" image smoothing results selected by a group of users. The groundtruth smoothing results were not directly annotated by human, but manually chosen from results produced by existing state-of-the-art edge-preserving smoothing algorithms. This is because directly annotating the image pixel by pixel is too labor intensive and error prone. A state-of-the-art algorithm can produce high-quality smoothing result at least over a small range of image contents with proper parameter setting. A collection of such algorithms with different parameter settings are able to produce high-quality results over a much wider range of image contents. Our image dataset contains 500 training and testing images with a variety of visual object categories, such as human, animal, plant, landscape, vehicle and etc. We further propose two quantitative metrics, Weighted Root Mean Squared Error (WRMSE) and Weighted Mean Absolute Error (WMAE), for quantitative evaluation of edge-preserving image smoothing algorithms.

The benchmark also includes baseline learning algorithms that produce models capable of generating reasonable smoothing results for a wide range of image contents. To set up the baseline algorithms, we seek assistance from the latest deep neural networks. We note that deep learning has been successfully applied to many low-level computer vision problems, such as image denoising [56, 85], image super-resolution [37, 38, 68, 69, 42], and JPEG deblocking [19, 85]. Recently, several deep learning based algorithms [77, 49, 45, 21] strive to reproduce individual edge-preserving filters. In contrast, we do not aim to reproduce individual filters, but best results among a number of filters. A well trained deep neural network with fixed weights can produce high-quality results on a wide range of image contents. Besides, it is not necessary to tune parameters for different images once the weights are fixed. Specifically, we use two existing representative network architectures as our baseline methods, very deep convolutional networks (VDCNN) and deep residual networks (ResNet), on top of which we define novel loss functions well suited for edge-preserving image smoothing. The trained deep model run faster than most existing smoothing algorithms while the performance of our ResNet model outperforms previous algorithms both qualitatively and quantitatively.

1.2 Thesis Outline

This thesis is organized as follows:

Chapter 2 proposes a novel deep learning architecture for exemplar-based image stylization, which learns local enhancement styles from image pairs. The deep learning architecture consists of fully convolutional networks (FCNs) for global feature and semantics feature extraction, color histogram layer for contextual color histogram feature extraction, pixel color layer for pixel feature extraction. The concatenation of these features pass through fully connected neural layers to predict color transform. To extend the deep network from image stylization to video stylization, we exploit temporal superpixels (TSPs) to facilitate the transfer of artistic styles from image exemplars to videos. Experiments on a number of datasets for image stylization as well as a diverse set of video clips demonstrate the effectiveness of our deep learning architecture.

Chapter 3 exposes the factors that hinder the further development of edge-preserving image smoothing algorithms. To remove the hurdles in performance evaluation and further advance the state of the art, we propose a benchmark for edge-preserving image smoothing. This benchmark includes an image dataset with "groundtruth" image smoothing results as well as baseline learning algorithms that produce models capable of generating reasonable edge-preserving smoothing results for a wide range of image contents. Our image dataset contains

500 training and testing images with a large number of representative visual object categories. The baseline methods in our benchmark are existing representative deep convolutional network architectures, on top of which we design novel loss functions well suited for edge-preserving image smoothing. The trained deep networks run faster than most state-of-the-art smoothing algorithms while the smoothing performance of our ResNet-based model outperforms such algorithms both qualitatively and quantitatively.

Chapter 4 concludes the thesis and discusses possible directions for future research.

Chapter 2

Exemplar-Based Image and Video Stylization Using Fully Convolutional Semantic Features

Color and tone stylization in images and videos strives to enhance unique themes with artistic color and tone adjustments. It has a broad range of applications from professional image postprocessing to photo sharing over social networks. Mainstream photo enhancement softwares, such as Adobe Lightroom and Instagram, provide users with predefined styles, which are often hand-crafted through a trial-and-error process. Such photo adjustment tools lack a semantic understanding of image contents and the resulting global color transform limits the range of artistic styles it can represent. On the other hand, stylistic enhancement needs to apply distinct adjustments to various semantic regions. Such an ability enables a broader range of visual styles. In this chapter, we first propose a novel deep learning architecture for exemplar-based image stylization, which learns local enhancement styles from image pairs. Our deep learning architecture consists of fully convolutional networks (FCNs) for automatic semantics-aware feature extraction and fully connected neural layers for adjustment prediction. Image stylization can be efficiently accomplished with a single forward pass through our deep network. To extend our deep network from image stylization to video stylization, we exploit temporal superpixels (TSPs) to facilitate the transfer of artistic styles from image exemplars to videos. Experiments on a number of datasets for image stylization as well as a diverse set of video clips demonstrate the effectiveness of our deep learning architecture.

2.1 Introduction

Stylistic enhancement adjusts an image or video for enhancing artistic styles that convey unique themes. Unlike conventional image enhancement focusing on fixing photographic artifacts (under/over exposure, insufficient contrast, etc.), stylistic enhancement involves dramatic color and tone adjustments to achieve distinctive visual effects. For example, the *X-PRO II* filter from mobile photo App Instagram expresses a wistful mood by simulating the cross processing procedure of photographic films. Professional image editing software (such as Adobe Lightroom) and social mobile Apps (such as Instagram) provide users with predefined styles, which are often hand-crafted through a trial-and-error process.

Conventional automatic photo adjustment has difficulty in representing complex color transforms between images before and after adjustment. Most of them merely model global color transforms without considering local semantic contexts. Although more sophisticated adjustments introduce spatially varying effects according to local image statistics, they still lack a semantic understanding of image contents. On the contrary, professional photographers often manually enhance images in a semantics-aware manner. For instance, when enhancing photos to create a nostalgic theme, photographers might apply more exaggerated adjustments to a photo of *Broadway* taken in year 1950 than a photo of *Burj Khalifa*, which is the tallest skyscraper built in year 2010, as the former is more fitting with the theme. At a local scale, they employ selection tools to isolate semantic regions (faces, buildings, etc.), which are enhanced with distinct sets of adjustments. For instance, there may exist a demand to apply exaggerated adjustments to foreground objects to help them stand out. The ability of applying distinct adjustments to semantic regions enables a broader range of visual styles.

As stylistic adjustments interact with image semantics and contexts in a complicated manner, it is extremely challenging to manually define the relationships between them. To automatically learn stylistic enhancement from a small set of image exemplars, in this chapter, we propose a novel deep learning architecture. Unlike existing work that integrates hand-crafted features with a small-scale multilayer neural network [81], our solution is a large-scale deep network. It consists of fully convolutional networks (FCNs) for automatic feature extraction and fully connected neural layers for adjustment prediction. Recently, fully convolutional networks [52, 13, 44] have proven to be efficient and powerful deep learning architectures for image processing and visual understanding tasks, such as semantic image segmentation, contour detection and salient object detection, that need to generate high-resolution outputs. In our deep network, feature maps with sufficiently large receptive fields are computed to model contexts. We further employ fully connected neural layers, which predict color transforms according to contexts and pixel features. We seamlessly

Input image Ground truth Our result

Fig. 2.1 Examples of learning semantics-aware photo adjustment styles. **Left**: Input image. **Middle**: Manually enhanced by photographer. Distinct adjustments are applied to different semantic regions. **Right**: Automatically enhanced by our deep learning model trained from image exemplars

integrate the FCNs with fully connected layers, and an input image can be enhanced with a single forward pass in our deep network.

Furthermore, our deep learning model trained on image exemplars can be readily deployed to enhance videos with artistic styles. Compared with image stylization, video stylization faces extra challenges. Simply enhancing a video in a frame-by-frame manner not only results in an inefficient solution, but also cannot preserve the temporal coherence. Instead, we adopt temporal superpixels (TSPs) [12], which are spatiotemporal primitive regions consistently tracking image regions and object parts across frames. In our video stylization pipeline, a color transform is predicted for each TSP and applied to all pixels within the same TSP. To accelerate computation, features are only extracted for a minimum number of video frames intersecting all TSPs.

In summary, this chapter has the following contributions.

• We propose a novel deep learning architecture for stylistic image enhancement. It consists of fully convolutional networks and fully connected neural layers. Our deep network is capable of learning distinct enhancement styles from a small set of training exemplars. Enhancing a novel image only requires a single forward pass through our network.
• Fully convolutional networks in our architecture extracts global features and contextual features. Our novel contextual features have two parts. The first part is a semantics-aware feature extracted from deep layers of a fully convolutional network; the second part consists of a set of color histograms over a small spatial grid.
• We demonstrate that deep neural networks trained with image exemplars can be used to enhance videos as well. We segment a video into temporal superpixels, and apply both temporally coherent and spatially smooth adjustments to them. A greedy frame selection algorithm is developed to reduce the computational cost of feature extraction.

2.2 Related Work

Image Enhancement. On the basis of whether example data is used, image enhancement approaches can be broadly classified into two categories, hand-crafted approaches and data-driven approaches. Hand-crafted filters for image enhancement are commonly seen in image processing softwares and photo management Apps, such as Adobe Lightroom, Google Photos and Instagram. They support a range of adjustments, from exposure correction, contrast enhancement to artistic retouching. Meanwhile, researchers have made an enormous amount of effort to develop fully automatic methods for tone adjustment [5, 59], color management [15, 7], detail manipulation [22, 67, 66] and image smoothing [61, 14, 8]. On the other hand, interactive enhancement techniques allow users to perform adjustments at sparse locations, and propagate them to the full image domain [2, 48] while preserving image structures.

In contrast to hand-crafted approaches, which merely achieve a predefined set of effects, data-driven approaches are capable of learning new effects from examples, and thus offer a more flexible set of adjustments. They replace time-consuming manual design with automatic model learning [16, 72, 79]. Bychkovsky *et al.* [11] predict global tonal adjustments using a Gaussian process regression model built from a large dataset of images. Their regression model only extracts image global features and does not accommodate semantics-aware local adjustments. Kang *et al.* [36] introduce user preference in image global enhancement, and retouch a novel image by finding most similar examples in a database and transferring their tone and color adjustments. Joshi *et al.* [35] retouch imperfect personal photos by leveraging existing high-quality photos of the same person. Lee *et al.* [43] develop an unsupervised

technique for learning content-specific style rankings and transfers highly ranked styles from exemplars to an input photo. However, their styles are still limited to global color and tone transforms. Wang *et al.* [73] approximate complex spatially varying tone and color adjustments with piecewise polynomial functions, which rely on low-level image statistics only and are not aware of image semantics. In contrast, our model applies adjustments to local semantic regions using features extracted with a deep convolutional neural network pretrained on thousands of semantic categories. Shih *et al.* [64] synthesize images associated with different times of day by learning locally affine models after locating a matching video within a time-lapse video database. Gatys *et al.* [27] perform image style transfer using convolutional neural networks. A new image is synthesized by matching the coarse structures of a content image and the texture features of a style image.

The work closely related to ours is presented in [81], where local tone and color adjustments are predicted using a combination of image global statistics, contextual semantic features and pixelwise color and spatial features. They rely on existing computationally expensive scene parsing [70] and object detection [74] tools to explicitly generate a semantic label map, from which contextual features are formed by multiscale spatial pooling. These scene parsing and object detection tools have limited accuracy and robustness. Our method in this chapter does not rely on explicit scene labeling, instead, perform scene understanding implicitly by extracting contextual semantic features using a fully convolutional network, which offers higher accuracy and improved robustness, and also runs faster than traditional scene understanding tools.

Video Enhancement. Traditional professional video editing softwares (Adobe After Effects, Nuke, etc.) offer a suite of predefined operations with tunable parameters that apply common global adjustments (exposure/color correction, white balancing, sharpening, denoising, etc). Local adjustments within specific spatiotemporal regions are usually accomplished with masking layers created with intensive user interaction. Both parameter tuning and masking layer creation are labor intensive processes.

Example-based approaches have been proposed to automatically transfer adjustments from exemplars to novel videos, and alleviate the needs of user interaction. Bonneel *et al.* [10] propose to transfer the color palette of an example video to a novel input video to achieve color grading. However, they rely on users to provide foreground-background segmentation for estimating separate color transforms. In contrast, our approach is fully automatic and implicitly distinguishes semantic regions from each other by using deep features from convolutional neural networks. Xue *et al.* [78] study the relationships between film tags (director, emotion, etc.) and color styles for movie color grading. Their method is solely dependent on low-level image statistics (luminance, hue and saturation), and is

not able to support semantics-aware local adjustments. Ruder *et al.* [63] extend [27] to video stylization by proposing a temporal loss term between frames to maintain temporal coherence.

2.3 Overview

Given a set of exemplar image pairs, each representing a photo before and after pixel-level color and tone adjustments following a particular style, we wish to learn a computational model that can automatically adjust a novel input photo in the same style. We still cast this learning task as a regression problem as in [81]. For completeness, let us first review their problem definition and then present our new deep learning based architecture and solution.

We seek a color transformation function ϕ such that, for every pixel p_i in the exemplar images, the color transform returned by ϕ is $\phi(\theta, x_i)$, which maps the pixel color at p_i before adjustment, $c_i = [L_i\ a_i\ b_i]^T$ (CIELab color space), to its corresponding pixel color y_i after adjustment. Here θ denotes the model parameters and x_i the feature vector at pixel p_i. The quadratic color basis $V(c_i) = [L_i^2\ a_i^2\ b_i^2\ L_ia_i\ L_ib_i\ a_ib_i\ L_i\ a_i\ b_i\ 1]^T$ is used to absorb high-frequency pixelwise color variations. The product of the color transform $\phi(\theta, x_i)$ and the color basis is a prediction of the enhanced color. Since our color space has 3 channels and the quadratic color basis is a 10-dimensional vector, $\phi(\theta, x_i)$ is in fact a 3x10 matrix. The model parameters of θ are learnt by minimizing the following objective function, which measures the squared differences between the predicted and groundtruth enhanced colors.

$$\arg\min_{\theta} \sum_{i} ||\phi(\theta, x_i)V(c_i) - y_i||^2 \tag{2.1}$$

Since each color transform is a matrix with 30 elements, solving a distinct color transform at every pixel is an under-constrained problem. To sufficiently constrain every color transform, we group pixels in an image into a predefined number of superpixels, $\{s_v\}_{v=1}^{N_s}$, and let all pixels within a superpixel s_v share a single color transform $\mathscr{F}_v = \phi(\theta, x_v)$. Thus, the above objective function is revised as follows.

$$\arg\min_{\theta} \sum_{v} \sum_{j \in s_v} ||\phi(\theta, x_v)V(c_j) - y_j||^2. \tag{2.2}$$

Refer to [81] for more details.

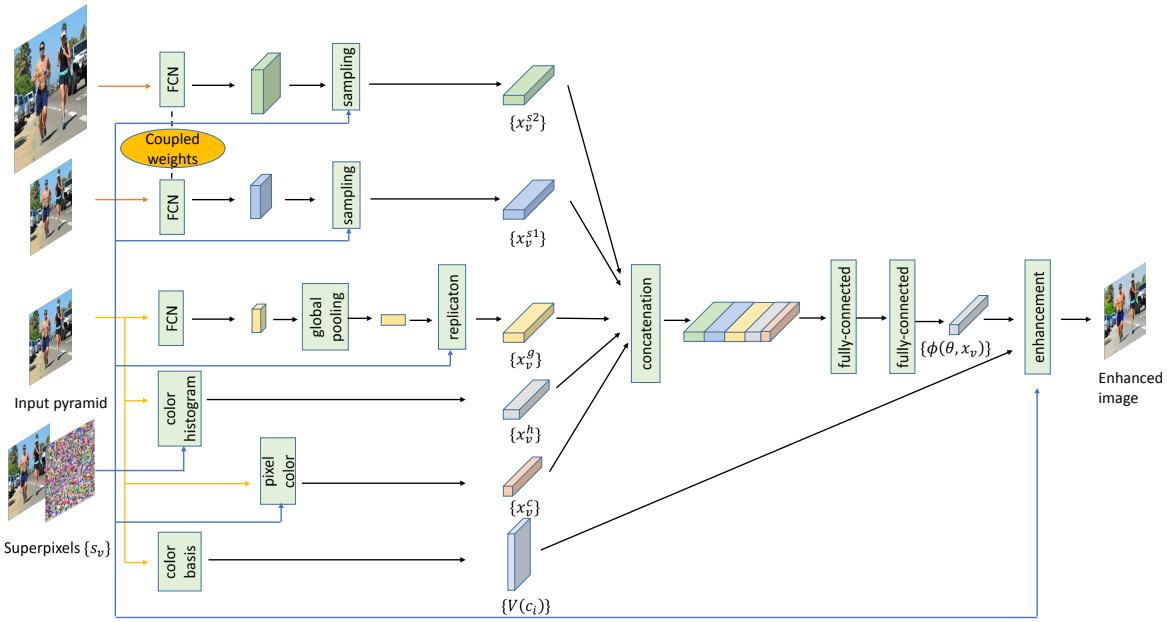


Fig. 2.2 The architecture of our deep neural network for stylistic image enhancement.

2.3.1 Photo Stylization Using FCNs

In this chapter, we model the entire process to produce an enhanced image using deep neural networks. The complete architecture of our deep network is shown in Figure 2.2. Our deep network makes use of fully convolutional networks to extract global and contextual semantic features for every superpixel in the input image. The global feature of a superpixel is the globally pooled deep CNN feature. And the pixel feature is simply the pixel color at the centroid of the superpixel. As shown in Figure 2.3, the contextual feature of a superpixel consists of three components. The first two components are deep CNN features extracted over two differently sized receptive fields centered at the centroid of the superpixel. The third component is a set of concatenated color histograms computed over a 3x3 grid also centered at the centroid of the superpixel. Let us now explain the architecture of our deep network and how we compute these features in greater details.

An input image is fed into a fully convolutional network (FCN). Deep features extracted using this FCN pass through a *global pooling* layer and become a single global feature vector x^g. We replicate x^g at the centroids of all superpixels in a *replication* layer to obtain per-superpixel global features $\{x_v^g\}_v$ (Section 2.4.1). We also upsample the input image, and feed the original and upsampled images into two FCNs to extract two feature maps. The receptive fields of these two feature maps respectively have the same size as the two (blue) windows in Figure 2.3. These feature maps represent semantic contexts at two different scales, and are sampled at the centroids of superpixels in a *sampling* layer to obtain contextual

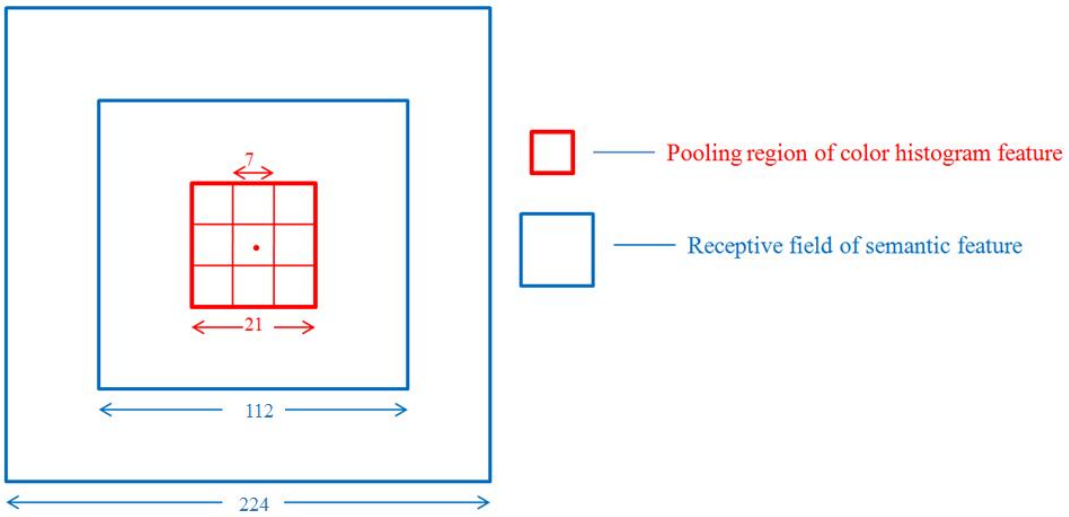


Fig. 2.3 Contextual feature descriptors. There are both semantic features and color histogram features in our contextual feature description. Two-scale contextual semantic features are extracted from the two large blue windows using a fully convolution network while color histogram features are extracted from 9+1=10 pooling regions over a 21×21 window.

semantic features $\{x_v^{s1}\}_v$ and $\{x_v^{s2}\}_v$ (Section 2.4.2). We compute contextual color histogram features $\{x_v^h\}_v$ over two-scale pooling regions (red grid in Figure 2.3) in a *color histogram* layer (Section 2.4.2). The global feature, contextual semantic features, contextual color histogram feature as well as the pixel feature for every superpixel are concatenated and passed through two fully connected neural layers to produce the set of per-superpixel color transforms $\{\phi(\theta, x_v)\}_v$. Each of these color transforms is applied to the per-pixel color basis vectors $\{V(c_i)\}_i$ in the same superpixel in an *enhancement* layer to produce the final enhanced image.

Global pooling layer. Given an incoming feature map $\{f_{x,y,c}\}$ of size $H \times W \times C$, a global pooling layer performs average pooling over the full spatial domain to compute a global feature vector $x^g = \frac{\sum_x \sum_y f_{x,y,c}}{H \times W}$.

Replication layer. This layer replicates the input feature vector x^g with C dimensions at the centroid of every superpixel and produces a feature map $\{x_v^g\}$ of size $N_s \times C$.

Sampling layer. Given a feature map of size $H \times W \times C$, this layer samples a feature vector at the centroid of every superpixel and assembles them into a feature map of size $N_s \times C$.

Color histogram layer. This layer computes a contextual color histogram feature with D dimensions at the centroid of every superpixel. These color histogram features are assembled into a feature map of size $N_s \times D$. The details of contextual color histogram computation are elaborated in Section 2.4.2.

Fig. 2.4 Visualization of global feature where images are displayed exactly at their embedded location.

Enhancement layer. This layer predicts the enhanced colors at all pixels within every superpixel s_v. For each pixel p_j in s_v, if the original color at p_j is c_j, the enhanced color is computed as the product of the predicted color transform $\phi(\theta, x_v)$ and the quadratic color basis vector $V(c_j)$.

Discussion. Inspired by the design of feature descriptors in [81], we use the combination of the global feature, contextual features and the pixel feature to predict the color transform. However, there exist significant differences between our features and those in [81]. While various types of low-level image statistics (e.g. *intensity distritbuion, scene brightness, equalization curves*) are used as image global features in [81], our global feature is computed with a global pooling layer which spatially averages deep features from the last convolutional layer of a fully convolutional network. The resulting global feature is shown to be discriminative with respect to the semantics of image contents (e.g. an indoor portrait vs. an outdoor landscape) [51]. We use t-SNE [54] to compute a two-dimensional embedding of global

features. This embedding arranges images with similar global features close to each other in a plane. A visualization of the resulting spatial layout is shown in Figure 2.4. To form a contextual feature descriptor for representing the local surrounding of a superpixel, Yan *et al.* [81] rely on a traditional scene parser [70] to label background image regions, and a set of cascaded object detectors [74] to detect and classify foreground objects into a small set of predefined categories. By merging scene parsing results and object detections into a semantic label map, they compute label histograms using a multiscale spatial pooling scheme. However, both the scene parser and the object detectors they use have been substantially outperformed by recent deep CNN models [52, 13, 62, 50]. Furthermore, in contrast to the continuous deep CNN features used in our contextual features, their category-oriented discrete label map is more prone to quantization errors, which can lead to severe artifacts in the final results.

Moreover, compared to our fully convolutional network with an efficient GPU implementation, the scene parser and object detectors they use are at least one order of magnitude slower, and cannot be seamlessly integrated with their multi-layer Perceptron network. In contrast, our entire deep network performs photo stylization in an end-to-end manner without any external dependency.

2.3.2 Video Stylization

To make our deep network trained on image exemplars perform video stylization, we choose to represent the video with temporal superpixels (TSPs) [12] (Section 2.6.1). TSPs pay particular attention to the temporal dimension, and explicitly model the motion flow between frames in a probabilistic generative framework. As a result, they can track image regions and object parts over a period of time.

To exploit the tracking ability of TSPs in maintaining the temporal coherence in the enhanced video, we associate each TSP with a single color transform (Section 2.6.2). To reuse our trained deep network for predicting per-TSP color transforms, we can project a TSP onto video frames to obtain a sequence of temporally adjacent superpixels. As described in Section 2.3.1, we could predict color transforms associated with those superpixels and aggregate them to obtain the per-TSP color transform. Since each TSP only needs one color transform, it is not necessary to predict a color transform for every projected superpixel. To minimize the number of frames used for feature extraction, we propose a greedy algorithm to choose a small set of representative frames. In Section 2.7, we empirically demonstrate that this technique clearly reduces the computational cost without compromising the visual quality.

2.4 Feature Description

An image is decomposed into a set of superpixels with the graph-based segmentation algorithm in [24], and we seek to predict a single color transform $\phi(\theta, x_v)$ for every superpixel s_v using the feature vector x_v described in this section. This feature vector consists of three components, namely the global feature, the contextual feature and the pixel feature.

2.4.1 Global Features

The overall image content affects how professional photographers adjust the image. We employ a fully convolutional network to extract image global features. FCN is introduced in [52] for semantic image segmentation. It can be set up by transforming a convolutional neural network (CNN). The CNN can be pretrained on a large-scale dataset for image classification, such as ImageNet [17], to learn discriminative feature representations, which are generically useful for a set of related tasks [18, 28, 55]. We take the VGG-16 network [65] pretrained on ImageNet images from 1000 object categories. This network consists of 5 groups of convolutional layers (*conv1-conv5*), 5 pooling layers (*pool1-pool5*) and 2 fully connected layers (*fc6* and *fc7*). We replace layers *fc6* and *fc7* with new convolutional layers *conv6* and *conv7* while kernel parameters in *conv6* and *conv7* are copied from *fc6* and *fc7*, respectively. An input image of size $H \times W \times 3$ is sent into this FCN and a feature map of size $H^g \times W^g \times 4096$ is extracted from the deepest layer *conv7* which has a sufficiently large receptive field of size 224×224. The spatial dimensions of the feature map is reduced by a factor of 32 due to the 5 pooling layers. Thus $H^g = \frac{H}{32}$, and $W^g = \frac{W}{32}$. We perform global average pooling to obtain a 4096D global feature, whose dimensionality is further reduced to 200 using PCA to prevent overfitting during the network training stage. The resulting 200D feature x_v^g is replicated at the centroids of all superpixels.

2.4.2 Contextual Features

As photographers apply spatially varying adjustments to various regions in a photo according to the local contents and their appearances in these regions, we introduce semantic features over two different scales and color histogram features to differentiate among local contexts.
Semantic Features. Apart from the FCN taking the input image in its original size to extract the global feature, two more FCNs with identical weights are used to extract contextual semantic features. On the basis of the FCN used for global feature extraction, we make two crucial modifications to the two FCNs used here. First, we reduce the stride of pooling kernels in layers *pool4* and *pool5* from 2 to 1, and thus the spatial resolution of feature

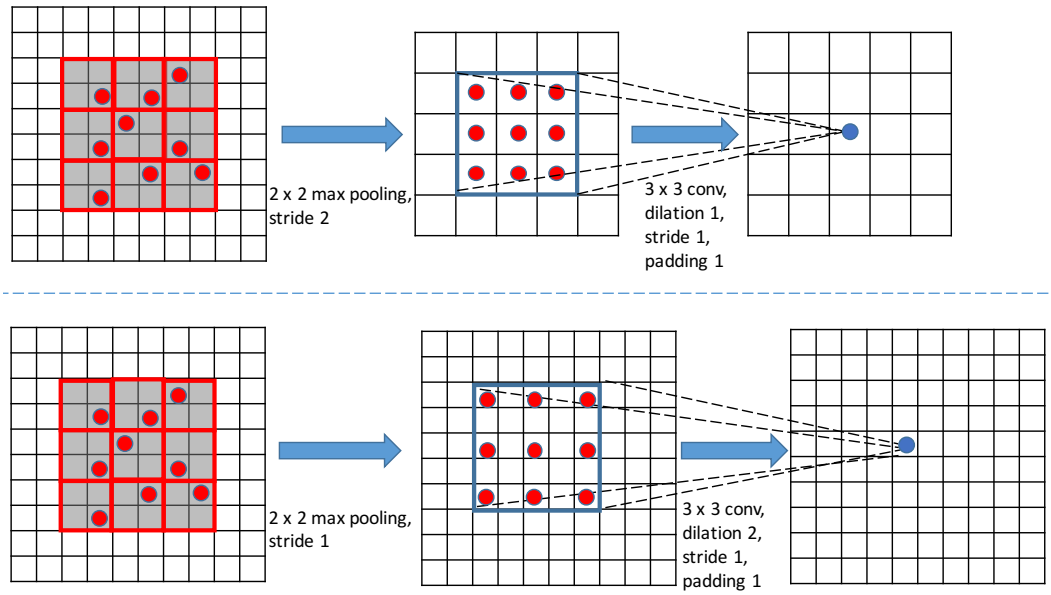


Fig. 2.5 Dilated convolution. **Top**: The spatial resolution of feature maps is reduced by half after a max-pooling operation with stride 2, and the subsequent 3×3 convolution is not dilated (i.e. dilation = 1). **Bottom**: The stride of max-pooling is reduced to 1. The spatial resolution is preserved and the convolution is dilated to have input stride 2 (i.e. dilation = 2).

maps is not reduced across these layers. As a result, the overall spatial resolution is merely reduced by a factor of 8. Feature maps with high spatial resolutions have been shown to be vital in attaining superior performance in image processing tasks [52, 80]. Changing the kernel stride at a pooling layer alone makes the following convolutional layer have a different receptive field and extract meaningless features. We use *dilated convolution* [83] to increase the input stride of feature maps for layers *conv5* and *conv6* by factors of 2 and 4 respectively to compensate the change of pooling kernels in *pool4* and *pool5* (Figure 2.5). We further fine-tune this transformed VGG16 network on the Places2 scene classification dataset [88], which makes deep features produced by *conv7* more discriminative across different semantic regions.

Second, besides extracting semantic features at the original image resolution, we also upsample the input image by a factor of 2 and feed the upsampled image into one of the aforementioned FCNs to extract features from the *conv7* layer, the size of whose receptive field is reduced from 224×224 to 112×112. This two-scale scheme is better suited for context modeling. To ensure meaningful features are extracted at image boundaries, we apply 112-pixel wide reflection padding to the upsampled images. Finally, if an input image of size $H^u \times W^u \times 3$ is sent into any of these modified FCNs, a feature map of size $\frac{H^u}{8} \times \frac{W^u}{8} \times 4096$ is extracted from layer *conv7*. To prevent overfitting, the 4096D feature at each pixel of

this feature map is reduced to a 800D feature using PCA. Thus, at the centroid of every superpixel, there are two concatenated contextual features, which are respectively sampled from the nearest pixels in layer *conv7* of these two FCNs. Thus, every superpixel has a 1600D contextual semantic feature.

Color Histogram Features. To represent the appearances of local contents, we also compute a color histogram feature by performing two-scale spatial pooling over a grid of cells, as illustrated in Figure 2.3. Inspired by spatial pyramid pooling [41], we place a 3×3 grid of cells at the centroid of every superpixel. For each channel in the CIELab color space, we compute a histogram over each of the 9 cells, each of which has 7×7 pixels, as well as over their bounding box, which has 21×21 pixels. Such spatial pooling makes our local color histograms more robust to spatial deformations. We use 32 bins in each histogram and concatenate all histograms into a 960D feature vector.

Due to the relatively large spatial extent of the grid used for histogram computation, color histograms for a superpixel near an object or region boundary may be heavily influenced by pixels on the other side of the boundary. Such color histograms can mislead our deep network to apply incorrect adjustments to the superpixel and further give rise to halo artifacts, especially when pixel colors on both sides of the boundary have significant differences. We mitigate this problem by altering pixel weights during histogram binning. Specifically, before computing color histograms at the centroid of a superpixel, the weight associated with a pixel in the aforementioned grid is set to the following bilateral coefficient, $w_i = e^{\frac{|c_i - c_v|^2}{\sigma_c}} e^{\frac{|p_i - p_v|^2}{\sigma_p}}$, where c_v and p_v respectively denote the color and location of the centroid, c_i and p_i respectively denote the color and location of the pixel, and σ_c and σ_p represent the estimated standard deviations of colors and distances respectively. We further normalize all pixel weights and use such weights during histogram binning afterwards.

2.4.3 Pixel Features

Pixel colors represent high-resolution spatial variations. We represent pixel colors in the CIELab color space. The pixel feature of a superpixel is simply the 3D pixel color at the centroid of the superpixel.

2.5 Experimental Results on Image Stylization

2.5.1 Experimental Setup

As shown in Figure 2.2, we employ FCNs to extract image global features and contextual semantic features. All the features of a superpixel are concatenated and fed into a deep neural network with one input layer, two fully connected hidden layers, and one output layer. The number of neurons in the hidden layers are set empirically to 256, and the number of neurons in the output layer are set equal to the number of coefficients in the predicted color transform. Since we use quadratic color transforms, there are 30 neurons in the output layer, 10 for each of the three color channels.

At the training stage, only the weights associated with the fully connected layers are updated with the classic error back-propagation algorithm. In practice, each image is segmented into around 7000 superpixels, from each of which 10 pixels are sampled. Therefore, even if we only have 50 example image pairs for learning one specific style, the total number of training samples is still as large as 3.5 million. Such a size of the training set can largely eliminate the risk of overfitting. It typically takes a few hours to finish training the deep neural network on a training dataset with hundreds of image pairs. At the testing stage, we still segment each image into around 7000 superpixels and the features extracted for each superpixel at its centroid are shared among all pixels within the same superpixel. The enhanced result of a novel testing image can be computed in just one forward pass through our deep network. It takes around 25 seconds to enhance an image 512-pixel wide.

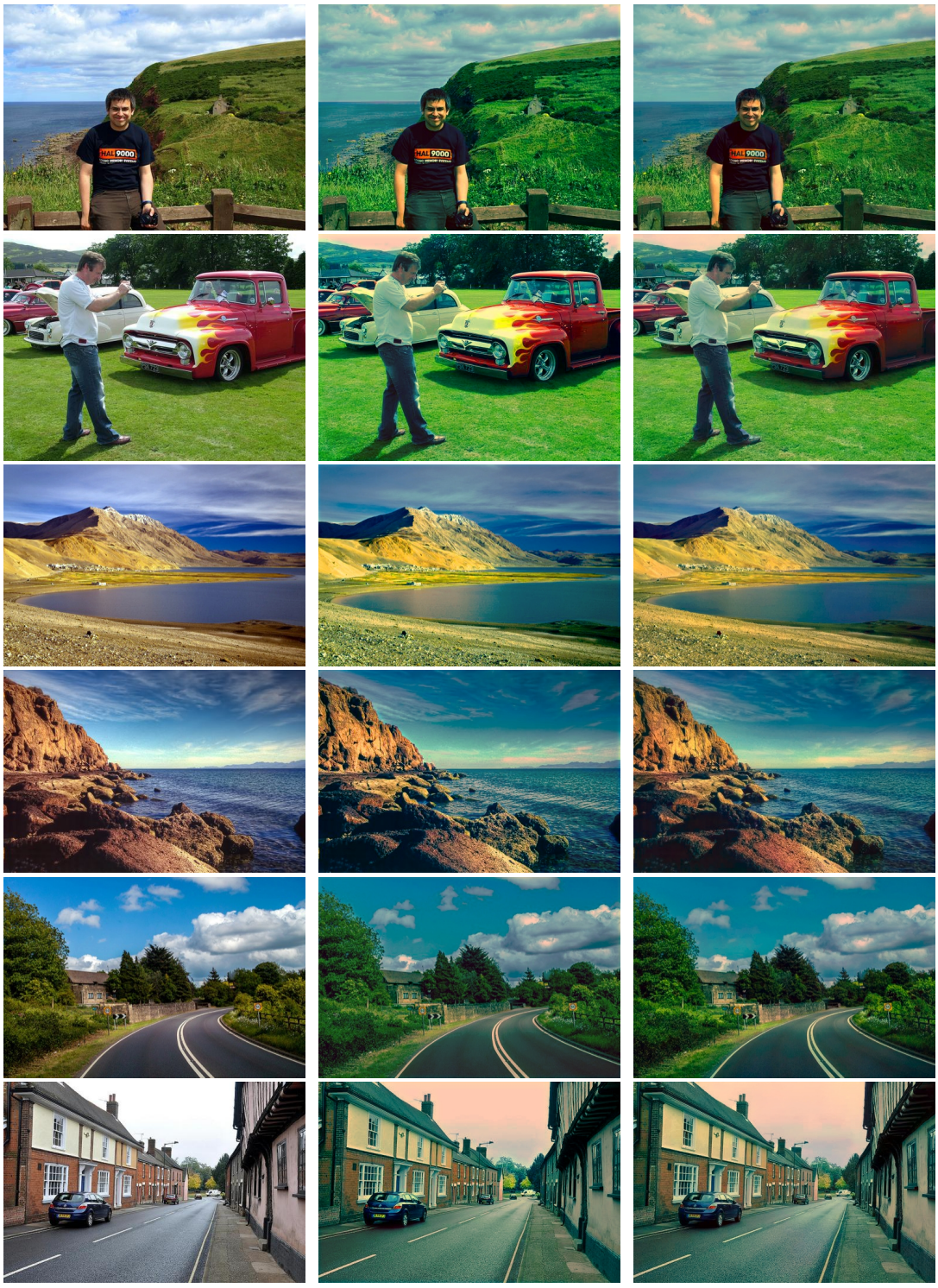
2.5.2 Results on the Uniform Dataset

Here we report image stylization results from our deep network on the Uniform dataset introduced in [81]. This dataset includes 115 images and three local enhancement styles. In the following, we briefly review these three styles.

The first style, "**Foreground Pop-out**", increases the contrast and color saturation of foreground objects while decreasing the color saturation of background. In general, this style makes the foreground objects more salient and colorful while deemphasizing the background. The second style, "**Local Xpro**", is more complex compared to the previous effect. It generalizes the popular "cross processing" effect in a local manner. When creating this style, a photographer first defined multiple profiles in Photoshop, each of which is specifically tailored for a subset of semantic categories. All the profiles share a common series of operations, such as hue/saturation adjustment and brightness/contrast manipulation. Nonetheless, each profile defines a distinct set of adjustment parameters tailored for its corresponding semantic

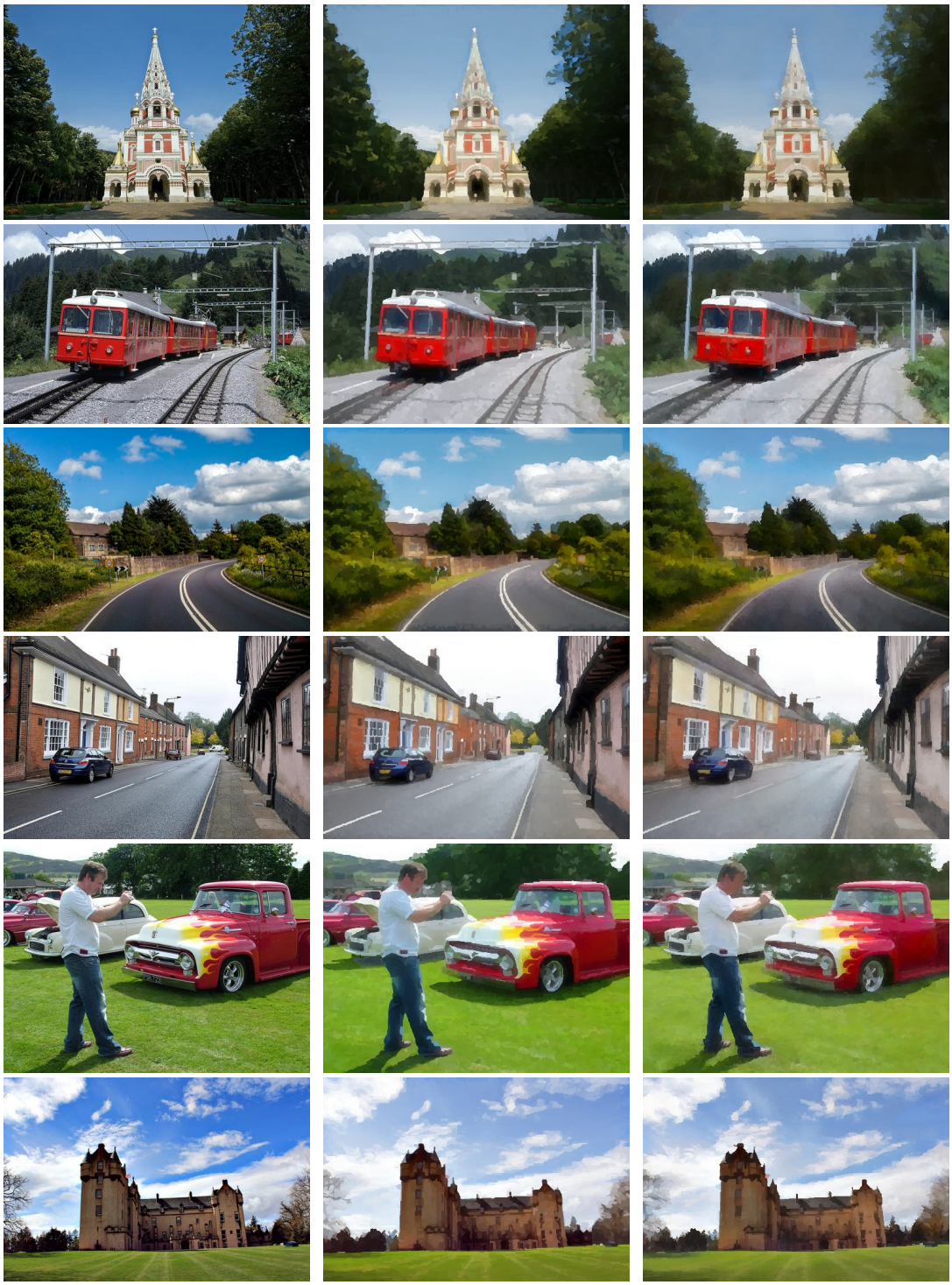
Input image Ground truth Our result

Fig. 2.6 Examples of Foreground Popout effect. **First column:** Input image. **Second column:** Ground truth. **Third column:** Our result.



Input image Ground truh Our result

Fig. 2.7 Examples of Local Xpro effect. **First column:** Input image. **Second column:** Ground truth. **Third column:** Our result.



Input image Ground truh Our result

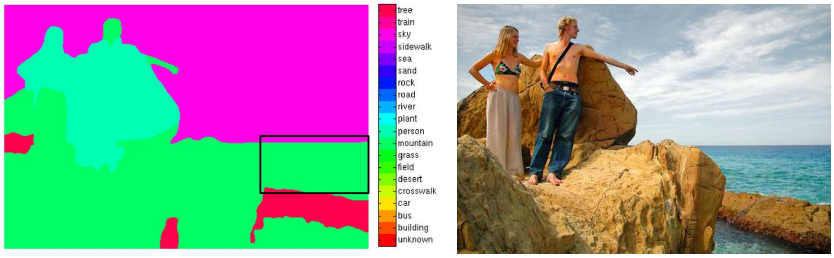
Fig. 2.8 Examples of Watercolor effect. **First column:** Input image. **Second column:** Ground truth. **Third column:** Our result.

categories. Although the profiles roughly follow the “cross processing" style, the choice of local profiles and additional minor image editing operations were heavily influenced by the photographer’s personal taste which can be naturally learned through exemplars. The third style, “**Watercolor**", gives viewers artistic impressions. It creates “brush" effects over the input image. All pixels inside the region covered by a single brush stroke share the same color. This mimics the “watercolor" painting style. This style also gives rise to complex spatially varying color adjustments.

Style	Groundtruth L^2 distance	Global Transform	Yan et al. [81]	Our Method
Foreground Pop-out	13.86	6.60	7.08	6.39
Local Xpro	19.71	6.31	7.43	5.55
Watercolor	15.30	7.23	7.20	6.44

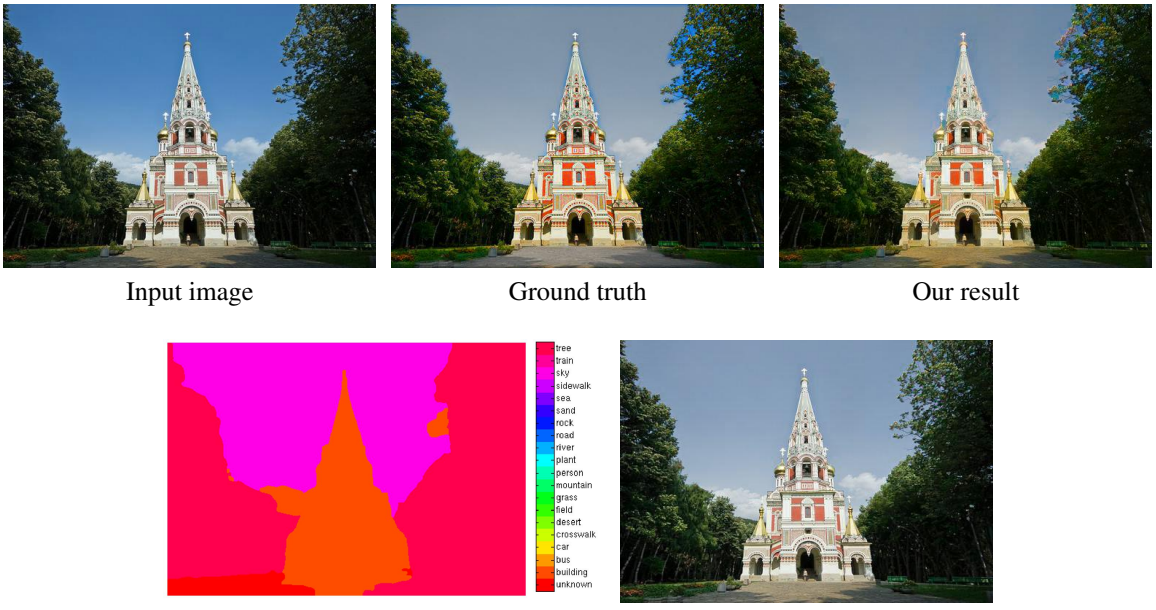
Table 2.1 Mean per-pixel L^2 distances between input images and groundtruth adjusted images and mean per-pixel L^2 testing errors between automatically adjusted results and groundtruth results.

Input image Ground truth Our result



Scene labels from [81] Result from [81]

Fig. 2.9 Comparison with the method in [81] on an example from the Foreground Popout style. Their method mislabels ‘sea’ as ‘mountain’ and the saturation of this mislabeled region is incorrectly increased. In contrast, our method produces much more robust visual results by using deep contextual and global features extracted using fully convolutional networks.



Input image | Ground truth | Our result

Scene labels from [81] | Result from [81]

Fig. 2.10 Comparison with the method in [81] on another example from the Foreground Popout style. Although their method labels the 'building' region correctly, it still adjusts its color incorrectly, which reveals the limitation of their feature description. In contrast, our features give rise to an enhanced image closer to the ground truth.

We have trained our deep network to learn all three styles from this dataset. As in [81], for each style, 70 images are used for training and the remaining 45 images are used for testing. Fig. 2.6, Fig. 2.7 and Fig. 2.8 show some testing results in these three styles. We have calculated the mean per-pixel L^2 distance in the CIELab color space between our enhanced images and the groundtruth results as well as between the input images and the groundtruth results. They are shown in Table 2.1. We also add a baseline which is the L^2 distance between the ground truth and the result obtained with per-image best global quadratic color transform. It provides the theoretical lower error bound for image adjustment using a global quadratic color transform. Our method can achieve even lower numerical errors, which indicates the importance of spatially-varying local color transforms.

In this chapter, we primarily compare our method against the one proposed by Yan *et al.* [81] under the same experimental setting. This is because the method in [81] is the most recent work capable of performing local semantics-aware color and tone stylization. It has been demonstrated in [81] that their method outperforms all other earlier relevant techniques. Comparison of numerical errors on testing images are shown in the fourth and fifth columns of Table 2.1. Our method achieves lower numerical errors on all the three local enhancement styles.

In addition to lower numerical errors, our method achieves clearly higher visual quality for the following reasons. First, their contextual feature is based on a label map generated from a scene parser [70] and object detectors [74], which tend to make unreliable discrete labeling decisions while our method avoids such discrete decisions by using continuous deep features. Therefore, their enhanced results are more likely to have artifacts due to incorrect labeling. One such example from the Foreground Popout style is shown in Fig 2.9 , where 'sea' is mislabeled as 'mountain' with their method and the saturation of this mislabeled region is incorrectly increased. Second, their method requires the definition of a set of semantic categories. When this set is limited (20 categories used in [81]), their contextual feature would not work well on testing images containing object categories beyond this predefined set. In addition, the limited number of categories also limits the effectiveness of their feature. Another example from the Foreground Popout style is shown in Fig 2.10, where the saturation of 'building' is not increased with their method even though it is labeled correctly. In contrast, our FCN based architecture extracts contextual features without using

Input image | Result from [73]

Our result | Ground truth

Fig. 2.11 Comparison with the method in [73] on an example in the Local Xpro style. Our enhanced result is visually closer to the manually enhanced ground truth.

a category set and is more robust. The color of the 'building' region is correctly adjusted with our method.

Another comparison has been conducted against the method proposed by Wang *et al.* [73], which tries to approximate complex spatially varying tone and color adjustments with piecewise polynomial functions based on low-level image statistics. Our method relies on more powerful features, which not only include low-level image statistics such as color histograms but also high-level contextual semantic information. An example is shown in Fig 2.11, where we can see that our enhanced result is much closer to the ground truth.

2.5.3 Effectiveness of Semantic and Color Histogram Features

Our contextual feature consists of two parts: semantic feature and color histogram feature. In this subsection, we demonstrate the importance of these individual features as well as their integration in our method.

We conduct experiments using four different contextual feature combinations, including concatenated semantic and color histogram features, semantic feature alone, color histogram feature alone and no contextual feature at all. The mean per-pixel L^2 testing errors in the CIELab color space are summarized in Table 2.2. Without using any contextual features, the testing errors are 10.10, 9.76 and 8.45 on the Foreground Pop-out, Local Xpro and Watercolor styles respectively. These errors drop to 8.75, 8.45 and 7.72 respectively after adding semantic features and drop to 8.68, 8.60 and 7.66 respectively after adding color histogram features. These results numerically indicates the importance of these individual features. Moreover, the testing errors can be further reduced to 7.14, 7.19 and 6.78 after integrating these two features together, which indicates the necessity of using both features in our method.

We also provide an example of visual comparison in Figure 2.12. This example is from the Foreground Popout style. In this example, 'person' is classified as foreground and 'sky' and 'sea' are classified as background. We can see that the result obtained using both

Style	Both features	Semantic features only	Color features only	Without context features
Foreground Pop-out	**6.39**	8.75	8.68	10.10
Local Xpro	**5.55**	8.45	8.60	9.76
Watercolor	**6.44**	7.72	7.66	8.45

Table 2.2 Comparison of mean per-pixel L^2 testing errors achieved with different combinations of contextual semantic and color histogram features.

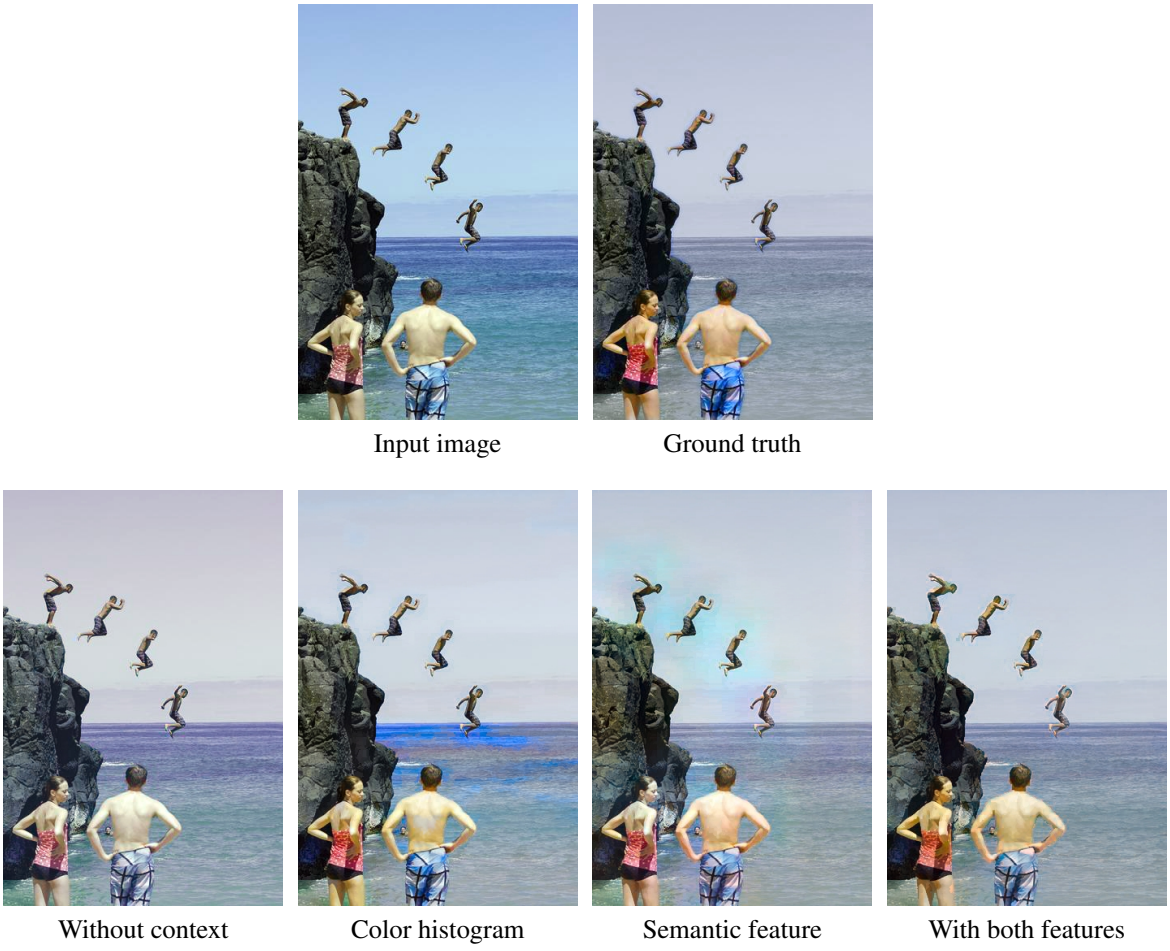


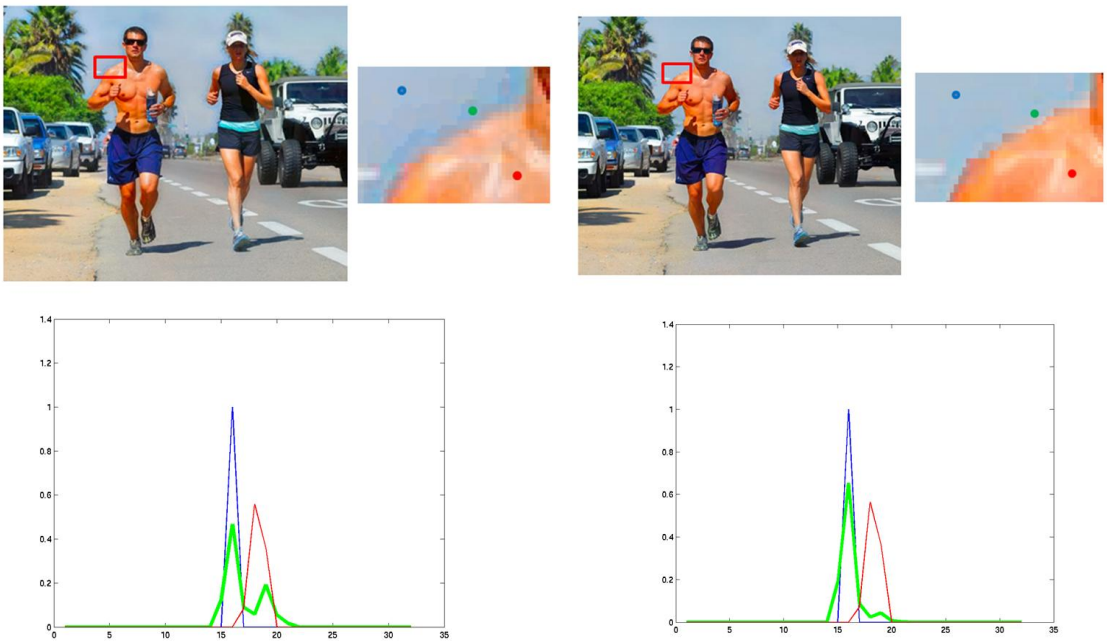
Fig. 2.12 Comparison of visual results produced with different combinations of contextual semantic and color histogram features.

semantic and color histogram features is visually closer to the groundtruth result than those obtained using one type of contextual features only.

Bilateral Weights in Color Histogram Features

Here we verify the effectiveness of bilateral weights in color histogram features. A comparison of enhanced results with and without using bilateral weights are shown in Fig 2.13. We can see that there are obvious halo artifacts around heads and shoulders when bilateral weights are not used, and such artifacts are suppressed when bilateral weights are used.

Let us take a closer look at the color histograms at three sample points, p_b (blue), p_g (green) and p_r (red), in the top images of Fig 2.13. When bilateral weights are not used (Fig 2.13(a)), the color histogram at p_g can be seen as a blended version of color histograms at p_b and p_r since p_g is close to a region boundary. However, p_g actually belongs to the 'sky'



(a) Histogram w/o bilateral weights (b) Histogram with bilateral weights

Fig. 2.13 Histograms of channel 'a' in the CIELab color space at three pixels ('blue', 'green' and 'red') in the top image are shown in the bottom with corresponding colors. Note that the histogram at pixel 'green' looks more similar to that of pixel 'blue' after bilateral weights have been incorporated.

region, and adjustments applied to p_g should be similar to those applied to p_b. After using bilateral weights, we can see that the color histogram at p_g looks more similar to the color histogram at p_b, and it is likely that similar adjustments will be applied to both of them.

2.5.4 Number of Training Images

In the previous section, we use a fixed number of training and testing images as in [81] for fair comparison. To further verify the learning ability of our deep network, here we compare the performance of models obtained with different numbers of training images. The Uniform dataset has 115 images, from which 25 images are randomly selected as testing images. Then, we conduct a series of experiments which randomly choose an increasing number (50, 60, 70, 80 and 90) of images from the remaining images as the training set. We repeat each experiment in this series three times each using a different subset of randomly chosen images for training. The average testing error of the three trials with respect to the number of training images is shown in Fig. 2.14, where we can see that our network can achieve better

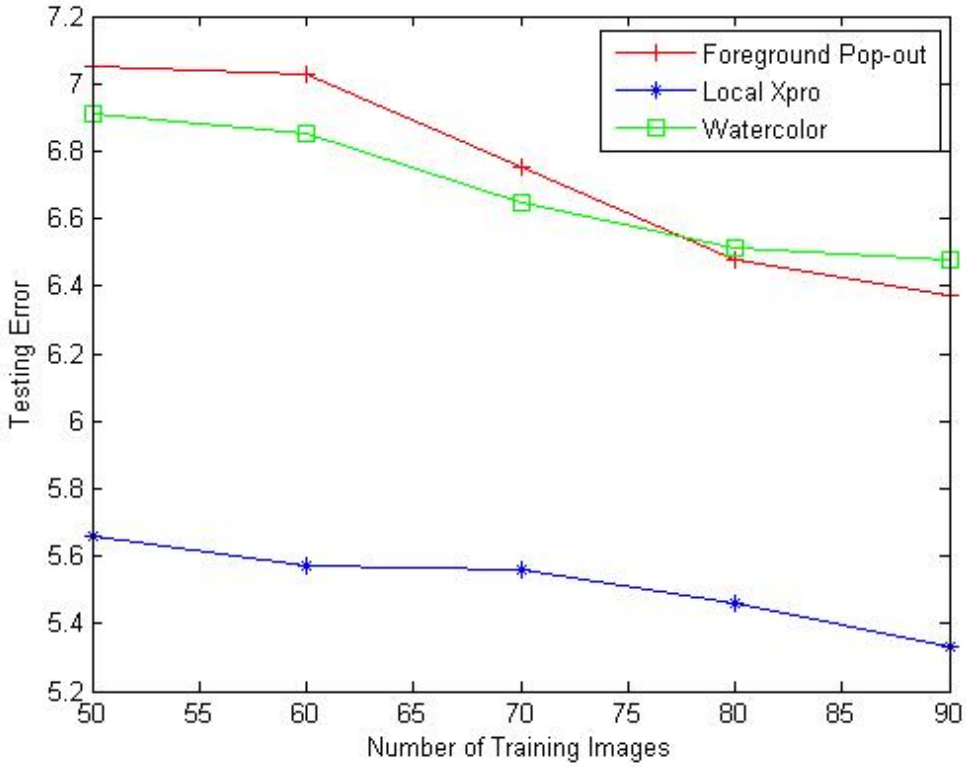


Fig. 2.14 Average testing error with respect to the number of training images.

performance with more training images. This is reasonable since larger training sets exhibit more content diversity and give rise to more accurate predictions of color transforms.

2.5.5 Global Styles

Our deep network can readily learn global image adjustment styles as well. We asked a photographer to create additional stylistic effects which are saved as “action" files in Photoshop, each of which contains a series of operations such as saturation adjustment, tone adjustment, and curve tuning. An “action" is globally applied to the original images in the Uniform dataset [81] to form a set of image exemplars for a global enhancement style. It is also convenient for us to obtain from the Internet various stylistic “action" files shared by photo retouching enthusiasts. Here we demonstrate model training and testing for two global effects as examples.

The first global effect (called “Spring") gives viewers the feeling of vitality, and the second global effect (called “Cold") tends to make a photo look vintage. Fig 2.15 shows enhancement examples in these two styles. The mean per-pixel L^2 distance between the

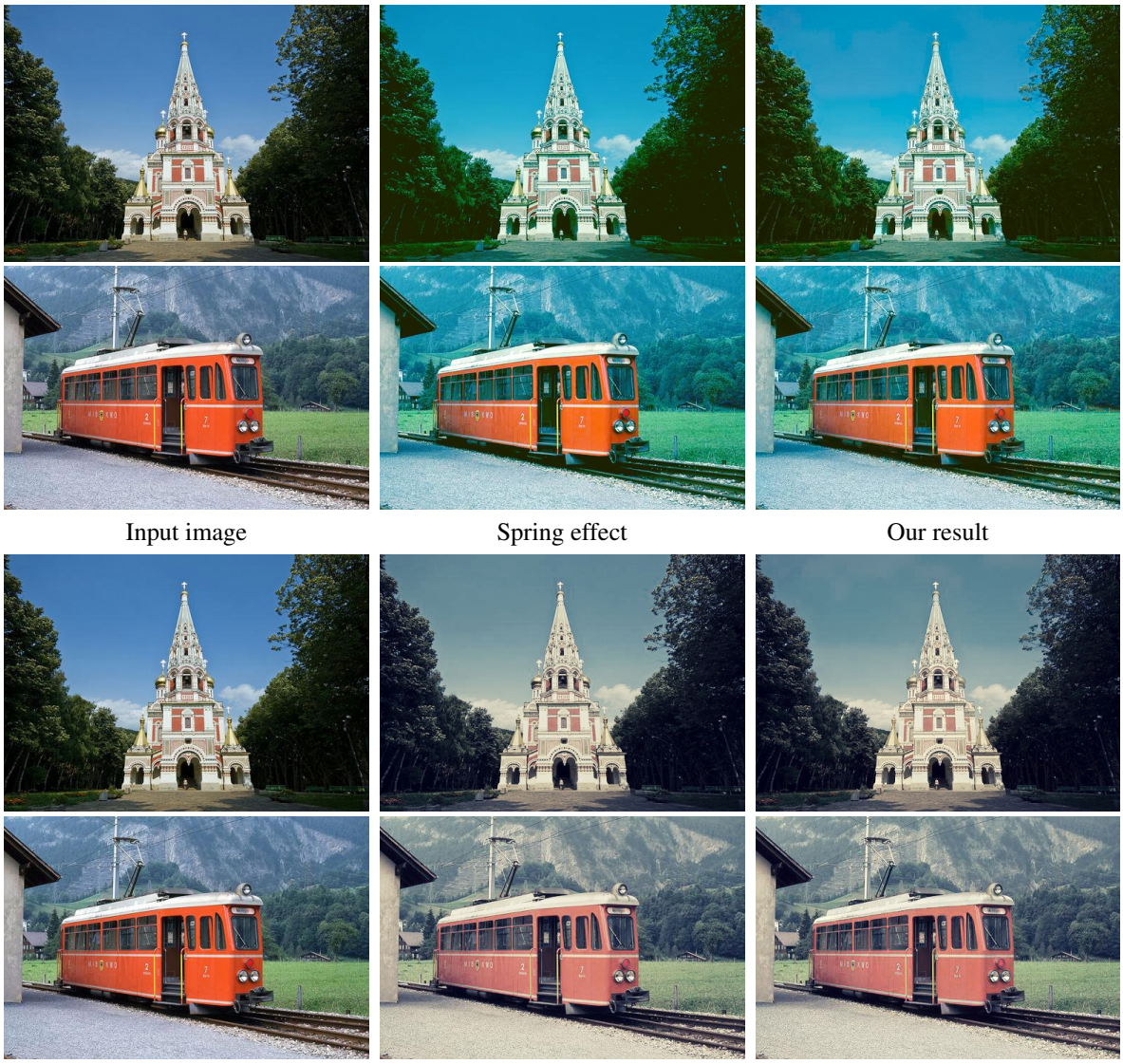


Fig. 2.15 Examples of global ‘Spring’ and ‘Cold’ styles. **Left**: Input image. **Middle**: Ground truth. **Right**: Our result.

original images and their groundtruth enhanced images reaches 16.95 and 18.56 for the “Cold" and “Spring" styles, respectively. This indicates that these two effects make significant color changes to the original images. The testing errors of our trained models for these two styles are 1.88 and 4.13 respectively, which numerically demonstrates the strength of our method in learning such global effects. Comparing the middle column and the right column in Fig 2.15, we can see that the enhanced results from our model look nearly the same as the ground truth.

2.6 Video Stylization

Our deep network trained on image exemplars can be extended to enhance artistic styles in videos as well. Naive video stylization in a frame-by-frame manner ignores the temporal coherence between adjacent frames, and is also inefficient. Therefore, we seek a more compact video representation to facilitate video stylization.

2.6.1 Temporal Superpixels

To generalize 2D superpixels used in image stylization for the purpose of video stylization, it is tempting to adopt the graph-based segmentation algorithm for decomposing a video into 3D supervoxels[75]. However, without explicitly modeling motions in a video, supervoxels cannot track image regions and object parts, and temporal coherence is hardly guaranteed. To overcome this difficulty, we represent a video as a set of temporal superpixels (TSPs), which are inferred from a probabilistic generative model explicitly considering motion flows between frames [12]. Moreover, TSPs are more uniform than supervoxels, which make them better suited for tracking image regions in a temporally coherent manner.

2.6.2 Frame Selection

A TSP can be projected onto video frames as a sequence of temporally adjacent 2D superpixels. As in image stylization, a single color transform is predicted for every TSP, and applied to all pixels within the TSP during video stylization. As a result, we only need to choose one 2D superpixel from the aforementioned sequence of projected superpixels. Our FCN can extract features at the centroid of this chosen superpixel and predict its color transform by processing the video frame containing this superpixel. The predicted color transform can then be used for enhancing all the pixels within the TSP.

On the basis of the above analysis, to reduce the computational cost, we seek to minimize the total number of video frames processed by the FCN while ensuring at least one of every TSP's projected 2D superpixels is included in the processed frames. We develop an iterative frame selection algorithm to solve the above problem in a greedy manner. At the beginning of our algorithm, all TSPs are unlabeled. During each iteration, we first choose the video frame intersecting the largest number of unlabeled TSPs, and then label these intersected TSPs. This process is repeated until all TSPs have been labeled. The resulting list of chosen frames will be processed by our FCN. In Section 2.7, we will demonstrate the advantage of our frame selection algorithm in the context of video stylization.

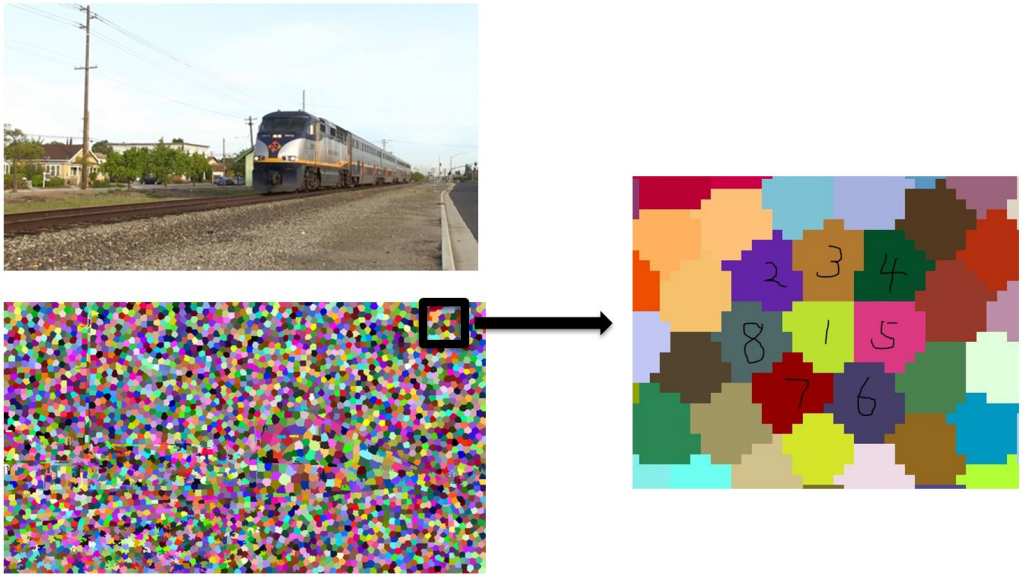


Fig. 2.16 Neighboring superpixels of a superpixel (labeled as '1') in a video frame. These neighboring superpixels are used in guided bilateral filtering of color transform coefficients within a single video frame.

2.6.3 Guided Spatial Smoothing

Within each TSP, we only extract features and predict a color transform for one of the projected 2D superpixels, and the resulting color transform is used for enhancing all pixels in the TSP. This strategy works well in terms of maintaining temporal coherence. Nonetheless, for two spatially adjacent TSPs, if their chosen 2D superpixels for feature extraction do not lie on the same video frame, the extracted features as well as the predicted color transforms of these superpixels may not have spatial coherence and blocky artifacts may appear in the enhanced video.

To address this problem, we apply guided bilateral filtering [61] to predicted color transform coefficients. As shown in Figure 2.16, to obtain smoothed color transform coefficients for a superpixel p_1 in a video frame f, we first identify its immediately neighboring superpixels p_i in the frame, Here, $i \in 2,3,4...8$. The smoothed color transform coefficients ϕ_1^f for superpixel p_1 in frame f are computed with the following equation, $\phi_1^f = \sum_i w_{1i}\phi_i$, where ϕ_i denotes the predicted color transform coefficients at the i-th neighboring TSP, $w_{1i} = \exp\frac{|c_i - c_1|^2}{\sigma_c}\exp\frac{|p_i - p_1|^2}{\sigma_p}$ represents the bilateral weight computed using the original input image as the guidance, and c_i, p_i are the color and position at the centroid of the i-th neighboring superpixel.

2.7 Experimental Results on Video Stylization

2.7.1 Datasets and Statistics

For testing our video stylization algorithm, we train image enhancement models for five styles using our proposed deep network. These five styles include three local styles (Local Xpro, Foreground Pop-out, and a new local style called “Golden") and two global styles (Cold and Spring). The local style **Golden** is created by applying distinct stylistic adjustments to three types of semantic regions, namely natural objects, man-made objects, and sky regions. Natural objects include trees, land, mountains, people, etc. while man-made objects include buildings, cars, etc. Unlike the **Local Xpro** effect, where different adjustment profiles share a common series of operations and only differ in adjustment parameters, the Golden effect has distinct adjustment operations for each type of semantic regions. For example, the operations for natural objects include channel mixer and color curve tuning while the operations for man-made objects include gradient mapping and brightness/contrast manipulation.

We apply each of five enhancement styles to a set of five testing videos named "Fly", "Mountain", "City", "Railway" and "Motor". Each groundtruth stylized video is produced by manually applying local profiles to individual video frames independently. The mean per-pixel L^2 distance in the CIELab color space between original frames and groundtruth enhanced frames over all videos for each style is 8.25, 4.74, 6.05, 6.33 and 6.87, respectively, as shown in Table 2.3. Once our trained models have been applied to the videos, the mean per-pixel L^2 distance between automatically enhanced frames and groundtruth enhanced frames drops to 3.19, 2.68, 1.60, 0.60 and 1.51, respectively. This numerically confirms the effectiveness of our overall video stylization approach based on models trained from images.

Effect	Groundtruth L^2 distance	Our Method	Frame-based Method
Local Xpro	8.25	3.19	3.23
Foreground Pop-out	4.74	2.68	2.67
Golden	6.05	1.60	1.61
Cold	6.33	0.60	0.56
Spring	6.87	1.51	1.35

Table 2.3 Comparison of mean per-pixel L^2 testing errors of our method and the frame-based method.

Fig 2.17 shows an example frame chosen from "City". We can see that our enhanced frames in the aforementioned five styles are visually similar to the groundtruth enhanced

One frame from video 'city'

Local Xpro groundtruth

Local Xpro enhanced

Pop-out groundtruth

Pop-out enhanced

Golden groundtruth

Golden enhanced

Fig. 2.17 (A).

Cold groundtruth | Cold enhanced

Spring groundtruth | Spring enhanced

Fig. 2.17 (B). Enhanced Local Xpro, Foreground Pop-out, Golden, Cold and Spring styles at a single frame from the "City" video.

frames. A real challenge in video stylization is maintaining temporal coherence, which is one of the strengths of our proposed video stylization algorithm.

2.7.2 Effectiveness of TSP-Based Video Stylization

The most straightforward way to enhance a video using our learned model simply considers the video frames as independent images and enhance these frames individually. This implies that superpixels within each frame are also generated independently. In this chapter, this approach is called the frame-based method. To demonstrate the effectiveness of our proposed video stylization algorithm, we compare our method against the frame-based method in terms of both numerical accuracy and visual quality. For numerical accuracy, we compute the mean per-pixel L^2 testing error between enhanced results and the ground truth for the five styles. As shown in the third and fourth columns of Table 2.3, the numerical accuracy of our method only differs slightly from that of the frame-based method. However, as expected, the frame-based method cannot preserve temporal coherence, and the enhanced videos from our method have significantly higher visual quality.

We also compare the computational cost between our method and the frame-based method. Taking the video "Fly" as an example which has 129 frames, the frame-based method would

need 3225 seconds since each frame takes 25 seconds. The processing time of our video stylization algorithm is significantly shorter because we do not have to extract features at all superpixels in all frames. The feature of a TSP is extracted only once in one of the selected frames and the computed quadratic color transform is shared among all pixels within the same TSP. In addition, we do not need to compute semantic feature maps for those unselected frames. The running times for video stylization are shown in Table 2.4.

Video	#frames	1k TSPs	3k TSPs	Runtime 1k TSPs	Runtime 3k TSPs
Fly	129	53(41%)	123(95%)	910s	1200s
City	155	58(37%)	114(74%)	1050s	1350s
Railway	120	82(68%)	98(82%)	990s	1110s
Motor	156	111(71%)	144(92%)	1340s	1500s
Mountain	202	112(55%)	180(89%)	1580s	1920s

Table 2.4 Comparison of the number and percentage (in parentheses) of chosen frames at two levels of TSP granularity. Runtime is shown in the last two columns.

The TSP algorithm [12] has parameters to control the approximate number of superpixels in each frame. To verify the effectiveness and flexibility of our frame selection scheme, we produce TSPs at two scales, Seg_1000 and Seg_3000, which indicates there are around 1000 and 3000 superpixels in each frame. Table 2.4 lists the number and percentage of frames chosen by our frame selection algorithm when a video is segmented at each of these two scales as well as the total number of frames in each video. The computational cost of our method decreases when the granularity of superpixels increases since there are less TSPs needed to extract features and less mapped frames needed to compute semantic feature map.

We also compare the mean per-pixel L^2 testing error between enhanced videos and the ground truth at these two scales. At 3000 superpixels per frame, the mean per-pixel L^2 error is 3.19, 2.68, 1.60, 0.60 and 1.51 for the Local Xpro, Foreground Popout, Golden, Cold and Spring styles, respectively. The mean per-pixel L^2 error only rises slightly to 3.27, 2.77, 1.65, 0.67 and 1.70 respectively at 1000 superpixels per frame. This comparison indicates that the numerical error will not increase significantly when we increase the granularity of superpixels within a certain range to reduce computational cost.

2.8 Conclusions and Discussion

In this chapter, we have presented a novel deep learning architecture for exemplar-based image and video stylization, which learns local enhancement styles from image pairs. Our deep learning architecture consists of fully convolutional networks for automatic semantics-aware feature extraction and fully connected neural layers for adjustment prediction. Image stylization can be efficiently accomplished with a single forward pass through our deep network. To extend our deep network from image stylization to videos, we exploit temporal superpixels to facilitate the transfer of artistic styles from image exemplars to videos. Experiments on a number of datasets for image stylization as well as a diverse set of video clips demonstrate the effectiveness of our deep model.

Input image | Local Laplacian [59] | Our result

Input image | Pop-out | Our result

Fig. 2.18 Two failure cases. **Top row:** Local Laplacian filter [59] is used to increase image details exaggeratedly. Our result produces insufficient detail increasement. **Bottom row:** One test result of Foreground Pop-out effect, which has color artifacts marked in the red box.

Limitations Our method has limitations. Detail manipulation is considered one type of image stylization. We use the local Laplacian filter [59] to exaggerate image details in the Uniform dataset. Such exaggerated results are used as training data for our method. The top row of Fig. 2.18 shows our trained model can only enhance details slightly but cannot achieve results similar to the ground truth. The reason is that our method only applies color transforms to individual pixels independently while detail manipulation needs to adjust local contrast among nearby pixels. The bottom row of Fig. 2.18 shows that if the predicted color transform at a certain superpixel (highlighted area) is incorrect, all the pixels within the same

superpixel will be adjusted incorrectly, which is another limitation of our superpixel-based method.

Future Work An interesting direction to extend this work of image and video stylization is to solve this problem using an end-to-end trainable network. That is we do not perform spatial subsampling using superpixels on the semantic feature maps produced by the fully convolutional networks. The fully connected layers used for predicting color transforms can be replaced with convolutional layers with 1×1 kernels. Such a network will be able to make pixel-level predictions. However, since it has a large number of parameters, overfitting can easily occur with insufficient training data.

Claim: the presented material in this chapter has been published in [89].

Chapter 3

A Benchmark for Edge-Preserving Image Smoothing

Edge-preserving image smoothing is a fundamental problem in image processing and low-level computer vision. At present, there are reasons that seriously hinder its further development. First, its performance evaluation remains subjective. Second, there does not exist widely accepted datasets for both evaluation and research. Third, most existing algorithms cannot perform well on a wide range of image contents using a single parameter setting. In this chapter, to remove the aforementioned hurdles in performance evaluation and further advance the state of the art, we propose a benchmark for edge-preserving image smoothing. This benchmark includes an image dataset with “groundtruth" image smoothing results as well as baseline learning algorithms that produce models capable of generating reasonable edge-preserving smoothing results for a wide range of image contents. Our image dataset contains 500 training and testing images with a large number of representative visual object categories. The baseline methods in our benchmark are existing representative deep convolutional network architectures, on top of which we design novel loss functions well suited for edge-preserving image smoothing. The trained deep networks run faster than most state-of-the-art smoothing algorithms while the smoothing performance of our ResNet-based model outperforms such algorithms both qualitatively and quantitatively.

3.1 Introduction

To effectively carry out certain image analysis and manipulation tasks, such as contour detection, image segmentation, and image stylization, it is important to preserve major image structures, such as salient edges and contours, while eliminating insignificant details. This is

the goal of edge-preserving image smoothing, a fundamental problem in image processing and low-level computer vision. Edge-preserving image smoothing has received a great deal of attention, and a number of algorithms with a diverse set of approaches have been proposed. Nevertheless, there exist three factors that seriously hinder the further development of edge-preserving image smoothing algorithms.

The first factor is that performance evaluation of edge-preserving smoothing algorithms remains subjective. Indeed, it is hard to objectively assess the quality of smoothing results. At present, the prevailing method is visual evaluation with verbal justification. User studies appear to be better. But a user study typically compares results from different algorithms instead of against the ground truth.

The second factor is that any new edge-preserving smoothing algorithm is typically evaluated on a very small image collection against existing algorithms. And even worse, there are no widely accepted small image collections for this purpose. A direct consequence is that while a smoothing algorithm produces impressive results on a chosen set of images, it is fairly easy to find images outside the chosen set on which the algorithm cannot perform well.

The third factor is that smoothing algorithms typically have tunable parameters and different categories of image contents might need different parameter settings. To the best of our knowledge, there do not exist smoothing algorithms that perform equally well on a wide range of image contents using a single parameter setting.

In this chapter, to remove the aforementioned hurdles in performance evaluation and further advance the state of the art, we propose a benchmark for edge-preserving image smoothing. This benchmark includes an image dataset with “groundtruth" image smoothing results as well as baseline learning algorithms that produce models capable of generating reasonable edge-preserving smoothing results for a wide range of image contents. Our image dataset contains 500 training and testing images with a large number of representative visual object categories, including humans, animals, plants, indoor scenes, landscapes and vehicles. For low-level computer vision problems, a dataset with hundreds of images is already quite large because an image in such a context is not just a single training sample but hundreds of thousands of training samples. The groundtruth smoothing results in our dataset were not directly annotated by humans, but manually chosen from results generated by existing state-of-the-art edge-preserving smoothing algorithms. This is justified by two rationales. First, as discussed earlier, a state-of-the-art algorithm is capable of producing high-quality smoothing results at least over a small range of image contents especially when its parameters have been fine-tuned. Therefore, collectively, such algorithms are able to generate high-quality results over a much wider range of contents. The only caveat is that the

Fig. 3.1 In our dataset, each source image is associated with 14 edge-preserving smoothing results selected by different subjects from various edge-preserving smoothing algorithms. We present 3 human-selected results for each source image here.

best results generated by these algorithms for a specific image need to be hand-picked by humans. Second, since an image has hundreds of thousands of pixels, directly annotating pixelwise smoothing results by humans is too labor-intensive and error-prone.

To set up the baseline algorithms in our benchmark, we seek assistance from the latest deep neural networks. Deep neural networks have a large number of parameters (weights). Nevertheless, once these weights have been training, they can be fixed and the resulting network has very strong capabilities in generalization and dealing with noisy inputs. That is, a well trained deep neural network with fixed weights can produce high-quality results for a wide range of inputs. Thus, it is not necessary to tune parameters for novel image contents any more, a goal we wish to achieve for edge-preserving image smoothing. We also note that deep learning has been broadly applied to low-level computer vision problems and has achieved state-of-the-art results in many such problems. Examples include reproducing edge-preserving filters [77, 49, 45], image denoising [56, 85], image super-resolution [37, 38, 68, 69, 42], and JPEG deblocking [19, 85]. Specifically, we use the following two existing representative network architectures as our baseline methods, very deep convolutional networks (VDCNN) and deep residual networks (ResNet). On top of these network architectures, we design novel loss functions well suited for edge-preserving image smoothing. The deep networks trained over the training split of our dataset run faster than most state-of-the-art edge-preserving smoothing algorithms while the smoothing

performance of our ResNet-based model still outperforms such algorithms both qualitatively and quantitatively. Our benchmark will be publicly released.

3.2 Related Work

Edge-Preserving Smoothing: There has been much work focused on edge-preserving smoothing. Representative algorithms include Bilateral Filter [71], Anisotropic Diffusion (AD) [60], Weighted Least Square smoothing (WLS) [22], Edge-avoiding Wavelet (EAW) [23], Rolling Guidance Filter (RGF) [86], SD filter [31], L_0 smoothing [76], Fast Global Smoother (FGS) [58], Tree Filtering [6], Weighted Median Filter (WMF) [87], L_1 smoothing [9] and Local Laplacian filter (LLF) [59]. These algorithms generally aim to avoid edge blurring, halo artifacts, gradient reversal, and global intensity shift.

Quantitative Evaluation: Bao *et al.* [6] applied different edge-preserving filters to the test images in Berkeley Segmentation Dataset (BSDS300) [57] prior to boundary detection. F-measure is used to evaluate the filters' effectiveness in suppressing trivial details and preserving edges. Ham *et al.* [30, 31] report ODS and OIS [3] using the gradient magnitudes of filtered images to measure the effectiveness of filters. However, these evaluation approaches may suffer from the deviation of detected boundaries. In contrast, we construct a dataset of source images and their associated "ground truth" (human-selected edge-preserving smoothed images). Objective evaluation can be performed by directly comparing the results of an edge-preserving filter against such "ground truth".

Deep Edge-aware Filter Learning: Xu *et al.* [77] propose a method to learn and reproduce individual edge-preserving filters. They use a convolutional neural network to predict smoothed image gradients and then run an expensive step to reconstruct the smoothed image itself. However, the β parameter in their reconstruction step is not fixed and varies with different filters. Liu *et al.* [49] propose a hybrid network by incorporating spatially varying recurrent neural networks (RNN) conditioned on the input image. A Deep CNN is used to learn the weight map of the RNN. Li *et al.* [45] propose a learning-based method to construct a CNN-based joint filter to transfer the structure of a guidance image to a target image for structure-texture separation. Fan *et al.* [21] exploits edge information by separating the image smoothing problem into two steps. The first sub-network is supervised to predict the edge map and the second sub-network reconstructed the target image by leveraging the predicted edge map.

In contrast, we do not aim to reproduce individual filters, but best results among a number of filters over a wide range of image contents. Furthermore, our deep neural networks directly learn pixelwise colors in the smoothed result instead of smoothed image gradients. Thus

an input image can be smoothed efficiently with a single forward pass through the network without the need of a gradient-based reconstruction step.

3.3 A Dataset for Edge-Preserving Smoothing

A major source of our image dataset is the database reported in [53]. This database was originally constructed for comparing image quality models, and has a large number of high-quality natural images. Another source of our dataset is the Berkeley Segmentation Dataset (BSDS500) [4] which has been widely used in the computer vision community. We

Fig. 3.2 Sample source images from our dataset for edge-preserving image smoothing.

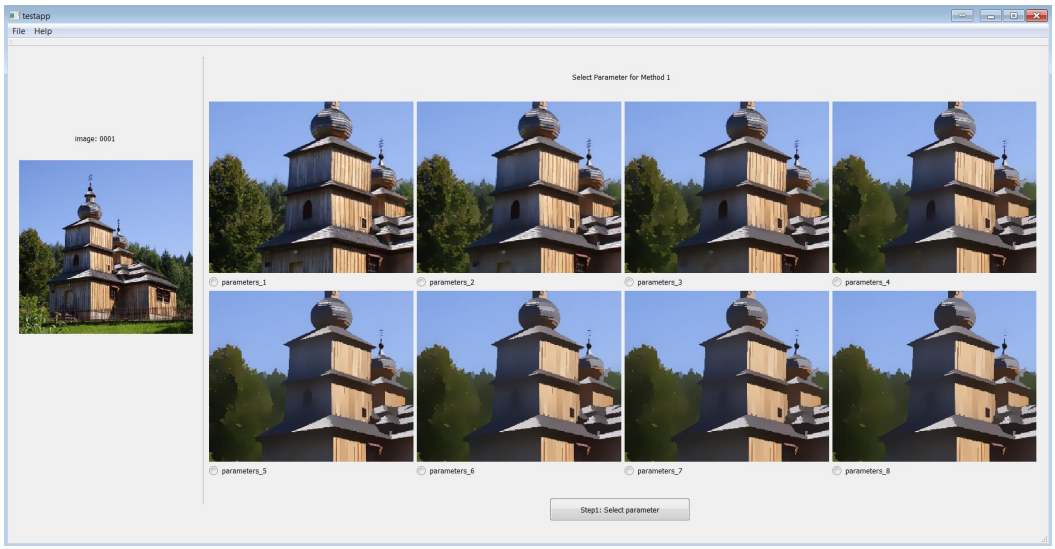
manually chose 500 images of a wide variety of objects and scenes (e.g. humans, animals, plants, indoors, landscape, vehicles, etc.). Some sample images are shown in Figure 3.2. We excluded a small number of images not suitable for edge-preserving smoothing (e.g. images without clear structures but filled with textured regions).

As mentioned earlier, the groundtruth smoothing results in our dataset were not directly annotated by humans, but manually chosen from results generated by existing state-of-the-art edge-preserving smoothing algorithms. This is because an image has hundreds of thousands of pixels, directly annotating pixelwise smoothing results by humans is too labor-intensive and error-prone. On the other hand, a state-of-the-art algorithm is capable of producing high-quality smoothing results at least over a small range of image contents especially with fine-tuned parameters. A collection of such algorithms are able to generate high-quality results over a much wider range of contents. However, since there does not exist any automatic mechanism to choose the best smoothing results from these algorithms, we still have to rely on humans to manually select the best results for every original image.

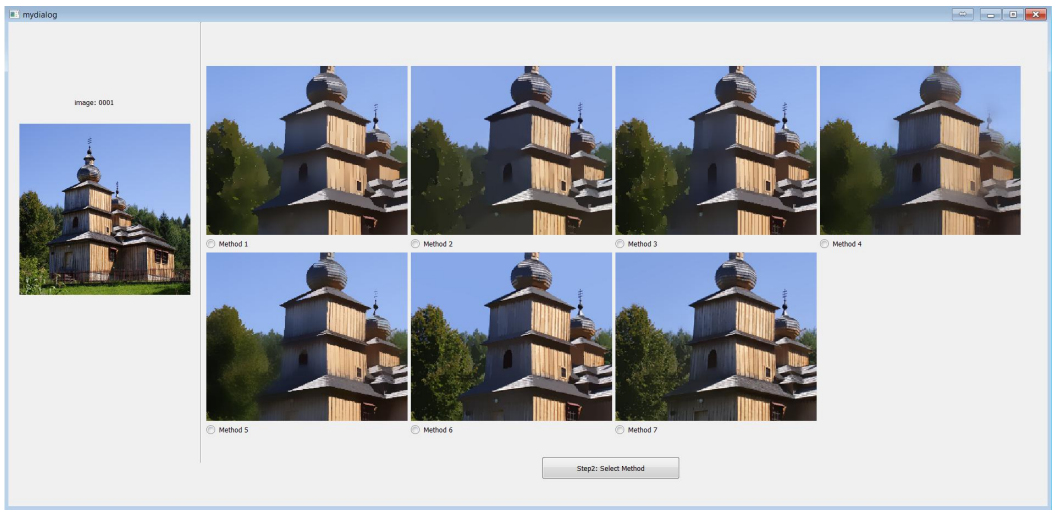
3.3.1 Selection Tool

We performed an initial screening and chose seven representative algorithms to form our collection of state-of-the-art edge-preserving filters, which includes SD filter [31], L_0 smoothing [76], Fast Global Smoother (FGS) [58], Tree Filtering [6], Weighted Median Filter (WMF) [87], L_1 smoothing [9] and Local Laplacian filter (LLF) [59].

According to the observation that the proper smoothing result of different images may come from different algorithms with different parameter settings, we have developed an interactive interface where one can choose the proper edge-preserving smoothing result in two steps. **Step 1: Given a source image, choose a parameter setting for each method so that the chosen setting gives rise to the best visual quality of the smoothed result. Step 2: Choose the best result from seven methods.** Figure 3.3 shows snapshots of the selection tool. The source image is shown in the top left corner. We pre-defined eight parameter settings for each algorithm, as shown in Table 3.1. Eight settings are believed to be sufficient for covering a wide range of coarse-to-fine smoothing levels. For each algorithm, the smoothing results corresponding to these eight parameter settings are presented on the same screen (Figure 3.3(a)). A user first chooses the best parameter setting for each method. Afterwards, the seven smoothing results chosen from the previous step, one for each of the seven algorithms, are shown side by side on a pop-out window (Figure 3.3(b)). The user chooses one result from the seven as the final selection for the current source image.



(a) Step 1: Choose a best parameter for each method



(b) Step 2: Choose the best result from 7 methods

Fig. 3.3 Snapshots of our two-step selection interface.

Since the above two steps heavily rely on visual comparisons and multiple images need to be shown on the same screen, we use a 32-inch Truecolor IPS monitor with a high resolution (3840×2160) for our result selection tool.

3.3.2 Selection Protocol

Since humans could have different perceptual comprehension of trivial details and major structures, they might select different smoothed images as their preferred edge-preserving smoothing results. To model such perceptual differences among human users, we formed

Parameters		1	2	3	4	5	6	7	8
SD filter	λ	1	5	15	30	50	70	90	110
L_0 smoothing	λ	0.005	0.01	0.02	0.03	0.04	0.05	0.06	0.08
FGS	σ_c	0.02	0.02	0.025	0.025	0.03	0.03	0.04	0.04
	λ	400	900	600	900	500	900	400	1200
Tree Filtering	σ	0.05	0.05	0.1	0.1	0.2	0.2	0.4	0.4
	σ_s	8	4	8	4	8	4	8	4
WMF	σ	10	30	50	70	90	110	130	150
L_1 smoothing	α	10	10	20	20	100	100	200	200
	θ	200	50	200	50	200	50	200	50
LLF	σ_r	0.1	0.1	0.2	0.2	0.4	0.4	0.6	0.6
	α	2	4	2	4	2	4	2	4

Table 3.1 We predefined 8 sets of parameters for each smoothing algorithm. Additional parameters are set to default values suggested by original authors.

a subject group of 26 users, all of whom are graduate students. Each source image in our dataset and its associated smoothing results were randomly assigned to 14 users. That is, one user on average needed to select proper smoothing images for 267 source images. Most of the users do not have prior experiences with edge-preserving image smoothing. Simple but key instructions were given to the users in order to familiarize them with the nature of edge-preserving smoothing.

• Instruction 1: Strong edges should be preserved and blurry effects at significant edges are extremely undesired.

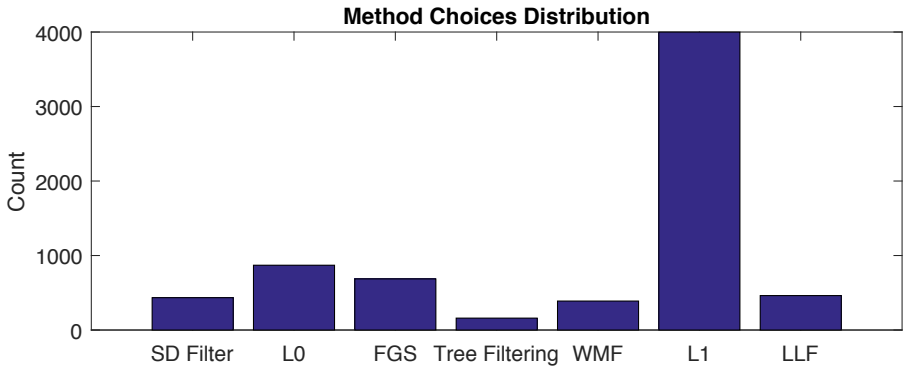
• Instruction 2: The color of a smoothed image should be close to the original image.

• Instruction 3: Under instructions 1 and 2, the smoother, the better.

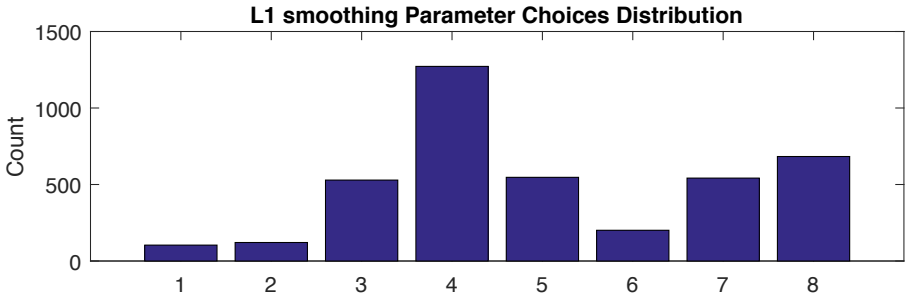
The users were further presented with a few unambiguous examples as intuitive instructions. It typically takes user around 2 minutes to select one final result. In order to minimize the negative effects of fatigue, only a single work session up to 60 minutes is allowed in any single day.

3.3.3 Dataset Statistics

Our dataset for edge-preserving smoothing contains 500 natural images (400 for training and 100 for testing). To reduce the bias of subject preference during manual selection, we



(a) Vote distribution among different algorithms



(b) Vote distribution among parameter settings of L_1 smoothing

Fig. 3.4 Distributions of user selection results. Two distributions of users' votes over different smoothing algorithms and over different parameter settings of one of the algorithms (L_1 smoothing).

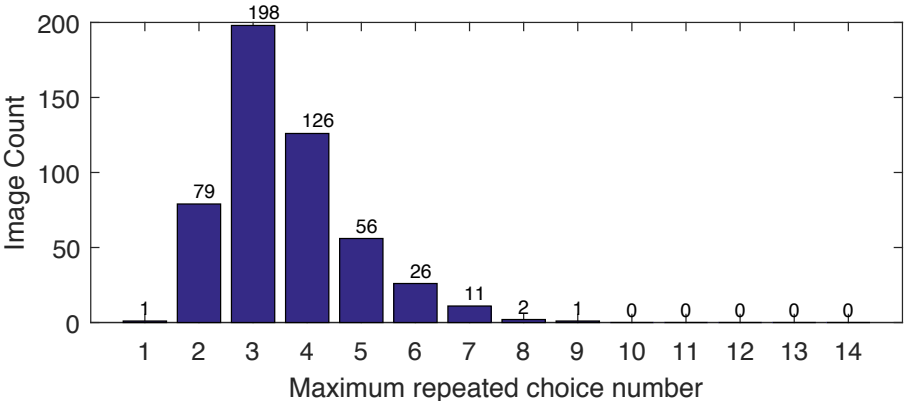


Fig. 3.5 Number of images vs. the maximum number of repeated choices.

collected 14 human-selected smoothing results for each image. The entire process lasted for one and half months.

Since each image is finally associated with 14 human-selected smoothing results, there are 7000 choices in total. As shown in Figure 3.4(a), a large proportion of the choices (3999

choices) are the L_1 smoothing algorithm. Nevertheless, there are still a considerable number of choices (3001) distributed among the other 6 smoothing algorithms. Figure 3.4(b) shows the distribution of the parameter choices for the L_1 smoothing method. This distribution confirms the previously mentioned observation that the proper smoothing result of different images may come from different algorithms with different parameter settings.

To show the consistency of choices of different users on the same source image, we compute the maximum number of repeated choices for each image. Let $count_t(m,p)$ denote the number of users who chose method m with parameter setting p to compute the proper smoothing result of the t-th source image. Note that $\sum_{m=1}^{7}\sum_{p=1}^{8} count_t(m,p) = 14$. The maximum number of repeated choices for image t is defined as $\max_m \max_p count_t(m,p)$. The distribution of the maximum number of repeated choices across all images is shown in Figure 3.5. We can see that there are 420 (out of 500) images whose maximum number of repeated choices is greater than or equal to 3. In another word, for 84% of the images, at least 3 users chose the result from the same algorithm with the same parameter.

3.4 Quantitative Measures

We let $x^t_{i,j}$ denote the pixel value at position (i,j) of the t-th source image in our dataset and $y^{t,k}_{i,j}$ denote the pixel value of the corresponding "groundtruth" smoothed image selected by the k-th user ($k \in [1,2,...,14]$) . We wish to measure the quality of an edge-preserving filter F on the basis of the Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). In our context, where we have multiple "groundtruth" smoothed images selected by different users, RMSE and MAE are defined as follows:

$$RMSE = \left(\frac{\sum_t \sum_{i,j} \sum_{k=1}^{14} \frac{1}{14} ||F(x^t)_{i,j} - y^{t,k}_{i,j}||^2}{\sum_t \sum_{i,j}} \right)^{\frac{1}{2}} \tag{3.1}$$

$$MAE = \frac{\sum_t \sum_{i,j} \sum_{k=1}^{14} \frac{1}{14} ||F(x^t)_{i,j} - y^{t,k}_{i,j}||_1}{\sum_t \sum_{i,j}} \tag{3.2}$$

where $F(x^t)_{i,j}$ is the pixel value in the smoothed image produced by the edge-preserving filter F. The denominator $\sum_t \sum_{i,j}$ denotes the total number of pixels in all images.

Due to the subjective nature of image quality assessment, inevitably there exist noises and outliers in the "groundtruth" smoothed images selected by different users. To prevent such noises and outliers from interfering with our performance evaluation, we take a voting strategy to focus on those smoothed images chosen by more users.

Here, we first introduce some notations. As mentioned in Section 3.3.3, $count_t(m,p)$ denotes the number of users who chose method m with parameter setting p as the proper smoothing result of the t-th source image. Each image is associated with 14 human-selected smoothing results, and there are 7000 choices in total. The total number of times method m with parameter setting p was chosen by any user is denoted by $\mathrm{COUNT}(m,p)$.

The voting strategy is that for each source image, we sort its repeated choice numbers ($count_t(m,p)$) in a descending order. If there is a tie between different combinations of method and parameter setting, we sort them according to $\mathrm{COUNT}(m,p)$. That is because the total number of times a method with one of its parameter settings were chosen is an indicator of the overall performance. We tend to choose the smoothed images produced by reliable methods when user preferences are the same. We only keep the first five results as "groundtruth" smoothed images. For example, if method m with parameter p were selected by most users for the t-th source image, we let $Y^{t,1}$ denote the smoothed image produced by method m and parameter p, and set $count(Y^{t,1}) = count_t(m,p)$. We set $Y^{t,2}$ as the second most frequently chosen smoothed image and so on. The above quantitative measures (Equation 3.1-3.2) become weighted RMSE (WRMSE) and weighted MAE (WMAE) as follows:

$$WRMSE = \left(\frac{\sum_t \sum_{i,j} \sum_{k=1}^{5} w_{t,k} ||F(x^t)_{i,j} - Y_{i,j}^{t,k}||^2}{\sum_t \sum_{i,j}} \right)^{\frac{1}{2}} \tag{3.3}$$

$$WMAE = \frac{\sum_t \sum_{i,j} \sum_{k=1}^{5} w_{t,k} ||F(x^t)_{i,j} - Y_{i,j}^{t,k}||_1}{\sum_t \sum_{i,j}} \tag{3.4}$$

where $w_{t,k}$ is defined as:

$$w_{t,k} = \frac{count(Y^{t,k})}{\sum_{k=1}^{5} count(Y^{t,k})} \tag{3.5}$$

In our quantitative measures, the weight of a combination of method and parameter setting varies across different source images because none of the existing smoothing algorithms performs equally well over a wide range of image contents. The proposed quantitative measures assign higher weights to "groundtruth" smoothed images chosen by more users while excluding noises and outliers at the same time. We use Equations 3.3 and 3.4 to quantitatively measure the performance of various edge-preserving smoothing algorithms as well as our trained models in the rest of this chapter.

Error	SD filter	L_0 smooth	FGS	TreeFilter	WMF	L_1 smooth	LLF	VDCNN	ResNet
WRMSE*	11.57	10.64	10.67	14.31	11.83	9.89	11.06	9.78	9.03
WMAE*	7.65	6.93	6.82	9.24	7.96	5.76	7.29	6.15	5.55

Table 3.2 The minimum WRMSE and WMAE of existing state-of-the-art edge-preserving smoothing methods and deep models. The optimal parameter setting of each algorithm is used across the entire dataset. Red, Green and Blue color indicates the best, second best and third best performance respectively.

3.4.1 Evaluation of Existing Algorithms

The parameter setting of a smoothing algorithm affects its performance. We measure WRMSE and WMAE across the entire testing set of our dataset for various different parameter settings for each existing algorithm. The minimum WRMSE and WMAE are denoted by WRMSE* and WMAE* respectively, and the optimal parameter setting of each method was determined by a greedy search. The process is as follows: set a group of parameter settings for each algorithm; apply them to all testing images; record WRMSE and WMAE. The parameter settings that make the seven chosen methods achieve their WRMSE* are given as follows: SD filter ($\lambda = 5$), L_0 smoothing (λ=0.02), FGS ($\sigma_c = 0.025, \lambda = 600$), Tree Filtering ($\sigma = 0.05, \sigma_s = 8$), WMF ($\sigma$=30), L_1 smoothing ($\alpha = 20, \theta = 50$), LLF ($\sigma_r = 0.4, \alpha = 2$). The parameter settings that make the seven methods achieve their WMAE* are given as follows: SD filter ($\lambda = 5$), L_0 smoothing (λ=0.01), FGS ($\sigma_c = 0.025, \lambda = 600$), Tree Filtering ($\sigma = 0.05, \sigma_s = 8$), WMF ($\sigma$=50), L_1 smoothing ($\alpha = 20, \theta = 50$), LLF ($\sigma_r = 0.2, \alpha = 4$). Additional parameters are set to default values suggested by original authors.

From Table 3.2, we can see that the L_1 smoothing algorithm has lower WRMSE* and WMAE* compared to other smoothing algorithms because the users chose results generated by the L_1 smoothing algorithm most frequently as their preferred results when our dataset was constructed, as shown in Figure 3.4.

3.5 Deep Learning Models

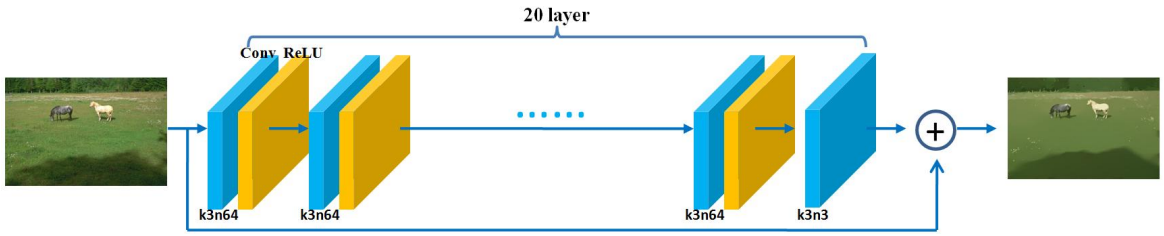
We used deep neural networks to set up learning-based baseline algorithms in our benchmark. Deep neural networks have a large number of parameters (weights). Nevertheless, a well trained deep neural network with fixed weights can produce high-quality results for a wide range of inputs. Thus, it is not necessary to tune parameters for novel image contents any more, a goal we wish to achieve for edge-preserving image smoothing.

Specifically, we seek representative deep neural network architectures for low-level computer vision tasks as our baseline methods because recently deep fully convolutional networks with high-resolution output have had many successes in low-level computer vision [56, 37, 38, 68, 69, 85] and such network architectures can typically be adapted for different low-level vision tasks.

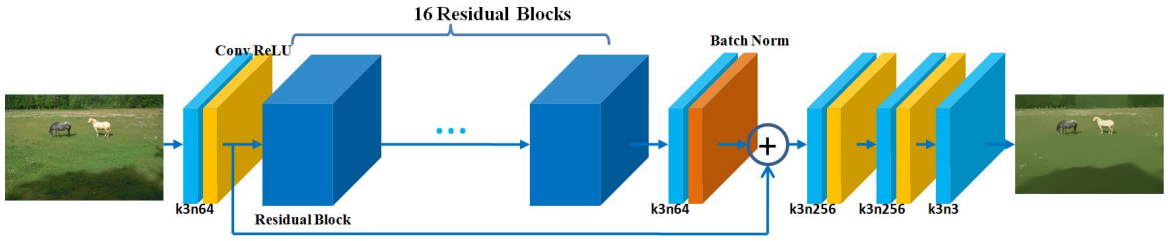
In this section, we present two representative network architectures and report their performance as a baseline for our edge-preserving smoothing dataset.

3.5.1 Network Architecture

VDCNN: 20 layers are stacked to form a very deep convolutional network, as shown in Figure 3.6(a). The layers maintain the same spatial resolution, the same kernel size (3×3) and the same number of feature maps (64) except the last output layer which has 3 channels



(a) VDCNN



(b) ResNet

Fig. 3.6 Network architecture of VDCNN and ResNet. Each convolutional layer is denoted with kernel size (k) and number of feature maps (n). The stride is 1 for all convolutional layers. Residual block is illustrated in Figure 3.7.

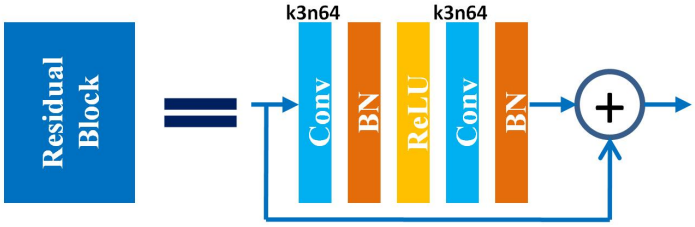


Fig. 3.7 Internal structure of a residual block used in ResNet.

only. The model originally proposed by Kim *et al.* [37] only learns the luminance channel which is the common practice in the single image super-resolution literature. We directly learn three RGB channels since all color channels change their values during edge-preserving smoothing.

ResNet: Residual networks [42, 47, 33] have exhibited outstanding performance in both low-level and high-level computer vision problems. As shown in Figure 3.7, a basic residual block includes convolutional layers (Conv), batch normalization (BN), rectified linear units (ReLU) and a skip connection. A complete ResNet architecture is shown in Figure 3.6(b). It was originally proposed in [42], where it is used for inferring photo-realistic high-resolution images from low-resolution ones. We replace ParametricReLU [32] with ReLU and remove the up-sampling layer here.

3.5.2 Loss Functions

Refer to the notations we use in Section 3.4. Since there exist multiple "groundtruth" smoothed images for each source image in our dataset, we define a weighted L_2 loss (Equation 3.6) and a weighted L_1 loss (Equation 3.7) in a manner similar to the weighted RMSE in Equation 3.3 and the weighted MAE in Equation 3.4.

$$loss_{l_2} = \sum_{t} \sum_{i,j} \sum_{k=1}^{5} w_{t,k} \left\| M_\theta(x^t)_{i,j} - Y_{i,j}^{t,k} \right\|^2, \tag{3.6}$$

$$loss_{l_1} = \sum_{t} \sum_{i,j} \sum_{k=1}^{5} w_{t,k} \left\| M_\theta(x^t)_{i,j} - Y_{i,j}^{t,k} \right\|_1, \tag{3.7}$$

where $M_\theta(x^t)$ represents the output from a deep network when x^t is given as the input image.

In addition to the weighted L_2 loss and weighted L_1 loss which enforce the consistency between predicted smoothed images and their corresponding groundtruth smoothed images,

Error	VDCNN			ResNet		
	$loss_{l_2}$	$loss_{l_1}$	$loss_{l_1} + loss_{nb}$	$loss_{l_2}$	$loss_{l_1}$	$loss_{l_1} + loss_{nb}$
RMSE	10.14	9.90	9.78	9.58	9.51	9.03
MAE	6.92	6.20	6.15	6.50	6.12	5.55

Table 3.3 Performance comparison between two network architectures under different loss functions.

Fig. 3.8 Example of smoothed output using different losses. We can see that the VDCNN trained with $loss_{l_1} + loss_{nb}$ produces smoother result at sky, roof and grass regions than $loss_{l_2}$ alone or $loss_{l_1}$ alone.

we propose a neighborhood loss (Equation 3.8) to explicitly encourage a network to favor the gradient of groundtruth images.

$$loss_{nb} = \sum_{t} \sum_{i,j} \sum_{k=1}^{5} \sum_{(p,q)\in N_{i,j}} w_{t,k} \left\| (M_\theta(x^t)_{i,j} - M_\theta(x^t)_{p,q}) - (Y^{t,k}_{i,j} - Y^{t,k}_{p,q}) \right\|_1, \tag{3.8}$$

where $N_{i,j}$ denotes the 5×5 neightborhood centered at pixel (i, j). The neighborhood term $||(M_\theta(x^t)_{i,j} - M_\theta(x^t)_{p,q}) - (Y^{t,k}_{i,j} - Y^{t,k}_{p,q})||_1$ explicitly penalizes deviations in the gradient domain. We add this neighborhood term to the weighted L_1 loss since edge-preserving smoothing involves evident gradient changes.

Quantitative results are presented in Table 3.3, where we can see that ResNet achieves better performance than VDCNN when the same loss is used. $loss_{l_1} + loss_{nb}$ achieves better performance than $loss_{l_1}$ alone or $loss_{l_2}$ alone when the same network architecture is used, which validates the effectiveness of $loss_{nb}$. Figure 3.8 shows an example where we can see that the VDCNN trained with $loss_{l_1} + loss_{nb}$ produces smoother result at sky, roof and grass regions than $loss_{l_2}$ alone or $loss_{l_1}$ alone.

We take the VDCNN model and ResNet model trained with $loss_{l_1} + loss_{nb}$ as our baseline algorithms for our edge-preserving smoothing benchmark.

3.5.3 Network Training

We augment the training data with horizontal flips. RGB training patches are randomly sampled from source images and the corresponding smoothed images. We set different training patch size and mini-batch size for VDCNN and ResNet since they have different receptive fields and model complexity. For VDCNN, patch size is set to 41×41, and mini-batch size is set to 64. For ResNet, patch size is set to 96×96, and mini-batch size is set to 16.

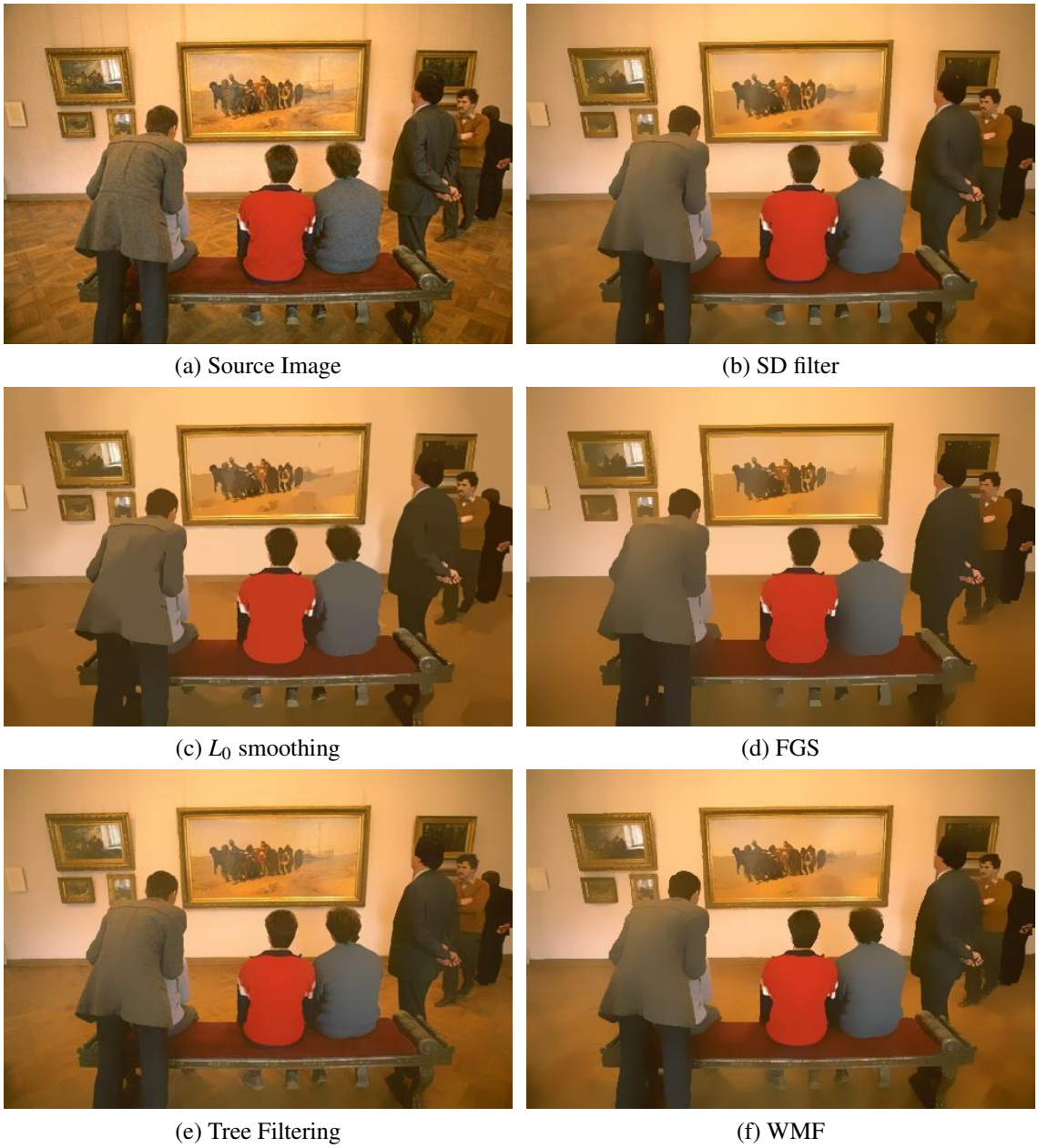
Our deep models are trained using the ADAM optimizer [39] with $\beta_1 = 0.9, \beta_2 = 0.999, \varepsilon = 10^{-8}$. The initial learning rate is set to 10^{-3}. After the initial model converges, the learning rate is decreased by a factor of 10. Training is terminated once the model converges again.

We have implemented the VDCNN and ResNet models in the Tensorflow framework and trained them using NVIDIA GeForce GTX 1080TI GPU. It takes one day to train VDCNN and two days to train ResNet respectively.

3.5.4 Evaluation

VDCNN model and ResNet model trained with $loss_{l_1} + loss_{nb}$ are taken as the baseline algorithms for our edge-preserving smoothing benchmark. In this section, we report their performance both quantitatively and qualitatively.

As shown in Table 3.2, the ResNet model achieves the lowest WRMSE and WMAE when compared to existing smoothing algorithms. The VDCNN model also achieves favorable



(a) Source Image (b) SD filter

(c) L_0 smoothing (d) FGS

(e) Tree Filtering (f) WMF

Fig. 3.9 (A)

(g) L_1 smoothing (h) LLF

(i) VDCNN (j) ResNet

Fig. 3.9 (B) Comparison of edge-preserving smoothing results of existing state-of-the-art algorithms and deep models. (a)Source Image. (b-h) The parameters are set as the optimal parameters for WMAE* illustrated in Section 3.4.1. (i)VDCNN with $loss_{l_1} + loss_{nb}$. (j) ResNet with $loss_{l_1} + loss_{nb}$.

performance. An example is presented in Figure 3.9. It can be seen that some algorithms, such as SD filter, L_0 smoothing, Tree Filtering and LLF, can't effectively remove inessential details at such floor regions. Some algorithms, such as FGS and WMF, blur the area around the foot. L_1 smoothing and the deep models achieve overall higher quality of edge-preserving smoothing than other algorithms. Existing smoothing algorithms use the optimal parameter setting for achieving WMAE*, as discussed in Section 3.4.1.

A more detailed visual comparison is conducted against the L_1 smoothing algorithm from [9] in Figure 3.10 because the users chose the L_1 smoothing algorithm most frequently and the L_1 smoothing algorithm achieves the lowest WRMSE and WMAE among existing state-of-the-art algorithms. From Figure 3.10 we can see that the L_1 smoothing algorithm incorrectly increases the color contrast between two flattened regions on the airplane. The results from VDCNN and ResNet model do not have such artifacts.

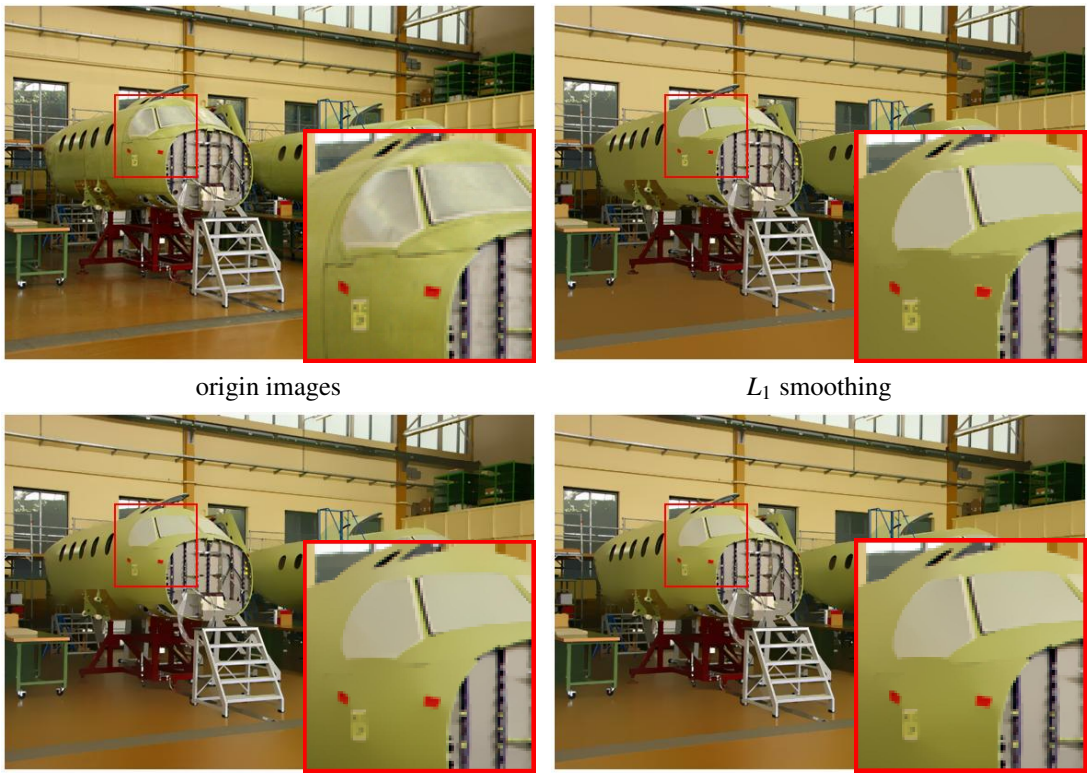
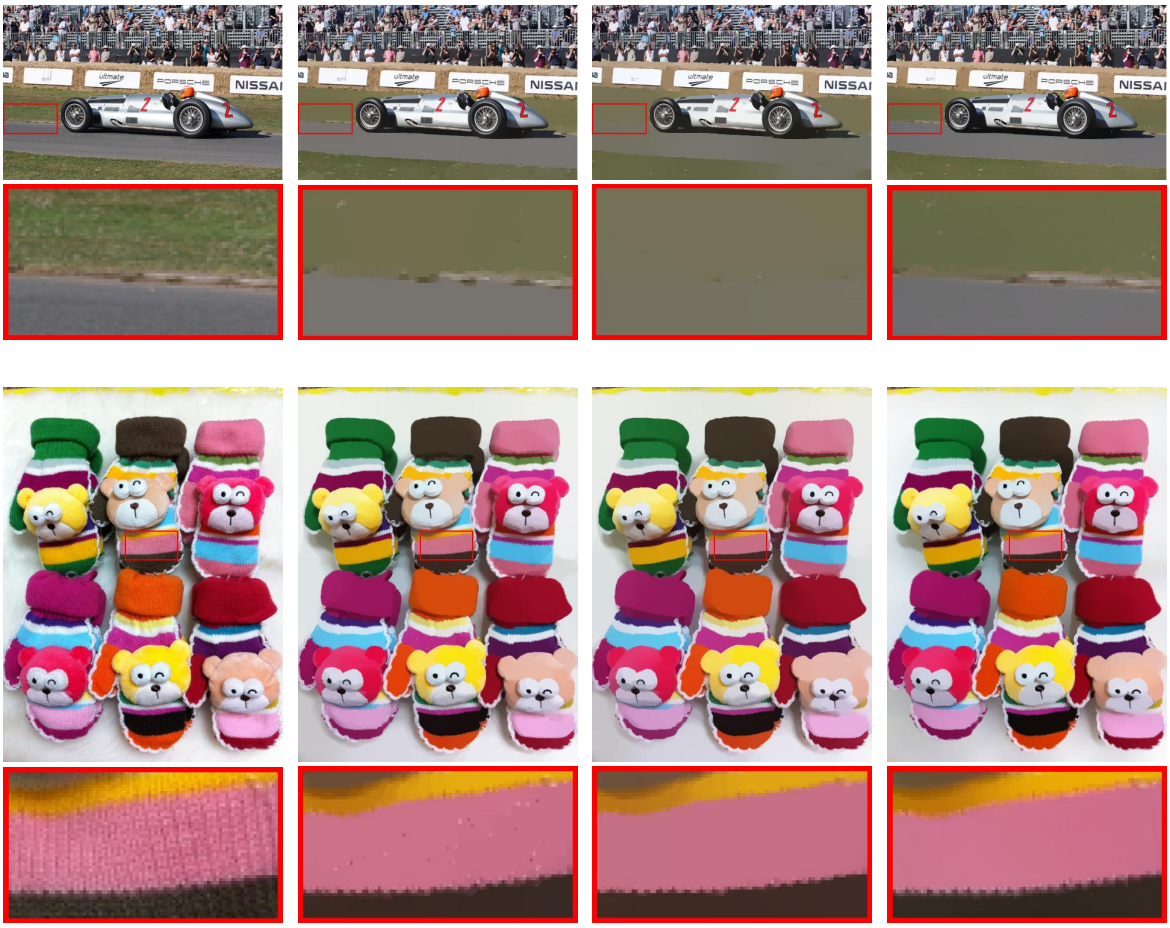


Fig. 3.10 Comparison between L_1 smoothing algorithm and deep models.

As mentioned earlier, we do not aim to reproduce individual filters like [77, 49, 45, 21]. Our image dataset contains 500 training and testing images with a large number of representative visual objects. By utilizing the dataset, our baseline algorithm aims to train a deep CNN model that produces reasonable edge-preserving smoothing results for a wide range of image contents without the need of finely tuning parameters. To the best of our knowledge, there do not exist smoothing algorithms that perform equally well on a wide range of image contents using a single parameter setting. A comparison between L_0 smoothing [76] and our ResNet model is shown in Figure 3.11. We can see that L_0 smoothing algorithm needs to set different parameters for the 'Racing car' and the 'Gloves' image. If we set λ=0.03 for the 'Racing Car' image, the edge between grass and road will blur. λ=0.01 is the proper setting for the 'Racing Car' image. However, if we set λ=0.01 for the 'Gloves' image, there still remains undesirable noises. λ=0.03 is the proper setting for the 'Gloves' image. In contrast, our ResNet model produces robust visual results on a wide range of image contents without tuning parameter.



origin images | L_0 smoothing(λ=0.01) | L_0 smoothing(λ=0.03) | ResNet

Fig. 3.11 Comparison between L_0 smoothing algorithm and our ResNet model. L_0 smoothing algorithm needs different parameter setting for the 'Racing car' and the 'Gloves' image. If we set λ=0.03 for the 'Racing car' image, the edge between grass and road will blur. λ=0.01 is the proper setting. However, if we set λ=0.01 for the 'Gloves' image, there still remains undesirable noises. λ=0.03 is the proper setting. In contrast, our ResNet model produces robust visual results on a wide range of image contents without tuning parameter.

3.5.5 Run Time

In addition to visual quality, testing speed is also an important aspect for image smoothing method. We report the running times of existing smoothing algorithms using the author-provided Matlab code on a 3.4GHz Intel i7 processor. The average running time over 100 testing images is shown in Table 3.4. The L_1 smoothing algorithm has lower WRMSE* and WMAE* than other smoothing algorithms, but spends hundreds of seconds on solving a series of large sparse linear systems. The slow processing speed of L_1 smoothing algorithm will prevent it from being an ideal pre-processing tool for other image processing applications,

e.g., edge detection. In contrast, our ResNet-based model achieves the lowest WRMSE and WMAE while its GPU implementation runs faster than most existing state-of-the-art smoothing algorithms.

Method	SD filter	L_0 smoothing	FGS	Tree Filtering	WMF
Run time (second)	10.46	1.24	0.05	0.18	0.52
Method	L_1 smoothing	LLF	Our VDCNN	Our ResNet	
Run time (second)	328	199	0.41	0.78	

Table 3.4 Run time (second) of existing state-of-the-art edge-preserving smoothing algorithms and our deep models.

3.6 Application

The smoothing of high-contrast details while preserving edges have been shown useful in many applications [20, 46]. In this section, we briefly review several applications by applying the trained ResNet model as the edge-preserving smoothing filter.

3.6.1 Tone mapping

Tone mapping is a popular technique to map one set of colors to another to reproduce the appearance of a high dynamic range (HDR) image on a low dynamic range (LDR) displayer which has a limited dynamic range that is inadequate to produce the full range of luminance values. The state-of-the-art tone mappers commonly adopt layer decomposition scheme where the HDR image is decomposed into low- and high-frequency layers and then processed separately. In particular, the low frequency layer is estimated by applying an edge-preserving filter to the original HDR image. The edge-preserving property is most important for avoiding halo artifact and achieving naturalness in the tone-mapped image, which, however, is not stably addressed by most tone mappers.Thus, a stable and effective edge-preserving filter is highly desirable to improve the tone mapping performance.

To stably avoid halo artifact, the requirement for an edge-preserving filter is to preserve the strong edge regions and flatten other regions in the image, regardless of the image contents and types. Our ResNet baseline model is able to handle this task, because it is trained with our dataset which is constructed with such criteria. We use the tone mapping framework in [20] where the original bilateral filter is replaced by our ResNet model. We compare the tone mapped results with several state-of-the-art tone mappers, including bilateral filter method

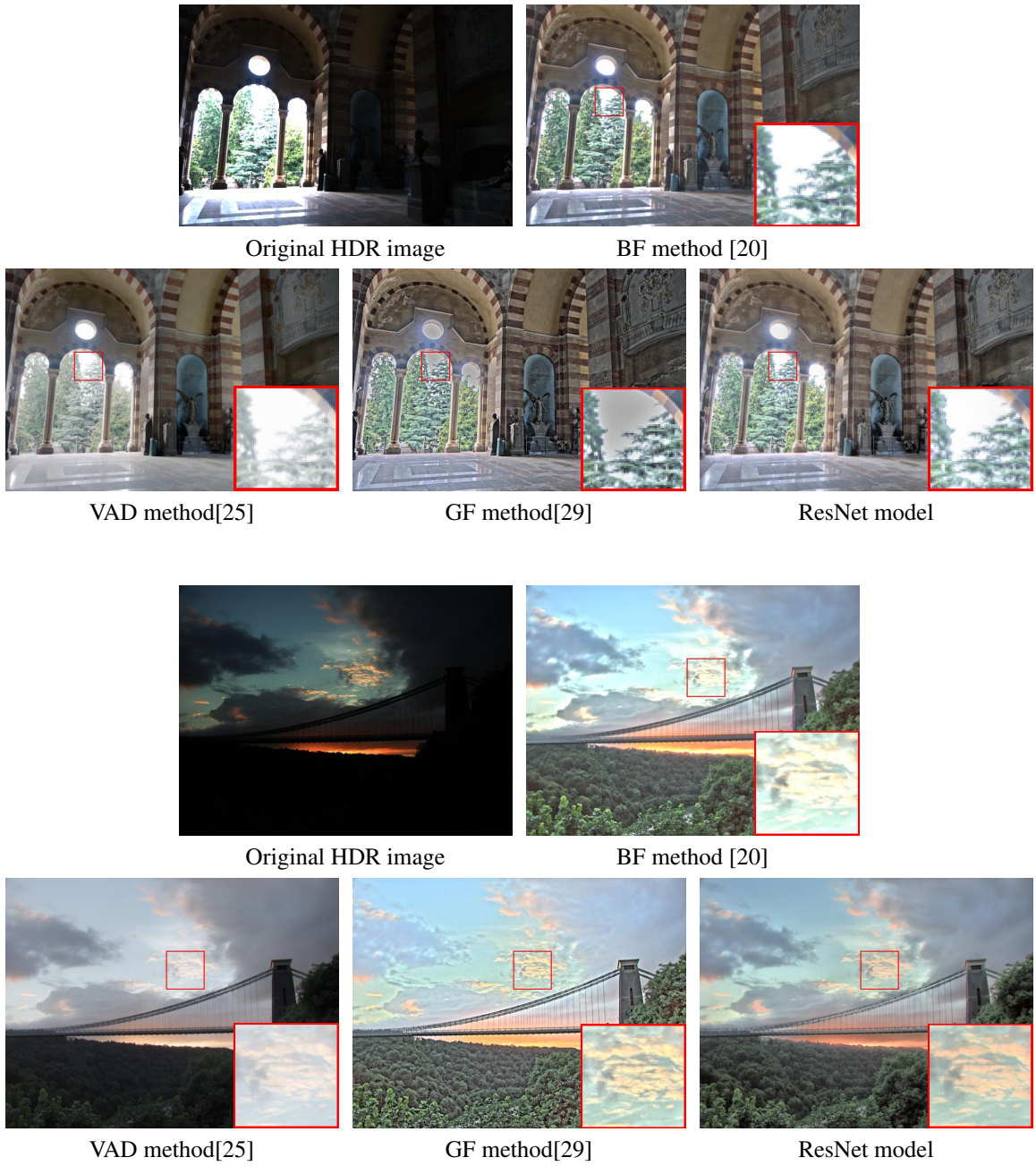


Fig. 3.12 Comparison of tone mapping results. We can see that the results produced by VAD method still miss many details. The lower image of GF method is over-enhanced and seems unnatural. In contrast, our results preserve the details everywhere and look natural and clean. An objective evaluation is shown in Table 3.5.

(BF) [20], Visual adaptation method (VAD) [25], and Guided filter method (GF) [29]. Fig .3.12 shows our tone mapping results compared with these tone mappers. We can see that our

	BF	VAD	GF	Ours
TMQI	0.9020	0.9041	0.8750	**0.9095**

Table 3.5 Comparison of TMQI scores. Tone Mapped image Quality Index (TMQI) [82] measures the structural fidelity and statistical naturalness.

tone mapper with ResNet model reaches a excellent balance of halo removal and naturalness preservation. Other tone mappers suffers from either halo artifact or over-enhancement.

To investigate the consistency of the competitive tone mappers, we collect 100 HDR images online for objective evaluation. Specifically, the TMQI metric [82] is used to score each tone mapped image by each method. Table 3.5 shows the average TMQI score of each tone mapper. We can see that our tone mapper with ResNet model achieves the highest TMQI score. This indicates that our edge-preserving benchmark can facilitate the tone mapper to gain consistently high performance, regardless of image type and content.

3.6.2 Contrast enhancement

Contrast enhancement aims to enhance the local contrast of an image that suffers from large illumination variation. Similar with the tone mapping algorithm, the contrast enhancement framework decomposes an image into two components, illumination and reflectance. The estimation of illumination also requires a high-performing edge-preserving filter. In [46], the

Original image WV method [26] ND method[46] ResNet model

Fig. 3.13 Comparison of contrast enhancement results for low-light images.

	WV[26]	ND[46]	Ours
IEM	1.59	2.11	2.18

Table 3.6 Comparison of IEM. Image Enhancement Metric (IEM) [34] measures the improvement in contrast of enhanced images.

author proposed a criterion for optimal contrast enhancement, that the edge-preserving filter should preserve the boundary regions of an image and should flatten the texture regions as much as possible. This coincides with the criterion developing our benchmark.

Now we construct a contrast enhancement to test our benchmark in this application. We adopt the algorithm framework in [46] and only replace the Diffusion-based filtering component with our ResNet model. Fig .3.13 shows the enhancement results of our algorithm compared with other methods, including nonlinear diffusion method (ND) [46] and weighted variational method (WV) [26]. We can see that our enhancement results contains clearer structures and higher local contrast.

To measure the contrast enhancement effectiveness, we here report the Image Enhancement Metric (IEM) [34], a full reference IQA metric, to assess the contrast of the enhanced images. We used the 18 test images denoted as 'a' to 'r' [84] for evaluation. Fig. 3.13 have shown some results. As Table 3.6 shows, the improved performance validates that the ResNet model trained on our benchmark can be adopted as an efficient edge-preserving smoothing filter.

3.7 Conclusions

We have presented a benchmark for edge-preserving image smoothing for the purpose of quantitative performance evaluation and further advancing the state of the art. This benchmark includes an image dataset with "groundtruth" image smoothing results as well as baseline learning algorithms. The baseline algorithms are representative deep convolutional network architectures, on top of which we design novel loss functions well suited for edge-preserving image smoothing. Our trained deep networks run fast at test time while the smoothing performance of our ResNet-based model outperforms state-of-the-art smoothing algorithms both quantitatively and qualitatively.

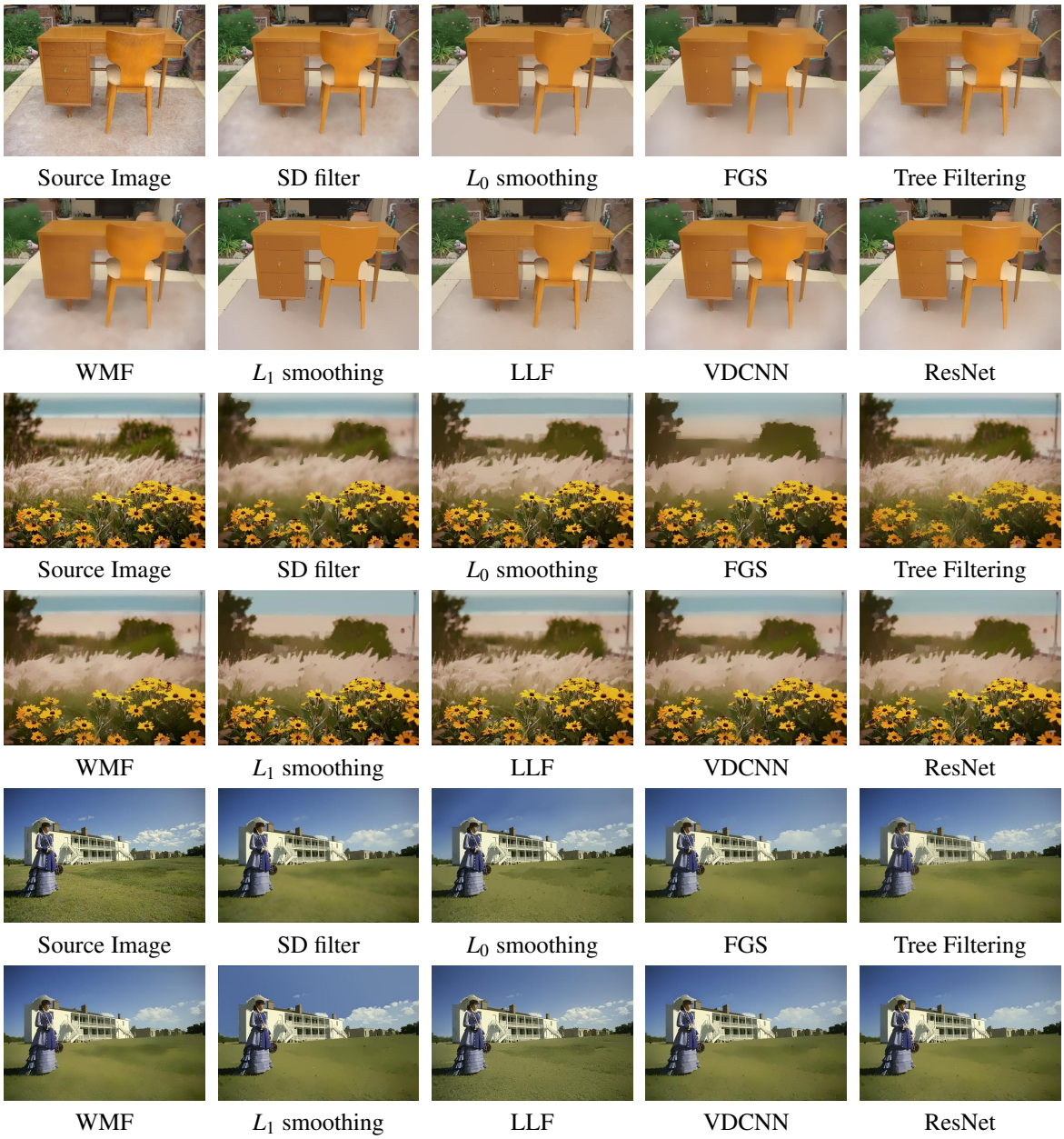


Fig. 3.14 (A). Comparison of edge-preserving smoothing results of existing state-of-the-art algorithms and deep models. (a)Source Image. (b-h) The parameters are set as the optimal parameters for WMAE*. (i)VDCNN with $loss_{l_1} + loss_{nb}$. (j) ResNet with $loss_{l_1} + loss_{nb}$.

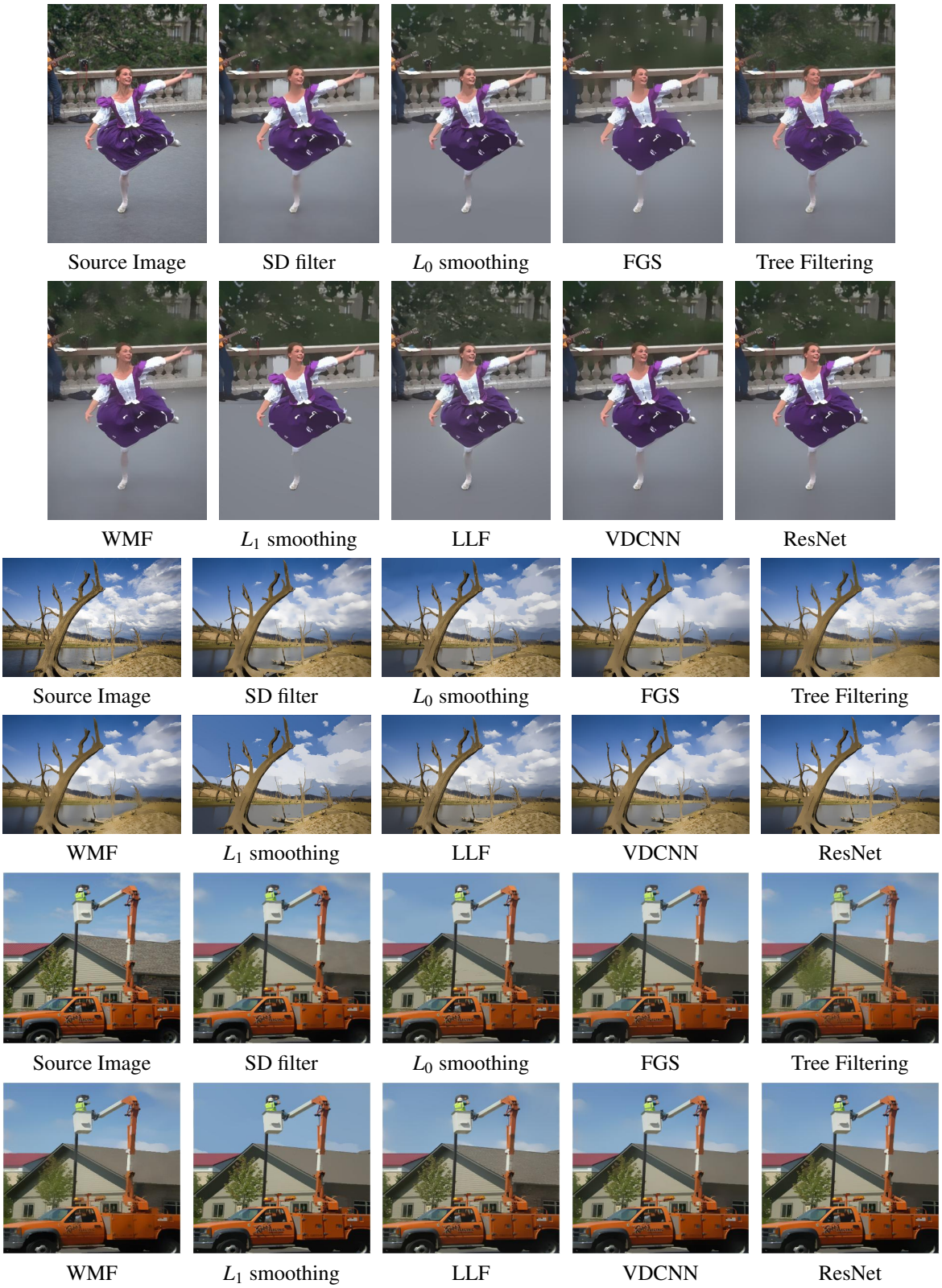


Fig. 3.14 (B). Comparison of edge-preserving smoothing results of existing state-of-the-art algorithms and deep models. (a)Source Image. (b-h) The parameters are set as the optimal parameters for WMAE*. (i)VDCNN with $loss_{l_1} + loss_{nb}$. (j) ResNet with $loss_{l_1} + loss_{nb}$.

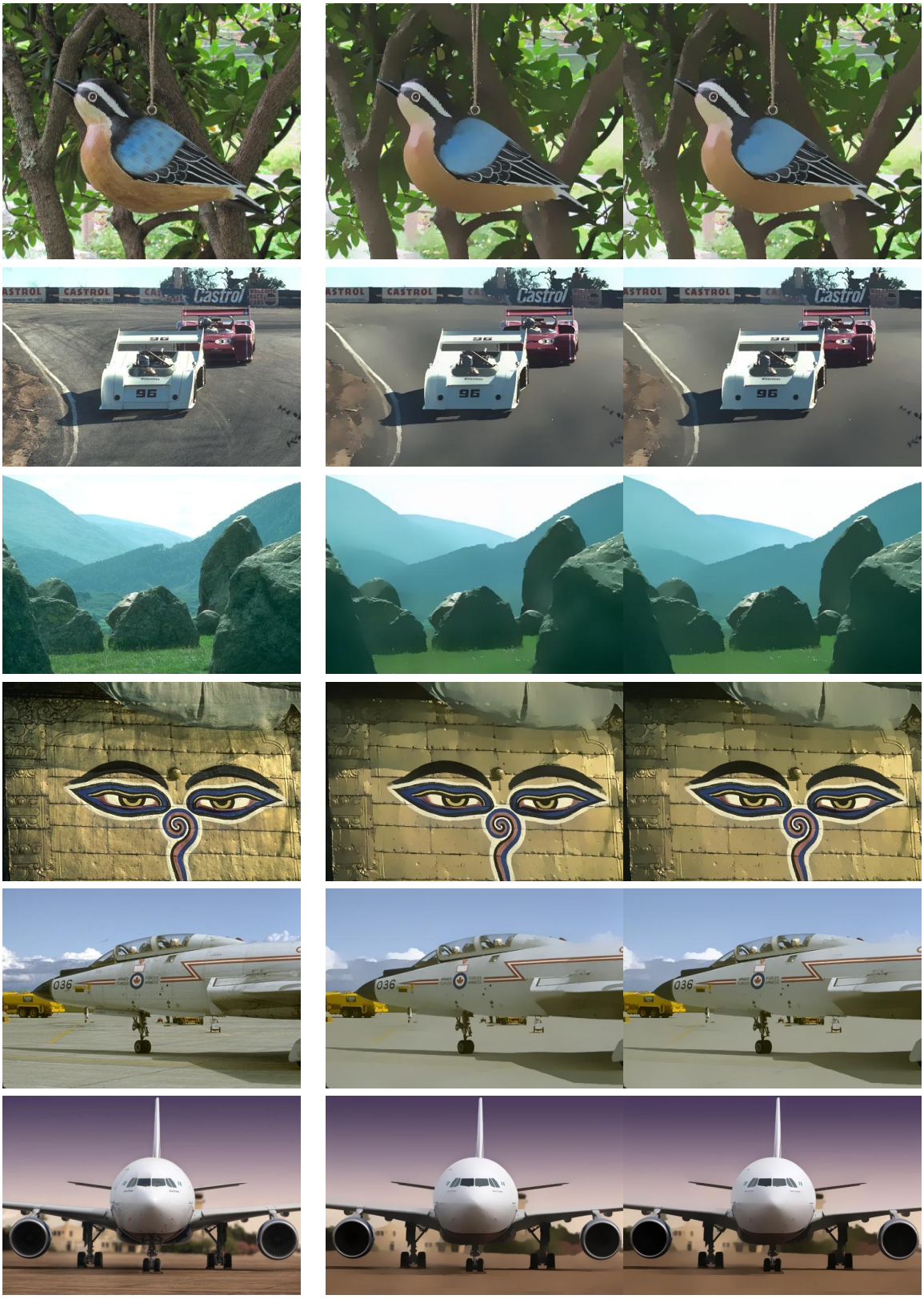


Fig. 3.15 More results from our VDCNN-based and ResNet-based models. **Left column:** Source image. **Middle column:** Results from our VDCNN-based model. **Right column:** Results from our ResNet-based model.

Chapter 4

Conclusions and Future Research

4.1 Principal Contributions

In this thesis, we focus on the problems with respect to image analysis and enhancement. Specifically, we propose a novel deep learning architecture for exemplar-based image and video stylization as well as a benchmark for edge-preserving image smoothing. The contributions of the work are as follows:

For image and video stylization:

- We propose a novel deep learning architecture for stylistic image enhancement. It consists of fully convolutional networks and fully connected neural layers. Our deep network is capable of learning distinct enhancement styles from a small set of training exemplars. Enhancing a novel image only requires a single forward pass through our network.
- Fully convolutional networks in our architecture extracts global features and contextual features. Our novel contextual features have two parts. The first part is a semantics-aware feature extracted from deep layers of a fully convolutional network; the second part consists of a set of color histograms over a small spatial grid.
- We demonstrate that deep neural networks trained with image exemplars can be used to enhance videos as well. We segment a video into temporal superpixels, and apply both temporally coherent and spatially smooth adjustments to them. A greedy frame selection algorithm is developed to reduce the computational cost of feature extraction.

For edge-preserving image smoothing:

- We construct a benchmark for edge-preserving image smoothing for the purpose of quantitative performance evaluation and further advancing the state of the art.

The image dataset contains 500 training and testing images of a large number of representative visual object categories with "groundtruth" image smoothing results selected by a group of users.

- The baseline algorithms are representative deep convolutional network architectures, on top of which we design novel loss functions well suited for edge-preserving image smoothing. Our trained ResNet model run fast at test time and can produce high-quality results on a wide range of image contents, outperforming existing state-of-the-art smoothing algorithms both quantitatively and qualitatively.

4.2 Future Research

There are many interesting research directions which are worth exploring in future:

- For image and video stylization, an interesting direction to extend this work is to solve the problem using an end-to-end trainable network. That is we do not perform spatial subsampling using superpixels on the semantic feature maps produced by the fully convolutional networks. The fully connected layers used for predicting color transforms can be replaced with convolutional layers with 1×1 kernels. Such a network will be able to make pixel-level predictions. However, since it has a large number of parameters, overfitting can easily occur with insufficient training data.
- For image smoothing, we construct the dataset "groundtruth" images by manually selecting the best result from various smoothing algorithms with different parameter settings. A potential more delicate way is to select groundtruth result patch by patch rather than image by image. That is, different regions of a single image can be smoothed by different smoothing algorithms. However, the patch may lose meaningful contextual information if it's too small and the selection process may be too labor intensive.

References

[1] Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Süsstrunk, S. (2012). Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282.

[2] An, X. and Pellacini, F. (2008). Appprop: all-pairs appearance-space edit propagation. *ACM Transactions on Graphics (TOG)*, 27(3):40.

[3] Arbelaez, P., Maire, M., Fowlkes, C., and Malik, J. (2011a). Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):898–916.

[4] Arbelaez, P., Maire, M., Fowlkes, C., and Malik, J. (2011b). Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):898–916.

[5] Bae, S., Paris, S., and Durand, F. (2006). Two-scale tone management for photographic look. *ACM Trans. Graph.*, 25(3):637–645.

[6] Bao, L., Song, Y., Yang, Q., Yuan, H., and Wang, G. (2014). Tree filtering: Efficient structure-preserving smoothing with a minimum spanning tree. *IEEE Transactions on Image Processing*, 23(2):555–569.

[7] Baveye, Y., Urban, F., Chamaret, C., Demoulin, V., and Hellier, P. (2013). Saliency-guided consistent color harmonization. In *Computational Color Imaging*, pages 105–118. Springer.

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

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

[10] Bonneel, N., Sunkavalli, K., Paris, S., and Pfister, H. (2013). Example-based video color grading. *ACM Trans. Graph.*, 32(4):39–1.

[11] Bychkovsky, V., Paris, S., Chan, E., and Durand, F. (2011). Learning photographic global tonal adjustment with a database of input/output image pairs. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 97–104.

[12] Chang, J., Wei, D., and Fisher, J. (2013). A video representation using temporal superpixels. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2051–2058.

[13] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2015). Semantic image segmentation with deep convolutional nets and fully connected crfs. *ICLR*.

[14] Cho, H., Lee, H., Kang, H., and Lee, S. (2014). Bilateral texture filtering. *ACM Transactions on Graphics (TOG)*, 33(4):128.

[15] Cohen-Or, D., Sorkine, O., Gal, R., Leyvand, T., and Xu, Y.-Q. (2006). Color harmonization. *ACM Trans. Graph.*, 25(3):624–630.

[16] Dale, K., Johnson, M., Sunkavalli, K., Matusik, W., and Pfister, H. (2009). Image restoration using online photo collections. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2217–2224.

[17] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE.

[18] Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. (2013). Decaf: A deep convolutional activation feature for generic visual recognition. *arXiv preprint arXiv:1310.1531*.

[19] Dong, C., Deng, Y., Change Loy, C., and Tang, X. (2015). Compression artifacts reduction by a deep convolutional network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 576–584.

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

[21] Fan, Q., Yang, J., Hua, G., Chen, B., and Wipf, D. (2017). A generic deep architecture for single image reflection removal and image smoothing. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

[22] Farbman, Z., Fattal, R., Lischinski, D., and Szeliski, R. (2008). Edge-preserving decompositions for multi-scale tone and detail manipulation. In *ACM Transactions on Graphics (TOG)*, volume 27, page 67. ACM.

[23] Fattal, R. (2009). Edge-avoiding wavelets and their applications. *ACM Transactions on Graphics (TOG)*, 28(3):22.

[24] Felzenszwalb, P. F. and Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181.

[25] Ferradans, S., Bertalmio, M., Provenzi, E., and Caselles, V. (2011). An analysis of visual adaptation and contrast perception for tone mapping. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(10):2002–2012.

[26] Fu, X., Zeng, D., Huang, Y., Zhang, X. P., and Ding, X. (2016). A weighted variational model for simultaneous reflectance and illumination estimation. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2782–2790.

[27] Gatys, L. A., Ecker, A. S., and Bethge, M. (2016). Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2414–2423.

[28] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE conference on computer vision and pattern recognition*, pages 580–587.

[29] Gu, B., Li, W., Zhu, M., and Wang, M. (2013). Local edge-preserving multiscale decomposition for high dynamic range image tone mapping. *IEEE Transactions on Image Processing*, 22(1):70–79.

[30] Ham, B., Cho, M., and Ponce, J. (2015). Robust image filtering using joint static and dynamic guidance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4823–4831.

[31] Ham, B., Cho, M., and Ponce, J. (2017). Robust guided image filtering using nonconvex potentials. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[32] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

[33] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[34] Jaya, V. and Gopikakumari, R. (2013). Iem: a new image enhancement metric for contrast and sharpness measurements. *International Journal of Computer Applications*, 79(9).

[35] Joshi, N., Matusik, W., Adelson, E. H., and Kriegman, D. J. (2010). Personal photo enhancement using example images. *ACM Trans. Graph*, 29(2):1–15.

[36] Kang, S. B., Kapoor, A., and Lischinski, D. (2010). Personalization of image enhancement. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1799–1806.

[37] Kim, J., Kwon Lee, J., and Mu Lee, K. (2016a). Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1646–1654.

[38] Kim, J., Kwon Lee, J., and Mu Lee, K. (2016b). Deeply-recursive convolutional network for image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1637–1645.

[39] Kingma, D. and Ba, J. (2015). Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[40] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *NIPS*.

[41] Lazebnik, S., Schmid, C., and Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2169–2178. IEEE.

[42] Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al. (2017). Photo-realistic single image super-resolution using a generative adversarial network. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[43] Lee, J.-Y., Sunkavalli, K., Lin, Z., Shen, X., and Kweon, I. S. (2016). Automatic content-aware color and tone stylization. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[44] Li, G. and Yu, Y. (2016). Deep contrast learning for salient object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[45] Li, Y., Huang, J.-B., Ahuja, N., and Yang, M.-H. (2016). Deep joint image filtering. In *European Conference on Computer Vision*, pages 154–169. Springer.

[46] Liang, Z., Liu, W., and Yao, R. (2016). Contrast enhancement by nonlinear diffusion filtering. *IEEE Transactions on Image Processing*, 25(2):673–686.

[47] Lim, B., Son, S., Kim, H., Nah, S., and Lee, K. M. (2017). Enhanced deep residual networks for single image super-resolution. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

[48] Lischinski, D., Farbman, Z., Uyttendaele, M., and Szeliski, R. (2006). Interactive local adjustment of tonal values. *ACM Trans. Graph.*, 25(3):646–653.

[49] Liu, S., Pan, J., and Yang, M.-H. (2016). Learning recursive filters for low-level vision via a hybrid neural network. In *European Conference on Computer Vision*, pages 560–576. Springer.

[50] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., and Reed, S. (2015a). Ssd: Single shot multibox detector. *arXiv preprint arXiv:1512.02325*.

[51] Liu, W., Rabinovich, A., and Berg, A. C. (2015b). Parsenet: Looking wider to see better. *arXiv preprint arXiv:1506.04579*.

[52] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *CVPR*.

[53] Ma, K., Wu, Q., Wang, Z., Duanmu, Z., Yong, H., Li, H., and Zhang, L. (2016). Group MAD competition - a new methodology to compare objective image quality models.

[54] Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605.

[55] Malinowski, M., Rohrbach, M., and Fritz, M. (2015). Ask your neurons: A neural-based approach to answering questions about images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1–9.

[56] Mao, X., Shen, C., and Yang, Y.-B. (2016). Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In *Advances in Neural Information Processing Systems*, pages 2802–2810.

[57] Martin, D., Fowlkes, C., Tal, D., and Malik, J. (2001). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423.

[58] Min, D., Choi, S., Lu, J., Ham, B., Sohn, K., and Do, M. N. (2014). Fast global image smoothing based on weighted least squares. *IEEE Transactions on Image Processing*, 23(12):5638–5653.

[59] Paris, S., Hasinoff, S. W., and Kautz, J. (2011). Local laplacian filters: edge-aware image processing with a laplacian pyramid. *ACM Trans. Graph.*, 30(4):68.

[60] Perona, P. and Malik, J. (1990). Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on pattern analysis and machine intelligence*, 12(7):629–639.

[61] Petschnigg, G., Agrawala, M., Hoppe, H., Szeliski, R., Cohen, M., and Toyama, K. (2004). Digital photography with flash and no-flash image pairs. *ACM Transactions on Graphics*, 23(3):664–672.

[62] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99.

[63] Ruder, M., Dosovitskiy, A., and Brox, T. (2016). Artistic style transfer for videos. *arXiv preprint arXiv:1604.08610*.

[64] Shih, Y., Paris, S., Durand, F., and Freeman, W. T. (2013). Data-driven hallucination of different times of day from a single outdoor photo. *ACM Transactions on Graphics (TOG)*, 32(6):200.

[65] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*.

[66] Son, M., Lee, Y., Kang, H., and Lee, S. (2014). Art-photographic detail enhancement. In *Computer Graphics Forum*, volume 33, pages 391–400. Wiley Online Library.

[67] Subr, K., Soler, C., and Durand, F. (2009). Edge-preserving multiscale image decomposition based on local extrema. In *ACM Transactions on Graphics (TOG)*, volume 28, page 147. ACM.

[68] Tai, Y., Yang, J., and Liu, X. (2017a). Image super-resolution via deep recursive residual network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[69] Tai, Y., Yang, J., Liu, X., and Xu, C. (2017b). Memnet: A persistent memory network for image restoration. In *Proceedings of International Conference on Computer Vision*.

[70] Tighe, J. and Lazebnik, S. (2010). Superparsing: Scalable nonparametric image parsing with superpixels. In *European Conference on Computer Vision: Part V*, ECCV'10, pages 352–365.

[71] Tomasi, C. and Manduchi, R. (1998). Bilateral filtering for gray and color images. In *Computer Vision, 1998. Sixth International Conference on*, pages 839–846. IEEE.

[72] Wang, B., Yu, Y., Wong, T.-T., Chen, C., and Xu, Y.-Q. (2010). Data-driven image color theme enhancement. In *ACM Transactions on Graphics (TOG)*, volume 29, page 146. ACM.

[73] Wang, B., Yu, Y., and Xu, Y.-Q. (2011). Example-based image color and tone style enhancement. *ACM Transactions on Graphics (TOG)*, 30(4):64.

[74] Wang, X., Yang, M., Zhu, S., and Lin, Y. (2013). Regionlets for generic object detection. In *IEEE International Conference on Computer Vision*, pages 17–24.

[75] Xu, C. and Corso, J. J. (2012). Evaluation of super-voxel methods for early video processing. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1202–1209. IEEE.

[76] Xu, L., Lu, C., Xu, Y., and Jia, J. (2011). Image smoothing via l 0 gradient minimization. In *ACM Transactions on Graphics (TOG)*, volume 30, page 174. ACM.

[77] Xu, L., Ren, J., Yan, Q., Liao, R., and Jia, J. (2015). Deep edge-aware filters. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1669–1678.

[78] Xue, S., Agarwala, A., Dorsey, J., and Rushmeier, H. (2013). Learning and applying color styles from feature films. In *Computer Graphics Forum*, volume 32, pages 255–264. Wiley Online Library.

[79] Yan, J., Lin, S., Kang, S. B., and Tang, X. (2014). A learning-to-rank approach for image color enhancement. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 2987–2994. IEEE.

[80] Yan, Z., Zhang, H., Jia, Y., Breuel, T., and Yu, Y. (2016a). Combining the best of convolutional layers and recurrent layers: A hybrid network for semantic segmentation. *arXiv preprint arXiv:1603.04871*.

[81] Yan, Z., Zhang, H., Wang, B., Paris, S., and Yu, Y. (2016b). Automatic photo adjustment using deep neural networks. *ACM Transactions on Graphics*, 35(2).

[82] Yeganeh, H. and Wang, Z. (2013). Objective quality assessment of tone-mapped images. *IEEE Transactions on Image Processing*, 22(2):657–667.

[83] Yu, F. and Koltun, V. (2015). Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*.

[84] Yue, H., Yang, J., Sun, X., Wu, F., and Hou, C. (2017). Contrast enhancement based on intrinsic image decomposition. *IEEE Transactions on Image Processing*, 26(8):3981–3994.

[85] Zhang, K., Zuo, W., Chen, Y., Meng, D., and Zhang, L. (2017). Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*.

[86] Zhang, Q., Shen, X., Xu, L., and Jia, J. (2014a). Rolling guidance filter. In *European Conference on Computer Vision*, pages 815–830. Springer.

[87] Zhang, Q., Xu, L., and Jia, J. (2014b). 100+ times faster weighted median filter (wmf). In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2830–2837.

[88] Zhou, B., Khosla, A., Lapedriza, A., Torralba, A., and Oliva, A. (2016). Places: An image database for deep scene understanding. *arXiv preprint arXiv:1610.02055*.

[89] Zhu, F., Yan, Z., Bu, J., and Yu, Y. (2017). Exemplar-based image and video stylization using fully convolutional semantic features. *IEEE Transactions on Image Processing*, 26(7):3542–3555.