

Advances in Computer Vision and Pattern Recognition



Marcelo Bertalmío *Editor*

# Denoising of Photographic Images and Video

Fundamentals, Open Challenges and  
New Trends

 Springer

The Springer logo, which is a stylized white chess knight (horse) facing left, positioned to the left of the word "Springer" in a white serif font.

# **Advances in Computer Vision and Pattern Recognition**

## **Founding editor**

Sameer Singh, Rail Vision, Castle Donington, UK

## **Series editor**

Sing Bing Kang, Microsoft Research, Redmond, WA, USA

## **Advisory Board**

Horst Bischof, Graz University of Technology, Austria

Richard Bowden, University of Surrey, Guildford, UK

Sven Dickinson, University of Toronto, ON, Canada

Jiaya Jia, The Chinese University of Hong Kong, Hong Kong

Kyoung Mu Lee, Seoul National University, South Korea

Yoichi Sato, The University of Tokyo, Japan

Bernt Schiele, Max Planck Institute for Computer Science, Saarbrücken, Germany

Stan Sclaroff, Boston University, MA, USA

More information about this series at <http://www.springer.com/series/4205>

Marcelo Bertalmío  
Editor


# Denoising of Photographic Images and Video

Fundamentals, Open Challenges and New  
Trends

 Springer



*Editor*

Marcelo Bertalmío   
Pompeu Fabra University  
Barcelona, Spain

ISSN 2191-6586

ISSN 2191-6594 (electronic)

Advances in Computer Vision and Pattern Recognition

ISBN 978-3-319-96028-9

ISBN 978-3-319-96029-6 (eBook)

<https://doi.org/10.1007/978-3-319-96029-6>

Library of Congress Control Number: 2018948593

© Springer International Publishing AG, part of Springer Nature 2018

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG  
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland



*A Serrana, Graciela, Lucas y Vera*

# Preface

Noise is always present in images, regardless of the way they have been acquired. This is why noise removal or denoising is a key image processing problem, especially with respect to photo and video cameras, where the push toward ever-increasing resolution, dynamic range, and frame rate implies ever higher demands on denoising performance.

Denoising has a long and rich history, with early works dating back to the 1960s. Classic denoising techniques were mostly based in one of these two approaches: modification of transform coefficients (using the Fourier transform, the DCT, some form of wavelet, etc.) or averaging image values (in a neighborhood, along contours, with similar but possibly distant pixels, etc.). Both types of approaches yielded results that were modest, in terms of objective error metrics and also in terms of visual quality, with frequent problems such as oversmoothing, staircase effects, or ringing artifacts.

In 2005, the groundbreaking nonlocal means method proposed comparing image neighborhoods (patches) in order to denoise single pixels. This approach produced results that were shockingly superior to the state of the art at the time, so much so that from then on virtually all image denoising algorithms have been patch-based. Actually, the increase in quality of the denoising algorithms in the past few years has been so dramatic that several recent works have questioned whether or not there is still room for improvement in denoising, with some researchers considering the problem pretty much solved and no longer relevant.

One of the goals of this book is to show that, in fact, that is not the case: in denoising there are some fundamental challenges that remain unsolved and that include how to properly model noise in real scenarios, how to tailor denoising algorithms to these models, and how to evaluate the results in a way that is consistent with perceived image quality.

Another goal was to have a book dealing exclusively with noise removal for photographs and video. Despite the commercial significance of the image and video industry and the fact that many academic works on denoising are evaluated on regular photos and videos, this would be, surprisingly, the first book centered on this specific topic.

This volume provides a comprehensive look on the subject, from problem formulation to the evaluation of denoising methods, from historical perspectives to state-of-the-art algorithms, and from fast real-time techniques that can be implemented in camera to powerful and computationally intensive methods for off-line processing. All topics are explained in detail yet in a clear and concise manner. The intended audience comprises researchers and advanced undergraduate and graduate students in computer science, applied mathematics, and related fields, as well as professionals from the media industries.

Finally, I would like to point out that it's been a great pleasure to edit this volume and have contributions from so many outstanding researchers, sharing their insights on this fascinating problem.

And now, paraphrasing the British rock band Slade: come on feel the noise!

Barcelona, Spain  
May 2018

Marcelo Bertalmío

“The relentless quest for higher image resolution, greater ISO sensitivity, faster frame rates and smaller imaging sensors in digital imaging and videography has demanded unprecedented innovation and improvement in noise reduction technologies. This book provides a comprehensive treatment of all aspects of image noise including noise modelling, state of the art noise reduction technologies and visual perception and quantitative evaluation of noise.”

—Geoff Woolfe, *Former President of the Society for Imaging Science and Technology*

“This book on denoising of photographic images and video is the most comprehensive and up-to-date account of this deep and classic problem of image processing. The progress on its solution is being spectacular. This volume therefore is a must read for all engineers and researchers concerned with image and video quality.”

—Jean-Michel Morel, *Professor at Ecole Normale Supérieure de Cachan, France*

# Contents

<b>1</b>	<b>Modeling and Estimation of Signal-Dependent and Correlated Noise</b> . . . . .	<b>1</b>
	Lucio Azzari, Lucas Rodrigues Borges and Alessandro Foi	
<b>2</b>	<b>Sparsity-Based Denoising of Photographic Images: From Model-Based to Data-Driven</b> . . . . .	<b>37</b>
	Xin Li, Weisheng Dong and Guangming Shi	
<b>3</b>	<b>Image Denoising—Old and New</b> . . . . .	<b>63</b>
	Michael Moeller and Daniel Cremers	
<b>4</b>	<b>Convolutional Neural Networks for Image Denoising and Restoration</b> . . . . .	<b>93</b>
	Wangmeng Zuo, Kai Zhang and Lei Zhang	
<b>5</b>	<b>Gaussian Priors for Image Denoising</b> . . . . .	<b>125</b>
	Julie Delon and Antoine Houdard	
<b>6</b>	<b>Internal Versus External Denoising—Benefits and Bounds</b> . . . . .	<b>151</b>
	Maria Zontak and Michal Irani	
<b>7</b>	<b>Patch-Based Methods for Video Denoising</b> . . . . .	<b>175</b>
	A. Buades and J. L. Lisani	
<b>8</b>	<b>Image and Video Noise: An Industry Perspective</b> . . . . .	<b>207</b>
	Stuart Perry	
<b>9</b>	<b>Noise Characteristics and Noise Perception</b> . . . . .	<b>235</b>
	Tamara Seybold	

**10 Pull-Push Non-local Means with Guided and Burst Filtering Capabilities** ..... 267  
John R. Isidoro and Peyman Milanfar

**11 Three Approaches to Improve Denoising Results that Do Not Involve Developing New Denoising Methods** ..... 295  
Gabriela Ghimpeteanu, Thomas Batard, Stacey Levine and Marcelo Bertalmío

**Index** ..... 331



# Chapter 1

## Modeling and Estimation of Signal-Dependent and Correlated Noise



Lucio Azzari, Lucas Rodrigues Borges and Alessandro Foi

**Abstract** The additive white Gaussian noise (AWGN) model is ubiquitous in signal processing. This model is often justified by central-limit theorem (CLT) arguments. However, whereas the CLT may support a Gaussian distribution for the random errors, it does not provide any justification for the assumed additivity and whiteness. As a matter of fact, data acquired in real applications can seldom be described with good approximation by the AWGN model, especially because errors are typically correlated and not additive. Failure to model accurately the noise leads to inaccurate analysis, ineffective filtering, and distortion or even failure in the estimation. This chapter provides an introduction to both signal-dependent and correlated noise and to the relevant models and basic methods for the analysis and estimation of these types of noise. Generic one-parameter families of distributions are used as the essential mathematical setting for the observed signals. The distribution families covered as leading examples include Poisson, mixed Poisson–Gaussian, various forms of signal-dependent Gaussian noise (including multiplicative families and approximations of the Poisson family), as well as doubly censored heteroskedastic Gaussian distributions. We also consider various forms of noise correlation, encompassing pixel and readout cross-talk, fixed-pattern noise, column/row noise, etc., as well as related issues like photo-response and gain nonuniformity. The introduced models and methods are applicable to several important imaging scenarios and technologies, such as raw data from digital camera sensors, various types of radiation imaging relevant to security and to biomedical imaging.

---

L. Azzari · A. Foi (✉)  
Tampere University of Technology, Korkeakoulunkatu 1, 33720 Tampere, Finland  
e-mail: alessandro.foi@tut.fi

L. Azzari  
e-mail: lucio.azzari@tut.fi

L. R. Borges  
São Carlos School of Engineering, University of São Paulo,  
400 Trabalhador São Carlense Avenue, São Carlos 13566-590, Brazil  
e-mail: lucas.rodrigues.borges@usp.br

© Springer International Publishing AG, part of Springer Nature 2018  
M. Bertalmío (ed.), *Denoising of Photographic Images and Video*,  
Advances in Computer Vision and Pattern Recognition,  
[https://doi.org/10.1007/978-3-319-96029-6\\_1](https://doi.org/10.1007/978-3-319-96029-6_1)

## 1.1 Introduction: Acquisition Devices and Noise Sources

A digital image is generated by converting the light coming from a natural scene to numerical pixel values. In particular, a typical camera performs this conversion using a semiconducting array of sensing elements positioned after an aperture: when the shutter opens, the light from the scene goes through the lenses and the aperture, finally colliding with the sensor array. Each element in the array converts the energy of the incident light beam to electric charges that are successively accumulated in an electric potential. The electric potentials are then converted to digital values, and finally stored collectively as a raw image, whose pixel values are ideally proportional to the intensity of the light that shone onto the corresponding sensing elements.

The most common digital camera sensors are Charge Coupled Semiconductor Devices (CCD) and Complementary Metal-Oxide Semiconductor (CMOS). While CCD used to be the most common technology, nowadays CMOS sensors dominate the market, being the preferred capture technology for smartphones and digital cameras. The main difference between the two is that, while in CCD arrays the charge of a row of sensors is transported via the same circuit, sharing also the same amplifier, CMOS arrays are based on the Active Pixel Sensor (ASP) technology, for which every single sensor is treated independently, having a unique transport line [16].

To get a basic understanding of the nature of the noise in imaging sensors, let us consider the acquisition of a still scene; although the average incident energy over a relatively long period of time might be virtually constant, the amount of photons incident on the camera sensors during the exposure fluctuates in time. Furthermore, not all the incident photons are converted to electric charge. This whole phenomenon is known as shot noise, and it is well modeled by the family of Poisson distributions [23]. An important feature of this type of noise is that it is *signal-dependent*, in the sense that the electric charge fluctuates in time with a variance that is proportional to the photon flux. Thus, different parts of a captured scene are subject to different noise strengths with the stronger noise affecting the brighter content.

Another relevant source of noise is the so-called thermal noise. Thermal noise is generated by thermal agitation [17, 26] and is due to the fact that at any given temperature (except absolute zero), conductors have a probability to emit charges due to heat, even when there is no electric potential to stimulate them. This results in a background current, present also in the absence of input signals (dark current) [16], which alters the measurements of the sensors. The inevitable fluctuations of this current are thus modeled as noise. By definition, this type of noise is proportional to the working temperature of the device, and it can therefore be reduced by decreasing the temperature of sensor. In high-end devices for scientific applications (e.g., optical astronomy), this is achieved by means of a thermoelectric cooler such as a Peltier heat pump; however, in most consumer applications, sensor cooling is not feasible and thermal noise, suitably modeled by a Gaussian distribution, becomes a significant component of the measurement errors. Particularly when capturing scenes in low-light conditions, thermal noise can dominate the overall noise.

Furthermore, there is a possibility that the charges accumulated by neighboring sensor elements coupled with each other introducing correlation between measured quantities. In other words, the charge accumulated by a sensor element is influenced not only by the number of incident photons, but also by the charges accumulated by the surrounding sensor elements. Analogously, during the readout phase, when the device reads and transports the charges from the sensor elements, there could be some electrical coupling of the quantities. This introduces an error in the acquisition process that is commonly referred to as *cross-talk*, and it is usually well modeled by the adoption of correlated noise, in which the measurement error for a pixel is influenced also by the surrounding errors.

Finally, the electric potential is often amplified (analog gain) before being converted to a digital value by an analog-to-digital converter. This analog amplification may introduce further noise and, because the digital values are discrete with a certain bit depth, we eventually encounter also quantization noise, which is sometimes approximated by uniformly distributed errors over one quantization step, or as a generic additive noise with comparable variance (i.e., one-twelfth of the quantization step, as per basic properties of the uniform distribution).

## 1.2 Additive White Gaussian Noise

As highlighted above, a signal acquired by a digital device is affected by noise from several sources. It is often difficult to separate and treat each noise source individually, as this requires in-depth knowledge of the device and direct access to some of its inner components; therefore, the various sources are conventionally grouped together and addressed as a single noise process. This procedure is encouraged by the central-limit theorem (CLT) [27, 31]: for a set of  $N$  independent random variables  $X_1, \dots, X_N$ , with respective means  $\mu_1, \dots, \mu_N$  and standard deviations  $\sigma_1, \dots, \sigma_N$ , as  $N \rightarrow \infty$  we have

$$\frac{1}{s} \sum_{i=1}^N (X_i - \mu_i) \xrightarrow{d} \mathcal{N}(0, 1) \quad \text{with} \quad s = \sqrt{\sum_{i=1}^N \sigma_i^2}, \quad (1.1)$$

where  $\xrightarrow{d}$  denotes the convergence in distribution, and  $\mathcal{N}(0, 1)$  indicates a Gaussian (also called normal) random variable with mean and variance equal, respectively, to the first and second arguments within parenthesis (in this case 0 and 1). In other words, we can represent the sum of various random noise sources as a Gaussian random variable, irrespective of the noise distribution of the individual sources. The CLT establishes the importance of the Gaussian distribution in modeling complex physical processes.

The Gaussian noise is further commonly assumed additive, zero-mean, independent, and identically distributed (i.i.d.); under these extra assumptions, a captured image  $z$  is modeled as

$$z(x) = y(x) + \eta(x), \quad (1.2)$$

where  $y$  is the underlying deterministic noise-free image,  $x \in \Omega \subset \mathbb{Z}^2$  is the pixel coordinate, and  $\eta(\cdot) \sim \mathcal{N}(0, \sigma^2)$  is the zero-mean Gaussian random variable with variance  $\sigma^2$ . Each coordinate  $x$  results in an independent (hence different) realization of the random variable  $\eta(x)$ , which collectively for all  $x \in \Omega$  yields the additive white Gaussian noise (AWGN) field corrupting  $y$ . The term *white* is inspired by spectroscopy: like white light dispersed through a prism reveals components for every frequency in the visible spectrum (from 400 THz of red to 789 THz of violet), Fourier analysis of white noise reveals components for every frequency within the Fourier spectrum. Specifically, when working on a 2D rectangular image domain of  $N_1 \times N_2$  pixels, i.e.,  $x = (x_1, x_2) \in \Omega = [0, \dots, N_1 - 1] \times [0, \dots, N_2 - 1]$ , a generic Fourier coefficient of  $\eta$  can be written as

$$\mathcal{F}[\eta](\xi_1, \xi_2) = \sum_{x_2=0}^{N_2-1} \sum_{x_1=0}^{N_1-1} e^{-2\pi i \left( \xi_1 \frac{x_1}{N_1} + \xi_2 \frac{x_2}{N_2} \right)} \eta(x_1, x_2), \quad (1.3)$$

where  $\mathcal{F}$  denotes the Fourier transform and  $\xi_1, \xi_2$  are spatial frequencies. The noise power spectrum (also called power spectral density, PSD) corresponds to the variance of  $\mathcal{F}[\eta]$ , which can be computed as

$$\text{var} \{ \mathcal{F}[\eta](\xi_1, \xi_2) \} = \sum_{x_2=0}^{N_2-1} \sum_{x_1=0}^{N_1-1} \text{var} \left\{ e^{-2\pi i \left( \xi_1 \frac{x_1}{N_1} + \xi_2 \frac{x_2}{N_2} \right)} \eta(x_1, x_2) \right\} = \quad (1.4)$$

$$= \sum_{x_2=0}^{N_2-1} \sum_{x_1=0}^{N_1-1} \left| e^{-2\pi i \left( \xi_1 \frac{x_1}{N_1} + \xi_2 \frac{x_2}{N_2} \right)} \right|^2 \text{var} \{ \eta(x_1, x_2) \} = \quad (1.5)$$

$$= \sum_{x_2=0}^{N_2-1} \sum_{x_1=0}^{N_1-1} \text{var} \{ \eta(x_1, x_2) \} = \quad (1.6)$$

$$= \sum_{x_2=0}^{N_2-1} \sum_{x_1=0}^{N_1-1} \sigma^2 = N_1 N_2 \sigma^2, \quad (1.7)$$

i.e., the power (variance) of the noise is constant in the Fourier domain and directly proportional to the variance in the pixel domain. We can say that the Fourier spectrum of white noise is *flat*. Equalities (1.4)–(1.7) leverage few basic properties: (1)  $\eta$  is independently distributed; (2) multiplication of  $\eta$  by a deterministic function (i.e., the complex exponential) does not affect the independence; hence (3) the Fourier coefficient  $\mathcal{F}[\eta](\xi_1, \xi_2)$  (1.3) is simply a sum of independent random variables; (4) the variance of the sum of independent random variables is the sum of their variances (1.4); (5) multiplication of a random variable by a deterministic factor scales the variance by the squared modulus of the factor (1.5); (6) complex exponentials are always on the unit circle of the complex plane, i.e., they have unit modulus (1.6);

and (7)  $\eta$  is identically distributed with constant variance  $\sigma^2$  (1.7). We can see that of the i.i.d. hypothesis, the independence alone is sufficient to reach (1.6) and that having identical distributions (hence, identical variance  $\sigma^2$  for every  $x \in \Omega$ ) is used only for obtaining (1.7). Already (1.6) shows that the Fourier spectrum of the noise is flat, since  $\text{var} \{ \mathcal{F}[\eta](\xi_1, \xi_2) \}$  no longer depends on  $\xi_1$  or  $\xi_2$ . Indeed, we can have white noise with a flat Fourier power spectrum for models different from the AWGN (1.2).

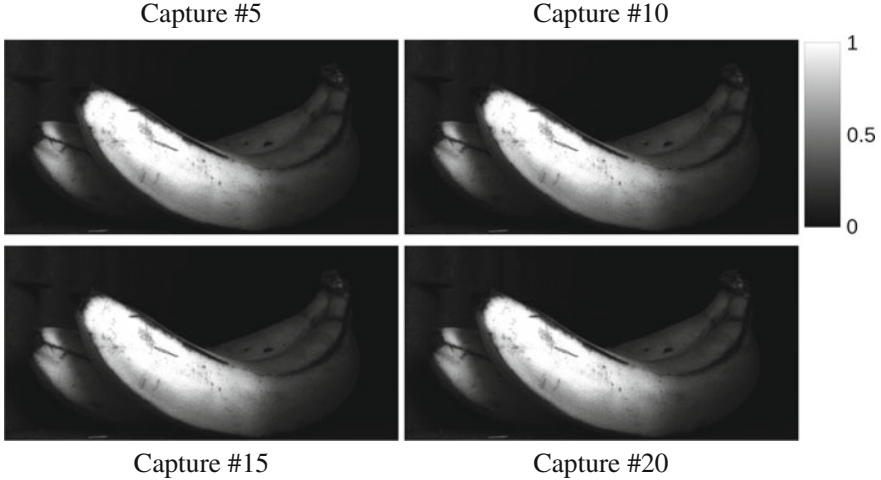
At this point, it is important to emphasize that although theoretically the CLT justifies using a Gaussian distribution for modeling the measurement random errors, it does not provide arguments supporting the additional assumptions of the AWGN model, namely, that the errors are independent and identically distributed over the image. Indeed, in (1.1), we can observe that the term on the left-hand side is essentially a standardization of the errors, which thus explicitly depends on the means  $\mu_i$  and variances  $\sigma_i^2$  of the individual contributors. In general, we have that each noisy pixel  $z(x)$  results from its own sequence  $X_i(x)$ ,  $i = 1, \dots, N$ , where the means and the variances of these contributors can be different at different pixels; in other words, the errors may not be identically distributed and even when a Gaussian model (as per the CLT) may be appropriate, then mean and (most often) the variance of such Gaussian errors may change from pixel to pixel. Furthermore, contributors of different pixels can be subject to a mutual interaction, possibly resulting in a statistical dependence between the measurement errors at different pixels; in other words, the errors may not be independently distributed over the image.

It is clear from these premises and from the summary in Sect. 1.1 that the AWGN model is not suitable for modeling the above measurement errors, first because it inherently uses a single distribution, and second because it assumes independent errors.

In Sect. 1.4, we begin from generalizing the observation model to accommodate a multiplicity of distributions. Specifically, we adopt the formalism of one-parameter families of distributions, where the observation at each pixel follows a specific distribution that depends on a known or unknown univariate parameter. Further, in Sect. 1.6.1, we address the issue of correlation in the errors.

### 1.3 Raw Image Dataset

Throughout this chapter, we use real sensor raw data to validate the presented models and methods. As leading example, we use a dataset consisting of  $M = 30$  raw images of the same still scene acquired repeatedly in a short time interval (at a rate of about 1 frame/s) under identical capture settings. We identify the individual images in the set as  $\tilde{z}^{(m)}$ ,  $m = 1, \dots, 30$ , while  $\tilde{z}$  denotes a generic such image; the reason for using the tilde decoration here will become clear in the further sections. The images have been captured by a Samsung S5K2L2 CMOS ISOCELL sensor with a  $1.4 \mu\text{m}$  pixel size at ISO 1250; this type of sensor can be found in modern mobile devices such



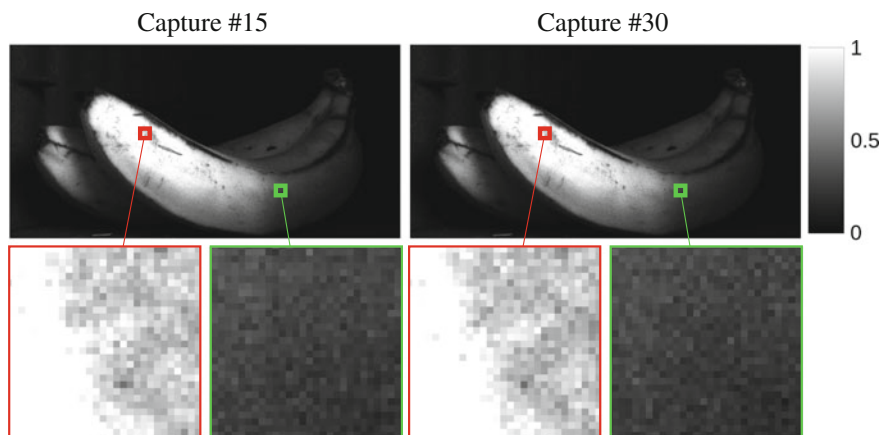
**Fig. 1.1** Examples from the dataset of 30 raw images captured under identical settings with a Samsung S5K2L2 CMOS ISOCELL sensor at ISO 1250

as the Samsung Galaxy S8 smartphone. Raw images from this sensor are stored in 10-bit format, which we normalize to the range  $[0, 1]$  by dividing the raw integer values by  $2^{10} - 1$ . Four images from the dataset are reported in Fig. 1.1: the scene features a dark background with two bananas in the foreground. The presence of both dark and bright regions makes the dataset suitable for validating the noise models described in the following sections. Even though these sensors are typically used in conjunction with a color filter array such as the RGGGB Bayer filter mosaic, for the sake of simplicity of presentation we consider only single-channel monochromatic (green) acquisition.

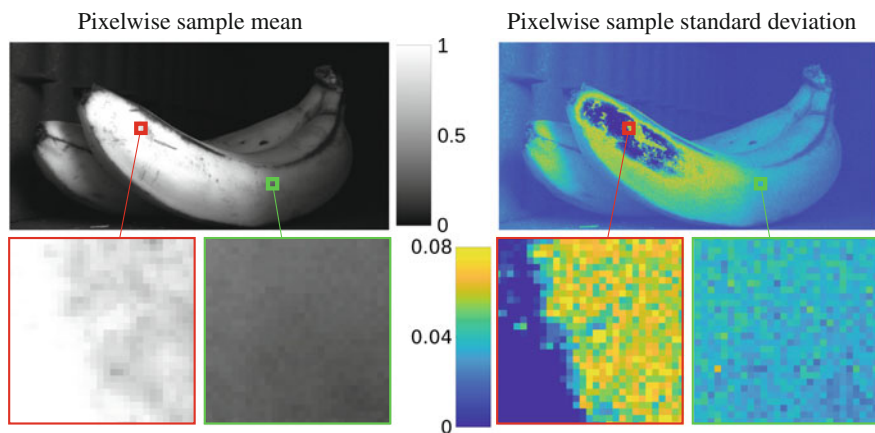
In the top row of Fig. 1.2, we show two images of the dataset. By inspecting the enlarged fragments, we can observe how the two images are practically equal except for the individual noise realizations. Indeed, we can formally define the image noise as the difference between the captured images and their mathematical expectation, i.e., the average of infinitely many images like those captured in the dataset, where the latter, denoted by  $\tilde{y}$ , can be treated as the ideal noise-free image. Since the dataset contains only finitely many images ( $M = 30$ ), we can approximate the mathematical expectation by the pixelwise sample average,

$$\tilde{y}(x) = \mathbb{E}\{\tilde{z}(x)\} \approx \frac{1}{M} \sum_{m=1}^M \tilde{z}^{(m)}(x) = \mathbb{E}\{\widehat{\tilde{z}}(x)\}, \quad (1.8)$$

resulting in the image in the left-hand side of Fig. 1.3. Note in the enlarged fragments how the noise is virtually removed everywhere.



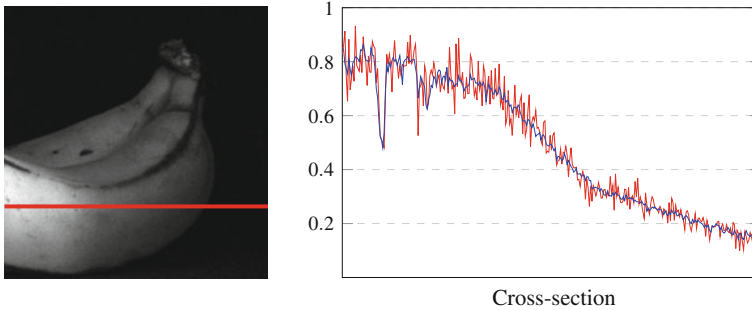
**Fig. 1.2** Two raw images of the same scene, captured under identical settings. A total of 30 images like these were captured



**Fig. 1.3** Pixelwise sample mean (left) and sample standard deviation (right) across all the captured images of the dataset

The cross section shown in Fig. 1.4 introduces the reader to a main feature that is shared by most of the noise types under consideration: noise affecting the bright parts of the image is significantly stronger (i.e., larger errors) compared to the noise affecting dark regions. This can be quantified as the standard deviation of the noise at each pixel (again, computed over infinitely many such captured images), which we can approximate by the pixelwise sample standard deviation over the finite dataset,

$$\text{std} \{ \tilde{z}(x) \} \approx \sqrt{\frac{1}{M-1} \sum_{m=1}^M \left( \tilde{z}^{(m)}(x) - \frac{1}{M} \sum_{l=1}^M \tilde{z}^{(l)}(x) \right)^2} = \text{std} \{ \widehat{\tilde{z}}(x) \}, \quad (1.9)$$



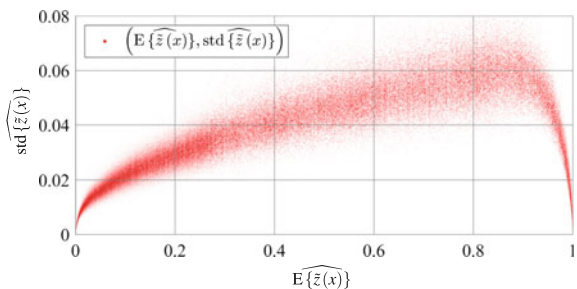
**Fig. 1.4** Left: detail from the dataset with highlighted cross section. Right: cross section (red line) plotted against its expectation  $\widehat{E\{\tilde{z}(x)\}}$  (in blue). Note how the noise is *signal-dependent*, with different variances at each pixel  $x$  depending on the value of the underlying expectation

which is shown in the image in the right-hand side of Fig. 1.3. To provide a visual exploration of the relation between the expectation and the standard deviation of the noisy raw pixels, Fig. 1.5 shows a scatterplot where each red dot represents a  $\left(\widehat{E\{\tilde{z}(x)\}}, \widehat{\text{std}\{\tilde{z}(x)\}}\right)$  pair for  $x \in \Omega$ . The scatterplot can be interpreted as a cloud of points about an unknown smooth curve that describes the noise standard deviation as function of the signal expectation. However, this interpretation is admissible only if the dispersion of the scatterpoints is compatible with the existence of such curve. Indeed, leveraging the CLT and the first-order Taylor expansion of the square root at  $\text{var}\{\tilde{z}(x)\}$ , the distributions of  $\widehat{E\{\tilde{z}(x)\}}$  and  $\widehat{\text{std}\{\tilde{z}(x)\}}$  can be approximated for large  $M$  as

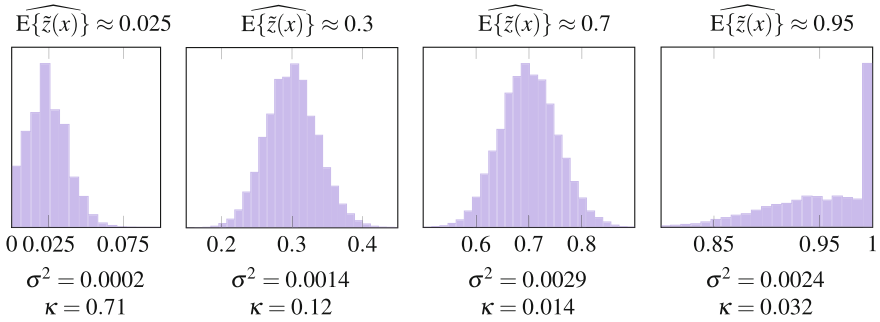
$$\widehat{E\{\tilde{z}(x)\}} \sim \mathcal{N}\left(\widehat{E\{\tilde{z}(x)\}}, \frac{1}{M} \text{var}\{\tilde{z}(x)\}\right), \quad (1.10)$$

$$\widehat{\text{std}\{\tilde{z}(x)\}} \sim \mathcal{N}\left(\widehat{\text{std}\{\tilde{z}(x)\}}, \frac{2 + \kappa}{4M} \text{var}\{\tilde{z}(x)\}\right), \quad (1.11)$$

**Fig. 1.5** Scatterplot of the pairs  $\left(\widehat{E\{\tilde{z}(x)\}}, \widehat{\text{std}\{\tilde{z}(x)\}}\right)$  drawn as red dots. The dispersion visible in the scatterplot is described by the distributions (1.10)–(1.11) of the estimated pairs







**Fig. 1.6** Histograms of the pixels  $\tilde{z}(x)$  from the dataset with pointwise sample mean  $E\{\tilde{z}(x)\} \approx [0.025, 0.3, 0.7, 0.95]$ . Below each histogram, we report its variance  $\sigma^2$  and excess kurtosis  $\kappa$

where  $\kappa$  is the excess kurtosis of  $\tilde{z}(x)$ . Taking into account the sample histograms plotted in Fig. 1.6, the distributions (1.10)–(1.11) fully explain the dispersion visible in the scatterplot and suggest a functional relation between  $E\{\tilde{z}(x)\}$  and  $\text{std}\{\tilde{z}(x)\}$ , which may be obtained, e.g., by processing the scatterplot with a smoother. The histograms also illustrate that  $\tilde{z}$  is not identically distributed and that its distribution varies from pixel to pixel according to the expectation.

Another important feature that can be ascertained from Figs. 1.2 and 1.3 is the fact that the brightest areas of the image *saturate* to white. This phenomenon is commonly referred to as *clipping* and results, in particular, in the drop of sample standard deviation that can be observed in Figs. 1.3 and 1.5, and in the lack of Gaussianity in some of the histograms in Fig. 1.6.

Overall, the above analysis indicates that the noise affecting the dataset images is *signal-dependent* in the sense that its characteristics at each pixel  $x$  depend on the value of the underlying noise-free image, i.e., on  $\tilde{y}(x) = E\{\tilde{z}(x)\}$ .

In the next section, we will derive, step-by-step, a simple yet effective mathematical model that accurately describes the behavior of the points of the scatterplot in Fig. 1.5 that provides a direct functional relation between  $E\{\tilde{z}(x)\}$  and  $\text{std}\{\tilde{z}(x)\}$  and that explains the shape and moments of the histograms as a function of  $E\{\tilde{z}(x)\}$ .

## 1.4 One-Parameter Families of Distributions

A one-parameter family of distributions  $\{\mathcal{D}_\theta\}_{\theta \in \Theta}$  is a collection of distributions, each of which is identified by the value of a univariate parameter  $\theta \in \Theta \subseteq \mathbb{R}$ .

Let  $z$  be a random variable distributed according to a one-parameter family of distributions  $\{\mathcal{D}_\theta\}_{\theta \in \Theta}$ : this means that for each individual  $\theta \in \Theta$ , the conditional distribution of  $z$  given  $\theta$  is  $\mathcal{D}_\theta$ , i.e.,  $z|\theta \sim \mathcal{D}_\theta$ . Hence, the conditional expectation and the conditional standard deviation of  $z$  given  $\theta$ , i.e.,  $E\{z|\theta\}$  and  $\text{std}\{z|\theta\}$ , are two functions of  $\theta$ .

In the following sections, we cover some of the most important one-parameter families of distributions for modeling noise of digital imaging sensors, describing them in detail through the corresponding probability density functions (PDFs) or probability mass functions (PMFs), and their mean and variances.

### 1.4.1 Poisson Noise and Poisson Family of Distributions

The simplest way to model an image captured by a photodetector array is to represent it as a realization of independent *Poisson* random variables. In particular, the captured image  $z$  is modeled as

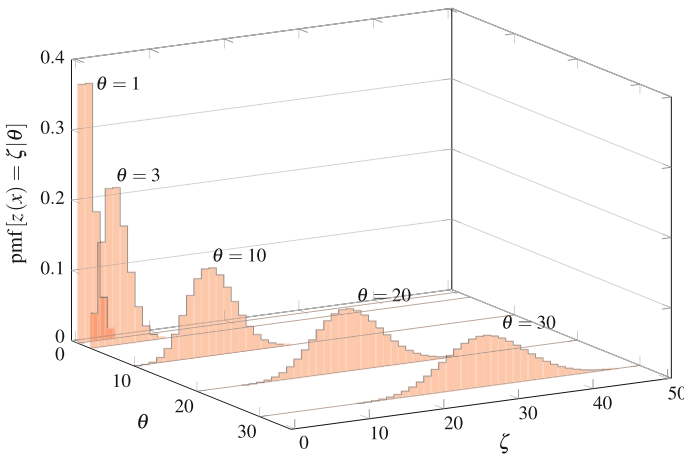
$$z(x) \sim \mathcal{P}(y(x)),$$

$$\text{pmf}[z(x) = \zeta | y(x)] = \begin{cases} \frac{y(x)^\zeta e^{-y(x)}}{\zeta!} & \zeta \in \mathbf{N} \cup \{0\} \\ 0 & \text{elsewhere,} \end{cases} \quad (1.12)$$

where  $y \geq 0$  is the noise-free image, which can be thought as a proxy for the photon flux, and the symbol  $\mathcal{P}$  denotes the Poisson family of distributions. This is a one-parameter family of distributions with parameter  $\theta = y(x)$ , which coincides with the mean and variance of the conditional distributions:

$$\mathbf{E}\{z(x) | y(x)\} = \text{var}\{z(x) | y(x)\} = y(x). \quad (1.13)$$

Figure 1.7 shows examples of distributions from the Poisson family.



**Fig. 1.7** Illustration of the family of Poisson distributions (1.12). We show the distributions with mean  $y(x) = \theta = 1, 3, 10, 20, 50$

*Poisson noise* can be formally defined as  $z - y$ . It is clear from (1.13) that the mean of Poisson noise is zero, i.e.,  $E\{z(x) - y(x) \mid y(x)\} = 0$ , and its variance is  $y$ , i.e.,  $\text{var}\{z(x) - y(x) \mid y(x)\} = \text{var}\{z(x) \mid y(x)\} = y(x)$ . From (1.13), we can also observe an important property of Poisson noise: since the variance is equal to the mean of the signal, there is a square root relation between mean and standard deviation. This implies that the signal-to-noise ratio SNR

$$\text{SNR} = \frac{E\{z(x) \mid y(x)\}}{\sqrt{\text{var}\{z(x) \mid y(x)\}}} = \frac{y(x)}{\sqrt{y(x)}} = \sqrt{y(x)} \quad (1.14)$$

increases when the intensity of the noise-free signal  $y$  increases and converges to zero when  $y$  approaches zero. This means that Poisson images captured at lower intensities, even though in absolute terms feature a lower variance, they are in practice noisier relative to their mean intensity, and when  $y < 1$  they are effectively dominated by noise. Such conditions correspond to what is commonly termed *photon-limited imaging*, which is one of the most challenging imaging scenarios, requiring binning or special denoising procedures (see, e.g., [4]). Although infrequent in the context of consumer imaging and photography, photon-limited imaging is an increasingly important scenario in scientific imaging, particularly in astronomical imaging, fluorescence microscopy, and low-dose radiation imaging for medicinal diagnostics.

### 1.4.2 Scaled Poisson Distribution Family

In many cases, it is convenient to use the so-called *scaled Poisson* distributions, where a positive scaling parameter controls the noise variance relative to the signal mean. Such scaling factor is commonly used to model the *quantum efficiency* of the imaging sensor, i.e., the ratio between the (average) number of converted electrons to the number of incident photons.

A scaled Poisson distribution with scale parameter  $a > 0$  and mean  $y(x) \geq 0$  is of the form

$$a^{-1}z(x) \sim \mathcal{P}(a^{-1}y(x)), \quad (1.15)$$

$$\text{pmf}[z(x) = \zeta \mid y(x)] = \begin{cases} \frac{(y(x)/a)^{\zeta/a} e^{-y(x)/a}}{(\zeta/a)!} & \zeta \in \{0, a, 2a, 3a, \dots\} \\ 0 & \text{elsewhere.} \end{cases} \quad (1.16)$$

The resulting family of distributions can be parametrized by  $\theta = y(x)$ . The mean and variance of the scaled Poisson distributions are

$$\begin{aligned} E\{z(x) \mid y(x)\} &= y(x), \\ \text{var}\{z(x) \mid y(x)\} &= ay(x). \end{aligned} \quad (1.17)$$

Observe that the parameter  $a$  scales only the variance but does not affect the expectation; hence, it controls the relative strength of the noise, and in particular we have

$$\text{SNR} = \frac{\text{E}\{z(x) | y(x)\}}{\sqrt{\text{var}\{z(x) | y(x)\}}} = \sqrt{\frac{y(x)}{a}}. \quad (1.18)$$

### 1.4.3 Poisson–Gaussian Noise

The Poisson–Gaussian noise model is given by the sum of two independent sources of noise: Poisson or scaled Poisson, whose variance is signal-dependent (and proportional to the signal mean) and Gaussian, whose variance is signal-independent. Its formal model is

$$z(x) = ap(x) + n(x), \quad (1.19)$$

where

$$p(x) \sim \mathcal{P}(a^{-1}y(x)) \quad \text{and} \quad n(x) \sim \mathcal{N}(0, b), \quad (1.20)$$

and the constants  $a > 0$  and  $b \geq 0$  are, respectively, the scaling factor for the scaled Poisson addend  $ap$  and the variance of the Gaussian addend  $n$ . We have

$$\text{pdf}[z(x) | y(x)](\zeta) = \sum_{k=0}^{+\infty} \frac{(y(x)/a)^k e^{-y(x)/a}}{k!} \times \frac{1}{\sqrt{2\pi b}} e^{-\frac{(\zeta - ka)^2}{2b}}, \quad (1.21)$$

which also corresponds to a one-parameter family of distributions with parameter  $\theta = y(x)$ . The conditional mean and variance of  $z$  are

$$\begin{aligned} \text{E}\{z(x) | y(x)\} &= y(x) \\ \text{var}\{z(x) | y(x)\} &= ay(x) + b. \end{aligned} \quad (1.22)$$

The *Poisson–Gaussian noise* is formally defined as  $z(x) - y(x)$ , and by (1.22) it has zero-mean and affine variance on  $y$ .

The SNR is calculated as

$$\text{SNR} = \frac{\text{E}\{z(x) | y(x)\}}{\sqrt{\text{var}\{z(x) | y(x)\}}} = \frac{y(x)}{\sqrt{ay(x) + b}}. \quad (1.23)$$

### 1.4.4 Gaussian Approximation of the Poisson Distribution

For large mean values, the Poisson distribution is well approximated by a Gaussian distribution with mean and variance equal to the mean of the Poisson random variable:

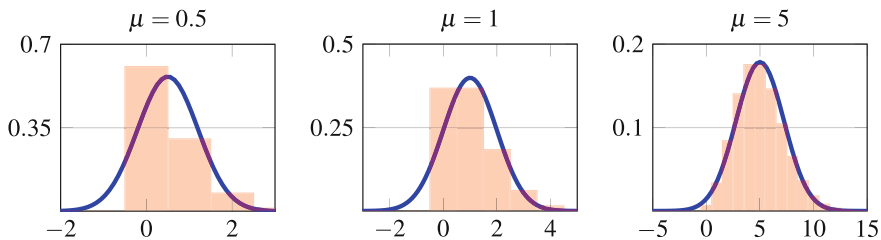
$$\mathcal{P}(\mu) \approx \mathcal{N}(\mu, \mu). \tag{1.24}$$

Here, we derive an intuitive proof based on the CLT and on the fact that the Poisson distributions are closed family with respect to summation of variables.

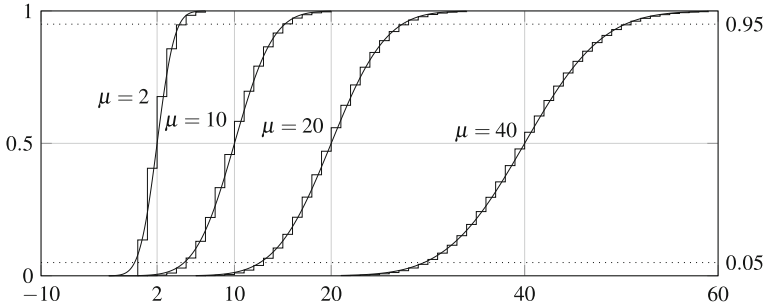
Let us consider two independent Poisson random variables  $X_1$  and  $X_2$ , and their sum  $Y = X_1 + X_2$ . Assuming that  $X_1$  and  $X_2$  have means  $\mu_{X_1}$  and  $\mu_{X_2}$ , respectively, the probability that the sum  $X_1 + X_2$  takes a given value  $\zeta \in \mathbb{N}$  is the sum of the probabilities that  $X_1$  takes value  $i \in \mathbb{N}$  and  $0 \leq i \leq \zeta$  and  $X_2$  takes value  $\zeta - i$  (thus summing up to  $\zeta$ ):

$$\begin{aligned} \text{pmf}[X_1 + X_2 = \zeta] &= \sum_{i=0}^{\zeta} \text{pmf}[X_1 = i] \text{pmf}[X_2 = \zeta - i] = \sum_{i=0}^{\zeta} \frac{\mu_{X_1}^i e^{-\mu_{X_1}}}{i!} \frac{\mu_{X_2}^{\zeta-i} e^{-\mu_{X_2}}}{(\zeta - i)!} \\ &= e^{-(\mu_{X_1} + \mu_{X_2})} \sum_{i=0}^{\zeta} \frac{\mu_{X_1}^i \mu_{X_2}^{\zeta-i}}{i! (\zeta - i)!} = e^{-(\mu_{X_1} + \mu_{X_2})} \frac{1}{\zeta!} \sum_{i=0}^{\zeta} \binom{\zeta}{i} \mu_{X_1}^i \mu_{X_2}^{\zeta-i} \\ &= \frac{(\mu_{X_1} + \mu_{X_2})^\zeta e^{-(\mu_{X_1} + \mu_{X_2})}}{\zeta!}, \end{aligned} \tag{1.25}$$

which shows that  $Y = X_1 + X_2$  is a Poisson random variable with mean and variance  $\mu_{X_1} + \mu_{X_2}$ , i.e.,  $Y \sim \mathcal{P}(\mu_{X_1} + \mu_{X_2})$ . Hence, any Poisson random variable with large enough mean can be expressed by a summation of many Poisson random variables with smaller means. Therefore, according to the CLT, as the mean value increases, the Poisson distribution converges in distribution to a Gaussian distribution with mean and variance equal to the mean of the Poisson random variable. Figure 1.8 shows in red three Poisson distributions with means  $\mu = [0.5, 1, 5]$  overlaid to three Gaussian distributions, blue lines, with means and variances equal to  $\mu$ . We can observe that already for  $\mu = 5$  the Gaussian provides a relatively good approximation of the Poisson distribution. This is further illustrated by the cumulative distribution



**Fig. 1.8** Gaussian approximation of the Poisson distribution. From left to right we draw in red the discrete Poisson distributions  $\mathcal{P}(\mu)$ , and in blue their Gaussian approximations  $\mathcal{N}(\mu, \mu)$  for three different mean values  $\mu = [0.5, 1, 5]$ . Note how the accuracy of the approximation improves as  $\mu$  increases



**Fig. 1.9** Cumulative distribution functions of the Poisson  $\mathcal{P}(\mu)$  and of the Gaussian  $\mathcal{N}(\mu, \mu)$ ,  $\mu = 2, 10, 20, 40$ , showing convergence in distribution for large  $\mu$

functions shown in Fig. 1.9 for  $\mu = 2, 10, 20, 40$ . Most imaging applications deal with Poisson models well above such values and can thus leverage the approximation (1.24).

With the above approximation in mind, in many applications, it is possible to replace the family of Poisson distributions with a family of Gaussian distributions with nonconstant variance that depends on the signal expectation. This approximation in the modeling is appealing as it often results in simplification of analysis and processing operations such as noise estimation and denoising.

#### 1.4.5 Signal-Dependent Heteroskedastic Gaussian Models

As a consequence of (1.24), we can approximate the Poisson–Gaussian model (1.19) by the sum of a deterministic signal  $y(x)$  and two zero-mean Gaussian random variables, one with signal-independent variance  $b$  and one with signal-dependent variance  $ay(x)$ . Since the sum of two zero-mean Gaussian random variables is still a zero-mean Gaussian random variable with variance equal to the sum of the variances, we have

$$z(x) = y(x) + \sigma(y(x))\xi(x), \quad (1.26)$$

where  $\xi(x) \sim \mathcal{N}(0, 1)$  and

$$\sigma(y(x)) = \sqrt{ay(x) + b}, \quad (1.27)$$

$\sigma : \mathbb{R} \rightarrow [0, 1)$  being a univariate<sup>1</sup> function (so-called *standard deviation function*<sup>1</sup>) that gives the signal-dependent standard deviation of the noise as a function of the deterministic noise-free signal  $y(x)$ . Hence,  $z(x) \sim \mathcal{N}(y(x), \sigma^2(y(x)))$ , i.e.,

<sup>1</sup>Throughout the chapter, we use the expressions *standard deviation function* and *standard deviation curve* interchangeably; similarly for the variance we consider the equivalent concepts of *variance function* or *variance curve*.

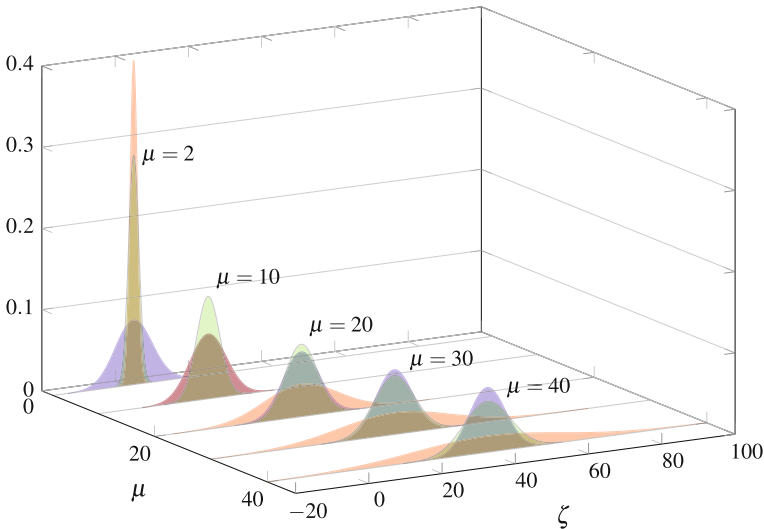
$$\text{pdf} [z(x) | y(x)] (\zeta) = \frac{1}{\sigma(y(x))\sqrt{2\pi}} e^{-\left(\frac{\zeta - y(x)}{\sigma(y(x))}\right)^2}, \quad (1.28)$$

which constitutes a one-parameter family of distributions that depends only on the location parameter  $\theta = y(x)$  that consequently defines the standard deviation of the noise. Trivially,

$$E \{z(x | y(x))\} = y(x) \quad \text{and} \quad \text{std} \{z(x | y(x))\} = \sigma(y(x)). \quad (1.29)$$

It is evident that this model is extremely general and not limited to  $\sigma$  in the affine-variance form (1.27), but can adopt arbitrary nonnegative standard deviation functions. We call these noise models *signal-dependent heteroskedastic Gaussian*, meaning that the variance of the Gaussian noise is not constant and depends directly on the noise-free signal. When  $\sigma$  is fixed and constant, i.e., such as when  $a = 0$  in (1.27), (1.26) trivially reduces to the AWGN model (1.2), where the noise is *signal-independent* and *homoskedastic* (constant variance).

Figure 1.10 illustrates three different families of distributions of the form (1.28): homoskedastic Gaussian distribution with constant  $\sigma \equiv 5$ , i.e.,  $\mathcal{N}(\mu, 5^2)$ ; heteroskedastic Gaussian distributions approximating the Poisson family, i.e.,  $\mathcal{N}(\mu, \mu)$ , where the variance is equal to the mean; and the multiplicative noise with  $\mathcal{N}(\mu, c\mu^2)$ , where the standard deviation is proportional to the mean.



**Fig. 1.10** Homoskedastic Gaussian distributions  $\mathcal{N}(\mu, 5^2)$  (drawn in blue); heteroskedastic Gaussian distributions with affine variance (drawn in green) from the family of distributions  $\mathcal{N}(\mu, \mu)$ ; heteroskedastic multiplicative Gaussian distributions (drawn in red) from the family of distributions  $\mathcal{N}(\mu, (0.5\mu)^2)$ . We show the distributions with means  $\mu = [2, 10, 20, 30, 40]$

To clarify why the third case is multiplicative, we note that if  $\sigma(y(x)) = \sqrt{c}y(x)$  we can rewrite (1.26) as

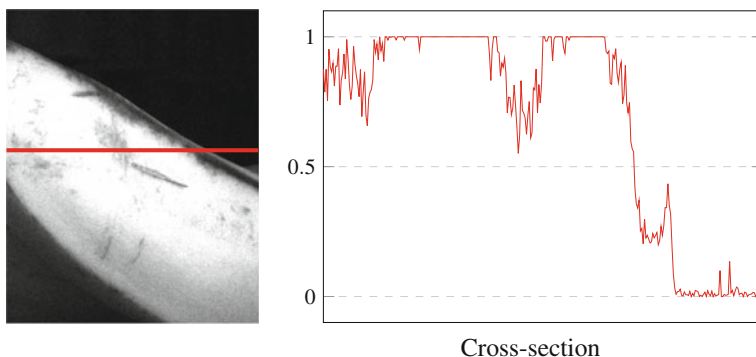
$$z(x) = y(x)\eta(x), \quad \eta(x) \sim \mathcal{N}(1, c). \quad (1.30)$$

#### 1.4.6 Doubly Censored Heteroskedastic Gaussian Noise: A Model for Clipped Noisy Data

All acquisition devices have a finite dynamic range that may not represent the large variation in luminosity in the scene. The device (typically at the analog-to-digital conversion stage) replaces values of intensities that exceed the range with the boundary values; in other words, the captured image can be modeled as

$$\tilde{z} = \max(0, \min(1, z)), \quad (1.31)$$

where the range of the captured image  $\tilde{z}$  is normalized to  $[0, 1]$  and where the image before the min and max operations is denoted by  $z$  and is modeled as in (1.26). This procedure is commonly known as *clipping*. In Fig. 1.11, we show an example of clipped noisy signal: see how the values that exceed 0 and 1 are replaced by these bounds. The assumption that the image range is normalized to  $[0, 1]$  is made, without loss of generality, for the sake of mathematical simplicity. It is clear that the noise statistics of a clipped image are not preserved by the clipping operator; in other words, if for example the acquired image  $z$  is affected by Poisson–Gaussian noise, the statistics of the noise affecting the clipped image  $\tilde{z}$  are not the same. Thus,  $\tilde{z}$  follows a different one-parameter family of distribution and is subject to a different noise



**Fig. 1.11** Example of a clipped noisy signal. Left: clipped detail from one of the images of the sample dataset with highlighted cross section in red. Right: plot of the cross section. Note how the signal is clipped about the boundaries  $[0, 1]$



model than  $z$ . In what follows, we use the tilde decoration to denote variables directly related to clipped observations, following the main development and notation from Foi et al. [13] and Foi [12].

The corresponding noise model for the clipped observations (1.31) is

$$\tilde{z}(x) = \tilde{y}(x) + \tilde{\sigma}(\tilde{y}(x))\tilde{\xi}(x), \quad (1.32)$$

where  $\tilde{y}(x) = \mathbb{E}\{\tilde{z}(x)\}$ ,  $\tilde{\sigma} : \tilde{y} \rightarrow \mathbb{R}^+$  gives the standard deviation of the clipped noisy data as a function of their expectation, i.e.,  $\tilde{\sigma}(\tilde{y}(x)) = \text{std}\{\tilde{z}(x)\}$ , and  $\mathbb{E}\{\tilde{\xi}(x)\} = 0$ ,  $\text{std}\{\tilde{\xi}(x)\} = 1$ . Because of clipping, in general, we have that

$$\tilde{y}(x) = \mathbb{E}\{\tilde{z}(x) \mid y(x)\} \neq \mathbb{E}\{z(x) \mid y(x)\} = y(x), \quad (1.33)$$

$$\tilde{\sigma}(\tilde{y}(x)) = \text{std}\{\tilde{z}(x) \mid y(x)\} \neq \text{std}\{z(x) \mid y(x)\} = \sigma(y(x)), \quad (1.34)$$

and  $y \neq \tilde{y} \subseteq [0, 1]$ . Rewriting (1.32) as

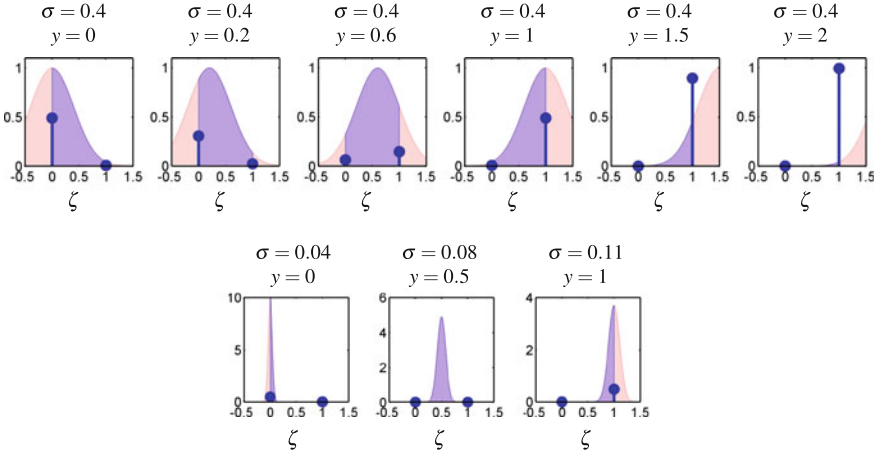
$$\tilde{z}(x) = y(x) + [\tilde{y}(x) - y(x) + \tilde{\sigma}(\tilde{y}(x))\tilde{\xi}(x)],$$

we can see that, with respect to the underlying noise-free signal  $y$ , the clipped observations  $\tilde{z}$  are corrupted by a random error (the term in square brackets) which has nonzero mean. Observe also that, even though  $\text{std}\{\tilde{\xi}(x)\} = \text{std}\{\xi(x)\} = 1$ , the distributions of  $\xi$  and  $\tilde{\xi}$  are different. In particular, assuming  $\xi(x) \sim \mathcal{N}(0, 1)$ , we have that  $\tilde{\xi}(x)$  follows a doubly censored Gaussian distribution [9] supported on  $\left[\frac{-\tilde{y}}{\tilde{\sigma}(\tilde{y})}, \frac{1-\tilde{y}}{\tilde{\sigma}(\tilde{y})}\right]$ .

Figure 1.13 shows an example of the curves  $(y, \sigma(y))$  and  $(\tilde{y}, \tilde{\sigma}(\tilde{y}))$ , for  $\sigma(y) = \sqrt{0.01y + 0.04^2}$ . We emphasize that each curve is drawn in the corresponding expectation/standard deviation Cartesian plane (i.e., we plot the “non-clipped”  $\sigma(y)$  against the  $y, \sigma$  axes and the “clipped”  $\tilde{\sigma}(\tilde{y})$  against the  $\tilde{y}, \tilde{\sigma}$  axes). The figure illustrates the correspondence between points on the two curves given by Eqs. (1.31)–(1.33). Note that the curves from Fig. 1.13 are extremely similar to the scatterplot in Fig. 1.5, hinting that our dataset is effected by clipping.

#### 1.4.6.1 Expectations and Standard Deviations of Clipped Variables and Their Transformations

A crucial point when working with clipped noisy signals is to understand how the variables and functions of the observation model (1.26) relate to those of the clipped observations’ model (1.32). In particular, it is important to compute the functions  $\tilde{y}$  and  $\tilde{\sigma}$  given  $\sigma$  and  $y$ , and vice versa. The PDF of the unobserved non-clipped noisy data  $z \sim \mathcal{N}(y, \sigma^2(y))$  is simply  $\frac{1}{\sigma(y)}\phi\left(\frac{z-y}{\sigma(y)}\right)$  (1.28), whereas the clipped  $\tilde{z} = \max\{0, \min\{z, 1\}\}$  is distributed according to a doubly censored Gaussian distribution having a generalized PDF of the form



**Fig. 1.12** Examples of doubly censored Gaussian distributions drawn in blue, and underlying uncensored Gaussian probability density function (PDF) drawn in red. Top row: the standard deviation  $\sigma$  of the uncensored Gaussian PDF is fixed and equal to 0.04. Bottom row: the standard deviation of the uncensored Gaussian PDF varies according to the function  $\sigma(y) = \sqrt{0.01y + 0.04^2}$ , as illustrated in Fig. 1.13. Compare with the empirical histograms in Fig. 1.6

$$\begin{aligned} \text{pdf}[\tilde{z} = \zeta | y] = & \Phi\left(\frac{-y}{\sigma(y)}\right) \delta_0(\zeta) + \frac{1}{\sigma(y)} \phi\left(\frac{\zeta-y}{\sigma(y)}\right) \chi_{[0,1]} \\ & + \Phi\left(\frac{y-1}{\sigma(y)}\right) \delta_0(1-\zeta), \end{aligned} \quad (1.35)$$

where  $\chi_{[0,1]}$  denotes the characteristic function of the interval  $[0, 1]$  and  $\delta_0$  is the Dirac delta impulse at 0. Here,  $\phi$  and  $\Phi$  are, respectively, the PDF and cumulative distribution function (CDF) of the standard Gaussian  $\mathcal{N}(0, 1)$ . The first and last addends in (1.35) correspond to the probabilities of clipping from below and from above (under- or over-exposure), and are expressed as Dirac deltas with masses equal to the areas of the underlying Gaussian distribution that fall outside the boundaries. In Fig. 1.12, we give some examples of doubly censored Gaussian distributions (drawn in blue) against their uncensored counterparts, i.e., Gaussian PDFs (drawn in red): note how the impulses at the boundaries 0 and 1 have mass (shown as height) that depend on the corresponding censored parts of the Gaussian PDF outside the allowed intensity range. Thus, (1.35) defines a one-parameter family of distributions with parameter  $y$  that also identifies the heteroskedastic Gaussian family of distributions used to build the doubly censored family.

Tedious calculations provide the following exact expressions of the expectation and variance of  $\tilde{z}$  (see, e.g., [15], Chap. 20 or [18]):

$$\begin{aligned} \text{E}\{\tilde{z} | y\} = \tilde{y} = & \Phi\left(\frac{y}{\sigma(y)}\right) y - \Phi\left(\frac{y-1}{\sigma(y)}\right) (y-1) + \\ & \sigma(y) \phi\left(\frac{y}{\sigma(y)}\right) - \sigma(y) \phi\left(\frac{y-1}{\sigma(y)}\right), \end{aligned} \quad (1.36)$$

$$\begin{aligned} \text{var} \{ \tilde{z} \mid y \} = \tilde{\sigma}^2(\tilde{y}) = & \Phi\left(\frac{y}{\sigma(y)}\right) (y^2 - 2\tilde{y}y + \sigma^2(y)) + \tilde{y}^2 - \\ & \Phi\left(\frac{y-1}{\sigma(y)}\right) (y^2 - 2\tilde{y}y + 2\tilde{y} + \sigma^2(y) - 1) + \sigma(y) \quad (1.37) \\ & \phi\left(\frac{y-1}{\sigma(y)}\right) (2\tilde{y} - y - 1) - \sigma(y) \phi\left(\frac{y}{\sigma(y)}\right) (2\tilde{y} - y). \end{aligned}$$

For a given function  $\sigma$ , these expressions explicitly define the two functions

$$\mathcal{A}_\sigma(y) = \tilde{y}, \quad (1.38)$$

$$\mathcal{B}_\sigma(y) = \tilde{\sigma}(\tilde{y}). \quad (1.39)$$

and a mapping

$$(y, \sigma(y)) \mapsto (\tilde{y}, \tilde{\sigma}(\tilde{y})) \quad (1.40)$$

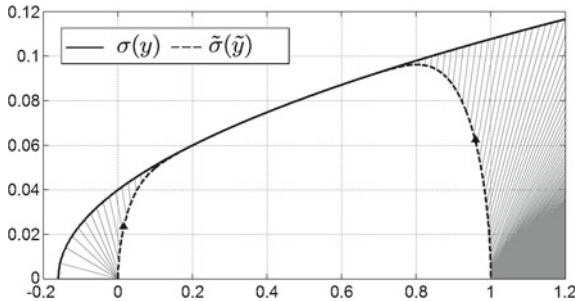
that brings the standard deviation curve  $(y, \sigma(y))$  to its clipped counterpart  $(\tilde{y}, \tilde{\sigma}(\tilde{y}))$ . In Fig. 1.13, we give an example of standard deviation function  $\sigma(y) = \sqrt{0.01y + 0.04^2}$  (solid line) and clipped standard deviation curve  $\tilde{\sigma}(\tilde{y})$  (dashed line). The gray segments are used to illustrate the mapping  $\sigma(y) \mapsto \tilde{\sigma}(\tilde{y})$  (1.40).

The inverse of  $\mathcal{A}_\sigma$  will be formally denoted as

$$\mathcal{C}_\sigma : \tilde{y} \mapsto y = \mathcal{C}_\sigma(\tilde{y}). \quad (1.41)$$

Invertibility requires some hypotheses on the standard deviation function  $\sigma$ ; for instance, it can be shown that (1.41) is well defined provided that  $\sigma(y) = \sqrt{ay + b}$  with  $a > 0$  and  $-\frac{b}{a} \leq \frac{1}{2}$  [12]. The function  $\mathcal{B}_\sigma$  is instead not invertible for the most common types of standard deviation functions, for which  $\tilde{\sigma}(0) = \tilde{\sigma}(1) = 0$  [12].

Although the expressions (1.36) and (1.37) can be eventually useful for a numerical implementation, they are cumbersome and cannot be easily manipulated for further analysis.



**Fig. 1.13** Standard deviation function  $\sigma(y) = \sqrt{0.01y + 0.04^2}$  (solid line) and the corresponding standard deviation curve  $\tilde{\sigma}(\tilde{y})$  (dashed line). The gray segments illustrate the mapping  $\sigma(y) \mapsto \tilde{\sigma}(\tilde{y})$ . The small black triangles  $\blacktriangle$  indicate points  $(\tilde{y}, \tilde{\sigma}(\tilde{y}))$  which correspond to  $y = 0$  and  $y = 1$ . Distributions corresponding to these curves are shown in the bottom row of Fig. 1.12

### 1.4.6.2 Approximation Using Singly Censored Variables

We can simplify the above analysis under the assumption that there are no values of  $y$  for which the clipping from below ( $z < 0$ ,  $\tilde{z} = 0$ ) and clipping from above ( $z > 1$ ,  $\tilde{z} = 1$ ) may both occur with significant probabilities. This assumption means that, for a fixed  $y$ , at most one of the impulses in the PDF (1.35) has mass appreciably larger than 0. In practice, this assumption is always verified under normal capture settings. Exceptions are extreme situations where the noise is dramatically strong with respect to the  $[0, 1]$  range (e.g.,  $\sigma(y) \gg 0.2$  for all  $y \in [0, 1]$ , like in the case illustrated in Fig. 1.12).

Let  $v \sim \mathcal{N}(\mu, 1)$  be a normally distributed random variable with mean  $E\{v\} = \mu$  and unitary variance, and let  $\tilde{v} = \max\{0, v\}$ . Similar to (1.36) and (1.37), it can be shown that the expectation  $E\{\tilde{v}\}$  and the variance  $\text{var}\{\tilde{v}\}$  of the clipped (from below)  $\tilde{v}$  are

$$E\{\tilde{v}\} = \mathcal{E}_m(\mu) = \Phi(\mu)\mu + \phi(\mu), \quad (1.42)$$

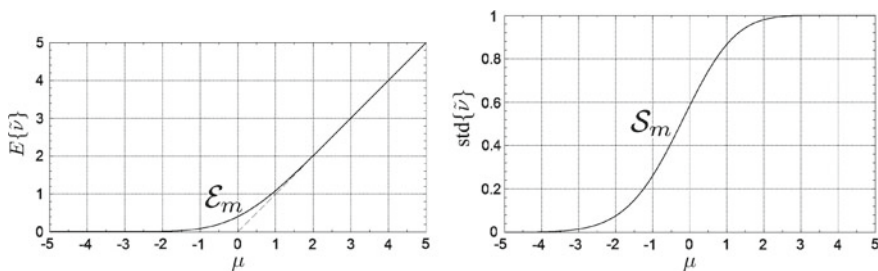
$$\begin{aligned} \text{var}\{\tilde{v}\} &= \mathcal{S}_m^2(\mu) = \Phi(\mu) + \mathcal{E}_m(\mu)\mu - \mathcal{E}_m^2(\mu) = \\ &= \Phi(\mu) + \phi(\mu)\mu - \phi^2(\mu) + \\ &+ \Phi(\mu)\mu(\mu - \Phi(\mu)\mu - 2\phi(\mu)). \end{aligned} \quad (1.43)$$

The plots of the expectation  $E\{\tilde{v}\} = \mathcal{E}_m(\mu)$  and of the standard deviation  $\text{std}\{\tilde{v}\} = \mathcal{S}_m(\mu)$  are shown, as functions of  $\mu$ , in Fig. 1.14. Figure 1.15 combines these two functions and visualizes the mean–standard deviation curve characteristic of standardized clipped variables.

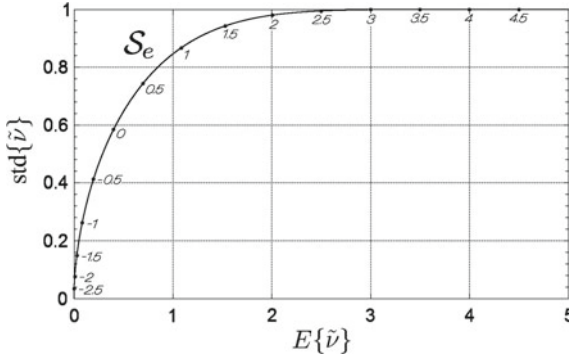
Exploiting these functions, the direct and inverse transformations which link  $\sigma$  and  $y$  to  $\tilde{y}$  and  $\tilde{\sigma}$  can be expressed in the following compact forms [13].

*Direct transformation: obtain  $\tilde{y}$  and  $\tilde{\sigma}$  from  $y$  and  $\sigma$*

Provided that  $y = E\{z\}$  and  $\sigma(y) = \text{std}\{z\}$  from the basic model (1.32) are known, the expectation  $\tilde{y} = E\{\tilde{z}\}$  and the standard deviation  $\tilde{\sigma}(\tilde{y}) = \text{std}\{\tilde{z}\}$  from the observation model (1.32) are obtained as



**Fig. 1.14** Expectation  $E\{\tilde{v}\}$  and standard deviation  $\text{std}\{\tilde{v}\}$  of the clipped  $\tilde{v} = \max\{0, v\}$  as functions  $\mathcal{E}_m$  and  $\mathcal{S}_m$  of  $\mu$ , where  $\mu = E\{v\}$  and  $v \sim \mathcal{N}(\mu, 1)$



**Fig. 1.15** Standard deviation  $\text{std}\{\tilde{v}\}$  of the clipped  $\tilde{v} = \max\{0, v\}$  as function  $\mathcal{S}_e$  of the expectation  $E\{\tilde{v}\}$ . The numbers in italic indicate the corresponding value of  $\mu$ , where  $\mu = E\{v\}$  and  $v \sim \mathcal{N}(\mu, 1)$

$$\begin{aligned} \tilde{y} &= \mathcal{A}_\sigma(y) \approx \mathcal{A}(y, \sigma(y)) = \\ &= \sigma(y) \mathcal{E}_m\left(\frac{y}{\sigma(y)}\right) + 1 - y - \sigma(y) \mathcal{E}_m\left(\frac{1-y}{\sigma(y)}\right), \end{aligned} \quad (1.44)$$

$$\tilde{\sigma}(\tilde{y}) = \mathcal{B}_\sigma(y) \approx \mathcal{B}(y, \sigma(y)) = \sigma(y) \mathcal{S}_m\left(\frac{y}{\sigma(y)}\right) \mathcal{S}_m\left(\frac{1-y}{\sigma(y)}\right). \quad (1.45)$$

Compared to the exact (1.38) and (1.39), the approximate equations (1.44) and (1.45) provide a more intuitive description of the transformations that bring the standard deviation curve  $(y, \sigma(y))$  to its clipped counterpart  $(\tilde{y}, \tilde{\sigma}(\tilde{y}))$ . For instance, provided  $y$  is sufficiently smaller than 1, by observing Fig. 1.14, it is easy to realize that  $\mathcal{E}_m\left(\frac{1-y}{\sigma(y)}\right)$  and  $\mathcal{S}_m\left(\frac{1-y}{\sigma(y)}\right)$  can be substituted by  $\frac{1-y}{\sigma(y)}$  and 1, respectively (the substitution is asymptotically exact). Thus, for describing the clipping from below, (1.44) and (1.45) can be reduced to, respectively,  $\sigma(y) \mathcal{E}_m\left(\frac{y}{\sigma(y)}\right)$  and  $\sigma(y) \mathcal{S}_m\left(\frac{y}{\sigma(y)}\right)$ , which allows to construct the graph of  $(\tilde{y}, \tilde{\sigma}(\tilde{y}))$  in the vicinity of  $(0, 0)$  by simple manipulations of the graphs of  $\mathcal{E}_m$  and  $\mathcal{S}_m$ .

*Inverse transformation: obtain  $y$  from  $\tilde{\sigma}$  and  $\tilde{y}$*

The approximation of (1.41) for calculating the non-clipped  $y$  (1.26) from the clipped  $\tilde{y}$  and  $\tilde{\sigma}(\tilde{y})$  can be given as

$$\begin{aligned} y &= \mathcal{C}_\sigma(\tilde{y}) \approx \mathcal{C}(\tilde{y}, \tilde{\sigma}(\tilde{y})) = \\ &= \tilde{y} \mathcal{E}_r\left(\frac{\tilde{y}}{\tilde{\sigma}(\tilde{y})}\right) - \tilde{y} + 1 - (1 - \tilde{y}) \mathcal{E}_r\left(\frac{1-\tilde{y}}{\tilde{\sigma}(\tilde{y})}\right), \end{aligned} \quad (1.46)$$

where  $\mathcal{E}_r$  is defined implicitly as function of  $\rho = \frac{\mathcal{E}_m(\mu)}{\mathcal{S}_m(\mu)} = \frac{E\{\tilde{v}\}}{\text{std}\{\tilde{v}\}}$  by  $\mathcal{E}_r(\rho) = \frac{\mu}{\mathcal{E}_m(\mu)}$ .

## 1.5 Estimation of the Standard Deviation Curve

The main purpose of noise estimation algorithms is to estimate the standard deviation curve. The most common framework first builds a scatterplot with mean values of the signal on the abscissa, and corresponding standard deviations (variances) on the ordinate (see the scatterplot in Fig. 1.5); then, it fits a parametric curve over these points. When only a single image is used for estimation, it is common practice to compute each scatterpoint from homogeneous samples, i.e., each element in a sample shares a unique common expectation value (hence, they also share a common unique variance value). This practice is based on the fact that the sample variance of homogeneous samples is an unbiased estimator of the noise variance for that particular expectation value. Consequently, each point in the scatterplot has a direct relation to a point on the curve we want to estimate.

Regardless of the estimation algorithm, it is convenient to model each mean–standard deviation pair estimate  $(\hat{y}_i, \hat{\sigma}_i)$  with a bivariate PDF of the form

$$\text{pdf}[(\hat{y}_i, \hat{\sigma}_i) | \tilde{y}_i = \tilde{y}] = \text{pdf}[\hat{y}_i | \tilde{y}_i = \tilde{y}] \text{pdf}[\hat{\sigma}_i | \tilde{y}_i = \tilde{y}], \quad (1.47)$$

where the subscript  $i$  indicates the generic  $i$ -th estimated pair in the scatterplot. Note that the joint probability is reduced to a product of univariate distributions because one can assume that the estimates  $\hat{y}_i$  and  $\hat{\sigma}_i$  are independent [18]. Given the distributions of all the pairs  $\{\hat{y}_i, \hat{\sigma}_i\}_{i=1}^N$ , the posterior likelihood function  $L$  can be calculated as the product of all the densities  $\text{pdf}[(\hat{y}_i, \hat{\sigma}_i) | \tilde{y}_i = \tilde{y}]$  with the prior density  $\text{pdf}[y]$  of  $y$ :

$$L(\boldsymbol{\theta}) = \prod_{i=1}^N \int \text{pdf}[(\hat{y}_i, \hat{\sigma}_i) | \tilde{y}_i = \tilde{y}] \text{pdf}[y] dy, \quad (1.48)$$

where  $\boldsymbol{\theta}$  is an  $m$ -dimensional vector composed by the model parameters to be estimated. The likelihood function  $L$  expresses the joint probability function of the whole collection  $\{\hat{y}_i, \hat{\sigma}_i\}_{i=1}^N$ . The vector  $\boldsymbol{\theta}$  determines univocally the standard deviation curve  $\sigma(y)$ , e.g.,  $\boldsymbol{\theta} = [a, b]$ ,  $m = 2$  in case of the affine mean–variance relation (1.27). The maximization of  $L$  leads to the estimation of the noise curve parameters  $\boldsymbol{\theta}$ . Note that although clipped data is bound to the  $[0, 1]$  interval, assuming that  $\text{pdf}[y]$  is a uniform density supported over  $[0, 1]$  is incorrect, since this prior probability refers to the signal before clipping and this signal can naturally exceed the boundaries of the unit interval, e.g., an overexposed signal  $y$  is typically larger than 1. Therefore, one may assume either a noninformative prior for which all  $y \in \mathbb{R}$  are equiprobable or most commonly a nonnegative signal prior for which all  $y \geq 0$  are equiprobable and  $y < 0$  is impossible, which simplifies (1.48) to

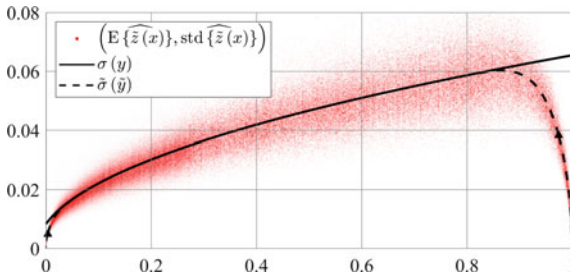
$$L(\boldsymbol{\theta}) = \prod_{i=1}^N \int_0^{\infty} \text{pdf}[(\hat{y}_i, \hat{\sigma}_i) | \tilde{y}_i = \tilde{y}] dy. \quad (1.49)$$

The joint PDF can be, for example, a product of two univariate Gaussian PDFs (see, e.g., [13]), a product of Gaussian-Cauchy mixtures [2], or other PDFs depending on the data and on the method adopted.

Figure 1.16 shows an example of scatterplot fitting through maximization of the likelihood function (1.49): the scatterpoints, drawn as red dots, are the pairs  $(\widehat{E\{\tilde{z}(x)\}}, \widehat{\text{std}\{\tilde{z}(x)\}})$  estimated in (1.8) and (1.9) using our dataset (see the same scatterplot in Fig. 1.5); the estimated lines are drawn in black and have been estimated maximizing (1.49) using the PDFs introduced in (1.10)–(1.11), where the index  $i$  takes the place of the index  $x$ . Note how the estimate of the clipped standard deviation (dashed line) diverges from the non-clipped one (continuous line) when approaching the boundaries  $[0, 1]$ , which follows the divergence between the basic heteroskedastic Gaussian distribution and the proper heteroskedastic singly censored Gaussian (e.g., as illustrated in Fig. 1.12).

Most algorithms in the literature for estimating the variance function  $\sigma^2$  are designed for an affine function of the signal mean, because it provides a reasonably accurate description of the output (without clipping) of imaging sensors commonly found in digital cameras. Nonetheless, some algorithms such as [2, 30], inherently support models of arbitrary order (e.g., a variance function defined as a quadratic or higher-order polynomial of the signal mean) and can be used in scenarios where a higher-order of approximation of the noise variance function is needed, such as the case considered in Sect. 1.7.

In the remainder of this section, we describe relevant approaches for the estimation of the noise standard deviation curve (or equivalently the variance curve) under the assumption of white noise. While the problem of estimating the correlation (power spectral density) of colored noise is investigated in Sect. 1.6.3.2, the majority of the methods below can be extended, with due modifications, to the more generic case of the estimation of the standard deviation curve of signal-dependent colored noise. Note that our overview is without any pretension of completeness, with the only goal of briefly introducing the fundamentals behind the most popular approaches for noise estimation.



**Fig. 1.16** Scatterplot of the pairs  $(\widehat{E\{\tilde{z}(x)\}}, \widehat{\text{std}\{\tilde{z}(x)\}})$  drawn as red dots, and estimated clipped (black dashed line) and non-clipped (black continuous line) noise standard deviation curves. The small black triangles  $\blacktriangle$  indicate points  $(\tilde{y}, \tilde{\sigma}(\tilde{y}))$  which correspond to  $y = 0$  and  $y = 1$

### 1.5.1 Patch-Based Methods

A prominent *patch-based* algorithm is introduced by Lee and Hoppel [20]. The algorithm divides the image into small patches and computes their mean and variance to build the scatterplot of mean–variance pairs. Since image blocks might contain heterogeneous elements that would mislead the estimation of the local variances, the authors estimate the noise parameters by finding the curve that intersects most of the scatterpoints. In this way, they reduce the effect of outliers that usually appear far from the majority of the scatterpoints. In a similar work, Amer and Dubois [1] evaluate, using directional derivative filters, the uniformity of each patch that generated a data point. Comparing the uniformity against a threshold, the algorithm decides whether to use the scatterpoint (if the patch elements are homogeneous) or to discard it. Finally, since the outliers have been already excluded, a simple least square (LS) fitting is adopted. In [30], the authors divide the image into nonoverlapping blocks; based on the Kendall’s  $\tau$  coefficients, adopted to find the correlation between elements from the same block, the blocks are then classified as homogeneous or heterogeneous. The heterogeneous blocks are discarded, while the homogeneous ones are used to compute the local statistics for the fitting of the noise variance curve. An important aspect of the algorithm is that a robust fitting is performed by minimizing the  $\ell_1$  error of the residuals. Similar works can be found in [19, 24, 25]; however, we decide not to go into further details.

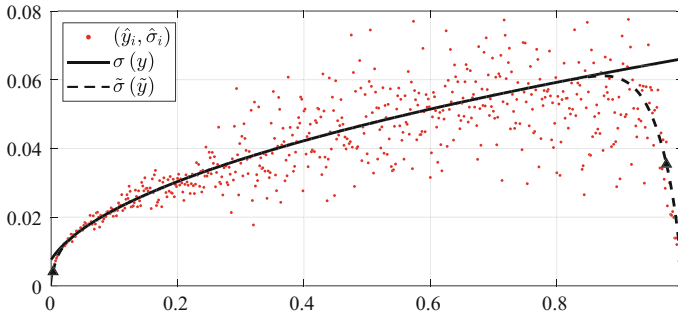
An interesting variant has been proposed by Boulanger et al. [8]. They divide the image into adaptive blocks whose size depends on the variance of their elements (homogeneity). If the variance of a block matches the variance model (Fisher test is used to compare the two), then the block is considered homogeneous, otherwise the block is further split into four parts and each *subblock* is then analyzed as before. Finally, the authors perform noise parameters estimation via robust linear regression of the local estimates.

### 1.5.2 Segmentation-Based Methods

We now describe the most relevant *segmentation-based* approaches for noise estimation. Gravel et al. [14] segment the observed noisy image into homogeneous samples that are each used to compute a scatterpoint. The segmentation is performed by first smoothing the observed image, and then by grouping pixels with similar intensity. This leverages the fact that a smoothing operator suppresses the noise and facilitates the segmentation process. Pixels from edges and texture are excluded from the estimation, since the segmentation is inaccurate in those regions. The noise parameters are finally estimated using a weighted regression of the scatterplot points.

Another type of segmentation is proposed in [21], where the authors do not filter the image, but they bin the image elements using a K-means clustering method. A robust fitting algorithm is then adopted to cope with possible inaccuracies of the





**Fig. 1.17** Noise standard deviation (black curve)  $\sigma(y)$  estimated from one image from the dataset. We show also the estimate of the clipped standard deviation (dashed curve)  $\tilde{\sigma}(\hat{y})$  and the scatterplot used for the fitting. For the estimation, we used the algorithm [2], and the estimated parameters are  $a = 4.315 \times 10^{-3}$  and  $b = 5.814 \times 10^{-5}$

K-means clustering: the noise parameters are estimated by fitting a lower envelope of the scatterplot, computed by maximizing a likelihood function (see (1.49)) that takes into consideration the possible overestimation of the scatterpoints variances. Similarly, Foi et al. [13] filter the observed image, segment it, and then maximize a likelihood function to estimate the noise parameters. A major novelty introduced by this work is that it takes under consideration the clipping of the data. Figure 1.17 shows the estimation results of the algorithm [13] applied to an image of our dataset. Note how, although the number of scatterpoints used for the likelihood maximization is much smaller compared to the scatterpoints in Fig. 1.16, the estimated curves are quite accurate. The algorithm deals with clipped data using the mappings (1.45), (1.44), and (1.46), and estimates a parametric model of the noise affecting the non-clipped signal from the clipped noisy observation, simultaneously providing the standard deviation functions  $\tilde{\sigma}$  for the clipped noise and  $\sigma$  for the underlying data before clipping.

### 1.5.3 Alternative Approaches

In [3], it is shown that the use of homogeneous samples is not a requirement to estimate the affine-variance model (1.27), and thus the noise parameters may be estimated without any image segmentation and by leveraging instead robust filters.

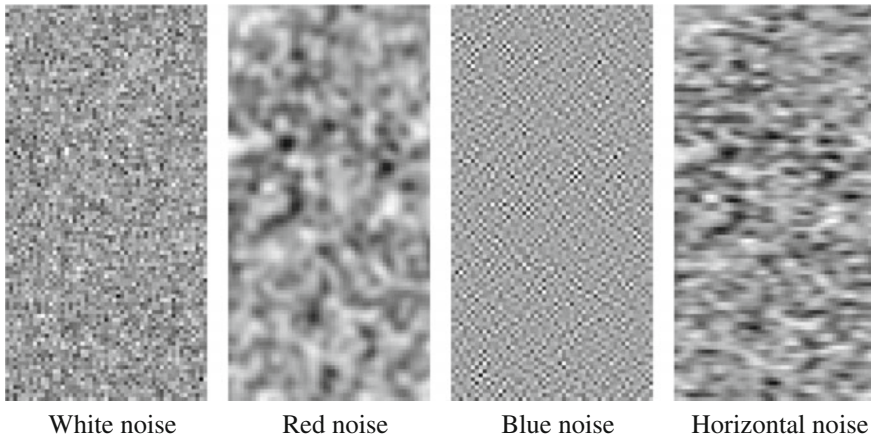
The algorithm described in [22] estimates the noise by exploiting the variance stabilization achieved by the generalized Anscombe transformation (GAT). The GAT is a nonlinear univariate transformation that converts signal-dependent Poisson–Gaussian noise into approximately additive and signal-independent noise with unitary variance. The GAT is defined based on the same parameters  $a$  and  $b$  adopted to describe the Poisson–Gaussian noise. This approach for noise estimation is based on the observation that the variance stabilization is not accurate when the GAT

parameters do not match the ones from the noise model. Hence, the noise parameters are obtained by finding the GAT parameters that yield the most accurate variance stabilization.

Finally, the algorithm in [28] estimates the noise parameters by analyzing the last eigenvalues of the singular value decomposition (SVD) of homogeneous patches from a noisy image. These eigenvalues capture noise and virtually no signal, and are therefore used to estimate accurately the noise parameters.

## 1.6 Correlated Noise

So far we have described noise models characterized by a flat power spectral density, i.e., various types of white noise. According to those models, the noise affecting different pixels is uncorrelated. However, in many practical applications, there could be correlation in the errors: this correlation might be due to the physics of the acquisition system, to the sensor's readout process, to cross-talk between neighboring pixels, or due to processing performed on the raw image after the acquisition. These types of acquisition errors can be modeled by the so-called *colored* noise models, which assume a stationary spatial correlation among noise realizations, as illustrated by the examples in Fig. 1.18, and that can be characterized by a nonconstant power spectral density of the noise. In this section, we discuss models for correlated noise, starting from the simpler case of signal-independent stationary colored noise and then introducing two forms of signal-dependent colored noise.



**Fig. 1.18** White noise versus three examples of colored noise

### 1.6.1 Stationary Correlated Noise

The generic model for a noisy image  $z$  corrupted by a zero-mean stationary additive correlated noise  $\eta$  is

$$z(x) = y(x) + \eta(x), \quad \eta(x) = (\nu \circledast g)(x), \quad (1.50)$$

where  $x \in \Omega \subset \mathbb{Z}^2$ ,  $y$  is a deterministic noise-free image,  $\nu$  is zero-mean independent noise with unit variance, and  $g$  is a convolution kernel that determines the variance and the spatial correlation of the noise  $\eta$ . Specifically, the variance and the power spectrum (PSD) (1.4) of  $\eta$  are

$$\text{var}\{\eta(x)\} = \|g\|_2^2, \quad \Psi = \text{var}\{\mathcal{F}[\eta]\} = |\mathcal{F}[g]|^2 |\Omega|, \quad (1.51)$$

and by Parseval's isometry we have

$$\text{var}\{\eta(x)\} = |\Omega|^{-1} \|\mathcal{F}[g]\|_2^2 = |\Omega|^{-2} \|\Psi\|_1. \quad (1.52)$$

For the sake of simplicity, we assume a circulant convolution in order to leverage the convolution theorem. Model (1.50) reduces to the AWGN model (1.2) when  $g$  is a scaled Dirac impulse with mass  $\sigma$  and  $\nu$  is Gaussian.

We say that the noise is *colored* when the noise power spectrum is markedly not flat. Whenever there is a dominant spectral band characterized by a significantly larger noise variance, the spectral position of this band determines the “color” of the noise; thus, noise predominately affecting the low frequencies is often called “red noise”, as opposed to a “blue noise” which is mostly localized on the high frequencies. Figure 1.18 compares white noise with three examples of correlated noise. Different types of spatial correlation can be appreciated, with the noise affecting a pixel influenced by the surrounding noise realizations.

### 1.6.2 Correlated Signal-Dependent Noise model

Depending on the physics and hardware of the acquisition process, noise can feature both correlation and signal-dependent characteristics. However, a PSD and a variance function cannot be defined exactly within the same model, because their underlying generative processes are incompatible with each other. The two models presented in the next subsections show two extremes of a compromise: the first model ignores the means of neighboring pixels and thus the variance is defined exactly at the expense of the PSD model, which is only approximate; in the second model the PSD is expressed exactly while the variance function is approximate. In intermediate cases, neither the PSD nor the variance function may be defined exactly. Although formally quite different, the model discrepancies are shown to be typically small in smooth areas of the image, making these approximate models useful and constructive [5, 7].

### 1.6.2.1 Noise Scaling Post Correlation

The first model assumes that the signal-dependent part of the noise acts as a deterministic scaling term to a stationary correlated noise:

$$\begin{aligned} z(x) &= y(x) + \sigma(y(x)) \eta(x), \\ \eta &= \nu \otimes g, \quad \nu(\cdot) \sim \mathcal{N}(0, 1), \quad \sigma : y \rightarrow \mathbf{R}^+, \end{aligned} \quad (1.53)$$

where  $\sigma$  is a generic standard deviation function. The expectation and variance of  $z$  are, respectively,

$$\mathbf{E}\{z\} = y, \quad (1.54)$$

$$\begin{aligned} \text{var}\{z\} &= \text{var}\{\sigma(y) \nu \otimes g\} = \sigma^2(y) \text{var}\{\nu \otimes g\} = \\ &= \sigma^2(y) \|g\|_2^2 = \sigma^2(\mathbf{E}\{z\}) \|g\|_2^2. \end{aligned} \quad (1.55)$$

With regard to the PSD, it can be approximated as

$$\text{var}\{\mathcal{F}[z]\} \approx |\mathcal{F}[g]|^2 \|\sigma^2(y)\|_1. \quad (1.56)$$

Roughly speaking, (1.53) describes a physical process where the correlating process takes place before the signal-dependent amplification of errors.

### 1.6.2.2 Noise Scaling Prior to Correlation

The second model considers instead a case where the correlating process operates on a ready signal-dependent white noise model:

$$z'(x) = y(x) + \sigma(y(x)) \nu(x), \quad (1.57)$$

$$z(x) = (z' \otimes g)(x). \quad (1.58)$$

The expected value and variance of  $z$  are, respectively,

$$\mathbf{E}\{z\} \approx \mathbf{E}\{z'\} \|g\|_1 = y \|g\|_1, \quad (1.59)$$

$$\text{var}\{z\} \approx \text{var}\{z'\} \|g\|_2^2 = \sigma^2(y(x)) \|g\|_2^2, \quad (1.60)$$

where the approximations become accurate in large smooth areas of the image where the intensity changes gradually. Combining (1.59) and (1.60), we obtain

$$\text{var}\{z\} \approx \sigma^2\left(\frac{\mathbf{E}\{z\}}{\|g\|_1}\right) \|g\|_2^2. \quad (1.61)$$

The PSD for (1.58) is

$$\text{var} \{ \mathcal{F} [z] \} = |\mathcal{F} [g]|^2 \|\sigma^2(y)\|_1 \approx |\mathcal{F} [g]|^2 \left\| \sigma^2 \left( \frac{\mathbb{E} \{z\}}{\|g\|_1} \right) \right\|_1. \quad (1.62)$$

Thus, both Model 1 and Model 2 express the variance of  $z$  as a function of its expectation, where the main differences consist merely in a scaling of the variables, and this scaling is determined by the  $\ell_1$  and  $\ell_2$  norms of the convolution kernel  $g$ . Therefore, macroscopically, the two models are comparable.

Before we proceed, let us observe that the degree of approximation in (1.59)–(1.60) can be quantified by expanding the expression of  $z(x_0)$  at the generic coordinate  $x_0$ :

$$z(x_0) = [z' \circledast g](x_0) = \sum_{x \in \Omega} z'(x_0 - x) g(x). \quad (1.63)$$

The expected value and variance of the generic pixel  $z(x_0)$  can be thus calculated, respectively, as

$$\begin{aligned} \mathbb{E} \{z\}(x_0) &= \sum_{x \in \Omega} \mathbb{E} \{z'(x_0 - x)\} g(x) = \sum_{x \in \Omega} y(x_0 - x) g(x) = \\ &= \sum_{x \in \Omega} \sum_{k=0}^{+\infty} \frac{\partial^k y(x_0) x^k}{(-1)^k k!} g(x) = \sum_{k=0}^{+\infty} \frac{\partial^k y(x_0)}{(-1)^k k!} \sum_{x \in \Omega} x^k g(x), \end{aligned} \quad (1.64)$$

$$\begin{aligned} \text{var} \{z\}(x_0) &= \sum_{x \in \Omega} \text{var} \{z'(x_0 - x)\} g(x) = \sum_{x \in \Omega} \sigma^2(x_0 - x) g(x) = \\ &= \sum_{x \in \Omega} \sum_{k=0}^{+\infty} \frac{\partial^k (\sigma^2 \circ y)(x_0) x^k}{(-1)^k k!} (g(x))^2 = \\ &= \sum_{k=0}^{+\infty} \frac{\partial^k (\sigma^2 \circ y)(x_0)}{(-1)^k k!} \sum_{x \in \Omega} x^k (g(x))^2, \end{aligned} \quad (1.65)$$

where the final expressions of (1.64)–(1.65) come from the Maclaurin expansions of  $y(x_0 - x)$  and  $\sigma^2(x_0 - x)$  at the generic coordinate  $x_0 \in \Omega$ , and  $\partial^k (\sigma^2 \circ y)(x)$  is the  $k$ th derivative of the composite function  $\sigma^2(y(x))$ , and it can be computed, for example, with the formula of Faà Di Bruno [11].

For approximation of order zero, i.e., when  $k = 0$ , (1.64)–(1.65) reduce to (1.59)–(1.60). Thus, in regions where  $y$  is constant, the approximations are always valid. If we assume  $g$  even symmetric, both  $\sum_{x \in \Omega} x^k g(x)$  and  $\sum_{x \in \Omega} x^k (g(x))^2$  vanish when  $k$  is odd since  $x^k$  is odd symmetric. Therefore, for even symmetric kernels  $g$ , the approximations (1.59)–(1.60) differ from (1.64)–(1.65) only for approximation terms of  $y(x_0 - \cdot)$  and  $\sigma^2(x_0 - \cdot)$  of even order 2 or greater.

In case of affine noise variance (1.22), i.e.  $\sigma^2(y(x)) = ay(x) + b$ , (1.65) becomes

$$\begin{aligned} \text{var}\{z\}(x_0) &= (ay(x_0) + b) \sum_{x \in \Omega} (g(x))^2 - \\ &\quad a (\partial y)(x_0) \sum_{x \in \Omega} x^k (g(x))^2 + \\ &\quad a \sum_{k=2}^{+\infty} \frac{(\partial^k y)(x_0)}{(-1)^k k!} \sum_{x \in \Omega} x^k (g(x))^2 . \end{aligned} \quad (1.66)$$

### 1.6.3 Estimation

#### 1.6.3.1 Estimation of the Noise Standard Deviation Function

The methods for estimating the standard deviation curve which were briefly reviewed in Sect. 1.5 are formally designed for independently distributed noise, i.e., white noise. If these methods are applied to correlated noise, they may fail to estimate correctly the curve. In this regard, we note that many of these methods (e.g., [3, 13, 14, 22, 28]) employ some kind of high-pass filtering to reduce the impact of the signal  $y$  on the estimation of noise variance. Whereas white noise affects in equal manner different frequency bands, correlated noise is distributed unevenly over the frequency spectrum: without prior knowledge of the noise power spectrum, there is uncertainty about the proportion of noise that is effectively maintained after high-pass filtering, hence a potential risk of significant overestimation or underestimation of the curve. To understand this phenomenon, let us consider the observation model (1.53) and assume that scatterpoints are obtained from estimating the variance of the output of filtering  $z$  by a high-pass kernel  $h$ . Restricting the estimation over a sufficiently large region where  $y$  is homogeneous, we can treat  $\sigma(y)$  as a constant and we have

$$\begin{aligned} \text{var}\{z \circledast h|y\} &= \text{var}\{\sigma(y) \eta \circledast h\} = \sigma^2(y) \text{var}\{v \circledast g \circledast h\} = \\ &= \sigma^2(y) \|g \circledast h\|_2^2 = \sigma^2(y) |\Omega|^{-1} \|\mathcal{F}[g \circledast h]\|_2^2 = \\ &= \sigma^2(y) |\Omega|^{-1} \|\mathcal{F}[g]\|_2 \|\mathcal{F}[h]\|_2^2 = \\ &= \sigma^2(y) |\Omega|^{-2} \|\Psi \circledast [h]\|_1^2 . \end{aligned} \quad (1.67)$$

Thus, even when  $\|h\|_2 = 1$ , it may happen that  $\text{var}\{z \circledast h|y\} \neq \text{var}\{z|y\} = \sigma^2(y) |\Omega|^{-2} \|\Psi\|_1$ , with various outcomes depending on how  $\mathcal{F}[h]$  correlates with the PSD  $\Psi$ . In particular,

$$\text{var}\{z \circledast h|y\} = \text{var}\{z|y\} \|\Psi \circledast [h]\|_1^2 \|\Psi\|_1^{-1} . \quad (1.68)$$

This means that if all scatterpoints are estimated using a unique filter  $h$ , we can estimate the noise variance function modulo a scaling factor that depends on the PSD  $\Psi$ . As we show next, the fact that this scaling factor is unknown does not prevent estimating the PSD  $\Psi$ , from which we can then resolve (1.68) and obtain an estimate of the proper noise variance function.

### 1.6.3.2 PSD Estimation

The typical approach to estimating the noise PSD consists of processing the standardized noise with a filter bank of band-pass filters, producing a collection of subband images and by calculating the sample variance (or a robust estimate of the variance) of the noise in these subbands. Modulo a, possibly unknown, scaling factor  $\lambda$ , standardized noise can be represented in the form

$$\check{\eta} = \lambda \nu \otimes \check{g}, \quad (1.69)$$

where  $\check{g} = g \|g\|_2^{-1}$ ,  $\nu$  is zero-mean independent noise with unit variance, and hence  $\check{\eta}$  has zero-mean and variance  $\lambda^2$ .

The signal-independent colored noise model (1.50) already has the form (1.69) with  $\check{\eta} = \eta$  and  $\lambda = \|g\|_2$ . Signal-dependent colored noise of the forms (1.53) can be reduced to (1.69) in several ways, e.g., by standardization of the samples with respect to the noise model fitted to the mean, or by variance stabilizing transformations [22, 29]. If the standardization is obtained via a noise variance function estimated modulo a scaling factor as in (1.68), then we arrive at (1.69) with  $\lambda^2 = \|\Psi | \mathcal{F}[h]|^2\|_1^{-1} \|\Psi\|_1$ .

Let us consider a set of band-pass filters  $\mathcal{H} = \{h_1, \dots, h_J\}$ . The generic output from the  $j$ th filter is

$$\check{\eta} \otimes h_j = (\lambda \nu \otimes \check{g}) \otimes h_j, \quad (1.70)$$

and its variance

$$\text{var} \{ \check{\eta} \otimes h_j \} = \lambda^2 \text{var} \{ \nu \otimes \check{g} \otimes h_j \} = \lambda^2 \text{var} \{ \nu \} \| \check{g} \otimes h_j \|_2^2 = \lambda^2 \| \check{g} \otimes h_j \|_2^2. \quad (1.71)$$

By Parseval's isometry and (1.51), this variance takes the form

$$\text{var} \{ \check{\eta} \otimes h_j \} = \lambda^2 |\Omega|^{-2} \left\| | \mathcal{F}[h_j] |^2 \check{\Psi} \right\|_1, \quad (1.72)$$

where  $|\Omega| = N_1 N_2$  denotes the cardinality of  $\Omega$  and  $\check{\Psi}$  is a normalized PSD linked to the normalization of the kernel  $\check{g}$ ,

$$\check{\Psi} = \Psi \|g\|_2^{-1} = |\Omega|^2 \Psi \| \Psi \|_1^{-1} \quad \text{and} \quad \left\| \check{\Psi} \right\|_1 = |\Omega|^2. \quad (1.73)$$

Hence,  $\|\lambda^2 \check{\Psi}\|_1 = \lambda^2 |\Omega|^2$ , which shows that estimating  $\lambda^2 \check{\Psi}$  trivially yields also separate  $\lambda = |\Omega|^{-1} \sqrt{\|\lambda^2 \check{\Psi}\|_1}$  and consequently  $\check{\Psi}$ .

Therefore, the problem at hand amounts to estimating  $\lambda^2 \check{\Psi}$  from  $\text{var}\{\check{\eta} \otimes h_j\}$ ,  $j = 1, \dots, J$ . To this end, it is convenient to rewrite (1.72) in matrix notation as

$$\mathbf{V} = \mathbf{F}^T \mathbf{P}, \quad (1.74)$$

where

$$\mathbf{V} = \begin{bmatrix} \text{var}\{z_1\} \\ \vdots \\ \text{var}\{z_J\} \end{bmatrix}, \quad \mathbf{P} = \lambda^2 \begin{bmatrix} \check{\Psi}(1) \\ \vdots \\ \check{\Psi}(|\Omega|) \end{bmatrix} \quad (1.75)$$

$$\mathbf{F} = \frac{1}{|\Omega|^2} \begin{bmatrix} |\mathcal{F}[h_1](1)|^2 & \dots & |\mathcal{F}[h_J](1)|^2 \\ \vdots & \ddots & \vdots \\ |\mathcal{F}[h_1](|\Omega|)|^2 & \dots & |\mathcal{F}[h_J](|\Omega|)|^2 \end{bmatrix}. \quad (1.76)$$

Typically,  $J < |\Omega|$ , making the system (1.74) under-determined. The unconstrained minimum-norm estimate of the PSD  $\lambda^2 \check{\Psi}$  can be computed as

$$\widehat{\mathbf{P}} = \mathbf{F} \left( \mathbf{F}^T \mathbf{F} \right)^\dagger \mathbf{V}, \quad (1.77)$$

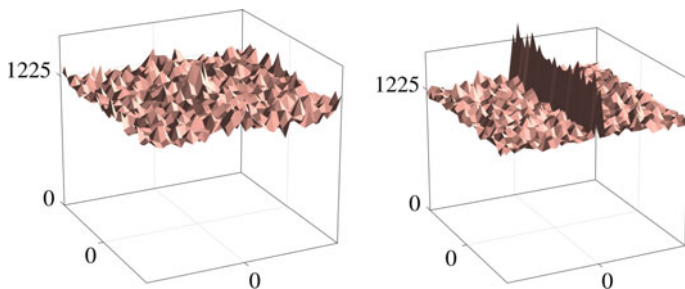
where  $\dagger$  denotes the pseudoinverse. A typical constraint that must be enforced onto  $\widehat{\mathbf{P}}$  is its nonnegativity, since this is not automatically guaranteed by the pseudoinverse. An iterative system such as the following one can be used to this end:

$$\begin{aligned} \mathbf{R}_k &= \mathbf{V} - \mathbf{F}^T \widehat{\mathbf{P}}_k, \\ \widehat{\mathbf{P}}_{k+1} &= \left| \widehat{\mathbf{P}}_k + \mathbf{F} \left( \mathbf{F}^T \mathbf{F} \right)^\dagger \mathbf{R}_k \right|. \end{aligned} \quad (1.78)$$

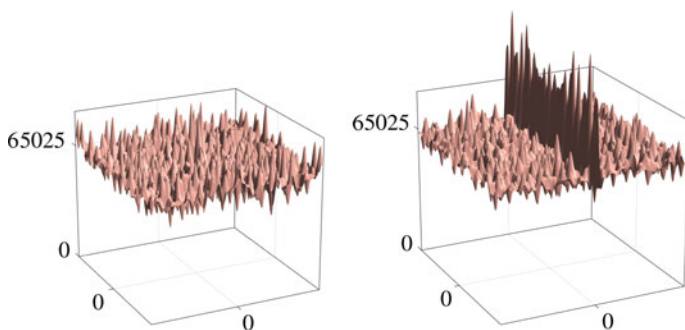
This system can be modified so to enforce further constraints or priors on  $\widehat{\mathbf{P}}_{k+1}$ , such as its smoothness or sparsity with respect to some representation [10], or the symmetries of the Fourier-domain PSDs.

As bank of filters, one can use the Fourier transform of a smaller size, block transforms such as the discrete cosine transform, wavelets, etc. Figure 1.19 shows two examples of the variances (1.71) with respect to the Fourier transform of size  $35 \times 35$ , one estimated for a Samsung S5K2L2 CMOS sensor from the dataset of Sect. 1.3 and one for a Foveon X3 sensor at ISO 6400 of a Sigma DP1 Merrill camera. While the latter PSD shows a nonuniformity representing colored noise, the former





**Fig. 1.19**  $35 \times 35$  Fourier-domain PSD estimated for a Samsung S5K2L2 CMOS sensor from the dataset of Sect. 1.3 (left) and for a Foveon X3 sensor (right). The former is practically flat (thus representing white noise), whereas the latter shows a mild nonuniformity characteristic of either row or column striping noise



**Fig. 1.20** Estimates of the  $255 \times 255$  Fourier-domain PSDs computed through the iterative system (1.78) from the  $35 \times 35$  measurements shown in Fig. 1.19

is virtually flat (i.e., white noise), with random fluctuations merely due to finite sample set. Larger size PSDs, estimated via the recursive method (1.78), are shown in Fig. 1.20. Both in Fig. 1.19 and in Fig. 1.20, we indicate the power level of an ideal flat spectrum of i.i.d. standard white noise (i.e.,  $35^2 = 1225$  and  $255^2 = 65,025$  for a  $35 \times 35$  and  $255 \times 255$  Fourier transform, respectively, as can be computed from (1.7) or (1.51)). The PSDs estimated from the Samsung sensor accurately match this level, since they are approximately flat and were estimated from standardized errors with  $\lambda = 1$  (1.69); the match is very good also for the Foveon X3 PSDs, since the nonuniformity is anyway quite mild and localized along one line.

## 1.7 Photo-Response Nonuniformity

When a pixel array is exposed to a uniform light source, every pixel element is expected to yield the same underlying signal. However, physical imperfections of the detector elements, such as slight discrepancies in the pixel size and substrate

material, cause a deviation from the expected true signal output. This deviation is defined as the photo-response nonuniformity (PRNU).

In some cases, the PRNU represents a relevant portion of the image degradation, and thus the image degradation model presented in (1.26) is no longer adequate. The variable  $y$  previously defined as the true signal becomes a random variable dependent on the detector element. Let us define this new variable as  $u$ . Then,

$$\begin{aligned} z(x) &= u(x) + \sigma(u(x)) \xi(x), \\ u(x) &= \mathcal{N}(y(x), cy^2(x)). \end{aligned} \quad (1.79)$$

Above we model the physical discrepancies of the detector elements as a Gaussian distribution, with variance depending on the underlying signal  $y$  and on a scaling factor  $c$ . Although (1.79) describes the nonuniformity as a random process,  $u$  is a systematic error which is identically realized at every acquisition (i.e., at each frame), as opposed to  $\xi$  which changes randomly at every pixel and at every new capture.

Note that the errors caused by the PRNU are multiplicative, following the same model as described in (1.30). The univariate standard deviation function described by (1.27) becomes

$$\text{std}\{z \mid y\} = \sqrt{cy^2 + ay + b}, \quad (1.80)$$

which is a consequence of treating  $z$  as a mixture variate and of  $\sigma^2$  being affine. Here,  $a$ ,  $b$ , and  $c$  can be estimated using the methods detailed in Sects. 1.5.1 and 1.5.2 of this chapter, but considering a second-order polynomial for the noise variance curve estimation.

In general, the PRNU is more evident in images with low gain  $a$  or with large exposure time, i.e., large  $y(x)$ .

## 1.8 Conclusions

All acquisition systems are affected to some degree by noise. To successfully analyze and process any acquired noisy data, it is necessary to adopt an adequate noise model. This chapter presented several basic noise models that can be used within various imaging applications. First, we introduced white noise models characterized by a flat noise power spectrum. White noise models include the classic signal-independent AWGN noise, as well as various signal-dependent noise models, which are formalized by means of one-parameter families of distributions. Then, we introduced colored noise models, in which the noise power spectral density is not flat, implying that the noise affecting a pixel is correlated with the noise affecting neighboring pixels. An overview of the leading approaches for estimating the noise parameters sufficient to characterize the model was also provided. Overall, these models and methods are suitable for processing imagery from a wide range of acquisition devices, such as

digital consumer cameras, X-ray systems, microscopes, telescopes, etc. They especially play a fundamental role in processing pipelines for image restoration (see, e.g., [4–7, 12]).

**Acknowledgement** Figures 1.13, 1.14, and 1.15 are reprinted from Refs. [12, 13], with the permission of Elsevier and IEEE. This work was supported by the Academy of Finland (project no. 310779) and by the European Union’s Seventh Framework Programme (FP7-PEOPLE-2013-ITN, project no. 607290 SpaRTaN).

## References

1. Amer A, Dubois E (2005) Fast and reliable structure-oriented video noise estimation. *IEEE Trans Circuits Syst Video Technol* 15(1):113–118
2. Azzari L, Foi A (2014a) Gaussian-Cauchy mixture modeling for robust signal-dependent noise estimation. In: 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp 5357–5361
3. Azzari L, Foi A (2014b) Indirect estimation of signal-dependent noise with nonadaptive heterogeneous samples. *IEEE Trans Image Process* 23(8):3459–3467
4. Azzari L, Foi A (2016) Variance stabilization for noisy+estimate combination in iterative Poisson denoising. *IEEE Signal Process Lett* 23(8):1086–1090
5. Azzari L, Foi A (2017) Variance stabilization in Poisson image deblurring. In: 2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017). IEEE, pp 728–731
6. Borges LR, Azzari L, Bakic PR, Maidment AD, Vieira MA, Foi A (2018) Restoration of low-dose digital breast tomosynthesis. *Meas Sci Technol* 29(6):064003
7. Borges LR, Guerrero I, Bakic PR, Foi A, Maidment ADA, Vieira MAC (2017) Method for simulating dose reduction in digital breast tomosynthesis. *IEEE Trans Med Imaging* 36(11):2331–2342
8. Boulanger J, Kervrann C, Bouthemy P, Elbau P, Sibarita J-B, Salamero J (2010) Patch-based nonlocal functional for denoising fluorescence microscopy image sequences. *IEEE Trans Med Imaging* 29(2):442–454
9. Cohen CA (1991) Truncated and censored samples. CRC Press, Boca Raton
10. Daubechies I, Defrise M, De Mol C (2004) An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Commun Pure Appl Math* 57(11):1413–1457
11. Faà Di Bruno F (1857) Note sur une nouvelle formule de calcul différentiel. *Q J Pure Appl Math* 1:359–360
12. Foi A (2009) Clipped noisy images: heteroskedastic modeling and practical denoising. *Signal Process* 89(12):2609–2629
13. Foi A, Trimeche M, Katkovnik V, Egiazarian K (2008) Practical Poissonian-Gaussian noise modeling and fitting for single-image raw-data. *IEEE Trans Image Process* 17(10):1737–1754
14. Gravel P, Beaudoin G, De Guise JA (2004) A method for modeling noise in medical images. *IEEE Trans Med Imaging* 23(10):1221–1232
15. Greene WH (2000) *Econometric analysis*, International edn. Pearson US Imports & PHIPES
16. Jähne B (2004) *Practical handbook on image processing for scientific and technical applications*, 2nd edn. CRC Press Inc., Boca Raton, FL, USA
17. Johnson JB (1928) Thermal agitation of electricity in conductors. *Phys Rev* 32:97–109
18. Johnson NL, Kotz S, Balakrishnan N (1994) *Continuous multivariate distributions, models and applications*, vol 1. Wiley, New York
19. Lee JS (1981) Refined filtering of image noise using local statistics. *Comput Gr Image Process* 15(4):380–389

20. Lee JS, Hoppel K (1989) Noise modeling and estimation of remotely-sensed images. In: 12th International Canadian Symposium on Remote Sensing, Geoscience and Remote Sensing Symposium, 1989, IGARSS'89, vol 2, pp 1005–1008
21. Liu C, Freeman WT, Szeliski R, Kang SB (2006) Noise estimation from a single image. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol 1. IEEE, pp 901–908
22. Mäkitalo M, Foi A (2014) Noise parameter mismatch in variance stabilization, with an application to Poisson-Gaussian noise estimation. *IEEE Trans Image Process* 23(12):5348–5359
23. Mandel L (1959) Fluctuations of photon beams: the distribution of the photo-electrons. *Proc Phys Soc* 74(3):233
24. Mastin GA (1985) Adaptive filters for digital image noise smoothing: an evaluation. *Comput Vis Gr Image Process* 31(1):103–121
25. Meer P, Jolion JM, Rosenfeld A (1990) A fast parallel algorithm for blind estimation of noise variance. *IEEE Trans Pattern Anal Mach Intell* 12(2):216–223
26. Nyquist H (1928) Thermal agitation of electric charge in conductors. *Phys Rev* 32:110–113
27. Papoulis A, Pillai SU (2002) Probability, random variables, and stochastic processes, 3rd edn. Tata McGraw-Hill Education
28. Pyatykh S, Hesser J, Zheng L (2013) Image noise level estimation by principal component analysis. *IEEE Trans Image Process* 22(2):687–699
29. Starck J-L, Murtagh FD, Bijaoui A (1998) Image processing and data analysis: the multiscale approach. Cambridge University Press
30. Sutour C, Deledalle CA, Aujol JF (2015) Estimation of the noise level function based on a nonparametric detection of homogeneous image regions. *SIAM J Imaging Sci* 8(4):2622–2661
31. Tijms H (2007) Understanding probability: chance rules in everyday life, 2nd edn. Cambridge University Press, Cambridge

# Chapter 2

## Sparsity-Based Denoising of Photographic Images: From Model-Based to Data-Driven



Xin Li, Weisheng Dong and Guangming Shi

**Abstract** What makes photographic images different from random noise? It has been hypothesized that sparsity is a key factor separating the class of photographic images from noise observations. Accordingly, sparse representations have been widely studied in the literature of image denoising in the past decades. In this chapter, we present a critical review of the most important ideas/insights behind sparsity-based image denoising algorithms. In the first-generation (model-based) approaches, we will highlight the evolution from local wavelet-based image denoising in 1990s–2000s to nonlocal and patch-based image denoising from 2006 to 2015. In the second-generation (data-driven) approaches, we have opted to review several latest advances in the field of image denoising since 2016 such as learning parametric sparse models (for heavy noise removal) and deep learning-based approaches (including deep residue learning). The overarching theme of our review is to provide a unified conceptual understanding of why and how sparsity-based image denoising works—in particular, the evolving role played by *models* and *data*. Based on our critical review, we will discuss a few open issues and promising directions for future research.

---

X. Li (✉)

Lane Department of Computer Science and Electrical Engineering,  
West Virginia University, Morgantown, WV, USA  
e-mail: xin.li@ieee.org

W. Dong · G. Shi

School of Electronic Engineering, Xidian University, Xi'an, China  
e-mail: wsdong@mail.xidian.edu.cn

G. Shi

e-mail: gmshi@mail.xidian.edu.cn

## 2.1 Introduction

The collection of photographic images only occupy a small fraction characterized by a manifold in the high-dimensional space. To see this, one can think of two images of the same size (say  $H \times W$ ) as two points in the high-dimensional space  $R^{H \times W}$  and observe that any point lying along the line segment connecting these two points does not belong to the collection of photographic images. Such manifold constraint arises from the fundamental laws in quantum physics (Pauli exclusion principle)—namely, two objects cannot occupy the same space at the same time. By contrast, additive white Gaussian noise (AWGN) violates such manifold constraint because it is well known that any linear combination of two Gaussian processes is still Gaussian. When a photographic image is contaminated by AWGN, it is often desirable to remove the unwanted noise from the degraded image, which has become so-called image denoising problem. How to efficiently exploit the manifold constraint of photographic images to facilitate the task of image denoising has remained an open problem despite decades' research.

Sparsity-based denoising—one of the earliest attacks in the literature—is still influential as of today. The basic idea behind sparse representations is to formalize the abovementioned manifold constraint rigorously using well-established mathematical tools. Intuitively, there are two ways of formalizing this idea: *deterministic* and *probabilistic*. In a deterministic setting, photographic images are modeled by abstractions such as Besov-space functions whose regularity properties well match the characteristics of photographic images (e.g., images are mostly smooth except singularities such as edges and corners); in a probabilistic setting, photographic images are characterized by heavy-tail distributions in a transformed space (after sparsifying transforms) where the sparsity is specifically connected with the class of nonzero/significant coefficients (tail). Regardless of the formulation, the task of image denoising can be implemented by a simple thresholding operation—namely, if one can find an appropriate sparse representation transforming important image structures to a few exceptions (significant coefficients), thresholding becomes an effective strategy of separating them from noise (note that AWGN is unlikely to produce exceptions because of Gaussian distributions do not have heavy tails).

In the past two decades, various sparse representations have been developed reflecting the evolving thoughts of modeling important structures in photographic images. In 1990s–2000s, local sparse representations (e.g., DCT-based and wavelet-based) are constructed aiming at characterizing transient events such as edges, lines, and corners. It has been well documented in the literature that heavy-tailed distribution such as generalized Gaussian is a good fit for the probability distribution of local image structures in the wavelet domain, thanks to the good localization property of wavelet bases in both space and frequency. We note that similar observation with transient events has also led to the class of variational image models (e.g., total-variation-based [1] and Perona–Malik diffusion [2]), which have been shown to be equivalent to wavelet thresholding under certain conditions [3]. The relationship between soft/hard thresholding and Wiener filtering has also been studied in

[4], which shows that soft thresholding can be interpreted as an approximation of optimal Wiener filtering when image coefficients are modeled by Gaussian.

An important new insight—i.e., modeling nonlocal self-repeating patterns (e.g., textures) in photographic images—was introduced in 2006 [5]. Since then, a flurry of nonlocal sparse representations have been developed for various image restoration tasks including denoising. The unifying theme is to formulate a simultaneous sparse coding (SSC) problem [6] with a collection of similar patches in photographic images. Such SSC formulation can be implemented either by a 3D sparsifying transform such as in [5] (2D patches form a 3D array) or a 2D singular value decomposition transform [7] (each image patch is reshaped into a row vector of a data matrix). Since textures represent a significant portion in photographic images, nonlocal sparse representations—when applied to image denoising—have led to convincingly improved subjective quality for texture-abundant images (e.g., *Barbara*) over local models. Further optimization is possible by introducing a weighted nuclear norm minimization [8] to achieve better tradeoff between local adaptation and nonlocal invariance.

Most recently, the new wave of learning-based or data-driven approaches toward AI has swept through many applications in computer vision and pattern recognition [9]. With training data available, we suggest that there are two new opportunities to explore under the context of image denoising. First, when a small collection of correlated images are available, it is possible to learn the parametric sparse models from the training data and noisy observation together. Our recent work [10] has shown that such a hybrid (sparsity + learning) approach can lead to much more accurate estimation of image details especially in the presence of heavy noise. Second, when a large number of training images are available, it is feasible to learn a nonlinear mapping (implemented by deep convolutional neural networks) from the space of noisy observations to that of clean images via deep residue learning [11]. Such deep learning-based approach toward image denoising has rich connections with previous works (e.g., trainable nonlinear reaction-diffusion [12]) and can be combined with sparse representation as well (e.g., [13]).

The rest of this chapter is organized as follows. We have structured our review into three parts roughly following the timeline of image denoising literature: *local sparsity* (before 2006), *nonlocal sparsity* (2006–2016) and *deep learning* (2016–present). In Sect. 2.2, we start from early works on wavelet-based image denoising and focus on the paradigm shift from construction of (prefixed) basis functions to adaptively learning them from noisy observation data. In Sect. 2.3, we highlight the key ideas behind patch-based image denoising including finding similar patches and exploiting nonlocal sparsity. In Sect. 2.4, we reformulate image denoising as a discriminative learning problem, which leads to the latest advance via feedforward convolutional neural networks [11]. Experimental results are selectively reported in Sec. 2.4 to demonstrate the latest advances in the field. We make some concluding remarks in Sect. 2.5.

## 2.2 Image Denoising via Local Sparsity Models: From Sparsifying Transform to Dictionary Learning

In this section, we briefly review the class of image denoising algorithms built upon local sparsity models. The key unifying theme will be basis functions (a.k.a. dictionary)—early attempts directly construct data-independent basis functions such as wavelets and their extensions; more recent approaches obtain data-dependent basis functions via dictionary learning.

### 2.2.1 Transform-Based Image Denoising

The basic idea behind sparse representation is to decompose a signal  $\vec{x} \in R^n$  with respect to a group of  $n$ -dimensional basis vectors/functions in the Hilbert space (also called dictionary)  $\Phi \in R^{n \times m}$ —namely  $\vec{x}_{n \times 1} = \Phi_{n \times m} \vec{\alpha}_{m \times 1}$ . The sparsity constraint of  $\vec{\alpha}$  dictates that it only contains *few* nonzero coefficients (and therefore sparse). It is relatively easy to understand sparsity from a compression point of view because fewer nonzero coefficients imply higher coding efficiency. The relevance of sparsity to image denoising requires a bit more reasoning—e.g., since linear combination of Gaussian noise remains Gaussian, it is unlikely that AWGN can produce nonzero coefficients after sparse decomposition.

For photographic images, the pursuit of sparse representations has led to a rich literature of so-called sparsifying transforms including discrete cosine transform (DCT) [14] and their shape-adaptive extension SA-DCT [15], wavelet transforms [16, 17] and X-lets (e.g., ridgelets [18], curvelet [19] and contourlets [20]). Retrospectively, geometry has been the most decisive factor pushing the field of sparse representations forward. Early constructions such as DCT and wavelet transforms are separable, which makes it difficult to exploit the geometric constraint of edges [21]. Non-separable transforms such as shape-adaptive DCT (SA-DCT) [15] and contourlet filter banks [20] directly pursue an *anisotropic* or *directional* image representation across different scales. Such directional multi-resolution representations of images have been proven theoretically more compact than their separable counterparts and demonstrated convincing improved performance for image denoising applications (e.g., [15, 19, 20]).

Despite those progress, the fundamental issue of an ideal sparse representation has remained open. On one hand, when several competing constructed dictionaries are available, one often faces the problem of *selection*—i.e., which subset of basis functions could lead to an “optimal” choice (e.g., basis pursuit [22])? It turns out that such problem is often ill-posed because of the difficulty with defining optimality. For instance, it is well known that  $l_0$ -based optimization is nonconvex and does not admit computationally efficient solutions; consequently, its  $l_1$ -based counterpart [23] is often preferred in practice. On the other hand, one can pursue a sparser representation by adding more and more atoms into the dictionary but at the risk of over-fitting (not to



mention the price of prohibitive computational complexity). In view of the diversity of image structures in photos, it seems unrealistic to manually construct a prefixed dictionary with all required dictionary elements. A more plausible solution is to *learn* (instead of construct) the set of dictionary elements from training data.

## 2.2.2 Image Denoising via Dictionary Learning

The idea of learning a dictionary from training data dated back to the classical principle component analysis (PCA) [24] (a.k.a. Karhunen–Loeve transform). However, the optimality of Karhunen–Loeve transform (KLT) for stationary Gaussian processes is not applicable to the source of photographic images. Nonstationarity of image source calls for the segmentation or clustering technique such as K-means method [25]. The basic idea behind K-means clustering is to overcome the chicken-and-egg dilemma by iteratively updating the centroids (representative codewords) and class labels (clustering outcome). An important milestone in dictionary learning is the development of K-SVD algorithm [26] which connects the concept of K-means clustering with sparse coding. Similar to K-means, K-SVD also alternates between sparse coding and dictionary updating; but unlike K-means, K-SVD puts data clustering on a solid theoretic foundation of optimal data representation (note that it works with any pursuit method).

When applied to photographic images, K-SVD has led to so-called *sparseland* model [27], which we will briefly review here. Let  $\mathbf{x}_i$  denote a patch cropped from image  $\mathbf{X}$  at the spatial location  $i$ ; then, we have

$$\mathbf{x}_i = \mathbf{E}_i \mathbf{X}, \quad (2.1)$$

where  $\mathbf{E}_i$  is an operator of sliding window. Since patch overlapping is allowed, the above representation is redundant and the reconstruction of image from cropped patches  $\{\mathbf{x}_i\}$  is overdetermined, which admits the following Least-Square solution

$$\mathbf{X} = \left( \sum_i \mathbf{E}_i^T \mathbf{E}_i \right)^{-1} \left( \sum_i \mathbf{E}_i^T \mathbf{x}_i \right). \quad (2.2)$$

For a given dictionary  $\Phi$ , each patch can be represented by its sparse coefficients  $\{\vec{\alpha}_i\}$ —namely

$$\mathbf{x}_i = \Phi \vec{\alpha}_i. \quad (2.3)$$

Combining Eqs. (2.3) and (2.2), we have

$$\mathbf{X} = \mathbf{R} \vec{\alpha} \doteq \left( \sum_i \mathbf{E}_i^T \mathbf{E}_i \right)^{-1} \left( \sum_i \mathbf{E}_i^T \Phi \vec{\alpha}_i \right), \quad (2.4)$$

where  $\mathbf{R}$  is the reconstruction operator dual to  $\mathbf{E}$ .

Under the context of image denoising, one can formulate the following variational problem

$$\vec{\alpha}_i = \operatorname{argmin}_{\vec{\alpha}_i} \frac{1}{2} \|\mathbf{Y} - \mathbf{D}\vec{\alpha}_i\|_2^2 + \lambda \|\vec{\alpha}_i\|_1, \quad (2.5)$$

where  $\mathbf{Y} = \mathbf{X} + \mathbf{N}$  is the noisy image,  $\mathbf{A}$  is matrix containing vectors of sparse coefficients, and  $\lambda$  is the Lagrangian multiplier. The basic idea behind K-SVD denoising [27] is to learn the dictionary  $\mathbf{D}$  from noisy observation image  $\mathbf{Y}$  by solving the following joint optimization problem

$$\operatorname{argmin}_{\mathbf{X}, \mathbf{D}, \vec{\alpha}} \sum_i \|\mathbf{D}\vec{\alpha}_i - \mathbf{R}_i \mathbf{X}\|_2^2 + \sum_i \gamma_i \|\vec{\alpha}_i\|_1 + \lambda \|\mathbf{Y} - \mathbf{X}\|, \quad (2.6)$$

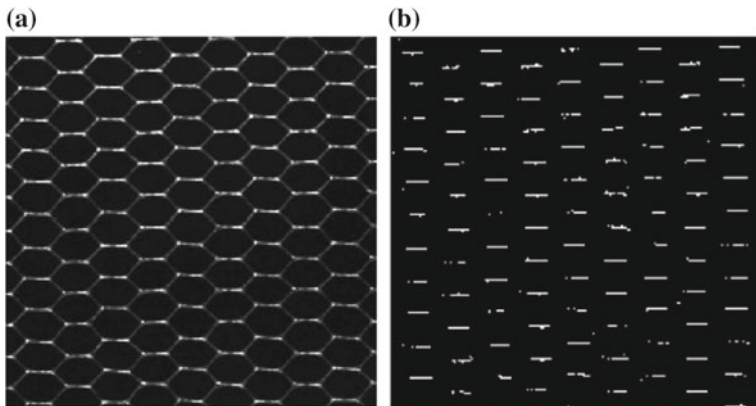
The solution to the above optimization problem consists of iterative updating of sparse coefficients  $\vec{\alpha}$  and dictionary elements  $\mathbf{D}$ . The step of updating  $\mathbf{D}$  can be implemented by singular value decomposition (SVD), which gives the name K-SVD.

## 2.3 Image Denoising via Nonlocal Sparsity Models: Beyond BM3D

The year of 2006 was fruitful for the field of image denoising. In addition to K-SVD, the conference version of block-matching 3D (BM3D) denoising [5] was also published in 2016 [28]. At the beginning, the community was baffled by the outstanding experimental results reported in the paper—why did it work so well? It took the whole community several years to develop and optimize the class of nonlocal sparse representations.

### 2.3.1 Image Denoising via Clustering-Based Sparse Representation (CSR)

To see why K-SVD falls behind BM3D, we can conduct an experiment with a toy example of regular texture. It can be observed that the sparse (nonzero) coefficients  $\vec{\alpha}$  are NOT randomly distributed as shown in Fig. 2.1. The striking pattern of sparse coefficients in the spatial domain arises from the self-repeating pattern in texture image itself. Such nonlocal similarity is beyond the reach of dictionary learning such as K-SVD because the regularization terms in Eq. 2.6 cannot exploit the high-order correlation among sparse coefficients  $\vec{\alpha}_i$ 's. So instead of extending K-means, one can combine K-means with sparse coding to obtain a nonlocal clustering-based sparse representation (CSR) as follows.



**Fig. 2.1** Limitation of K-SVD: **a** an image of regular texture; **b** spatial distribution of sparse coefficients corresponding to the sixth basis vector (note that their locations are NOT random)

The key new insight is to view the centroids of k-means clustering as peer hidden variables to unknown sparse coefficients. Referring to Fig. 2.1, one can envision re-encoding of similar sparse coefficients could render a more compact representation promoting sparsity (by exploiting self-repeating patterns). Along this line of reasoning, we propose the following cost function as a nonlocal extension of Eq. (2.6)

$$\begin{aligned}
 (\vec{\alpha}, \vec{\mu}) = \operatorname{argmin}_{\vec{\alpha}, \vec{\mu}_k} & \frac{1}{2} \|\mathbf{Y} - \mathbf{D}\vec{\alpha}\|_2^2 + \lambda_1 \|\vec{\alpha}\|_1 \\
 & + \lambda_2 \sum_{k=1}^K \sum_{i \in C_k} \|\Phi \vec{\alpha}_i - \vec{\mu}_k\|_1,
 \end{aligned} \tag{2.7}$$

where  $\vec{\mu}_k$  denotes the centroid of the  $k$ -th cluster  $C_k$  of coefficients  $\vec{\alpha}$ . When compared against Eq. (2.6), we note that the new clustering-based regularization term is introduced to exploit nonlocal similarity in photographic images. To gain deeper insight about the new regularization term, one can rewrite Eq. (2.7)

$$\begin{aligned}
 (\vec{\alpha}, \vec{\beta}) = \operatorname{argmin}_{\vec{\alpha}, \vec{\mu}_k} & \frac{1}{2} \|\mathbf{Y} - \mathbf{D}\vec{\alpha}\|_2^2 + \lambda_1 \|\vec{\alpha}\|_1 \\
 & + \lambda_2 \sum_{k=1}^K \sum_{i \in C_k} \|\Phi \vec{\alpha}_i - \Phi \vec{\beta}_k\|_1,
 \end{aligned} \tag{2.8}$$

where  $\vec{\mu}_k = \Phi \vec{\beta}_k$  (i.e., to represent all centroids by the same dictionary  $\Phi$  as  $\mathbf{x}_i$ ). Since  $\Phi$  is unitary, we have  $\|\Phi \vec{\alpha}_i - \Phi \vec{\beta}_k\|_2^2 = \|\vec{\alpha}_i - \vec{\beta}_k\|_2^2$ . Therefore, Eq. (2.7) boils down to the following joint optimization problem

$$\begin{aligned}
(\vec{\alpha}, \vec{\beta}) = \operatorname{argmin}_{\vec{\alpha}, \vec{\mu}_k} & \frac{1}{2} \|\mathbf{Y} - \mathbf{D}\vec{\alpha}\|_2^2 + \lambda_1 \|\vec{\alpha}\|_1 \\
& + \lambda_2 \sum_{k=1}^K \sum_{i \in C_k} \|\vec{\alpha}_i - \vec{\beta}_k\|_1.
\end{aligned} \tag{2.9}$$

Using the technique of conjugate function, we can solve the above double-header  $l_1$ -minimization problem and obtain the following iterative thresholding solution

$$\alpha_j^{(i+1)} = \begin{cases} \mathbf{S}_{\tau_1, \tau_2}(v_j^{(i)}) & \beta_j \geq 0 \\ -\mathbf{S}_{\tau_1, \tau_2}(-v_j^{(i)}) & \beta_j < 0 \end{cases} \tag{2.10}$$

where

$$\mathbf{v}^{(i)} = \frac{1}{c} \mathbf{D}^T (\mathbf{x} - \mathbf{D}\alpha^{(i)}) + \alpha^{(i)}, \tag{2.11}$$

and  $\tau_1 = \frac{\lambda_1}{c}$ ,  $\tau_2 = \frac{\lambda_2}{c}$  ( $c$  is an auxiliary parameter guaranteeing the convexity of surrogate function), superscript ( $i$ ) denotes iteration number and subscript  $j$  denotes the  $j$ -th entry in a vector. The detailed derivation of iterative thresholding solution is referred to [29].

### 2.3.2 Simultaneous Sparse Coding with Low-Rank Approximation

An alternative way of exploiting nonlocal similarity is the so-called *simultaneous sparse coding* (SSC) [6]. The basic idea of SSC (a.k.a. group/structured sparsity [30]) is to reshape each patch into a row vector and group those similar patches into a 2D matrix  $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_m] \in R^{n \times m}$ . Then, the *group sparsity* can be defined by a pseudo-matrix norm  $\|\mathbf{A}\|_{p,q}$ :

$$(\mathbf{U}, \mathbf{A}) = \operatorname{argmin}_{\mathbf{A}} \|\mathbf{Y} - \mathbf{U}\mathbf{A}\|_F^2 + \tau \|\mathbf{A}\|_{p,q}, \tag{2.12}$$

where  $\mathbf{A} = [\alpha^1; \alpha^2; \dots; \alpha^n]$  is the matrix formulation of sparse coding strategy  $\mathbf{X} = \mathbf{U}\mathbf{A}$  and the pseudo-matrix norm  $\|\cdot\|_{p,q}$  is given by [31]

$$\|\mathbf{A}\|_{p,q} \triangleq \sum_{i=1}^n \|\alpha^i\|_q^p, \tag{2.13}$$

where  $\alpha^i = [\alpha_{i,1}, \dots, \alpha_{i,m}]$  denotes the  $i$ th row of matrix  $\mathbf{A}$  in  $R^{n \times m}$ . Note that the above formulation does not treat the row and column spaces equally—namely, a matrix  $\mathbf{A}$  and its transpose  $\mathbf{A}^T$  will be characterized by varying amount of group

sparsity (though they have the same amount of sparsity). To restore the symmetry between row and column spaces, we have proposed to introduce a right-multiplying matrix  $\mathbf{V}$  in [7]. Then, the matrix  $\mathbf{A}$  can be rewritten as

$$\mathbf{A} = \mathbf{\Sigma} \mathbf{V}^T, \quad (2.14)$$

where  $\mathbf{\Sigma} = \text{diag}\{\lambda_1, \lambda_2, \dots, \lambda_K\}$  ( $K = \min(m, n)$ ) is a diagonal matrix in  $R^{K \times K}$  and each column of  $\mathbf{V}$  in  $R^{m \times K}$  consists of  $\mathbf{v}_i = \frac{1}{\lambda_i} (\boldsymbol{\alpha}^i)^T$ . It is enlightening to observe that the new sparsifying matrix  $\mathbf{V}$  plays the dual role of the dictionary  $\mathbf{U}$ , and therefore Eq. (2.12) boils down to

$$(\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}) = \underset{\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}}{\text{argmin}} \|\mathbf{Y} - \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T\|_F^2 + \tau \|\mathbf{A}\|_{p,q}. \quad (2.15)$$

The key new insight behind our low-rank approximation approach is that when  $p = 1, q = 2$ , the group sparsity regularizer  $\|\mathbf{A}\|_{1,2}$  computes the sum of standard deviations associated with sparse coefficient vector in each row—namely

$$\|\mathbf{A}\|_{1,2} = \sum_{i=1}^K \sqrt{\alpha_{i,1}^2 + \alpha_{i,2}^2 + \dots + \alpha_{i,m}^2} = \sum_{i=1}^K \sqrt{m} \sigma_i, \quad (2.16)$$

where  $\sigma_i$  is the standard deviation of sparse coefficients  $\boldsymbol{\alpha}^i$  in the  $i$ -th row. Therefore, we can write  $\boldsymbol{\alpha}^i = \lambda_i \mathbf{v}_i^T$  and obtain

$$\sigma_i^2 = \frac{1}{m} \|\boldsymbol{\alpha}^i\|_2^2 = \frac{1}{m} \|\lambda_i \mathbf{v}_i^T\|_2^2 = \frac{\lambda_i^2}{m}, \quad (2.17)$$

where the first identity comes from Eq. (2.16) and the last one is due to the unitary property of  $\mathbf{V}$ . Based on Eq. (2.17), one can conclude that any operation on sparse coefficient vector  $\boldsymbol{\alpha}$ 's can be equivalently implemented with respect to  $\sigma_i$  or  $\lambda_i$  (only differs by a constant scalar). Substituting Eq. (2.17) into Eq. (2.15), we have the following standard low-rank approximation problem [32]

$$(\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}) = \underset{\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}}{\text{argmin}} \|\mathbf{Y} - \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T\|_F^2 + \tau \sum_{i=1}^K \lambda_i, \quad (2.18)$$

which admits the well-known singular value decomposition (SVD)-based solution [32]

$$\begin{cases} (\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}) = \text{svd}(\mathbf{Y}); \\ \hat{\mathbf{\Sigma}} = \mathbf{S}_\tau(\mathbf{\Sigma}); \end{cases} \quad (2.19)$$

where  $\mathbf{S}_\tau$  is a soft thresholding operator with a threshold  $\tau$  (regularization parameter). The reconstructed data matrix can be conveniently obtained by  $\hat{\mathbf{X}} = \mathbf{U} \hat{\mathbf{\Sigma}} \mathbf{V}^T$ .

### 2.3.3 Simultaneous Sparse Coding with Gaussian Scalar Mixture (GSM)

The connection between sparse coding and Bayesian estimation has been well-established in the literature (e.g., refer to [29]). If we rewrite Eq. (2.5) into

$$\boldsymbol{\alpha} = \underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \|\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + \lambda \|\boldsymbol{\alpha}\|_1, \quad (2.20)$$

the above  $l_1$ -minimization is equivalent to the maximum a posterior (MAP) inference of  $\boldsymbol{\alpha}$  with an identically independent distributed (i.i.d) Laplacian prior  $P(\alpha_i) = \frac{1}{2\theta_i} e^{-\frac{|\alpha_i|}{\theta_i}}$ , wherein  $\theta_i$  denotes the standard derivation of  $\alpha_i$ . Such a fruitful connection can be exploited to shed new insight to SSC by leveraging advanced statistical modeling technique. For instance, in the literature of wavelet-based image denoising, it has been proposed in [4] to characterize sparse coefficients  $\boldsymbol{\alpha}$  by a Gaussian scale mixture (GSM) [33] model. The original motivation behind GSM-based image denoising is to more accurately characterize rapidly evolving local statistics (e.g., variance) of wavelet coefficients. As we have shown in [34], it is possible to combine GSM with SSC and obtain a class of more powerful nonlocal sparsity-based image denoising algorithms.

The basic idea behind GSM is to decompose sparse coefficient vector  $\boldsymbol{\alpha}$  into the point-wise product of a Gaussian vector  $\boldsymbol{\beta}$  of unitary variance and a hidden scalar multiplier  $\boldsymbol{\theta}$  ( $\alpha_i = \theta_i \beta_i$ ). It follows from the Bayesian formula that the GSM prior of  $\boldsymbol{\alpha}$  can be expressed by

$$P(\boldsymbol{\alpha}) = \prod_i P(\alpha_i), \quad P(\alpha_i) = \int_0^\infty P(\alpha_i|\theta_i)P(\theta_i)d\theta_i. \quad (2.21)$$

In view of the difficulty with computing the MAP estimate of  $\alpha_i$  for most choices of  $P(\theta_i)$ , wavelet-based GSM denoising [4] treated  $\theta_i$  as a hidden variable, which can be canceled through integration; later work [35] explicitly uses the field of Gaussian scalar multiplier (field-of-GSM) to characterize the variability and dependencies among local variances while at the price of prohibitive computational complexity. A *computationally more tractable* approach is to formulate a *joint* estimation of  $(\alpha_i, \theta_i)$  by exploiting nonlocal similarity of photographic images (same motivation as SSC) as follows

$$\begin{aligned} (\boldsymbol{\alpha}, \boldsymbol{\theta}) &= \operatorname{argmax} \log P(\mathbf{x}|\boldsymbol{\alpha}, \boldsymbol{\theta}) P(\boldsymbol{\alpha}, \boldsymbol{\theta}) \\ &= \operatorname{argmax} \log P(\mathbf{x}|\boldsymbol{\alpha}) + \log P(\boldsymbol{\alpha}|\boldsymbol{\theta}) + \log P(\boldsymbol{\theta}). \end{aligned} \quad (2.22)$$

To gain deeper understanding toward the above formulation, we note that the first term  $P(\mathbf{x}|\boldsymbol{\alpha})$  is characterized by a Gaussian function with variance  $\sigma_n^2$ , the second term  $P(\boldsymbol{\alpha}|\boldsymbol{\theta})$  is given by

$$P(\boldsymbol{\alpha}|\boldsymbol{\theta}) = \prod_i P(\alpha_i|\theta_i) = \prod_i \frac{1}{\theta_i\sqrt{2\pi}} \exp\left(-\frac{(\alpha_i - \mu_i)^2}{2\theta_i^2}\right), \quad (2.23)$$

and the third term is so-called noninformative prior [36]  $P(\theta_i) \approx \frac{1}{\theta_i}$  (a.k.a. Jeffrey's prior [4]). Putting things together, we can rewrite Eq. (2.22) into

$$\begin{aligned} (\boldsymbol{\alpha}, \boldsymbol{\theta}) = \operatorname{argmin}_{\boldsymbol{\alpha}, \boldsymbol{\theta}} & \frac{1}{2\sigma_n^2} \|\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + \sum_i \log(\theta_i\sqrt{2\pi}) \\ & + \sum_i \frac{(\alpha_i - \mu_i)^2}{2\theta_i^2} + \sum_i \log \theta_i, \end{aligned} \quad (2.24)$$

where we have used  $P(\boldsymbol{\theta}) = \prod_i P(\theta_i)$ . To improve numerical stability, one can replace  $\log \theta_i$  by  $\log(\theta_i + \epsilon)$  ( $\epsilon > 0$ ) and obtain

$$\begin{aligned} (\boldsymbol{\alpha}, \boldsymbol{\theta}) = \operatorname{argmin}_{\boldsymbol{\alpha}, \boldsymbol{\theta}} & \|\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + 4\sigma_n^2 \log(\boldsymbol{\theta} + \epsilon) \\ & + \sigma_n^2 \sum_i \frac{(\alpha_i - \mu_i)^2}{\theta_i^2}, \end{aligned} \quad (2.25)$$

where  $\log(\boldsymbol{\theta} + \epsilon) = \sum_i \log(\theta_i + \epsilon)$ .

In the matrix form of original GSM model, we have  $\boldsymbol{\alpha} = \Lambda\boldsymbol{\beta}$  and  $\boldsymbol{\mu} = \Lambda\boldsymbol{\gamma}$  where  $\Lambda = \operatorname{diag}(\theta_i) \in R^{K \times K}$  is a diagonal matrix characterizing the variance field for the chosen image patch. Such connections allow us to translate the optimization problem in Eq. (2.24) from  $(\boldsymbol{\alpha}, \boldsymbol{\mu})$  domain to  $(\boldsymbol{\beta}, \boldsymbol{\gamma})$  domain—i.e.,

$$(\boldsymbol{\beta}, \boldsymbol{\theta}) = \operatorname{argmin}_{\boldsymbol{\beta}, \boldsymbol{\theta}} \|\mathbf{x} - \mathbf{D}\Lambda\boldsymbol{\beta}\|_2^2 + 4\sigma_n^2 \log(\boldsymbol{\theta} + \epsilon) + \sigma_n^2 \|\boldsymbol{\beta} - \boldsymbol{\gamma}\|_2^2. \quad (2.26)$$

Since the above equations are the sparse coding formulation for a single image patch, it is natural to formulate a simultaneous sparse coding (SSC) problem by generalizing the optimization problem in Eq. (2.26) from vector form to matrix form

$$\begin{aligned} (\mathbf{B}, \boldsymbol{\theta}) = \operatorname{argmin}_{\mathbf{B}, \boldsymbol{\theta}} & \|\mathbf{X} - \mathbf{D}\Lambda\mathbf{B}\|_F^2 + 4\sigma_n^2 \log(\boldsymbol{\theta} + \epsilon) \\ & + \sigma_n^2 \|\mathbf{B} - \Gamma\|_F^2, \end{aligned} \quad (2.27)$$

where  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_m]$  is a 2D matrix formed by  $m$  similar patches and  $\mathbf{A} = \Lambda\mathbf{B}$  is the matrix representation of  $\boldsymbol{\alpha} = \Lambda\boldsymbol{\beta}$ . The first-order and second-order statistics of  $\mathbf{A}$  are characterized by  $\Gamma = [\boldsymbol{\gamma}_1, \dots, \boldsymbol{\gamma}_m] \in R^{K \times m}$  and  $\mathbf{B} = [\boldsymbol{\beta}_1, \dots, \boldsymbol{\beta}_m] \in R^{K \times m}$  respectively, wherein  $\boldsymbol{\gamma}_j = \boldsymbol{\gamma}$ ,  $j = 1, 2, \dots, m$ . Given  $m$  similar patches, we have adopted the following strategy for estimating  $\boldsymbol{\mu}$

$$\boldsymbol{\mu} = \sum_{j=1}^m w_j \boldsymbol{\alpha}_j, \quad (2.28)$$

where  $w_j \sim \exp(-\|\mathbf{x} - \mathbf{x}_j\|_2^2/h)$  is the weighting coefficient based on patch similarity. It follows from  $\boldsymbol{\mu} = \Lambda \boldsymbol{\gamma}$  that

$$\boldsymbol{\gamma} = \sum_{j=1}^m w_j \Lambda^{-1} \boldsymbol{\alpha}_j = \sum_{j=1}^m w_j \boldsymbol{\beta}_j, \quad (2.29)$$

The above new formulation in Eq. (2.27) has been called *Simultaneous Sparse Coding for Gaussian Scalar Mixture* (SSC-GSM) and computationally efficient solution has been derived in [34]. When applied to image denoising, SSC-GSM has achieved the current state-of-the-art performance.

## 2.4 Learning-Based Image Denoising: Heavy Noise and Deep Learning

In previous sections, we have focused on estimating a clean image from its contaminated version without any help from training data. When training data are available, it is possible to further optimize image denoising by learning a more powerful prior. In this section, we present two latest advances in learning-based image denoising: one deals with parametric sparse models and the other with deep convolutional neural networks.

### 2.4.1 Learning Parametric Sparse Models for Heavy Noise Removal

In the literature of image denoising, one of under-researched topic is the removal of heavy noise. As the power of noise increases, it becomes more challenging to estimate the unknown image characteristics (e.g., mean and variance) from the noisy observation. One way of alleviating this difficulty is to assume that some training data are available as reference images—e.g., images of the same scene but acquired at different time or from varying viewpoints. The key new insight brought by the availability of additional training data is that they could facilitate the estimation of sparse model parameters (e.g., the biased mean vector in Eq. 2.29) by providing noise-free similar patches. Figure 2.2 describes the proposed framework of learning parametric sparse models from training data, which consists of four steps (image retrieval, global registration, patch matching, and expectation estimation). We will elaborate each step, respectively, next.



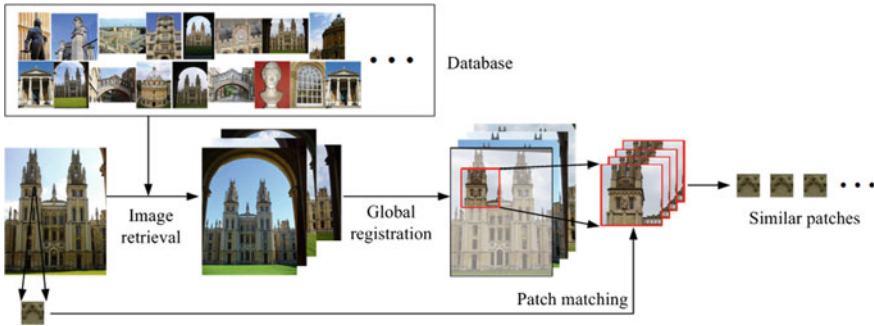


Fig. 2.2 Framework of learning parametric sparse model from a set of training data

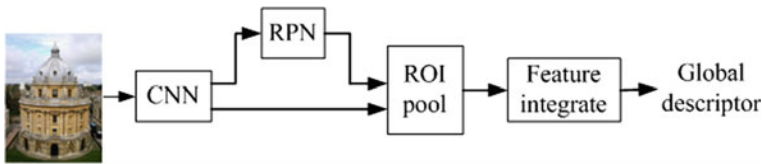


Fig. 2.3 Framework of generating global descriptor for image retrieval by the method of [40]

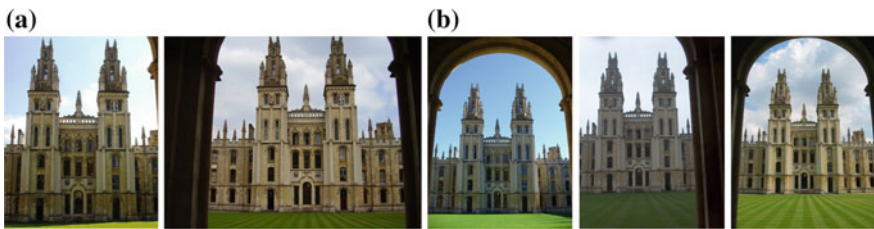


Fig. 2.4 An example result of image retrieval. **a** The query image. **b** The retrieved images

**Image retrieval** The first step is to retrieve images that contain similar content to a query image from the training data. Image retrieval has been extensively studied in the literature (e.g., [37–40]); here we have adopted an image retrieval method of [40]. In [40], a global descriptor is generated for both the query image and the candidates, as shown in Fig. 2.3. To create such a global descriptor, a Convolutional Neural Network (CNN) is first applied to the given image; then, a pretrained Region Proposal Network (RPN) is activated to pool regions of interest (ROI) together; and finally, a global descriptor is produced by integrating ROI-pooled features. With generated global descriptors, the similarity between two images can simply be measured by the dot product of two corresponding global descriptors. To address the issue of noise interference, we can obtain a pre-denoised image by any conventional approach (without the use of training data) and apply image retrieval to the pre-denoised image while searching for correlated images. Figure 2.4 shows an example of image retrieval results.



**Fig. 2.5** The retrieved images after global registration

**Global registration** The retrieved images cannot be directly used to find similar patches because of the difference in scale, illumination, and viewpoint. The next step is to align the retrieved image with the pre-denoised image, which has been known as global registration. In the literature of image registration, aligning two images using local features such as SIFT [41] and local binary pattern [42] is among the most popular approaches. Similar to [43, 44], we have extracted the SIFT [41] descriptors from the input image pair. Then, the eight-parameter homography matrix is estimated from matched SIFT descriptor pairs by the Random Sample Consensus (RANSAC) algorithm [45]. The retrieved correlated images can finally be aligned to the query image after the geometric transformation with the estimated homography matrix. Figure 2.5 presents an example of global registration. We denote the transformed images by  $\{z_r\}_{r=1}^R$ , which are used for patch matching next.

**Patch matching** Given an exemplar patch  $x_i$  extracted from the noisy image  $x$ , we search for similar patches from the aligned reference images (i.e.,  $\{z_r\}_{r=1}^R$ ). As the size of exemplar patch  $x_i$  is usually small (e.g.,  $7 \times 7$ ), direct patch matching with  $x_i$  is not accurate especially in the presence of heavy noise. Instead, we have used an enlarged *anchor* patch containing  $x_i$  at the center to facilitate patch matching to ensure its accuracy. Denote  $x'_i \in \mathbb{R}^r$  of size  $\sqrt{n'} \times \sqrt{n'}$  ( $n' > n$ ) as the query patch extracted from  $x$  at the position  $i$ . We first search for similar patches to  $x'_i$  within a local window centered at the position  $i$  in reference images; then group the found similar patches into a data matrix. To achieve illumination invariance, we remove the DC component of each patch before calculating the  $l_2$  distance among patches.

**Estimation of expectations  $\mu_i$**  Given an exemplar patch  $x_i$  and its similar patches found from reference images  $\{\tilde{z}_{i,l}\}_{l=1}^L$ , we assemble the exemplar patch and reference patches into a data matrix, each column of which corresponds to a patch (including the  $x_i$ ). Similar to [46], we first apply a median filtering to the data matrix along each row to remove the noise. Then, the expectation of  $\hat{x}_i$  from reference patches can be estimated as

$$x_i^{ref} = \text{median}(\hat{x}_i, \tilde{z}_{i,1}, \tilde{z}_{i,2}, \dots, \tilde{z}_{i,L}), \quad (2.30)$$

where superscript *ref* indicates that  $x_i^{ref}$  is learned from the reference images. Meanwhile, we note that expectations from nonlocal similar patches—i.e., Eq. 2.29 can be rewritten into

$$\mathbf{x}_i^{int} = \sum_{k=1}^m w_{i,k} \hat{\mathbf{x}}_{i,k}, \quad (2.31)$$

where  $w_{i,k} = (1/c) \exp(-\|\hat{\mathbf{x}}_{i,k} - \hat{\mathbf{x}}_i\|/h)$ ,  $c$  is the normalization parameter and  $h$  is a predefined constant. The superscript *int* indicates the intrinsic estimation from the noisy image itself. With  $\boldsymbol{\mu}_i^{ref} = \mathbf{D}^\top \mathbf{x}_i^{ref}$  and  $\boldsymbol{\mu}_i^{int} = \mathbf{D}^\top \mathbf{x}_i^{int}$  denote the sparse codes of  $\mathbf{x}_i^{ref}$  and  $\mathbf{x}_i^{int}$  with respect to the dictionary  $\mathbf{D}$ , a more accurate estimation of the expectation of  $\boldsymbol{\alpha}_i$  can be achieved by combining  $\boldsymbol{\mu}_i^{ref}$  and  $\boldsymbol{\mu}_i^{int}$  as follows

$$\boldsymbol{\mu}_i = \Delta \boldsymbol{\mu}_i^{ref} + (\mathbf{I} - \Delta) \boldsymbol{\mu}_i^{int}, \quad (2.32)$$

where  $\Delta = \text{diag}(\delta_j) \in \mathbb{R}^{K \times K}$  is the weights between  $\boldsymbol{\mu}_i^{ref}$  and  $\boldsymbol{\mu}_i^{int}$ . Similar to [44],  $\delta_j$  is computed according to the energy ratio of  $\boldsymbol{\mu}_i^{ref}(j)$  and  $\boldsymbol{\mu}_i^{int}(j)$ - i.e.,

$$\delta_j = \frac{r_j^2}{r_j^2 + 1/(r_j^2)}, \quad r_j = \boldsymbol{\mu}_i^{ref}(j)/\boldsymbol{\mu}_i^{int}(j), \quad (2.33)$$

## Experimental Results

We take 16 images from the Oxford5K<sup>1</sup> and Paris6K<sup>2</sup> datasets as test images (please refer to Fig. 2.6). In the proposed method, similar images are retrieved from the remaining images of the dataset. We have compared the proposed image denoising method against several well-known denoising methods—i.e., BM3D [5], NCSR [47], maximizing expected patch log likelihood (EPLL) [48], SSC-GSM [34] and DnCNN [11]. Note that DnCNN [11] is state-of-the-art denoising method, and BM3D [5], NCSR [47], SSC-GSM [34] are state-of-the-art single image denoising methods. For DnCNN [11], we use the same training data as that in [11]. The default parameter settings are applied in our experiments. The Gaussian noise deviation  $\theta_n$  is set to be 50, 70, and 90. The larger patch size—i.e.,  $n'$  for patch matching from reference images are set as 17, 19, and 21, respectively. We retrieve four reference images from the dataset (i.e.,  $R = 4$ ).

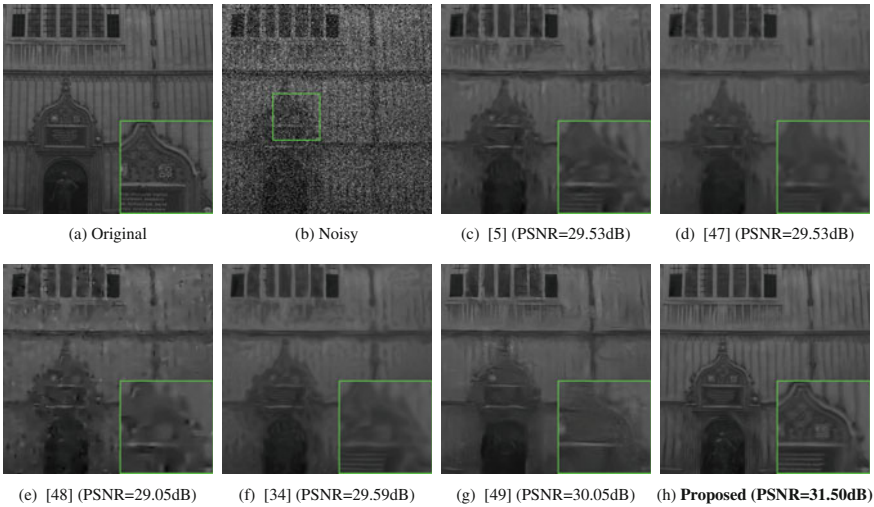
Besides, the method of DnCNN [11], which is implemented using a deep convolutional network, has the best performance among all benchmark methods. When compared against DnCNN [11], PSNR improvement of the proposed method is more than 1 dB on the average (up to 1.47 dB); the average SSIM improvement is around 0.0948. Parts of the reconstructed images by different denoising methods are shown in Fig. 2.7 for noise level of 50, Fig. 2.8 for noise level of 70, Fig. 2.9 for noise level of 90. From Figs. 2.7, 2.8 and 2.9, it can be seen that the restored image by previous methods such as BM3D[5], NCSR [47] and SSC-GSM [34] are often over-smoothed and the results by the proposed method contain much more faithfully restored details than previous methods.

<sup>1</sup><http://www.robots.ox.ac.uk/~vgg/data/oxbuildings/>.

<sup>2</sup><http://www.robots.ox.ac.uk/~vgg/data/parisbuildings/>.

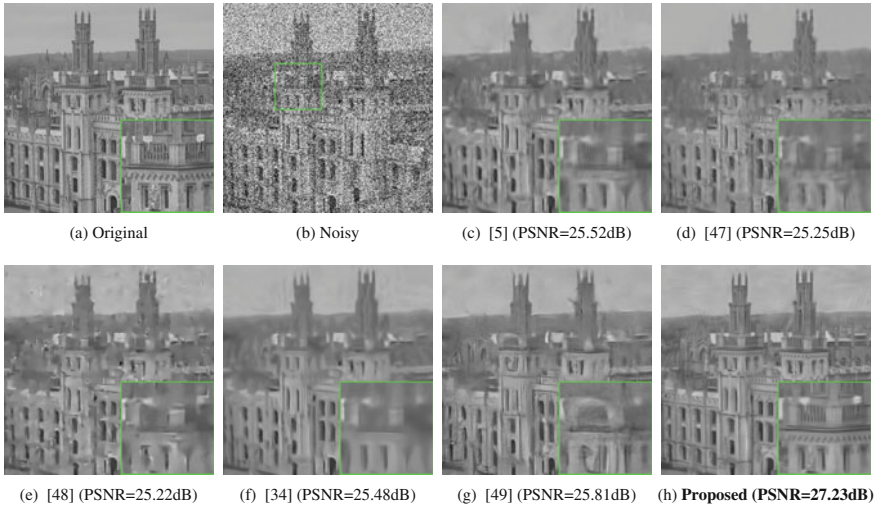


**Fig. 2.6** 16 test images. From left to right and from up to bottom, the images are named as “img1” to “img16”

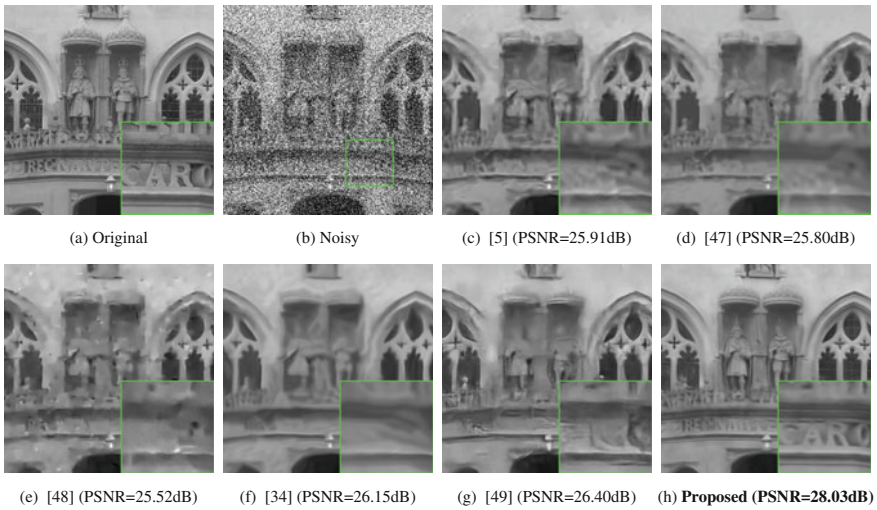


**Fig. 2.7** Denoising results of cropped image ‘img3’ ( $\sigma_n = 50$ )

The denoising method of [46] (denoted as CID), which is based on a combination of the BM3D method and retrieved correlated images, has outstanding performance over BM3D and EPLL as shown in [46]. As the implementation codes for [46] is not available, we make a comparison with the results given in [46], where the test images are exactly the images of “img1”–“img7”. The comparison of the PSNR and results with [46] is shown in Table 2.1, from which it can be seen that the proposed method has comparable SSIM performance to CID and the average PSNR improvement over CID can be up to 0.64 dB.



**Fig. 2.8** Denoising results of cropped image ‘img4’ ( $\sigma_n = 70$ )



**Fig. 2.9** Denoising results of cropped image ‘img6’ ( $\sigma_n = 90$ )

### 2.4.2 Learning Denoising Prior in Deep Convolutional Neural Network

Instead of learning a sparse model from training data, one can reformulate image denoising as a discriminative learning problem—i.e., can we directly learn a nonlinear mapping from the space of noisy images to that of clean images? Early attempts

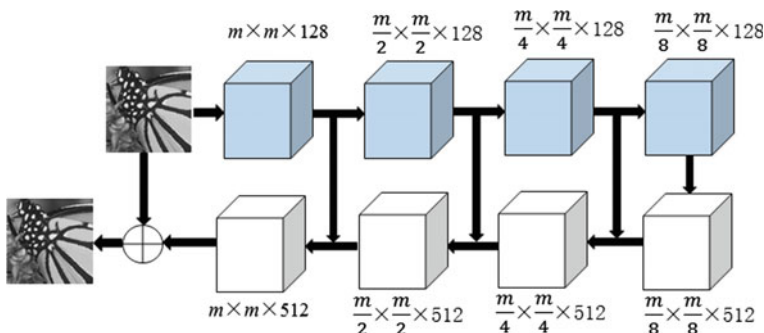
**Table 2.1** PSNR and SSIM results compared with CID [46]

	Noise	img1	img2	img3	img4	img5	img6	img7	Avg.
CID [46]	50	30.05	28.56	30.64	26.80	29.50	28.50	28.90	28.99
		0.882	0.861	0.855	0.766	0.909	0.799	0.837	0.844
	70	29.00	27.63	29.44	26.03	28.53	27.75	27.98	28.05
		0.860	0.837	0.816	0.734	0.885	0.777	0.819	0.818
	90	27.70	26.83	28.63	25.25	27.15	26.91	27.02	27.07
		0.845	0.822	0.809	0.706	0.873	0.753	0.803	0.802
Proposed	50	30.43	28.91	31.50	28.08	28.86	29.80	29.84	29.63
		0.8734	0.8408	0.8527	0.7820	0.8488	0.8242	0.8463	0.8383
	70	29.09	28.15	30.16	27.23	28.01	28.92	28.47	28.56
		0.8515	0.8202	0.8203	0.7535	0.8318	0.8045	0.8177	0.8142
	90	27.74	27.76	29.38	26.33	27.19	28.03	27.51	27.71
		0.8343	0.8157	0.8111	0.7220	0.8137	0.7881	0.7977	0.7975

such as multilayer perceptron [49] have shown plain neural networks can deliver comparable denoising performance to BM3D. However, as the layer of neural networks gets larger, the training of such a deep convolutional neural network faces the notorious problem called vanishing moment [50]. One of the key new insights introduced to the field of deep learning in recent years is the concept of residue learning [51]—instead of learning a mapping from the space of noisy images to that of clean images directly, it is better to learn a mapping from the space of noisy images to the residue (the difference between the target and degraded image). Although the framework of deep residue learning was originally introduced for image classification, its potential in low-level vision tasks such as image super-resolution and denoising has been quickly explored since 2016. In the past few years, many learning-based image super-resolution methods [52–54] have been proposed, where mapping functions from the low-resolution (LR) patches to high-resolution (HR) patches are learned. Inspired by the great successes of the deep convolution neural network (DCNN) for image classification [51, 55], DCNN models have also been successfully applied to image restoration tasks—e.g., SRCNN [54], FSRCNN [56] and VDSR [57] for image super-resolution, and TNRD [12] and DnCNN [11] for image denoising.

The other important novel insight brought to the field of deep learning is to allow multiple networks to interact with each other in order to optimize the system performance in an end-to-end manner. Examples of such network interaction involves DeepMind’s AlphaGo Zero [58] in which a Monte Carlo Tree Search (MCTS) network improves by playing against itself; generative adversarial network (GAN) [59] in which a generator and a discriminator works together by playing a minimax game; Facebook’s SharpMask algorithm [60] in which the feedforward network gets augmented by for a novel top-down refinement for object segmentation purpose. Inspired by those recent advances, we propose a novel architecture of the denoising network as illustrated in Fig. 2.10. Similar to the U-net [61] and the SharpMask net [60],





**Fig. 2.10** The architecture of the plugged DCNN-based denoiser

the proposed network contains two parts: the feature encoding and the decoding parts. In the feature encoding part, there are a series of convolutional layers followed by pooling layers to reduce the spatial resolution of the feature maps. The pooling layer helps increase the receipt field of the neurons. In the feature encoding stage, all the convolutional layers are grouped into  $L$  feature extraction blocks ( $L = 4$  in our implementation), as shown by the blue blocks in Fig. 2.10b. Each block contains four convolutional layers with ReLU nonlinearity and  $3 \times 3$  kernels. The first three layers generate 64-channel feature maps, while the last layer doubles the number of channels followed by a pooling layer to reduce the spatial resolution of the feature maps with scaling factor 0.5. In the pooling layers, the feature maps are first convoluted with  $2 \times 2$  kernels, and then subsampled by a scaling factor of 2 along both dimensions.

The feature decoding part also contains a series of convolutional layers, which are also grouped into four blocks followed by an upsampling layer to increase the spatial resolution of the feature maps. As the finally extracted feature maps lose a lot of spatial information, directly reconstructing images from the finally extracted features cannot recover fine image details. To address this issue, the feature maps of the same spatial resolution generated in the encoding stage are fused with the upsampled feature maps generated in the decoding stage, for obtaining newly upsampled feature maps. Each reconstruction block also consists of four convolutional layers with ReLU nonlinearity and  $3 \times 3$  kernels. In each reconstruction block, the first three layers produce 128-channels feature maps and the fourth layer generate 512-channels feature maps, whose spatial resolutions are upsampled with a scaling factor of 2 by a deconvolution layer. The upsampled feature maps are then fused with the feature maps of the same spatial resolution from the encoding part. Specifically, the fusion is conducted by concatenating the feature maps. The last feature decoding block reconstructed the output image. A skip connection from the input image to the reconstructed image is added to enforce the denoising network to predict the residuals, which has been verified to be more robust [11].

Note that the DCNN denoisers do not have to be pretrained. Instead, the overall deep network shown in Fig. 2.10a is trained by end-to-end training. To reduce the

number of parameters and thus avoid over-fitting, we enforce each DCNN denoiser to share the same parameters. Mean square error (MSE) based loss function is adopted to train the proposed deep network, which can be expressed as

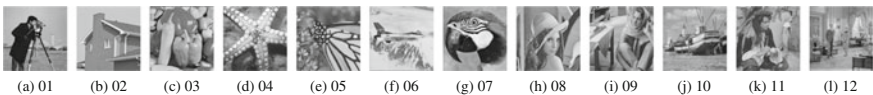
$$\Theta = \underset{\Theta}{\operatorname{argmin}} \sum_{i=1}^N \|\mathcal{F}(y_i; \Theta) - x_i\|_2^2, \quad (2.34)$$

where  $y_i$  and  $x_i$  denote the  $i$ -th pair of degraded and original image patches, respectively, and  $\mathcal{F}(y_i; \Theta)$  denotes the reconstructed image patch by the network with parameter set  $\Theta$ . It is also possible to train the network with other the perceptual based loss functions, which may lead to better visual quality. We remain this as future work. The ADAM optimizer [62] is used to train the network with setting  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$  and  $\epsilon = 10^{-8}$ . The learning rate is initialize as  $10^{-4}$  and halved at every  $2 \times 10^5$  minibatch updates. The proposed network is implemented with framework and trained using 4 Nvidia Titan X GPUs and takes about one day to converge.

### Experimental Results

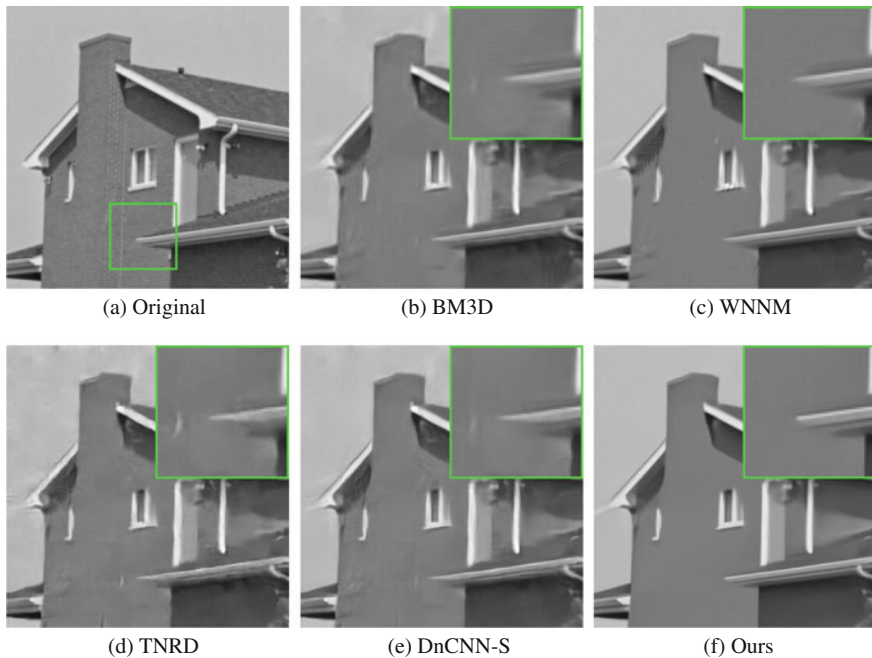
For image denoising,  $\mathbf{A} = \mathbf{I}$  and **Algorithm 1** reduce to the iterative denoising process, i.e., the weighted noise image is added back to the denoised image for the next denoising process. Such iterative denoising has shown improvements over conventional denoising methods that only denoise once [29]. Here, we also found that implementing multiple denoising iterations in the proposed network improves the denoising results. To train the network, we extracted image patches of size  $40 \times 40$  from the training images and added additive Gaussian noise to the extracted patches to generate the noisy patches. Totally,  $N = 450,000$  patches were extracted for training. Note that none of the test images was included into the training image set. The training patches were also augmented by flip and rotations. We compared the proposed network with several leading denoising methods, including three model-based denoising methods, i.e., BM3D method [5], the EPLL method [48], and the low-rank based method WNNM method [8], and two deep learning-based methods, i.e., the TNRD method [12] and the DnCNN-S method [11].

Table 2.2 shows the PSNR results of the competing methods on a set of commonly used test images shown in Fig. 2.11. It can be seen that both the DnCNN-S and the proposed network outperform other methods. For most of the test images and noise levels, the proposed network outperforms the DnCNN-S method, which is the current state-of-the-art denoising method. On average, the PSNR gain over DnCNN-S can be up to 0.32 dB. To further verify the effectiveness of the proposed method, we also employ the Berkeley segmentation dataset (BSD68) that contains 68 natural images



**Fig. 2.11** The test images used for image denoising





**Fig. 2.12** Denoising results for *House* image with noise level 50. The PSNR results: **b** BM3D [5] (29.69 dB); **c** WNNM [8] (30.33 dB); **d** TNRD [12] (29.48 dB); **e** DnCNN-S [11] (30.02 dB); **f** Ours (31.04 dB)

for comparison study. Table 2.3 shows the average PSNR and SSIM results of the test methods on BSD68. One can see that the PSNR gains over the other test methods become even larger for higher noise levels. The proposed method outperforms the DnCNN-S method by up to 0.78 dB on average on the BSD68, demonstrating the effectiveness of the proposed method. Parts of the denoised images by the test methods are shown in Figs. 2.12 and 2.13. One can see that the image edges and textures recovered by model-based methods, i.e., BM3D, WNNM and EPLL are over-smoothed. The deep learning-based methods, TNRD, DnCNN-S, and the proposed method produce much more visually pleasant image structures. Moreover, the proposed method generates even better results in recovering more details than TNRD and DnCNN-S.

## 2.5 Conclusions

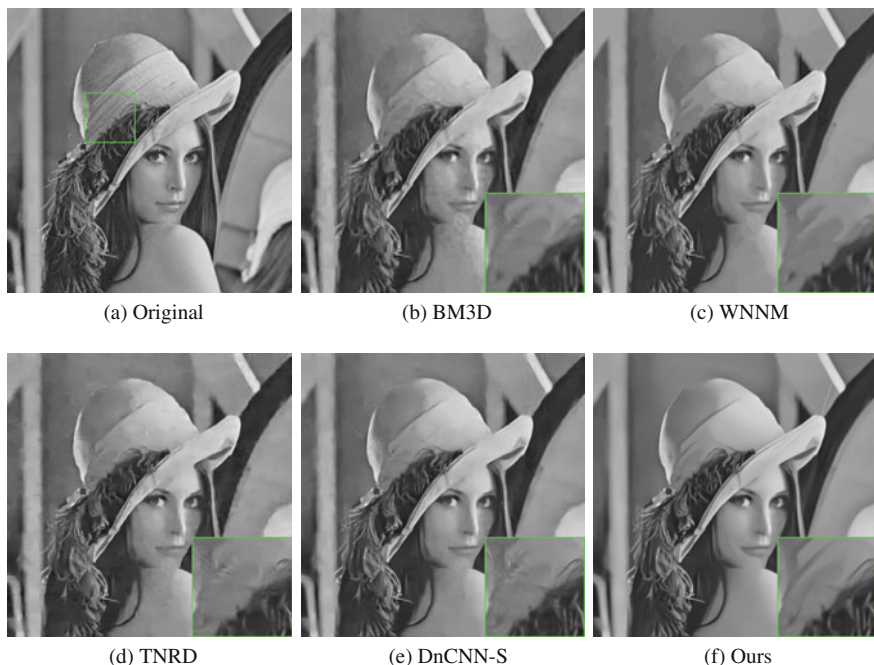
In this chapter, we have reviewed advances in the field of image denoising in the past decades—from model-based including sparse coding and simultaneous sparse coding to learning-based including the most recently developed deep neural net-

Table 2.2 The PSNR (dB) results of the test methods on a set of test images

Image	01	02	03	04	05	06	07	08	09	10	11	12	Avg
$\sigma = 15$													
Noise Level													
BM3D	31.92	34.94	32.70	31.15	31.86	31.08	31.38	34.27	33.11	32.14	31.93	32.11	32.38
WNNM	32.18	35.15	32.97	31.83	32.72	31.40	31.61	34.38	<b>33.61</b>	32.28	32.12	32.18	32.70
EPLL	31.82	34.14	32.58	31.08	32.03	31.16	31.40	33.87	31.34	31.91	31.97	31.90	32.10
TNRD	32.19	34.55	33.03	31.76	32.57	31.47	31.63	34.25	32.14	32.15	32.24	32.11	32.51
DnCNN-S	<b>32.62</b>	35.00	<b>33.29</b>	<b>32.23</b>	33.10	31.70	<b>31.84</b>	34.63	32.65	32.42	32.47	32.47	32.87
<b>Ours</b>	32.44	<b>35.40</b>	33.19	32.08	<b>33.33</b>	<b>31.78</b>	31.48	<b>34.80</b>	32.84	<b>32.55</b>	<b>32.53</b>	<b>32.51</b>	<b>32.91</b>
$\sigma = 25$													
Noise Level													
BM3D	29.45	32.86	30.16	28.56	29.25	28.43	28.93	32.08	30.72	29.91	29.62	29.72	29.98
WNNM	29.64	33.23	30.40	29.03	29.85	28.69	29.12	32.24	<b>31.24</b>	30.03	29.77	29.82	30.26
EPLL	29.24	32.04	30.07	28.43	29.30	28.56	28.91	31.62	28.55	29.69	29.63	29.48	29.63
TNRD	29.71	32.54	30.55	29.02	29.86	28.89	29.18	32.00	29.41	29.92	29.88	29.71	30.06
DnCNN-S	<b>30.19</b>	33.09	30.85	29.40	30.23	29.13	<b>29.42</b>	32.45	30.01	30.22	30.11	30.12	30.43
<b>Ours</b>	30.12	<b>33.54</b>	<b>30.90</b>	<b>29.43</b>	<b>30.31</b>	<b>29.14</b>	29.28	<b>32.69</b>	30.30	<b>30.34</b>	<b>30.15</b>	<b>30.24</b>	<b>30.54</b>
$\sigma = 50$													
Noise Level													
BM3D	26.13	29.69	26.68	25.04	25.82	25.10	25.90	29.05	27.23	26.78	26.81	26.46	26.73
WNNM	26.42	30.33	26.91	25.43	26.32	25.42	26.09	29.25	<b>27.79</b>	26.97	26.94	26.64	27.04
EPLL	26.02	28.76	26.63	25.04	25.78	25.24	25.84	28.43	24.82	26.65	26.72	26.24	26.35
TNRD	26.62	29.48	27.10	25.42	26.31	25.59	26.16	28.93	25.70	26.94	26.98	26.50	26.81
DnCNN-S	27.00	30.02	27.29	25.70	26.77	25.87	<b>26.48</b>	29.37	26.23	27.19	27.24	26.90	27.17
<b>Ours</b>	<b>27.12</b>	<b>31.04</b>	<b>27.44</b>	<b>25.95</b>	<b>27.00</b>	<b>25.97</b>	26.42	<b>29.85</b>	27.21	<b>27.42</b>	<b>27.32</b>	<b>27.23</b>	<b>27.50</b>

**Table 2.3** The PSNR (dB) results of the competing methods on BSD68 image set.

Dataset	$\sigma$	BM3D		EPLL		TNRD		DnCNN-S		Ours	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
BSD68	15	31.08	0.872	31.19	0.883	31.42	0.883	31.74	<b>0.891</b>	<b>32.29</b>	0.888
	25	28.57	0.802	28.68	0.812	28.91	0.816	29.23	<b>0.828</b>	<b>29.88</b>	0.827
	30	25.62	0.687	25.68	0.688	25.96	0.702	26.24	0.719	<b>27.02</b>	<b>0.726</b>

**Fig. 2.13** Denoising results for *Lena* image with noise level 50. The PSNR results: **b** BM3D [5] (29.05 dB); **c** WNNM [8] (29.25 dB); **d** TNRD [12] (28.93 dB); **e** DnCNN-S [11] (29.37 dB); **f** Ours (29.85 dB)

works. During our critical review, we have paid special attention to the new insights that have reshaped our thinking toward the image denoising problem (e.g., from local to nonlocal sparsity, from plain neural networks to deep residue learning). In particular, we have reported state-of-the-art experimental results for learning-based image denoising in two scenarios: one is to learn a sparse parametric model for heavy noise removal and the other is to learn a denoising prior in deep convolutional neural network.

One of the latest advances in the field of deep learning is the development of generative adversarial networks (GAN) [59]. The basic idea behind GAN is to introduce a discriminator to interact with the generator (e.g., denoising can be viewed as gen-

erating a clean image from a noisy observation). When discriminator and generator are jointly trained by a two-player minimax game, GAN makes a step toward unsupervised learning and has demonstrated impressive performance in several applications such as photo-realistic super-resolution [63], low-dose computer-tomography denoising [64], and image-to-image translation [65]. Exploring the potential of GAN for perceptual optimization of image denoising algorithms seems a natural next step along this line of research.

## References

1. Rudin L, Osher S, Fatemi E (1992) Nonlinear total variation based noise removal algorithms. *Physica D* 60:259–268
2. Perona P, Malik J (1990) Scale space and edge detection using anisotropic diffusion. *IEEE Trans Patt Anal Mach Intell.* 12(7):629–639
3. Steidl G, Weickert J, Brox T, Mrázek P, Welk M (2004) On the equivalence of soft wavelet shrinkage, total variation diffusion, total variation regularization, and sides. *SIAM J Numer Anal* 42(2):686–713
4. Portilla J, Strela V, Wainwright M, Simoncelli E (2003) Image denoising using scale mixtures of gaussians in the wavelet domain. *IEEE Trans Image Process* 12:1338–1351
5. Dabov K, Foi A, Katkovnik V, Egiazarian K (2007) Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Trans Image Process* 16:2080–2095
6. Mairal J, Bach F, Ponce J, Sapiro G, Zisserman A (2009) Non-local sparse models for image restoration. In: 2009 IEEE 12th international conference on computer vision, pp 2272–2279
7. Dong W, Shi G, Li X (2013) Nonlocal image restoration with bilateral variance estimation: a low-rank approach. *IEEE Trans Image Process* 22(2):700–711
8. Gu S, Zhang L, Zuo W, Feng X (2014) Weighted nuclear norm minimization with application to image denoising. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2862–2869
9. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. *Nature* 521(7553):436–444
10. Li Y, Dong W, Xie X, Shi G, Li X (2018 to appear) Learning parametric sparse models for heavy noisy removal from images. *IEEE Access*
11. Zhang K, Zuo W, Chen Y, Meng D, Zhang L (2017) Beyond a Gaussian denoiser: residual learning of deep cnn for image denoising. *IEEE Trans Image Process* 26(7):3142–3155
12. Chen Y, Pock T (2017) Trainable nonlinear reaction diffusion: a flexible framework for fast and effective image restoration. *IEEE Trans Pattern Anal Mach Intelligen* 39(6):1256–1272
13. Bae W, Yoo J, Ye JC (2017) Beyond deep residual learning for image restoration: Persistent homology-guided manifold simplification. In: IEEE conference on computer vision and pattern recognition (CVPR) workshops, pp 145–153
14. Pennebaker WB, Mitchell JL (1992) JPEG: still image data compression standard. Kluwer
15. Foi A, Katkovnik V, Egiazarian K (2007) Pointwise shape-adaptive dct for high-quality denoising and deblocking of grayscale and color images. *IEEE Trans Image Process* 16:1395–1411
16. Mallat S (1989) Multiresolution approximations and wavelet orthonormal bases of  $l^2(\mathbf{r})$ . *Trans Amer Math Soc* 315:69–87
17. Daubechies I (1992) Ten lectures on wavelets, vol 61 of CBMS conference series in applied mathematics, Philadelphia, SIAM
18. Candes E (1998) Ridgelets: theory and applications. PhD thesis, Department of Statistics, Stanford University
19. Starck JL, Candes EJ, Donoho DL (2002) The curvelet transform for image denoising. *IEEE Trans Image Process* 670–684

20. Do MN, Vetterli M (2005) The contourlet transform: an efficient directional multiresolution image representation. *IEEE Trans Image Process* 14:2091–2106
21. Mallat S (1999) *A wavelet tour of signal processing*, 2nd ed. Academic Press
22. Chen S, Donoho D, Saunders M (1998) Atomic decomposition by basis pursuit. *SIAM J Scientific Comput* 20:33–61
23. Donoho D (2006) Compressed sensing. *IEEE Trans Infor Theor* 52(4):1289–1306
24. Jolliffe IT (1986) *Principal component analysis*. Springer, Springer Series in Statistics, Berlin
25. Duda R, Hart P, Stork D (2001) *Pattern cclassification*, New York, Wiley, 2nd ed.
26. Aharon M, Elad M, Bruckstein A (2006) *rmK*-SVD: an algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Trans Signal Process* 54(11):4311–4322
27. Elad M, Aharon M (2006) Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Trans Image Proc* 15:3736–3745
28. Dabov K, Foi A, Katkovnik V, Egiazarian K (2006) Image denoising with block-matching and 3D filtering. In: *Proceedings of SPIE electronic imaging: algorithms and systems*, vol 6064, San Jose, CA, USA
29. Dong W, Li X, Zhang L, Shi G (2011) Sparsity-based image denoising via dictionary learning and structural clustering. In: *IEEE conference on computer vision and pattern recognition*
30. Yuan M, Lin Y (2006) Model selection and estimation in regression with grouped variables. *J Royal Statistic Soc Ser B (Statist Methodol)* 68(1):49–67
31. Cotter S, Rao B, Engan K, Kreutz-Delgado K (2005) Sparse solutions to linear inverse problems with multiple measurement vectors. *IEEE Trans Signal Process* 53(7):2477–2488
32. Cai J, Candes E, Shen Z (2010) A singular value thresholding algorithm for matrix completion. *SIAM J Optimiz* 20:1956
33. Andrews DF, Mallows CL (1974) Scale mixtures of normal distributions. *J Royal Statistic Soc Ser B (Statist Methodol)* 36(1):99–102
34. Dong W, Shi G, Ma Y, Li X (2015) Image restoration via simultaneous sparse coding: where structured sparsity meets Gaussian scale mixture. *Int J Comput Vis* 1–16
35. Lyu S, Simoncelli E (2009) Modeling multiscale subbands of photographic images with fields of Gaussian scale mixtures. *IEEE Trans Pattern Anal Mach Intelligen* 31(4):693–706
36. Box GE, Tiao GC (2011) *Bayesian inference in statistical analysis*, vol 40. Wiley
37. Philbin J, Chum O, Isard M, Sivic J, Zisserman A (2007) Object retrieval with large vocabularies and fast spatial matching. In: *IEEE conference on computer vision and pattern recognition, CVPR '07*, pp 1–8. IEEE
38. Dai L, Sun L, Wu E, Yu N (2013) Large scale image retrieval with visual groups. In: *IEEE international conference on image processing (ICIP)*, pp 2582–2586
39. Babenko A, Lempitsky V (2015) Aggregating local deep features for image retrieval. In: *Proceedings of the IEEE international conference on computer vision*, pp 1269–1277
40. Gordo A, Almazán J, Revaud J, Larlus D (2016) Deep image retrieval: learning global representations for image search. In: *European Conference on Computer Vision*, pp 241–257. Springer
41. Lowe DG (2004) Distinctive image features from scale-invariant keypoints. *Int J Comp Vis* 60:91–110
42. Ojala T, Pietikainen M, Maenpaa T (2002) Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Trans Pattern Anal Mach Intell*, pp 971–987
43. Yue H, Sun X, Yang J, Wu F (2013) Landmark image super-resolution by retrieving web images. *IEEE Trans Image Process* 22(12):4865–4878
44. Li Y, Dong W, Shi G, Xie X (2015) Learning parametric distributions for image super-resolution: where patch matching meets sparse coding. In: *Proceedings of the IEEE international conference on computer vision*, pp 450–458
45. Fischler MA, Bolles RC (1987) Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Read Comput Vis*, pp 726–740. Elsevier

46. Yue H, Sun X, Yang J, Wu E (2014) Cid: Combined image denoising in spatial and frequency domains using web images. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2933–2940
47. Dong W, Zhang L, Shi G, Li X (2013) Nonlocally centralized sparse representation for image restoration. *IEEE Trans Image Process* 22(4):1620–1630
48. Zoran D, Weiss Y (2011) From learning models of natural image patches to whole image restoration. In: Proceedings of the ICCV
49. Burger H, Schuler C, Harmeling S (2012) Image denoising: can plain neural networks compete with bm3d? In: 2012 IEEE conference on computer vision and pattern recognition (CVPR), pp 2392–2399
50. Bengio Y, Simard P, Frasconi P (1994) Learning long-term dependencies with gradient descent is difficult. *IEEE Trans Neural Netw* 5(2):157–166
51. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
52. Freeman WT, Jones TR, Pasztor EC (2002) Example-based super-resolution. *IEEE Comput Graph Appl* 22:56–65
53. Timofte R, De Smet V, Van Gool L (2014) A+: adjusted anchored neighborhood regression for fast super-resolution. *Comput Vis–ACCV 2014*, pp 111–126. Springer
54. Dong C, Loy CC, He K, Tang X (2014) Learning a deep convolutional network for image super-resolution. *Comput Vis–ECCV 2014*:184–199
55. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. *Adv Neural Inf Process Syst*, pp 1097–1105
56. Dong C, Loy CC, Tang X (2016) Accelerating the super-resolution convolutional neural network. In: European conference on computer vision, pp 391–407. Springer
57. Kim J, Kwon Lee J, Mu Lee K (2016) Deeply-recursive convolutional network for image super-resolution. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1637–1645
58. Silver D, Schrittwieser J, Simonyan K, Antonoglou I, Huang A, Guez A, Hubert T, Baker L, Lai M, Bolton A et al (2017) Mastering the game of go without human knowledge. *Nature* 550(7676):354–359
59. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial nets. *Adv Neural Inf Process Syst*, pp 2672–2680
60. Pinheiro PO, Lin TY, Collobert R, Dollár P (2016) Learning to refine object segments. In: European conference on computer vision, pp 75–91. Springer
61. Ronneberger O, Fischer P, Brox T (2015) U-net: convolutional networks for biomedical image segmentation. In: International conference on medical image computing and computer-assisted intervention, pp 234–241. Springer
62. Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. In: International conference on learning representations (ICLR)
63. Ledig C, Theis L, Huszár F, Caballero J, Cunningham A, Acosta A, Aitken A, Tejani A, Totz J, Wang Z et al (2016) Photo-realistic single image super-resolution using a generative adversarial network [arXiv:1609.04802](https://arxiv.org/abs/1609.04802)
64. Yang Q, Yan P, Zhang Y, Yu H, Shi Y, Mou X, Kalra MK, Wang G (2017) Low dose ct image denoising using a generative adversarial network with Wasserstein distance and perceptual loss. [arXiv:1708.00961](https://arxiv.org/abs/1708.00961)
65. Isola P, Zhu JY, Zhou T, Efros AA (2017) Image-to-image translation with conditional adversarial networks

# Chapter 3

## Image Denoising—Old and New



Michael Moeller and Daniel Cremers

**Abstract** Image Denoising is among the most fundamental problems in image processing, not only for the sake of improving the image quality, but also as the first proof-of-concept for the development of virtually any new regularization term for inverse problems in imaging. While variational methods have represented the state of the art for several decades, they are recently being challenged by (deep) learning-based approaches. In this chapter, we review some of the most successful variational approaches for image reconstruction and discuss their structural advantages and disadvantages in comparison to learning-based approaches. Furthermore, we present a framework to incorporate deep learning approaches in inverse problem formulations, so as to leverage the descriptive power of deep learning with the flexibility of inverse problems' solvers. Different algorithmic schemes are derived from replacing the regularizing subproblem of common optimization algorithms by neural networks trained on image denoising. We conclude from several experiments that such techniques are very promising but further studies are needed to understand to what extent and in which settings the power of the data-driven network transfers to a better overall performance.

### 3.1 Introduction

Fired by the continuously growing popularity of social media and communication applications, the number of digital photos that is taken everyday is rapidly increasing. While the hardware and with it the quality of the photographs is improving constantly,

---

M. Moeller (✉)  
University of Siegen, Siegen, Germany  
e-mail: michael.moeller@uni-siegen.de

D. Cremers  
Technical University of Munich, Munich, Germany  
e-mail: daniel.cremers@in.tum.de

© Springer International Publishing AG, part of Springer Nature 2018  
M. Bertalmío (ed.), *Denoising of Photographic Images and Video*,  
Advances in Computer Vision and Pattern Recognition,  
[https://doi.org/10.1007/978-3-319-96029-6\\_3](https://doi.org/10.1007/978-3-319-96029-6_3)

the demand for small imaging devices such as smartphones makes it challenging to acquire high-quality images in low-light conditions. Thus, there is an urgent need to digitally remove the noise from such images while keeping the main characteristics of a realistic photograph.

Among the most powerful and well-studied methods for image denoising are energy minimization methods. One defines an energy or cost function  $E$  that depends on the noisy image  $f$ , and maps from a suitable space of candidate images to the real numbers in such a way, that a low number corresponds to an image with desirable properties, i.e., to a realistic and (ideally) noise-free image. Subsequently, one determines a denoised image  $\hat{u}$  as the argument that minimizes  $E$ , i.e.,

$$\hat{u} \in \arg \min_u E(u). \quad (3.1)$$

In Sect. 3.2, we will provide a more systematic derivation of such variational approaches from the perspective of Bayesian inference. In Sect. 3.3, we will then summarize some of the most influential variational denoising methods, along with their underlying assumptions, advantages, and drawbacks.

An entirely different line of research that has become hugely popular and that has shown impressive performance over the past 5 years are data-driven learning-based methods: Whenever a sufficient amount of training data pairs of noisy and noise-free images  $(f^i, u^i)$  are available or can be simulated faithfully, one designs a parameterized function  $G(f; \theta)$  and *learns* the parameters  $\theta$  that lead to the best coincidence of  $G(f^i; \theta)$  with  $u^i$  with respect to some predefined loss  $\mathcal{L}$ . To prevent overfitting, one often defines a regularization  $R$  on the weights  $\theta$  and solves the energy minimization problem

$$\hat{\theta} \approx \arg \min_{\theta} \sum_i \mathcal{L}(G(f^i; \theta), u^i) + \alpha R(\theta). \quad (3.2)$$

Once the above (generally nonconvex) problem has been solved approximately (either by finding a critical point or by stopping early as an additional “regularization”), the inference simply passes new incoming noisy images  $f$  through the network, i.e., computes  $G(f; \hat{\theta})$ . We summarize some influential learning-based approaches to image denoising in Sect. 3.4.

Learning-based strategies are a strong trend in the current literature and they have also been shown to compare favorably in several denoising works. Nevertheless, we are convinced that learning-based strategies alone are not addressing the problem of image denoising exhaustively: First, recent studies question the generalizability of learning-based approaches to realistic types of noise [52]. More importantly, networks can be difficult to train. Solving (3.2) for a highly nested function  $G$  (often consisting of more than 20 layers) requires huge amounts of training data, sophisticated engineering, and good initializations of the parameters  $\theta$  as well as a considerable amount of manual fine-tuning. Since the network architecture and weights may remain fixed during inference, it typically only works in the specific



setting that it has been trained for. Finally, networks do not provide much control and guarantees about the output of the network. Although the training often leads to good results during inference, test images with characteristics different from the training data can easily lead to unpredictable behavior.

In Sect. 3.5, we will analyze these drawbacks in more detail. Moreover, we will analyze whether there is some potential in fusing concepts from energy minimization approaches with concepts from data-driven methods so as to combine the best of both worlds. To this end, we will present a framework for combining learning-based approaches with variational methods. Indeed, preliminary numerical results indicate that the latter holds great promise in addressing some of the aforementioned challenges.

## 3.2 Denoising as Statistical Inference and MAP Estimation

A frequent motivation for energy-based denoising methods of the form (3.1) are maximum a-posteriori probability (MAP) estimates: One aims at maximizing the conditional probability  $p(u|f)$  of  $u$  being the true noise-free image, if one observed the noisy version  $f$ . According to the Bayesian formula, the posterior probability density can be written as

$$p(u|f) = \frac{p(f|u)p(u)}{p(f)}.$$

Instead of looking for the argument  $u$  that maximizes the above expression, by convention one equivalently minimizes its negative logarithm to obtain

$$\hat{u} \in \arg \min_u -\log(p(f|u)) - \log(p(u)). \quad (3.3)$$

The first term contains the probability of observing  $f$  given a true noise-free image  $u$ , and is referred to as the *data fidelity term*. For example, under the assumption of independent zero-mean Gaussian noise with standard deviation  $\sigma$  a spatially discrete formulation gives rise to

$$p(f|u) = \prod_{\text{pixel } i} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(u_i - f_i)^2}{2\sigma^2}\right),$$

leading to the most commonly used  $\ell^2$ -squared term for measuring data fidelity.<sup>1</sup> Many works have investigated the data fidelity terms arising from different distributions of the noise, see [2] for an example considering Poisson noise.

The quest for the right type of data fidelity term for denoising real photographs is, however, quite difficult and camera dependent: The raw sensor data undergoes several processing steps, such as white balancing, demosaicking, color correction/color space

---

<sup>1</sup>For a more detailed spatially continuous formulation, we refer to [22].

transformation, tone mapping, and possibly even compression. Depending on where in this processing chain the denoising is applied, different noise distributions have to be expected. In particular, color space transformations couple the noise over the color channels and demosaicking introduces a spatial correlation [51, 60]. The raw sensor data itself seems to follow a Poisson distribution and (for a reasonably high photon count) is well approximated by a Gaussian distribution with intensity dependent standard deviation—see e.g., [60].

### 3.3 Variational Image Denoising Methods

As suggested by the MAP estimate (3.3), typical energy minimization-based techniques can be written as

$$E(u) = H_f(u) + R(u). \quad (3.4)$$

Here, the *data fidelity term*  $H_f = -\log(p(f|u))$  measures how well the current estimate  $u$  fits to its noisy version  $f$ , while the *regularization*  $R = -\log(p(u))$  imposes consistency with the prior and typically penalizes oscillatory behavior of the noise. While the data fidelity term  $H_f$  can be motivated from the expected distribution of the noise in the data and can often be precalibrated by studying the sensor noise characteristics, the quest for a reasonable prior probability distribution  $p$  of natural images is significantly more challenging. In fact, the modeling of prior probabilities can be expected to benefit tremendously from suitable learning-based approaches such as deep neural networks—see Sect. 3.5.

#### 3.3.1 Total Variation (TV)-Based Image Regularization

Even apart from the interpretation of MAP estimates, researchers have studied the properties of penalty functions  $R$  and their respective influence on the properties of the solution—often in a setting of ill-posed inverse problems in function spaces. Starting with penalties based on Tikhonov regularization, the advantageous properties of non-quadratic regularizations and nonlinear filtering techniques in imaging have been studied from the 1980s on, see the references in [57] for some examples. The total variation (TV) regularization [56, 61] arguably is the most influential work in the field. For images  $u : \Omega \subset \mathbb{R}^2 \rightarrow \mathbb{R}$ , it is defined as [30]

$$|u|_{TV} := \sup_{q \in C_0^\infty, |q(x)| \leq 1} \int_{\Omega} \operatorname{div}(q)(x) u(x) dx. \quad (3.5)$$

It has had an immense success in image denoising, because the functional is convex (enabling the efficient computation of optimal solutions), and because it applies to discontinuous functions  $u$  (enabling the preservation of sharp edges). For continuously differentiable functions  $u$ , the total variation reduces to the integral over  $|\nabla u(x)|$ .

In parallel to the development of TV-based regularization methods, a tremendous amount of research has been conducted on image smoothing using (nonlinear) partial differential equations (PDEs) many of which arise as gradient flows of suitable regularization energies. For the sake of brevity, we will, however, not discuss these methods here.

### 3.3.2 Generalizations: Vectorial TV, Total Generalized Variation

A particularly interesting question for TV-based methods are suitable extensions to color images

$$u : \Omega \subset \mathbb{R}^2 \rightarrow \mathbb{R}^d$$

with  $d$  color channels. Note that (assuming  $u$  to be differentiable) the Jacobian  $Ju$  is a  $2 \times d$  matrix at each point  $x$ , which raises the question in which norm  $Ju(x)$  should be penalized for a suitable extension of the TV to color images. For non-differentiable functions  $u$  the analogue question is the quest for the most natural norm used to bound  $q(x)$  in (3.5). Studies along these directions include the seminal work of Sapiro and Ringach in [58], Blomgren and Chan [7], and a systematic study of different penalties of the Jacobian, e.g., [27]. Instead of using a penalty that strongly couples the color channels, some other lines of research consider

$$\int_{\Omega} \|\nabla C(u)(x)\| dx$$

for a suitable norm  $\|\cdot\|$  and a linear operator  $C$  that changes the color space, e.g., [21], and possibly maps from three to more than three color-related channels, e.g., [3]. All studies agree that the alignment of edges of the RGB channels is of utmost importance to avoid visually disturbing color artifacts.

The success of total variation as a convex regularizer, which can preserve the discontinuities that induced a quest for suitable higher order variants of the TV. To avoid the staircasing effect inherent to TV-based models, a second-order derivative of the input image has to be considered in such a way, that the ability to reconstruct sharp edges is not lost. Higher order TV models include the infimal-convolution regularization [12] as well as the total generalized variation (TGV) [8]. The latter generalizes the total variation in (3.5) as follows:

$$\text{TGV}_{\alpha}^k(u) = \sup \left\{ \int_{\Omega} u \operatorname{div}^k q dx \mid v \in C_c^k(\Omega, \operatorname{Sym}^k(\mathbb{R}^d)), \|\operatorname{div}^l q\|_{\infty} \leq \alpha_l, l=0, \dots, k-1 \right\},$$

where  $\text{Sym}^k(\mathbb{R}^d)$  denotes the space of symmetric tensors of order  $k$  with arguments in  $\mathbb{R}^d$ . Clearly through integration by parts, the higher powers of the divergence operator correspond to higher order derivatives of the function  $u$  being penalized. Whereas the kernel of total variation merely contains the constant functions, the kernel of total generalized variation contains more interesting functions. Second-order TGV, for example, contains the set of affine functions. Combined with the applicability to non-differentiable and discontinuous functions, this makes it well suited for denoising piecewise affine signals.

Similar to the extension of the total variation to color images, extensions of the total generalized variation to color images have been investigated, e.g., in [47]. Note that in its discrete form, the second-order TGV of a color image  $u \in \mathbb{R}^{n_x \times n_y \times d}$  can be written as

$$TGV(u) = \inf_w \|D_1(u) - w\|_* + \|D_2(w)\|_+,$$

where  $D_1$  and  $D_2$  are linear operators approximating suitable derivatives such that  $D_1(u) \in \mathbb{R}^{n_x \times n_y \times d \times 2}$ , and  $D_2(w) \in \mathbb{R}^{n_x \times n_y \times d \times 2 \times 2}$ . Thus, the TGV offers even more freedom in choosing different types of (tensor-based) norms  $\|\cdot\|_*$  and  $\|\cdot\|_+$  for different extensions to color images.

### 3.3.3 Nonconvex Regularizers

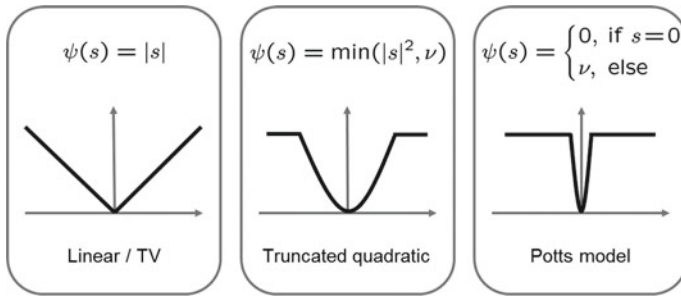
Given the success of total variation type regularizers in preserving sharp discontinuities, one may wonder if respective nonconvex generalizations may be even more suitable in preserving or even enhancing discontinuities.

More specifically, for a one-dimensional function which transitions monotonously between two values  $a < b$ , its total variation is exactly  $b - a$ , independent of how sharp this transition is. Discontinuities are hence associated with a finite penalty corresponding to the size of the step. An often undesired side effect of this property is the tendency of total variation to induce contrast loss.

In order to reduce this contrast loss, iterative techniques such as the Bregman iteration [50] can be considered. Similar ideas have also been investigated in [5], in which it was shown that an image's curvature is easier to reconstruct than the image itself, thus suggesting to use a two-step reconstruction procedure.

A different class of approaches, which can not only preserve but also possibly even enhance discontinuities, penalize the gradient in a sublinear and, therefore, nonconvex manner. In the literature, there exist numerous variants of this idea. Some of the most popular choices can be summarized in a regularization of the form

$$R(u) = \int_{\Omega} \psi(|\nabla u(x)|) dx, \quad (3.6)$$



**Fig. 3.1** Unified representation of various regularizers in the form (3.6) including the convex total variation (left) and the nonconvex truncated quadratic and (as its limiting case) the Potts model. The latter two regularizers essentially correspond to the weak membrane [6] or Mumford–Shah model [48]



**Fig. 3.2** While total variation regularization (center) induces a contrast loss, truncated regularizers like the Mumford–Shah model (right) better preserve discontinuities and contrast. The right image was computed using a convex relaxation of the vectorial Mumford–Shah model proposed in [64]

where typical choices of  $\psi$  include the linear one (absolute norm, i.e., total variation), the truncated linear, the truncated quadratic, and (as the limiting case of the previous two) the Potts model (which penalizes any nonzero gradient with a constant value). See Fig. 3.1 for a visualization. The truncated quadratic regularizer essentially corresponds to the Mumford–Shah model [48]. Such truncated regularizers are likely to preserve contrast because discontinuities are penalized with a constant cost  $\nu$  independent of their size. This is indeed confirmed in the example in Fig. 3.2.

The Mumford–Shah model has been studied intensively in the applied mathematics literature, because it is an interesting hybrid between a denoising and a segmentation approach. It is defined as follows:

$$E(u) = \int_{\Omega} (f(x) - u(x))^2 dx + \lambda \int_{\Omega \setminus S_u} |\nabla u|^2 dx + \nu \mathcal{H}^1(S_u). \quad (3.7)$$

The aim is to approximate the input image  $f : \Omega \subset \mathbb{R}^2 \rightarrow \mathbb{R}$  in terms of a piecewise smooth function  $u : \Omega \rightarrow \mathbb{R}$ . The functional contains a data fidelity term and two regularity terms imposing smoothness of  $u$  in areas separated by the discontinuity set  $S_u$  and regularity of  $S_u$  in terms of its one-dimensional Hausdorff measure

$\mathcal{H}^1(S_u)$ . Related approaches were proposed in a spatially discrete setting by Geman and Geman [28] and by Blake and Zisserman [6]. The two regularizers in (3.7) clearly correspond to the truncated quadratic penalty above in the sense that energetically image locations, where  $\lambda|\nabla u|^2 > \nu$  will be associated with the discontinuity set  $S_u$  and hence, are assigned a cost  $\nu$ .

Sublinear penalties of the gradient norm are also more consistent with the statistics of natural images. Based on the observation that the regularizer is nothing but the negative logarithm of the prior—see Sect. 3.2—one can study the statistics of gradient filter responses on natural images [35]. These statistics show heavy-tailed distributions, which correspond to sublinear regularizers. An alternative representation of sublinear regularizers are the so-called TV- $q$  models defined as

$$\text{TV}_q(u) := \int_{\Omega} |\nabla u(x)|^q dx, \quad (3.8)$$

where for  $q < 1$  the gradient is penalized sublinearly.

A challenging problem for the actual implementation of the aforementioned non-convex variants of the total variation regularization is their optimization, in which one can only hope to determine local minimizers. While provably convergent methods typically have to rely on smoothing the non-differentiable, nonconvex part, several works have shown very promising behavior of splitting techniques such as the alternating directions method of multipliers (ADMM), e.g., [16, 17, 73], or primal versions of primal-dual algorithms, e.g., [46, 65, 66]. We refer the interested reader to [70] to a recent summary on the convergence of the ADMM algorithm in a nonconvex setting.

### 3.3.4 Non-local Regularization

The most notable improvement—particularly for the problem of image denoising—was the development of non-local smoothing methods, starting with the non-local means (NLM) algorithm [4, 10]: Based on the idea that natural images are often self-similar, one denoises images by first computing the similarity of pixels in a robust way, e.g. by comparing image patches, and subsequently determines the value of each denoised pixel by a weighted average based on pixel similarities. By considering the first step, i.e., the estimation of pixel similarities, as the formation of an image-dependent graph, regularization methods based on (different variants of) graph Laplacians were developed, see e.g., [39] for details. The extension of non-local methods to TV regularization was proposed in [29].

One of the most popular and powerful denoising algorithms is the block matching 3D (BM3D) algorithm [23], which is based on very similar assumptions as the above self-similarity methods, but sacrifices the interpretability in terms of a regularization function for a more sophisticated filtering strategy of patches that are considered to be similar. In particular, it estimates a first denoised version of an image to then recompute the similarity between pixels/patches, and denoises again. An interpreta-

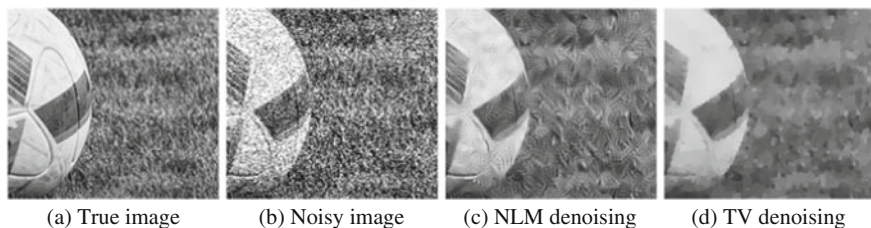
tion in terms of a frame-based regularization in a variational framework was given later in [25]. Further prominent extensions and improvements are based on learning the likelihood of natural image patches [76], and exploiting the low-rank structure of similar image patches using weighted nuclear norm minimization [31].

While the above methods are based on the assumption that every patch in an image has multiple similarly looking variants, the idea of *sparse representations* and *dictionary learning* relaxes this constraint. It merely demands that each patch can be represented as a linear combination of a few suitable patches from an overcomplete dictionary. The latter can not only be learned from a representative dataset, but also even on the image to be reconstructed itself, with the K-SVD algorithm [1] being one of the most popular and powerful numerical methods for tackling the underlying nonconvex energy minimization problem. Hybrid self-similarity and dictionary learning techniques have been developed in [43], and a focus on dictionary learning for color image reconstruction has been set in [44].

### 3.3.5 A Discussion Within Classical Denoising Methods

Before we discuss the extension of the partially data-driven model of dictionary learning to the mostly data-driven methods, let us compare the denoising methods we have recalled so far.

TV-type regularization methods are based on rather weak regularity assumptions and can, therefore, be applied to a wide range of different applications and types of images. Second- order extensions such as TGV often improve upon TV, while still depending on (weak) regularity assumptions only. The price for such improvements is an additional hyperparameter as well as a more complex minimization problem. Non-local methods such as NLM and BM3D rely on the reconstructed images to be self-similar. While they often improve the results of local methods significantly, a faithful estimate of pixel/patch similarity is required. In settings of inverse problems where such an estimate is difficult to obtain, or in cases of strong noise in which similarity estimates become unreliable, such methods come at the risk of hallucinating self-similar structures as illustrated in Fig. 3.3—also see [9, Fig. 6].



**Fig. 3.3** Illustrating a failure of the self-similarity-based NLM algorithm in a case, where a faithful estimate of pixel correspondences is impossible: the NLM- denoised image (c) contains strong artificial structures. While TV denoising is also unable to reconstruct the grass, it erases all high frequencies instead of hallucinating structures

Nonconvex variants of the above, e.g.,  $\text{TV}^q$  or  $\text{TGV}^q$  models, can improve the results of their convex relatives—particularly in the presence of strong edges—but do come with the usual drawbacks of nonconvex optimization: No algorithm can guarantee not to get stuck in a bad local minimum. Similarly, dictionary learning approaches such as the K-SVD algorithm are based on nonconvex optimization problems and exploit a particular structure of the data, i.e., the ability to represent each patch as a sparse linear combination of a few (learned) dictionary atoms.

In general, the regularizing quality of the above approaches greatly improves with the strength of the assumptions that are made. This leads to self-similarity and dictionary-based techniques clearly being the more powerful choice in usual practical settings of small or moderate noise and natural images. Strong assumptions can, however, influence the result in a very undesirable way if they do not hold, as we have illustrated in Fig. 3.3. This makes the simpler (local) models possibly more attractive in applications where a structurally systematic error in the reconstruction can have dramatic consequences, e.g., in the field of medical imaging.

### 3.4 Learning-Based Denoising Methods

In recent years researchers have had great success in replacing the implicit characterization of solutions as arguments that minimize a suitable energy function by explicit functions that directly map the input data to the desired solution. In the case of image denoising, such functions typically take the form

$$\begin{aligned} G : \mathbb{R}^{n \times m \times c} \times \mathbb{R}^k &\rightarrow \mathbb{R}^{n \times m \times c} \\ (f, \theta) &\mapsto G(f, \theta) \end{aligned} \quad (3.9)$$

where  $f \in \mathbb{R}^{n \times m \times c}$  is the noisy input image,  $G(f, \theta)$  is the denoising result and  $\theta \in \mathbb{R}^k$  are weights that parametrize the function  $G$ . The latter are determined during *training*, which is the approximate solution of problem (3.2) for a suitable loss function  $\mathcal{L}$ , e.g., the  $\ell^2$ -squared loss, when optimizing for high peak signal-to-noise ratios (PSNRs). The pairs  $(f^i, u^i)$  of noisy and clean images used during training have to be representative for the setting the network is used in during inference, i.e., the types of images and the type of noise used for the training should originate from the same distribution as the test images.

The typical architecture of a *network*  $G$  is a nested function

$$G(f, \theta) = g_L(g_{L-1}(\dots g_2(g_1(f, \theta^1), \theta^2) \dots, \theta^{L-1}), \theta^L), \quad (3.10)$$

where each function  $g_i$  is referred to as a *layer*. The most common layers in basic architectures are parameterized affine *transfer functions* followed by a nonlinearity called *activation function*.

The specific architecture of  $G$  and its individual layers has evolved over the past years. The first networks to challenge the previous dominance of BM3D and



K-SVD-type algorithms were *fully connected* using tangens hyperbolici as activation functions [11], e.g.,

$$\begin{aligned} g_i(x, \theta^i) &= \tanh(\theta^i[x; 1]), \quad \forall i \in \{1, \dots, L-1\}, \\ g_L(x, \theta^L) &= \theta^L[x; 1]. \end{aligned}$$

Small vectorized image patches of a noisy image are fed into the network. In each layer, a 1 is attached to the input vector to allow for an offset, typically called *bias*. A crucial aspect of these powerful learning-based denoising approaches was a comparably large number of layers, relating to the overall trend of developing deep neural networks.

The work [72] proposed a sparse autoencoder architecture, also using fully connected layers and sigmoid activation functions. While [11] and [72] performed on par with BM3D and K-SVD on removing Gaussian noise, respectively, architectures based on convolutions, e.g., [37], or more recently convolutions with rectified linear units as activations, i.e.,

$$g_i(x, \theta^i) = \max(\theta_k^i * x + \theta_b^i, 0), \quad \forall i \in \{1, \dots, L-1\},$$

have shown promising results, e.g., in [42, 74]. Moreover, [74] proposed the idea of deep residual learning to the field of image denoising, i.e., the strategy of learning to output the estimated noise instead of the noise-free image itself.

Recent learning techniques such as [20, 41, 69] furthermore exploit the idea to filter image patches in (non-local) groups to mimic and improve upon the behavior of their designed relatives such as BM3D.

Besides a focus on more realistic types of noise (as pointed out in [52]), a promising direction for future denoising networks is to move from the (PSNR-optimizing)  $\ell^2$ -squared loss function to perceptual [38], or GAN-based [40] loss functions that are able to reflect the subjective quality perception of the human visual system much more accurately.

Beyond the specific architecture and training of networks, further improvements can be made by tailoring denoising networks to specific classes of images determined by a prior classification network, see [53].

A drawback of most learning-based approaches is that they are trained on a specific type of data, as well as a specific type and strength of noise. Thus, whenever one of these quantities changes, an expensive retraining is required. Although promising approaches for a more generic use of neural networks for varying strengths of Gaussian noise exist, see, e.g., [71, 74], retaining a high-quality solution over varying types of noise remains challenging.

In the next section, we will discuss hybrid learning and energy minimization-based approaches which represent a promising class of methods to not only adapt to different types of noise but also even to different types of restoration problems easily.

## 3.5 Combining Learning and Variational Methods

### 3.5.1 Lacking Flexibility

Deep neural networks have proven to be extraordinarily effective for a wide range of high- and low-level computer vision problems. Their effectiveness does, however, come at the costs of a complicated and expensive training procedure in which aspects such as different training algorithms, hyperparameters, initializations (e.g., [32]), dropout [63], dropconnect [68], batch-normalization [36], or the introduction of shortcuts such as in ResNet [33], have to be considered to achieve good results. Moreover, networks often do not generalize well beyond the specific type of data they have been trained on. In the case of image denoising, for example, the authors of [52] showed that the classical BM3D algorithm yields better denoising results on real photograph than state-of-the-art deep learning techniques that were all trained on Gaussian noise.

While one might argue that the dominance of learning-based approaches can be reestablished by training on more realistic datasets, several drawbacks remain:

1. Neural networks often do not generalize well beyond the specific setting they have been trained on. While approaches for training on a variety of different noise levels exist (e.g., in [71, 74]), networks typically cannot address arbitrary image restoration problem of reconstructing an image  $u$  from noisy data  $f \approx Ku$  for a linear operator  $K$ , if the operator  $K$  was not already known during training time. Typically, every time the type of noise, the strength of noise, or the linear operator  $K$  changes, neural networks require additional training.
2. The separability of the data formation process from the regularization, and hence the negative log-likelihood of the distribution of “natural images”, is lost in usual deep learning strategies despite the fact that learning-from-data seems to be the only way to realistically give a meaning to the term “natural images” in the first place.
3. Even though a network might be trained on returning  $u^i$  for a given measurement  $f^i = Ku^i + n^i$  for noise  $n^i$  drawn from a suitable distribution, there is no guarantee for the networks output  $G(f, \hat{\theta})$  to follow the data formation  $KG(f, \hat{\theta}) \approx f$  during inference, i.e., there is no *guarantee* for the output to be a reasonable explanation of the data.

On the other hand, one can constitute that

1. Variational methods have a plug-and-play nature in which one merely needs to adapt the data fidelity term  $H_f$  as the strength or type of noise or the linear operator  $K$  changes.
2. They clearly separate the data fidelity term from the regularization with each of the two being exchangeable.
3. The proximity to a given forward model can easily be guaranteed in variational methods by using suitable indicator functions for  $H_f$ .

4. Despite the above advantages, the expressive power of regularizations terms to measure how “natural” or “realistic” a given image is, is very limited. In particular, local (e.g., total variation based) or non-local smoothness properties do not capture the full complexity of textures and structures present in natural images. In fact, exploiting large databases seems to be the most promising way for even defining what “natural images” are.

The complementary advantages of both methods indicate that strategies for combining variational- and learning-based techniques constitute an attractive field of research. But how can we systematically derive methods which combine the best of both worlds?

Considering the derivation of energy minimization methods from MAP estimates in Sect. 3.2, it seems natural to estimate  $p(u)$  in (3.3) from training images. This, however, means estimating a probability distribution of natural images in a number-of-pixel dimensional space, which seems to be extremely ambitious even for moderately sized images. In fact, the knowledge of such a distribution would allow to sample natural images—a task researchers currently try to tackle with generative adversarial networks (GANs), but still face many difficulties, e.g., for generating high-resolution images. We refer the reader to [26, 49] for recent approaches that tackle inverse problems by estimating the distribution of natural images.

### 3.5.2 Learning the Regularizer

Researchers have already considered the general idea to learn the probability distribution of natural images more than a decade ago by settling for the probability distribution of separate patches, assuming a particular form of the underlying probability distribution, see the field of experts model by Roth and Black [55] for an example. While the latter actually tries to approximate the probability distribution of training data by combining a gradient ascent on the log-likelihood with a Monte Carlo simulation, the work [18] by Chen et al. proposes a different strategy: They show that an analysis-based sparsity regularization of the form

$$R(u, A) = \sum_{\text{patches } u_p} \sum_{\text{filters } A_i} \Phi(A_i * u_p) \quad (3.11)$$

is equivalent to the negative log-likelihood in the field of experts model. In the above,  $\Phi$  denotes a robust penalty function such as  $\log(1 + z^2)$  and  $A_i * u_p$  are the convolution of a filter  $A_i$  with the image patch  $u_p$ . For finding suitable filters  $A$  the authors, however, propose a bilevel optimization framework, which—in the case of image denoising—takes the form

$$\min_A \left( \sum_{\text{training examples } (\hat{u}^j, f^j)} \|u^j(A) - \hat{u}^j\|_2^2 \right), \quad (3.12)$$

subject to  $u^j(A) \in \arg \min_u \lambda \|u - f^j\|_2^2 + R(u, A)$ ,

for pairs  $(\hat{u}^j, f^j)$  of noise-free and noisy training images  $\hat{u}^j$  and  $f^j$ , respectively.

Although the problem (3.12) is difficult to solve, the results in [18] are promising, the approach retains the interpretation of an energy minimization method, and the regularization can potentially generalize to arbitrary image restoration problems. Its limitation is, however, given by the manual choice and specific parametrization of  $R$  in (3.11).

### 3.5.3 Developing Network Architectures from Optimization Methods

For the sake of more freedom, the authors of [19] considered the minimization of energies like (3.4) for a parameterized regularization  $R$  with learnable weights. By using a gradient descent iteration, a discretization of a reaction–diffusion type of equation is obtained in which the authors, however, allow the parameterized regularization to change at each iteration of their scheme. Note that although this does not allow the interpretation as an energy minimization method anymore, it led to improved denoising performances.

Similarly, in [59] Schmidt and Roth construct a method based on minimizing

$$\frac{1}{2} \|Ku - f\|_2^2 + \sum_{\text{filters } A_i} \sum_{\text{patches } u_p} \rho_i(A_i * u_p),$$

where  $\rho_i$  are suitable penalty functions to be learned. By considering a half-quadratic splitting that minimizes

$$E(u, z) = \frac{1}{2} \|Ku - f\|_2^2 + \sum_{\text{filters } A_i} \sum_{\text{patches } u_p} \rho_i(z_{i,p}) + \frac{\beta}{2} (z_{i,p} - A_i * u_p)^2, \quad (3.13)$$

for  $u$  and  $z$  in an alternating fashion, the update for  $u$  becomes a simple linear equation. The update for  $z$  reduces to what the authors call *shrinkage function* in [59], and which is called *proximal operators* in the optimization community. The proximal operator of a (typically proper, closed, convex) function  $R : \mathbb{R}^n \rightarrow \mathbb{R} \cup \{\infty\}$  is defined as

$$\text{prox}_R(h) = \arg \min_v \frac{1}{2} \|v - h\|^2 + R(v). \quad (3.14)$$

In the case of minimizing for  $z$  in (3.13), all  $\rho_i$  are functions from  $\mathbb{R}$  to  $\mathbb{R}$ , and so are  $\text{prox}_{\rho_i}$ . At this point, the idea of Schmidt and Roth is twofold:

1. They propose to parameterize and *learn* the proximal operators  $\text{prox}_{\rho_i}$  instead of the functions  $\rho_i$ , and even intentionally drop the constraint that the learned operators  $r_i$  must correspond to proximal operators. In fact, it is shown in [59] that the final  $r_i$  provably cannot be proximal operators anymore.
2. They allow the learned operators  $r_i$  to change in each iteration of the half-quadratic minimization.

By changing the operator in each step, the resulting algorithmic scheme does not resemble the structure of a minimization algorithm anymore. By omitting the monotonicity constraint which is necessary to even be able to identify an operator as the proximal operator of a function, not even a single iteration of the respective scheme can be interpreted as an energy minimization step. Nevertheless, as the training basically ‘roles out’ the algorithm for a fixed number of iterations, the paper naturally resembles a (deep) neural network, whose architecture is motivated by the half-quadratic minimization method.

The methods from [19, 59] yield a nice motivation for the (otherwise somewhat handcrafted) architecture of a neural network, and both allow the extension from image denoising to more general linear inverse problems. Because both works, [59] and [19], do, however, have changing operators in each iterations and ‘roll out’ the iterations during training, they cannot be interpreted as an iteration yielding a (hopefully convergent) sequence of iterates as usual minimization algorithms. Moreover, the end-to-end training of the resulting algorithmic schemes still tailors the parameters to the specific setting (i.e., the specific operator  $K$ , type, and strength of noise) they have been trained on.

### 3.5.4 Algorithmic Schemes Based on Learned Proximal Operators

To avoid the aforementioned drawbacks recent research [15, 45, 54, 75] has considered fully decoupling the data formation process from learning a function that introduces the required regularity. All these approaches develop algorithmic schemes based on classical optimization methods and replace the proximal operator of the regularization by a neural network. The general idea originates from previous publications [24, 34, 67, 76] in which general algorithmic schemes were developed by replacing the proximal operator of the regularization by denoising algorithms such as BM3D or NLM. The premise that learning-based approaches have the power to learn even more complex smoothness properties than the non-local similarities captured by NLM and BM3D subsequently motivated the introduction of neural networks. Let us review some of these ideas in more detail.

### 3.5.4.1 Deriving Different Schemes

As a motivation consider the problem of minimizing (3.4), i.e.,

$$\min_u H_f(u) + R(u), \quad (3.15)$$

where the data term shall remain flexible as in usual variational methods, but the regularization shall be replaced by a data-driven approach in order to benefit from the power of learning-based strategies.

Following the idea of half-quadratic splitting, we have already seen in (3.13), we could replace (3.15) by

$$\min_{u,z} H_f(u) + R(z) + \frac{\beta}{2} \|z - u\|^2, \quad (3.16)$$

which—under mild conditions—yields a minimizer of (3.15) for  $\beta \rightarrow \infty$ .

By applying alternating minimization to (3.16), one has to solve

$$\begin{aligned} u^{k+1} &= \text{prox}_{\frac{1}{\beta} H_f}(z^k), \\ z^{k+1} &= \text{prox}_{\frac{1}{\beta} R}(u^{k+1}). \end{aligned} \quad (3.17)$$

As such an algorithm decouples the regularization from the data fidelity term, it is natural to replace the proximal operator of the regularization by a neural network. Based on the fact that the proximal operator of a regularization represents a denoising procedure, or—in the extreme case—the projection onto a natural feasible set of natural images, researchers have trained respective networks to perform exactly these tasks, see [15, 45, 75]. In the above example of half-quadratic splitting, the resulting algorithmic scheme becomes

$$\begin{aligned} u^{k+1} &= \text{prox}_{\frac{1}{\beta} H_f}(z^k), \\ z^{k+1} &= G(u^{k+1}; \hat{\theta}), \end{aligned} \quad (3.18)$$

for a network  $G$  that has been trained on denoising or, more generally, on “making the image more realistic”.

To illustrate the flexibility and possible advantages of such a scheme, consider Fig. 3.4. Shown in (a) is an image contaminated with salt-and-pepper noise. If one applies a network that has been trained on removing Gaussian noise, one obtains the middle image (b) in Fig. 3.4. While some of the texture of the fur of the giraffe is smoothed out, almost no noise is removed. While the typical learning-based approach is to retrain the network on example images with simulated salt-and-pepper noise, it is well known that energy minimization methods can handle such a type of noise efficiently by using a robust data fidelity term such as the  $\ell^1$  norm. Figure 3.4c shows that applying iteration (3.18) with the very same network as in (b) and  $H_f(u) = \|u - f\|_1$  is able to remove the noise almost perfectly while preserving the underlying texture.



**Fig. 3.4** A network that has been trained on removing Gaussian noise, often does not generalize well to other types of noise. Feeding the image (a) with salt-and-pepper noise into such a network results in image (b). Instead of retraining the network on salt-and-pepper noise, one can exploit the idea of (3.18) with a robust data fidelity term to obtain significantly better results using the very same network as in (b)—see (c)

The above idea and derivation of the algorithmic scheme is of course not limited to the method of half-quadratic splitting, but actually applies to almost any minimization method for (3.16). Due to its flexibility in handling multiple terms, the alternating direction method of multipliers (ADMM) and preconditioned variants thereof have mostly been used in this context, see [15, 24, 34, 45, 54, 67, 75, 76]. Since ADMM is known to not necessarily converge on nonconvex problems, this choice does not seem to be natural considering that approaches that replace a proximal operator by an arbitrary function are even beyond the setting of nonconvex minimization.

In Table 3.1 we provide an overview of a wide variety of different optimization methods and their corresponding algorithmic schemes that could be used in the very same fashion. Note that we not only considered replacing the proximal operator of the regularization with a neural network, but also its explicit counterpart—the explicit gradient descent step on  $R$ ,

$$u - \tau \nabla R(u) \rightarrow G(u, \hat{\theta}).$$

This, for instance, leads to the algorithmic schemes of *proximal gradient 2* and *HQ splitting* to coincide despite originating from different optimization algorithms, which do not even converge to the same minimizer in a suitable convex setting.

**Table 3.1** Different algorithms for minimizing  $H_f(u) + R(u)$  and the corresponding algorithmic schemes that replace explicit or implicit (proximal) gradient steps on the regularization by a neural network  $G$ . For the *Primal-dual 2* algorithm we assumed that  $H_f = T_f \circ K$  for a linear operator  $K$ . Note that—even in a convex setting with some additional assumptions—the *HQ splitting* algorithm does not converge to a minimizer of  $H_f(u) + R(u)$  but rather replaces  $R$  or  $H_f$  by the (smoother) Moreau envelope. The choice  $\beta \rightarrow \infty$  can usually reestablish the convergence

Method	Iteration	Algorithmic scheme
Gradient descent	$z^1 = u^k - 2\tau \nabla H_f(u^k),$ $z^2 = u^k - 2\tau \nabla R(u^k),$ $u^{k+1} = \frac{1}{2}(z^1 + z^2)$	$z^1 = u^k - 2\tau \nabla H_f(u^k),$ $z^2 = G(u^k; \hat{\theta}),$ $u^{k+1} = \frac{1}{2}(z^1 + z^2)$
Proximal gradient 1	$z^k = u^k - \tau \nabla H_f(u^k),$ $u^{k+1} = \text{prox}_{\tau R}(z^k)$	$z^k = u^k - \tau \nabla H_f(u^k),$ $u^{k+1} = G(z^k; \hat{\theta})$
Proximal gradient 2	$z^k = u^k - \tau \nabla R(u^k),$ $u^{k+1} = \text{prox}_{\tau H_f}(z^k)$	$z^k = G(u^k; \hat{\theta}),$ $u^{k+1} = \text{prox}_{\tau H_f}(u^k)$
HQ splitting	$u^{k+1} = \text{prox}_{\frac{1}{\beta} H_f}(z^k),$ $z^{k+1} = \text{prox}_{\frac{1}{\beta} R}(u^{k+1})$	
ADMM	$u^{k+1} = \text{prox}_{\frac{1}{\beta} H_f}(z^k - p^k),$ $z^{k+1} = \text{prox}_{\frac{1}{\beta} R}(u^{k+1} + p^k),$ $p^{k+1} = p^k + u^{k+1} - z^{k+1}$	$u^{k+1} = \text{prox}_{\frac{1}{\beta} H_f}(z^k - p^k),$ $z^{k+1} = G(u^{k+1} + p^k; \hat{\theta}),$ $p^{k+1} = p^k + u^{k+1} - z^{k+1}$
Primal-dual 1	$p^{k+1} = p^k + \beta \bar{u}^k$ $-\beta \text{prox}_{\frac{1}{\beta} R} \left( \frac{p^k}{\beta} + \bar{u}^k \right),$ $u^{k+1} = \text{prox}_{\tau H_f}(u^k - \tau p^{k+1}),$ $\bar{u}^{k+1} = u^{k+1} + (u^{k+1} - u^k)$	$p^{k+1} = p^k + \beta \bar{u}^k$ $-\beta G \left( \frac{p^k}{\beta} + \bar{u}^k; \hat{\theta} \right),$ $u^{k+1} = \text{prox}_{\tau H_f}(u^k - \tau p^{k+1}),$ $\bar{u}^{k+1} = u^{k+1} + (u^{k+1} - u^k)$
Primal-dual 2	$z^{k+1} = z^k + \beta K \bar{u}^k$ $-\beta \text{prox}_{\frac{1}{\beta} T_f} \left( \frac{z^k}{\beta} + K \bar{u}^k \right),$ $p^{k+1} = p^k + \beta \bar{u}^k,$ $-\beta \text{prox}_{\frac{1}{\beta} R} \left( \frac{p^k}{\beta} + \bar{u}^k \right),$ $u^{k+1} = u^k - \tau K^T z^{k+1} - \tau p^{k+1},$ $\bar{u}^{k+1} = u^{k+1} + (u^{k+1} - u^k)$	$z^{k+1} = z^k + \beta K \bar{u}^k$ $-\beta \text{prox}_{\frac{1}{\beta} T_f} \left( \frac{z^k}{\beta} + K \bar{u}^k \right),$ $p^{k+1} = p^k + \beta \bar{u}^k,$ $-\beta G \left( \frac{p^k}{\beta} + \bar{u}^k; \hat{\theta} \right),$ $u^{k+1} = u^k - \tau K^* z^{k+1} - \tau p^{k+1},$ $\bar{u}^{k+1} = u^{k+1} + (u^{k+1} - u^k)$

While the *HQ splitting* algorithm is unconditionally stable it converges to a minimizer of a smoothed version of the original energy (replacing  $R$ , respectively,  $H_f$ , by



its Moreau envelope). The *proximal gradient 2* algorithm on the other hand requires a step size  $0 < \tau < \frac{2}{L}$  with  $L$  being the Lipschitz constant of  $\nabla R$  to converge to a minimizer of  $H_f + R$ . This along with the long list of possible algorithmic schemes in Table 3.1, which could further be extended by the corresponding methods with inertia/momentum, raises the question which method should be used in practice. An exhaustive answer to this question (if it can be provided at all) requires a tremendous number of experiments involving different problems, different networks, data terms, parameters, and initializations, and goes beyond the scope of this paper. We do, however, provide some first experiments involving all algorithms in Sect. 3.6.

### 3.5.4.2 Hyperparameters of the Algorithmic Schemes

When comparing algorithmic schemes like *Proximal gradient 1* with their optimization algorithm counterpart, one observes that replacing the proximity operator with a neural network eliminates the step size parameter  $\tau$ . The missing dependence of the “regularization-step” on  $\tau$  in the *Proximal gradient 1* scheme means that the step size  $\tau$  merely rescales the data fidelity term: The resulting algorithmic scheme may always pick  $\tau = 1$ , i.e., eliminate the step size completely, and interpret any  $\tau \neq 1$  as a part of  $H_f$ , see [45] for details. Interestingly, even simple choices like the function  $G$  being the identity may lead to divergent algorithmic schemes for large data fidelity parameters. This may motivate training a network function  $G$  on a rather small noise level such that even moderate data fidelity parameters can lead to a large emphasis on data fidelity over the course of the iteration. Note that—at least in the context of optimization—the aforementioned difficulties can be avoided by an implicit treatment of the data fidelity term as arising in the *Proximal Gradient 2* or *HQ splitting* algorithms.

The elimination of hyperparameters in the algorithmic schemes becomes even more interesting for the more sophisticated primal-dual and ADMM-based schemes. Note that the parameter  $1/\beta$  in the ADMM scheme also merely rescales the data fidelity term. As shown in [45], in the *primal-dual 1* scheme, we can define  $\tilde{p} = p/\beta$  to arrive at the update equations

$$\tilde{p}^{k+1} = \tilde{p}^k + \tilde{u}^k - G(\tilde{p}^k + \tilde{u}^k; \hat{\theta}), \quad (3.19)$$

$$u^{k+1} = \text{prox}_{\tau H_f}(u^k - \tau\beta\tilde{p}^{k+1}), \quad (3.20)$$

$$\tilde{u}^{k+1} = u^{k+1} + (u^{k+1} - u^k). \quad (3.21)$$

In this case, the parameter  $\tau$  scales the data fidelity term, but the product of  $\tau$  and  $\beta$  remains a factor for  $\tilde{p}^{k+1}$  in the update of  $u^{k+1}$ . In the world of convex optimization, the product  $\tau\beta$  has to remain smaller than the operator norm of the linear operator used in the primal-dual splitting, which—in our specific case of *primal-dual 1*—is the identity. Due to the equivalence of ADMM and the primal-dual algorithm in a convex setting, the largest value of the product  $\beta\tau$  for which convergence can still be guaranteed is 1, which is also the choice we make in all our numerical experiments below. This allows to again eliminate  $\tau$  completely as it merely rescales the data fidelity term  $H_f$ .

A similar computation allows to reduce the *primal-dual 2* scheme to a rescaling of  $T_f$  by  $1/\beta$  and the product of  $\tau\beta$ . Since this splitting involves the linear operator of two stacked identities, the step size restriction in the convex setting would be  $\tau\beta < 1/\sqrt{2}$ . We chose  $\tau\beta = 0.5$  in all our experiments.

### 3.5.4.3 Algorithm Equivalence

A particularly interesting aspect in the above discussion is the equivalence of ADMM and primal-dual in the convex setting. Considering the ADMM scheme from Table 3.1, we notice that

$$z^k = u^k - p^k + p^{k-1}$$

such that the update in  $u^{k+1}$  can equivalently be written as

$$u^{k+1} = \text{prox}_{\frac{1}{\beta}H_f}(u^k - (2p^k - p^{k-1})).$$

By entirely eliminating the variable  $z$  from the update equations, we arrive at the equivalent form

$$u^{k+1} = \text{prox}_{\frac{1}{\beta}H_f}(u^k - (2p^k - p^{k-1})), \quad (3.22)$$

$$p^{k+1} = p^k + u^{k+1} - G(p^k + u^{k+1}; \hat{\theta}). \quad (3.23)$$

In the convex setting, i.e., if  $G$  is the proximity operator of a proper, closed, convex function, Moreau's identity yields the commonly used form of the primal-dual algorithm as presented in [13]. Note that the Eqs. (3.19)–(3.21) match those of (3.22) and (3.23) up to the extrapolation: While  $2u^{k+1} - u^k$  appears in (3.19)–(3.21), the scheme (3.22)–(3.23) uses  $2p^{k+1} - p^k$ . In this sense, the *primal-dual* schemes of Table 3.1 represent algorithms arising from applying the convex ADMM optimization method to the dual optimization problem

$$\min_p (H_f)^*(p) + R^*(-p),$$

writing the algorithm in a primal-dual form, using Moreau's identity to obtain primal proximity operators only, and finally replacing one of the proximity operators by a neural network. While an algorithmic scheme motivated from a purely dual (and therefore inherently convex) point of view does not seem to have a clear intuition, our numerical experiments indicate that the two variants (3.19)–(3.21) and (3.22)–(3.23) perform quite similarly.

## 3.6 Numerical Experiments: Denoising by Denoising?

### 3.6.1 Different Noise Types and Algorithmic Schemes

So far, the literature on replacing proximal operators by neural networks [15, 45, 54, 75], has focused on the linear inverse problems with a quadratic  $\ell^2$  norm as a data fidelity term, i.e.,

$$H_f(u) = \frac{\alpha}{2} \|Ku - f\|^2,$$

and ADMM or primal-dual type of algorithmic schemes. Interestingly, the behavior of such methods for image denoising with different types of noise, i.e.,  $K$  being the identity and  $H_f$  being a penalty function different from the squared  $\ell^2$  norm, has received little attention despite the fact that adapting the type of penalty is known to be extremely important, particularly in the presence of outliers.

To investigate the behavior of the different algorithmic schemes presented in Table 3.1, we consider images with Gaussian and Salt-and-Pepper noise and use a Huber Loss

$$H_f^v(u) = \sum_{i,j} h^v(u_{ij} - f_{ij}), \quad h^v(x) = \begin{cases} \frac{1}{2v}x^2 & \text{if } |x| \leq v, \\ |x| & \text{otherwise,} \end{cases}$$

as a data fidelity term. The Huber loss has the advantage that it is differentiable with a  $L$ -Lipschitz continuous derivative for  $L = \frac{1}{v}$ , and, at the same time, also allows an efficient computation of its proximal operator, which is given by

$$\text{prox}_{\tau h}(y) = \begin{cases} y/(1 + \frac{\tau}{v}) & \text{if } |y| \leq v + \tau \\ \text{sign}(y)(|y| - \tau) & \text{otherwise.} \end{cases}$$

In our experiments, we evaluate the gradient descent (GD), proximal gradient 1 called forward–backward (FB) here, the half-quadratic splitting (HQ), the ADMM, the primal-dual 1 (PD1), and primal-dual 2 (PD2) (with  $K$  being the identity) schemes from Table 3.1 for denoising grayscale images using MATLAB’s built-in implementation of the DnCNN denoising network [74] as a proximal operator. For the sake of comparability, we also include the plain application of this denoising network (Net), and a total variation-based denoising (TV) in our comparison. To each clean image, we add white Gaussian noise of standard deviation  $\sigma = 0.05$  (for images with values in  $[0, 1]$ ), and additionally destroy 1% of the pixels using Salt-and-Pepper noise. While we are aware of the fact that this does not necessarily reflect a realistic data formation process for camera images, our goal here is to study to what extent each of the algorithmic schemes from Table 3.1 is able to adapt to different settings by changing the data fidelity term.

We fixed the smoothing parameter  $\nu = 0.025$  for the Huber loss and then tuned the hyperparameters of TV and the algorithmic schemes on a validation image, where we found a data fidelity weight of 0.02 to be a good choice for all network-based algorithmic schemes. Note that this means that the factor in front of the Huber loss is smaller than  $2/L$ , where  $L$  is the Lipschitz constant of  $\nabla H_f^\nu$ . Clearly, the latter is important as the schemes that descent on  $H_f^\nu$  in an explicit fashion typically require this condition even in a convex setting. Furthermore, we also met the requirements for “convex convergence” in the primal-dual schemes by choosing  $\beta\tau = 1$  in the “primal-dual 1” scheme, and  $\beta\tau = 0.5$  for “primal-dual 2”.

We keep all parameters fixed over a run on 7 different test images and show the resulting PSNR values for all algorithmic schemes in Table 3.2.

**Table 3.2** PSNR values for denoising images with Gaussian and Salt-and-Pepper noise obtained by applying a neural network trained on Gaussian noise (Net), total variation denoising (TV), and different algorithmic schemes with a neural network replacing the proximal operator of the regularization, and a Huber loss being used as a measure for data fidelity

	Cats	Xmax	Food	Ball	Car	Monkey	Pretzel	Avg.
TV	27.53	24.12	29.46	24.89	27.27	28.00	30.57	27.41
Net	26.76	24.87	27.78	25.25	26.90	26.39	28.41	26.62
HQ	<b>28.97</b>	26.94	<b>29.97</b>	<b>26.97</b>	28.90	<b>29.42</b>	30.32	<b>28.79</b>
FB	28.86	26.73	29.84	26.81	28.97	29.29	<b>30.79</b>	28.76
GD	28.33	<b>26.96</b>	28.89	26.76	28.05	28.72	28.76	28.07
ADMM	28.86	26.73	29.84	26.81	28.97	29.29	<b>30.79</b>	28.76
PD1	28.86	26.73	29.84	26.81	28.97	29.29	<b>30.79</b>	28.76
PD2	28.85	26.76	29.83	26.81	<b>28.99</b>	29.30	<b>30.79</b>	28.76

Bold indicates largest number(s)

As we can see, algorithmic schemes are able to improve the results of plainly applying the network by more than 2db on average. Interestingly, the results among different algorithmic schemes vary very little with the gradient descent-based algorithmic scheme being the only one that yields some deviation in terms of PSNR. While similar behavior of different algorithms is to be expected for convex optimization methods, it is quite remarkable that the algorithmic schemes behave similarly.

To investigate the robustness of the algorithmic schemes, we investigate their sensitivity with respect to the starting point. While we used a constant image whose mean coincides with the mean of the noisy image as a starting point for the results in Table 3.2, Table 3.3 shows the average PSNR values over the same test images when initializing with different images. As we can see, the results remain remarkably stable with respect to different initializations.

In the above test, we ran all algorithmic schemes for a fixed number of 100 iterations. An interesting question is, whether the algorithmic schemes actually converge or if they just behave somewhat nicely for a while, but do not yield any fixed points.

**Table 3.3** Average PSNR values each method achieved on the test set of 7 images used in Table 3.2 when initializing each method with a constant image (constant), with random numbers uniformly sampled in  $[0, 1]$  (random), with the noisy input image (noisy), or with MATLAB’s cameraman image, i.e., a different image (different). The final results of the algorithmic schemes remain remarkably stable and do not vary significantly more than the TV result (whose variations are merely due to different realizations of the noise)

	TV	Net	HQ	FB	GD	ADMM	PD1	PD2
Constant	27.41	26.62	28.79	28.76	28.07	28.76	28.76	28.76
Random	27.40	26.65	28.81	28.77	28.10	28.77	28.77	28.80
Noisy	27.41	26.64	28.79	28.76	28.06	28.76	28.76	28.77
Different	27.40	26.68	28.79	28.75	27.98	28.76	28.76	28.70

### 3.6.1.1 Numerical Convergence of Algorithmic Schemes

Several works in the literature investigate the question whether algorithmic schemes arising from the ADMM algorithm converge:

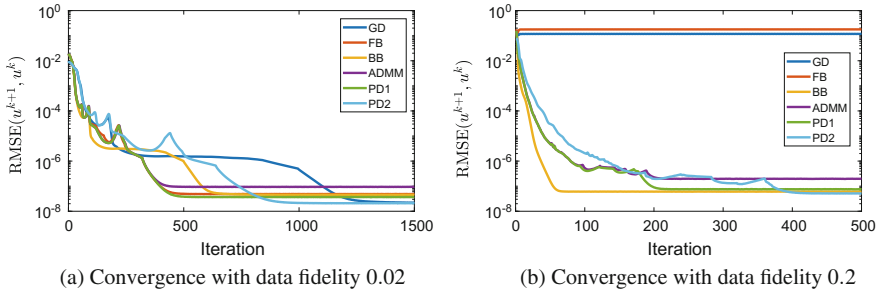
- The work [62] gives sufficient conditions under which a general denoiser, e.g., a neural network  $G$ , represents the proximal operator of some implicitly defined function. As  $G$  is assumed to be continuously differentiable and  $\nabla G(u)$  has to be doubly stochastic for any  $u$ , the assumptions are, however, quite restrictive.
- The authors of [14] state a converge result of an ADMM-based algorithmic scheme with adaptive penalty parameter under the assumption of a *bounded denoiser*. The adaptive scheme, however, possibly allows an exponential growth of the penalty parameter. While the latter safeguards the convergence the point it converges to might not be a fixed-point of the algorithmic scheme anymore.
- The work by Romano et al. in [54] proposes a flexible way to incorporate denoiser  $G$  (such as neural networks) into different algorithmic frameworks by providing quite general conditions under which the function

$$R(u) = \frac{1}{2} \langle u, u - G(u) \rangle$$

has a gradient  $\nabla R(u) = u - G(u)$ , such that it can easily be incorporated into existing optimization algorithms. While the assumption  $G(\alpha u) = \alpha G(u)$  for all  $\alpha \geq 0$  made in their work does hold for several denoisers, neural networks often have a bias in each layer which prevents the above homogeneity. We, therefore, investigate the question if the algorithmic schemes converge numerically for a state-of-the-art denoising network which did not adapt its design to any particular convergence criteria.

Figure 3.5a shows the decay of the root mean square error (RMSE) of successive iterates

$$\text{RMSE}(u^k, u^{k+1}) = \sqrt{\frac{1}{\text{number of pixels}} \sum_{i,j} (u_{ij}^k - u_{ij}^{k+1})^2}$$



**Fig. 3.5** Numerical convergence test of different algorithmic schemes. The schemes seem to behave similar to convex minimization techniques in the sense that they converge numerically if the convex stability criteria are satisfied

for each of the algorithmic schemes from Table 3.2. As we can see, all algorithmic schemes converge to a reasonably small level (considering that all computations are done on a GPU in single precision).

We rerun the same test as above after multiplying the data fidelity term by a factor of 10 and illustrate the results in Fig. 3.5b. As we discussed in Sect. 3.5.4.2, the data fidelity weight is directly connected to a step size of the algorithmic schemes. As expected based on the respective behavior in a convex optimization setting, methods that take explicit steps on the data fidelity term do not exhibit convergence anymore. Interestingly, the methods that evaluate the proximity operator of the data fidelity term still converge and seem to be quite independent of the magnitude of the data fidelity parameter.

While the numerical convergence behavior in our denoising test is closely related to the convergence behavior of the respective methods in the case of convex optimization, an analysis with sufficient conditions on the network to yield a provably convergent algorithm remains an interesting question of future research.

### 3.6.2 Handling Constraints

Besides the lack of versatility of learning-based approaches, a significant drawback is the lacking control over their output: For instance, once a network has been trained there is no parameter that allows to tune the amount of denoising. Moreover, although many types of constraints can be encouraged during training, there is no guarantee for the networks output to meet such constraints during testing. This is of utmost importance in any application, where critical decisions depend on the networks output.

Interestingly, the framework of algorithmic schemes based on optimization algorithms allows to guarantee certain constraints by choosing the data fidelity term  $H_f$  to be the indicator function of the desired (convex) constrained set. As an example, consider the case where want to denoise an image under the constraint that each pixel may at most be altered by  $\delta$ , i.e., we want our reconstruction  $u$  to meet  $\|u - f\|_\infty \leq \delta$

**Table 3.4** Average PSNR values each method achieved on the test set of 7 images with uniform noise and a suitable bound on  $\|u - f\|_\infty$ 

	TV	Net	HQ	ADMM	PD1	PD2
PSNR	32.77	32.66	31.99	32.54	32.67	32.80

for  $f$  being the noisy input image. Note that such constraints can easily be extended to a setting of inverse problem, e.g., requiring  $\|Ku - f\|_2 \leq \delta$ . The fact that indicator functions are not differentiable excludes the gradient descent, as well as the proximal gradient 1 algorithms. Moreover, the primal-dual 2 scheme does not guarantee the output  $u^{k_{\max}}$  to meet the constraint exactly unless it converged. We, therefore, return

$$\text{prox}_{H_f}(z^{k_{\max}} + u^{k_{\max}}),$$

which satisfies the constrain and is supposed to coincide with  $u^{k_{\max}}$  upon convergence.

We simulate images with uniform noise and set our data fidelity term to be the indicator function of  $\|u - f\|_\infty \leq \delta$ , which has an easy-to-evaluate proximity operator. We run the algorithmic schemes HQ splitting, ADMM, primal-dual 1, and primal-dual 2, as well as TV denoising (as a baseline), and compare to the plain application of the denoising network.

The average PSNR values are shown in Table 3.4. Interestingly, the PSNR values do not differ significantly, and the algorithmic schemes may perform worse (HQ), or slightly better (PD2) than the plain application of the network. While these results would not justify the additional computational effort of the algorithmic schemes, note that the **Net** result violated the  $\|u - f\|_\infty \leq \delta$  bound at about 25% of the pixels on average. Although the simple projection of the network’s result would yield satisfactory results in this simple application, the constraint violation illustrates the lacking control of neural networks.

Finally, comparing the results of the network and the algorithmic schemes to plain TV denoising, we can see that TV denoising is (at least) on par with the other methods. This yields the interesting conclusions that the advantages certain methods have as a denoiser do not necessarily carry over to other applications via the algorithmic schemes we presented in Table 3.1. In particular, an important question for future research is how networks (or general denoisers) can be designed in such a way that they work well in various different setting, in particular in such a way that they perform well with additional constraints on the output.

### 3.7 Conclusions

We have summarized some classical denoising methods including self-similarity-based filtering and variational methods, and discussed various learning-based methods that profit from a dataset of natural images. The framework of replacing proxi-

mal operators within optimization algorithms for energy minimization methods with denoising networks holds great promise in tackling various imaging tasks, using different data fidelities, and being able to adjust the amount of regularity without having to retrain the underlying neural network. Interestingly, the particular choice of algorithmic scheme had little influence on the final result in our numerical experiments and the convergence behavior of all algorithms was similar. Changing the algorithmic scheme from a penalty formulation to a constrained formulation changed the results quite significantly in the sense that the advantages of the neural network over TV regularization for image denoising did not transfer to the corresponding algorithmic scheme. Hence, an understanding of desirable properties of denoising algorithms for optimal results in the setting of algorithmic schemes remains an important question for future research.

## References

1. Aharon M, Elad M, Bruckstein A (2006) *rmk*-svd: an algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Trans Signal Process* 54(11):4311–4322
2. Asaki TJ, Le T, Chartrand R (2007) A variational approach to reconstructing images corrupted by Poisson noise. *J Math Imaging Vis (JMIV)* 27:257–263
3. Aström F, Schnörr C (2017) A geometric approach for color image regularization. *Comput Vis Image Underst* 165:43–59
4. Awate SP, Whitaker RT (2005) Higher-order image statistics for unsupervised, information-theoretic, adaptive, image filtering. In: *IEEE conference on computer vision and pattern recognition (CVPR)*, vol 2, pp 44–51
5. Bertalmo M, Levine S (2014) Denoising an image by denoising its curvature image. *SIAM J Imaging Sci* 7(1):187–211
6. Blake A, Zisserman A (1987) *Visual reconstruction*. MIT Press
7. Blomgren P, Chan TF (1998) Color tv: total variation methods for restoration of vector-valued images. *IEEE Trans Image Process* 7(3):304–309
8. Bredies K, Kunisch K, Pock T (2010) Total generalized variation. *SIAM J Imaging Sci* 3(3):492–526
9. Brox T, Cremers D (2007) Iterated nonlocal means for texture restoration. In: Sgallari F, Murli A, Paragios N (eds) *International conference on scale space and variational methods in computer vision (SSVM)*. LNCS, vol 4485. Springer, pp 13–24
10. Buades A, Coll B, Morel JM (2005) A non-local algorithm for image denoising. In: *IEEE conference on computer vision and pattern recognition (CVPR)*, vol 2, pp 60–65
11. Burger HC, Schuler CJ, Harmeling S (2012) Image denoising: can plain neural networks compete with BM3D? In: *2012 IEEE conference on computer vision and pattern recognition*, pp 2392–2399
12. Chambolle A, Lions P-L (1997) Image recovery via total variation minimization and related problems. *Numerische Mathematik* 76(2):167–188
13. Chambolle A, Pock T (2011) A first-order primal-dual algorithm for convex problems with applications to imaging. *J Math Imaging Vis (JMIV)*
14. Chan SH, Wang X, Elgendy OA (2017) Plug-and-play admm for image restoration: fixed-point convergence and applications. *IEEE Trans Comput Imaging* 3(1):84–98
15. Chang J-H, Li C-L, Poczós B, Vijaya Kumar BVK, Sankaranarayanan AC (2017) One network to solve them all—solving linear inverse problems using deep projection models. In: *IEEE international conference on computer vision (ICCV)*



16. Chartrand R (2012) Nonconvex splitting for regularized low-rank + sparse decomposition. *IEEE Trans Signal Process* 60(11):5810–5819
17. Chartrand R, Wohlberg B (2013) A nonconvex ADMM algorithm for group sparsity with sparse groups. In: 2013 IEEE international conference on acoustics, speech and signal processing, pp 6009–6013
18. Chen Y, Ranftl R, Pock T (2014) Insights into analysis operator learning: from patch-based sparse models to higher order MRFs. *IEEE Trans Image Process* 23(3):1060–1072
19. Chen Y, Yu W, Pock T (2015) On learning optimized reaction diffusion processes for effective image restoration. In: IEEE conference on computer vision and pattern recognition (CVPR), June 2015
20. Cho NI, Ahn B (2017) Block-matching convolutional neural network for image denoising. <https://arxiv.org/abs/1704.00524>
21. Condat L, Mosaddegh S (2012) Joint demosaicking and denoising by total variation minimization. In: IEEE international conference on image processing (ICIP), pp 2781–2784
22. Cremers D, Rousson M, Deriche R (2007) A review of statistical approaches to level set segmentation: integrating color, texture, motion and shape. *Int J Comput Vis (IJCV)* 72(2):195–215
23. Dabov K, Foi A, Egiazarian K (2007) Video denoising by sparse 3d transform-domain collaborative filtering. In: European signal processing conference (EUSIPCO)
24. Danielyan A, Katkovnik V, Egiazarian K (2010) Image deblurring by augmented lagrangian with BM3D frame prior
25. Danielyan A, Katkovnik V, Egiazarian K (2012) BM3D frames and variational image deblurring. *IEEE Trans Image Process* 21(4):1715–1728
26. Dave A, Vadathya AK, Mitra K (2017) From learning models of natural image patches to whole image restoration. In: IEEE international conference on image processing (ICIP)
27. Duran J, Moeller M, Sbert C, Cremers D (2016) Collaborative total variation: a general framework for vectorial tv models. *SIAM J Imaging Sci* 9(1):116–151
28. Geman S, Geman D (1984) Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans Pattern Anal Mach Intell (PAMI)* 6(6):721–741
29. Gilboa G, Osher S (2009) Nonlocal operators with applications to image processing. *Multiscale Model Simul* 7(3):1005–1028
30. Giusti E (1984) Minimal surfaces and functions of bounded variation. Birkhäuser
31. Gu S, Zhang L, Zuo W, Feng X (2014) Weighted nuclear norm minimization with application to image denoising. In: IEEE conference on computer vision and pattern recognition (CVPR), pp 2862–2869
32. He K, Zhang X, Ren S, Sun J (2015) Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: IEEE international conference on computer vision (ICCV), pp 1026–1034
33. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: IEEE conference on computer vision and pattern recognition (CVPR), pp 770–778
34. Heide F, Steinberger M, Tsai YT, Rouf M, Pajk D, Reddy D, Gallo O, Liu J, Heidrich W, Egiazarian K, Kautz J, Pulli K (2014) Flexisp: a flexible camera image processing framework. In: ACM special interest group on computer graphics (SIGGRAPH)
35. Huang J, Mumford D (1999) Statistics of natural images and models. In: IEEE computer society conference on computer vision and pattern recognition, 1999, vol 1. IEEE, pp 541–547
36. Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning (ICML)
37. Jain V, Seung S (2009) Natural image denoising with convolutional networks. In: Koller D, Schuurmans D, Bengio Y, Bottou L (eds) *Advances in neural information processing systems*, vol 21. Curran Associates, Inc., pp 769–776
38. Johnson J, Alahi A, Li F-F (2016) Perceptual losses for real-time style transfer and super-resolution. In: European conference on computer vision (ECCV)
39. Kheradmand A, Milanfar P (2014) A general framework for regularized, similarity-based image restoration. *IEEE Trans Image Process* 23(12):5136–5151

40. Ledig C, Theis L, Huszar F, Caballero J, Cunningham A, Acosta A, Aitken A, Tejani A, Totz J, Wang Z, Shi W (2016) Photo-realistic single image super-resolution using a generative adversarial network. <https://arxiv.org/abs/1609.04802>
41. Lefkimmiatis S (2017) Non-local color image denoising with convolutional neural networks. <https://arxiv.org/abs/1611.06757>
42. Liu P, Fang R (2017) Learning pixel-distribution prior with wider convolution for image denoising. <https://arxiv.org/abs/1707.09135>
43. Mairal J, Bach F, Ponce J, Sapiro G, Zisserman A (2009) Non-local sparse models for image restoration. In: IEEE international conference on computer vision (ICCV), pp 2272–2279
44. Mairal J, Elad M, Sapiro G (2008) Sparse representation for color image restoration. *IEEE Trans Image Process* 17(1):53–69
45. Meinhardt T, Moeller M, Hazirbas C, Cremers D (2017) Learning proximal operators: Using denoising networks for regularizing inverse imaging problems. In: IEEE international conference on computer vision (ICCV)
46. Möllenhoff T, Strelakovsky E, Moeller M, Cremers D (2014) The primal-dual hybrid gradient method for semiconvex splittings. 8:07
47. Möllenhoff T, Strelakovsky E, Moeller M, Cremers D (2015) Low rank priors for color image regularization. In: Energy minimization methods in computer vision and pattern recognition (EMMCVPR)
48. Mumford D, Shah J (1989) Optimal approximations by piecewise smooth functions and associated variational problems. *Commun Pure Appl Math* 42:577–685
49. Nguyen A, Clune J, Bengio Y, Dosovitskiy A, Yosinski J (2017) Plug & play generative networks: conditional iterative generation of images in latent space. In: IEEE conference on computer vision and pattern recognition (CVPR)
50. Osher S, Burger M, Goldfarb D, Xu J, Yin W (2005) An iterative regularization method for total variation-based image restoration. *Multiscale Model Simul* 4(2):460–489
51. Park SH, Kim HS, Langel S, Parmar M, Wandell BA (2009) A case for denoising before demosaicking color filter array data. In: Asilomar conference on signals, systems and computers, pp 860–864
52. Plötz T, Roth S (2017) Benchmarking denoising algorithms with real photographs. In: IEEE conference on computer vision and pattern recognition (CVPR)
53. Remez T, Litany O, Giryas R, Bronstein AM (2017) Deep class-aware image denoising. In: 2017 international conference on sampling theory and applications (SampTA), pp 138–142
54. Romano Y, Elad M, Milanfar P (2017) The little engine that could: regularization by denoising (red). *SIAM J Imaging Sci* 10(4):1804–1844
55. Roth S, Black MJ (2005) Fields of experts: a framework for learning image priors. In: IEEE conference on computer vision and pattern recognition (CVPR), vol 2, pp 860–867
56. Rudin LI, Osher S, Fatemi E (1992) Nonlinear total variation based noise removal algorithms. *Phys D*
57. Saint-Marc P, Chen JS, Medioni G (1989) Adaptive smoothing: a general tool for early vision. In: IEEE conference on computer vision and pattern recognition (CVPR), pp 618–624
58. Sapiro G, Ringach DL (1996) Anisotropic diffusion of multivalued images. In: Images, wavelets and PDEs, pp 134–140
59. Schmidt U, Roth S (2014) Shrinkage fields for effective image restoration. In: IEEE conference on computer vision and pattern recognition (CVPR), pp 2774–2781
60. Seybold T, Keimel C, Knopp M, Stechele W (2013) Towards an evaluation of denoising algorithms with respect to realistic camera noise. In: IEEE international symposium on multimedia, pp 203–210
61. Shulman D, Herve J-Y (1989) Regularization of discontinuous flow fields. In: Workshop on visual motion, 1989, proceedings. IEEE, pp 81–86
62. Sreehari S, Venkatakrishnan SV, Wohlberg B, Drummy LF, Simmons JP, Bouman CA (2016) Plug-and-play priors for bright field electron tomography and sparse interpolation. *IEEE Trans Comput Imaging* 2:408–423

63. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. *J Mach Learn Res* 15:1929–1958
64. Strekalovskiy E, Chambolle A, Cremers D (2012) A convex representation for the vectorial Mumford-Shah functional. In: *IEEE conference on computer vision and pattern recognition (CVPR)*, Providence, Rhode Island, June 2012
65. Strekalovskiy E, Cremers D (2014) Real-time minimization of the piecewise smooth Mumford-Shah functional. In: *European conference on computer vision (ECCV)*
66. Valkonen T (2014) A primal dual hybrid gradient method for nonlinear operators with applications to MRI. *Inverse Probl* 30(5)
67. Venkatakrishnan S, Bouman CA, Wohlberg B (2013) Plug-and-play priors for model based reconstruction. In: *Global conference on signal and information processing (GlobalSIP)*
68. Wan L, Zeiler M, Zhang S, Le Cun Y, Fergus R (2013) Regularization of neural networks using dropconnect. In: Sanjoy D, David M (eds) *International conference on machine learning (ICML)*, Proceedings of machine learning research, PMLR, Atlanta, Georgia, USA, vol 28, pp 1058–1066
69. Wang R, Tao D (2016) Non-local auto-encoder with collaborative stabilization for image restoration. *IEEE Trans Image Process* 25(5):2117–2129
70. Wang Y, Yin W, Zeng J (2017) Global convergence of ADMM in nonconvex nonsmooth optimization. <https://arxiv.org/abs/1511.06324>
71. Wang YQ, Morel JM (2014) Can a single image denoising neural network handle all levels of Gaussian noise? *IEEE Signal Process Lett* 21(9):1150–1153
72. Xie J, Xu L, Chen E (2012) Image denoising and inpainting with deep neural networks. In: Pereira F, Burges CJC, Bottou L, Weinberger KQ (eds) *Advances in neural information processing systems*, vol 25. Curran Associates, Inc., pp 341–349
73. Zhang X, Esser E (2014) Nonlocal path-based image inpainting through minimization of a sparsity promoting nonconvex functional. <https://www.eoas.ubc.ca/~eesser/papers/nliPDHGM.pdf>
74. Zhang K, Zuo W, Chen Y, Meng D, Zhang L (2017) Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Trans Image Process* 26(7):3142–3155
75. Zhang K, Zuo W, Gu S, Zhang L (2017) Learning deep CNN denoiser prior for image restoration. In: *IEEE conference on computer vision and pattern recognition (CVPR)*, pp 2808–2817
76. Zoran D, Weiss Y (2011) From learning models of natural image patches to whole image restoration. In: *IEEE international conference on computer vision (ICCV)*, pp 479–486

# Chapter 4

## Convolutional Neural Networks for Image Denoising and Restoration



Wangmeng Zuo, Kai Zhang and Lei Zhang

**Abstract** With the tremendous progress of convolutional neural networks (CNNs), recent years have witnessed a dramatic upsurge of exploiting CNN toward solving image denoising. Compared to traditional model-based methods, CNN enjoys the principal merits of fast inference and good performance. In this chapter, brief survey and discussions are also given to CNN-based denoising methods from the aspects of effectiveness, interpretability, modeling ability, efficiency, flexibility, and applicability. Then, we provide a gentle introduction of CNN-based denoising methods by presenting and answering the following three questions: (i) can we learn a deep CNN for effective image denoising, (ii) can we learn a single CNN for fast and flexible non-blind image denoising, and (iii) can we leverage CNN denoiser prior to versatile image restoration tasks. Finally, we point out that image denoising remains far from solved. The real image noise is much more sophisticated than additive white Gaussian noise, making the existing CNN denoisers generally perform poorly on real noisy images. As a result, it is still very challenging and valuable to study the issues such as noise modeling, acquisition of noisy-clean image pairs and unsupervised CNN learning for real image denoising.

---

W. Zuo (✉) · K. Zhang  
School of Computer Science and Technology, Harbin Institute of Technology,  
Harbin, China  
e-mail: wmzuo@hit.edu.cn

K. Zhang  
e-mail: cskazhang@gmail.com

L. Zhang  
Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China  
e-mail: cslzhang@comp.polyu.edu.hk

## 4.1 Introduction

During past decade, driven by the easy access to large-scale dataset, efficient training implementation on modern powerful GPUs and the advances in deep learning methods such as Rectified Linear Unit (ReLU) [22], network initialization [14], stochastic gradient-based optimization [20], batch normalization [17] and residual learning [15], the convolutional neural networks have shown great success in handling various low-level vision tasks. Concurrently, several attempts have been made to exploit CNN for image denoising due to the following reasons. First, the inference can be very efficient due to the parallel computation ability of GPU. Second, deep CNN exhibits powerful modeling capacity and can be trained without knowing the explicit degradation model when abundant noisy/clean image pairs are provided. Thus, the denoising performance can be easily boosted. Third, CNN exploits the external prior which is complementary to the internal prior of many existing denoisers such as nonlocal similarity (NSS). In other words, its combination with NSS is expected to further improve the performance. Fourth, great progress in designing CNN can facilitate the flexibility and practicability in real applications.

The goal of image denoising is to recover a latent clean image  $\mathbf{x}$  from its noisy observation  $\mathbf{y}$  which follows the image degradation model  $\mathbf{y} = \mathbf{x} + \mathbf{v}$ . One common assumption is that  $\mathbf{v}$  is the additive white Gaussian noise (AWGN) with known noise level  $\sigma$ . Due to the ill-posed nature of denoising, regularization needs to be imposed to constrain the solution. From a Bayesian perspective, the solution  $\hat{\mathbf{x}}$  can be obtained by solving the Maximum A Posteriori (MAP) model,

$$\hat{\mathbf{x}} = \arg \max_{\mathbf{x}} \log p(\mathbf{y}|\mathbf{x}) + \log p(\mathbf{x}) \quad (4.1)$$

where  $\log p(\mathbf{y}|\mathbf{x})$  represents the log-likelihood term, and  $\log p(\mathbf{x})$  models the prior of  $\mathbf{x}$  which is independent of  $\mathbf{y}$ . More formally, Eq. (4.1) can be reformulated as

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x}} \frac{1}{2\sigma^2} \|\mathbf{x} - \mathbf{y}\|^2 + \lambda \Phi(\mathbf{x}) \quad (4.2)$$

where  $\frac{1}{2\sigma^2} \|\mathbf{x} - \mathbf{y}\|^2$  is the data fidelity term,  $\Phi(\mathbf{x})$  is the regularization term (or prior term) and  $\lambda$  is the trade-off parameter. It is worth noting that in practice  $\lambda$  governs the compromise between noise reduction and detail preservation. When it is too small, some noise will be retained in the denoising result; in contrast, when  $\lambda$  is too large, small-scale details will also be smoothed out along with the suppression of noise.

Generally, the methods to solve Eq. (4.2) can be divided into two main categories, model-based optimization methods and discriminative learning methods. Model-based optimization directly solves Eq. (4.2) with some optimization algorithms which usually involve a time-consuming iterative inference. On the contrary, discriminative learning methods try to learn a compact inference through an optimization of a loss function on a training set containing degraded-clean image pairs [4, 8, 38, 43]. The CNN-based denoising methods belong to this category. Specifically, CNN-based

denoising methods can be further divided into MAP-CNN based and generic CNN-based denoising methods.

MAP-CNN based denoising methods refer in particular to the MAP inference based approaches which involve a series of convolution operations. Instead of first learning the prior (e.g., high-order MRF priors) and then performing the inference, this category of methods aims to learn the prior parameters along with a compact unrolled inference through solving a bi-level optimization problem [4]. With a slight abuse of notation, the objective can be given by

$$\min_{\Theta} \ell(\hat{\mathbf{x}}(\Theta), \mathbf{x}) \quad s.t. \quad \hat{\mathbf{x}}(\Theta) = \arg \min_{\mathbf{x}} \frac{1}{2\sigma^2} \|\mathbf{y} - \mathbf{x}\|^2 + \lambda \Phi(\mathbf{x}) \quad (4.3)$$

where  $\Theta$  denotes the trainable parameters in the inference,  $\ell(\hat{\mathbf{x}}, \mathbf{x})$  measures the similarity between estimated clean image  $\hat{\mathbf{x}}$  and ground-truth clean image  $\mathbf{x}$ . While such kind of approaches are not directly connected with CNN, their unrolled inferences actually can be viewed as CNN variants with stagewise architecture. Following the pioneer work of fields of experts [34], Barbu [4] trained a discriminative Markov random field (MRF) model together with a gradient descent inference for image denoising. Samuel and Tappen [36] independently proposed a compact gradient descent inference learning framework, and discussed the advantages of discriminative learning over model-based optimization. Sun and Tappen [42] proposed a novel nonlocal range MRF (NLR-MRF) framework, and employed the gradient-based discriminative learning method to train the model. Chen and Pock [8] further proposed a trainable nonlinear reaction–diffusion (TNRD) model through discriminative learning of a compact gradient descent inference. Lefkimmiatis [25] adopted a proximal gradient-based denoising inference from a variational model to incorporate the NSS prior.

Discriminative inference learning methods enjoy some merits. First, they are efficient due to much fewer inference steps. Second, they have better interpretability because the discriminative architecture is derived from optimization algorithms such as half quadratic splitting and gradient descent [4, 8, 36, 38, 42]. However, the interpretability may come at the expense of performance since the learned priors and inference procedure are limited by the form of the MAP model [50]. In addition, the unrolled inference actually can be viewed as a network with stagewise architecture, which restricts the dataflow in each immediate output layer [51].

Instead of modeling image priors explicitly, the generic CNN-based denoising methods learn a predefined nonlinear function consisting of various CNN building blocks to model image prior implicitly and can be modeled by

$$\min_{\Theta} \ell(\hat{\mathbf{x}}, \mathbf{x}) \quad s.t. \quad \hat{\mathbf{x}} = \mathcal{F}(\mathbf{y}, \sigma; \Theta). \quad (4.4)$$

The use of CNN for image denoising can be traced back to [18], where a five-layer network with sigmoid nonlinearity was developed. During the past few years, various generic CNN-based denoising methods have been proposed and the denoising

performance has been greatly boosted in comparison to [18]. In this chapter, we focus on this type of denoising methods, and assume that the noise type is AWGN with the known noise level  $\sigma$ .

### 4.1.1 Recent Advances

In addition to DnCNN and FFDNet described in this chapter, there have been several other attempts to design CNN for image denoising. Some works propose to improve convolutional filters or nonlinearities for better trade-off between effectiveness and efficiency. Wang et al. [47] developed a dilated residual CNN for fast Gaussian denoising. The main idea is to enlarge receptive field by dilated filter with different dilation factors. They showed that the expansion of receptive field can boost the denoising performance. In another denoising work by Wang et al. [46], they proposed a denoising network which uses exponential linear unit (ELU) as the nonlinearity. To better accommodate ELU and batch normalization layer, they further designed a novel structure by incorporating  $1 \times 1$  convolutional layer. Kligvasser et al. [21] proposed a learnable nonlinear function with spatial connections as activation unit to replace the widespread per-pixel activation units such as ReLUs and sigmoids. They showed that the activation unit can enable CNN to capture much more complex features, thus leading to better denoising results.

While CNN-based denoising methods are effective, they lack the explicit ability to handle images with regular and repetitive patterns. Some researchers resort to incorporate the nonlocal self-similarity prior into CNN. Ahn and Cho [1] proposed a block matching convolutional neural network which combines nonlocal self-similarity prior and CNN for image denoising. The main idea is to group similar local patches into a tensor and then feed it to CNN for denoising. Bae et al. [2] proposed a new denoising network motivated from a novel persistent homology analysis on residual learning for image processing tasks. Specifically, they showed that the residual manifold is topologically simpler than the original image manifold and the wavelet transform can provide topologically simpler manifold structures. Yang and Sun [48] proposed a BM3D-inspired convolutional neural network (BM3D-Net) for image denoising. BM3D-Net directly builds the convolutional neural network by faithfully implementing the transform domain collaborative filtering in the BM3D framework. Lefkimmiatis [26] proposed a novel network architecture which involves convolutional layers as a core component and nonlocal filtering layers to exploit the inherent nonlocal self-similarity property of natural images. By training the denoiser with a wide range of different noise levels, the networks do not need to know the noise level of the noisy image and are very robust when the noise model does not match the statistics of the one used during training.

To design a principled network architecture, an interesting line of denoising methods has focused on incorporating CNN building blocks into MAP-based unrolled inference. Kim et al. [19] proposed a deeply aggregated alternating minimization (DeepAM) based on two of the steps in the conventional AM algorithm: proximal

mapping and  $\beta$ -continuation. The DeepAM framework enables the convolutional neural networks to operate as a regularizer in the AM algorithm for image denoising. Vogel and Pock [45] proposed a primal-dual network for image denoising that leverages the algorithmic structure provided by energy optimization techniques into learning a generalized optimization algorithm.

There also exist some works that focus on class-aware image denoising, generic image denoising and boosting-based image denoising. Remez et al. [31] pointed out a denoiser is aware of the type of image content and proposed a new fully convolutional deep neural network architecture for image denoising. They further showed that the performance can be significantly improved by fine-tuning the denoiser for images belonging to a specific semantic class. Santhanam et al. [37] developed a recursively branched deconvolutional network (RBDN) for image denoising as well as generic image-to-image regression. To integrate multiple weak denoisers with different capabilities to denoise complex scenes, Choi et al. [9] proposed a consensus neural network (ConsensusNet) which comprises a weighting stage to weigh the relevance of the individual denoisers and a boosting neural network to recover the lost features as well as improve contrast. They studied ConsensusNet on various scenarios, including the integration of denoisers with different noise levels, different image classes, and different denoiser types. Experimental results show that ConsensusNet can consistently improve denoising performance for both deterministic denoisers and neural network denoisers.

Apart from single image denoising, multi-image denoising has also attracted considerable interest. Godard et al. [12] proposed a recurrent fully CNN to handle an arbitrary number of noisy input frames for burst denoising. Instead of directly predicting the final denoised pixel values [3], Mildenhall et al. [29] proposed a CNN architecture to predict spatially varying weighting kernels of different frames. They further proposed a synthetic data generation approach based on signal-dependent Gaussian noise model, and an optimization guided by an annealed loss function to avoid undesirable local minima.

The rest of this chapter is organized as follows. We first introduce a simple denoising CNN which embraces the progress in learning and designing CNN in Sect. 4.2. We show that residual learning and batch normalization are particularly beneficial to Gaussian denoising. We also analyze the rationale of residual learning and the modeling capacity of CNN. In Sect. 4.3, we provide a fast and flexible denoising CNN with a tunable noise level map as input and thus can handle a wide range of noise levels as well as spatially variant AWGN via a single model. To demonstrate the wide applications of CNN denoisers, in Sect. 4.4, we show that the CNN denoisers can be plugged into model-based optimization methods as a modular part to solve other image restoration tasks such as image deblurring, single image super-resolution (SISR), and image inpainting. We provide a short review of recent advances, and discuss the challenges and some possible solutions in Sect. 4.5.



## 4.2 Learning Deep CNN for Image Denoising

Discriminative model learning for image denoising has been recently attracting considerable attention due to its favorable denoising performance. In this section, we take one step forward by investigating the construction of feed-forward denoising convolutional neural networks (DnCNN) to embrace the progress in deep architecture, learning algorithm, and regularization method into image denoising.

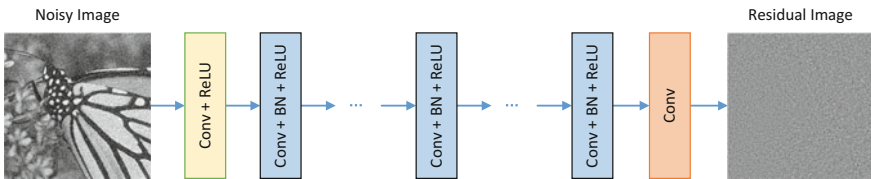
### 4.2.1 Architecture Design: DnCNN

The architecture of DnCNN is shown in Fig. 4.1. The input of DnCNN is a noisy observation  $\mathbf{y} = \mathbf{x} + \mathbf{v}$ . Discriminative denoising models such as MLP [5] and CSF [38] aim to directly learn the original mapping function  $\mathcal{F}(\mathbf{y}) = \mathbf{x}$  to predict the latent clean image. For DnCNN, the residual learning formulation is instead to train a residual mapping  $\mathcal{R}(\mathbf{y}) = \mathbf{v}$ , and then  $\mathbf{x}$  can be obtained by  $\mathbf{x} = \mathbf{y} - \mathcal{R}(\mathbf{y})$ .

Following the principle in [40], the size of convolutional filters is set to  $3 \times 3$ . Therefore, the receptive field of DnCNN with depth of  $d$  should be  $(2d + 1) \times (2d + 1)$ . Increasing the network depth can make use of the context information in larger image region at the cost of computational burden. For better trade-off between performance and efficiency, one important issue in architecture design is to set a proper depth for DnCNN. The receptive field size of DnCNN for Gaussian denoising with a certain noise level is set to  $35 \times 35$  with the corresponding depth of 17.

Given the DnCNN with depth  $D$ , there are three types of layers (shown in Fig. 4.1) with three different colors. (i) Conv + ReLU: for the first layer, 64 filters of size  $3 \times 3 \times c$  are used to generate 64 feature maps, and rectified linear units (ReLU,  $\max(0, \cdot)$ ) are then utilized for nonlinearity. Here,  $c$  represents the number of image channels, i.e.,  $c = 1$  for grayscale image and  $c = 3$  for color image. (ii) Conv + BN + ReLU: for layers  $2 \sim (D - 1)$ , 64 filters of size  $3 \times 3 \times 64$  are used, and batch normalization [17] is added between convolution and ReLU. (iii) Conv: for the last layer,  $c$  filters of size  $3 \times 3 \times 64$  are used to reconstruct the denoising result.

To sum up, DnCNN has two main features: the residual learning formulation is adopted to learn  $\mathcal{R}(\mathbf{y})$ , and batch normalization is incorporated to speed up training as well as boost the denoising performance. In particular, it turns out that residual



**Fig. 4.1** The architecture of the DnCNN network

learning and batch normalization can benefit from each other, and their integration is effective in speeding up the training and boosting the denoising performance.

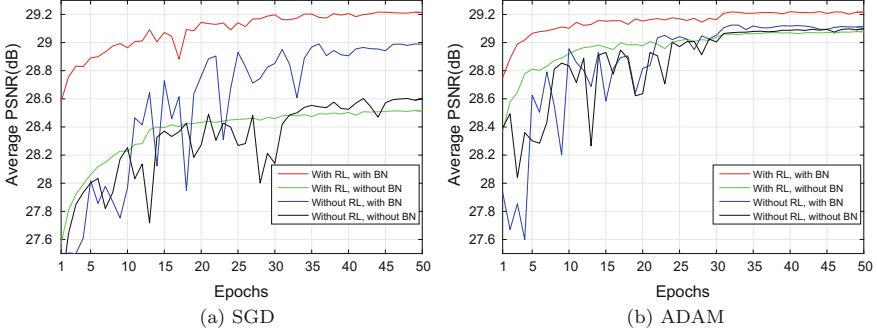
### 4.2.2 Residual Learning and Batch Normalization

To start with, it is useful to briefly review residual learning and batch normalization. The main idea of residual learning [15] is that the residual mapping is much easier to be learned than the original mapping. Typically, residual network stacks a number of residual units to alleviate the degradation of training accuracy. Benefited from residual network, deep CNN can be easily trained and improved accuracy has been achieved for image classification and object detection [15]. Different from the residual network [15] that uses many residual units (i.e., identity shortcuts), DnCNN employs a single residual unit to predict the residual image. It should be noted that, prior to the residual network [15], the strategy of predicting the residual image has already been adopted in some low-level vision problems such as SISR and image demosaicking. As for batch normalization [17], it was originally proposed to alleviate the internal covariate shift by incorporating a Gaussian normalization step and a scale and shift step. It enjoys several merits, such as fast training, better performance, and low sensitivity to initialization. For further details on batch normalization, please refer to [17].

The DnCNN network can be used to train either the original mapping  $\mathbf{y}$  to predict  $\mathbf{x}$  or the residual mapping  $\mathcal{R}(\mathbf{y})$  to predict  $\mathbf{v}$ . According to [15], when the original mapping is more like an identity mapping, the residual mapping will be much easier to optimize. Since the noisy observation  $\mathbf{y}$  is much more like the latent clean image  $\mathbf{x}$  than the residual image  $\mathbf{v}$  (especially when the noise level is low),  $\mathcal{F}(\mathbf{y})$  would be more close to an identity mapping than  $\mathcal{R}(\mathbf{y})$ . Thus, the residual learning formulation is more suitable for image denoising.

For Gaussian noise removal, residual learning is also helpful to stabilize the training with batch normalization. Under the residual learning setting, the noise output of a specific noise level should be an ideal Gaussian distribution. Moreover, DnCNN with residual learning implicitly removes the latent clean image with the operations in the hidden layers. This makes that the inputs of each layer are Gaussian-like distributed, less correlated, and less related with image content. As a result, residual learning and batch normalization are particularly beneficial to each other for Gaussian denoising.

Figure 4.2 shows the average PSNR values obtained using these two learning formulations with/without batch normalization under the same setting on gradient-based optimization algorithms and network architecture. Two gradient-based optimization algorithms are adopted: one is the stochastic gradient descent algorithm with momentum (i.e., SGD) and the other one is the ADAM algorithm [20]. One can observe that both the SGD and ADAM optimization algorithms can enable the network with residual learning and batch normalization to have the best results. In other words,



**Fig. 4.2** The quantitative Gaussian denoising results of four specific models under two gradient-based optimization algorithms, i.e., **a** SGD, **b** ADAM and respect to epochs. The four specific models are in different combinations of residual learning (RL) and batch normalization (BN) and are trained with noise level 25. The results are evaluated on 68 natural images from Berkeley segmentation dataset

it is the integration of residual learning formulation and batch normalization rather than the optimization algorithms (SGD or ADAM) that lead to the best denoising performance.

### 4.2.3 Connection with TNRD

To have a further understanding of residual learning for denoising, we analyze its connection with TNRD [8] which is a MAP-CNN based denoising method. The main idea of TNRD is to train a discriminative solution for the following problem:

$$\min_{\mathbf{x}} \Psi(\mathbf{y} - \mathbf{x}) + \lambda \sum_{k=1}^K \sum_{p=1}^N \rho_k((\mathbf{f}_k * \mathbf{x})_p) \quad (4.5)$$

from an abundant set of degraded-clean training image pairs. Here,  $N$  denotes the image size,  $\lambda$  is the regularization parameter,  $\mathbf{f}_k * \mathbf{x}$  stands for the convolution of the image  $\mathbf{x}$  with the  $k$ -th filter kernel  $\mathbf{f}_k$ , and  $\rho_k(\cdot)$  represents the  $k$ -th penalty function which is adjustable in the TNRD model. For Gaussian denoising, we set  $\Psi(\mathbf{z}) = \frac{1}{2} \|\mathbf{z}\|^2$ .

The diffusion iteration of the first stage can be interpreted as performing one gradient descent inference step at starting point  $\mathbf{y}$ , which is given by

$$\mathbf{x}_1 = \mathbf{y} - \alpha \lambda \sum_{k=1}^K (\bar{\mathbf{f}}_k * \phi_k(\mathbf{f}_k * \mathbf{y})) - \alpha \left. \frac{\partial \Psi(\mathbf{z})}{\partial \mathbf{z}} \right|_{\mathbf{z}=\mathbf{0}} \quad (4.6)$$

where  $\bar{\mathbf{f}}_k$  is the adjoint filter of  $\mathbf{f}_k$  (i.e.,  $\bar{\mathbf{f}}_k$  is obtained by rotating  $180^\circ$  the filter  $\mathbf{f}_k$ ),  $\alpha$  corresponds to the stepsize and  $\rho'_k(\cdot) = \phi_k(\cdot)$ . For Gaussian denoising, we have  $\frac{\partial \Psi(\mathbf{z})}{\partial \mathbf{z}} \Big|_{\mathbf{z}=\mathbf{0}} = \mathbf{0}$ , and Eq. (4.6) is equivalent to the following expression,

$$\mathbf{v}_1 = \mathbf{y} - \mathbf{x}_1 = \alpha \lambda \sum_{k=1}^K (\bar{\mathbf{f}}_k * \phi_k(\mathbf{f}_k * \mathbf{y})) \quad (4.7)$$

where  $\mathbf{v}_1$  is the estimated residual of  $\mathbf{x}$  with respect to  $\mathbf{y}$ .

Since the influence function  $\phi_k(\cdot)$  can be regarded as pointwise nonlinearity applied to convolution feature maps, Eq. (4.7) actually is a two-layer feed-forward CNN. DnCNN further generalizes one-stage TNRD from three aspects: (i) replacing the influence function with ReLU to ease CNN training; (ii) increasing the CNN depth to improve the capacity in modeling image characteristics; (iii) incorporating with batch normalization to boost the performance. The connection with one-stage TNRD provides insights in explaining the use of residual learning for CNN-based image restoration.

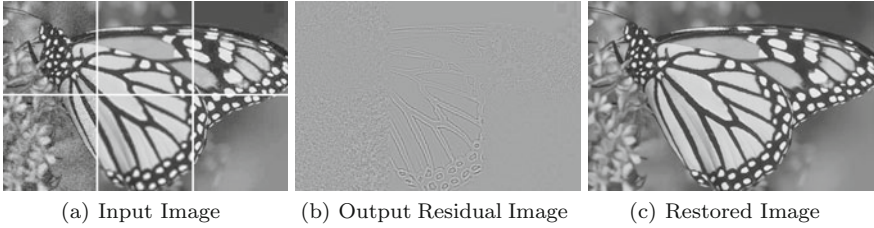
#### 4.2.4 Understanding the CNN Modeling Capacity

The existing Gaussian denoising methods, such as MLP [5] and TNRD [8], all train a specific model for a fixed noise level. It is interesting to investigate the modeling capacity of CNN for different noise levels via a single model. According to Eq. (4.7), one can see that most of the parameters are derived from the analysis prior term of Eq. (4.5). In this sense, the parameters are mainly representing the image priors which are task-independent. Therefore, CNN has the modeling capacity to deal with multiple degradations via a single model. For example, even the noise is not Gaussian distributed, one still can utilize Eq. (4.6) to obtain  $\mathbf{v}_1$  if

$$\frac{\partial \Psi(\mathbf{z})}{\partial \mathbf{z}} \Big|_{\mathbf{z}=\mathbf{0}} = \mathbf{0}. \quad (4.8)$$

Note that Eq. (4.8) holds for many types of noise distributions, e.g., generalized Gaussian distribution. It is natural to assume that it also holds for the noise caused by image downsampling and JPEG compression. Thus, it is possible to train a single CNN model for several general image denoising tasks, e.g., SISR and JPEG deblocking.

To demonstrate the potential of DnCNN in general image denoising, DnCNN is extended for learning a single model for three specific tasks, i.e., blind Gaussian denoising for noise level range of  $[0, 55]$ , SISR, and JPEG deblocking are considered. In the training stage, the images with AWGN from a wide range of noise levels, downsampled images with multiple upscaling factors, and JPEG images with



**Fig. 4.3** An example to show the capacity of DnCNN for three different tasks. The input image is composed by noisy images with noise level 15 (upper left) and 25 (lower left), bicubically interpolated low-resolution images with upscaling factor 2 (upper middle) and 3 (lower middle), JPEG images with quality factor 10 (upper right) and 30 (lower right). Note that the white lines in the input image are just used for distinguishing the six regions, and the residual image is normalized into the range of  $[0, 1]$  for visualization

different quality factors are utilized. Experimental results show that the learned single DnCNN model is able to yield excellent results for any of the three general image denoising tasks. Figure 4.3 shows an example of DnCNN for these tasks. As one can see, even the input image is corrupted with different distortions in different regions, the restored image looks natural and does not have obvious artifacts.

#### 4.2.5 Implementation and Experiments

Following [8], 400 images of size  $180 \times 180$  are used for training the Gaussian denoiser with known noise level. It has been found that using a larger training dataset can only bring negligible improvements, especially on BSD68 test set. Three noise levels, i.e.,  $\sigma = 15, 25$  and  $50$ , are considered and thus three models are trained. The patch size is set as  $40 \times 40$ .  $128 \times 1,600$  patches are used to train the model and the mini-batch size is set to 128. The mean squared error between the desired residual images and estimated ones from noisy input

$$\mathcal{L}(\Theta) = \frac{1}{2N} \sum_{i=1}^N \|\mathcal{R}(\mathbf{y}_i; \Theta) - (\mathbf{y}_i - \mathbf{x}_i)\|^2 \quad (4.9)$$

is adopted as the loss function to learn the trainable parameters  $\Theta$ . Here  $\{(\mathbf{y}_i, \mathbf{x}_i)\}_{i=1}^N$  represents  $N$  noisy-clean training image patch pairs. The learning rate was decayed exponentially from  $10^{-1}$  to  $10^{-4}$  for 50 epochs. It takes about 6 hours to train a DnCNN model.

To evaluate the model, a dataset containing 68 natural images from Berkeley segmentation dataset (BSD68) [35] is adopted. The average PSNR results of different methods are shown in Table 4.1. As one can see, compared to the benchmark BM3D, the methods MLP and TNRD have a PSNR gain of about 0.35 dB. Notably, DnCNN outperforms BM3D by 0.6 dB on all the three noise levels.

**Table 4.1** The average PSNR (dB) results of different methods on the BSD68 dataset. The best results are highlighted in bold

Noise levels	BM3D [5]	WNNM [13]	MLP [5]	TNRD [8]	DnCNN
$\sigma = 15$	31.07	31.37	–	31.42	<b>31.73</b>
$\sigma = 25$	28.57	28.83	28.96	28.92	<b>29.23</b>
$\sigma = 50$	25.62	25.87	26.03	25.97	<b>26.23</b>

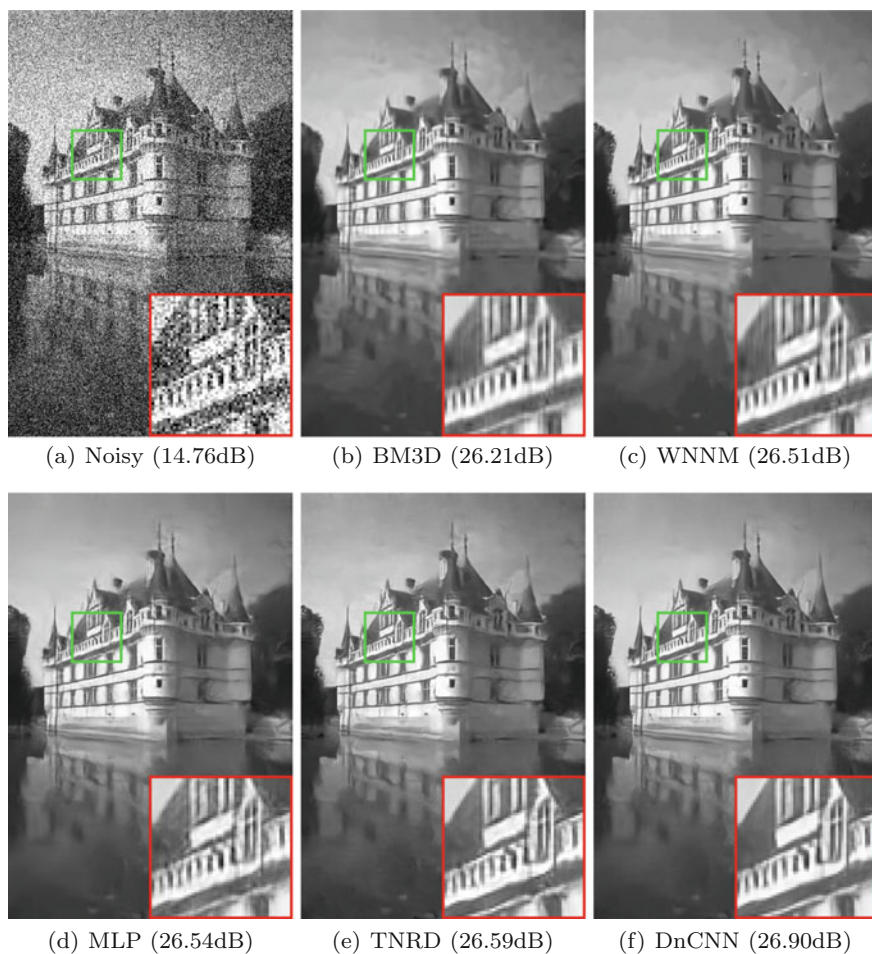
**Fig. 4.4** Denoising results of different methods on noise level 50

Figure 4.4 illustrates the visual results of different methods. It can be seen that BM3D, WNNM and MLP tend to produce over-smooth edges and textures. While preserving sharp edges and fine details, TNRD is likely to generate artifacts in the smooth region. In contrast, DnCNN can not only recover sharp edges and fine details but also yield visually pleasant results in the smooth region.

### 4.3 CNN for Fast and Flexible Image Denoising

In order to handle practical image denoising problems, a flexible image denoiser is expected to have the following desirable properties: (i) it is able to perform denoising using a single model; (ii) it is efficient, effective, and user-friendly; and (iii) it can handle spatially variant noise. Such a denoiser can be directly deployed to recover the clean image when the noise level is known or can be well estimated. When the noise level is unknown or is difficult to estimate, the denoiser should allow the user to adaptively control the trade-off between noise reduction and details preservation. Furthermore, the noise can be spatially variant and the denoiser should be flexible enough to handle spatially variant noise.

The FFDNet which is shown in Fig. 4.5 is proposed to meet with such desirable properties. Specifically, one of the major differences of FFDNet is that it can be formulated as  $\mathbf{x} = \mathcal{F}(\mathbf{y}, \mathbf{M}; \Theta)$ , where  $\mathbf{M}$  is a noise level map. Here, the noise level map  $\mathbf{M}$  is modeled as an input and the model parameters  $\Theta$  are invariant to noise level. Hence, FFDNet provides a flexible way to handle various types of noise with a single network. By introducing a noise level map as input, network design and training methods are required to be further studied for effective and efficient denoising. Another main difference of FFDNet is that it works on downsampled subimages, which largely accelerates the training and testing speed, and enlarges the receptive field as well.

#### 4.3.1 Network Architecture: FFDNet

As shown in Fig. 4.5, the first layer of FFDNet is a reversible downsampling operator which reshapes an  $H \times W$  noisy image  $\mathbf{y}$  into four downsampled subimages. Then,

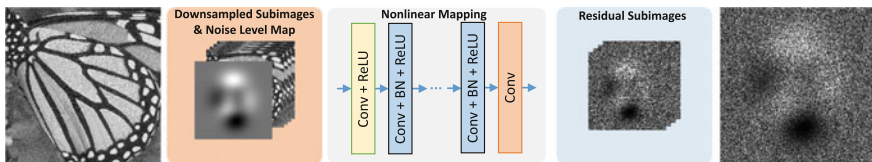


Fig. 4.5 The architecture of the FFDNet network



a tunable downsampled noise level map  $\mathbf{M}$  with the downsampled subimages is concatenated to form a tensor  $\tilde{\mathbf{y}}$  of size  $\frac{H}{2} \times \frac{W}{2} \times (4c + 1)$  as the inputs to CNN. The following CNN consists of a series of  $3 \times 3$  convolutional layers. Each layer is composed of three types of operations: Convolution (Conv), Rectified Linear Units (ReLU) [22], and Batch Normalization (BN) [17]. More specifically, “Conv+ReLU” is adopted for the first convolutional layer, “Conv+BN+ReLU” for the middle layers, and “Conv” for the last convolutional layer. Zero-padding is employed to keep the size of feature maps unchanged after each convolution. After the last convolutional layer, an upscaling operation is applied as the reverse operator of the downsampling operator applied in the input stage to produce the residual noisy image  $\tilde{\mathbf{v}}$  of size  $H \times W \times c$ . The denoised image is then obtained by  $\tilde{\mathbf{x}} = \mathbf{y} - \tilde{\mathbf{v}}$ . Since FFDNet operates on downsampled subimages, it is not necessary to employ the dilated convolution [49] to further increase the receptive field.

By considering the balance of complexity and performance, the number of convolutional layers are set to 15 for grayscale image and 12 for color image. As for the number of feature maps, they are set to 64 for grayscale image and 96 for color image. The reason of using different settings for grayscale and color images is twofold. First, since there are high dependencies among the R, G, B channels, using a smaller number convolutional layers is good enough to exploit the interchannel dependency for denoising. Second, color image has more channels as input, and hence more feature (i.e., a larger number of feature maps) is required.

### 4.3.2 Taking Noise Level as CNN Input

Most of the deep learning based denoising methods such as the MLP [5] and convolutional neural network (CNN) based methods [18, 50] are limited in flexibility, and the learned model is usually tailored to a specific noise level. From the perspective of regression, some CNN-based denoisers such as DnCNN aim to learn a mapping function  $\mathbf{x} = \mathcal{F}(\mathbf{y}; \Theta_\sigma)$  between the input noisy observation  $\mathbf{y}$  and the desired output  $\mathbf{x}$ . The model parameters  $\Theta_\sigma$  are trained for noisy images corrupted by AWGN with a fixed noise level  $\sigma$ , while the trained model with  $\Theta_\sigma$  is hard to be directly deployed to images with other noise levels.

From the viewpoint of MAP framework, the solution of Eq. (4.2) actually defines an implicit function  $\hat{\mathbf{x}} = \mathcal{F}(\mathbf{y}, \sigma, \lambda; \Theta)$  of the noisy image  $\mathbf{y}$ , noise level  $\sigma$ , and parameter  $\lambda$ . Since  $\lambda$  can be absorbed into  $\sigma$ , the solution  $\hat{\mathbf{x}} = \mathcal{F}(\mathbf{y}, \sigma, \lambda; \Theta)$  can be rewritten as  $\hat{\mathbf{x}} = \mathcal{F}(\mathbf{y}, \sigma; \Theta)$ . In this sense, setting noise level  $\sigma$  also plays the role of setting  $\lambda$  to control the trade-off between noise reduction and detail preservation.

Since the inputs  $\mathbf{y}$  and  $\sigma$  have different dimensions, it is not easy to directly feed them into CNN. To resolve this, a simple dimension stretching strategy can be employed to stretch the noise level  $\sigma$  into a noise level map  $\mathbf{M}$ . In  $\mathbf{M}$ , all the elements are  $\sigma$ . Then, the formulation is changed into  $\hat{\mathbf{x}} = \mathcal{F}(\mathbf{y}, \mathbf{M}; \Theta)$ . For color image,  $\mathbf{M}$  can have multiple channels to represent the noise level map of R, G, B channels and can also be extended to degradation map which parameterizes the degradation



process [52]. As such, a trained CNN model is expected to inherit the flexibility of handling images with different noise levels, even spatially variant noises via a single model.

### 4.3.3 *Single Non-blind Model Versus Single Blind Model*

So far, we have know that it is possible to learn a single model for blind and non-blind Gaussian denoising, respectively, it is necessary to point out their differences. First, the generalization ability is different. Although the blind model performs favorably for synthetic AWGN removal without knowing the noise level, it does not generalize well to real noisy images. In contrast, the non-blind model with noise level map has the ability to control the trade-off between noise removal and detail preservation, thus it can deal with real noise to some extent. In addition, the non-blind model can handle the out-of-range noise levels, whereas the blind one lacks such an ability. Second, the performance for AWGN removal is different. The non-blind model with noise level map has better performance for AWGN removal than the blind one. This phenomenon has also been recognized in the task of SISR [32]. Third, the application range is different. The non-blind model can be plugged into model-based optimization methods to solve various image restoration tasks, such as image deblurring, SISR and image inpainting. However, the blind model does not have this merit.

### 4.3.4 *Denoising on Subimages*

Efficiency is another crucial issue for practical CNN-based denoising. One straightforward idea is to reduce the depth and number of filters. However, such a strategy will sacrifice much the modeling capacity and receptive field of CNN [50]. Shi et al. [39] proposed to extract deep features directly from the low-resolution image for super-resolution, and introduced a sub-pixel convolution layer to improve computational efficiency. In the application of image denoising, we introduce a reversible downsampling layer to reshape the input image into a set of small subimages. Here, the downsampling factor is set to 2 since it can largely improve the speed by slightly reducing modeling capacity. The CNN is deployed on the subimages, and finally a sub-pixel convolution layer is adopted to reverse the downsampling process.

Denoising on downsampled subimages can also effectively expand the receptive field which in turn leads to a moderate network depth. For example, the proposed network with a depth of 15 and  $3 \times 3$  convolution will have a large receptive field of  $62 \times 62$ . In contrast, a plain 15-layer CNN only has a receptive field size of  $31 \times 31$ . We note that the receptive field of most state-of-the-art denoising methods ranges from  $35 \times 35$  to  $61 \times 61$  [50]. Further increase of receptive field actually benefits

little in improving denoising performance [27]. What is more, the introduction of subsampling and sub-pixel convolution is also effective in reducing the memory burden.

### 4.3.5 Dataset Generation and Network Training

To train the FFDNet model, we need to prepare a training dataset of patches  $(\mathbf{y}, \mathbf{M}; \mathbf{x})$ . Here,  $\mathbf{y}$  is obtained by adding AWGN to latent image  $\mathbf{x}$ , and  $\mathbf{M}$  is the corresponding noise level map. The reasons of using AWGN to generate the training dataset are as follows. First, AWGN is a natural choice when there is no specific prior information on noise source. Second, real-world noise can be locally approximated as AWGN [24]. It is worth noting that FFDNet model is trained on the noisy images  $\mathbf{y} = \mathbf{x} + \mathbf{v}$  without quantization to 8-bit integer values. Though the real noisy images are generally 8-bit quantized, we empirically found that the learned model still works effectively on real noisy images. For the noise level of each noisy, it is uniformly sampled from the range of [0, 75].

The ADAM algorithm [20] is adopted to optimize FFDNet by minimizing the mean squared error loss. The learning rate starts from  $10^{-3}$  and reduces to  $10^{-4}$  when the training error stops decreasing. When the training error keeps unchanged in five sequential epochs, we merge the parameters of each batch normalization into the adjacent convolution filters. Then, a smaller learning rate of  $10^{-6}$  is adopted for additional 20 epochs to fine-tune the FFDNet model. As for the other hyper-parameters of ADAM, we use their default settings. The mini-batch size is set as 128, and the rotation and flip based data augmentation is also adopted during training. The FFDNet models are trained in Matlab (R2015b) environment with MatConvNet package [44] and an Nvidia Titan X Pascal GPU. The training of a single model can be done in about one day.

### 4.3.6 Experiments on Synthetic and Real Images

#### 4.3.6.1 Experiments on AWGN Removal

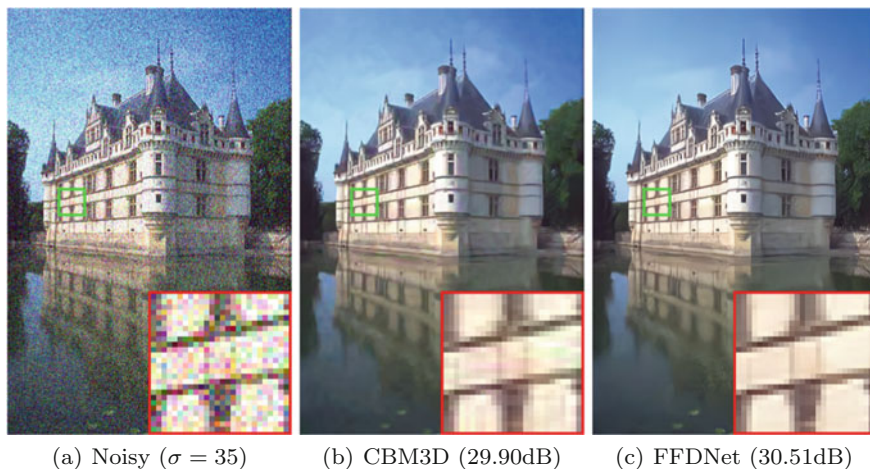
The PSNR results of different methods on BSD68 dataset are reported in Table 4.2, from which we have the following observations. First, FFDNet surpasses BM3D by a large margin and outperforms WNNM, MLP, and TNRD by about 0.2 dB for a wide range of noise levels. Second, FFDNet is slightly inferior to DnCNN when the noise level is low (e.g.,  $\sigma \leq 25$ ), but gradually outperforms DnCNN with the increase of noise level (e.g.,  $\sigma > 25$ ). This phenomenon may result from the trade-off between receptive field size and modeling capacity. FFDNet has a larger receptive field than DnCNN, thus favoring for removing strong noise, while DnCNN has better modeling capacity which is beneficial for denoising images with lower noise level.

**Table 4.2** The average PSNR (dB) results of different methods on BSD68 with noise levels 15, 25, 35, 50, and 75. The best results are highlighted in bold

Methods	BM3D [11]	WNNM [13]	MLP [5]	TNRD [8]	DnCNN [50]	FFDNet
$\sigma = 15$	31.07	31.37	–	31.42	<b>31.72</b>	31.62
$\sigma = 25$	28.57	28.83	28.96	28.92	<b>29.23</b>	29.19
$\sigma = 35$	27.08	27.30	27.50	–	27.69	<b>27.73</b>
$\sigma = 50$	25.62	25.87	26.03	25.97	26.23	<b>26.30</b>
$\sigma = 75$	24.21	24.40	24.59	–	24.64	<b>24.78</b>

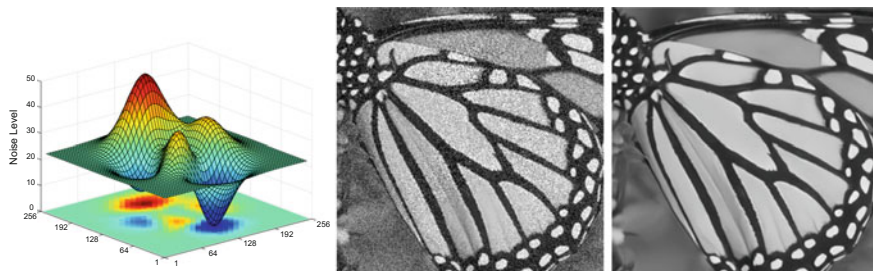
**Table 4.3** The average PSNR (dB) results of CBM3D and FFDNet on CBSD68 dataset with noise levels 15, 25, 35, 50, and 75

Methods	$\sigma = 15$	$\sigma = 25$	$\sigma = 35$	$\sigma = 50$	$\sigma = 75$
CBM3D	33.52	30.71	28.89	27.38	25.74
FFDNet	33.80	31.18	29.57	27.96	26.24



**Fig. 4.6** Color image denoising results by CBM3D and FFDNet

Table 4.3 reports the performance of CBM3D and FFDNet on color version of BSD68 datasets, and Fig. 4.6 presents the visual comparisons. It can be seen that FFDNet consistently outperforms CBM3D on different noise levels in terms of both quantitative and qualitative evaluation.



**Fig. 4.7** Examples of FFDNet on removing spatially variant AWGN. Left: noise level maps. Middle: noisy images with PSNR 20.55 dB. Right: denoised images (PSNR: 29.97 dB) by FFDNet

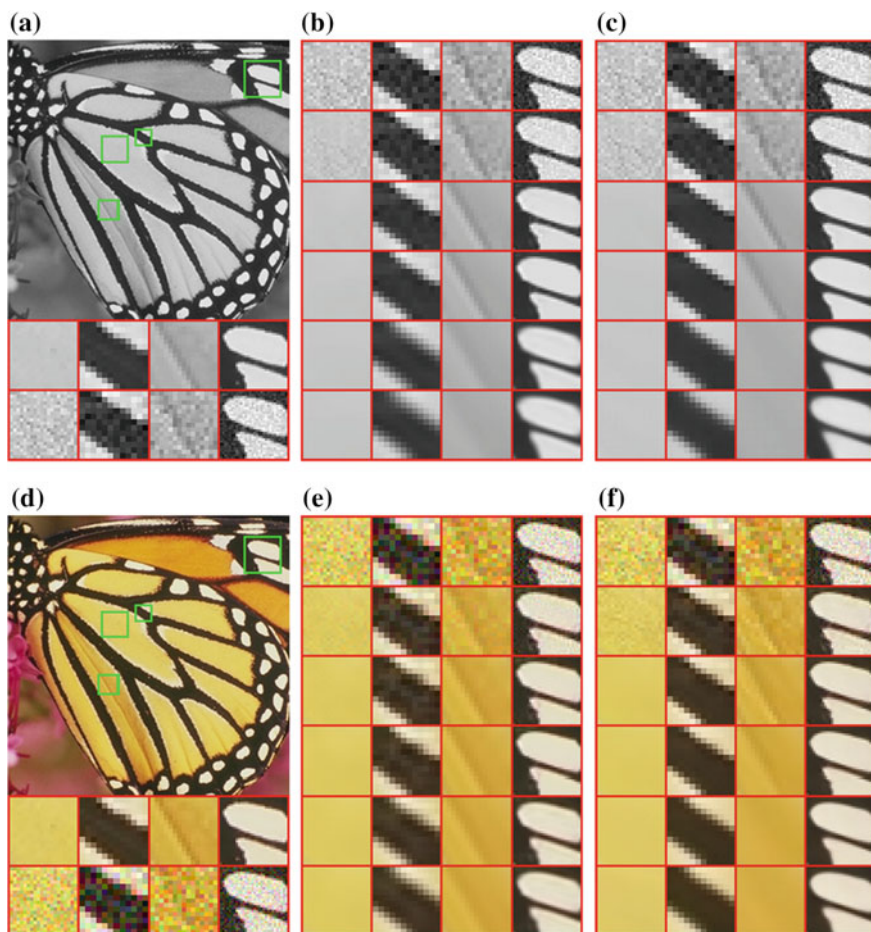
#### 4.3.6.2 Experiments on Spatially Variant AWGN Removal

We then test the flexibility of FFDNet to deal with spatially variant AWGN. To synthesize spatially variant AWGN, we first generate an AWGN image  $\mathbf{v}$  with the standard deviation 1 and a noise level map  $\mathbf{M}$  of the same size. Then, elementwise multiplication is applied on  $\mathbf{v}$  and  $\mathbf{M}$  to produce the spatially variant AWGN, i.e.,  $\mathbf{v} \odot \mathbf{M}$ . In the denoising stage, we take the bilinearly downsampled noise level map as the input to FFDNet. Figure 4.7 shows the denoising result of FFDNet for a spatially variant AWGN. One can see that FFDNet is flexible and powerful to remove spatially variant AWGN.

#### 4.3.6.3 Experiments on Noise Level Sensitivity

In practical applications, the noise level map may not be accurately estimated from the noisy observation, and mismatch between the input and real noise levels is inevitable. If the input noise level is lower than the real noise level, the noise cannot be completely removed. Therefore, users often prefer to set a higher noise level to guarantee the removal of noise. However, this may also remove too much image details together with noise. In this subsection, we evaluate FFDNet in comparison with benchmark BM3D by varying different input noise levels for a given ground-truth noise level.

Figure 4.8 shows the visual comparisons between BM3D/CBM3D and FFDNet by setting different input noise levels to denoise a noisy image. Four typical image structures, including flat region, sharp edge, line with high contrast, and line with low contrast, are selected for visual comparison to investigate the noise level sensitivity of BM3D and FFDNet. The following observations can be obtained. The best visual quality is obtained when the input noise level matches the ground-truth one. BM3D and FFDNet produce similar visual results with lower input noise levels, while they exhibit certain difference with higher input noise levels. Both of them will smooth out noise in flat regions, and gradually smooth out image structures with the increase in input noise levels. Particularly, FFDNet may wipe out some low contrast line structure, whereas BM3D can still preserve the mean patch regardless of the input



**Fig. 4.8** Visual comparisons between FFDNet and BM3D/CBM3D by setting different input noise levels to denoise a noisy image. **a** From top to bottom: ground-truth image, four clean zoom-in regions, and the corresponding noisy regions (AWGN, noise level 15). **b** From top to bottom: denoising results by BM3D with input noise levels 5, 10, 15, 20, 50, and 75, respectively. **c** Results by FFDNet with the same settings as in **(b)**. **d** From top to bottom: ground-truth image, four clean zoom-in regions, and the corresponding noisy regions (AWGN, noise level 25). **e** From top to bottom: denoising results by CBM3D with input noise levels 10, 20, 25, 30, 45, and 60, respectively. **f** Results by FFDNet with the same settings as in **(e)**

noise levels due to its use of nonlocal information. Using a higher input noise level can generally produce better visual results than using a lower one. In addition, there is no much visual difference when the input noise level is a little higher than the ground-truth one.

#### 4.3.6.4 Experiments on Real Noisy Images

In this subsection, real noisy images are used to further assess the practicability of FFDNet. However, such an evaluation is difficult to conduct due to the following reasons. (i) Both the ground-truth clean image and noise level are unknown for real noisy image. (ii) The real noise comes from various sources such as camera imaging pipeline (e.g., shot noise, amplifier noise and quantization noise), scanning, lossy compression and image resizing [10, 28], and it is generally non-Gaussian, spatially variant, and signal-dependent. As a result, the AWGN assumption in many denoising algorithms does not hold, and the associated noise level estimation methods may not work well for real noisy images.

Instead of adopting any noise level estimation methods, we adopt an interactive strategy to handle real noisy images. First of all, we empirically found that the assumption of spatially invariant noise usually works well for most real noisy images. We then employ a set of typical input noise levels to produce multiple outputs, and select the one which has best trade-off between noise reduction and details preservation. Second, for spatially variant noise, we sample several typical image patches which represent the distinct regions of different noise levels and apply different input noise levels to them. By observing the denoising results, we then choose the proper noise level for each typical patch. The noise levels at other locations are interpolated from the noise levels of the typical patches. An approximation of nonuniform noise level map can then be obtained. In our following experiments, unless otherwise specified, we assume spatially invariant noise for the real noisy images.

Since there is no ground-truth image for a real noisy image, visual comparison is employed to evaluate the performance of FFDNet. We choose BM3D for comparison because it is widely accepted as a benchmark for denoising applications. Given a noisy image, the same input noise level is used for BM3D and FFDNet.

Figure 4.9 compares the grayscale image denoising results on four noisy images. As one can see, BM3D and FFDNet exhibit similar behaviors to those on denoising grayscale images. In particular, for image “*Building*” which contains some structured noises, BM3D fails to yield visually pleasant results because the structured noises fit the nonlocal self-similarity prior. In contrast, FFDNet removes such noise without losing underlying image textures. Figure 4.10 shows the denoising results of CBM3D and FFDNet on four color noisy images. It can be seen that FFDNet can handle various kinds of noises, including Gaussian-like noise (see image “*Pattern*”), JPEG lossy compression noise (see image “*Audrey Hepburn*”), and low-frequency noise (see image “*Boy*”). Similarly, from the denoising results of “*Boy*”, one can see that CBM3D remains the structured low-frequency noise unremoved whereas FFDNet removes successfully such kind of noise. As a result, we can conclude that while the nonlocal self-similarity prior helps to remove random noise, it hinders the removal of structured noise. In comparison, the prior implicitly learned by CNN is able to remove both random noise and structured noise.

Figure 4.11 shows a more challenging example to demonstrate the advantage of FFDNet for denoising noisy images with spatially variant noise. As one can see, while FFDNet with a small input noise level can recover the details of regions with



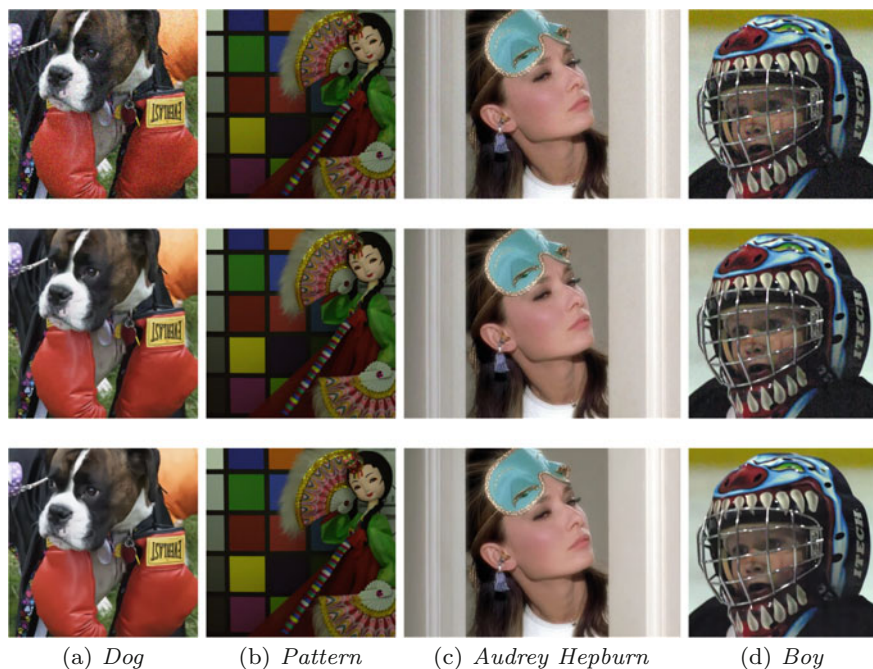


**Fig. 4.9** Grayscale image denoising results by different methods on real noisy images. From top to bottom: noisy images, denoised images by BM3D, denoised images by FFDNet. **a**  $\sigma = 15$ ; **b**  $\sigma = 10$ ; **c**  $\sigma = 20$ ; **d**  $\sigma = 20$

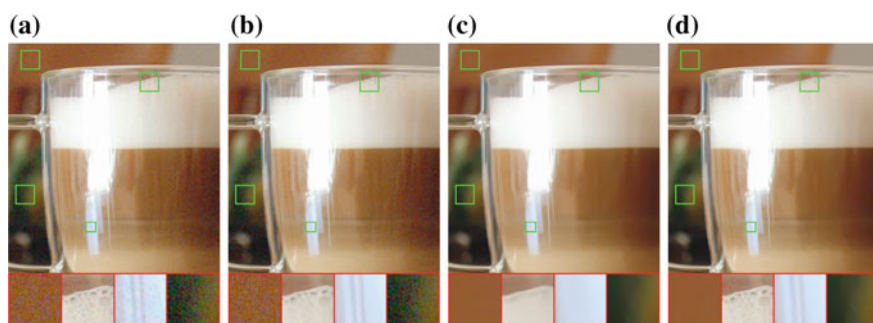
low noise level, it fails to remove strong noise. On the other hand, FFDNet with a large input noise level can remove strong noise but it will also smooth out the details in the region with low noise level. In comparison, the denoising result with a proper nonuniform noise level map not only preserves image details but also removes the strong noise.

### 4.3.7 Running Time

Table 4.4 lists the running time results of BM3D, DnCNN and FFDNet for denoising grayscale level and color images with size  $256 \times 256$ ,  $512 \times 512$ , and  $1,024 \times 1,024$ . The evaluation was performed in Matlab (R2015b) environment on a computer with a six-core Intel(R) Core(TM) i7-5820K CPU @ 3.3 GHz, 32 GB of RAM and a



**Fig. 4.10** Color image denoising results by different methods on real noisy images. From top to bottom: noisy images, denoised images by CBM3D, denoised images by FFDNet. **a**  $\sigma = 28$ ; **b**  $\sigma = 12$ ; **c**  $\sigma = 10$ ; **d**  $\sigma = 45$



**Fig. 4.11** An example of FFDNet on image “Glass” with spatially variant noise. **a** Noisy image; **b** denoised image by FFDNet with  $\sigma = 10$ ; **c** denoised image by FFDNet with  $\sigma = 35$ ; **d** denoised image by FFDNet with nonuniform noise level map



**Table 4.4** Running time (in seconds) of different methods for denoising images with size  $256 \times 256$ ,  $512 \times 512$ , and  $1,024 \times 1,024$ 

Methods	Device	$256 \times 256$		$512 \times 512$		$1024 \times 1024$	
		Gray	Color	Gray	Color	Gray	Color
BM3D	CPU(ST)	0.59	0.98	2.52	3.57	10.77	20.15
DnCNN	CPU(ST)	2.14	2.44	8.63	9.85	32.82	38.11
	CPU(MT)	0.74	0.98	3.41	4.10	12.10	15.48
	GPU	0.011	0.014	0.033	0.040	0.124	0.167
FFDNet	CPU(ST)	0.44	0.52	1.81	2.14	7.24	8.51
	CPU(MT)	0.18	0.19	0.73	0.79	2.96	3.15
	GPU	0.006	0.007	0.012	0.016	0.038	0.054

Nvidia Titan X Pascal GPU. For BM3D, we evaluate its running time by denoising images with noise level 25. For DnCNN, the grayscale and color image denoising models have 17 and 20 convolutional layers, respectively. The Nvidia cuDNN-v5.1 deep learning library is used to accelerate the computation of DnCNN and FFDNet. The memory transfer time between CPU and GPU is also counted. Note that DnCNN and FFDNet can be implemented with both single-threaded (ST) and multi-threaded (MT) CPU computations.

From Table 4.4, we have the following observations. First, BM3D spends much more time on denoising color images than grayscale images. The reason is that, compared to gray-BM3D, CBM3D needs extra time to denoise the chrominance components after luminance-chrominance color transformation. Second, while DnCNN can benefit from GPU computation for fast implementation, it has comparable CPU time to BM3D. Third, FFDNet spends almost the same time for processing grayscale and color images. More specifically, FFDNet with multi-threaded implementation is about three times faster than DnCNN and BM3D on CPU, and much faster than DnCNN on GPU. Even with single-threaded implementation, FFDNet is also faster than BM3D. Taking denoising performance and flexibility into consideration, FFDNet is very competitive for practical applications.

#### 4.4 CNN Denoiser Prior Based Image Restoration

Motivated by the impressive achievement on image denoising, it is natural to ask whether CNNs can be applied to more general image restoration tasks. Although CNNs can be directly adopted with promising performance and fast testing speed, their application range is greatly restricted by the specialized task. In contrast, model-based optimization methods are flexible for handling different inverse problems but are usually time-consuming with sophisticated priors for the purpose of good performance. Fortunately, it has been revealed that, with the aid of variable splitting

techniques such as alternating direction method of multipliers (ADMM) algorithm, half quadratic splitting (HQS) algorithm, and the primal-dual algorithm [6], denoiser prior can be plugged in as a modular part of model-based optimization methods to solve other image restoration problems, and particularly, the regularization term only corresponds to a denoising subproblem [7, 16, 33, 41, 53]. Consequently, such an integration induces considerable advantage when the denoiser is based on CNN.

Very recently, various methods have been proposed to incorporate the CNN denoiser prior into model-based optimization methods. In those methods, the CNN denoiser can be either pretrained or jointly trained with data fidelity term for a specific task. In other words, there exist two general CNN denoiser prior based frameworks, i.e., model-based optimization and discriminative learning, for different image restoration tasks. In this section, we focus on the former since once the CNN denoiser is trained, no additional training is needed for other tasks. As for the variable splitting algorithm, we choose half quadratic splitting (HQS) algorithm due to its simplicity.

In the following, we first give a brief review of HQS algorithm and then show how to plug CNN denoiser into the optimization procedure to solve other image restoration problems, including image deblurring, single image super-resolution, and image inpainting.

#### 4.4.1 Half Quadratic Splitting Algorithm

In general, the purpose of image restoration is to recover the latent clean image  $\mathbf{x}$  from its degraded observation  $\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{v}$ , where  $\mathbf{H}$  is a degradation matrix,  $\mathbf{v}$  is additive white Gaussian noise of standard deviation  $\sigma$ . By specifying different degradation matrices, one can correspondingly get different image restoration tasks. Three classical IR tasks would be image denoising when  $\mathbf{H}$  is an identity matrix, image deblurring when  $\mathbf{H}$  is a blurring operator, and image super-resolution when  $\mathbf{H}$  is a composite operator of blurring and downsampling.

Due to the ill-posed nature of general image restoration problems, regularization needs to be imposed to constrain the solution. Mathematically, the latent clean image of a degraded image  $\mathbf{y}$  can be estimated by solving the following MAP problem:

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x}} \frac{1}{2\sigma^2} \|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2 + \lambda\Phi(\mathbf{x}) \quad (4.10)$$

where the solution minimizes an energy function composed of a data fidelity term  $\frac{1}{2\sigma^2} \|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2$ , a regularization term  $\Phi(\mathbf{x})$  and a trade-off parameter  $\lambda$ .

In HQS, by introducing an auxiliary variable  $\mathbf{z}$ , Eq. (4.10) can be reformulated as a constrained optimization problem which is given by

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x}} \frac{1}{2\sigma^2} \|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2 + \lambda\Phi(\mathbf{z}) \quad s.t. \quad \mathbf{z} = \mathbf{x} \quad (4.11)$$

Then, HQS tries to minimize the following cost function:

$$\mathcal{L}_\mu(\mathbf{x}, \mathbf{z}) = \frac{1}{2\sigma^2} \|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2 + \lambda\Phi(\mathbf{z}) + \frac{\mu}{2} \|\mathbf{z} - \mathbf{x}\|^2 \quad (4.12)$$

where  $\mu$  is a penalty parameter which varies iteratively in a non-descending order. Equation (4.12) can be solved via the following iterative scheme:

$$\begin{cases} x_{k+1} = \arg \min_{\mathbf{x}} \|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2 + \mu\sigma^2 \|\mathbf{x} - \mathbf{z}_k\|^2 & (4.13a) \\ \mathbf{z}_{k+1} = \arg \min_{\mathbf{z}} \frac{\mu}{2} \|\mathbf{z} - \mathbf{x}_{k+1}\|^2 + \lambda\Phi(\mathbf{z}). & (4.13b) \end{cases}$$

As one can see, the data fidelity term and regularization term are decoupled into two individual subproblems. Specifically, the data fidelity term is associated with a quadratic regularized least-squares problem (Eq. (4.13a)) which has various fast solutions for different degradation matrices. A direct solution is given by

$$\mathbf{x}_{k+1} = (\mathbf{H}^T \mathbf{H} + \mu\sigma^2 \mathbf{I})^{-1} (\mathbf{H}^T \mathbf{y} + \mu\sigma^2 \mathbf{z}_k) \quad (4.14)$$

The regularization term is involved in Eq. (4.13b) which can be rewritten as

$$\mathbf{z}_{k+1} = \arg \min_{\mathbf{z}} \frac{1}{2(\sqrt{1/\mu})^2} \|\mathbf{x}_{k+1} - \mathbf{z}\|^2 + \lambda\Phi(\mathbf{z}) \quad (4.15)$$

Equation (4.15) corresponds to denoising the image  $\mathbf{x}_{k+1}$  by a CNN-based Gaussian denoiser with noise level  $\sqrt{1/\mu}$ . As a consequence, any CNN-based Gaussian denoisers can be acted as a modular part to solve Eq. (4.10). To address this, Eq. (4.15) can be rewritten as

$$\mathbf{z}_{k+1} = \mathcal{F}(\mathbf{x}_{k+1}, \sqrt{1/\mu}) \quad (4.16)$$

We point out that the CNN denoiser from Eq. (4.15) should be designed for AWGN removal and the noisy image in the training should not be quantized to 8-bit integer values.

So far, we have obtained that the image prior  $\Phi(\cdot)$  can be implicitly replaced by a denoiser prior. Such a promising property actually offers several advantages. First, it enables to use fast and effective CNN-based denoisers to solve a variety of inverse problems. Second, the explicit image prior  $\Phi(\cdot)$  can be unknown in solving Eq. (4.10). Third, several complementary denoisers which exploit different image priors can be jointly utilized to solve one specific problem.

### 4.4.2 CNN Denoisers and Parameter Setting

For the architecture of the CNN denoiser, it consists of seven layers with three different blocks, i.e., “Dilated Convolution+ReLU” block in the first layer, five “Dilated Convolution+Batch Normalization+ReLU” blocks in the middle layers, and “Dilated Convolution” block in the last layer. The dilation factors of  $(3 \times 3)$  dilated convolutions from first layer to the last layer are set to 1, 2, 3, 4, 3, 2, and 1, respectively. The number of feature maps in each middle layer is set to 64. We trained a set of denoisers on noise level range  $[0, 50]$  and divided it by a step size of 2 for each model, resulting in a set of 25 denoisers for each grayscale and color image prior modeling.

Once the denoisers are provided, the subsequent crucial issue would be parameter setting. There involve two parameters,  $\lambda$  and  $\mu$ , to tune. For the setting of  $\lambda$ , since it is implicitly optimized in the CNN denoiser and can be absorbed into  $\sigma$ , one can instead tune  $\sigma$  to obtain the best results. In practice, this can be achieved by multiplying  $\sigma$  by a scalar around 1. For the setting of  $\mu$ , it is better to set the noise level of denoiser in each iteration to implicitly determine  $\mu$ . Note that the noise level of denoiser  $\sqrt{1/\mu}$  should be set from large to small. In the following experiments, it is decayed exponentially from 49 to a value in  $[1, 15]$  for 30 iterations. Note that all the experimental results are reproducible, and the source code can be downloaded from <https://github.com/csxn/IRCNN>.

### 4.4.3 Image Deblurring

For image deblurring, by assuming the convolution is carried out with circular boundary conditions, the fast implementation of Eq. (4.13a) is given by

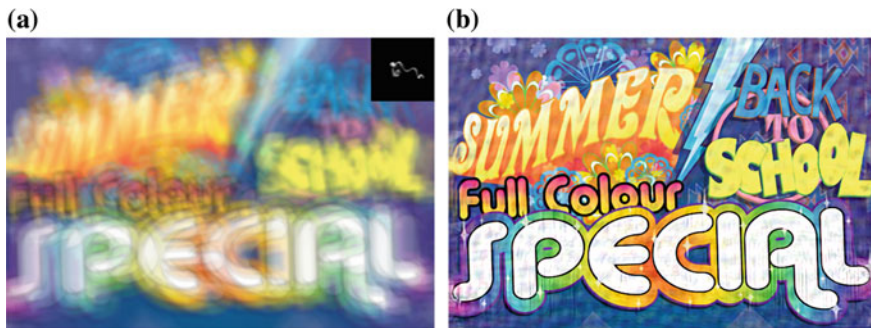
$$\mathbf{x}_{k+1} = \mathcal{F}^{-1} \left( \frac{\overline{\mathcal{F}(\mathbf{k})} \mathcal{F}(\mathbf{y}) + \mu \sigma^2 \mathcal{F}(\mathbf{z}_k)}{\overline{\mathcal{F}(\mathbf{k})} \mathcal{F}(\mathbf{k}) + \mu \sigma^2} \right) \quad (4.17)$$

where the  $\mathcal{F}(\cdot)$  and  $\mathcal{F}^{-1}(\cdot)$  denote the fast Fourier transform (FFT) and inverse FFT,  $\overline{\mathcal{F}(\cdot)}$  denotes complex conjugate of  $\mathcal{F}(\cdot)$  and  $\mathbf{k}$  is a blurring kernel corresponding to the degradation matrix  $\mathbf{H}$ .

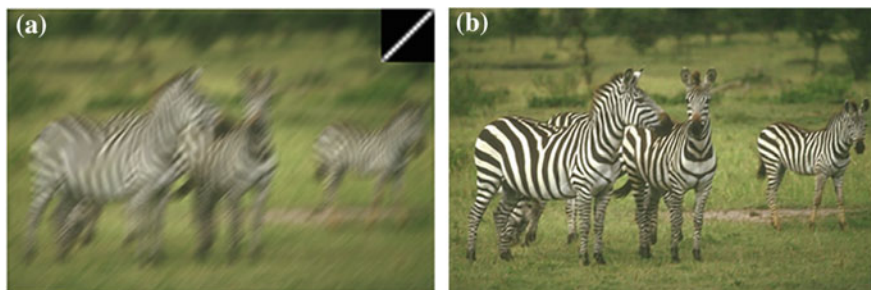
Figure 4.12 gives an example of IRCNN for image deblurring. It can be seen that IRCNN can yield visually pleasant result with sharp edges and fine details.

### 4.4.4 Single Image Super-Resolution

There exist several degradation settings for single image super-resolution (SISR), among which bicubic degradation (default setting of Matlab function *imresize*) and



**Fig. 4.12** An example of IRCNN for image deblurring. **a** Blurred image with estimated kernel by Pan et al. [30]; **b** Deblurring result



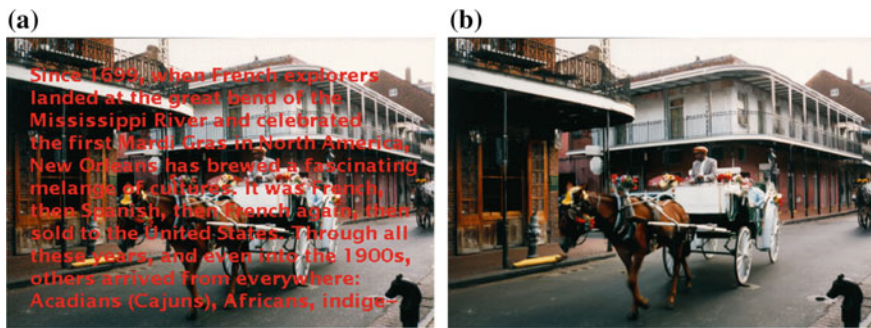
**Fig. 4.13** An example of IRCNN for single image super-resolution (the blur kernel is a motion blur, the scale factor is 2). **a** LR image with motion blur kernel; **b** SISR result

Gaussian blurring followed by anti-aliasing downsampling are the two most widely used ones. For these two degradations, we use the following back-projection iteration to solve Eq. (4.13a),

$$\mathbf{x}_{k+1} = \mathbf{z}_k - \alpha(\mathbf{y} - \mathbf{z}_k \downarrow_{sf}) \uparrow_{bicubic}^{sf} \tag{4.18}$$

where  $\downarrow_{sf}$  denotes the degradation operator with downscaling factor  $sf$ ,  $\uparrow_{bicubic}^{sf}$  represents bicubic interpolation operator with upscaling factor  $sf$ , and  $\alpha$  is the step size which is fixed to 1.75.

It is worth noting that when the downsampler is the standard  $K$ -fold downsampler (Matlab function *downsample*), Eq. (4.13a) has a fast closed-form solution by benefiting FFT [7]. Furthermore, the blur kernel can go beyond Gaussian blur. Figure 4.13 shows an example of IRCNN for super-resolving LR image degraded by motion blurring and standard  $K$ -fold downsampler. It can be seen that the super-resolved image is much more visually pleasing than the LR image.



**Fig. 4.14** An example of IRCNN for image inpainting. **a** Original image with overlaid text; **b** inpainting result

### 4.4.5 Image Inpainting

For image inpainting,  $\mathbf{H}\mathbf{x}$  can be rewritten as  $\mathbf{M} \odot \mathbf{x}$ , where  $\mathbf{M}$  is matrix with binary elements indicating the missing pixels of  $\mathbf{y}$ , and  $\odot$  denotes elementwise multiplication. A closed-form solution of Eq. (4.13a) is given by

$$\mathbf{x}_{k+1} = (\mathbf{M} \odot \mathbf{y} + \mu\sigma^2\mathbf{z}_k) \oslash (\mathbf{M} + \mu\sigma^2) \quad (4.19)$$

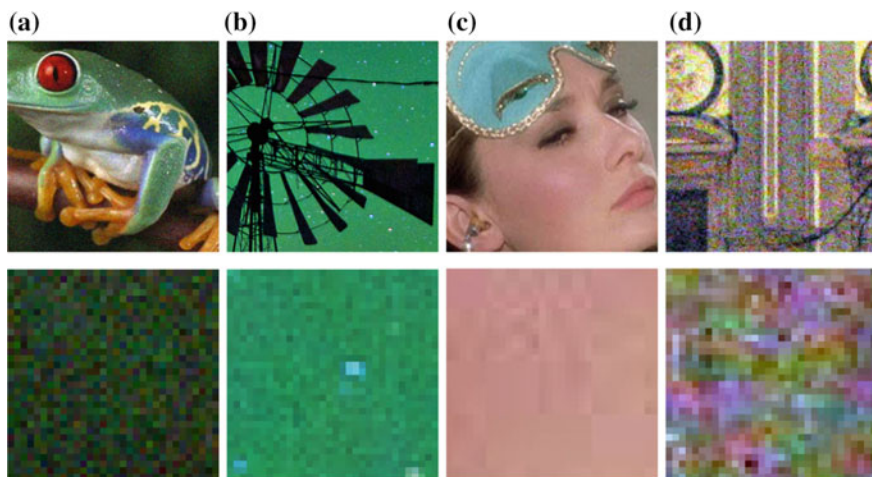
where  $\oslash$  denotes elementwise division.

Figure 4.14 shows an example of IRCNN for image inpainting. As one can see, there is no visible artifacts in the inpainted image.

## 4.5 Challenges and Possible Solutions

While the image denoising for AWGN removal has been well-studied, little work has been done on real image denoising. The main difficulty arises from the fact that real noises are much more sophisticated than AWGN and it is not an easy task to thoroughly evaluate the performance of a denoiser. Figure 4.15 shows four typical noise types in real world. It can be seen that the characteristics of those noises are very different and a single noise level may be not enough to parameterize those noise types. In most cases, a denoiser can only work well under a certain noise model. For example, a denoising model trained for AWGN removal is not effective for mixed Gaussian and Poisson noise removal. This is intuitively reasonable because the CNN-based methods can be treated as general cases of Eq. (4.3) and the important data fidelity term corresponds to the degradation process. In spite of this, the image denoising for AWGN removal is still valuable due to the following reasons. First, it is an ideal test bed to evaluate the effectiveness of different CNN-based denoising models and learning algorithms. Second, in the unrolled inference





**Fig. 4.15** Different noise types. **a** Additive white Gaussian noise; **b** interchannel correlated Gaussian noise; **c** JPEG compression noise; **d** low-frequency noise

via variable splitting techniques, many image restoration problems can be addressed by sequentially solving a series of Gaussian denoising subproblems, which further broadens the application fields.

To improve the practicability of a CNN denoiser, perhaps the most straightforward way is to capture adequate amounts of real noisy-clean training pairs for training so that the real degradation space can be covered. This solution has advantage that there is no need to know the complex degradation process. However, deriving the corresponding clean image of a noisy one is not a trivial task due to the need of careful postprocessing steps, such as spatial alignment and illumination correction. Alternatively, one can simulate the real degradation process to synthesize noisy images for a clean one. However, it is not easy to accurately model the complex degradation process. In particular, the noise model can be different across different cameras. Nevertheless, it is practically preferable to roughly model a certain noise type for training and then use the learned CNN model for type-specific denoising.

Besides the training data, the robust architecture and robust training also play vital roles for the success of a CNN denoiser. For the robust architecture, designing a deep multiscale CNN which involves a coarse-to-fine procedure is a promising direction. Such a network is expected to inherit the merits of multiscale [23]: (i) the noise level decreases at larger scales; (ii) the ubiquitous low-frequency noise can be alleviated by multiscale procedure; and (iii) downsampling the image before denoising can effectively enlarge the receptive field. For the robust training, the effectiveness of the denoiser trained with generative adversarial networks (GAN) for real image denoising still remains uninvestigated. The main idea of GAN-based denoising is to introduce an adversarial loss to improve the perceptual quality of denoised image. A distinctive advantage of GAN is that it can do unsupervised learning, and thus

is expected to be helpful in training denoising CNNs without ground-truth clean images.

**Acknowledgement** This work is partially supported by the National Natural Scientific Foundation of China (NSFC) under Grant No. 61671182 and 61471146, and the HK RGC GRF grant (under no. PolyU 152124/15E).

## References

1. Ahn B, Cho NI (2017) Block-matching convolutional neural network for image denoising. [arXiv:1704.00524](https://arxiv.org/abs/1704.00524)
2. Bae W, Yoo J, Ye JC (2017) Beyond deep residual learning for image restoration: persistent homology-guided manifold simplification. In: The IEEE conference on computer vision and pattern recognition (CVPR) workshops, pp 145–153
3. Bako S, Vogels T, McWilliams B, Meyer M, Novák J, Harvill A, Sen P, DeRose T, Rousselle F (2017) Kernel-predicting convolutional networks for denoising Monte Carlo renderings. *ACM Trans Gr* 36(4):97
4. Barbu A (2009) Training an active random field for real-time image denoising. *IEEE Trans Image Process* 18(11):2451–2462
5. Burger HC, Schuler CJ, Harmeling S (2012) Image denoising: can plain neural networks compete with BM3D? In: IEEE conference on computer vision and pattern recognition, pp 2392–2399
6. Chambolle A, Pock T (2011) A first-order primal-dual algorithm for convex problems with applications to imaging. *J Math Imaging Vis* 40(1):120–145
7. Chan SH, Wang X, Elgandy OA (2017) Plug-and-Play ADMM for image restoration: fixed-point convergence and applications. *IEEE Trans Comput Imaging* 3(1):84–98
8. Chen Y, Pock T (2017) Trainable nonlinear reaction diffusion: a flexible framework for fast and effective image restoration. *IEEE Trans Pattern Anal Mach Intell* 39(6):1256–1272
9. Choi JH, Elgandy O, Chan SH (2017) Integrating disparate sources of experts for robust image denoising. [arXiv:1711.06712](https://arxiv.org/abs/1711.06712)
10. Colom M, Lebrun M, Buades A, Morel JM (2014) A non-parametric approach for the estimation of intensity-frequency dependent noise. In: IEEE international conference on image processing, pp 4261–4265
11. Dabov K, Foi A, Katkovnik V, Egiazarian K (2007) Image denoising by sparse 3-D transform-domain collaborative filtering. *IEEE Trans Image Process* 16(8):2080–2095
12. Godard C, Matzen K, Uyttendaele M (2017) Deep burst denoising. [arXiv:1712.05790](https://arxiv.org/abs/1712.05790)
13. Gu S, Zhang L, Zuo W, Feng X (2014) Weighted nuclear norm minimization with application to image denoising. In: IEEE conference on computer vision and pattern recognition, pp 2862–2869
14. He K, Zhang X, Ren S, Sun J (2015) Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: ICCV, pp 1026–1034
15. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: IEEE conference on computer vision and pattern recognition, pp 770–778
16. Heide F, Steinberger M, Tsai YT, Rouf M, Pajak D, Reddy D, Gallo O, Liu J, Heidrich W, Egiazarian K et al (2014) FlexISP: a flexible camera image processing framework. *ACM Trans Gr* 33(6):231
17. Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning, pp 448–456
18. Jain V, Seung S (2009) Natural image denoising with convolutional networks. In: Advances in neural information processing systems, pp 769–776



19. Kim Y, Jung H, Min D, Sohn K (2017) Deeply aggregated alternating minimization for image restoration. In: IEEE conference on computer vision and pattern recognition, pp 6419–6427
20. Kingma D, Ba J (2015) Adam: a method for stochastic optimization. In: International conference for learning representations
21. Kligvasser I, Shaham TR, Michaeli T (2017) xUnit: learning a spatial activation function for efficient image restoration. [arXiv:1711.06445](https://arxiv.org/abs/1711.06445)
22. Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105
23. Lebrun M, Colom M, Morel JM (2015) The noise clinic: a blind image denoising algorithm. *Image Process On Line* 5:1–54. <http://demo.ipol.im/demo/125/>
24. Lee JS (1981) Refined filtering of image noise using local statistics. *Comput Gr Image Process* 15(4):380–389
25. Lefkimmiatis S (2017) Non-local color image denoising with convolutional neural networks. In: IEEE conference on computer vision and pattern recognition, pp 3587–3596
26. Lefkimmiatis S (2018) Universal denoising networks: a novel CNN-based network architecture for image denoising. In: IEEE conference on computer vision and pattern recognition
27. Levin A, Nadler B (2011) Natural image denoising: optimality and inherent bounds. In: IEEE conference on computer vision and pattern recognition, pp 2833–2840
28. Liu C, Szeliski R, Kang SB, Zitnick CL, Freeman WT (2008) Automatic estimation and removal of noise from a single image. *IEEE Trans Pattern Anal Mach Intell* 30(2):299–314
29. Mildenhall B, Barron JT, Chen J, Sharlet D, Ng R, Carroll R (2017) Burst denoising with kernel prediction networks. [arXiv:1712.02327](https://arxiv.org/abs/1712.02327)
30. Pan J, Hu Z, Su Z, Yang MH (2014) Deblurring text images via L0-regularized intensity and gradient prior. In: IEEE conference on computer vision and pattern recognition, pp 2901–2908
31. Remez T, Litany O, Giryes R, Bronstein AM (2017) Deep class-aware image denoising. In: International conference on sampling theory and applications, pp 138–142
32. Riegler G, Schuler S, Ruther M, Bischof H (2015) Conditioned regression models for non-blind single image super-resolution. In: IEEE international conference on computer vision, pp 522–530
33. Romano Y, Elad M, Milanfar P (2016) The little engine that could: Regularization by denoising (RED). *SIAM J Imaging Sci* (submitted)
34. Roth S, Black MJ (2005) Fields of experts: a framework for learning image priors. In: IEEE computer society conference on computer vision and pattern recognition, vol 2, pp 860–867
35. Roth S, Black MJ (2009) Fields of experts. *Int J Comput Vis* 82(2):205–229
36. Samuel KG, Tappen MF (2009) Learning optimized MAP estimates in continuously-valued MRF models. In: IEEE conference on computer vision and pattern recognition, pp 477–484
37. Santhanam V, Morariu VI, Davis LS (2017) Generalized deep image to image regression. In: IEEE conference on computer vision and pattern recognition, pp 5609–5619
38. Schmidt U, Roth S (2014) Shrinkage fields for effective image restoration. In: IEEE conference on computer vision and pattern recognition, pp 2774–2781
39. Shi W, Caballero J, Huszár F, Totz J, Aitken AP, Bishop R, Rueckert D, Wang Z (2016) Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In: IEEE conference on computer vision and pattern recognition, pp 1874–1883
40. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: International conference for learning representations
41. Sreehari S, Venkatakrisnan S, Wohlberg B, Drummy LF, Simmons JP, Bouman CA (2015) Plug-and-play priors for bright field electron tomography and sparse interpolation. [arXiv:1512.07331](https://arxiv.org/abs/1512.07331)
42. Sun J, Tappen MF (2011) Learning non-local range Markov random field for image restoration. In: IEEE conference on computer vision and pattern recognition, pp 2745–2752
43. Sun J, Tappen MF (2013) Separable Markov random field model and its applications in low level vision. *IEEE Trans Image Process* 22(1):402–407
44. Vedaldi A, Lenc K (2015) MatConvNet: convolutional neural networks for matlab. In: ACM conference on multimedia conference, pp 689–692

45. Vogel C, Pock T (2017) A primal dual network for low-level vision problems. In: German conference on pattern recognition. Springer, pp 189–202
46. Wang T, Qin Z, Zhu M (2017) An ELU network with total variation for image denoising. In: International conference on neural information processing. Springer, pp 227–237
47. Wang T, Sun M, Hu K (2017) Dilated residual network for image denoising. [arXiv:1708.05473](https://arxiv.org/abs/1708.05473)
48. Yang D, Sun J (2018) Bm3d-net: a convolutional neural network for transform-domain collaborative filtering. *IEEE Signal Process Lett* 25(1):55–59
49. Yu F, Koltun V (2016) Multi-scale context aggregation by dilated convolutions. In: International conference on learning representations
50. Zhang K, Zuo W, Chen Y, Meng D, Zhang L (2017) Beyond a Gaussian denoiser: residual learning of deep CNN for image denoising. *IEEE Trans Image Process* 26(7):3142–3155
51. Zhang K, Zuo W, Gu S, Zhang L (2017) Learning deep CNN denoiser prior for image restoration. In: IEEE conference on computer vision and pattern recognition, pp 3929–3938
52. Zhang K, Zuo W, Zhang L (2018) Learning a single convolutional super-resolution network for multiple degradations. In: IEEE conference on computer vision and pattern recognition
53. Zoran D, Weiss Y (2011) From learning models of natural image patches to whole image restoration. In: IEEE international conference on computer vision, pp 479–486

# Chapter 5

## Gaussian Priors for Image Denoising



Julie Delon and Antoine Houdard

**Abstract** This chapter is dedicated to the study of Gaussian priors for patch-based image denoising. In the last 12 years, patch priors have been widely used for image restoration. In a Bayesian framework, such priors on patches can be used for instance to estimate a clean patch from its noisy version, via classical estimators such as the conditional expectation or the maximum a posteriori. As we will recall, in the case of Gaussian white noise, simply assuming Gaussian (or Mixture of Gaussians) priors on patches leads to very simple closed-form expressions for some of these estimators. Nevertheless, the convenience of such models should not prevail over their relevance. For this reason, we also discuss how these models represent patches and what kind of information they encode. The end of the chapter focuses on the different ways in which these models can be learned on real data. This stage is particularly challenging because of the curse of dimensionality. Through these different questions, we compare and connect several denoising methods using this framework.

### 5.1 Introduction

This chapter focuses on patch priors for image denoising. In the last decade, patch-based models (also known as nonlocal models) have created a new paradigm in image processing, leading to very significant improvements both for classical image restoration problems (denoising, *inpainting*, interpolation) or for image synthesis and editing. These models represent images by a set of local neighborhoods or *patches*, and make them collaborate regardless of their spatial position in the image, relying on the observation that most natural images present a remarkable redundancy at a semi-local scale. A patch  $y_i(v)$  is a piece (most of the time a square) of an image  $v$

---

J. Delon (✉)

Université Paris Descartes, Paris, France  
e-mail: julie.delon@parisdescartes.fr

A. Houdard

Télécom ParisTech, Paris, France  
e-mail: antoine.houdard@telecom-paristech.fr

centered at the pixel  $i$ . As pointed out by Mumford and Desolneux [1], patches are “the analogs of the phonemes of speech”.

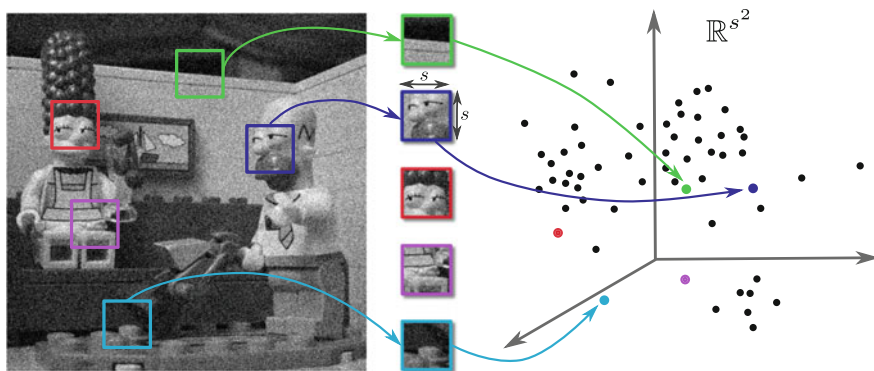
Patch-based models have been the subject of numerous works, especially in the context of image denoising. Assuming that the noise is additive, image denoising amounts to estimate an image  $u$  from its noisy version  $v \in \mathbb{R}^m$  ( $m$  is the image size) such that

$$v = u + \varepsilon, \quad (5.1)$$

with  $\varepsilon$  a noise with known statistics (not necessarily Gaussian). In digital cameras, the two major sources of noise during the acquisition process are the thermal agitation, which produces an almost white and Gaussian noise, and the discrete nature of light, which is behind the photon *shot noise*, modeled as a Poisson variable (for a complete description of the sources of noise in a digital camera, see [2]). Stabilizing the noise variance by a generalized Anscombe transform [3] results in a noise model well approximated by a white Gaussian noise  $\varepsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_m)$ . The vast majority of works on image denoising focus on this simplified model and it is also our assumption in this chapter.

In this framework, patch-based methods usually attempt at rewriting (5.1) into a degradation model that can be expressed for each patch separately. All patches  $\{y_i, i = 1, \dots, m\}$  of size  $p = s \times s$  are first extracted from the image  $v$  and seen as noisy vectors in a high dimensional space, as illustrated in Fig. 5.1 (in the whole chapter, when writing patches as vectors, we assume that the patches are read column-wise). Then the noisy patches are restored sequentially, before reconstructing the whole image. The degradation model on the patches becomes

$$y_i = x_i + \varepsilon_i, \quad i \in \{1, \dots, m\} \quad (5.2)$$



**Fig. 5.1** Image patches can be seen as vectors in a high-dimensional space. Most of the patch-based methods use the patch space of an image which is the set of all the sliding patches of size  $p = s \times s$  extracted from the image

where  $x_i$  is the patch centered at pixel  $i$  in  $u$ ,  $y_i$  the same patch in  $v$ , and  $\varepsilon_i$  the additive noise. In practice, it is almost always assumed that the  $\{\varepsilon_i, i = 1, \dots, m\}$  are independent samples from the Gaussian distribution  $\mathcal{N}(0, \sigma^2 I_p)$ , although this hypothesis is obviously wrong since patches are overlapping. We will briefly discuss this issue in Sect. 5.4, along with the aggregation of the restored patches to reconstruct the whole image.

The first denoising methods relying on patches appear in 2004 [4–7]. Among these methods, one of the most popular remains the Non-Local Means [7], which sees similar patches as independent realizations of the same distribution and averages these repeated structures to reduce noise variance. If numerous approaches have built on the same core ideas since 2004, the recent and most convincing approaches in patch-based denoising rely on a Bayesian reformulation of the denoising problem, using local or global statistical priors for the distribution of each patch [8–13]. Under the white Gaussian noise model (5.2), the conditional distribution of a noisy patch  $y$  knowing its original version  $x$  (we omit the index  $i$  in the following) can be written

$$p(y|x) \propto e^{-\frac{\|x-y\|^2}{2\sigma^2}}. \quad (5.3)$$

The Bayesian model assumes that the original patch  $x$  is a realization of a random vector  $X$  with a probability distribution  $p(x)$  called the *prior distribution*. Therefore, the noisy patch  $y$  is a realization of the random vector

$$Y = X + N, \quad (5.4)$$

with  $N \sim \mathcal{N}(0, \sigma^2 I_p)$ . Under these hypotheses, and assuming that  $N$  and  $X$  are independent, we can compute the *posterior distribution*

$$p(x|y) \propto p(y|x)p(x) \propto e^{-\frac{\|x-y\|^2}{2\sigma^2}} p(x). \quad (5.5)$$

Ideally, in order to reconstruct the (unknown) original patch  $x$  from the degraded version  $y$ , we would like to compute the conditional expectation  $\mathbb{E}[X|Y]$  (*i.e.*, the mean of the posterior distribution), which minimizes the quadratic risk under the previous model. This estimator is also called the minimum mean square error (MMSE) estimator. In practice, computing this conditional expectation is often complex, and it is classical to compute instead the affine function (called linear MMSE) of  $Y$  minimizing the quadratic risk, *i.e.*, the affine estimator  $DY + \alpha$  (with  $D$  a  $p \times p$  real matrix and  $\alpha$  a vector in  $\mathbb{R}^p$ ) minimizing the risk

$$\mathbb{E}[\|DY + \alpha - X\|^2].$$

This affine estimator is called the *Wiener estimator* and will be denoted  $\mathbb{E}_{Wiener}[X|Y]$  in the following. It can be easily shown by deriving the previous risk that (assuming that the following quantities exist),

$$\mathbb{E}_{Wiener}[X|Y] = \mathbb{E}[X] + \Sigma_{X,Y} \Sigma_Y^{-1} (Y - \mathbb{E}[Y]), \quad (5.6)$$

where  $\Sigma_{X,Y} := \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])^t]$  and  $\Sigma_Y := \mathbb{E}[(Y - \mathbb{E}[Y])(Y - \mathbb{E}[Y])^t]$ . This affine estimator only relies on second-order moments of the signal and noise. Under model (5.4) and assuming that  $N$  and  $X$  are independent, the Wiener estimator becomes

$$\mathbb{E}_{Wiener}[X|Y] = \mathbb{E}[X] + \Sigma_X (\Sigma_X + \sigma^2 I_p)^{-1} (Y - \mathbb{E}[Y]), \quad (5.7)$$

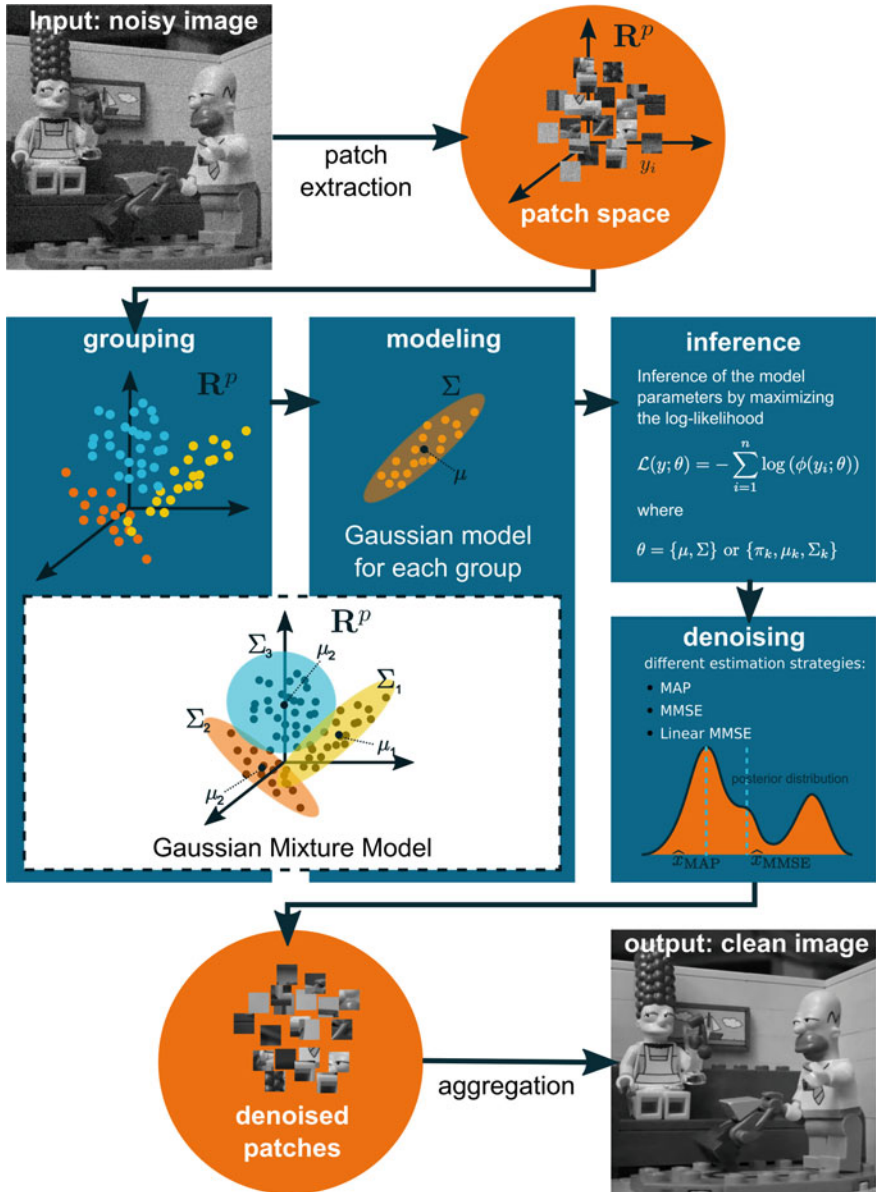
with  $\Sigma_X$  the covariance matrix of the random vector  $X$ .

Another classical solution to reconstruct  $x$  is to compute the maximum (MAP) of the *a posteriori* distribution  $p(y|x)$ , which yields

$$\begin{aligned} \hat{x}(y) &= \arg \max_{x \in \mathbb{R}^p} p(x|y) = \arg \max_{x \in \mathbb{R}^p} p(y|x) p(x) \\ &= \arg \min_{x \in \mathbb{R}^p} -\log p(y|x) - \log p(x) \\ &= \arg \min_{x \in \mathbb{R}^p} \frac{\|x - y\|^2}{2\sigma^2} - \log p(x). \end{aligned}$$

From this point of view, restoring each patch is equivalent to solve a variational problem, with a quadratic fidelity term and a smoothness term derived from the prior.

The most convenient prior for computing the previous estimators is the Gaussian distribution. Indeed, on the one hand, Gaussian priors are well suited to encode patch structures with some kind of contrast invariance, as we will see in Sect. 5.2. On the other hand, under a Gaussian prior, the conditional expectation, Wiener estimator, and MAP coincide, as we will see in Sect. 5.3. For these reasons, these priors are favored in most recent works on patch-based image denoising [8, 12, 14]. A slightly more involved prior used in the literature is the Gaussian Mixture Model (GMM) [9–11, 13, 15]. In this case, computing the conditional expectation remains simply tractable. All these works differ among other things in the way they infer the parameters of the Gaussian or GMM distributions. These distributions live in  $\mathbb{R}^p$  and estimation in such high-dimensional spaces is complex. We will see in Sect. 5.5 the different possibilities to infer these parameters and how some of these works tackle the curse of dimensionality. Figure 5.2 illustrates the main steps common to all these patch-based denoising methods, and each of these steps is described in the following sections.



**Fig. 5.2** The whole process of patch-based image denoising with Gaussian prior models. First, patches are extracted from the noisy image. Next, these noisy patches are grouped and modeled with local Gaussian models or Gaussian mixture models, whose parameters are inferred by maximum likelihood (Sect. 5.5). Each patch is then denoised with an estimator derived from the model (Sect. 5.3). Finally, the clean patches are aggregated to recover the denoised image (Sect. 5.4)

## 5.2 What Is Encoded in Gaussian and GMM Priors?

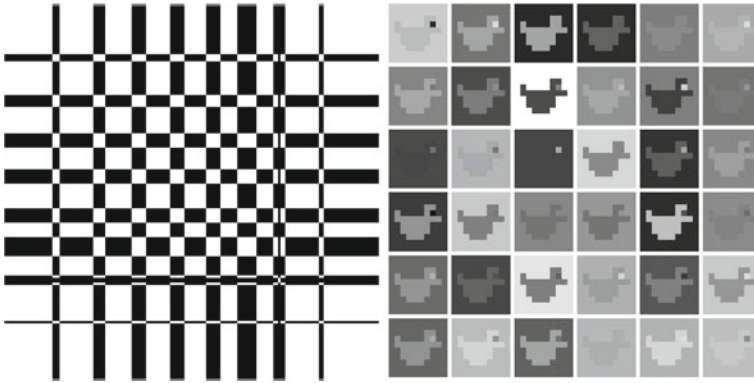
Before going into the details of estimation under Gaussian priors, we provide in this section a few insights on the actual structures they encode. Assume a Gaussian model  $\mathcal{N}(\mu, \Sigma)$  for  $p = s \times s$  patches ( $\mu \in \mathbb{R}^p$  and  $\Sigma \in \mathcal{M}_p(\mathbb{R})$ ). The diagonal coefficients of the covariance matrix  $\Sigma$  represent the variance of each pixel in the patch, while the non-diagonal coefficients represent the covariances between pixels. A positive covariance coefficient means that the two pixels tend to be either both greater or smaller than their means, while a negative coefficient implies that they tend to be on opposite sides of their means. Since we are dealing with Gaussian vectors, a null covariance coefficient means that the two pixels are independent. Clearly, if  $\Sigma$  is purely diagonal, patches drawn from the model  $\mathcal{N}(\mu, \Sigma)$  will only be noisy versions of the mean patch  $\mu$ . In this case, the only structure information is contained in  $\mu$ . More interesting models contain geometric information directly in the covariance matrix  $\Sigma$ .

To illustrate this point, we propose to create models encoding different patch structures. For instance, in order to model a vertical edge, we define a Gaussian distribution with constant mean  $\mu = (0.5, \dots, 0.5)$  and a covariance matrix with coefficient 1 in the second and third quarter of  $\Sigma$ , and coefficient 0 in the first and fourth quarters of  $\Sigma$  (see Fig. 5.3). In this simplistic example, the matrix  $\Sigma$  has rank two, with (non trivial) eigenvectors  $(1, \dots, 1, 0, \dots, 0)$  and  $(0, \dots, 0, 1, \dots, 1)$ , so all the patches drawn from this distribution can be written  $0.5 + (\alpha, \dots, \alpha, \beta, \dots, \beta)$  with  $\alpha \sim \mathcal{N}(0, 1)$  and  $\beta \sim \mathcal{N}(0, 1)$ . These patches all contain a vertical edge in their middle, with gray levels  $\alpha$  and  $\beta$  on both sides of the edge. In this example, we see that the model encodes a structure and authorizes different contrasts on both sides of the structure. With the same mechanic, we can create a covariance matrix encoding any desired shape, see for instance Fig. 5.4. Again, the samples from the

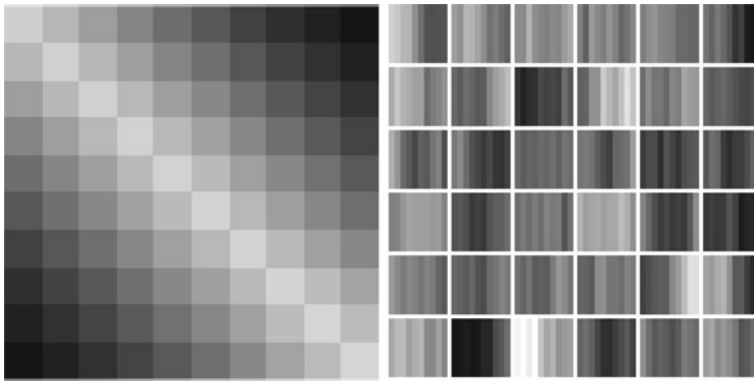


**Fig. 5.3** Left: a covariance matrix  $\Sigma$  with 1 (white) on the second and third quarters, and 0 (black) on the first and fourth quarters. Right: patches drawn from the Gaussian distribution  $\mathcal{N}(\mu, \Sigma)$  with  $\mu$  a constant patch equal to 0.5





**Fig. 5.4** Left: a covariance matrix  $\Sigma$  composed of 1 (white) and 0 (black). Right: patches drawn from the Gaussian distribution  $\mathcal{N}(\mu, \Sigma)$  with  $\mu$  a constant patch equal to 0.5



**Fig. 5.5** Left: a covariance matrix  $\Sigma$  learned as the sample covariance matrix of a set of vertical edges at different spatial positions, and with also different choices of gray levels on both sides of the edge. Right: patches drawn from the corresponding Gaussian distribution  $\mathcal{N}(\mu, \Sigma)$  with  $\mu$  a constant patch equal to 0.5

corresponding distribution exhibit all possible gray levels in the different regions defined by the covariance matrix, even if all these gray levels are not all equally likely.

Now, although these models authorize contrast changes or contrast inversions, they are not well suited to encode geometric invariances on patches. For instance, if we try to learn a model encoding different vertical edges with invariance to translation, we end up with an average model encoding a vertical gradient image (see Fig. 5.5).

### 5.3 How to Derive Estimators Under Gaussian and GMM Priors

Now that we have seen more precisely what could be contained in Gaussian priors, we will now see more precisely how they can be used to derive estimators under the Bayesian model described in the introduction.

In the whole section, we assume that we work with the model (5.4)

$$Y = X + N,$$

with  $N \sim \mathcal{N}(0, \sigma^2 I_p)$  independent from  $X$ . We wish to estimate  $X$  knowing  $Y$ .

We first recall some classical results on the conditioning of Gaussian vectors, and on the links between the conditional expectation, Wiener estimator and MAP for Gaussian and GMM priors. These different estimators will serve in the rest of the chapter as denoising strategies for image patches.

#### 5.3.1 Estimation with Gaussian Priors

We first assume that  $X$  follows a Gaussian distribution  $\mathcal{N}(\mu_X, \Sigma_X)$  and that the noise  $N$  is independent from  $X$ . The classical properties of Gaussian vectors make it possible to show that in this case the estimator  $\mathbb{E}[X|Y]$  is an affine function of  $Y$  (thus equivalent in this case to the Wiener estimator). Indeed, recall that if  $(T, V)$  is a Gaussian vector, then the conditional expectation  $\mathbb{E}[T|V]$  is the affine function of  $V$

$$\mathbb{E}[T|V] = \mathbb{E}[T] + \Sigma_{T,V} \Sigma_V^{-1} (V - \mathbb{E}[V]), \quad (5.8)$$

where  $\Sigma_V$  is the covariance matrix of  $V$  and  $\Sigma_{T,V} = \mathbb{E}[(T - \mathbb{E}[T])(V - \mathbb{E}[V])']$  (if  $\Sigma_V$  is not full rank, the result is still true by taking the Moore–Penrose pseudo-inverse of  $\Sigma_V$ ).

Now, if  $X$  and  $N$  are independent Gaussian random vectors, the concatenated vector  $(X, Y) = (X, X + N)$  is also Gaussian. We directly deduce the following result.

**Proposition 1** *Assume that  $X$  and  $Y$  follow the model (5.4), with  $X \sim \mathcal{N}(\mu_X, \Sigma_X)$  and  $N \sim \mathcal{N}(0, \sigma^2 I_p)$  independent, then the conditional expectation and Wiener estimator of  $X$  knowing  $Y$  coincide and can be written*

$$\mathbb{E}[X|Y] = \mathbb{E}_{Wiener}[X|Y] = \mu_X + \Sigma_X (\Sigma_X + \sigma^2 I_p)^{-1} (Y - \mu_X).$$

*Proof* On the one hand, since  $(X, Y)$  is a Gaussian vector, the conditional expectation  $\mathbb{E}[X|Y]$  can be written

$$\begin{aligned}
\mathbb{E}[X|Y] &= \mathbb{E}[X] + \Sigma_{X,Y} \Sigma_Y^{-1} (Y - \mathbb{E}[Y]) \\
&= \mathbb{E}[X] + \mathbb{E}[(X - \mathbb{E}[X])(X + N - \mathbb{E}[X + N])^t] (\Sigma_X + \sigma^2 I_p)^{-1} (Y - \mathbb{E}[Y]). \\
&= \mathbb{E}[X] + \Sigma_X (\Sigma_X + \sigma^2 I_p)^{-1} (Y - \mathbb{E}[Y]) \\
&= \mu_X + \Sigma_X (\Sigma_X + \sigma^2 I_p)^{-1} (Y - \mu_X).
\end{aligned}$$

Under the same hypothesis, if we try to maximize the *a posteriori* probability on the patch  $X$ , we obtain

$$\begin{aligned}
\arg \max_X \log \mathbb{P}[X|Y] &= \arg \max_X (\log \mathbb{P}[Y|X] + \log \mathbb{P}[X]) \\
&= \arg \min_X \left( (X - Y)^t (X - Y) / \sigma^2 + (X - \mathbb{E}[X])^t \Sigma_X^{-1} (X - \mathbb{E}[X]) \right).
\end{aligned}$$

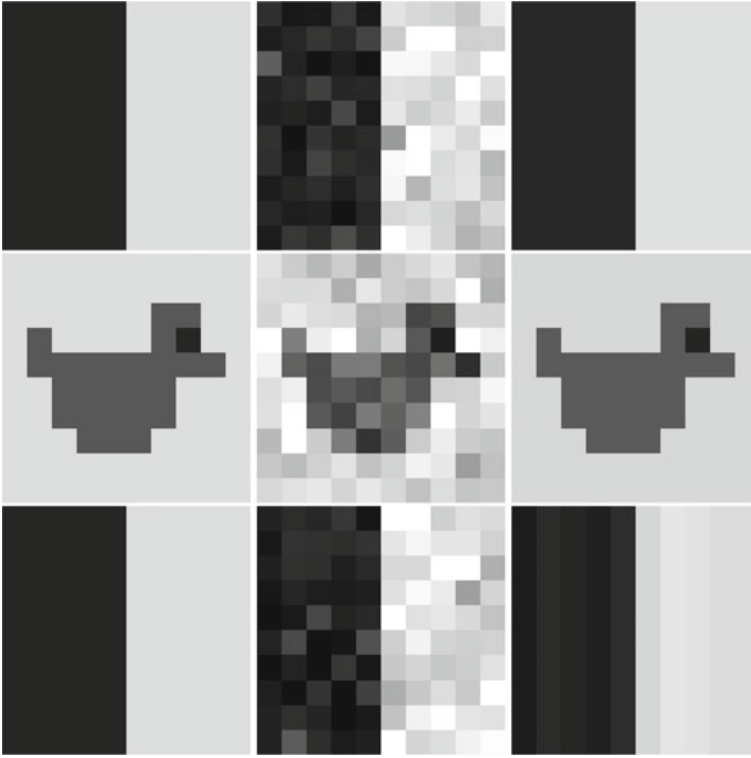
We check easily that the solution of this minimization problem is also given by

$$\psi(Y) = \mu_X + \Sigma_X (\Sigma_X + \sigma^2 I_p)^{-1} (Y - \mu_X).$$

Said otherwise, for a Gaussian prior, the MMSE, linear MMSE and MAP all coincide and all these estimators only require linear operations. This property makes Gaussian priors particularly convenient in practice and explains their success in the restoration literature.

We can illustrate the interest of this estimator on the Gaussian model  $\mathcal{N}(\mu_X, \Sigma_X)$  presented in Fig. 5.3 and representing a vertical edge. If  $X$  is an (unknown) realization of this model and  $Y = X + N$  with  $N \sim \mathcal{N}(0, \sigma^2 I_p)$  independent from  $X$ , then  $\mathbb{E}[X|Y]$  will also be a patch  $(\alpha, \dots, \alpha, \beta, \dots, \beta)$  with  $\alpha = 0.5 + \frac{1}{p/2 + \sigma^2} \sum_{k=1}^{p/2} (Y_k - 0.5)$  and  $\beta = 0.5 + \frac{1}{p/2 + \sigma^2} \sum_{k=p/2+1}^p (Y_k - 0.5)$  (assuming  $p$  is even for the sake of simplicity). Said otherwise, the denoised patch  $\mathbb{E}[X|Y]$  represents the same vertical edge as  $X$  and its values  $\alpha$  and  $\beta$  on both sides of the edge are (if  $\sigma^2 \ll p/2$ ) the averages of  $Y$  on these two half patches.

Figure 5.6 represents three denoising experiments with the previous estimator. On the first line, a vertical edge is denoised with the Gaussian model of Fig. 5.3. On the second line, a “duck” patch is denoised with the Gaussian model of Fig. 5.4. In both cases, using the conditional expectation works extremely well because the Gaussian model used in the estimator fits perfectly the image to be denoised. On the third line, the noisy edge is denoised with the Gaussian model of Fig. 5.5. In this case, the denoised patch is constant on each column (since the model is learned from a set of translated vertical edges). Although the model imposes a strong correlation between columns of the first half of the patch on the one hand, and between columns of the second half of the patch on the other hand, this is not enough to restore the patch perfectly.



**Fig. 5.6** For each line, from left to right, clean patch, noisy patch ( $\sigma = 10\%$ ), denoised patch with the Wiener estimator. First line, the edge Gaussian model of Fig. 5.3 is used to denoise (PSNR = 37.17). Second line, the duck Gaussian model of Fig. 5.4 is used to denoise (PSNR = 34.29). Third line, the gradient model of Fig. 5.5 is used to denoise (PSNR = 29.68). In this last case, the image to be denoised is not well represented by the model and the result is less convincing

### 5.3.2 Estimation with Gaussian Mixture Models

The case of Gaussian Mixture Models is a bit more involved but remains globally simple. Assume that  $X$  follows a Gaussian Mixture Model

$$X \sim \sum_{k=1}^K \pi_k \mathcal{N}(\mu_k, \Sigma_k), \quad (5.9)$$

with  $\sum_{k=1}^K \pi_k = 1$ . There exists a latent random variable  $Z$  on  $\{1, \dots, K\}$  such that  $\mathbb{P}[Z = k] = \pi_k$  and such that  $X|Z = k \sim \mathcal{N}(\mu_k, \Sigma_k)$ . In the following, we note  $\psi_k(y)$  the Wiener estimator for the  $k$ th Gaussian, *i.e.*,

$$\psi_k(y) = \mu_k + \Sigma_k(\Sigma_k + \sigma^2 I_p)^{-1}(y - \mu_k).$$

Under this model, we have the following proposition.

**Proposition 2** Assume that  $X$  and  $Y$  follow the model (5.4), with  $X \sim \sum_{k=1}^K \pi_k \mathcal{N}(\mu_k, \Sigma_k)$  and  $N \sim \mathcal{N}(0, \sigma^2 I_p)$  independent, then the conditional expectation of  $X$  knowing  $Y$  can be written as

$$\mathbb{E}[X|Y] = \sum_{k=1}^K \psi_k(Y) \mathbb{P}[Z = k|Y]. \quad (5.10)$$

*Proof* To compute the conditional expectation, we can start by noting that if  $Z = k$ ,  $(X, Y)$  is a Gaussian vector and the results of the previous section apply. We can now compute the conditional expectation

$$\mathbb{E}[X | Y, Z] = \psi_Z(Y) = \sum_{k=1}^K \psi_k(Y) \mathbf{1}_{Z=k}.$$

It follows that

$$\begin{aligned} \mathbb{E}[X|Y] &= \mathbb{E}[\mathbb{E}[X | Y, Z] | Y] \quad \text{because } \sigma(Y) \subset \sigma(Y, Z) \\ &= \mathbb{E}[\psi_Z(Y) | Y] = \sum_{k=1}^K \mathbb{E}[\psi_k(Y) \mathbf{1}_{Z=k} | Y] \\ &= \sum_{k=1}^K \psi_k(Y) \mathbb{E}[\mathbf{1}_{Z=k} | Y] \quad \text{because } \psi_k(Y) \text{ is } \sigma(Y)\text{-measurable.} \end{aligned}$$

We deduce that

$$\mathbb{E}[X|Y] = \sum_{k=1}^K \psi_k(Y) \mathbb{E}[\mathbf{1}_{Z=k} | Y] = \sum_{k=1}^K \psi_k(Y) \mathbb{P}[Z = k | Y].$$

The conditional expectation  $\mathbb{E}[X|Y]$  can be seen as a linear combination of affine functions of  $Y$ , with weight  $\mathbb{P}[Z = k|Y]$  representing the probability that the patch belongs to the class  $k$ . However, the weights  $\mathbb{P}[Z = k | Y]$  are not linear functions of  $Y$ .

The expression of the Wiener estimator  $\mathbb{E}_{Wiener}[X|Y]$  can be deduced directly from Eq. (5.7), by replacing  $\mathbb{E}[X]$  by  $\sum_{k=1}^K \pi_k \mu_k$  and  $\Sigma_X$  by the complete covariance of the GMM.

Finally, computing the MAP  $\arg \max_X \log \mathbb{P}[X|Y]$  under a GMM prior on  $X$  is much less convenient and does not lead to a closed-form solution. Indeed, it boils down to compute the maximum of the posterior distribution, which is another Gaussian Mixture distribution.

In other words, the linear MMSE, MMSE, and MAP do not coincide for Gaussian Mixture priors. In practice, the conditional expectation is favored since it is much simpler to compute than the MAP.

### 5.3.3 Other Estimation Strategies

Estimation under Gaussian or GMM models has several links with other estimation strategies found in the literature. For a noisy patch  $y$ , and a Gaussian model  $\mathcal{N}(\mu, \Sigma)$ , we have seen that the conditional expectation strategy consists in computing the denoised patch

$$\hat{x}(y) = \mu + \Sigma(\Sigma + \sigma^2 I_p)^{-1}(y - \mu).$$

Now, if we consider the eigendecomposition  $\Sigma = Q\Delta Q^t$  with  $\Delta = \text{diag}(\lambda_1, \dots, \lambda_p)$ , this can be rewritten

$$\hat{x}(y) = \mu + Q \text{diag} \left( \frac{\lambda_1}{\lambda_1 + \sigma^2}, \dots, \frac{\lambda_p}{\lambda_p + \sigma^2} \right) Q^t (y - \mu). \quad (5.11)$$

More generally, denoting  $Q_1, \dots, Q_p$  the columns of  $Q$  representing the eigenvectors, we can write

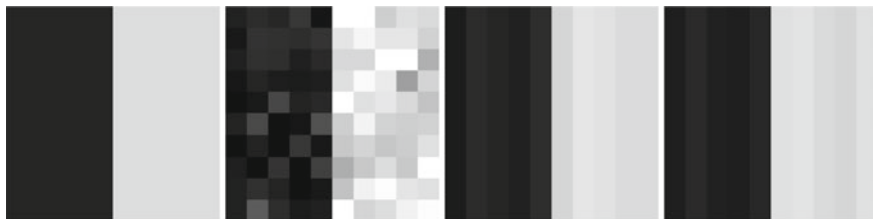
$$\hat{x}(y) = \mu + \sum_{k=1}^p \eta_k (Q_k^t (y - \mu)) Q_k, \quad (5.12)$$

with  $\eta_k(z) = \frac{\lambda_k}{\lambda_k + \sigma^2} z$ . Although the previous Wiener estimator is used in numerous recent patch-based denoising methods [8, 11, 15], other choices are obviously possible for  $\eta_k$ , such as hard or soft thresholding [16], or all estimators classically used in diagonal estimation.

Writing  $\tilde{x} = Q^t(x - \mu)$ , we can see that the conditional expectation  $\hat{x}(y)$  is also solution of the optimization problem

$$\underset{\tilde{x}}{\text{argmin}} \|Q\tilde{x} - (y - \mu)\|^2 + \sigma^2 \tilde{x}^t \Delta^{-1} \tilde{x} = \underset{\tilde{x}}{\text{argmin}} \|Q\tilde{x} - (y - \mu)\|^2 + \sigma^2 \sum_{k=1}^p \frac{\tilde{x}_k^2}{\lambda_k}.$$

This permits to see the link between the previous approach and the dictionary-based approaches, the dictionary here being given by  $Q$  and the second term corresponding to a regularization of the solution  $\tilde{x}$ . Figure 5.7 represents the denoising of a noisy patch with the same Gaussian model and two different denoising strategies: the conditional expectation (Wiener) and hard thresholding at  $2.7\sigma$  (as recommended in [16]).



**Fig. 5.7** Clean patch, noisy patch (10% noise), denoised patch with gradient model (from Fig. 5.5) and Wiener estimator (PSNR = 29.68 dB), and denoised patch with gradient model and hard thresholding (PSNR = 31.12 dB,  $\text{th} = 2.7\sigma$ )

## 5.4 From Patches to Images: Aggregation Procedures

In the previous sections, we have seen how to derive Bayesian estimators to perform denoising on each patch separately. In this framework, each observed patch  $y_i$  from a noisy image  $v$  is denoised into  $\hat{x}_i$ , which is an estimate of the unknown patch  $x_i$ . Each pixel of the image  $v$  is contained in  $p$  patches, which provide  $p$  denoised versions for this pixel. Most aggregation procedures consist in defining a reprojection function  $\psi : \mathbb{R}^{m \times p} \rightarrow \mathbb{R}^m$  which reconstructs an image from the set of its denoised patches. Observe that since denoised patches usually do not coincide on their overlap, this operation is not invertible. Moreover, since the noise on overlapping patches is not independent, the  $p$  denoised versions of the pixel carry this dependence under the form of low-frequency noise. In the literature, we find three main strategies for this reprojection step:

- **Central pixel reprojection.** The idea is to keep only the central pixel of each denoised patch.
- **Uniform reprojection.** All the estimators coming from the different patches containing the pixel are averaged with uniform weights. This strategy is the most commonly used in practice, and this is the one we use in this chapter for the sake of simplicity.
- **Weighted reprojection.** All the estimators coming from the different patches containing the pixel are averaged with weights representing the precision of the corresponding estimator. For some details, see [14, 17, 18].

A more involved strategy is explored in [9]. The authors propose to reconstruct the denoised image  $u$  as the solution of

$$\underset{u}{\operatorname{argmin}} \frac{\lambda}{2} \|u - v\|_2^2 - \sum_j \log p(x_j),$$

where the  $\{x_j\}$  are the patches extracted from the unknown image  $u$  and  $p$  is a GMM prior on the image patches. This formulation includes both the denoising and aggregation step into a single variational problem.

## 5.5 Inference of Gaussian and GMM Priors

Gaussian models and GMMs appear to be well suited for patch-based denoising. However, the quality of the restoration strongly depends on the relevance of the model. Unfortunately, in real denoising problems, the perfect model is never known and the most challenging step is to find a good prior for each patch. In the literature, we find essentially two strategies to learn these models. The first one consists in learning the model on some external set of patches that represent the diversity of natural images [9]. The second one consists in learning the model directly on the noisy patches [8, 13, 15]. In this section, we discuss different approaches adopting the second strategy. Before going further, we recall some basics about statistical inference.

Given a set of patches  $\{y_1, \dots, y_n\} \in \mathbb{R}^p$  extracted from an image, we consider them as independent realizations of a random variable  $Y$  with density  $\phi$  depending on some parameters  $\theta$ . The parameters  $\theta$  of the model are inferred by maximizing the likelihood of the data *w.r.t.*  $\theta$ , where the likelihood is defined as

$$\ell(y; \theta) = \prod_{i=1}^n \phi(y_i; \theta). \quad (5.13)$$

Maximizing the likelihood is equivalent to minimize the negative log-likelihood

$$\mathcal{L}(y; \theta) = -\log(\ell(y; \theta)) = -\sum_{i=1}^n \log(\phi(y_i; \theta)), \quad (5.14)$$

which is usually more convenient for computation.

In the context of denoising, we put a prior model on the random vector  $X$  representing the clean patches. When  $X$  follows a Gaussian model of parameters  $(\mu_X, \Sigma_X)$ , resp. a Gaussian mixture model of parameters  $\{\pi_k, \mu_k, \Sigma_k\}_{k=1\dots K}$ , then  $Y = X + N$  also follows a Gaussian model of parameters  $\{\mu_X, \Sigma_X + \sigma^2 I\}_k$ , resp. a GMM of parameters  $(\pi_k, \mu_k, \Sigma_k + \sigma^2 I)$ . Since  $\Sigma_X$  (resp.  $\Sigma_k$ ) is positive semi-definite and  $\sigma > 0$ ,  $\Sigma_X + \sigma^2 I$  (resp.  $\Sigma_k + \sigma^2 I$ ) is always positive definite. Thus, the random vector  $Y$  always has a probability density function  $\phi$  and the likelihood is always defined.

### 5.5.1 Gaussian Models

In the case of a Gaussian prior  $X \sim \mathcal{N}(\mu_X, \Sigma_X)$  on the clean patches, the set of parameters on the noisy patches is given by  $\theta = \{\mu_Y, \Sigma_Y\}$  where  $\Sigma_Y = \Sigma_X + \sigma^2 I$  and  $\mu_X = \mu_Y$ . The negative log-likelihood for a set of noisy data  $\{y_1, \dots, y_n\}$  becomes



$$\mathcal{L}(y; \theta) = \frac{1}{2} \sum_{i=1}^n (y_i - \mu_Y)^T \Sigma_Y^{-1} (y_i - \mu_Y). \quad (5.15)$$

The computation of the maximum likelihood estimators (MLE) of the parameters, *i.e.*  $\operatorname{argmin}_{\theta} \mathcal{L}(x; \theta)$ , for  $\mu_Y$  and  $\Sigma_Y$  yields the sample mean

$$\widehat{\mu}_Y(n) = \frac{1}{n} \sum_{i=1}^n y_i, \quad (5.16)$$

and the sample covariance matrix

$$\widehat{\Sigma}_Y(n) = \frac{1}{n} \sum_{i=1}^n (y_i - \widehat{\mu}_Y)^T (y_i - \widehat{\mu}_Y). \quad (5.17)$$

These estimators depend on the number  $n$  of samples and from the strong law of large numbers

$$\widehat{\mu}_Y(n) \xrightarrow[n \rightarrow \infty]{a.s.} \mu_Y \quad \text{and} \quad \widehat{\Sigma}_Y(n) \xrightarrow[n \rightarrow \infty]{a.s.} \Sigma_Y. \quad (5.18)$$

This gives us an estimator  $\widehat{\Sigma}_X := \widehat{\Sigma}_Y - \sigma^2 I$  for  $\Sigma_X$  satisfying

$$\widehat{\Sigma}_X(n) \xrightarrow[n \rightarrow \infty]{a.s.} \Sigma_X. \quad (5.19)$$

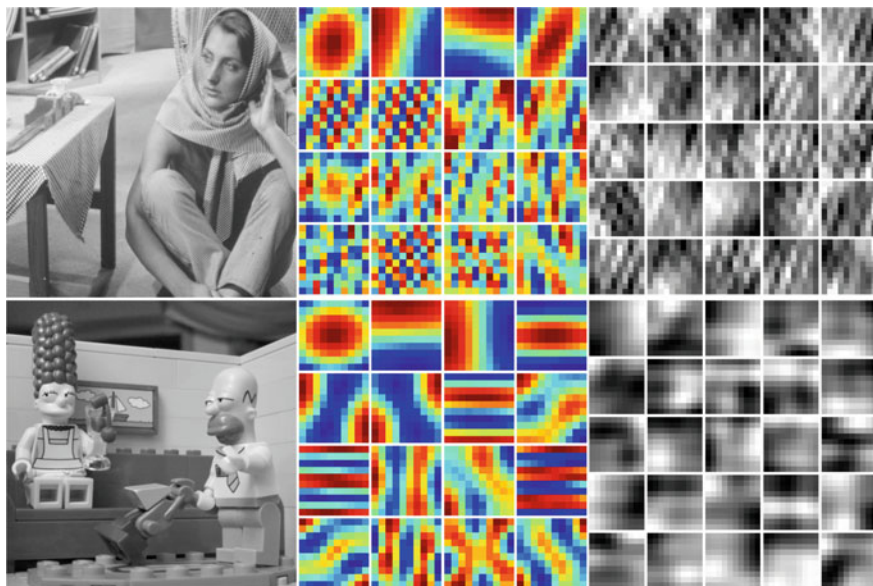
In summary, for a given set of noisy patches  $\{y_1, \dots, y_n\}$ , we can easily compute the MLE of the parameters  $(\mu_X, \Sigma_X)$  for the Gaussian model on the underlying clean patches. Now, since we showed in Sect. 5.2 that Gaussian models are representing really precise structures, the most challenging part is to choose the set of noisy patches from which the model can be derived.

## 5.5.2 How to Group Patches to Infer Gaussian Priors?

In this section, we discuss how patches can be grouped in order to learn the previous Gaussian models directly from a noisy image.

### 5.5.2.1 Global Gaussian Prior

The first really basic idea is to model the set of all image patches with a unique Gaussian prior. In this case, we are modeling the whole “patch space” by a unique Gaussian model of mean  $\widehat{\mu}_X$  and covariance  $\widehat{\Sigma}_X$ . This model poorly represents the complexity of the patch space but still encodes some proper image information.



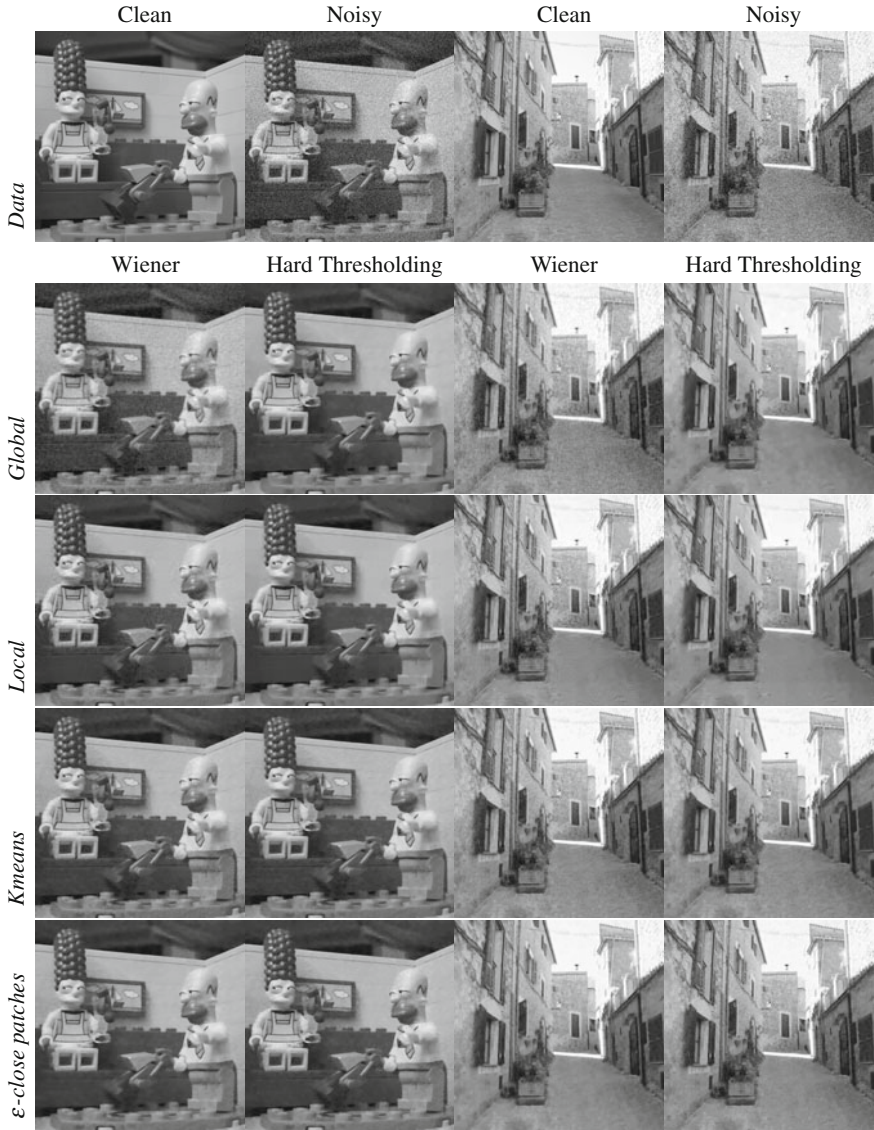
**Fig. 5.8** Visualization of the first 16 eigenvectors of the sample covariance matrix of the whole patch space for two different images. Left: original images. Middle: the 16 first eigenvectors. Right: patches generated with the low rank covariance matrix created from these eigenvectors

This modeling is adopted in [16] to perform a basic denoising by performing the eigendecomposition  $\Sigma_X = Q\Delta Q^t$  and denoising the patches with an estimator of the form (5.12). Figure 5.8 illustrates the fact that the eigenvectors of the covariance matrix learned on the whole patch space encode some proper information about the image.

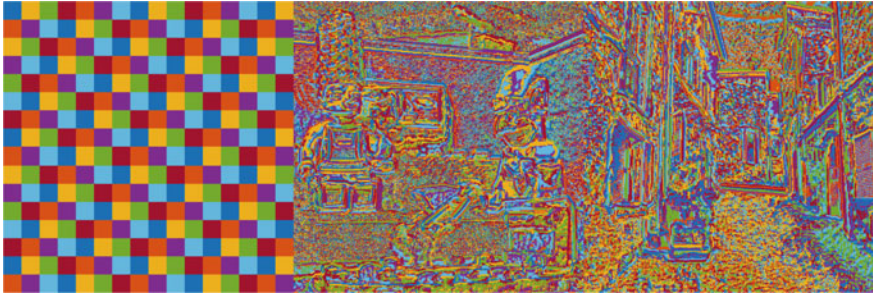
In this case, since the Gaussian model is very broad, we do not expect the Wiener estimator to yield good results. But since the eigenbasis seems to encode some proper information about the image patches, the hard-thresholding strategy manages surprisingly good denoising. The second line of Fig. 5.9 shows the denoising result for this global grouping with the two denoising strategies and shows that in this case, the hard-thresholding strategy is better than the Wiener one.

### 5.5.2.2 Spatially Local Gaussian Priors

To derive more precise prior models, it is necessary to group “similar” patches and to restrict the inference to each of these groups. A first possibility is to group patches based on their spatial proximity in the image. This makes sense in homogeneous regions, but the risk is high to group patches representing really different structures. The third line of Fig. 5.9 shows that the result of this strategy is not really better, PSNR-wise, than the result of the global strategy. However, the Wiener strategy for



**Fig. 5.9** First line: two images and their noisy versions ( $\sigma = 30$ ). Columns correspond to denoising strategies (Wiener or Hard thresholding). Lines correspond to grouping strategies: 1. one Gaussian model for all patches (PSNR, from left to right: 29.18, 31.22, 25.94, 26.85 dB), 2.  $K = 256$  local Gaussian models in the image space, see Fig. 5.10 (PSNR, from left to right: 29.14, 30.72, 26.28, 26.88dB), 3.  $K = 256$  local Gaussian models from a k-means clustering, see Fig. 5.10 (PSNR: 31.30, 31.09, 26.92, 27.08dB), 4. local Gaussian models for group of  $\epsilon$ -close patches (PSNR: 30.45, 29.65, 26.72, 25.95 dB)



**Fig. 5.10** Left: the local grouping used in the local strategy. Middle and right: the grouping used in the K-means strategy for the two images Simpson and Alley

this local approach seems nicer than in the global approach, while the result of the hard-thresholding strategy does not really change.

### 5.5.2.3 Local Gaussian Priors in the Space of Patches

In order to learn more precise models, patches can be clustered directly in the patch space and a Gaussian model can be inferred for each cluster. All patches from the cluster can then be denoised using this model. This clustering implies to use an appropriate similarity measure between patches. The fourth line of Fig. 5.9 shows such a denoising experiment with a K-means clustering relying on the Euclidean distance, with  $K = 256$  clusters (Fig. 5.10 shows the corresponding clustering). This usually yields a better denoising than the global and the local grouping strategies.

This way of grouping patches in the patch space together with a Wiener filtering is also one of the main ideas behind the two steps of the NL-Bayes algorithm [8]. In this algorithm, each patch  $y_i$  is associated with the group of all its  $\varepsilon$ -close patches for the Euclidean norm. A Gaussian model is inferred from this group and the whole group is denoised using this model. The final estimator for each patch is the average of all its denoised versions. The NL-Bayes algorithm uses this strategy twice: in the first step, distances are computed directly between noisy patches in  $\mathbb{R}^P$ ; in the second step, distances between patches are computed between the versions which have been denoised during the first step. Grouping  $\varepsilon$ -close patches presents the advantage of putting together patches representing the same structures. However, a straightforward one-step implementation (fifth row of Fig. 5.9) of this idea shows that it does not work as well as expected in practice. Two major issues arise in this context:

- The high dimensionality of the patch space makes the estimation of the covariance matrices difficult;
- The use of the Euclidean distance for grouping does not allow similar patches with different contrast to be in the same group.

The first issue, discussed in Sect. 5.5.4, is crucial and related to the curse of dimensionality. Unfortunately, it is hardly taken into account in the image denoising literature.

To tackle the second issue, other norms were investigated in the literature [19]. Another idea is to use the Gaussian models previously learned for recalculating new clusters. Indeed, each covariance matrix of the different Gaussian models provides a semi-norm that can be used to recompute the  $\varepsilon$ -nearest patches of each group.

### 5.5.3 Inference for Gaussian Mixture Models

The inference in the case of a mixture model is slightly more challenging since a direct maximization of the likelihood is not possible. The negative log-likelihood of the noisy data  $\{y_1, \dots, y_n\}$  is given by

$$\mathcal{L}(y; \theta) = \sum_{i=1}^n \log \left( \sum_{k=1}^K \pi_k \phi(y_i; \theta_k) \right) \quad (5.20)$$

and the minimization of this function w.r.t  $\theta$  is a complex problem. However, if we know to which group each sample  $x_i$  belongs, the log-likelihood becomes

$$\mathcal{L}(y, z; \theta) = \sum_{i=1}^n \sum_{k=1}^K z_{ik} \log (\pi_k \phi(y_i; \theta_k)) \quad (5.21)$$

with  $z_{ik} = 1$  if  $y_i$  belongs to the group  $k$  and 0 otherwise.  $\mathcal{L}(y, z; \theta)$  is the log-likelihood of the data completed with the latent random variable  $Z$  that determines the group from which the observations come from, that is  $Y_i | (Z_i = k) \sim \mathcal{N}(\mu_k, \Sigma_k)$  and  $p(Z_i = k) = \pi_k$ .

The EM algorithm consists in iterating two steps : the expectation (E) step that calculates the expected value of (5.21) with respect to the conditional distribution of  $Z$  given  $Y$  for the current value of the parameters  $\theta$ , and the maximization (M) step that updates the parameters by minimizing the expectation of the complete log-likelihood from the E-step:

$$\mathbf{E}(\mathcal{L}(y, z; \theta)) = \sum_{i=1}^n \sum_{k=1}^K \mathbf{E}(z_{ik} | x_i, \theta) \log (\pi_k \phi(y_i; \theta_k)) \quad (5.22)$$

which leads to tractable expressions for the MLE of the parameters. It can be shown (see for example [20]) that this algorithm converges to a local minimum of the log-likelihood (5.20).

In the precise case of a Gaussian mixture model, the two steps of the algorithm become

- **E-step**, computation of  $t_{ik} := \mathbf{E}(z_{ik}|y_i, \theta)$

$$t_{ik} = \frac{\pi_k \phi(y_i; \theta_k)}{\sum_{l=1}^K \pi_l \phi(y_i; \theta_l)} \quad (5.23)$$

- **M-step**, update of the parameters

$$\widehat{\pi}_k = \frac{1}{n} \sum_{i=1}^n t_{ik}, \quad (5.24)$$

$$\widehat{\mu}_k = \frac{\sum_{i=1}^n t_{ik} y_i}{\sum_{i=1}^n t_{ik}}, \quad (5.25)$$

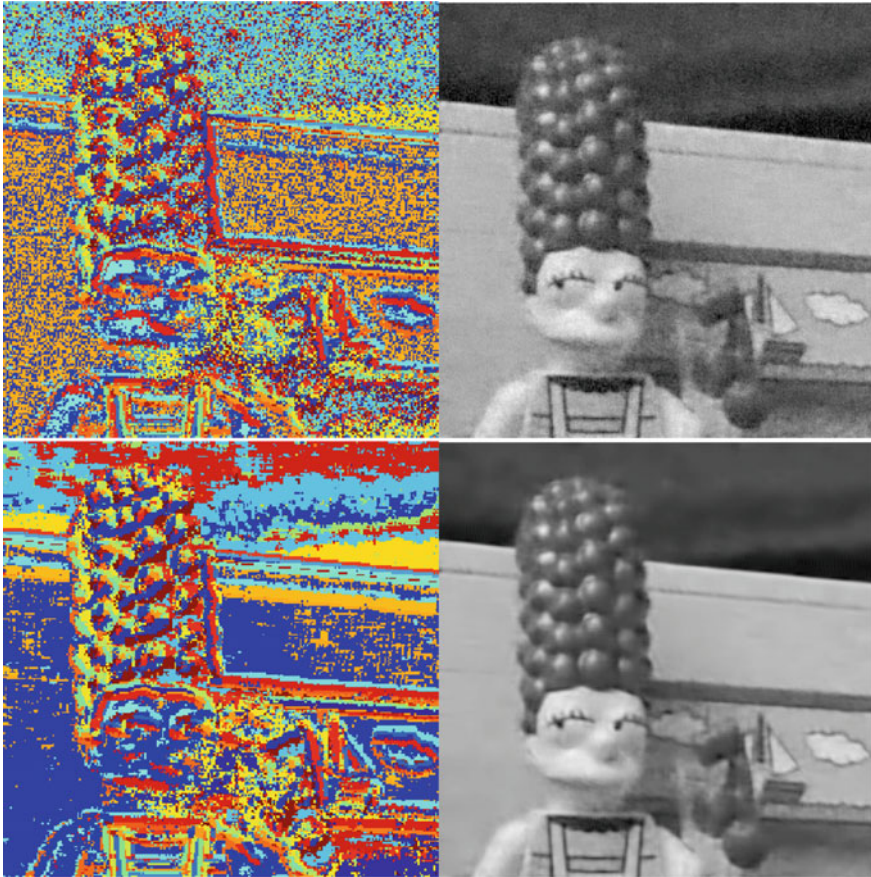
$$\widehat{\Sigma}_k = \frac{\sum_{i=1}^n t_{ik} (y_i - \mu_k)(y_i - \mu_k)^T}{\sum_{i=1}^n t_{ik}}. \quad (5.26)$$

Observe that if we impose the  $t_{ik}$  to be 1 when the patch  $i$  belongs to the group  $k$  and 0 otherwise, the M-step consists in inferring the parameters of the Gaussian models for the groups, while the E-step uses the knowledge of the inferred model to update the groups themselves. This model provides a better clustering of the patches than a K-means clustering with the Euclidean norm (which only produces isotropic clusters) and consequently should yield better denoising results. This idea is used in [11, 13, 21] and the GMM model on patches is also used in [10]. A straightforward implementation of the denoising with a GMM model on the patches gives the result in the first line of Fig. 5.11. However, this inference of a GMM also strongly suffers from the curse of the dimensionality and algorithms such S-PLC [11] or HDML [13, 22] propose to use Gaussian Mixture models with intrinsic lower dimensions in order to reduce the number of parameters to estimate, as detailed in the following section.

### 5.5.4 Inference in High Dimension

The dimensions of the patch spaces are usually high, from  $p = 9$  (for  $3 \times 3$  patches) to  $p = 100$  for  $10 \times 10$  patches, or even higher. Estimating the parameters of Gaussian models (or GMM) in such high-dimensional spaces is complex. When  $p$  is large, patches seen as points in  $\mathbb{R}^p$  are essentially isolated, the Euclidean distance and the notion of nearest neighbor become much less reliable than in low-dimensional spaces [23]. These phenomena, known as the curse of dimensionality, cause difficulties to decide which patches should be grouped together in a common Gaussian model. Besides, parametric models such as Gaussian Mixture Models in high dimension are usually over-parametrized: the covariance matrix of a Gaussian model in dimension  $p = 100$  contains 5050 different coefficients. They necessitate huge quantities of data to be estimated correctly. Indeed, the convergence of the sample





**Fig. 5.11** First line: denoising with a full GMM model (50 groups) on all the patches. The clustering (left) is quite noisy and the denoising result (right) is not very good (PSNR: 28.50 dB). Second line: denoising with a GMM model (50 groups) with intrinsic dimension regularization as in [13]. The clustering (left) is smoother and the denoising yields quite good results (PSNR: 31.23 dB)

covariance matrices to the true covariance matrix depends on the ratio between the number  $n$  of samples and the dimension  $p$ . More precisely, if  $n$  and  $p$  both tend toward infinity while  $\frac{n}{p}$  tends toward a constant  $c > 0$ , the eigenvalues of the sample covariance matrix  $\widehat{\Sigma}(n)$  do not necessarily converge towards the eigenvalues of the model covariance matrix (Marčenko-Pastur Theorem [24] describes the limit law of the empirical distribution of these eigenvalues).

A consequence of the curse of dimensionality is that clustering methods such as K-means or GMM are often disappointing in high dimension, or do not converge at all if  $p$  is too large. Solutions to circumvent these problems usually rely on dimension reduction, or regularization of the model parameters. For instance, if the sample covariance matrix  $\Sigma$  is singular or ill-conditioned, or is not definite positive, it is usual

to add a small  $\varepsilon I_p$  to it. This is the strategy followed by [8, 10]. In the case of Gaussian Mixture Models, another approach consists in assuming that the intrinsic dimension of the Gaussian is lower than  $p$ . This is the idea adopted in [11], where the groups' intrinsic dimensions are heuristically fixed to 1 (flat regions),  $\frac{p}{2}$  or  $p - 1$ . A more involved method consists in inferring for each group its own intrinsic dimension [13] (see Fig. 5.11). The corresponding parsimonious model assumes that each Gaussian of the mixture lives in its own specific subspace.

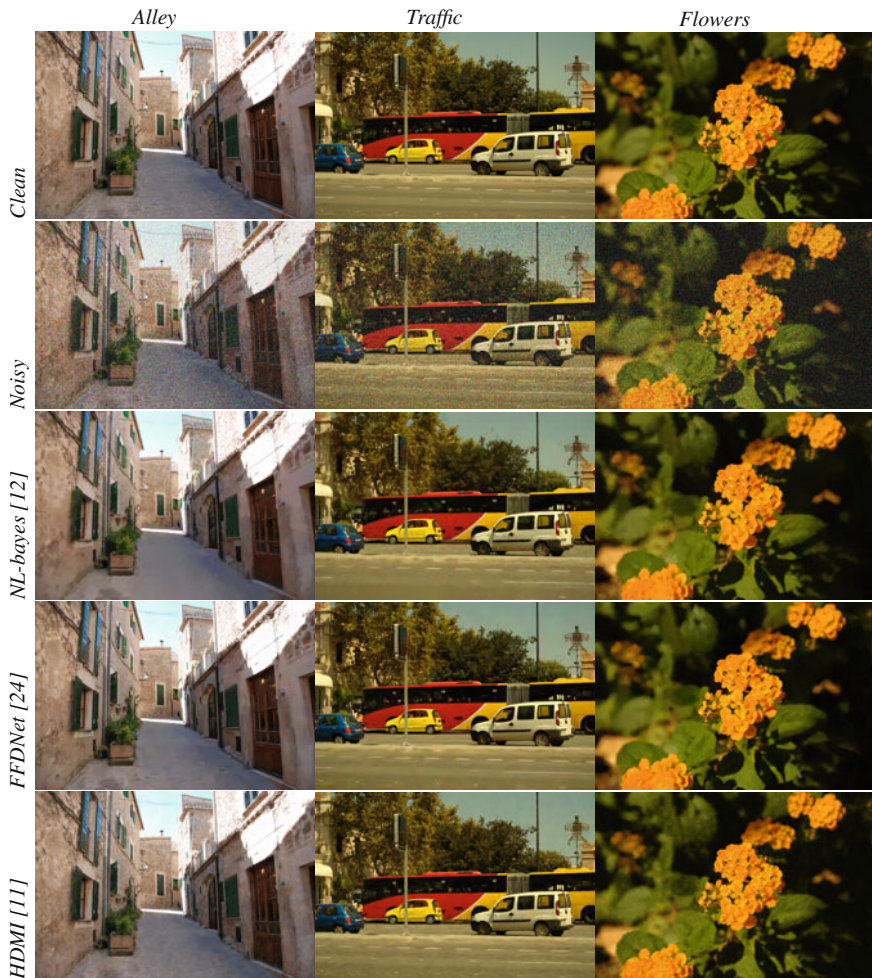
**Table 5.1** This table summarizes the main features of the different methods mentioned in this chapter. Each line refers to a patch-based denoising method and the reference paper where it has been introduced. The columns correspond to the different steps we discussed in this chapter

Method	Grouping	Modeling	Dimension reduction	Remarks	Denoising	Aggregation
Global [16]	All patches	Gaussian models	No	–	Wiener/HT	Uniform
Local [16]	Local grouping in the image space	Gaussian models	No	–	Wiener/HT	Uniform
K-means	k-means in the patch space	Gaussian models	No	–	Wiener/HT	Uniform
NL-Bayes [8]	Nearest neighbors in the patch space	Gaussian models	No	Flat areas are treated separately	Wiener	Uniform
PLE [10]	GMM		No	MAP-EM algorithm	Wiener at each step of the MAP-EM algorithm	Uniform
S-PLE [11]	GMM		Yes	Fixed intrinsic dimensions	MMLE	Uniform
HDMI [13]	GMM		Yes	Estimation of the intrinsic dimensions	MMLE	Uniform
EPLL [9]	–	GMM	No	GMM parameters inferred on an external base	Variational formulation	



## 5.6 Discussion and Conclusion

In this chapter, we have focused on patch priors for image denoising. As we have seen, assuming Gaussian and GMM priors on image patches is now quite common in the restoration literature. These approaches yield simple image models, usually quite easy to interpret. We have tried to provide a unified point of view for all of these methods, in order to underline their similarities and differences. Table 5.1 summarizes



**Fig. 5.12** Some denoising results for the RGB images *Alley*, *Traffic* and *Flowers* at  $\sigma = 50$  with different methods of the literature. The NL-Bayes method is used with default settings and the HDMI method uses  $K = 50$  groups. FFDNet is used with a noise map at  $\sigma = 50$ . Images should be seen at full resolution on the electronic version of the chapter

the main features of the methods mentioned in this chapter. We have also described some of their limitations, such as the inference difficulties in high dimension or the absence of invariance properties to geometric transformations.

While these patch-based methods have now reached a good level of maturity and are used in both academic and industrial settings (the NL-Bayes [8] algorithm for instance is behind the PRIME denoising technology in DxO PhotoLab software), we cannot finish this chapter without mentioning the recent success of deep learning methods in the restoration literature. These methods, which had already shown their efficiency in computer vision, also started to show impressive results in the field of image restoration and editing. Despite this success, their results are strongly dependent on the learning databases and they often create new type of image artifacts. We provide in Fig. 5.12 color denoising results for several images and three denoising methods: two patch-based methods relying either on local Gaussian models (NL-Bayes [8]) or on Gaussian Mixture Models with specific low-dimensional subspaces (HDMI [13]) and a state-of-the-art approach using convolutional neural networks (FFDNet [25]). These three methods clearly show complementary strengths and weaknesses. While FFDNet yields excellent results in smooth or constant areas, it tends to oversmooth fine textures (window shutters in *Alley*, trees in *Traffic*). This might come from the fact that these specific textures are not well represented in the learning database. On the contrary, the GMM-based approach gives good results on these repetitive structures but misclassifications tend to yield artifacts, for instance under the form of low-frequency noise in flat areas. To conclude, we think that these two paradigms should be seen as complementary rather than competitors.

## References

1. Mumford D, Desolneux A (2010) Pattern theory: the stochastic analysis of real-world signals. CRC Press
2. Aguerrebere C, Delon J, Gousseau Y, Musé P (2012) Study of the digital camera acquisition process and statistical modeling of the sensor raw data. Preprint Hal 00733538
3. Markku M, Alessandro F (2011) A closed-form approximation of the exact unbiased inverse of the Anscombe variance-stabilizing transformation. *IEEE Trans Imag Process* 20(9):2697–2698
4. Ordentlich E, Seroussi G, Verdu S, Weinberger M, Weissman T (2003) A discrete universal denoiser and its application to binary images. In: 2003 international conference on image processing, 2003. *ICIP 2003. Proceedings*, vol 1, pp 1–117. IEEE
5. Weissman T, Ordentlich E, Seroussi G, Verdú S, Weinberger MJ (2005) Universal discrete denoising: known channel. *IEEE Trans Inf Theory* 51(1):5–28
6. Awate S, Whitaker R (2004) Image denoising with unsupervised information-theoretic adaptive filtering. In: International conference on computer vision and pattern recognition (CVPR 2005), pp 44–51
7. Buades A, Coll B, Morel JM (2006) A review of image denoising algorithms, with a new one. *Multiscale Model Simul* 4(2):490–530
8. Lebrun M, Buades A, Morel JM (2013) A nonlocal Bayesian image denoising algorithm. *SIAM J Imag Sci* 6(3):1665–1688
9. Zoran D, Weiss Y (2011) From learning models of natural image patches to whole image restoration. In: 2011 International Conference on Computer Vision, pp 479–486. IEEE

10. Yu G, Guillermo S, Stéphane M (2012) Solving inverse problems with piecewise linear estimators: from Gaussian mixture models to structured sparsity. *IEEE Trans Imag Process* 21(5):2481–2499
11. Yi-Qing W, Jean-Michel M (2013) SURE guided Gaussian mixture image denoising. *SIAM J Imag Sci* 6(2):999–1034
12. Aguerrebere C, Almansa A, Delon J, Gousseau Y, Musé P (2017) A Bayesian hyperprior approach for joint image denoising and interpolation, with an application to HDR imaging. *IEEE Trans Comput Imag*
13. Houdard A, Bouveyron C, Delon J (2017) High-dimensional mixture models for unsupervised image denoising (HDMI) (preprint)
14. Kostadin D, Alessandro F, Vladimir K, Karen E (2007) Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Trans Imag Process* 16(8):2080–2095
15. Teodoro AM, Almeida MSC, Figueiredo MAT (2015) Single-frame image denoising and inpainting using gaussian mixtures. In: *ICPRAM*, vol 2, pp 283–288
16. Charles-Alban D, Salmon J, Dalalyan AS et al (2011) Local versus global. Image denoising with patch based PCA. *BMVC* 81:425–455
17. Salmon J, Strozecki Y (2010) From patches to pixels in non-local methods: weighted-average reprojection. In: *2010 17th IEEE international conference on image processing (ICIP)*, pp 1929–1932. IEEE
18. Nicola P, Jean-Michel M, Gabriele F (2017) Multi-scale DCT denoising. *Imag Process* 7:288–308
19. Charles-Alban D, Loïc D, Florence T (2012) How to compare noisy patches? patch similarity beyond gaussian noise. *Int J Comput Vis* 99(1):86–102
20. Bishop CM (2006) *Pattern recognition and machine learning*. Springer
21. Jianbo Y, Xuejun L, Xin Y, Patrick L, Brady DJ, Sapiro G, Carin L (2015) Compressive sensing by learning a gaussian mixture model from measurements. *IEEE Trans Imag Process* 24(1):106–119
22. Houdard A, Bouveyron C, Delon J (2017) Clustering en haute dimensions pour le débruitage d'image. In: *XXVlième colloque GRETSI*
23. Giraud C (2014) *Introduction to high-dimensional statistics*, vol 138. CRC Press
24. Marčenko VA, Pastur LA (1967) Distribution of eigenvalues for some sets of random matrices. *Math USSR-Sbornik* 1(4):457
25. Zhang K, Zuo W, Zhang L (2018) Ffdnet: toward a fast and flexible solution for CNN based image denoising. *IEEE Trans Imag Process*

# Chapter 6

## Internal Versus External Denoising—Benefits and Bounds



Maria Zontak and Michal Irani

**Abstract** Image denoising has been a popularly studied problem for several decades in image processing and low-level computer vision communities. Many effective denoising approaches, such as BM3D, utilize spatial redundancy of patches (relatively small, cropped windows) either within a single natural image, or within a large collection of natural images. In this chapter, we summarize our previous finding that “Internal-Denoising” (based on internal noisy patches) can outperform “External Denoising” (based on external clean patches), especially in the presence of high noise levels. We explain this phenomenon in terms of “Patch Signal-to-Noise Ratio” (*PatchSNR*), an inherent characteristic of a noisy patch that determines its preference of either internal or external denoising. We further experiment with the recent state-of-the-art convolutional residual neural network for Gaussian denoising. We show that it closes the gap on the previously reported external denoising bounds. We further compare its performance to internal local multi-scale Oracle (that has the same receptive field as the network). We show that for patches with low *PatchSNR*, the network does not manage to reconstruct the best “clean” patch that resides in the network’s receptive field. This suggests that the future challenge of denoising community is to train an image-specific CNN that will exploit local recurrence of patches, without relying on external examples, as was recently successfully done for super-resolution task. Combining such a model with external-based models may push PSNR bounds further up and improve denoising by  $\sim 1\text{--}2$  dB, especially for higher noise levels.

---

M. Zontak (✉)

Northeastern University Seattle, Seattle, USA

e-mail: m.zontak@northeastern.edu; zontakm@uw.edu; maria.zontak@northeastern.edu

M. Irani

Weizmann Institute of Science, Rehovot, Israel

e-mail: michal.irani@weizmann.ac.il

© Springer International Publishing AG, part of Springer Nature 2018

M. Bertalmío (ed.), *Denoising of Photographic Images and Video*,

Advances in Computer Vision and Pattern Recognition,

[https://doi.org/10.1007/978-3-319-96029-6\\_6](https://doi.org/10.1007/978-3-319-96029-6_6)

## 6.1 Introduction

Image denoising has been a popularly studied problem for several decades in image processing and low-level computer vision. In this problem, we are given a corrupted noisy image,  $I_N = I + N$ , and want to recover the original clean image  $I$ . This problem is very under-constrained and ill-posed. Given a noisy image  $I_N$ , there are many possible solutions of  $I$ , since the noise  $N$  is unknown. However, natural images occupy only a tiny portion of the space of all possible solutions  $I$ , a fact that encouraged researchers to continue seeking for new ways to perform better natural image denoising. Most of the proposed approaches assumed Gaussian additive noise of zero mean and known variance  $\sigma^2$ . We will follow this assumption in our discussion below.

Until 2012, the most effective denoising approaches [1, 6, 7, 11, 21, 22] utilized spatial redundancy of patches (relatively small, cropped windows) within a single natural image, as well as within a large collection of natural images. The fact that small image patches tend to recur within natural images has been successfully used in many denoising algorithms, such that similar patches were combined to reduce the effects of noise. The seminal patch-based methods were the Non-Local Means (NLM) denoising of Baudes et al. [6, 7] and the K-SVD denoising of Aharonov and Elad [1]. These methods were further improved by BM3D denoising [11] and LSSC denoising of Mairal et al. [21, 22].

The Non-Local Means denoising [6, 7] solely relies on patch recurrence within a single noisy image, and eliminates noise by averaging similar patches. While demonstrating state-of-the-art performance upon publication, the NLM algorithm was limited by the ability of finding “true” similar patches in a noisy regime. This problem was addressed by the BM3D [11] algorithm, that suggested two-stage denoising, where the image was first roughly denoised and afterwards the denoising was refined by searching for similar patches in previously denoised image. This and other improvements, such as a different strategy for combination of similar noisy patches, made BM3D a very effective algorithm. Despite the “non-local” intention, both Baudes et al. [6, 7] and Davob et al. [11] restrained the search for similar patches to the relatively close vicinity of the patch. Usually, this was motivated by computational efficiency, however in [32], we explained that local neighborhood might be more beneficial for performance as well. We elaborate on this in Sect. 6.2.1.

The K-SVD denoising [1, 2, 12, 13], proposed to reconstruct an underlying clean patch by sparsely representing a noisy patch with an over-complete learned dictionary. Initially, such dictionary was learned from a big collection of clean natural images [2], however later [13], the dictionary was directly learned from the patches of a noisy image. Interestingly, the authors did not find any advantage of a particular method. This might seem surprising, since “External” dictionary is learned from patches of a large collection of *clean* images, while “Internal” dictionary is learned from the patches of a **noisy** images. We analyzed and explained this surprising behavior in [23]. We showed that some patches in the image benefit more from *internal denoising*, while others benefit more from **external denoising**. As such, on average

both algorithms can perform equally well. We elaborate on Internal versus External preference in Sect. 6.2.2 and discuss the inherent patch characteristics, “Patch Signal-to-Noise Ratio”, which determines this preference, in Sect. 6.2.3. We further suggested how the power of Internal and External denoising can be combined in Sect. 6.3.

Image denoising took a new, promising route with the revival of deep neural networks. The general idea is simple, during the training stage, the network learns how to map the provided noisy input into a clean output based on noisy-clean image pairs from the external database of training examples. While the training stage is time consuming, the test stage, especially if implemented on GPU, is fast, which makes such networks very compelling. Several different architectures have been proposed, including fully connected denoising autoencoders [29], multilayer perceptron (MLP) [14] and convolution networks (CNNs) [4, 31]. While the denoising autoencoders proposed in [29] did not manage to surpass the state-of-the-art, the MLP [14] did. MLP network performed well when tailored to a specific noise level, but lacked the ability to generalize to a blind denoising task, where the same network should be able to perform well on several noise level. The residual denoising convolutional neural network, which was introduced in [31], demonstrated that a single neural network can address a wide range of noise levels, while still improving the state-of-the-art PSNR. Inspired by ResNet [15], the authors proposed to learn to reconstruct the noise, instead of the clean signal, and demonstrated substantial improvement in PSNR values for both single noise-level denoising tasks and blind denoising (as well as other image processing applications). This network was further improved by [4], which suggested wavelet domain deep residual learning network for Gaussian denoising task. The main idea of this paper is that image transform into wavelet domain reduces the topological complexity of data and label manifold and hence improves the network predictive power. In Sect. 6.4, we discuss how such CNN-based methods perform with respect to the internal and external denoising bounds, and we conclude that there is still room for improvement.

## 6.2 On the Preference of Local Denoising

The denoising problem has a rich history. Early work includes anisotropic denoising [3, 25], various wavelet based techniques (e.g., [26]), and more. The underlying principle in these techniques relies on the variance law in probability theory, which ensures that if  $d$  independent noise samples are averaged, the noise standard deviation diminishes by a factor of  $1/\sqrt{d}$ . Thus, to denoise a noisy pixel,  $d - 1$  similar pixels should be averaged. Typically, the similar pixels were assumed to be found in the immediate vicinity of the pixel to be denoised. The major leap in performances happened in 2005, when Buades et al. [6] observed that the most similar pixels need not necessarily be near. For example, periodic patterns, or elongated edges, which appear in most images, have many similar pixels that do not lie in the close vicinity of each other. Thus, in the Non-Local Means (NLM) algorithm, the noisy pixel is

replaced by the mean value obtained from other image pixels, weighted by their degree of similarity of their surrounding image patch to that of the denoised pixel:

$$\hat{u}(p) = \frac{1}{\sum_{q \in \mathcal{N}_r^p} w(p, q)} \sum_{q \in \mathcal{N}_r^p} u(q) w(p, q), \quad (6.1)$$

where  $u$  is the noisy image,  $p$  is the denoised pixel and the similarity weight  $w(p, q)$  is typically based on the similarity of small images patches centered around the pixels  $p$  and  $q$  (a similar formula can be applied to denoise a patch  $p$ ).

The name ‘‘Non-Local’’ is somewhat misleading, as can be seen in the above formula, only the pixels that are found in  $\mathcal{N}_r^p$ , a neighborhood centered around  $p$  of size  $2r + 1 \times 2r + 1$ , are averaged. In typical NLM implementation (as well as in the state-of-the-art BM3D [11]), the radius  $r$  is relatively small, typically around 20 pixels. Usually, this choice is justified by computational efficiency consideration. In [32], we showed that such a constrained search has performance advantages beyond the computational ones. We elaborate on that next.

### 6.2.1 Internal Local Versus Global Denoising

When defining a neighborhood, the primary goal is to discover as many patches as possible with similar underlying clean signal. In [32], we observed that an image patch is much more likely to recur near itself than far away. By quantifying the patch recurrence within an image, we showed that a patch density is high within its closest vicinity, and that this density drops rapidly as the distance from the patch grows.

Our experiments were conducted on the 300 images from [5]. For each image patch  $p$ , we estimated its *empirical density* within an image neighborhood  $\mathcal{N}_{dist}$  of radius ‘‘ $dist$ ’’ around the patch, using Parzen window estimation [24]:  $density(p; dist) = \sum_{q \in \mathcal{N}_{dist}} \mathcal{K}_h(\|p - q\|_2^2) / area(\mathcal{N}_{dist})$ , where  $q$  are all the image patches within a spatial neighborhood  $\mathcal{N}_{dist}$ , and  $\mathcal{K}_h(\cdot)$  is a Gaussian kernel. Averaging these individually computed patch densities over the set of all patches with the same gradient magnitude  $|grad|$ , we obtain the following average density:

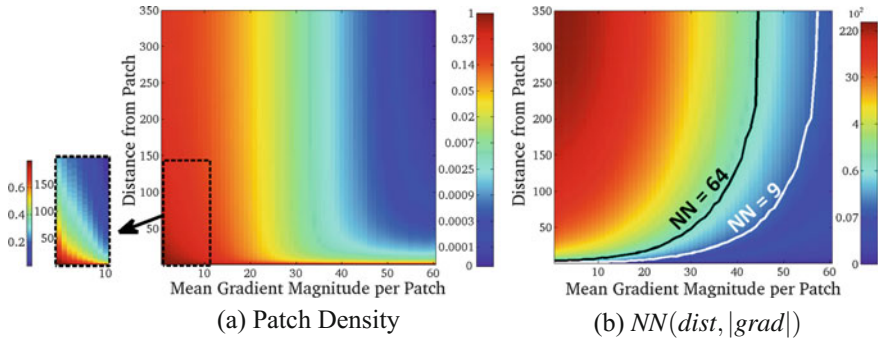
$$Density(dist, |grad|) = Mean_{p \in \{|grad|\}} density(p, dist). \quad (6.2)$$

The average number of ‘‘good Nearest Neighbors’’  $NN$  within a distance  $dist$  from the patch is defined as:

$$NN(dist, |grad|) = Density(dist, |grad|) \cdot area(\mathcal{N}_{dist}). \quad (6.3)$$

Note that the Parzen estimation does not distinguish between 10 perfectly similar patches, and 100 partially similar patches. We loosely refer to these as 10 good NNs.





**Fig. 6.1 Internal statistics of a single natural image.** (a–b The empirical density ( $dist, |grad|$ ) of patches and the number of “similar” patches,  $NN(dist, |grad|)$ , as a function of the mean gradient magnitude  $|grad|$  in the patch, and the spatial distance  $dist$  from the patch location (Red signifies high values and blue signifies low values)

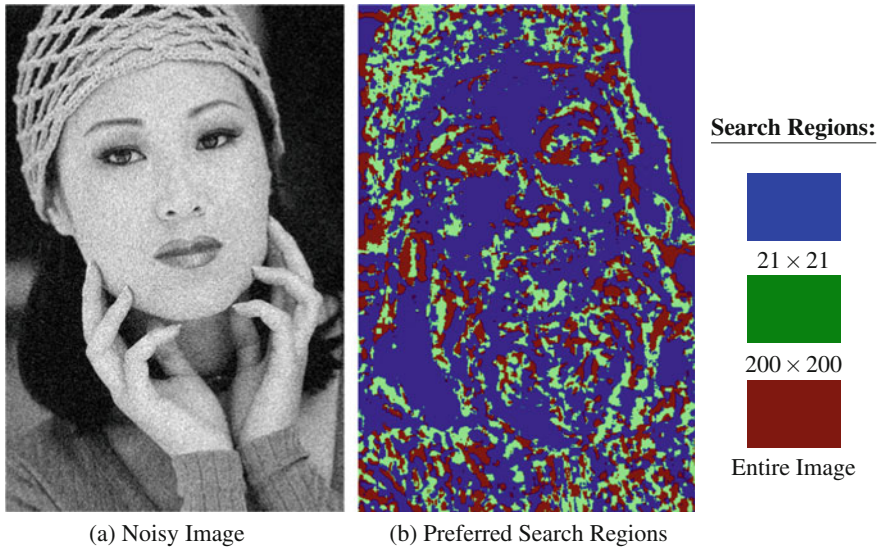
Figure 6.1a displays the empirical density  $Density(dist, |grad|)$  and Fig. 6.1b displays the number of “similar” patches  $NN(dist, |grad|)$ , both as a function of the mean gradient magnitude  $|grad|$  of the patch, and the spatial distance  $dist$  from the patch location. In both maps, red signifies high values and blue signifies low values. Observing these maps, we note that a patch tends to recur densely in its closest vicinity (small  $dist$ ), and its frequency of recurrence decays rapidly as the distance from the patch increases (see the zoomed-in part in Fig. 6.1a).

Namely, patches in a natural image are likely to reside in spatial clusters of similar patches. Therefore, since a patch has enough similar patches in its close vicinity, it is *not* surprising that NLM works well, despite its relatively local search.

However, *what is surprising* is that the local search is sometimes *preferable* over a ‘global’ search in the entire image. Figure 6.2 visualizes this surprising finding. We ran the NLM algorithm on the noisy image of Fig. 6.2a, with three different search regions: (i)  $21 \times 21$ , (ii)  $200 \times 200$ , (iii) the entire image. For each pixel in the image, we marked which of the three search regions gave it the smallest error relative to the ground-truth clean image (Fig. 6.2b). We further note that relatively smooth region (face, hair and background), which are characterized by low variance, benefit more from locally constrained neighborhood, while more detailed patches prefer bigger search regions.

This seems counterintuitive, because previously we explained that the noise standard deviation decreases by factor of  $\sqrt{d}$ , where  $d$  is the number of averaged samples. It is obvious from Fig. 6.1b that bigger neighborhoods contain more similar patches, and therefore should be better for denoising. However, evidently this is not the case for some patches. We will further analyze this phenomenon in Sect. 6.2.3.





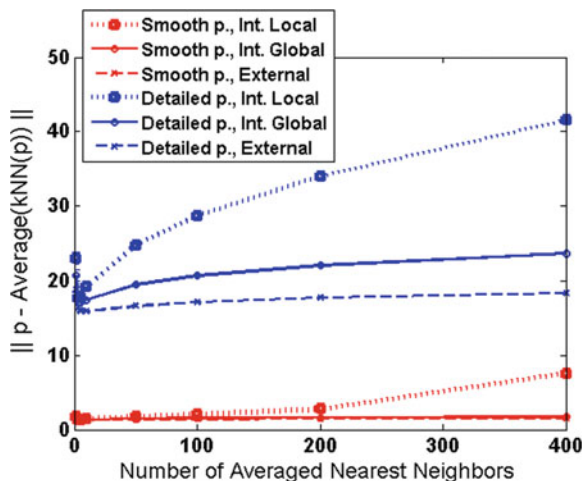
**Fig. 6.2 Preferred search regions per patch in NLM:** Smooth patches obtain better denoising results in local search regions; detailed patches benefit from large search regions

## 6.2.2 Internal Local Versus External Denoising

We further show that for some patches internal local search among the **noisy** surrounding patches is preferable, even when compared to the external search in a collection of **clean images**. Note that when searching for similar patches for a noisy patch  $p_n = p + n$ , we aim to find patches that **fit the underlying signal**  $p$ . Hence, successful denoising will rely on existence of sufficiently similar patches to  $p$  among the available patches. Next, we check how well a **clean** patch  $p$  is represented internally versus externally, in other words, we analyze internal versus external **signal fitting** quality.

Figure 6.3 shows the root mean squared error (RMSE) between a clean patch  $p$ , and the average of its most similar patches, i.e., its k-Nearest Neighbors (k-NNs), computed for  $7 \times 7$  patches. We examined three cases: (i) The k-NNs are selected from an internal  $21 \times 21$  neighborhood surrounding  $p$  (dotted curve); (ii) The k-NNs are selected internally from the entire image (solid curve); (iii) The k-NNs are selected from an external database of 200 images (dashed curve). The red color refers to the 25% smoothest patches in the image (lowest  $\text{var}(p)$ ), and the blue curve—to the 25% most detailed patches in the image (highest  $\text{var}(p)$ ). These results were averaged over patches taken from 100 images.

Smooth patches (red curves) have many good NNs both internally and externally. Averaging up to 100 NNs yields approximately similar representation error for both local internal and external searches. On the other hand, the detailed patches  $p$  (blue curves) have much better representatives externally than internally. Averaging 100



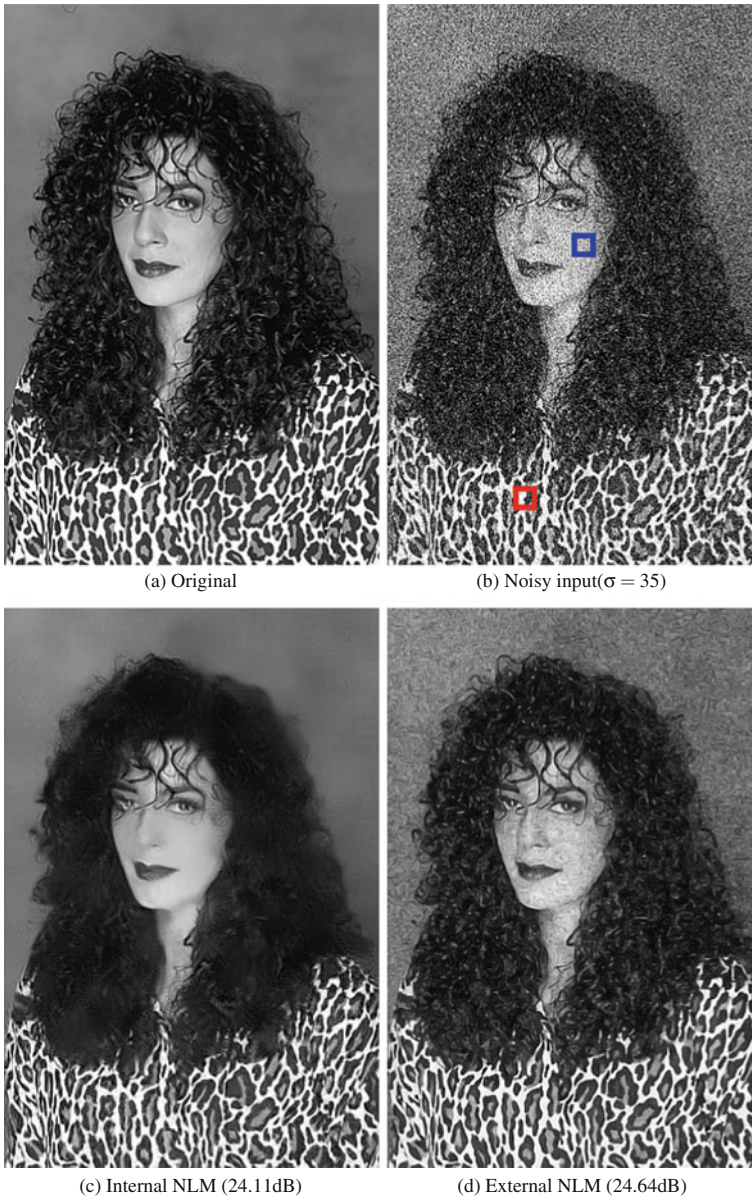
**Fig. 6.3 Signal fitting:** RMSE between a *clean* patch  $p$  and the average of its  $k$ -NNs (computed for  $7 \times 7$  patches). *Dotted curve (Internal-Local)*:  $k$ -NNs selected from an internal  $21 \times 21$  neighborhood around  $p$ ; *Solid curve (Internal-Global)*:  $k$ -NNs selected internally from the entire image; *Dashed curve (External)*:  $k$ -NNs selected from an external database. *Red curves*: the 25% smoothest patches in the image; *Blue curves*: the 25% most detailed patches in the image (Statistics over 100 images.)

internal NNs leads to a very high error. In fact, this error is much higher than averaging any number  $k$  of external NNs in the graph. Thus, while smooth patches have sufficient representation internally, detailed patches do not.

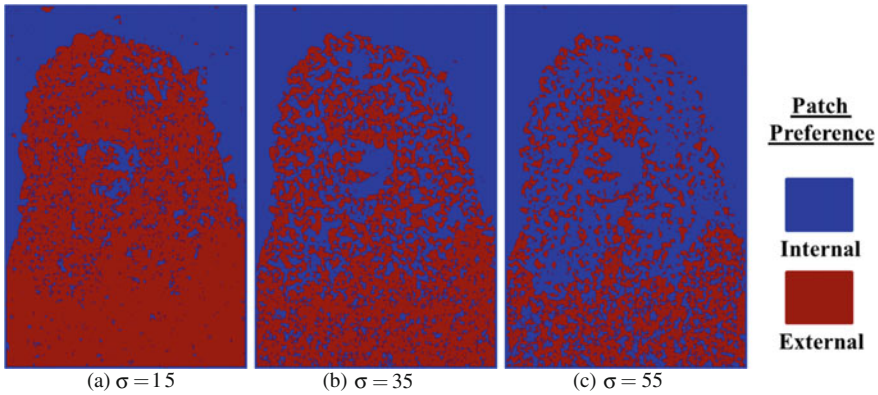
Moreover, once noise is added, the internal  $k$ -NNs (local or global) will be noisy, whereas the external ones will not, giving additional advantage to external denoising. Next we show that despite the apparent advantages of external denoising: better signal fit and clean patches, it often performs worse than internal denoising for many image patches.

Figures 6.4c, d show the denoising results of the Internal local NLM and External NLM (using  $7 \times 7$  patches) on the noisy image of Fig. 6.4b (Gaussian noise with  $\sigma = 35$ ). Internally, each noisy  $7 \times 7$  patch was denoised by taking a weighted average of its neighboring noisy image patches (weighted by their degree of similarity to the patch) within a local,  $21 \times 21$  neighborhood. Externally, each noisy  $7 \times 7$  patch was denoised by taking a weighted average of clean patches from an external database of 200 clean natural images (taken from the train set of the *BSDS300* [5]). Note that while some parts of the image are indeed better recovered by the External NLM (in particular, the textured regions, e.g., the woman’s hair), other parts (the smooth regions, e.g., the woman’s face, the background) are poorly denoised using external NLM and are better recovered by the Internal NLM.

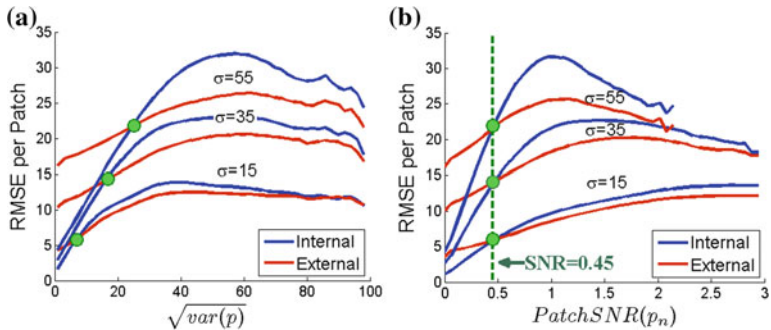
Surprisingly, this phenomenon for Internal denoising versus External denoising only grows as the noise level grows. This is illustrated in Fig. 6.5, which displays the Internal versus External patch preference for three different noise levels (added to the



**Fig. 6.4** Internal versus external denoising (NLM with  $7 \times 7$  patches,  $\sigma = 35$ ). Internal NLM is better for smooth patches; External NLM is better for detailed patches



**Fig. 6.5 Internal versus external patch preference.** Patch preference between Internal and External NLM for different noise levels: Red marks external preference; Blue marks internal preference. Note that the higher the noise in the image, the stronger the preference for internal denoising

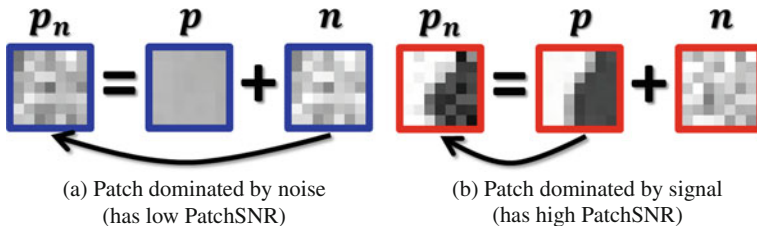


**Fig. 6.6 Denoising error as a function of patch variance and patchSNR.** Statistics performed on 100 natural images (noise levels  $\sigma = 15, 35, 55$ ). Note that the threshold between Internal and External preference depends only on the PatchSNR, and is independent of the global noise level  $\sigma$

image of Fig. 6.4a). For each noisy patch, we measured which of the two algorithms denoised it better (i.e., obtained a smaller RMSE) with respect to the ground-truth image of Fig. 6.4a. Blue marks *internal preference*, red marks *external preference*. Indeed, Fig. 6.5 shows that as the noise level grows (from left to right), more patches prefer the Internal denoising. This observation is surprising and counterintuitive, because one might expect that when the image patches are noisier, using clean patches (from an external database) would be preferable. Surprisingly, this is not the case.

To empirically validate that our observations hold in general for natural images, we repeated this experiment on 100 different natural images (taken from the test set of *BSDS300*) for three different noise levels ( $\sigma = 15, 35, 55$ ).

Figure 6.6a shows the average denoising error (RMSE) as a function of the patch variance (of the ground-truth clean patch). It further confirms that the preference



**Fig. 6.7 Signal versus noise dominance within a patch.** The noisy patches **a** and **b** are the red and blue patches marked in Fig. 6.4b. Patch **a** prefers Internal NLM, whereas Patch **b** prefers External NLM

for Internal NLM gets stronger as the noise level grows (the threshold between the Internal/External preference increases with the noise level  $\sigma$ ). Indeed, patches with low variance (smooth or low-content patches) prefer Internal NLM, whereas patches with high variance (patches with details) prefer External NLM.

### 6.2.3 Patch Signal-to-Noise Ratio (PatchSNR) and Its Implication on Internal Versus External Preference

To understand the growing preference presented in Figs. 6.5 and 6.6a, consider the smooth (blue) patch and the detailed (red) patch from Fig. 6.4b. In Fig. 6.7, we decompose these noisy patches,  $p_n$ , into their underlying signal,  $p$ , and added noise,  $n$ . The smooth patch is dominated by noise  $n$  (Fig. 6.7a), while the detailed patch is dominated by signal  $p$  (see Fig. 6.7b). In [23], we proposed to measure this dominance using “Signal-to-Noise Ratio” of a noisy patch, defining it as

$$PatchSNR(p_n) \stackrel{\text{def}}{=} \sqrt{\frac{\text{var}(p)}{\text{var}(n)}}$$

In the next section, we analytically show that patches with low  $PatchSNR(p_n)$  are prone to **noise fitting** instead of signal fitting. Moreover, the chance of such patches to overfit the noise grows as their search space grows (e.g., to the entire image or to a collection of external clean images). We further show that such noise-overfitting is avoided by a local internal denoising. As the global noise level  $\sigma$  increases, the PatchSNR of each patch decreases. Therefore, more patches have lower PatchSNR and hence more patches prefer Internal local denoising over External denoising.

To validate this claim empirically, Fig. 6.6b shows the RMSE per patch, but this time plotted as a function of the PatchSNR. As can be seen, there is a clear threshold between the Internal/External preference, which *does not depend on the global noise level*  $\sigma$ . This shows that the Internal/External denoising preference is tightly related to the PatchSNR.



### 6.3 Noise Overfitting

In this section, we quantify the noise overfitting phenomenon and its relation to the PatchSNR. As previously assumed, the noise is an additive random noise, sampled from Gaussian distribution with variance  $\sigma^2$  and zero mean. In [6], the authors explain that under this assumption, applying Euclidean distance between noisy patches ( $L_2$  norm of their differences) will yield:

$$\begin{aligned} E(\|p_n - q_n\|^2) &= E(\|p - q + n_p - n_q\|^2) = \\ &\|p - q\|^2 + E(\|n_p - n_q\|^2) = \|p - q\|^2 + 2\sigma^2, \end{aligned} \quad (6.4)$$

where  $n_p$  and  $n_q$  are the noise realizations added to patches  $p$  and  $q$  respectively. By assumption, these noise realizations are independent from each other, as well as from the signals  $p$  and  $q$ . The authors therefore conclude that in expectation the Euclidean distance between noisy patches preserves the original similarity of clean patches. However is this indeed the case for each patch  $p_n$ ?

In [23], we observed that although the mean of the noise in the entire image is zero, in practice, the *sample (empirical) mean* of the noise within an *individual* relatively small patch is not zero (as illustrated for  $7 \times 7$  in Fig. 6.8b). Similarly, the *sample (empirical) variance* within an individual relatively small patch is usually not  $\sigma^2$  (see Fig. 6.8a).

Generally, given a sample of size  $d \times d$ , consider  $d^2$  independent random variables  $n_1, n_2, \dots, n_{d^2}$ , each corresponding to one randomly selected observation. Each of these variables has the assumed Gaussian  $(0, \sigma^2)$  distribution. The *sample mean* is defined as

$$\bar{n} = \frac{1}{d^2} \sum_1^{d^2} n_i, \quad (6.5)$$

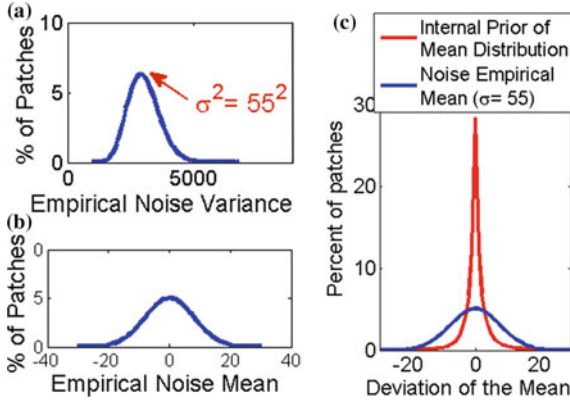
therefore, by the properties of means and variances of random variables, the mean and variance of the sample mean are the following:

$$\mu_{\bar{n}} = \mu \quad \& \quad \sigma_{\bar{n}} = \sigma/d, \quad (6.6)$$

For example, given a  $7 \times 7$  ( $d = 7$ ) patch  $n$  of random noise sampled from Gaussian distribution with  $(\mu = 0, \sigma = 55)$ , its sample mean  $\bar{n}$  will be distributed around 0 with standard deviation approximately equal to 7.85, in agreement with the empirical distribution shown in Fig. 6.8b (typically the spread of values for normal distribution is  $\mu \pm 3\sigma \cong \pm 23.55$ ).

Similarly, we can talk about *sample variance*:

$$\frac{1}{d^2} \sum_1^{d^2} (x_i - \bar{x})^2 \sim \frac{\sigma^2}{d^2} \chi_{d^2-1}, \quad (6.7)$$



**Fig. 6.8** Deviations of empirical mean and variance (shown for  $\sigma = 55$ ): **a** Distribution of the empirical variance of  $7 \times 7$  random noise patches. **b** Distribution of the empirical mean of  $7 \times 7$  random noise patches. **c** Red curve—deviations of the mean of  $7 \times 7$  clean natural patches w.r.t. the central patch mean within a restricted  $21 \times 21$  area. For comparison, the expected deviations of random noise mean are overlaid on top (marked in blue)

where  $\chi_{d^2-1}^2$  is a chi-squared distribution with  $d^2 - 1$  degrees of freedom. For example, for  $\sigma = 55$ , the distribution will have mean of  $\frac{d^2-1}{d^2}\sigma^2 \simeq 55^2$  and variance of  $2\frac{d^2-1}{d^2}\sigma^2$  which fits Fig. 6.8a.

In [23], we analyzed the influence of the difference between the expected and sample values of mean and variance within a small patch on the denoising performance. In particular, we observed that most of the *inherent* denoising error reported in [17] for the case of *optimal* unconstrained external denoising, is due to *overfitting the nonzero (sample) mean of the noise* within the patch (and is invariant of the deviations of the noise variance within the patch). Next, we summarize these findings.

#### (a) Overfitting the Noise Mean:

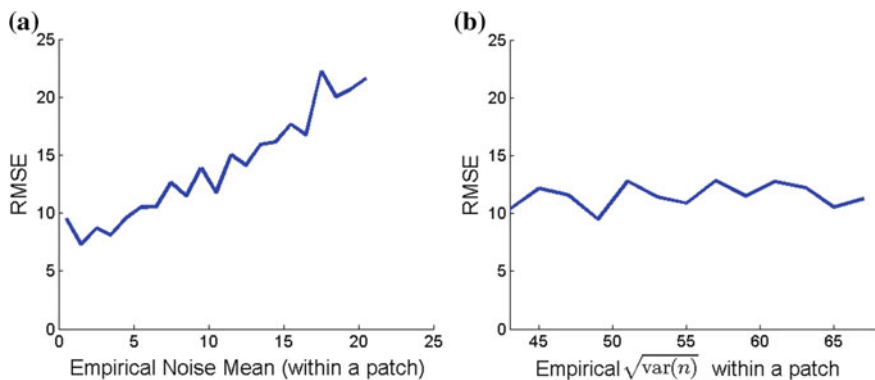
Given a noisy patch  $p_n = p + n$ , let us denote the sample mean of  $n$  as  $\bar{n}$ , which, as we saw, is rarely zero. First, we need to adjust Eq. 6.4 to the case of External denoising, where  $q$  is a clean patch (i.e.,  $n_q = 0$ ):

$$E(\|p_n - q\|^2) = E(\|p - q + n\|^2) = \|p - q\|^2 + E(\|n\|^2) = \|p - q\|^2 + \sigma^2, \quad (6.8)$$

The equation above assumes noise with zero mean. As explained above, for small patches the sample mean will not be zero, thus to keep the equality valid we can rewrite Eq. 6.8 as:  $E(\|p_n - q\|^2) = \|p - q + \bar{n}\|^2 + \sigma^2$ . Assuming that we search externally for a clean patch  $p$ , which is most similar to  $p_n$ , then:

$$\hat{p} = \arg \min_q E(\|p_n - q\|^2) = p + \bar{n}, \quad (6.9)$$

which does not match the desired signal  $p$ .



**Fig. 6.9 Fitting of the noise mean:** RMSE of the optimal external denoising of patches versus sample (empirical) mean and variance of the noise within the patch (computed on the data of [17] for  $7 \times 7$  patches,  $\sigma = 55$ ). The denoising error grows linearly with the deviation from zero of the empirical noise mean within the patch. In contrast, the denoising error is independent of the empirical noise variance within the patch. (Average RMSE values are plotted)

In other words, an *ideal* denoising algorithm (i.e., which has access to an external database containing all possible clean patches) will *inherently* have a residual error of  $\hat{p} - p = \bar{n}$ . Therefore, following Eq. 6.6, the expected error of an “ideal” denoising algorithm, due to the sample mean statistics within a  $d \times d$  patch, is

$$\text{RMSE}_{ideal} = \sqrt{\mathbb{E}(\bar{n}^2)} = \sigma/d, \quad (6.10)$$

For example, for  $\sigma = 55$  using  $7 \times 7$  patches, the minimal expected denoising error is  $\text{RMSE}_{ideal} = 7.85$ . Once the patch size becomes larger, the error due to the overfitting of the sample mean becomes smaller. The “mean-noise” fitting error provides a *lower bound* on the expected error in the case of *ideal* (external) denoising (which can be translated to an upper bound on the expected PSNR performance).

In [17], upper bounds for denoising algorithms are computed by denoising patches via *exhaustive* weighted average over a *huge* number of natural image patches (extracted from 20,000 clean natural images). This framework is close to an *ideal* denoising, because the chances to fit a signal  $p$  perfectly are very high. In [23], we empirically verified that most of the error in *ideal* external denoising, reported in [17], is indeed due to overfitting the sample mean of the noise within a patch. This empirical evaluation was performed using the data from [17] (kindly provided to us by the authors) and is shown in Fig. 6.9.

Figure 6.9a shows that the denoising error in the optimally denoised data of [17] *grows linearly* with the deviation from zero of the noise mean within the patch (shown for  $7 \times 7$  patches,  $\sigma = 55$ ). In contrast, the denoising error is *independent* of the sample variance of the noise within the patch (see Fig. 6.9b). This confirms our observation that overfitting the sample mean of the noise is inherent to unconstrained external denoising. This is the major component of the residual denoising error in the



optimally denoised data of [17], and is the main source for their derived denoising bounds. For example, the error of relatively smooth patches ( $\sqrt{\text{var}(p)} < 4$ ) from the data of [17] is RMSE = 8.13, which is only slightly larger than the above estimated  $\text{RMSE}_{ideal} = 7.85$  only due to fitting the sample mean of the noise within a patch.

Overfitting the “mean-noise” has less risk of occurring internally, when the denoising is restricted to a local,  $21 \times 21$  neighborhood surrounding a patch. This is because there exists a *strong nonuniform prior* on the mean of the clean patch  $p$  in the local neighborhood. Patches in clean natural images tend to recur very densely in their immediate surrounding (see Fig. 6.1a), hence their mean values tend to be very similar. This is especially true for uniform or low-content patches, which form a significant part of the image.

The red curve in Fig. 6.8c displays the small *deviations* of patch mean values within  $21 \times 21$  neighborhood in clean natural images (deviations are measured with respect to the mean value of the central patch). This distribution was empirically calculated for all relatively smooth  $7 \times 7$  patches from 100 natural images of the BSD300 (with  $\sqrt{\text{var}(p)} < 4$ , which is roughly 1/3 of the patches). This distribution is highly localized around zero.

Therefore, when restricting the denoising to averaging patches in a local neighborhood, the expected residual error due to fitting the sample mean of noise is much lower than in the unrestricted external search, especially for low-content patches. That explains how even NLM denoising (which is far from being state-of-the-art), achieves RMSE of 5.11 for relatively smooth patches. This is in contrast to 8.13 obtained for these patches by the optimal external denoising of [17].

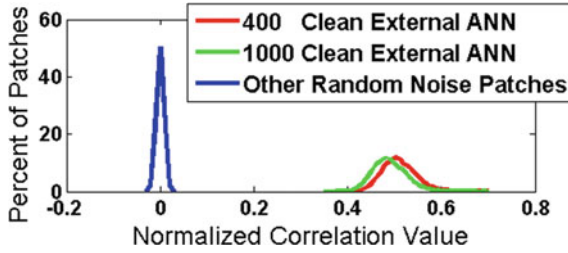
### (b) Overfitting the Noise Details:

We next analyze effects of overfitting the “details” of the noise. The result of Eq. 6.8 is valid under the assumption that signal  $q$  is not correlated with the noise  $n$  within a patch  $p_n$ . However, is this really the case? Next, we try to answer this question.

To isolate the effects of detail fitting, we first remove the mean of all the patches. For simplicity of notations, in the analysis below  $n$ ,  $p$ ,  $p_n$ , etc. will denote patches with zero mean. Let  $c_n$  denote the value of the Normalized Correlation between a *random noise patch*  $n$  and its “most similar” *natural patch*  $NN(n)$  in an external database of clean images. One would expect  $c_n$  to be low (closer to 0 than to 1). However, as shown in Fig. 6.10 (red and green curves), the value  $c_n$  is surprisingly high (around 0.5). Moreover, high normalized correlation values are obtained not only for the first nearest neighbor of  $n$ , but also for its other nearest neighbors (up to 1000 approximate nearest neighbors—ANN). For all our experiments, we used an external database of clean natural image patches from 200 images (from BSD300 [5]).

We experimented with several noise levels  $\sigma$ , and this correlation remains high (almost identical), regardless of  $\sigma$ . Finally, this correlation is even higher for smaller  $5 \times 5$  patches (typically in the range of [0.5 0.8], with an average value of 0.65).

For every random noise patch, we further calculated its “denoised” version, by averaging its  $k$ -ANN in the external database, using  $k = 400$  and  $k = 1000$ . One would hope that such denoised patches would have very low variance, due to averaging out the noise details (reducing the original  $\sigma^2$  by the factor of  $k$ ). But this is not



**Fig. 6.10** Random noise patches have high normalized correlation with natural image patches. Red and Green curves are distributions, showing high normalized correlation values of random noise patches  $n$  with their 400 NNs among an external database of natural image patches (red curve), and 1000 NNs (green curve). In contrast, random noise patches  $n$  have very low normalized correlation with other random noise patches (Blue curve). Shown for  $7 \times 7$  random noise patches  $n$ , for  $\sigma = 55$

the case. Since all nearest neighbors are correlated to  $n$ , their average is also strongly correlated to  $n$ , leaving a non-negligible noise residues:  $\|\text{Avg}(400\text{NNs})\| = 16.46$ ;  $\|\text{Avg}(1000\text{NNs})\| = 15.49$  (where,  $\|n\| = \sigma = 55$ ).

When considering relatively smooth patches, after removing their mean, they behave as random noise. Therefore based on the above, such patches will not be denoised well externally. In contrast, internally, within a local neighborhood of  $20 \times 20$  (where the mean of the patches is similar Fig. 6.8c), denoising of smooth patches is equivalent to averaging 400 random noise patches. The internal local denoising seeks other similar patches with independent random noise. The blue curve in Fig. 6.10 shows the distribution of normalized correlation values of  $7 \times 7$  random noise patches ( $\sigma = 55$ ), with other random noise patches. As expected, the average normalized correlation values are almost zero. The average of such 400 random noise patches yields a patch with zero variance.

The above empirical evaluations hold for smooth patches. Next, we consider a general noisy patch  $p_n = p + n$  and its nearest neighbor  $q = NN(p_n)$ , in an external database of clean patches  $Q$ . In general, the search for  $q$  will be guided both by the signal component  $p$  and by the noise component  $n$  and will minimize

$$NN(p_n) = \arg \min_{q \in Q} \|q - p_n\|^2, \quad (6.11)$$

Further developing the above expression yields:

$$\|q - p_n\|^2 = \|q\|^2 + \|p_n\|^2 - 2\langle p_n, q \rangle = \|q\|^2 + \|p_n\|^2 - 2c\|q\|\|p_n\|, \quad (6.12)$$

where  $c$  denotes the normalized correlation between  $q$  and  $p_n$ .

Differentiating Eq. 6.17 w.r.t.  $\|q\|$  and equating to 0 leads to:

$$\|NN(p_n)\| = \|q_{min}\| = c\|p_n\|. \quad (6.13)$$

Substituting Eq. 6.13 in Eq. 6.17 yields

$$dist_{NN(p_n)} = \|p_n\|^2 - c^2 \|p_n\|^2 = (1 - c^2)(\|p\|^2 + \|n\|^2). \quad (6.14)$$

Notice that  $c = \frac{\langle p_n, NN(p_n) \rangle}{\|p_n\| \|NN(p_n)\|} = \frac{\langle p, NN(p_n) \rangle + \langle n, NN(p_n) \rangle}{\sqrt{\|p\|^2 + \|n\|^2} \|NN(p_n)\|}$ . Hence

$$c = \frac{c_p \|p\| + c_n \|n\|}{\sqrt{\|p\|^2 + \|n\|^2}}, \quad (6.15)$$

where  $c_n$  and  $c_p$  are normalized correlation values between  $NN(p_n)$  and the noise  $n$  and signal  $p$ , respectively. Ideally, if  $c_n = 0$  then  $c_p = 1$  (from substitution of  $c$  in Eq. 6.14). In such case,  $NN(p_n) = p$ .

Let  $dist_p$  denote the distance between the noisy patch  $p_n$  and its clean version  $p$ :

$$dist_p = \|p_n - p\|^2 = \|n\|^2 \quad (6.16)$$

Now, consider relatively smooth patches, for which the  $\|p\|$  is small. In that case, according to Eq. 6.17, the search for nearest neighbor fits the noise,  $n$ , yielding

$$c \cong \frac{c_n \|n\|}{\sqrt{\|p\|^2 + \|n\|^2}}.$$

Plugging this into Eq. 6.14 yields

$$dist_{NN(p_n)} \cong dist_n = \|p\|^2 + (1 - c_n^2) \|n\|^2 \quad (6.17)$$

Therefore, noise overfitting will tend to occur if

$$dist_p > dist_n \iff PatchSNR(p_n) < c_n. \quad (6.18)$$

In other words, when  $PatchSNR(p_n) < c_n$ , an unconstrained external search for similar patches  $q = NN(p_n)$  will tend more toward the noise  $n$  and lead to overfitting the noise details. However, for detailed patches with  $PatchSNR(p_n) \gg c_n$ , the search for  $q$  will tend to fit the signal  $p$ . And since their “signal fitting” is much better externally than internally (Sect. 6.2.3), *External Denoising* is preferable for detailed patches.

### Combining the Power of Internal and External Denoising:

In this section, we discussed  $PatchSNR$ , an inherent characteristic of a noisy patch that determines the success of its Internal/External denoising. We concluded that smooth patches, dominated by the noise, should prefer internal denoising. Such patches have low  $PatchSNR(p_n)$  and are prone to overfit the noise details or noise mean. Internally this risk is lower, while signal fit is sufficiently good when performed locally for smooth patches. In contrast, detailed patches, dominated by the signal,

should prefer external denoising. Externally, such patches have much better signal fit, yet, they do not risk overfitting the noise details, since their  $PatchSNR(p_n)$  is high.

External denoising will inevitably suffer from fitting the noise mean, *regardless of the PatchSNR*. However, for *detailed patches*, this error is substantially smaller than the signal error introduced by their bad signal fit in internal denoising (compare the average signal error in Fig. 6.3 (dotted blue line) to the mean-error in Fig. 6.9). Thus, External versus internal denoising preference of a patch should be determined by the signal versus noise dominance within the patch, captured by its  $PatchSNR$ . Moreover, better denoising may be achieved by combining information from both internal and external patches. This approach may yield higher denoising performance compared to the bounds reported in [18]. Indeed, there has been a substantial amount of research that successfully combined both sources of information [8, 10, 19, 30]. Those works demonstrated considerable improvement of PSNR values, especially when image-specific external datasets were used [19, 30].

## 6.4 The Impact of Deep Learning on Denoising and Their Bounds

A major leap in performance happened when neural networks based denoising algorithms emerged [4, 8, 31], which allowed to achieve nonlinear mappings between the input noisy image and the output clean image. While being trained on big collections of noisy-clean pairs of images, these networks typically rely on larger receptive fields, thus utilizing more global internal information, when appropriate. In this section, we aim to examine how these networks approach previously investigated denoising bounds, and if such networks eliminate the need in combining internal and external denoising.

Over past years several attempts have been made to investigate possible denoising bounds. In [9], Chatterjee and Milanfar formulated a method of calculating the lower bounds on the Mean Squared Error (MSE) that accounts for the strength of the corrupting noise, the number of observations that are typically available to estimate a denoised patch, as well as the variability of the geometric structures in the image. Later Levin and Nadler [17, 18] analyzed denoising limits with respect to ideal external denoising, which is essentially equivalent to averaging a very large number of clean external patches, including adaptive approach that allowed to adapt patch size for better denoising. Finally, in [33] we introduced denoising bound for internal local multi-scale denoising. We empirically showed that for almost any noisy image patch (more than 99% of the patches), there exists a “good” clean version of itself at the same relative image coordinates in some coarser scale of the image (as illustrated in Fig. 6.12). We could therefore calculate the MSE with respect to this “good” patch. This MSE provides empirical lower bound for local multi-scale denoising error. Next

we analyze how neural network based denoising (e.g., [4, 31]) perform with respect to those denoising bounds.

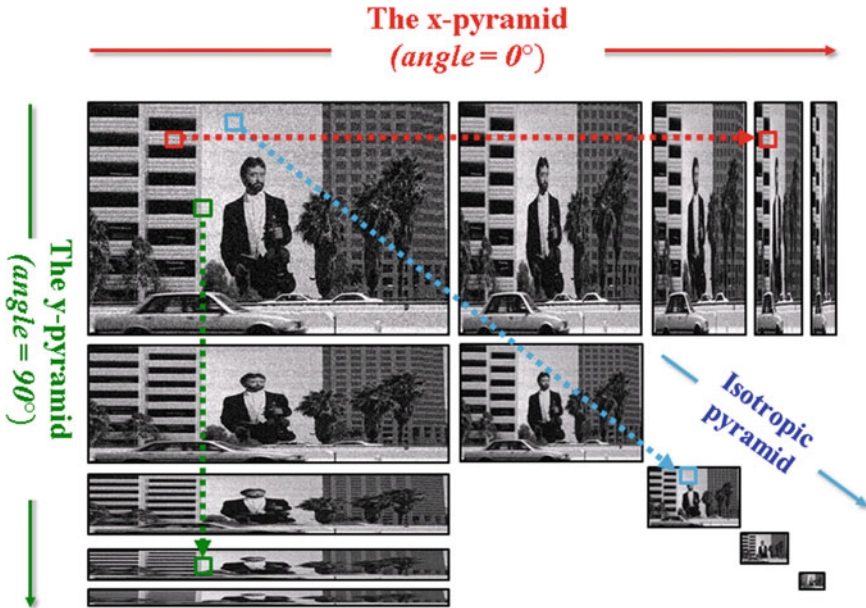
Recent residual learning, CNN-based denoising architectures of [31] and [4] demonstrated state-of-the-art denoising results. DnCNN [31] is based on a relatively simple deep architecture, which relies on stacked convolutional layers, each followed by a ReLU nonlinearity layer and batch normalization. Wavelet domain deep residual learning network (WDnCNN) [4] further improves DnCNN [31] by  $\sim 0.1$ – $0.2$  dB in Gaussian denoising task. The main idea of [4] is that image transform into wavelet domain reduces the topological complexity of data and label manifold and hence improves the network predictive power. (For further details on both architectures, please refer to the original papers [4, 31].)

Both networks considerably outperform previous state-of-the-art algorithms [11, 14, 34]. However, we note that neither DnCNN [31], nor WDnCNN [4], managed to reach the improvement bounds suggested in [9]. Those bounds predict possible improvement (over BM3D [11]) of up to 3dB for images like Lena and House in the case of a Gaussian noise with  $\sigma = 25$ . However, DnCNN [31] reports only 0.4dB and 0.2dB improvement for these images respectively. WDnCNN [4] achieves more considerable improvement of 0.8dB for both images for  $\sigma = 30$ .

On the other hand, DnCNN considerably improves over BM3D performance, when compared over BSD [5]. For example, for  $\sigma = 50$ , the predicted maximal possible improvement for external denoising based on [18] is bounded by 0.7dB. DnCNN [31] exceeds BM3D performance by 0.61dB, which is very close to the optimally possible improvement for external denoising (can also be seen from the closeness of the RMSE of DnCNN depicted in Fig. 6.13b versus RMSE of external ideal denoising depicted in Fig. 6.9b). As such, we conclude that DnCNN [31] reaches the bound of optimal external denoising. Note that WDnCNN [4] closes the gap by introducing a marginal increase of 0.1dB in PSNR over [31] (for BSD and  $\sigma = 50$ ). However, due to the simplicity of [31] and the availability of DnCNN models trained for wider range of noise levels, we further elaborate only on this approach.

At the first glance, denoising based on neural networks may be perceived solely as external denoising, because it relies on training dataset of noisy and clean images for learning the set of its features (neurons). Nevertheless, such denoising also strongly depends on the internal information captured by the receptive field of the network, which is influenced by its depth. The receptive field of the network is equivalent to the effective size of the neighborhood within the image required for denoising of a single pixel. DnCNN [31] adopts a receptive field of  $35 \times 35$  pixels.

Next we compare DnCNN with the local internal denoising Oracle that we suggested in [33]. The comparison is valid, because both DnCNN and the Oracle explore the same multi-scale information of a given region in the noisy image. In fact, the “good” clean patch retrieved by the Oracle, resides in the receptive field of the DnCNN. The DnCNN learns how to combine nonlinear responses to learned kernels at different scales (e.g., 17 scales for denoising with known variance), and as a universal approximator [16], should be able to reconstruct this patch (or “better” patch in a sense of  $L_2$  norm with respect to the original clean patch).

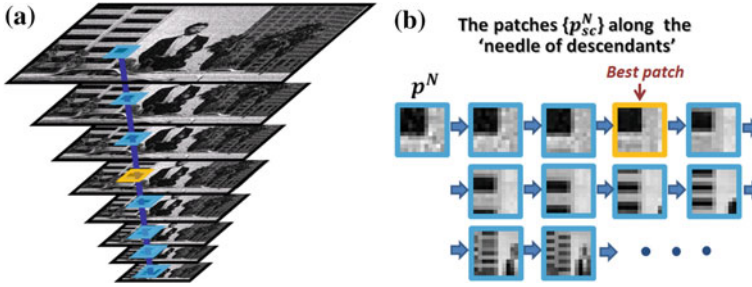


**Fig. 6.11** Pyramids. The 3 pyramids used for Oracle calculation: isotropic pyramid and two directional pyramids in x and y directions

Our multi-scale representation is based on 3 pyramids—traditional isotropic pyramid and two directional pyramids in x and y directions. We generate a *directional pyramid* by blurring and subsampling the image only in one direction. This is different from the commonly used isotropic image pyramid, which preserves the aspect ratio, as well as from the Steerable Pyramid [28], which applies 1D directional filtering, but subsamples the image in both directions. Figure 6.11 graphically illustrates the idea. The “hidden” clean patches, which are obscured by noise at the original input image, emerge in the coarser pyramid scales (see [33] for more details).

The Oracle experiment described below was performed on BSD100 [5] test dataset of natural images (size  $320 \times 480$ ). Gaussian noise with zero mean and variance  $\sigma^2$  was added to each clean image  $I$  (converted to grayscale), resulting in a noisy image  $I_N$ . Three types of pyramids are then generated from  $I_N$ : (i) Isotropic pyramid (blur and subsample<sup>1</sup> both in x and in y), (ii) X-pyramid (blur and subsample only in the x direction), and (iii) Y-pyramid (blur and subsample only in the y direction)—see Fig. 6.11. Each pyramid is a cascade of images  $\{I_{sc}^N\}$  of gradually decreasing scales, generated by scaling down the noisy image  $I_N$  using scale factors of  $sc = 0.9^s$  ( $s = 0, 1, \dots, 17$ ). The smallest scale was  $sc = 0.9^{17} \approx 0.16$  of the original image  $I_N$ .

<sup>1</sup>Using Matlab “imresize” with a bicubic kernel.



**Fig. 6.12** The multi-scale “needle” of patches. Each noisy patch  $p^N$  has a needle. **a** All the patches along the needle of the noisy patch are at the same relative image coordinates. Initially, the patches get better (cleaner), but eventually new structures enter the patch. The “best” representative patch on the needle is marked in orange. **b** Zooming in on the descendant patches  $\{p_{sc}^N\}$  along the needle ( $sc = 1, \dots, 0.1$ ) (Note For simplicity, the illustration here is made for one pyramid only, in practice patches along three needles, corresponding to pyramids from Fig. 6.11, are compared.)

Each *clean*  $5 \times 5$  patch  $p$  from the clean image  $I$  was compared using  $L_2$  norm (mean squared error) against the *noisy*  $5 \times 5$  patches  $p_{sc}^N$  in each of the three *noisy* pyramids, but only along its “needle of descendants”—at the same relative image coordinates in the coarser scales (see Fig. 6.12). Namely, if  $(x, y)$  are the coordinate of the clean  $5 \times 5$  patch  $p$  (and the noisy patch  $p^N$ ), then for each scale  $sc = 0.9^s$  we compare  $p$  only against the  $5 \times 5$  patch  $p_{sc}^N$  whose coordinates are

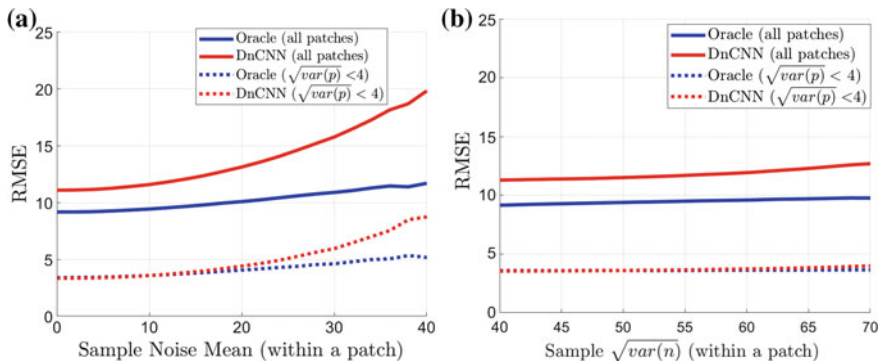
- $(0.9^s x, 0.9^s y)$  in the Isotropic pyramid
- $(0.9^s x, y)$  in the X-pyramid
- $(x, 0.9^s y)$  in the Y-pyramid

Among these descendant patches the Oracle chooses  $\hat{p} = p_{sc}^N = \arg \min_{sc} \|p - p_{sc}^N\|_2^2$  as the “best” representative of the clean patch  $p$  along the needle.

Finally, to obtain a fully denoised image we average the “best” representative overlapping patches (using spatial weighted kernel of  $5 \times 5$ ). The above choice of parameters yields approximately  $35 \times 35$  receptive field (effective image region size) used for the Oracle (the coarsest  $5 \times 5$  patch spans  $31 \times 31$  and is further average with  $5 \times 5$  kernel at the original scale), which compares to the receptive field employed by the DnCNN (whose denoising results are obtained using code provided by authors [31]).

Figure 6.13 analyzes the noise-overfitting of DnCNN. Red curve represents DnCNN error, while blue represents Oracle error (RMSE is calculated on  $7 \times 7$  patches extracted from the Oracle/DnCNN denoised images). Dashed and solid curves represent error of only smooth and all patches, respectively. First, Fig. 6.13 a shows that the error due to the noise mean overfitting has been substantially reduced, compared to Fig. 6.9a. This is not surprising, since a  $35 \times 35$  effective image region is used, which should reduce the minimum error, formulated in Eq. 6.6, to  $55/35 \simeq 1.4$ . Surprisingly however, the average RMSE obtained for smooth patches is considerably higher (around  $\sim 3.9 \simeq 55/14$ ) than the possible minimum, for both





**Fig. 6.13 Fitting of the noise mean:** RMSE of denoising (in case of  $\sigma = 55$ ) versus sample mean and variance of the noise within the patch. The denoising RMSE is computed for multi-scale local Oracle (blue) and DnCNN [31]. Dashed line represents error for smooth patches, solid for all patches. The denoising error grows considerably with the deviation from zero of the sample noise mean within the patch. In contrast, the denoising error is almost independent of the sample noise variance within the patch (Average RMSE values are plotted.)

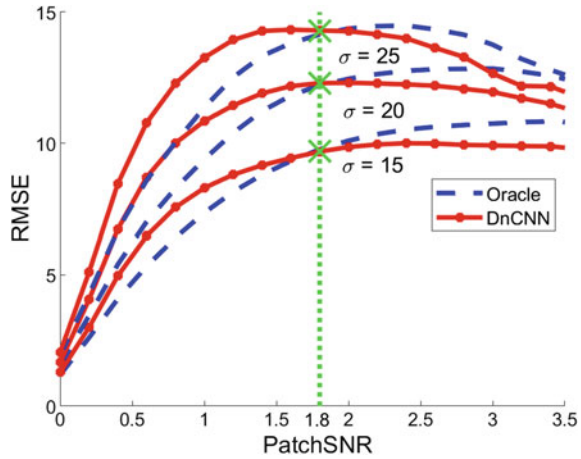
DnCNN and the Oracle. This is also true for other noise levels as can be seen in Fig. 6.14 (see the value of RMSE for  $PatchSNR = 0$ ). This can be explained using the observation of [20] that the effective receptive field of stacked deep CNNs is distributed as a Gaussian. In other words, the effective area in the receptive field only occupies a fraction of the theoretical receptive field, since Gaussian distribution generally decays quickly from the center. Similarly, internal multi-scale Oracle uses bicubic kernel to downsample images and hence not all pixels contribute equally to the patch at the coarsest scale.

Similarly to the ideal denoising Fig. 6.9b, the denoising error of DnCNN in Fig. 6.13b does not depend on sample noise variance within the patch. Since the Oracle maximizes the fit to the original clean signal, it is not affected by the sample noise mean or variance deviations from the expected values. In the case of smooth patches, the RMSE values of DnCNN and the Oracle are almost identical. Therefore, DnCNN performance is not influenced by deviations of the sample variance from the expected variance.

However, the RMSE of all the patches is considerably higher for DnCNN than for the Oracle ( $\sim 2$  dB). Figure 6.14 further illustrates the differences in RMSE of the DnCNN and the Oracle as a function of  $PatchSNR(p_n)$  (for  $7 \times 7$  patches extracted from the denoised images) for various noise levels ( $\sigma = 15, 20, 25$  from bottom to top). As can be seen for  $PatchSNR \leq 1.8$  internal Oracle error is lower, while for  $PatchSNR > 1.8$  DnCNN error is lower. The differences are statistically significant (verified by Wilcoxon rank sum test using Matlab’s “ranksum”). Notably, this threshold does not fit the expected threshold from Fig. 6.6b. This happens because in Sect. 6.2.2 we discussed a linear mapping from noisy to clean patch, while DnCNN minimizes a *nonlinear* mapping  $f(I_N)$  between the noisy image (or patch) and its clean counterpart. Hence, the threshold might differ.



**Fig. 6.14 Denoising error of DnCNN (red) versus multi-scale internal oracle (blue):** RMSE of the denoising is plotted versus  $PatchSNR$  for different noise levels, from  $\sigma = 15$  at the bottom, to  $\sigma = 25$  at the top; The green line and crosses show the threshold of  $PatchSNR$ , below which DnCNN does not manage to outperform the Oracle. (Statistics over 100 images)



While the Oracle denoising is *not* an algorithm, and is linear, it is a good indication of existing information within a certain receptive field. Based on this, we conclude that for patches with relatively low  $PatchSNR$ , the learned mapping  $f(I^N)$  of DnCNN [31] does not manage to predict their corresponding best “clean” patch representative that resides in the receptive field of the network. This behavior is consistent for a wide range of noise levels ( $\sigma \in [15, 50]$ ). However, for higher noise levels, there are not enough patches above the threshold to form substantial statistics, hence those plots are not presented in Fig. 6.14.

Note that while DnCNN performs better than the Oracle on patches with high  $PatchSNR$ , those constitute a smaller portion of the image, especially for higher noise level. As such, the average PSNR denoising score still exhibits a large gap in performance between DnCNN and the Oracle as shown in Table 6.1.

To summarize, the Gaussian denoising performance has been considerably improved by models obtained from convolutional neural networks, such as [4, 31]. In fact, those closed the gap on the denoising bounds of external denoising [18]. However, there is still room for improvement in the case of low  $PatchSNR$  patches. Such patches, which benefit more from strictly internal denoising (and which form a major portion of the image), have not reached their full potential yet, even with deep learning. Perhaps the next challenge is to train an **image-specific CNN** [27]

**Table 6.1** Comparison of PSNR (dB) on BSD100 [5]

$\sigma$	DnCNN [31]	Oracle [33]
15	31.52	32.15
20	30.06	31.13
25	29.02	30.37
35	27.49	29.26
50	26.05	28.05

that will exploit local recurrence of patches, without relying on external examples, as was successfully done for the task of super-resolution [27]. Combining such model with existing external-based models may push PSNR bounds further up and improve denoising by  $\sim 1\text{--}2$  dB, especially for higher noise levels (as can be seen from Table 6.1).

## References

1. Aharon M, Elad M, Bruckstein A (2006) K-SVD: an algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Trans Signal Process* 54(11):4311–4322
2. Aharon M, Elad M, Bruckstein A (2005) Design of dictionaries for sparse representation. In: Workshop on signal processing with adaptative sparse structured representations (SPARS). Rennes, France
3. Alvarez L, Lions PL, Morel JM (2005) Image selective smoothing and edge detection by nonlinear diffusion. *J Numer Analysis* 29:845–866
4. Bae W, Yoo J, Ye JC (2017) Beyond deep residual learning for image restoration: persistent homology-guided manifold simplification. In: Conference on computer vision and pattern recognition (CVPR)
5. Berkely segmentation dataset. <https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/>
6. Buades A, Coll B, Morel J (2005) A non-local algorithm for image denoising. In: IEEE international conference on computer vision and pattern recognition (CVPR)
7. Buades A, Coll B, Morel J (2005) A review of image denoising algorithms, with a new one. *Multiscale Model Simul* 4(2):490–530
8. Burger HC, Schuler CJ, Harmeling S (2013) Learning how to combine internal and external denoising methods. In: German conference on pattern recognition (GCPR13), pp 121–130
9. Chatterjee P, Milanfar P (2010) Is denoising dead? *IEEE Trans Image Process* 19(4):896–911
10. Chen F, Zhang L, Yu H (2015) External patch prior guided internal clustering for image denoising. In: IEEE international conference on computer vision (ICCV)
11. Dabov K, Foi A, Katkovnik V, Egiazarian K (2007) Image denoising by sparse 3D transform-domain collaborative filtering. *IEEE Trans Image Process* 16(8)
12. Elad M, Aharon M (2006) Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Trans Image Process* 54(12):3736–3745
13. Elad M, Aharon M (2006) Image denoising via learned dictionaries and sparse representations. In: IEEE conference on computer vision and pattern recognition (CVPR)
14. Harmeling S, Schuler CJ, Burger, HC (2002) Image denoising: can plain neural networks compete with BM3D? In: IEEE conference on computer vision and pattern recognition (CVPR), pp 2392–2399
15. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Conference on computer vision and pattern recognition (CVPR)
16. Hornik K, Stinchcombe M, White H (1989) Multilayer feedforward networks are universal approximators. *Neural Netw* 2(5)
17. Levin A, Nadler B (2011) Natural image denoising: optimality and inherent bounds. In: IEEE conference on computer vision and pattern recognition (CVPR)
18. Levin A, Nadler B, Durand F, Freeman WT (2012) Patch complexity, finite pixel correlations and optimal denoising. In: European conference on computer vision (ECCV)
19. Luo E, Chan SH, Nguyen TQ (2016) Adaptive image denoising by mixture adaptation. *IEEE Trans Image Process* 25(10)
20. Luo W, Li Y, Urtasun R, Zemel R (2016) Understanding the effective receptive field in deep convolutional neural networks. In: Advances in neural information processing systems

21. Mairal J, Elad M, Sapiro G (2008) Sparse representation for color image restoration. *IEEE Trans Image Process* 17(1):53–69
22. Mairal J, Bach F, Ponce J, Sapiro G, Zisserman A (2009) Non-local sparse models for image restoration (ICCV)
23. Mosseri I, Zontak M, Irani M (2013) Combining the power of internal and external denoising. In: *IEEE international conference on computational photography (ICCP)*
24. Parzen E (1962) On estimation of a probability density function and mode. *Ann Math Stat* 33:1065–1076
25. Perona P, Malik J (1990) Scale space and edge detection using anisotropic diffusion. *IEEE Trans Patt Anal Mach Intell* 12:629–639
26. Portilla J, Strela V, Wainwright M, Simoncelli E (2003) Image denoising using scale mixtures of Gaussians in the wavelet domain. *IEEE Trans Image Process* 12(11)
27. Shocher A, Cohen N, Irani M (2018) “Zero-Shot” super-resolution using deep internal learning. In: *Conference on computer vision and pattern recognition (CVPR)*
28. Simoncelli E, Freeman W (1995) The steerable pyramid: a flexible architecture for multi-scale derivative computation. In: *International conference on image processing (ICIP)*, vol 3, pp 444–447
29. Xie J, Chen E (2012) Image denoising and inpainting with deep neural networks. In: *Conference on neural information processing systems (NIPS)*
30. Yue H, Sun X, Yang J, Wu F (2015) Image denoising by exploring external and internal correlations. *IEEE Trans Image Process* 24(6)
31. Zhang K, Zuo W, Chen Y, Meng D, Zhang L (2017) Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Trans Image Process* 26(7):3142–3155
32. Zontak M, Irani M (2011) The internal statistics of a single natural image. In: *IEEE conference on computer vision and pattern recognition (CVPR)*
33. Zontak M, Mosseri I, Irani M (2013) Separating signal from noise using patch recurrence across scales. In: *IEEE Conference on computer vision and pattern recognition (CVPR)*
34. Zoran D, Weiss Y (2011) From learning models of natural image patches to whole image restoration. In: *IEEE international conference on computer vision (ICCV)*

# Chapter 7

## Patch-Based Methods for Video Denoising



A. Buades and J. L. Lisani

**Abstract** Video denoising is an important and open problem, which is less treated than the single-image case. Most image sequence denoising techniques rely on still image denoising algorithms; however, it is possible to take advantage of the redundant information contained in the sequence to improve the denoising results. Most recent algorithms are patch based. These methods have two clearly differentiated steps: select similar patches to a reference one and estimate a noise-free version from this group. We review selection and estimation strategies. In particular, we show that the performance is improved by introducing motion compensation. We use as example a recent video denoising technique inspired by fusion algorithms that use motion compensation by regularized optical flow methods, which permits robust patch comparison in a spatiotemporal volume. The use of principal component analysis ensures the correct preservation of fine texture and details, provided that the noise is Gaussian and white, with known variance. Video acquired by any video camera or mobile phone undergoes several processings from the sensor to the final output. This processing, including at least demosaicking, white balance, gamma correction, filtering, and compression, makes a white noise model unrealistic. Indeed, real video captured in dark environments has a very poor quality, with severe spatially and temporally correlated noise. We discuss a denoising framework including realistic noise estimation, multiscale processing, variance stabilization, and white noise removal algorithms. We illustrate the performance of such a chain with real dark and compressed movie sequences.

---

The authors were supported by grant TIN2014-53772R and TIN2017-85572-P.

---

A. Buades (✉) · J. L. Lisani  
DMI - IAC3, Universitat Illes Balears, Ctra Valldemossa km 7.5, Palma, Spain  
e-mail: toni.buades@uib.es

J. L. Lisani  
e-mail: joseluis.lisani@uib.es

## 7.1 Introduction

This chapter deals with the denoising of real image sequences and videos. By real video, we mean any image sequence captured with a video camera or mobile phone. Image sequences obtained as a consecutive acquisition of photographs with a camera might be also processed by the introduced algorithms in this chapter. We use the term “real”, to differentiate from the removal of uniform white noise which is the objective of most literature dealing with image or video denoising.

The uniform white noise model is not an adequate model for the denoising of photography and video, for which the most simplistic model should take into account at least that the noise is signal dependent. This might be a valid model for RAW data, the one acquired at the sensor. A signal-dependent model assumes that the noise value at a certain pixel depends on its true noise-free intensity. This is related to the Poisson process which models the photon counting at the sensor.

The denoising algorithms might be always applied as soon as possible in any image or video chain, before any processing on color, contrast, or compression. When noise is signal dependent but still uncorrelated at different pixel locations, a variance stabilization transform permits the use of any white uniform noise removal algorithm. This justifies the fact that most literature concentrates on this simplistic model. Most algorithms dealing with real noise actually rely on a white uniform noise removal method. This is then a good starting point and will be the object of the first part of current chapter. Let us denote the image sequence by  $I(x, y, t)$ , with  $(x, y)$  the spatial coordinates and  $t$  the temporal component, then this model assumes that

$$I(x, y, t) = I_0(x, y, t) + n(x, y, t), \quad (7.1)$$

where  $I_0$  is the true image sequence and  $n(x, y, t)$  is the noise i.i.d. realizations of a Gaussian variable of zero mean and standard deviation  $\sigma$ .

We will review and analyze the main video denoising algorithms. Recent methods are patch based, meaning that the minimal unity which will be denoised is a patch. The same procedure is shared by most state-of-the-art algorithms: for a reference patch, similar ones are selected across the sequence and a noise-free patch is estimated from these groups.

We will discuss the selection and estimation strategies used by state of the art. We will show that even if nowadays most literature concentrates on the estimation part, a drastic improvement might be obtained by accurately selecting similar patches. These algorithms are more robust to object motion than classical filters relying only on single-pixel comparisons. Patch-based algorithms do not need to compensate the research area with a motion estimation, since patch comparison implicitly adapts to it. However, we will show that the motion compensation might be effective when used for defining spatiotemporal patch distances.

We will pay special attention to state-of-the-art algorithm [16]. It is based on more general type of algorithms, whose aim is not only to remove noise but also

to improve image resolution as well as sharpness. This type of methods, known as image fusion, aims at improving the general image quality of an image by fusing it with other images of the scene.

In the second part of this chapter, we will deal with the denoising of any video acquired by a video camera or mobile phone. This part is inspired by the noise clinic [40], which is able to denoise any single image. The use of an image sequence permits the estimation of a more accurate noise amplitude, as well as, a more performant denoising. Such method will include the use of Laplacian pyramids, the estimation of signal-dependent noise amplitude per scale, the use of variance stabilization transforms, and state-of-the-art video white noise removal [16].

This chapter is organized as follows: In Sect. 7.2, we briefly review image denoising literature. In Sect. 7.3, we introduce white noise video removal algorithms and discuss in detail their design and performance. In Sect. 7.4, this white noise removal method is embedded in a framework aimed at denoising real video sequences. The last section of this chapter summarizes the conclusions of our experiments.

## 7.2 Image Denoising

Since most of the image sequence noise removal techniques rely on image denoising ones, we briefly review the literature for this case.

### 7.2.1 *White Noise Removal*

Techniques for noise removal in digital images comprise transform thresholding, local averaging, patch-based methods, and variational techniques. The sliding window DCT [60] and wavelet thresholding [25] are the main examples of the thresholding methods. These methods decompose the original data in a predefined basis and attenuate or cancel coefficients under a certain threshold related to noise statistics. Anisotropic filtering [2] and bilateral filtering [53, 55] or neighborhood filtering [61] aim at averaging close pixels belonging to the same object, thus reducing the noise amplitude and preserving the main object boundaries. Variational techniques share the same objective but use a variational framework, the total variation minimization being the main example [51].

NL-means [11] introduced patch-based methods into image denoising. The algorithm groups similar patches all over the image and averages them in order to reduce noise. The method is able to preserve texture and fine details additionally to the main boundaries of the image. The performance of patch-based methods has been drastically improved by the combination with transform thresholding as originally proposed in BM3D [22]. BM3D combined patch-based grouping with a threshold in a 3D DCT transform. Several methods appeared combining the grouping of similar patches and the learning of an adapted basis via PCA or SVD decomposition

[47, 64]. State-of-the-art results are obtained using Gaussian models for the group of similar patches (NL-Bayes [38]) or adapting the shape of patch before learning a PCA model [22].

Gaussian mixture [56] and Sparse representation [28] achieve a similar grouping plus adapted transform via the learning of a collection of orthogonal basis. When this set of basis is learnt from the image itself, an implicit clustering of image patches permits to learn adapted models for different geometrical configurations.

State-of-the-art denoising methods use additional techniques which increase noise reduction and detail preservation (see [39] for a review). These techniques, which were used by sliding DCT classical algorithms [60], were introduced into nonlocal patch-based ones in [22]. First, the whole patch is denoised and not only the central pixel, which permits an increase in the noise reduction by taking the average of all estimates per pixel (aggregation). Second, the denoised image is used as guide (“Oracle”) for a second iteration. The similarity between two patches is computed in the first denoised image, and the transformed coefficients are used to drive the thresholding in the second iteration.

For color images, most state-of-the-art methods [22, 38] prefer to use a chromatic decorrelating transform. The YUV or YCrCb color spaces compute a gray component  $Y$  as the average of RGB values, while the chromatic components encode the difference of the red and blue channels with  $Y$ . The image  $Y$  containing the geometry of the image is actually less noisy than each of the RGB channels, since it is computed as the average of them. Patch grouping is carried out by computing the distance in the  $Y$  channel, while each component is denoised independently. The smoothness of  $U$  and  $V$  permits a larger noise reduction than the one achieved in the RGB space. However, the use of chromatic separation tends to excessively attenuate the image colors, resulting in grayish images. For that reason, many methods prefer to use the original RGB through a vectorial form of the algorithm, as for example in [11].

Finally, let us mention that the patch-based algorithms have also been adapted to other kinds of noise than additive uniform white noise. For example, in [24], the authors adapted the patch distance to deal with different kinds of noise, including the Poisson model. Instead of modifying the algorithm formulation, a stabilization transform can be applied, permitting the use of the original denoising algorithms [46]. These models still suppose the noise values at different pixels to be independent. This assumption is valid for RAW images, the data acquired at the sensor but not for the final color images delivered by the camera.

There is a scarce literature dealing with the denoising of real images, having signal-dependent and spatially correlated noise. In order to design such methods, noise estimation plays an important role since a signal and even frequency-dependent model has to be estimated.

In [42], the authors proposed a denoising framework for JPEG images, automatically estimating a signal-dependent noise amplitude. Such a procedure needs to make assumptions on the imaging chain applied by the camera in order to correctly estimate the noise. In [40], the authors proposed a full denoising chain applying multiscale techniques and NL-Bayes [38]. This chain uses the noise estimation algorithms developed in [19, 20]. These methods are able to estimate a noise model dependent not only on color but also on frequency.

## 7.3 White Noise Removal in Video Sequences

In this section, we introduce the techniques used by state-of-the-art video denoising algorithms, we analyze in detail how and why they work, and we discuss their performance. A brief review of the literature is presented in Sect. 7.3.1. The main stages of a patch-based method are: (i) the selection of similar patches and (ii) the estimation of a noise-free patch from them. We will analyze different patch selection strategies in Sect. 7.3.2, paying attention to motion estimation, which plays a key role in this process. Estimation methods will be discussed in Sect. 7.3.3. Finally, a brief comparison among state-of-the-art algorithms will be presented in Sect. 7.3.4.

### 7.3.1 Introduction

Local average methods, as the bilateral filter [55], or patch-based methods as NL-means [11], or BM3D [22], and NL-Bayes [38] can be easily adapted to video just by extending the neighboring area to the adjacent frames. Simply, similar pixels or patches are searched for in previous and posterior frames. The same estimation techniques used for single image denoising might be used in this case, for example, VBM3D [21] extended collaborative filtering to denoising. The increase of similar patches due to the use of more frames improves the noise reduction capabilities of the methods. However, it might also increase the probability of selecting incorrect ones.

When sample selection is not robust enough, for example, in classical neighborhood filters or bilateral filters, their performance is improved by introducing motion compensation [37, 48]. These modified filters estimate explicitly the motion of the sequence and compensate the neighborhoods where similar pixels are searched for. It was shown in [12] that a compensation of the search areas is not necessary if a patch-based algorithm is used for denoising. The selection of the most similar patches across adjacent frames actually adapts to motion and selects similar patches wherever in the sequence. This is equivalent to the estimation of motion through a block matching algorithm. However, motion estimation by block matching loses accuracy as noise standard deviation increases.

Several other strategies have been proposed to incorporate motion into image sequence denoising rather than simply compensating the search areas. In [8], the NL-means was extended to video by growing adaptively the spatiotemporal neighborhood. In [59], the temporal filtering was separated from the spatial one using NL-means, and then both were combined using a motion indicator. In [62], the authors made use of an external database for still image denoising. The most similar images were retrieved from an external database and used for patch-based denoising, which was then combined with an internal denoising stage inspired by BM3D. In [45], a multiscale sparse representation was learnt for video restoration.

Possibly, the most successful algorithms for video denoising are the VBM4D [44] and state-of-the-art algorithm [16]. VBM4D exploits the mutual similarity between



3D spatiotemporal volumes constructed by tracking blocks along trajectories defined by the motion vectors. Mutually similar volumes are grouped together by stacking them along an additional fourth dimension, thus producing a 4D structure. Collaborative filtering is achieved by transforming each group through a decorrelating 4D separable transform, and then by shrinkage and inverse transformation. Motion vectors are computed by block matching.

State-of-the-art algorithm [16] is inspired by image fusion algorithms. These algorithms aim at the restoration of a single image via its combination with other images of the same scene [31, 34]. Image fusion is not directly interested in the removal of noise but in a more general restoration of the image, that is, deblurring, increase of detail, or even of resolution. The key of these approaches is the use of a global registration, more robust to noise, blur, and color or compression artifacts and, additionally, providing subpixel accuracy. These global registration techniques usually rely on feature matching, for example, SIFT [43], and on a parametric registration, either using an affinity or a homography. The viewfinder alignment [1] performs such a registration by an affine function, with the important characteristic of being extremely fast. The general approach is the use of a homography [17, 29, 31, 34, 54]. It must be noted that a homography is valid only for planar scenes or if the optical center is not modified; in other cases, optical flow techniques might be preferred.

The algorithm in [16] computes the optical flow between each frame and adjacent ones in a temporal neighborhood. If registration was accurate and the sequence free of occlusions, a temporal average in this aligned data would be optimal, even if the noise reduction would slowly decrease as  $1/M$ , where  $M$  is the number of adjacent frames involved in the process. Generally, this will not be the case; inaccuracies and errors in the computed flow and the presence of occlusions make this temporal average likely to blur the sequence and introduce artifacts near occlusions. These difficulties are compensated by a 3D spatiotemporal patch selection after warping and adapted model learning. Even if a 3D structure is used for the selection procedure, still 2D patches are used for the estimation of noise-free values. In [4], the authors proposed the estimation of noise-free 3D patches, containing pixels from contiguous frames. The use of 3D patches makes the estimation of an adapted model more complex and is more affected by object occlusions.

### 7.3.2 Patch Selection Criteria

While most interest in image or video denoising is dedicated to the estimation of the denoised patch from a set of similar ones, we show in this section that the most drastic improvement is achieved by accurately choosing the group of similar patches. The adaptation of the search area or the decoupling of spatial and temporal estimation has shown to be effective to discard very different patches in moving regions but has no impact in the accurate selection of candidate patches (Figs. 7.1 and 7.2).

Accurate selection of similar patches is achieved via spatiotemporal distances with motion compensation. When processing single image, we are forced to increase the



**Fig. 7.1** Central frame of the original sequences used in our tests. From left to right and from top to bottom: girls, army, truck, statB, taxi, iseq, bicycle, bus, tennis, sales. The last four sequences are composed of 30 frames, while the rest are composed of eight frames



**Fig. 7.2** Central frame of the original color sequences used in our tests. From left to right and from top to bottom: army, cooper, dog, truck. All the sequences are composed of eight frames

size of the patch when dealing with severe noise, which at the same time limits the number of similar patches found. The use of spatiotemporal (3D) patches allows comparison with many more pixels, thus being more robust without the need of increasing the 2D support.

A block matching method might be a valid motion estimation algorithm if images need not to be resampled. Block matching methods for motion estimation provide an integer precision displacement unless the image is previously zoomed in, which is time-consuming. Optical flow methods will be preferred if frames are motion compensated via the application of the flow. Another advantage of variational techniques is the presence of a regularization term. When the noise standard deviation increases, the precision of block matching algorithms quickly deteriorates. The regularization term of optical flow techniques permits a more accurate estimate in smooth parts of the flow.

In order to describe spatiotemporal patches, we need to define for each  $n \times n$  patch  $P$  of a reference frame  $I_k$ , the patch  $\mathcal{P}$  referring to its extension to the temporal dimension, having  $M$  times more pixels than the original one (assuming  $M$  patches in the temporal neighborhood,  $M = 2t + 1$ ),  $\mathcal{P} = (P_{k-t}, \dots, P_{k+t})$ . If the sequence is not resampled, the spatiotemporal patch distance is computed compensating the 2D patches taking into account the flow. In this case, the motion field  $(u_i, v_i)$  indicates which are the coordinates of the patch in the corresponding frame (integer precision),  $P_i = I_i(P_{(x+u_i, y+v_i)})$ . This is the case of VBM4D which uses block matching to select trajectories. If the set of adjacent frames to  $I_k$  has been previously warped (i.e., resampled) with the computed optical flow, denoted by  $\{I_{k-t}^W, \dots, I_{k+t}^W\}$ , the

spatiotemporal patch is composed of a temporal tube, where each 2D patch is located at the same pixel coordinates,  $P_i = I_i^W(P_{(x,y)})$ . This is the case of the state of the art [16], which computes the flow using the method in [63].

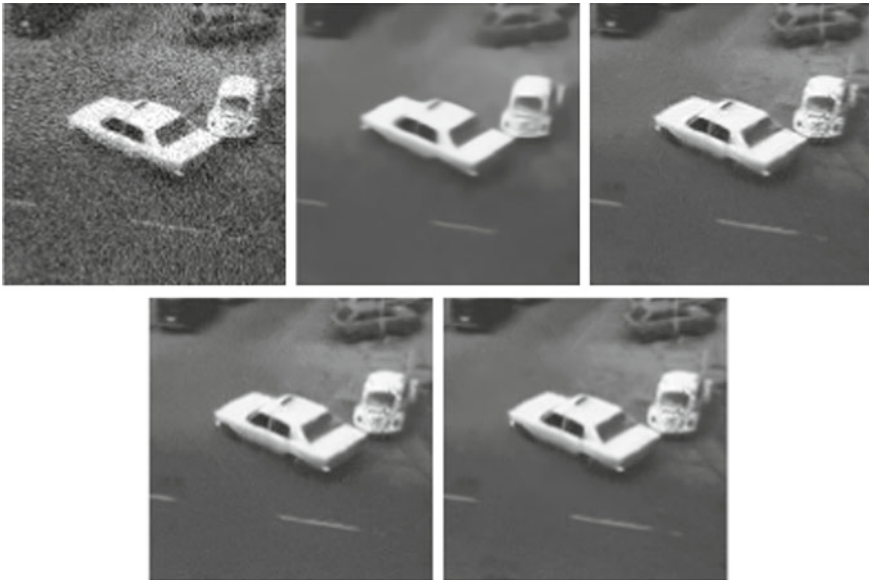
Then, a spatiotemporal patch selection looks for the  $K$  extended patches closest to  $\mathcal{P}$ . These extended patches are centered at frame  $I_k$  and the distance is written as

$$d(\mathcal{P}, \mathcal{Q}) = \sum_{i \in \{k-t, \dots, k+t\}} \|P_i - Q_i\|^2.$$

As each extended patch contains  $M$  2D image patches, the group contains  $K \cdot M$  selected 2D patches of size  $n \times n$ .

The algorithm in [16] actually uses an additional mask  $M_j$ , the occlusion mask between frames  $I_k$  and  $I_j$ ,  $j \in \{k-t, \dots, k+t\}$  to select the candidate patches. The adopted optical flow does not involve occlusion detection in the functional. Occlusions are detected a posteriori following the same approach in [52].

Figure 7.3 compares the criteria for selecting the similar patches. The options are (i) simple extension of the search area to neighboring frames, (ii) spatiotemporal distance with block compensation by block matching without resampling (integer



**Fig. 7.3** Discussion on motion estimation and resampling with noise standard deviation 20. From left to right and from top to bottom: noisy image and denoising results (see text for details) using different selection criteria: (i) simple extension of the search area to neighboring frames, (ii) spatiotemporal distance with block compensation by block matching without resampling, (iii) spatiotemporal distance with block compensation after warping (i.e., resampling) of neighboring frames with computed optical flow, and (iv) spatiotemporal distance with computed optical flow. In all the cases, the denoising of the selected patches is achieved by thresholding in an adapted basis. The RMSE values are, respectively, 6.80, 4.92, 5.05, and 4.46

precision), (iii) spatiotemporal distance with block compensation by optical flow without resampling (integer precision), and (iv) spatiotemporal distance after warping (i.e., resampling) of neighboring frames with computed optical flow. The figure illustrates the different degrees of texture preservation by all selection methods. The use of optical flow or block matching gives a similar result on the denoised image, provided that the block used for block matching estimation is large enough. The use of the warped neighboring frames in defining the distance seems to be crucial in order to get the best results. The errors on the denoised central frame of the sequence shown in Fig. 7.3 are 6.80, 4.92, 5.05, and 4.46, for (i), (ii), (iii), and (iv), respectively. In all the cases, the denoising of the selected patches is achieved by thresholding in an adapted basis. In order to warp the images, an accurate estimation of the optical flow is needed. A short review of optical flow methods is provided in the next section.

### 7.3.2.1 Optical Flow Estimation and Occlusion Detection

Optical flow constraint (OFC)-based methods suppose that each pixel has the same color during the whole trajectory, or at least in adjacent frames. That is, they suppose that

$$I(x, y, t) = I(x + u(x, y, t), y + v(x, y, t), t + \Delta t),$$

where  $u$  and  $v$  are the displacement vectors at time  $t$  and pixel  $(x, y)$ . This equation alone is unable to determine the flow. The uncertainty is solved by adding a spatial or spatiotemporal regularization term.

The optical flow constraint can be linearized into the well-known equation,  $I_x u + I_y v + I_t = 0$ , and methods differ on how this constraint and the regularization term are imposed. It is usually written,

$$\int_{\Omega} \psi(I_x u + I_y v + I_t) + \lambda \int_{\Omega} \phi(\nabla u, \nabla v),$$

where the definition of  $\psi$  and  $\phi$  might vary. The classical Horn–Schunck [32] method used  $\psi(s) = s^2$  and  $\phi(\nabla u, \nabla v) = |\nabla u|^2 + |\nabla v|^2$ . It is well known that the square function excessively regularizes the discontinuities of the flow and is not robust to outliers and occlusions. Robust functions and anisotropic regularization replaced this classical approach, see for instance Weickert and Alvarez [3].

Brox et al. [9] introduced a different linearization of the OFC and a warping strategy in order to minimize the functional

$$\int_{\Omega} \psi(I_0(x, y) - I_1(x + u(x, y), y + v(x, y))) + \lambda \int_{\Omega} \phi(\nabla u, \nabla v)$$

with  $\psi$  and  $\phi$  functions robust to occlusions and outliers. This approach permits the introduction of additional constraints [10] on the displacement of several points.

These displacements might be obtained by key point matching. The inclusion of these constraints improves the ability for capturing long-range motions.

Pock et al. [63] introduced total variation minimization into the flow computation. The flow between two images  $I_0$  and  $I_1$  is obtained by minimizing

$$\int_{\Omega} |I_0(x, y) - I_1(x + u(x, y), y + v(x, y))| + \lambda \int_{\Omega} (|\nabla u| + |\nabla v|), \quad (7.2)$$

where  $u$  and  $v$  being the desired flow. The total variation term is minimized via the Chambolle dual algorithm [18].

One of the major drawbacks of these approaches is the failure of the color constancy hypothesis, for which a constancy of the gradient [9] or the Laplacian [49] might be additionally imposed. Recently, Wedel et al. [57] proposed a method to decompose the sequence into a cartoon and a texture part and use only the texture part for estimating the flow.

Occlusion detection can be directly taken into account by modifying the functional. The modified functional might cancel the OFC for occluded pixels and add a term penalizing the number of occluded pixels, see for example [5]. This additional term needs the setting of a new parameter equivalent to fixing the percentage of the image being occluded, which is unknown a priori. Since the occluded points mostly coincide with points of negative divergence, Caselles et al. [6] introduced an additional term with the divergence of the flow as an occlusion indicator.

Occlusion identification in [16] depends on the divergence of the computed flow, the color difference check after flow compensation, and a forward–backward flow test. Negative divergence, a large color difference or forward–backward flow disagreement, indicates occlusions or at least failure of the color constancy assumption. All criteria are combined, for a pixel  $\mathbf{x} = (x, y)$  and the computed flow  $\mathbf{u} = (u, v)$  between  $I_0$  and  $I_1$ , into the weighting function

$$w(x, y) = e^{-\frac{|I_0(x) - I_1(x + \mathbf{u}(x))|^2}{\sigma_i^2}} \cdot e^{-\frac{\min(\operatorname{div} \mathbf{u}, 0)^2}{\sigma_d^2}}, \quad (7.3)$$

where  $\sigma_d$  is fixed while  $\sigma_i$  depends on the noise standard deviation. This weight function is binarized, providing a mask of occluded points. If forward and backward flows disagree, this weight is directly set to zero.

### 7.3.3 Patch Denoising

#### 7.3.3.1 Denoised Estimate Computation

Once similar patches have been selected, denoising strategies are analogous to the still image case. If the patches have been carefully selected, an average weighted by the 2D-patch distance might give already reasonable results comparable to much

more complex algorithms. Once the  $K \cdot M$  patches have been selected, such modified NL-means based algorithm (NL-means 3D) would compute

$$\tilde{P} = \sum_{i \in \{1, \dots, KM\}} w_i Q_i, \quad (7.4)$$

where  $w_i = \frac{1}{C_P} e^{-\frac{\|P-Q_i\|^2}{h^2}}$  and  $C_P = \sum_i e^{-\frac{\|P-Q_i\|^2}{h^2}}$ . A modified version of this spatiotemporal NL-means has been recently proposed for video super-resolution showing state-of-the-art results [13].

For single image, weighted average is improved by collaborative filtering [21] or local Bayes modeling [38]. The same techniques are adapted for video denoising. The VBM4D applies collaborative filtering via DCT [44], and SPTWO [16] learns a local Gaussian model via PCA. We describe this latter as local modeling.

The PCA analysis of the set of  $M \cdot K$  patches looks for the basis of  $R^{n^2}$  better explaining its structure in the sense that most of the information describing all patches is concentrated in a few vectors of the basis. The amount of information that each vector of the basis conveys is coded in its  $n^2$  principal values. The computation of the PCA of a set of patches is equivalent to the singular value decomposition (SVD) of the matrix  $X$  having  $M \cdot K$  rows and  $n^2$  columns with each selected patch in a different row,

$$X = U \Sigma V^T,$$

where  $U \Sigma$  are the coefficients in the new basis,  $\Sigma$  is a diagonal matrix containing the square root of the principal values, and each column of  $V$  contains a principal vector that is an element of the new basis. That is, by keeping only the coefficients associated with the most important vectors (the ones with highest corresponding principal value), we keep the maximum of information. By discarding coefficients related to less important vectors, we remove noise as proposed in [64]. The decision of canceling a coefficient of a certain patch is not taken depending on its magnitude but on the magnitude of the associated principal value. A more robust thresholding is obtained by comparing the principal values to the noise standard deviation and canceling or maintaining the coefficients of all the patches associated with a certain principal direction. The denoised set of patches can be computed as

$$\tilde{X} = F U \Sigma V^T,$$

where  $F$  is a  $n^2 \times n^2$  diagonal matrix such that  $F_{ii} = 1$  if  $\Sigma_{ii} > \tau \sigma$  and zero otherwise. The whole patch is restored in order to obtain the final estimate by aggregation.

### 7.3.3.2 Second ‘‘Oracle’’ Iteration

A second iteration of the algorithm is performed using the ‘‘Oracle’’ strategy. Once the whole sequence has been restored, the algorithm is reapplied on the initial noisy

sequence, but the motion estimation and patch selection are performed on the result of the first iteration. We detail the procedure for [16].

Let  $\{I_{k-t}^W, \dots, I_{k+t}^W\}$ , and  $\{I_{k-t}^{0W}, \dots, I_{k+t}^{0W}\}$  be the warped noisy and initially restored images in a temporal neighborhood of  $I_k$ , where the optical flow has been computed using initially restored images  $I_k^0$  and  $I_j^0$ . For each patch  $P$  of the reference frame  $I_k$ , we consider the extended patches  $\mathcal{P}$  and  $\mathcal{P}^0$  referring to the extension to the temporal dimension of the patch and its counterpart in the already denoised sequence. The  $K$  extended patches that will be selected as similar are the ones minimizing the distance

$$d(\mathcal{P}^0, \mathcal{P}) = \sum_{i \in \{k-t, \dots, k+t\}} \|P_i^0 - Q_i^0\|^2.$$

Now we have two different sets containing each one  $K \cdot M$  2D patches of size  $n \times n$ . One set is formed by the patches of the noisy sequence, and the other one by the corresponding patches of the already denoised sequence.

The PCA is computed in the set of already denoised patches. Let  $X$  denote the matrix containing the selected patches of the noisy sequence as rows and  $X^0$  the corresponding matrix with the same patches of the already filtered sequence. We compute the basis associated with  $X^0$  making use of the SVD,

$$X^0 = U^0 \Sigma^0 V^{0T}.$$

This basis is adapted to the already denoised patches which are noise free. The coefficients of the noisy patches are computed in this new basis and modified by a Wiener filter before reconstruction. This is written as

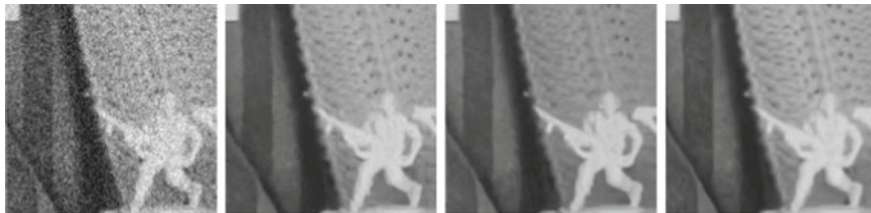
$$\tilde{X} = F(XV^0)V^{0T},$$

where  $XV^0$  are the coefficients of the noisy patches in the computed PCA basis and  $F$  is now a diagonal matrix implementing an optimal Wiener filter, that is,

$$F_{ii} = \frac{\Sigma_{ii}^2}{\Sigma_{ii}^2 + \tau^2 \sigma^2}.$$

In Fig. 7.4, the different estimation methods are compared using the same patch selection criteria. For a reference frame to be denoised, the neighboring frames are warped using the estimated motion and spatiotemporal blocks are used for selecting similar patches. We compare the use of weighted averaging, the local modeling via PCA with a single iteration, and with the Oracle iteration. Clearly, the texture is better preserved by using the Oracle scheme. However, the result by weighted averaging gives a high-quality denoised sequence, corroborating the importance of careful preselection. The error for the NL-means for the weighted average (NL-means 3D) is 4.87, for the local estimate via PCA is 4.46, and the two iteration with Oracle is 4.13.





**Fig. 7.4** Discussion on the denoising procedure for the selected patches. From left to right: noisy image ( $\sigma = 20$ ), denoising result using NL-means applied on spatiotemporal blocks (NL-means 3D), denoising with single iteration and two iteration denoising with Oracle. In both cases, similar spatiotemporal blocks were obtained after motion estimation with optical flow and resampling. The RMSE values for each denoised result were 4.87, 4.46, and 4.13, respectively

### 7.3.3.3 Color Processing

As for the single-image case, two different options are possible: First, the extension of the gray algorithm to RGB via a vectorial form of the method, or second, the use of a decorrelating transform. The application of the algorithm to each channel is usually discarded since it does not take advantage of the interchannel correlation and might yield to color artifacts.

The use of a decorrelating transform has the advantage of computing patch distances in the gray component of the image which is less noisy. However, the independent denoising of the chromatic components has the risk of obtaining excessively muted colors. For a weighted average, both the uses of a decorrelating transform or the vectorial form permit to take into account interchannel correlation. For collaborative filtering, a decorrelating transform might be used.

When selecting a local model, the use of a vector form permits to adapt to the local geometry of the patches and also to compute a color transform for each group of similar patches, thus increasing the effectiveness of the model. Each color patch is considered as a vector with three times more components than in the single-channel case. This is the option chosen in [16].

### 7.3.4 Brief Comparison

In this section, we illustrate the performance of the different methods obtained combining the selection and estimation strategies above described. Several of these options are actually proposed video denoising methods.

We take into account simple extension of the search area to neighboring frames, the classical NL-means [12] applying a weighted average, and the VBM3D [21] using collaborative filtering. We use the VBM4D [44] as representative of methods using spatiotemporal distance with block compensation by block matching without resampling (integer precision). We also compare spatiotemporal distance with



block compensation by optical flow without resampling (integer precision), denoted as SPTWO without warping. Finally, we use the SPTWO [16] and the NL-means 3D (Eq. 7.4) as representatives of spatiotemporal distance after warping (i.e., resampling) of neighboring frames with computed optical flow. This permits to assess the improvement due to the gradation from classical NL-means to SPTWO. The Matlab implementations of VBM3D and VBM4D were obtained from the author's website, and the default parameters were used in the tests. We used our own implementation for the rest of methods, applying the same parameters for all sequences.

A Gaussian noise with increasing levels of standard deviation ( $\sigma \in \{10, 20, 30, 40, 50\}$ ) was added to the sequences in Figs. 7.1 and 7.2, which were denoised using the compared methods. The root mean squared error (RMSE) with respect to the original (noise free) sequences is displayed in Table 7.1 for the gray dataset and in Table 7.2 for the colored one. The values in the table correspond to the RMSE computed for the central frame of each sequence. Moreover, the average RMSE for each method and each noise level is also displayed in the last column.

As expected, we observe in these RMSE values the same gradation of algorithms above stated. The largest RMSE values are obtained by those algorithms using a simple weighted average or not applying any type of motion compensation. It is interesting to note how the RMSE of NL-means is improved significantly by the use of motion estimation and warping. Similarly, the VBM3D applying collaborative filtering outperforms simple averaging. Unexpectedly, for gray sequences, the performance of VBM3D is slightly better than VBM4D for low levels of noise (below  $\sigma = 40$ ) and for color sequences, the performance of VBM3D is better than that of VBM4D, even for high levels of noise. Finally, the tables also show that the use of warped patches greatly improves the performance of SPTWO algorithm, for which the smaller RMSE values are obtained.

Figures 7.5, 7.6, and 7.7 display the denoising results of the compared algorithms. The figures also display the difference of the denoised image with the noisy one and the difference of the denoised image with the original one. The difference with the noisy image displays the noise removed by each algorithm. The absence of noticeable details in the removed noise should indicate the preservation of all texture and features of the original image. However, as illustrated by the denoised images, this absence of details in the removed noise does not guarantee that all meaningful information of the original image has been kept. Indeed, the removed structure might be hidden by the noise. For this reason, we also display the image error, actually containing removed information from the original image.

A first visual inspection illustrates that VBM3D and VBM4D perform similarly and that their main weakness is the excessive blurring of image details. Not only texture but also geometry may be removed by these approaches. This can be observed in the letters of the book in Fig. 7.6. The SPWO algorithm recovers better all the image details in all three cases, even in the presence of strong noise.

**Table 7.1** RMSE results. The values correspond to the RMSE computed for the central frame of each sequence. The average RMSE for each method and each noise level is displayed in the last column

	girls	army	truck	statBc	taxi	iseq	bus	tennis	sales	bicycle	average
$\sigma = 10$											
NL-means	4.33	4.02	3.95	4.02	4.02	3.62	5.46	4.84	4.09	4.53	4.28
NL-means 3D	3.71	3.28	3.39	3.19	3.18	3.18	4.53	7.07	4.00	4.33	3.98
VBM3D	3.86	3.49	<b>3.07</b>	3.11	3.01	2.89	5.73	4.75	3.92	3.78	3.76
VBM4D	3.66	3.33	3.28	3.16	3.21	2.80	5.62	4.83	3.93	3.88	3.77
SPTWO no warp	3.60	3.29	3.11	3.13	3.02	2.55	4.51	5.37	<b>3.58</b>	<b>3.66</b>	3.58
SPTWO	<b>3.46</b>	<b>2.99</b>	3.14	<b>3.03</b>	<b>2.98</b>	<b>2.09</b>	<b>4.01</b>	<b>4.70</b>	3.87	3.71	<b>3.40</b>
$\sigma = 20$											
NL-means	7.68	7.35	7.28	7.60	7.48	6.97	9.47	10.31	7.83	7.96	7.99
NL-means 3D	5.49	4.87	5.37	4.65	4.78	5.68	7.34	11.01	5.85	6.44	6.14
VBM3D	5.48	4.93	<b>4.63</b>	4.22	4.26	4.36	9.03	<b>7.42</b>	5.57	5.84	5.57
VBM4D	5.31	4.87	5.05	4.67	4.67	4.26	8.87	7.89	6.01	5.90	5.75
SPTWO no warp	5.15	4.79	4.80	4.41	4.51	4.13	7.10	8.12	<b>5.32</b>	5.86	5.41
SPTWO	<b>4.75</b>	<b>4.13</b>	4.71	<b>4.09</b>	<b>4.15</b>	<b>3.10</b>	<b>6.23</b>	7.53	5.74	<b>5.70</b>	<b>5.01</b>
$\sigma = 30$											
NL-means	11.10	10.79	10.56	11.19	11.04	10.50	13.28	14.25	11.89	11.59	11.61
NL-means 3D	7.22	6.39	7.13	6.14	6.41	8.19	9.86	12.59	7.43	8.35	7.97
VBM3D	6.78	6.06	<b>5.96</b>	5.39	5.43	5.71	11.41	10.88	7.27	7.54	7.24
VBM4D	6.58	6.02	6.41	5.89	5.93	5.58	11.25	11.84	7.84	7.63	7.50
SPTWO no warp	6.56	6.12	6.30	5.75	6.08	5.78	9.27	<b>10.46</b>	<b>6.77</b>	7.64	7.07
SPTWO	<b>5.74</b>	<b>5.06</b>	<b>5.96</b>	<b>5.01</b>	<b>5.21</b>	<b>4.07</b>	<b>8.02</b>	10.78	7.23	<b>6.96</b>	<b>6.40</b>

(continued)

Table 7.1 (continued)

	girls	army	truck	statBc	taxi	iseq	bus	tennis	sales	bicycle	average
$\sigma = 40$											
NL-means	14.54	14.22	13.84	14.74	14.38	13.89	16.96	17.73	15.83	15.27	15.14
NL-means 3D	8.96	7.86	8.80	7.65	7.84	10.43	12.48	13.69	8.93	10.24	9.68
VBM3D	7.91	6.90	7.18	6.50	6.41	6.70	13.25	13.45	8.82	9.10	8.62
VBM4D	7.75	6.85	7.62	6.94	6.89	6.52	13.11	13.96	9.33	9.26	8.82
SPTWO no warp	8.02	7.36	7.76	7.23	7.56	7.13	11.09	<b>12.04</b>	<b>8.14</b>	9.42	8.57
SPTWO	<b>6.74</b>	<b>5.86</b>	<b>7.14</b>	<b>5.92</b>	<b>6.02</b>	<b>4.87</b>	<b>9.57</b>	13.49	8.44	<b>8.28</b>	<b>7.63</b>
$\sigma = 50$											
NL-means	17.93	17.66	17.06	18.21	17.89	17.30	20.81	20.85	19.82	18.87	18.64
NL-means 3D	10.64	9.34	10.40	9.21	9.44	12.00	14.93	14.48	10.46	11.91	11.28
VBM3D	9.23	7.88	8.69	7.61	7.63	7.63	15.37	14.87	10.73	11.67	10.13
VBM4D	8.75	7.62	8.77	7.81	7.81	7.33	14.81	14.92	10.77	10.65	9.92
SPTWO no warp	9.31	8.68	9.21	8.71	9.06	8.21	13.11	<b>13.21</b>	<b>9.42</b>	11.12	10.00
SPTWO	<b>7.60</b>	<b>6.63</b>	<b>8.32</b>	<b>6.85</b>	<b>6.99</b>	<b>5.68</b>	<b>11.14</b>	14.41	9.67	<b>9.41</b>	<b>8.67</b>

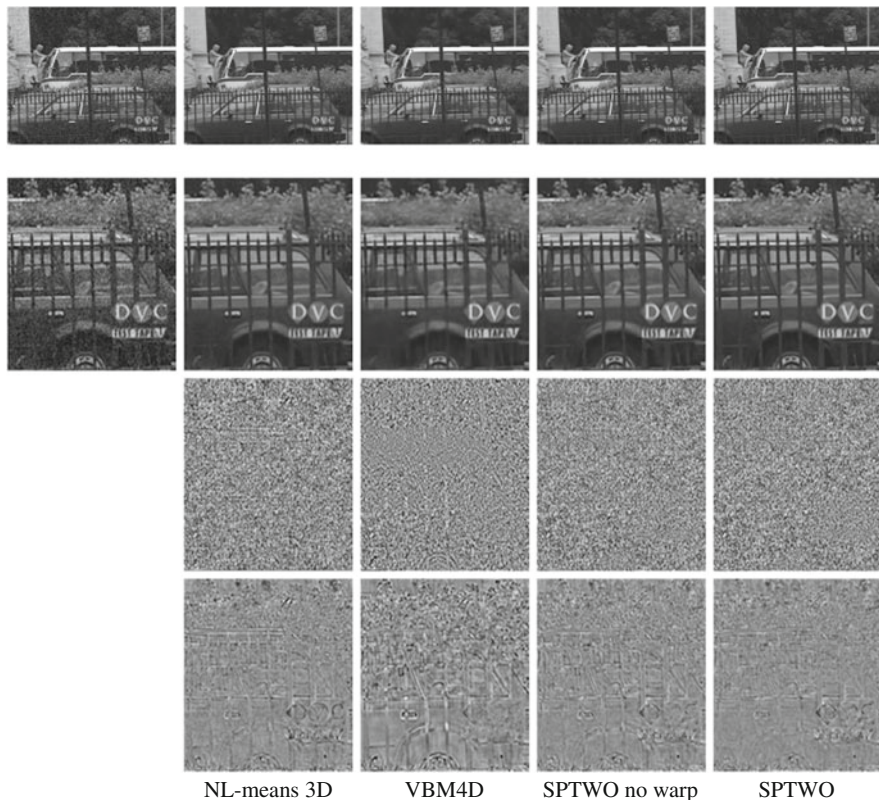
**Table 7.2** RMSE results for color sequences. The values correspond to the RMSE (averaged over the three channels) computed for the central frame of each sequence. The average RMSE for each method and each noise level is displayed in the last column

		army	cooper	dog	truck	average
$\sigma = 10$	NL-means	3.52	4.55	5.05	4.20	4.33
	NL-means 3D	3.47	5.15	5.38	4.46	4.61
	VBM3D	2.96	4.31	4.48	3.76	3.88
	VBM4D	3.26	4.48	4.47	4.04	4.06
	SPTWO no warp	2.94	4.08	4.02	3.61	3.66
	SPTWO	<b>2.72</b>	<b>4.00</b>	<b>4.09</b>	<b>3.56</b>	<b>3.59</b>
$\sigma = 20$	NL-means	5.35	7.71	7.27	6.87	6.80
	NL-means 3D	4.48	8.38	7.12	6.81	6.69
	VBM3D	4.42	6.84	6.37	5.70	5.83
	VBM4D	5.02	7.15	6.46	6.32	6.24
	SPTWO no warp	4.16	6.21	5.62	5.33	5.33
	SPTWO	<b>3.83</b>	<b>6.16</b>	<b>5.75</b>	<b>5.30</b>	<b>5.26</b>
$\sigma = 30$	NL-means	7.07	10.44	9.02	9.09	8.90
	NL-means 3D	5.30	10.54	8.21	8.59	8.16
	VBM3D	5.54	8.87	7.75	7.30	7.37
	VBM4D	6.40	9.25	8.00	8.12	7.94
	SPTWO no warp	5.05	7.87	6.79	6.69	6.60
	SPTWO	<b>4.71</b>	<b>7.84</b>	<b>6.95</b>	<b>6.70</b>	<b>6.55</b>
$\sigma = 40$	NL-means	8.79	12.66	10.72	11.19	10.84
	NL-means 3D	6.04	11.92	9.18	10.09	9.30
	VBM3D	6.40	10.49	8.76	8.63	8.57
	VBM4D	7.62	10.99	9.25	9.67	9.38
	SPTWO no warp	5.86	9.23	7.75	7.88	7.68
	SPTWO	<b>5.55</b>	<b>9.21</b>	<b>7.93</b>	<b>7.89</b>	<b>7.64</b>
$\sigma = 50$	NL-means	10.56	14.58	12.47	13.08	12.67
	NL-means 3D	6.82	12.98	10.08	11.24	10.28
	VBM3D	7.34	12.01	9.90	9.92	9.79
	VBM4D	8.71	12.49	10.55	11.01	10.69
	SPTWO no warp	6.54	10.35	8.68	8.83	8.60
	SPTWO	<b>6.41</b>	<b>10.41</b>	<b>8.95</b>	<b>8.91</b>	<b>8.67</b>

## 7.4 Real Noise Removal

In real video sequences (such as the ones displayed in Fig. 7.8), the results of a white noise removal algorithm are far from optimal. One reason is that in real scenes the uniform white noise model does not hold.

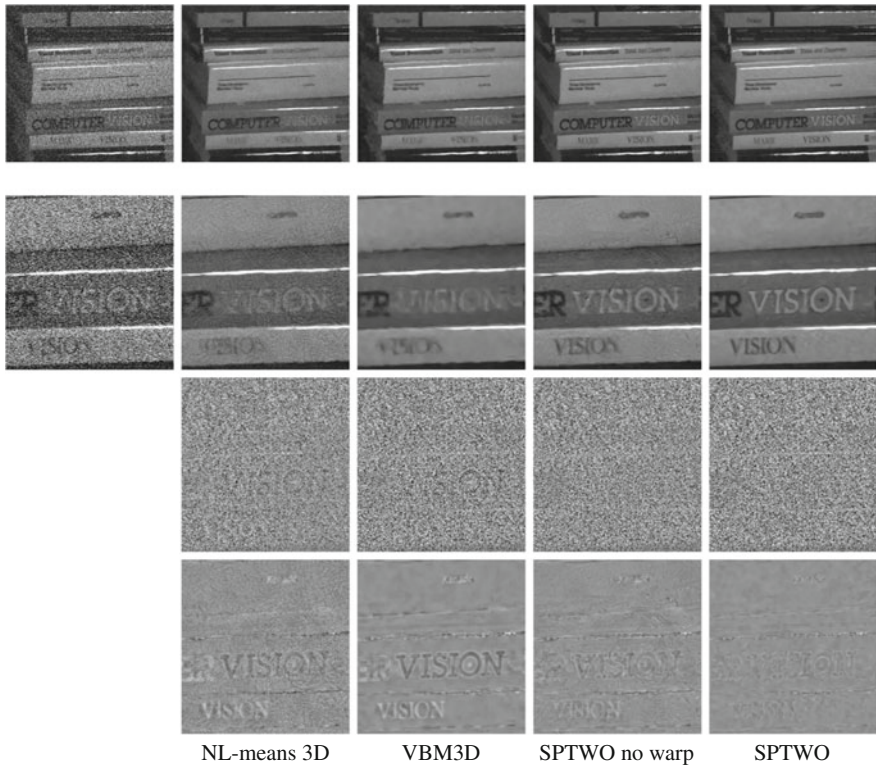
The camera sensors capture at each pixel only the red, the green, or the blue value, while the other two have to be interpolated. This process, called demosaicking, is usually performed by copying and averaging close values from the same channel or



**Fig. 7.5** Denoising results for the *gbus* sequence with  $\sigma = 20$ . First row, from left to right: noisy frame, NL-means 3D (RMSE = 7.34), VBM4D (RMSE = 8.87), SPTWO without warping (RMSE = 7.10), and SPTWO (RMSE = 6.23). Second row: detail of the above images. Third row: differences with respect to the noisy frame. Fourth row: differences with respect to the original (noise free) frame

the other two. As a result, the noise, being almost white at the sensor, gets correlated. The rest of the imaging chain, consisting mainly of color and gamma correction, enhances the noise in dark parts of the image leading to colored spots of several pixels. The size of these spots depends on the applied demosaicking method. Additionally, standard video compression algorithms, consisting of both transferring information from neighboring frames and DCT coding of frame differences, make the noise frequency and temporal dependent. Thus, noise estimation plays a key role in real video denoising, as described in Sect. 7.4.2.

We discuss in detail in Sect. 7.4.3 how a video denoising chain should be designed to deal with any type of noise. A signal-dependent model is estimated at each level of a multiscale pyramid and for each color. In order to apply standard video denoising algorithms, a variance stabilization transform is applied at each scale using the estimated noise amplitude values. We compare the use of such a chain with the SPTWO



**Fig. 7.6** Denoising results for the *iseq* sequence with  $\sigma = 50$ . First row, from left to right: noisy frame, NL-means 3D (RMSE = 12.00), VBM3D (RMSE = 7.63), SPTWO without warping (RMSE = 8.21), and SPTWO (RMSE = 5.68). Second row: detail of the above images. Third row: differences with respect to the noisy frame. Fourth row: differences with respect to the original (noise free) frame

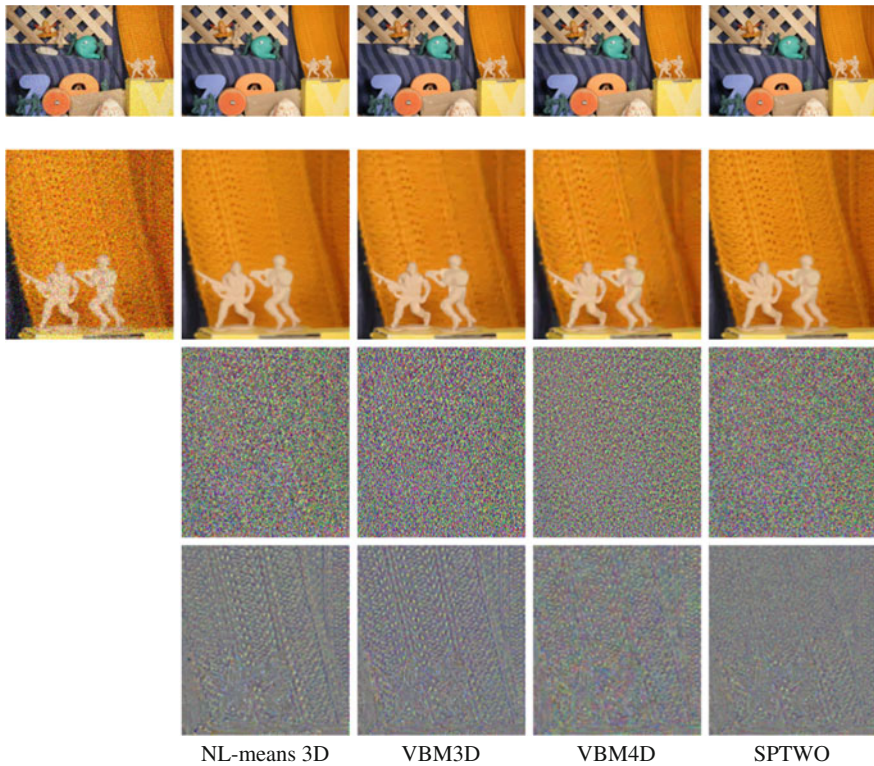
white noise removal algorithm, and other denoising algorithms designed to deal with real noise, Sect. 7.4.4.

### 7.4.1 Introduction

The bibliography on noise removal from real video is scarce. As in the case of single image denoising, most papers concentrate on uniform white noise removal. In addition, the literature is very heterogeneous, what makes rather difficult to establish a classification of the published methods.

Most of the recent publications describe patch-based approaches. In [41], a motion compensated strategy using NL-means algorithm was proposed. In [59], the authors apply the NL-means denoising algorithm to each image and then combine this estimate with a purely temporal application of the same algorithm. Both estimates are





**Fig. 7.7** Denoising results for the *army* color sequence with  $\sigma = 30$ . First row, from left to right: noisy frame, NL-means 3D (RMSE = 5.30), VBM3D (RMSE = 5.54), VBM4D (RMSE = 6.40), and SPTWO (RMSE = 4.71). Second row: detail of the above images. Third row: differences with respect to the noisy frame. Fourth row: differences with respect to the original (noise free) frame



**Fig. 7.8** Some frames of real (non-simulated) noisy image sequences

combined in static parts of the image, while the spatial estimate is preferred in moving regions. In [41], motion vectors are used and patches are grouped across adjacent frames but in a different manner. Instead of comparing patches to the reference patch, these are compared in each frame with the compensated patch of the reference one. NL-means is applied to this group of collected patches. In [30], the authors perform a bilateral filter in both luminance and chrominance (YCbCr) for each frame, for which the bilateral weight distribution is computed only on the luminance. A multiscale wavelet transform permits to deal with non-white noise.

Many methods combine noise removal with color enhancement. For example, [59] applies a second iteration of the denoising algorithm after a contrast enhancement stage. A similar approach is used in [36] but a purely temporal denoising algorithm is applied first, and a second purely spatial after the enhancement stage. In [7], spatiotemporal bilateral filtering is applied and combined with the enhancement technique in [27]. Filtering parameters at each pixel are fixed depending on the amount of enhancement to be applied.

More general methods perform, additionally to noise removal, color correction and sharpness improvement. In [35], the authors simultaneously perform multiview image sequence denoising, color correction, and sharpness improvement in slightly defocused regions for sequences of images coming from different cameras and thus with different degrees of blur and noise. In [23], a sequence of images acquired with a hand-held mobile phone are combined. The camera shake blurs each image with a different kernel, thus attenuating for each one a different part of the Fourier spectrum. The authors proposed to combine the different Fourier spectrums, choosing for each frequency the images with larger magnitude.

### 7.4.2 *Temporal Estimation for Signal-Dependent Noise*

For RAW images, which display the data acquired at the sensor, a realistic assumption for the noise model is to be signal dependent [19]. The noise characteristics are modified by the camera imaging chain and the compression. In [42], the authors studied the noise amplitude of JPEG images, and they estimated a model depending on the demosaicking algorithm and the applied color corrections. In [20], the authors also studied the noise amplitude in JPEG images but proposed a model dependent not only on color but also on frequency. However, these models do not take into account that for video a multitemporal estimation is possible.

Patch-based methods are the most used for estimating white noise amplitude in digital images. They compute the standard deviation or DCT coefficients of all the patches in the image [50]. However, these methods are not adequate for estimating non-white noise amplitude. Indeed, neighboring pixels have correlated noise values, which makes any statistical measure to systematically underestimate the true amplitude. While single-image algorithms are forced to use these measures, when dealing with image sequences, we may use values from different frames. Pixels belonging to different frames of the sequence have independent noise values, which are more ade-

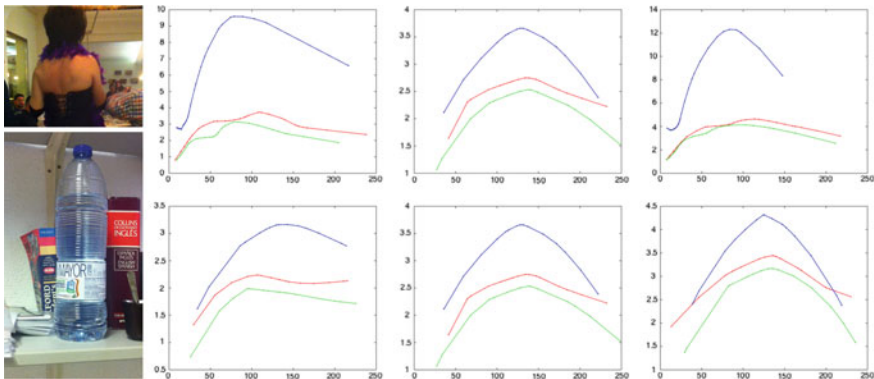


quate for estimating the noise amplitude in the non-white case. It must be noted that this is less true for highly compressed videos, since standard compression algorithms actually combine values of consecutive frames.

The method proposed in [14] adapts the method in [50] by selecting patches of neighboring frames. This solution is suboptimal since even if it combines information from different frames, it does not take into account that the noise spatial correlation can be mitigated by temporal averaging, which is the essence of fusion algorithms. The method proposed in [15] first estimates the optical flow and warps all images into a common reference frame. Using the motion compensated data, the temporal average and standard deviation are computed for all the pixels. The only assumption needed for the application of such estimator is that the noise has zero mean and that the noise values from different frames are independent.

The averages obtained for all the pixels might be classified into a disjoint union of variable intervals or bins, in such a way that each interval contains a large enough number of elements. That is, the color level range is not divided into uniform length intervals, but these intervals are adapted to the image sequence itself. This way, if the image is dark, most of the intervals will be of short length and belonging to dark values while none or very few bins will be in the lighter part of the color range. These measurements allow for the construction of a list of standard deviations whose corresponding averages belong to the given bin. The median of the standard deviations in each bin gives an estimation of the noise amplitude per bin.

This noise estimation algorithm might be applied at each level of a multiscale pyramid. Figure 7.9 displays the noise curves estimated for some real video sequences and three scales, the original fine scale and two more. In order to compute these



**Fig. 7.9** Display of noise curves for different real image sequences and three scales (from coarser (left) to finer (right)). The algorithm is applied at each scale obtained from the previous one by subsampling of factor two, obtained by agglomeration of four pixels. The noise pairs  $(u_i, \sigma(u_i))$  obtained by the algorithm proposed in Sect. 7.4.2 are displayed for each channel and linearly interpolated

curves, we have used a pyramid with a subsampling factor of two at each scale, obtained by simple averaging of four pixels.

### 7.4.3 Real Video Denoising Chain

We detail the video processing chain proposed in [14]. This is an adaptation of the single-image chain in [40] to video. The method decomposes the image sequence in a multiscale pyramid and, at each level, the noise is estimated, the variance stabilized, and the noise removed by the white noise removal algorithm [16]. Once the lowest scale is denoised, the result is upsampled and the details are added back. The procedure is repeated until the finest scale is attained.

#### 7.4.3.1 Multiscale Strategy

The multiscale denoising strategy consists of downsampling the image frames at several scales and then denoising the coarsest scales (the ones satisfying more closely the white noise assumption). Then, image details at finer scales are added to each upsampled denoised frame and the process is iterated. As a result of this process, noisy spots are removed from the sequence. Downsampling at each scale is performed by averaging groups of four pixels (factor 2 downsampling) while upsampling is done using cubic splines interpolation.

Figure 7.10 compares the application of the method with a single scale or with three scales. The denoised images are afterward enhanced (with the tone mapping algorithm described in [26]) in order to better illustrate the differences between the two methods. The single-scale algorithm actually removes noise but only noisy regions of slightly more than one pixel, not large spots. Large colored spots are removed only by the multiscale algorithm, which removes them at the scale for which they become almost white noise



**Fig. 7.10** Comparison of the application of the method proposed in [14] with a single scale or with three scales. From left to right: original noisy image, single-scale denoising, and three-scale denoising. The displayed denoised images have been enhanced [26] in order to better illustrate the differences between the two methods. The single-scale algorithm actually removes noise but only noisy regions of slightly more than one pixel, not large spots. Large colored spots are removed only by the multiscale algorithm, which removes them at the scale for which they become almost white noise

removed only by the multiscale algorithm, which removes them at the scale for which they become almost white noise.

### 7.4.3.2 Noise Standard Deviation Stabilization

Uniform white noise removal algorithms fail to denoise a real video sequence due to the fact that, in general, the noise level depends on the intensity level of the images. Using the estimated noise curve, the original video sequence can be manipulated in order to achieve a uniform noise distribution with arbitrary noise level. This manipulation is known as the Anscombe transform (described in the next paragraph) and it is an invertible transform. It is therefore possible to apply any white noise removal algorithm to denoise the transformed sequence and to obtain the final denoised video using the inverse Anscombe transform.

We usually refer to the Anscombe transform as to the transformation  $f(u) = 2\sqrt{u + \frac{3}{8}}$ , which is known to stabilize the variance of a Poisson noise model. However, any signal-dependent additive noise can be stabilized by a simple transform. Let  $v = u + g(u)n$  be the noisy signal, we search a function  $f$  such that  $f(v)$  has uniform standard deviation. When the noise is small compared to the signal, we can apply the decomposition  $f(v) = f(u) + f'(u)g(u)n$ . Forcing the noise term to be constant,  $f'(u)g(u) = c$ , and integrating, we obtain

$$f(u) = \int_0^u \frac{c dt}{g(t)}.$$

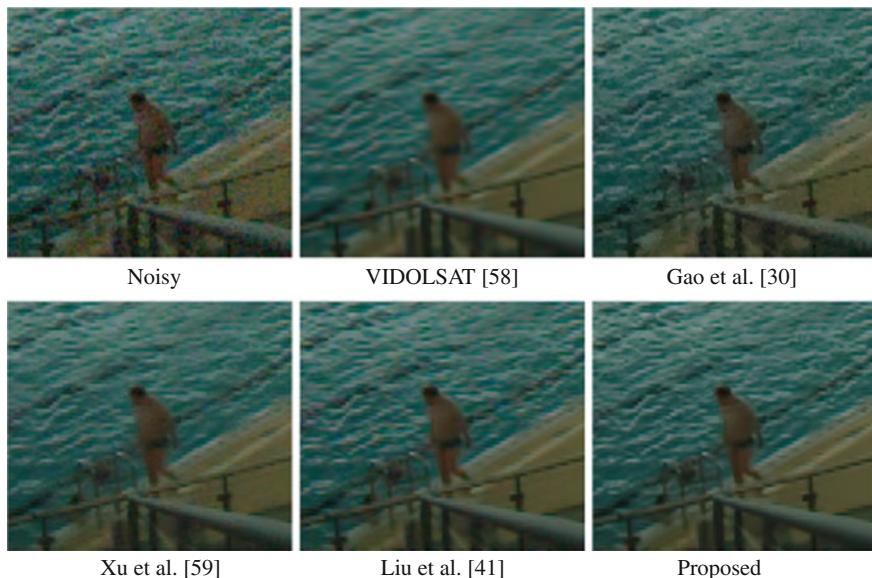
When a linear variance noise model is taken, this transformation gives back the known Anscombe transform. When the noise dominates the signal, as in very dark scenes, the linear approximation is less valid, and more complex techniques have to be used [46].

### 7.4.4 Experimentation and Comparison

In this section, we illustrate the performance of the proposed chain, comparing its results to those of the following state-of-the-art algorithms: Xu et al. [59], Liu [41], Ji et al. [33], Gao et al. [30], and VIDOLSAT [58].<sup>1</sup>

---

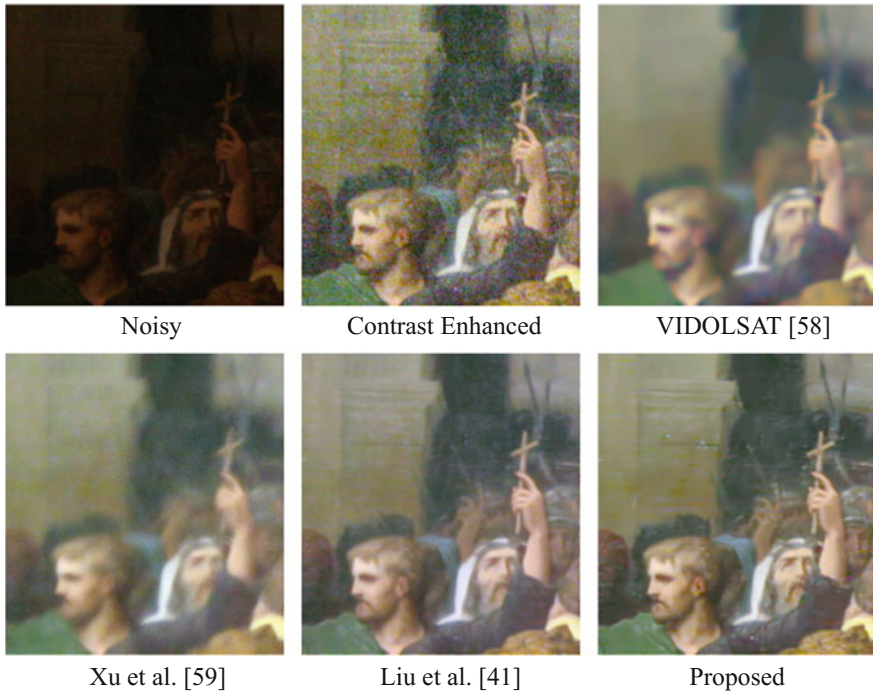
<sup>1</sup>We used the Matlab implementations of Ji et al. (by Sibin, Yuhong, and Yu, 2013), and VIDOLSAT (from <http://www.ifp.illinois.edu/~yoram>). The rest of algorithms were implemented following the descriptions in the published papers. Remark that Gao et al. method is an extension to video sequences of the method described in [65].



**Fig. 7.11** Detail of the denoising results for video sequence pool. From top to bottom and left to right, we display the central frame of the sequences: noisy, results of VIDOLSAT [58], Gao et al. [30], Xu et al. [59], Liu et al. [41], and the proposed method. Remark that our method keeps more details of the pool’s water while removing low-frequency color noise

All experiments use exactly the same parameters which are mainly the number of scales, the window size, and the filtering parameter. We used three scales, a  $5 \times 5$  window and the same filtering parameters in all cases. The denoising stage is applied after the variance stabilization transform, for which the standard deviation curve is estimated automatically. Remark that all the compared denoising algorithms, except the current chain and the one in [30], require as input the standard deviation  $\sigma$  of the image noise (which is assumed Gaussian in most cases). Since this value is unknown, we tested with several values of  $\sigma$  (ranging from 10 to 40) and we display the best result.

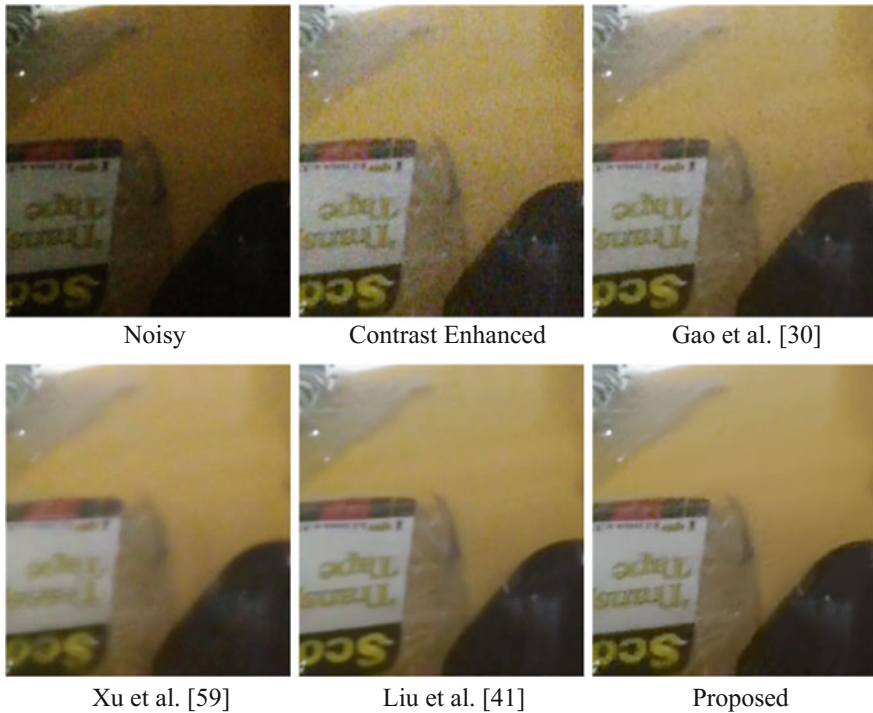
Figures 7.11, 7.12, and 7.13 compare the different methods when applied to the real data. We used several types of image sequences for completeness of the comparison. In Fig. 7.11, we display an example extracted from [15], where the noisy video has been simulated from original RAW data. In Fig. 7.12, we apply all methods to a sequence of images taken independently with a photographic camera. Each image is compressed by the camera independently of the rest, using the JPEG standard. Contrast enhancement [26] is applied after denoising for a better visualization of the differences, but it is not part of the proposed method. That is, the initial noisy sequence is denoised and afterward the result is contrast-enhanced for visualization.



**Fig. 7.12** Detail of the denoised central frame for a sequence of single image taken with a camera. Images were compressed independently since data is not a video. Top, left, and center: excerpt from the central frame of the original sequence, same detail after contrast enhancement applying method [26]. Top (third and second images) and bottom: excerpt from the central frame of the denoised sequences by VIDOLSAT [58], Xu et al. [59], Liu et al. [41], and the proposed method. Contrast enhancement [26] is applied after denoising for a better visualization of the differences, but it is not part of the proposed method. That is, the initial noisy sequence is denoised and afterward the result is contrast-enhanced

As it can be noticed from the enhanced noisy image, noise is quite severe for this case. The rest of methods blur the image in order to remove the grain noise of several pixels. These grains are smoothed creating an unnatural aspect of the image. Our multiscale algorithm removes completely the noise by keeping the sharpness of details of the scene and yielding a natural image.

In Fig. 7.13, we apply the method to a video sequence acquired with a mobile iPhone 4S. Again, contrast enhancement [26] is applied after denoising for a better visualization of the differences, but it is not part of the proposed method. Low-frequency and colored noise is completely removed by the proposed method, while for the rest, residual noise can be noticed in the denoised frames.



**Fig. 7.13** Detail of the denoised central frame from a video acquired with a mobile iPhone 4S. Top, left, and center: excerpt from the central frame of the original sequence, same detail after contrast enhancement applying method [26]. Top (rightmost image) and bottom: excerpt from the central frame of the denoised sequences by Gao et al. [30], Xu et al. [59], Liu et al. [41], and the proposed method. Contrast enhancement [26] is applied after denoising for a better visualization of the differences, but it is not part of the proposed method. This figure illustrates how details, as the text in this case, are better preserved by our method

## 7.5 Conclusion

We presented in this chapter state-of-the-art techniques for noise removal in video. Most recent techniques are patch based. We showed that a drastic improvement can be obtained by carefully selecting the set of similar patches. Such strategies involve the use of optical flow and motion compensation via the resampling of video frames. Standard techniques are used for these stages, not taking into account that videos might be noisy, and dealing with occlusions a posteriori. A refinement of used optical flow and resampling techniques paying attention to occlusions, would improve described state-of-the-art algorithms.

We analyzed estimation techniques, from classical averaging to more complex Bayesian modeling. We did not take into account Gaussian mixtures, but without any doubt, the large amount of samples available in an image sequence makes this



type of data more suitable for the use of Gaussian mixture techniques than single image.

The video processing chain described in the second part of the chapter has shown to be able to correctly denoise the real video from both video and mobile phone cameras. However, the noise estimation procedure as well as the denoising chain neglects compression information. Video standard techniques already estimate motion and transfer image patches from one frame to the neighboring ones. Such information could be useful for the denoising process. Frame differences after compensation are compressed by standard DCT techniques, for which quantization intervals are known. A more complex formulation could include such information as already proposed for image denoising.

Image and video denoising techniques should be applied at the first stages of the imaging chain, directly to the sensor data if possible. While for still images, reflex cameras permit to have access to the RAW data, this is not available to the general public by video cameras or mobile phones. The amount of storage needed for keeping all sensor data of a whole video justifies this absence of RAW data for video. Unfortunately, techniques presented in this chapter are not real time, and therefore cannot be used in an onboard video imaging chain of a camera or mobile phone. The acceleration and design of real-time techniques yielding similar performance to the ones illustrated in this chapter is a huge challenge.

Finally, in order to numerically assess the performance of real video denoising methods, a database of realistic videos with ground truth is necessary. The first attempt for a particular imaging chain was carried out in [15]. A more systematic database should contain videos with different imaging chains and different compression algorithms and ratios.

## References

1. Adams A, Gelfand N, Pulli K (2008) Viewfinder alignment. In: Computer graphics forum, vol 27, pp 597–606. Wiley Online Library
2. Alvarez L, Lions PL, Morel JM (1992) Image selective smoothing and edge detection by nonlinear diffusion II. *SIAM J Numer Anal* 29(3):845–866
3. Alvarez L, Weickert J, Sánchez J (2000) Reliable estimation of dense optical flow fields with large displacements. *Int J Comput Vis* 39(1):41–56
4. Arias P, Morel JM (2017) Video denoising via empirical Bayesian estimation of space-time patches. *J Math Imaging Vis* 1–24
5. Ayvaci A, Raptis M, Soatto S (2010) Occlusion detection and motion estimation with convex optimization. In: *Advances in neural information processing systems*, pp 100–108
6. Ballester C, Garrido L, Lazzcano V, Caselles V (2012) A TV-L1 optical flow method with occlusion detection. In: *Lecture notes in computer science*, vol 7476. Springer, pp 31–40
7. Bennett E, McMillan L (2005) Video enhancement using per-pixel virtual exposures. In: *ACM SIGGRAPH 2005 papers*. ACM, p 852
8. Boulanger J, Kervrann C, Boutheimy P (2007) Space-time adaptation for patch-based image sequence restoration. *IEEE Trans Pattern Anal Mach Intell* 29(6):1096–1102
9. Brox T, Bruhn A, Papenbergh N, Weickert J (2004) High accuracy optical flow estimation based on a theory for warping. In: *European conference on computer vision*. Springer, pp 25–36

10. Brox T, Malik J (2011) Large displacement optical flow: descriptor matching in variational motion estimation. *IEEE Trans Pattern Anal Mach Intell* 33(3):500–513
11. Buades A, Coll B, Morel J (2005) A non local algorithm for image denoising. *IEEE Comput Vis Pattern Recognit* 2:60–65
12. Buades A, Coll B, Morel J (2008) Nonlocal image and movie denoising. *Int J Comput Vis* 76(2):123–139
13. Buades A, Duran J (2017) Flow-based video super-resolution with spatio-temporal patch similarity. In: *Proceedings British machine vision conference 2017*. BMVA
14. Buades A, Lisani JL (2017) Denoising of noisy and compressed video sequences. In: *VISI-GRAPP (4: VISAPP)*, pp 150–157
15. Buades A, Lisani JL (2017) Enhancement of noisy and compressed videos by optical flow and non-local denoising. *IEEE Trans Image Process* (submitted)
16. Buades A, Lisani JL, Miladinović M (2016) Patch-based video denoising with optical flow estimation. *IEEE Trans Image Process* 25(6):2573–2586
17. Buades A, Lou Y, Morel JM, Tang Z (2009) A note on multi-image denoising. In: *International workshop on local and non-local approximation in image processing*. IEEE, pp 1–15
18. Chambolle A (2004) An algorithm for total variation minimization and applications. *J Math Imaging Vis* 20:89–97
19. Colom M, Buades A, Morel JM (2014) Nonparametric noise estimation method for raw images. *JOSA A* 31(4):863–871
20. Colom M, Lebrun M, Buades A, Morel JM (2015) Nonparametric multiscale blind estimation of intensity-frequency dependent noise. *IEEE Trans Image Process*
21. Dabov K, Foi A, Egiazarian K (2007) Video denoising by sparse 3D transform-domain collaborative filtering. In: *Proceedings of the 15th European signal processing conference*, vol 1, p 7
22. Dabov K, Foi A, Katkovnik V, Egiazarian K et al (2009) BM3D image denoising with shape-adaptive principal component analysis. In: *Proceedings of the workshop on signal processing with adaptive sparse structured representations*, Saint-Malo, France
23. Delbracio M, Sapiro G (2015) Hand-held video deblurring via efficient Fourier aggregation. *IEEE Trans Comput Imaging* 1(4):270–283
24. Deledalle CA, Tupin F, Denis L (2010) Poisson NL-means: unsupervised non local means for Poisson noise. In: *17th IEEE international conference on image processing*. IEEE, pp 801–804
25. Donoho D (1995) De-noising by soft-thresholding. *IEEE Trans Inf Theory* 41(3):613–627
26. Drago F, Myszkowski K, Annen T, Chiba N (2003) Adaptive logarithmic mapping for displaying high contrast scenes. In: *Proceedings of EUROGRAPHICS*, vol 22
27. Durand F, Dorsey J (2002) Fast bilateral filtering for the display of high-dynamic-range images. In: *ACM transactions on graphics (TOG)*, vol 21. ACM, pp 257–266
28. Elad M, Aharon M (2006) Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Trans Image Process* 15(12):3736–3745
29. Farsiu S, Robinson M, Elad M, Milanfar P (2004) Fast and robust multiframe super resolution. *IEEE Trans Image Process* 13(10):1327–1344
30. Gao Y, Hu HM, Wu J (2015) Video denoising algorithm via multi-scale joint luma-chroma bilateral filter. In: *2015 visual communications and image processing (VCIP)*. IEEE, pp 1–4
31. Haro G, Buades A, Morel JM (2012) Photographing paintings by image fusion. *SIAM J Imaging Sci* 5(3):1055–1087
32. Horn B, Schunck B (1981) Determining optical flow. In: *Technical symposium east*. International Society for Optics and Photonics, pp 319–331
33. Ji H, Huang S, Shen Z, Xu Y (2011) Robust video restoration by joint sparse and low rank matrix approximation. *SIAM J Imaging Sci* 4(4):1122–1142
34. Joshi N, Cohen M (2010) Seeing Mt. Rainier: lucky imaging for multi-image denoising, sharpening, and haze removal. In: *IEEE international conference on computational photography*, pp 1–8
35. Jovanov L, Luong H, Ružic T, Philips W (2015) Multiview image sequence enhancement. In: *SPIE/IS&T electronic imaging*. International Society for Optics and Photonics, pp 93,990K–93,990K



36. Kim M, Park D, Han DK, Ko H (2015) A novel approach for denoising and enhancement of extremely low-light video. *IEEE Trans Consum Electron* 61(1):72–80
37. Kokaram AC (2013) Motion picture restoration: digital algorithms for artefact suppression in degraded motion picture film and video. Springer Science & Business Media
38. Lebrun M, Buades A, Morel JM (2013) A nonlocal Bayesian image denoising algorithm. *SIAM J Imaging Sci* 6(3):1665–1688
39. Lebrun M, Colom M, Buades A, Morel JM (2012) Secrets of image denoising cuisine. *Acta Numerica* 21:475–576
40. Lebrun M, Colom M, Morel JM (2015) The noise clinic: a blind image denoising algorithm. *Image Process On Line* 5:1–54
41. Liu C, Freeman W (2010) A high-quality video denoising algorithm based on reliable motion estimation. In: *European conference on computer vision*. Springer, pp 706–719
42. Liu C, Szeliski R, Kang S, Zitnick C, Freeman W (2008) Automatic estimation and removal of noise from a single image. *IEEE Trans Pattern Anal Mach Intell* 30(2):299–314
43. Lowe D (2004) Distinctive image features from scale-invariant keypoints. *Int J Comput Vis* 60(2):91–110
44. Maggioni M, Boracchi G, Foi A, Egiazarian K (2011) Video denoising using separable 4D nonlocal spatiotemporal transforms. In: *IS&T/SPIE electronic imaging*. International Society for Optics and Photonics, pp 787,003–787,003
45. Mairal J, Sapiro G, Elad M (2008) Learning multiscale sparse representations for image and video restoration. *SIAM Multiscale Model Simul* 7(1):214–241
46. Mäkitalo M, Foi A (2013) Optimal inversion of the generalized Anscombe transformation for Poisson-Gaussian noise. *IEEE Trans Image Process* 22(1):91–103
47. Orchard J, Ebrahimi M, Wong A (2008) Efficient non-local-means denoising using the SVD. In: *Proceedings of the IEEE international conference on image processing*
48. Ozkan MK, Sezan MI, Tekalp AM (1993) Adaptive motion-compensated filtering of noisy image sequences. *IEEE Trans Circuits Syst Video Technol* 3(4):277–290
49. Papenberg N, Bruhn A, Brox T, Didas S, Weickert J (2006) Highly accurate optic flow computation with theoretically justified warping. *Int J Comput Vis* 67(2):141–158
50. Ponomarenko NN, Lukin VV, Abramov SK, Egiazarian KO, Astola JT (2003) Blind evaluation of additive noise variance in textured images by nonlinear processing of block DCT coefficients. In: *Electronic imaging*. International Society for Optics and Photonics, pp 178–189
51. Rudin L, Osher S, Fatemi E (1992) Nonlinear total variation based noise removal algorithms. *Phys D* 60(1–4):259–268
52. Sand P, Teller S (2008) Particle video: long-range motion estimation using point trajectories. *Int J Comput Vis* 80(1):72–91
53. Smith S, Brady J (1997) SUSAN: a new approach to low level image processing. *Int J Comput Vis* 23(1):45–78
54. Szeliski R (2006) Image alignment and stitching: a tutorial. *Found Trends Comput Gr Vis* 2(1):1–104
55. Tomasi C, Manduchi R (1998) Bilateral filtering for gray and color images. In: *Sixth international conference on computer vision*, pp 839–846
56. Wang YQ, Morel JM (2013) Sure guided gaussian mixture image denoising. *SIAM J Imaging Sci* 6(2):999–1034
57. Wedel A, Pock T, Zach C, Bischof H, Cremers D (2009) An improved algorithm for TV-L1 optical flow. In: *Statistical and geometrical approaches to visual motion analysis*. Springer, pp 23–45
58. Wen B, Ravishanker S, Bresler Y (2015) Video denoising by online 3d sparsifying transform learning. In: *2015 IEEE international conference on image processing (ICIP)*. IEEE, pp 118–122
59. Xu Q, Jiang H, Scopigno R, Sbert M (2010) A new approach for very dark video denoising and enhancement. In: *17th IEEE international conference on image processing*. IEEE, pp 1185–1188

60. Yaroslavsky L, Egiazarian K, Astola J (2001) Transform domain image restoration methods: review, comparison, and interpretation. *Proc SPIE* 4304:155
61. Yaroslavsky LP (1985) *Digital picture processing*. Springer, New York Inc, Secaucus, NJ, USA
62. Yue H, Sun X, Yang J, Wu F (2015) Image denoising by exploring external and internal correlations. *IEEE Trans Image Process* 24(6):1967–1982
63. Zach C, Pock T, Bischof H (2007) A duality based approach for realtime TV-L1 optical flow. In: *Pattern recognition*. Springer, pp 214–223
64. Zhang L, Dong W, Zhang D, Shi G (2010) Two-stage image denoising by principal component analysis with local pixel grouping. *Pattern Recognit* 43(4):1531–1549
65. Zhang M, Gunturk BK (2008) Multiresolution bilateral filtering for image denoising. *IEEE Trans Image Process* 17(12):2324–2333. <https://doi.org/10.1109/TIP.2008.2006658>

# Chapter 8

## Image and Video Noise: An Industry Perspective



Stuart Perry

**Abstract** Images and video are increasingly becoming a part of our everyday lives and with this growth, we find an increasing number of industrial and commercial applications of imagery. In this chapter, we will examine the problem of image noise from an industrial and commercial viewpoint. We will consider how noise enters the imaging chain in these settings and how noise is measured and quantified for later removal. We will also discuss standards and standardisation activities that relate to noise measurement in a commercial or industrial setting.

### 8.1 Overview

Images and video are a multibillion-dollar industry in the twenty-first century. The number of images taken per year continues to grow at an exponential rate spurred on by the ubiquitous presence of cameras in mobile phones and other small, inexpensive devices [1]. Cameras are vital pieces of equipment in a variety of commercial applications, with entire conferences, books and journals devoted to imagery in specific commercial application domains such as autonomous vehicles [2]. Given the large sums of money involved, it is not surprising that commercial and government entities have devoted substantial resources to understanding how noise enters image and video processing pipelines, how noise levels in images can be quantified and removed. Other chapters in this book provide excellent algorithms for the removal and filtering of noise in images and videos, but in this chapter, we will consider first how noise enters the imaging pipeline for commercial or industrial systems and how noise is typically measured by the commercial and industrial manufacturers of imaging systems and technologies that make use of visual imagery.

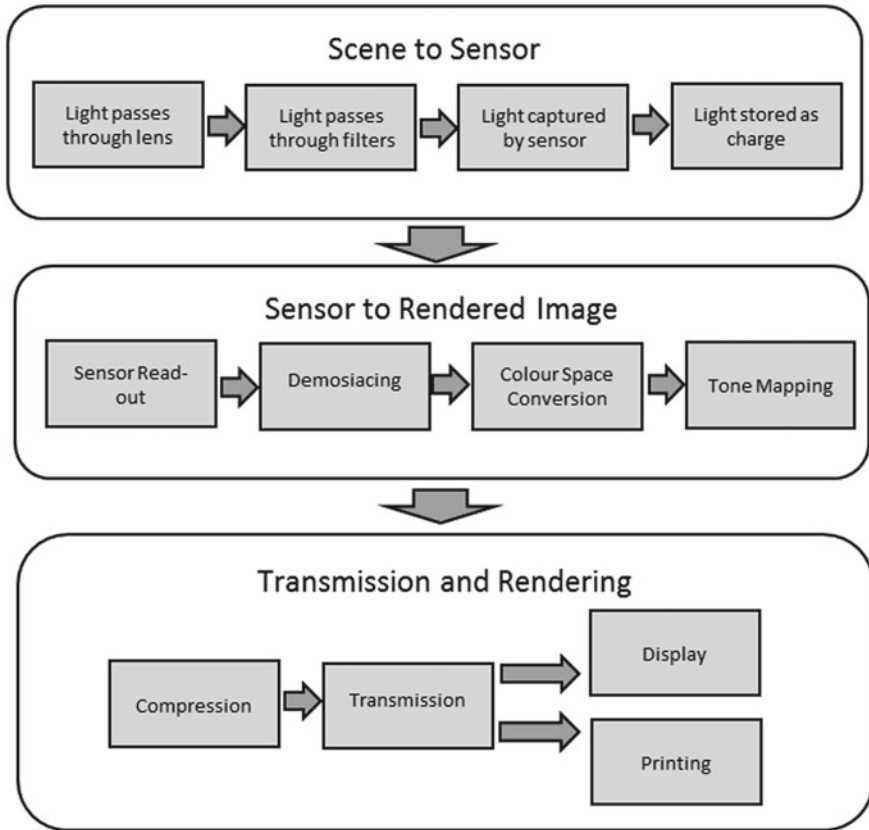
---

S. Perry (✉)

School of Electrical and Data Engineering, University of Technology  
Sydney, Sydney, NSW, Australia  
e-mail: Stuart.Perry@uts.edu.au

© Springer International Publishing AG, part of Springer Nature 2018  
M. Bertalmío (ed.), *Denosing of Photographic Images and Video*,  
Advances in Computer Vision and Pattern Recognition,  
[https://doi.org/10.1007/978-3-319-96029-6\\_8](https://doi.org/10.1007/978-3-319-96029-6_8)

207



**Fig. 8.1** A typical imaging chain

## 8.2 Noise in the Image Processing Pipeline

Digital imaging systems are rapidly improving in quality and functionality, but noise will always remain an issue for industrial and commercial applications. As we improve imaging technology, we continue to push it into more challenging environments and there will always be a need to remove noise from imagery in industrial applications.

There is a variety of types of noise commonly encountered in images and they occur at various points in the imaging chain. Understanding these types of noise will help us understand the way they affect the measurement of noise in imagery [3]. Often, we refer to ‘noise models’ to describe statistical models of noise in imagery.

Figure 8.1 shows an overview of a typical imaging chain from capture to rendering or display. A typical imaging chain can be divided into several stages:

First, light from the scene must be captured by an imaging sensor. This often involves the light passing through a lens system to be focused on the sensor. The

light may then pass through various filters to either remove unwanted wavelengths of light or isolate particular wavelengths. In the case of consumer digital cameras, one set of filters are used to isolate the red, green and blue wavelengths to represent colour in the final image. Another set of filters are used to remove the infrared wavelengths of light that human vision is not sensitive to, but silicon photosensitive elements are capable of recording. The light is then captured by the sensor. If the sensor is a film substrate, this occurs because of a reaction with photosensitive chemicals. If the sensor is digital, then light is often captured by the excitation of charge carriers in a semiconductor and stored as charge in each sensor site. In this chapter will concentrate on digital image capture.

Second, the captured charge must be read out of the sensor sites and quantised into a digital signal. If the sensor is a colour image sensor, then disparate information on colour captured at different locations on the sensor must often be merged to estimate the exact colour at each pixel location. This is called ‘demosaicing’ due to the mosaic arrangement of colour filters on modern digital image sensors. This is often followed by conversion of the image data into a device-independent colour space and remapping of the luminance levels to allow for perceptually pleasing display to a human observer.

Third, the image is generally sent to a rendering system where it can be displayed to a human observer. This often involves compression of the image or video data to conserve bandwidth and transmission over a communications channel. The rendering system might convert the image back into transmitted light such as a monitor display or render the image using light absorbing pigments on a medium such as a printer.

Not all imaging chains have all the above processes, but noise can enter the imaging chain at any of the above points.

## ***8.2.1 From Scene to Sensor***

The process of getting light from the scene to the sensor can add various distortions to captured imagery. There are a large variety of noise sources that affect images and video caused by the sensor and lens system of image capture equipment [4]. Some of the most important noise sources are mentioned below.

### **8.2.1.1 Gaussian Noise**

The most common noise model is Gaussian noise. Gaussian noise can arise from various processes in the camera and lens system. Gaussian noise is often a good approximation of other types of noise under a variety of conditions and, because of the central limit theorem [5], Gaussian noise can describe the output noise statistics when many other types of noise are passed through a complex system. Gaussian noise is distributed independently for each pixel in each colour channel according to the Gaussian distribution [5, 6]. Gaussian noise does not depend on the intensity of

pixels in the image and is mathematically convenient to deal with. This type of noise is a good description of noise associated with light capture during image acquisition or some forms of thermal noise in image capture sensors [7, 8]. Gaussian noise is often a good approximation for the noise associated with the capture of light by imaging sensors when image light levels are high [9].

### 8.2.1.2 Shot Noise and Poisson Noise

Shot noise is noise inherent in the capture of photons by a sensor [3, 4]. It arises due to the particle nature of light. The value of a pixel is dependent on the count of photons collected by the pixel during an exposure. For bright light situations, each pixel may be exposed to billions of photons and any noise from capture to capture might be only a tiny percentage of the pixel's value. However, when light levels are low, and only a few dozen photons are collected in an exposure, the variation in pixel values from capture to capture can be considerable. Hence, shot noise is dependent on the pixel intensity and is often modelled as a Poisson distribution [5]. Poisson distributions describe processes that count randomly occurring events (such as photons striking a pixel sensor site) and have the property that the noise variance is proportional to the value of the quantity being measured. As the quantity being measured (such as light level) increases, the Poisson distribution approaches a Gaussian distribution [9]. In practice, this means that shot noise deviates most strongly from Gaussian noise at low light levels. Shot noise can be the primary source of noise in imagery at high light levels [4].

### 8.2.1.3 Lens Flare

Modern camera equipment can have complicated lens structures to enable the change of focus, optical zoom, or the correction of various distortions [10]. Although these systems are often very efficient, light reflected from the interfaces between the lenses can appear as noise in captured imagery. This is often referred to as 'lens flare' and can manifest in a variety of ways. When the lens flare is distributed evenly within the captured image it can be modelled as an additional photon noise factor with a shot noise distribution [4]. Unlike photon noise associated with the image content, the statistical properties of this form of lens flare are often not strongly correlated with image content, but rather with the structure of lens system. Lens flare can also manifest in captured imagery as polygonal shapes in the image related to the structure of the camera lens or bright streaks [11]. This type of noise is so common that the structured form of lens flare is sometimes added to imagery for visual effect or in CGI imagery to simulate capture by a physical camera [12].

#### 8.2.1.4 Thermal Noise

Thermal noise in image sensors occurs when random thermal excitation of electrons in the silicon substrate causes additional false charge in the pixel sensor site. At high light levels, this noise source is often small compared to other noise sources such as photon noise; however, thermal noise can dominate at low light levels [9]. Thermal noise on image capture sensors can be well modelled by a Poisson or Gaussian distribution [9]. Thermal noise can be mitigated by cooling the imaging sensor and this technique is often used for extremely sensitive imaging applications.

#### 8.2.1.5 Flicker Noise

Flicker noise occurs in most electronic devices and is often due to impurities in the silicon or noise in the generation or recombination of charge carriers in the device. This noise causes variations in the performance of electronic components. In the case of imaging technology, flicker noise causes variations in the charge collected by semiconductor sensors. Flicker noise is also called ‘pink noise’ and is characterised by having a power density spectrum with a  $1/f$  shape where  $f$  is the temporal frequency of the noise [9].

#### 8.2.1.6 Aliasing

Most digital imaging sensors rely on an array of photosensitive elements to collect light. These elements are usually arranged on a grid, although they can be augmented by filters or micro-lens arrays to achieve various purposes such as the capture of colour or light field data [4, 13].

However, all imaging systems based on sensor arrays can only sample the real world at a set of points and as such are subject to the problem of aliasing [4]. Aliasing occurs when the number and spacing of samples of a signal are insufficient to describe spatial or temporal variation in the signal. When the sampling is insufficient, energy from high-frequency components of the signal is mapped back (or ‘aliased’) into the lower frequency components, producing distortions and artefacts in the sampled signal [6]. Digital imaging systems are subject to these problems as much as other signal sampling systems. In digital images, aliasing can produce jaggy edges on lines and curves, and Moiré Effect patterns in the output images [6]. Modern digital imaging systems generally have low-pass components in the optical path or have pixels small enough that aliasing is not noticeable unless the image is enlarged to the pixel level [14].

## ***8.2.2 From Sensor to Rendered Image***

Once light has been gathered by a sensor, it needs to be read off the sensor and then rendered in a way useful to further machine processing or viewing by human observers. There can be many stages in this process and noise can enter the system at most of these stages.

### **8.2.2.1 Noise When Reading Out the Sensor Value**

The process of converting the pixel from stored charge at a sensor site to an electrical signal can introduce a variety of artefacts and noise.

Quantization noise is caused by the quantization of pixel values as a result of the conversion from analogue to digital sampling. It is usually modelled by a uniform distribution [4]; however, a uniform distribution model can break down when the number of quantization levels is small [3]. As modern digital cameras increase in the available bit depth to represent pixel values, this problem becomes less of a concern.

Salt and Pepper Noise (also called Impulse Noise) refers to the random occurrence of pixels saturated to some set of values [3]. This type of noise often occurs because of faulty pixels on either capture or display systems or errors during pixel read-out on sensors. Salt and Pepper Noise is usually modelled by assuming that a particular percentage of pixels randomly selected in the image are subject to noise which manifests as 50% of affected pixels being set to white and 50% of affected pixels being set to black. Pixels not randomly selected as noise pixels in the image are unaffected.

### ***8.2.3 Fixed-Pattern Noise***

Due to irregularities in the manufacture of imaging sensors, individual sensor sites can have differing efficiencies in the capture of light. These differences result in pixels being slightly brighter or darker than their neighbours due to differences in gain [4]. This is termed ‘fixed-pattern noise’ as the pattern of differing gains on each pixel does not vary with the image content. Fixed-pattern noise is generally removed by careful calibration of the imaging sensor prior to capture so that differences in pixel gain can be compensated for following sensor read-out. In two-dimensional sensors, fixed-pattern noise often manifests as a pattern of points across the image; however, in line-based imaging systems such as document scanners, fixed-pattern noise can manifest as an aperiodic banding pattern [15]. Imaging sensor site efficiency can change over time, so some digital cameras allow for periodic recalibration to remove fixed-pattern noise. Commercial line scanners often have a calibration strip that is imaged prior to each scan and used to correct for fixed-pattern noise.



### 8.2.3.1 Noise from Demosaicing

The majority of silicon imaging sensor sites use colour filters to perceive light at different wavelets. For devices capturing images in the visible spectrum, red, green and blue filters are arranged in a pattern on the sensor hence devoting every pixel to the capture of red, green and blue light only. The most common pattern is known as a ‘Bayer Filter’ and assigns twice as many pixels to capture green light as those capturing red or blue light [4]. The higher sampling of the green part of the spectrum is motivated by the greater sensitivity of human vision to these wavelengths of light [16].

A result of this approach is that colour information recorded by imaging systems is often not perfectly aligned. Each pixel contains information on either the red, green or blue parts of the spectrum and to obtain a full-colour image with red, green and blue information at each pixel, interpolation is used to compute the missing colour information at every pixel in the image. This operation is often termed ‘Image Demosaicing’ [17]. Image Demosaicing attempts to recreate missing information in an image and can introduce various sources of image noise and artefacts. Demosaicing artefacts include the ‘zippering effect’ or the presence of false colours near edges. The zippering effect produces an unnatural pattern in the image around strong edges and is caused by the demosaicing algorithm’s interpolation not being aligned correctly relative to the edges in the image. To avoid this, many demosaicing algorithms are edge adaptive [17, 18].

### 8.2.3.2 Problems with Colour Space Conversion and Tone Mapping

Following demosaicing, images are often converted to a device-independent colour space such as sRGB and luminance levels are remapped for display to human observers. Both processes can introduce noise or artefacts in images.

Errors in gamut mapping between different colour spaces can produce saturation of portions of the image, colour shifts for bright or very chromatic areas, and false contours. False contours can be created when out-of-gamut pixels are changed, leaving adjacent in-gamut pixels unchanged [19].

False contours can also be created during when luminance levels are mapped for display on a specific device or more generally to make the image perceptually pleasing for a human observer. This operation is called ‘tone mapping’ and false contours can occur when the tone mapping operation increases a normally unnoticeable luminance difference between pixels in an area of gradually changing luminance to the extent that the luminance difference is now apparent to human observers. This produces unnatural lines and contours in areas such as sky where human observers would not expect structure to occur [6].

Tone mapping operations can also complicate the noise models used in imagery. The Gaussian distribution is a good approximation for many noise sources; however, the Gaussian distribution is only a good description in those cases when noise is measured on an image prior to non-linear tone mappings [20]. Tone mappings can

transform Gaussian noise into noise that is now dependent on the intensity of the pixels in the image. Other common image acquisition processing such as auto white balance or image sharpening can also produce non-Gaussian noise in the resultant images.

## **8.2.4 *Transmission and Rendering***

Once images have been captured by an imaging system and rendered for display, the images may then be transmitted to another location and displayed on a variety of systems. Noise can be added to the image during transmission and rendering. This can occur when the image is compressed to save bandwidth during transmission, during the transmission process itself, or when the image is displayed on a monitor or printed.

### **8.2.4.1 *Artefacts of Compression***

Modern images and videos are often extremely large and require image and video compression techniques to reduce the size of these images. Modern image and video compression algorithms can produce impressive bandwidth savings with little perceptual quality decrease; however when compression algorithms are pushed to compress the image or video to extreme levels, a variety of artefacts and forms of noise can occur. These include [21]:

- **Blocking:** Major image and video compression algorithms divide the image or video into small spatial blocks that are converted into a frequency-based representation and compressed by not encoding high-frequency information that is invisible to a human observer. However, when the compression levels are high, the boundaries between the blocks can be observed as a structured noise in the image.
- **Basis Image Error Noise:** Conversion of spatial blocks in images and video to a frequency representation for compression often involves the use of the Discrete Cosine Transform (DCT) [21]. Leakage of the basis images of the DCT into the final image results in Basis Image Error Noise. This is caused by coarse quantization of the higher frequency DCT coefficients resulting in only a few lower frequency coefficients remaining. During decoding, the basis images of the DCT transform corresponding to the remaining low-frequency coefficients are visible in the final image.
- **Staircase Effect:** This distortion causes diagonal edges in the image to look jagged with a ‘staircase’ pattern. This is caused by the nature of the discrete cosine transform. Whereas the DCT basis images are tuned to horizontal and vertical edges, they are not well tuned to diagonal image structure.
- **Ringing:** Ringing manifests as false edges in the decompressed image around strong edges in the original image. This occurs because of the quantisation of

the higher frequency DCT coefficients. Strong edges need higher frequency DCT basis images to adequately represent them, and when these are removed by the compression process, ringing results.

- **False Contouring:** As described above in relation to tone mapping, false contouring occurs when smoothly varying regions become quantised because of the compression process.
- **Temporal Effects:** Compression of video can produce a variety of temporal distortions and noise. One of the key processes in video compression is the matching of spatial blocks across video frames. By matching a spatial image block in one frame with that in a different frame, compression can be achieved by only encoding the position change of the block in the second frame rather than the image data. However, mismatched blocks across frames can produce disturbing temporal artefacts. Another source of temporal noise is the ‘mosquito effect’ due to varied coding of the same area across frames causing temporal changes such as luminance changes in smoothly varying areas of the video.

#### **8.2.4.2 Noise of Digital Transmission**

Periodic banding noise can occur when electrical interference occurs during either capture or transmission of imagery [22]. This can manifest in several ways, but one of the most disturbing is banding caused by electrical interference. This may appear as lines that are slanted relative to the image raster and is often the result of the frequency of the interfering noise being not a harmonic of the carrier frequency used during the transmission of the image.

#### **8.2.4.3 Noise of Display on Monitors**

Modern display systems when correctly calibrated are often not a major source of image noise. However, displays can have ‘dead’ pixels that manifest as salt and pepper noise, and spatial or temporal variations in the luminance of the pixels on the screen. These noise sources are often not noticed by consumers but are of concern to professionals such as radiographers that make use of monitors to detect faint structure in images for medical diagnostic purposes [23].

#### **8.2.4.4 Noise During Printing**

Printing of imagery is a complicated process. Problems with the absorption or delivery of ink on paper can cause irregularities of the printed image while mechanical slippage of the feed mechanism can result in line artefacts known as ‘banding’. Banding is a common source of noise in printed images.

Banding noise refers to any noise source that causes line artefacts to appear across an image. Banding noise may result in periodic or aperiodic lines occurring across

the image. Aperiodic banding can occur when the paper rollers slip during printing on an ink-jet printer. The slippage causes the alignment of the halftone printing pattern to be locally disrupted, often resulting in a band across the page either darker or lighter than the surrounding print. The band often runs across the width of the page perpendicular to the media feed direction during printing. However, there are many other causes of both periodic and aperiodic banding noise in printers [24] including blocked ink-jet nozzles which can result in banding that runs parallel to the media feed direction.

There are international standards [25] that define a number of noise sources for printers. These include a wide class of irregularities with ink deposition on paper such as background haze, graininess, mottle, extraneous marks, banding and voids. The difference between these noise sources is often simply the size and shape of the artefacts or whether the noise manifests as unwanted ink on unprinted sections of the paper or unwanted gaps in ink on printed sections of the image. Noise along printed edges in documents was also addressed in the standard and termed ‘raggedness’.

### 8.3 Measuring Noise in Images

There is a multitude of algorithms for the removal of noise from imagery [6, 22], but at the heart of these algorithms is the need to measure and quantify the level of noise in an arbitrary image. This is a difficult problem as imagery can contain a variety of content from low to high frequency that can resemble, enhance or mask the appearance of noise [26]. Often only phase coherency of image content as it is arranged into edges and shapes, rather than magnitude, differentiates true content from noise [27].

Often the problem of measuring noise is closely coupled to the problem of quantifying the difference between two arbitrary images. There are techniques and standards for measuring noise of imaging equipment through the use of test charts and specialised equipment [28]. However, often engineers need to measure noise as it appears in arbitrary image content. The visibility, effect on quality and nature of noise on an arbitrary image processed in an imaging chain in applications in the real world can be quite different to the results obtained in a controlled laboratory setting.

We might imagine that when a perfect copy of a noisy image exists, measuring and quantifying the noise is simply a matter of subtracting the perfect copy from the noisy image and applying simple summary statistics on the resulting ‘noise image’ such as measuring the power or energy of the noise image. In practice, this often fails for two important reasons.

First, in many applications, an original image may not be available. For example, when engineers need to measure the level of degradations caused during transmission of imagery through channels with time-varying noise levels [29].

Second, often the factor of interest to engineers is not the noise level in the sense of the power of the noise signal, but rather the visibility of the noise to a human

observer or the degree that the noise added to the signal degrades the quality of the signal as judged by a human observer.

There are many image difference operators that attempt to either measure the visibility of noise in an image or quantify the quality loss that occurs when noise is added [29, 30]. These image difference operators can be considered measures of the noise in imagery.

### ***8.3.1 Classifying Image Noise/Difference/Quality Metrics***

As mentioned above, noise in imagery is usually quantified by image difference or quality measures. These measures can be divided into a number of different categories depending on various properties. One of the most common divisions is that of the amount of information about the original perfect image that needs to be transmitted together with the noisy image to allow the level of noise to be quantified. Generally, this division classifies metrics into Full Reference (FR), Reduced Reference (RR) or No Reference (NR) [29, 30].

Full-Reference (FR) metrics require the perfect original image to be present to enable the level of noise present in the noisy image to be quantified. This is appropriate for controlled testing of a system or scientific studies into noise suppression algorithms. Full-reference metrics are generally not appropriate for adaptively changing the parameters of a system according to arbitrary transmitted content in real time as the original image is generally unavailable in practical applications. Examples of FR metrics include MSE, PSNR [6] and SSIM [31].

Reduced-Reference (RR) metrics do not require the original image to be available, but instead create a set of summary statistics of the original image that are transmitted to the receiver and compared to the same summary statistics collected from the received noisy image [29]. This approach can allow transmission channels to adapt to changing environmental conditions by continuously measuring the quality of imagery transmitted through the channel. Reduced-reference metrics are not applicable to measuring the quality of data directly from image capture devices and are generally not as accurate as full-reference metrics. The more data transmitted by the RR metric to summarise the image, the greater the accuracy, but the more bandwidth used. Examples of RR metrics can be found in the survey by Engelke and Zepernick [29].

No-Reference (NR) metrics require no information about the original image. Instead, they make certain assumptions about the original content such as the statistical distribution of natural images or the types of distortions present in the noisy image. In general, NR metrics do not perform as well as FR or RR metrics; however, they continue to be an active area of research as the applications of NR metrics are very broad compared to FR or RR metrics. A survey of RR and NR measures can be found in Engelke and Zepernick [29].

Full-reference, reduced-reference and no-reference metrics all need to make assumptions about what noise is and the effect it has on a human viewing the content and based on these assumptions we can further categorise the metrics. In particular,

we can arrange the metrics into distortion-specific metrics, Human Visual System-based metrics and natural image statistic-based metrics.

### 8.3.1.1 Distortion-Specific Metrics

Distortion-specific metrics attempt to measure noise by having very particular models of noise artefacts, such as banding, blur or blockiness and searching for those specific distortions in the image data. The goal is to find and quantify the distortions without regard to the image content and without considering the wider variety of image distortions and noise that may impact image quality. The vast majority of distortion-specific metrics are no-reference metrics since they generally make use of algorithms to detect the noise type being measured without reference to an original image. However, this category can be considered to include full-reference metrics such as MSE and PSNR since these measures subtract out the image content, and do not consider the human visual system when measuring image difference. Both measures will be described in more detail in a later section.

A difficulty with distortion-specific measures is that there is a wide variety of types of noise that can affect images and video, and discriminating all these different types of noise from the large variety of different types of image content is an impossible task. Solve this problem; distortion-specific measures often target only one particular type of distortion or a limited variety.

An important source of noise or distortions in images and video in real life applications are compression algorithms. Fortunately, compression algorithms produce well-known distortion types, such as blocking, blur or quantisation noise. Blocking artefacts result from the division of images and videos into regular sub-blocks prior to compression and are both readily noticeable to humans and degrade subjective quality quickly. A number of noise measures have been developed directly targeting blocking artefacts, for JPEG compressed imagery [32], for compressed video [33] or for both [34]. Some blocking measures are deliberately designed to ignore or to produce results that are intolerant of other types of image noise [34].

Blur is also a distortion caused by compression of both images and video, as well as being a problem inherent to capture systems as well. Measuring the degree of blur in an image is difficult even when the original image is known [35], but becomes even more difficult in the case of no-reference metrics [36]. Some blur metrics attempt to measure blur on the compressed image and video formats [37]. This has the advantage of enabling blur to be measured on a communications channel without explicitly de-compressing the content for this purpose. This can result in a substantial saving in computational resources.

In video sequences, distortions in motion, such as frozen frames or errors in the predicting the motion of objects in the scene can have a dramatic effect on image quality. It is not surprising that distortion-specific metrics have been designed to address noise associated with motion in video [38, 39].

Finally, there are distortion-specific metrics that merge individual specific distortion measurements together to create a final measure of image quality. A good

example of such a metric is the metric for JPEG images developed by Wang et al. [40]. This metric extracts measures of blocking, blur and noisiness and weights them according to their importance to human perception to create a single quality metric. The video metric developed by Cavallaro and Winkler [41] extracts blocking, blur and jerkiness measures and combines this with semantic segmentation of the video frames to incorporate human visual attention characteristics [29].

### 8.3.1.2 Human Visual System Based Metrics

HVS-based Metrics attempt to model the human visual system to determine how noise affects the perception of image quality. Often noise and image content is not directly modelled, but rather the affect that these have on a human observer. HVS-based metrics are almost always full-reference metrics due to the difficulty of completely modelling the human response to imagery. The presence of the original image simplifies the problem down to one of modelling the human response to the difference between two images. There are too many metrics that attempt to model the human visual system to describe them comprehensively in this section. The reader is referred to various survey papers and summaries for this purpose [29, 30, 42]. However, most metrics contain at least some of the following steps:

1. The images to be compared are converted into some colour space where differences between colours are perceptually uniform such as CIELAB [16, 43]. Many older metrics were designed to work on grayscale images [44, 45]. These metrics are still useful in the world of colour images due to the greater importance to human observers of the differences in luminance information between images compared to chrominance information [46]. However, there is a recognised need for colour metrics [46] to accurately characterise distortions that occur with colour imagery.
2. The images to be compared are filtered based on human sensitivities to contrast at different spatial frequencies, often considering, or making assumptions regarding, viewing conditions [47]. This is performed to remove details not visible to human observers [43].
3. The images to be compared are decomposed into some frequency representation designed to model the differing responses of the human visual system to content at different spatial frequencies, scales and orientations [45, 48]. Gabor filters [44], wavelet filters [49] or DCT [50] decomposition is commonly used at this stage, due to their ability to preserve local variations in frequency content.
4. Contrast and noise masking effects are modelled by combining coefficients in the frequency space representation across different spatial locations in the same frequency band or across different frequency bands [42, 51].
5. A summation step combines differences in coefficients across different frequency bands weighted according to the affect that each band has on human perception to create a final measure [42, 51].

Metrics based on models of the Human Visual System still face several difficulties that have restricted their adoption by industry:

1. HVS-based metrics are still subject to content dependence wherein the metrics do not perform consistently for all types of content [40].
2. Many aspects of the measures, in particular, the response of human vision to the spatial frequency of content, are modelled using functions developed from ‘threshold’ experiments. Threshold experiments determine the point at which humans can detect a change in a stimulus. However, many HVS-based metrics use these functions to weight the contribution of content at different spatial frequencies to the overall metric even when this content is at levels well above the threshold of visibility. When content is above the threshold of visibility, there is substantial evidence that these functions need to be modified [49].
3. Higher level concepts such as interactions between content and the perception of quality are often lacking [49]. It is known that the presence of edges, skin and faces and other salient areas can affect the perception of quality. Some error metrics have attempted to include limited higher level perceptual characteristics such as semantic segmentation of the image content with some success [41].

### 8.3.1.3 Natural Image Statistics-Based Metrics

These metrics rely on statistical models of natural images. The assumption of such methods is that the human visual system is evolved to extract information from natural imagery. As such, human subjective notions of quality are modelled as being closely related to the degree to which an image deviates from appearing like a natural image. Often these measures do not model the noise process or the physiology of human vision directly. Many of these metrics are no reference [52] or reduced reference [53], relying on the statistical model of an ideal image to replace knowledge of the original image.

Often natural images are modelled in some frequency space such as Fourier or wavelet spaces. In Fourier space, the average magnitude of the Fourier coefficients of natural images is known to fall off at a  $1/f$  rate [52], where  $f$  is the frequency of the coefficient. Images substantially deviating from this distribution can be considered as appearing unnatural to a human observer. In the wavelet domain, the distribution of coefficients is very non-Gaussian, characterised by distributions with heavy tails and high kurtosis. In addition, it has been observed that the wavelet coefficients display a great degree of self-similarity across scales with similar distributions across scales and a high degree of correlation across scales [54]. This scale independence is only true when natural images are considered as an ensemble. Individual natural images may violate this concept [54]. Some researchers have attempted to model the distributions of natural images by considering the random arrangement of shapes according to a Poisson process. Shapes can be drawn from a group with random textures, sizes and shapes with or without occlusion. This is commonly called a



‘dead leaves’ pattern [54] and has been used for testing image capture systems in industrial applications [55].

A representative example is the work of Simoncelli [56, 57] where the statistics of wavelet coefficients and their relationships to wavelets coefficients at other scales and orientations were encoded by representing the coefficients at a scale, orientation and location of interest as an estimate using coefficients in other scales and orientations at the same location, nearby neighbours at the same orientation and scale, and a noise term. The noise is modelled as Gaussian, but the weights used in the coefficient estimate are modelled using an empirical distribution. This was used as a model of natural image statistics for image compression. Shiekh et al. [52] extended this idea and expanded it to develop a no-reference image quality metric that could be applied in the DCT compressed domain.

This approach can also be used to develop reduced-reference metrics. In Wang and Simoncelli [53] the statistical distribution of the coefficients of a wavelet decomposition is approximated as a two-parameter Generalised Gaussian Density (GGD) model. The parameters of this model are transmitted with the image. On the receiver side, the distribution of the wavelet coefficients of the received image is computed and the Relative Entropy (also known as the Kullback–Leibler divergence [58]) is used to determine the difference between the distribution of wavelet coefficients between the original and received images.

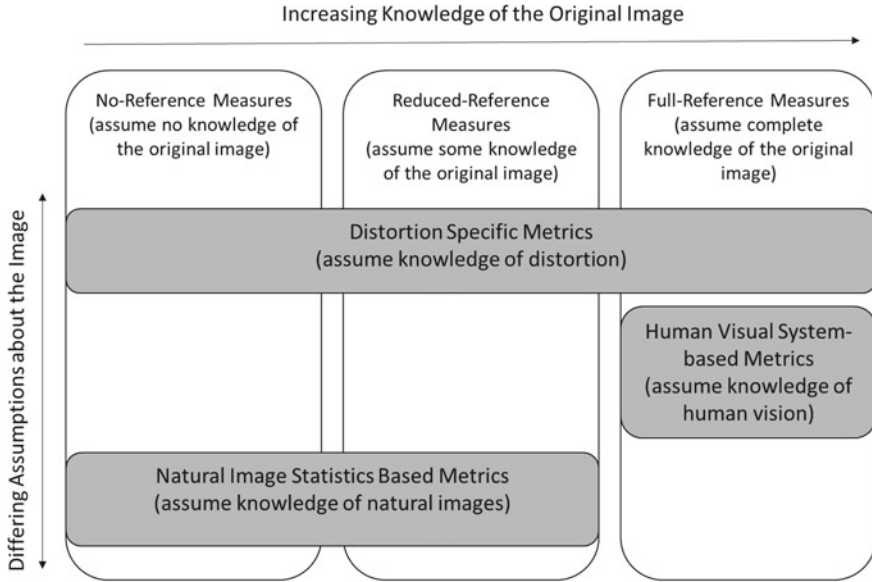
Figure 8.2 summarises, diagrammatically, the relationships between the measures that we have been exploring in this section.

### 8.3.2 *Commonly Used Metrics in Industry Applications*

Despite the wide range of image quality metrics described above, the problem of measuring noise in images is far from solved and very few of the metrics described above have found common use in industry. The most commonly cited noise metrics for natural images used in industry are Mean Square Error (MSE), Peak Signal-to-Noise Ratio (PSNR) and Structural SIMilarity Index (SSIM). All of these measures are full-reference metrics due to the better performance of FR metrics in general. We will describe these measures in this section, while the next section will describe efforts to standardise the measurement of noise and image quality in general.

#### 8.3.2.1 MSE and PSNR

Mean Square Error (MSE) is simply a measure of the residual energy in a test image once a perfect copy of the original image has been removed. For an image of size  $M$  by  $N$  pixels,



**Fig. 8.2** One possible classification of image quality and difference metrics based on the degree of knowledge assumed by the metric. Along the horizontal axis, the degree of knowledge about the original image increases. Along the vertical axis, the metrics are divided into those that assume knowledge of the distortion, those that assume knowledge of the human visual system and those that assume knowledge of the statistics of natural images

$$MSE(I, \hat{I}) = \frac{1}{MN} \sum_{i=1}^M \sum_{j=1}^N (I(i, j) - \hat{I}(i, j))^2 \tag{8.1}$$

where  $I(i, j)$  and  $\hat{I}(i, j)$  is the value of the pixel at the  $i$ th row and  $j$ th column of the original image and test image, respectively. MSE is a version of the more general Minkowski metric [30]. This measure is simple and directly proportional to the power of the noise signal. As the level of noise increases MSE increases, however, the value of the MSE does not give any indication of the relative visibility of the noise against the image content.

Peak signal-to-noise ratio maps MSE onto a logarithmic scale and scales it relative to the maximum pixel value in the image. For an 8-bit image, the maximum pixel value is 255 and PSNR is given by

$$PSNR(I, \hat{I}) = 10 \log_{10} \left( \frac{255^2}{MSE(I, \hat{I})} \right) \tag{8.2}$$

PSNR has a value of zero when the value of MSE is equal to the largest possible deviation of the noise and approaches infinity as the MSE approaches zero. Unlike MSE, PSNR increases in value with increasing image quality.

Due to their simplicity, PSNR and MSE are still used widely to measure noise in images, especially as baselines to compare new image quality measures [29, 30]. PSNR and MSE do not correlate particularly well with subjective evaluations of image quality [31]. However, there is evidence that more advanced image quality measures do not perform substantially better than PSNR for video quality applications [52]. Both PSNR and MSE suffer from a number of issues. PSNR and MSE do not take into account aspects of the human visual system such as content masking effects, luminance adaptation or the effect of viewing distance [42]. Image distortions that appear very differently to human observers can have identical values of PSNR and MSE [31]. MSE and PSNR do not consider the spatial arrangement of the pixels in the measure, hence ignoring image and noise structure. Noise artefacts such as blocking are particularly visible to human observers due to the fact that the disrupted pixels form regular patterns in the degraded image. This aspect is not captured by PSNR or MSE. On the other hand, a simple shift by a single pixel of the test image relative to the original will produce large values of MSE and reduce PSNR dramatically despite being unnoticeable to a human observer.

### 8.3.2.2 SSIM

Structural SIMilarity Index (SSIM) was presented in the seminal paper by Wang et al. [31]. This error metric bears many similarities with metrics based on natural scene statistics in that it concentrates on the structure of the image being analysed instead of directly modelling either the noise applied to the image or the physiology of human vision. SSIM is based on the concept that human vision has evolved to extract structural information from natural imagery [59]. Humans observe a decrease in image quality when the structural aspects of an image are distorted or masked by noise. For example, blocking artefacts which impose an unnatural regular grid on an image are much more disturbing to human observers than a simple global shift of pixels in an image or a global increase in luminance. The latter two distortions do not affect the overall structure of the image, whereas the blocking artefacts add artificial structure to the image.

In SSIM, image structure is measured by three quantities. The first quantity is designed to compare the luminance difference between two images:

$$l(I, \hat{I}) = \frac{2\mu_I\mu_{\hat{I}} + C_1}{\mu_I^2 + \mu_{\hat{I}}^2 + C_1} \quad (8.3)$$

where  $\mu_I$  and  $\mu_{\hat{I}}$  are the mean values of the original and test image, respectively, and  $C_1$  is a constant to avoid a zero denominator. The second quantity is designed to compare the contrast difference between two images:

$$c(I, \hat{I}) = \frac{2\sigma_I\sigma_{\hat{I}} + C_2}{\sigma_I^2 + \sigma_{\hat{I}}^2 + C_2} \quad (8.4)$$

where  $\sigma_I$  and  $\sigma_{\hat{I}}$  are the standard deviation of values of the original and test image, respectively, and  $C_2$  is a constant to avoid a zero denominator. The third quantity is designed to compare the structure difference between two images:

$$s(I, \hat{I}) = \frac{\sigma_{I\hat{I}} + C_3}{\sigma_I\sigma_{\hat{I}} + C_3} \quad (8.5)$$

where  $\sigma_{I\hat{I}}$  is the covariance between the original and test images and  $C_3$  is a constant to avoid a zero denominator. The final SSIM measure is given by

$$SSIM(I, \hat{I}) = l(I, \hat{I})^\alpha c(I, \hat{I})^\beta s(I, \hat{I})^\gamma \quad (8.6)$$

where  $\alpha$ ,  $\beta$  and  $\gamma$  are adjustable constants that may be adapted to different problems or datasets. In the original paper,  $\alpha = \beta = \gamma = 1.0$  [31]. To adapt SSIM to local distortions in images, SSIM is usually computed as a distortion map with SSIM computed for each pixel with the above summary statistics being computed over an 11 by 11 Gaussian weighted window centred on each pixel. In an earlier version of the metric called the Universal Quality Index (UQI) [60], the constants  $C_1$  and  $C_2$  were set to zero, but this was found to cause instability for very dark or smooth images.

SSIM performs consistently better than PSNR at predicting subjective judgements and is resilient to luminance changes and small shifts that would not be noticed by human observers. Under certain assumptions, SSIM has been found to be able to predict PSNR values for some types of distortions, although PSNR has been found to be more sensitive to Gaussian noise than SSIM [61].

### Extensions to Multi-scale SSIM

The original formulation of SSIM did not consider viewing distance in the formulation of the SSIM metric. To provide robustness to changes in viewing distance a multi-scale version of SSIM, MS-SSIM, was developed [62]. MS-SSIM is computed by performing a dyadic scale wavelet decomposition of the original and test images and computing the contrast (4) and structure (5) SSIM terms at each scale. The SSIM luminance comparison term (3) is only computed at the coarsest scale. The terms at the various stages are then combined multiplicatively by Eq. (8.7):

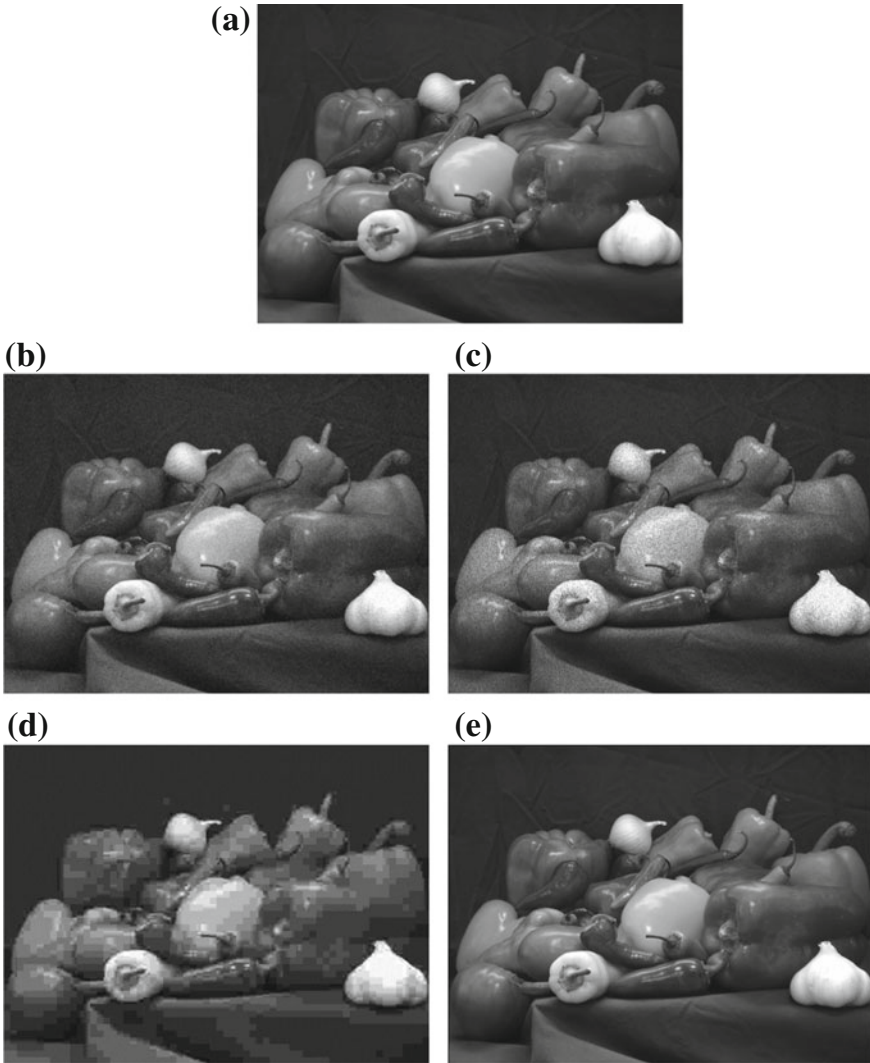
$$SSIM(I, \hat{I}) = l_M(I, \hat{I})^{\alpha_M} \prod_{j=1}^M c_j(I, \hat{I})^{\beta_j} s_j(I, \hat{I})^{\gamma_j} \quad (8.7)$$

where  $M$  is the number of scales in the wavelet decomposition. The coefficients of the contrast and structure terms at each scale,  $\beta_j$  and  $\gamma_j$ , in Eq. (8.7) are allowed to vary from scale to scale, however to simplify the problem,  $\beta_j = \gamma_j$  at each scale. The coefficients at each scale were determined by a subjective experiment involving eight participants [62]. MS-SSIM was found to outperform single-scale SSIM as well as other image quality measures, with the optimal performance obtained when  $M = 2$  [62].

### Extensions to Colour

SSIM in its most common formulation only applies to the luminance component of the image. Image Quality Metrics that rely only on luminance information are still in widespread use. This is partly due to the simplicity of these measures being attractive and the complexity of the human perception of colour [16] being a barrier to extending these measures into the colour domain. In addition, the luminance component of the image often has the most important impact on the perception of image quality [16, 46]. There exist image quality measures that are designed for colour images [43] including metrics based on SSIM [63–66], but often the method of extending luminance only metrics to colour has been to apply the metric on each of the colour planes and simply sum the result [46]. Kolamin and Yadid-Pecht showed that this method of extending luminance measures to colour images failed for certain types of colour degradations inherent to CMOS sensors [46]. Instead of applying the SSIM metric separately to the red, green and blue colour planes, they developed a quaternion implementation of SSIM and termed QSSIM—Quaternion Structural SIMilarity. Quaternions are a generalisation of vectors and provide a way of encoding the three components of the colour of a pixel into a single representation with a set of well-defined mathematical operations for addition and multiplication. Quaternions are better able to represent rotations in multidimensional spaces and Kolamin and Yadid-Pecht claimed that this property allowed QSSIM to better consider the differences in colour compared to previous approaches. They demonstrated an improvement over other implementations of SSIM for colour image noise and distortions for some datasets [46].

Figure 8.3 and Table 8.1 demonstrate some of the differences in behaviour between MSE, PSNR and SSIM. Figure 8.3 shows an 8-bit grayscale original image (a), distorted by four different processes. Image (b) has Gaussian noise of zero mean with variance 0.0015 added. Image (c) has multiplicative speckle noise added of the form  $\widehat{I(x, y)} = I(x, y) + n(x, y)I(x, y)$ , where  $I(x, y)$  is the original image,  $\widehat{I(x, y)}$  is the noisy image and  $n(x, y)$  is a uniformly distributed random variable with mean zero and variance 0.012 generated independently for each pixel in  $I(x, y)$ . Image (d) is a JPEG compressed image with the quality factor set to 5, corresponding to strong compression and image (e) is the original image shifted downwards and to the right by one pixel.



**Fig. 8.3** An 8-bit grayscale image with various distortions applied. **a** Original image, **b** Gaussian noise of variance 0.0015 applied, **c** speckle noise of variance 0.012 applied, **d** JPEG compression with quality factor of 5 applied, and **e** image shifted by one pixel in the x and y directions. All images have approximately the same values of PSNR and MSE, but different values of SSIM as shown in Table 8.1. Image reprinted with permission of The MathWorks, Inc

Table 8.1 shows the MSE, PSNR and SSIM values of the distorted images in Fig. 8.3. It can be seen that the MSE and PSNR values are similar for the four different distortions despite the very different levels of subjective quality of the distorted images to a human observer. To most human observers the translated image

shows very little if any quality loss, whereas the JPEG compressed image would be considered the most degraded image in Fig. 8.3. However, the MSE value of the JPEG compressed image is the lowest of the four degraded images and the PSNR value is the highest. This implies that of the four distorted images, the JPEG compressed image has the best quality. This does not match human observation. The SSIM measure however ranks the translated image as the best quality and the Gaussian image and speckle images as having considerably lower quality. The SSIM measure ranks the JPEG compressed image as having a lower quality than the translated image, but still higher than the Gaussian or speckle degraded images.

### 8.3.3 Standards Related to Noise Measurement

Activities in industry related to noise measurement are often heavily driven by international standards. By following established standards on noise measurement and image quality, imaging companies can ensure a consistent comparison between competing products and services [67]. Standardisation bodies such as the International Organization for Standardization (ISO) [68], Institute of Electrical and Electronics Engineers Standards Association (IEEE-SA) [69], Society of Motion Picture and Television Engineers (SMPTE) [70] and The International Telecommunication Union (ITU) [71] often test potential noise and image quality measures in addition to the testing performed by the original academic authors. Standardisation testing usually directly involves industry participants and occurs across multiple laboratories and hence can often be more rigorous than the testing applied by researchers developing noise and image quality metrics and motivated directly by the concerns of industry participants.

Some groups actively involved in standardisation efforts in the area of image noise and quality are:

- ISO/TC 42 Photography: Standards for still picture imaging including methods for measuring and testing media, materials and devices used in chemical and electronic still imaging [72].
- IEEE-SA—CPIQ—Camera Phone Image Quality: Concerned with the development of subjective and objective test methods for measuring the quality attributes of camera phones [73].

**Table 8.1** Image difference values for the images in Fig. 8.3

Image distortion	MSE	PSNR	SSIM
Gaussian noise	97.1	28.3	0.55
Speckle noise	101.7	28.0	0.64
JPEG compression	85.8	28.8	0.80
Pixel translation	100.9	28.1	0.86

- Video Quality Experts Group (VQEG): Concerned with subjective testing methodologies and objective tool development and verification for video quality assessment [74]. VQEG make recommendations to The International Telecommunication Union (ITU) that may in turn result in ITU recommendations.

Standards groups actively comparing and using noise measurement and image quality measures for natural images, but not concerned with the standardisation of such measures, include:

- Joint Photographic Experts Group (JPEG) ISO/JTC 1/SC29/WG1: Concerned with developing still image data compression standards [75]. Developers of the extremely popular JPEG image-coding standard, this group is expanding its efforts into 3D data structures such as point clouds and light field data, 360-degree imagery as well as continuing to develop the JPEG standard for still images. JPEG make extensive use of subjective and objective still image quality metrics to evaluate compression algorithms.
- Motion Picture Experts Group (MPEG) ISO/JTC 1/SC29/WG11: Concerned with the development of standards for the coded representation of digital audio and video data including extending the standards into new areas such as 3D video and 360-degree video [76]. To evaluate video compression algorithms MPEG uses a variety of video quality metrics including subjective testing.

The groups above and other standardisation groups in this field have been responsible for a wide variety of standards for measuring noise and image quality.

Some of these standards involve the use of test charts to measure noise levels on image capture or display devices. This includes the ISO 15739 [28] standard developed by ISO TC 42 and the P1858 IEEE Standard for Camera Phone Image Quality [77] developed by CPIQ. ISO 15739 makes use of test charts defined in an earlier standard by the same group, ISO 14524:2009 [78].

Jin et al. tested the objective measures derived from test charts as per IEEE P1858 against paired comparison and quality ruler subjective testing [55]. A set of camera phones was tested using the objective measures and test charts defined by IEEE P1858 [77]. The same camera phones were then used to capture natural images and the methods of Paired Comparison [79] and Softcopy Quality Ruler [80] were used to subjectively measure the quality of the natural images captured by the cameras. The objective measures defined in IEEE P1858 were found to correlate well with perceived quality as measured by the subjective methods [55].

The Video Quality Experts Group performed a set of validation experiments with the goal of finding image quality metrics that matched subjective evaluation of television. The first validation experiment (FRTV Phase I) [81] tested full-reference and no-reference objective video quality metrics as predictors of the subjective quality of standard definition television video. The metrics were evaluated on:

- Prediction Accuracy: The degree of the objective measures to predict the subjective mean opinion scores.
- Prediction Monotonicity: The degree of the objective measures to vary monotonically in relation to the subjective mean opinion scores.



- **Prediction Consistency:** The degree of correspondence of the objective measures across different video sequences. This attribute is designed to test the variability of the objective measure to different forms of content.

Ten different image quality measures were tested on a variety of content. All no-reference metrics were withdrawn during the experiment and as a result of the experiment, VQEG recommended to The International Telecommunication Union (ITU) in 2000 that the accuracy of FR models was at that time not sufficiently accurate to justify standardisation [81].

In 2003, Phase II of the validation experiment was performed with six full-reference image quality models tested [82]. Once again, the goal was to validate the ability of objective video quality models to predict the subjective quality of standard definition television. As in Phase I, all no-reference metrics were withdrawn during the experiment and of the six metrics tested, four metrics were recommended to the ITU as being sufficiently accurate to justify standardisation. It was found that while none of the metrics was statistically equal to the theoretically perfect model, the four metrics recommended were statistically equivalent to each other and statistically better than the remaining two metrics [82]. The VQEG recommendations resulted in ITU-T Recommendation J.144 (2004) [83] and ITU-R Rec. BT.1683 (2004) [84].

In 2008, VQEG conducted experiments to validate the ability of Reduced-Reference (RR) and No-Reference (NR) objective video quality models to predict the quality of standard definition television [85]. Once again, all NR models were withdrawn during the course of the experiments. In the final report, VQEG recommended to the ITU that the accuracy of some reduced-reference video quality metrics was sufficient to justify standardisation. This activity resulted in ITU Recommendation J.249 [86] and the standardisation of the implementation of the PSNR metric as a baseline for performance measurement in ITU Recommendation J.340 [87].

There are several standards to describe how subjective testing of various imaging systems should be performed. This includes viewing conditions, test materials, how to present the stimuli and conduct experiments:

- ITU-T P.910 (04/08) is a recommendation on digital video quality assessment for multimedia applications with transmission rates below 1.5Mbit/s [88].
- ITU-R BT.1129 (02/98) is an ITU recommendation that defines methods for subjective assessment of standard definition video sequences [89].
- ITU-R Recommendation BT.500-13 (01/2012) is an ITU recommendation for subjective assessment of television that continues to be one of the key references for conducting subjective assessment of still images [90]. The reader is referred to BT.500-13 for a comprehensive description of a wide range of subjective quality assessment methods and analysis techniques.

In the above recommendations, subjective quality is judged by either a ‘single stimulus’ or ‘double stimulus’ case [90]. In the single stimulus case, the subject sees only the image to be tested and judges the quality on a linear quality scale. In the double stimulus case, the subject is presented with both the original and test images and judges the test image relative to the original image on a linear quality

scale. Quality scores are collected across multiple subjects and averaged for each test image to compute a 'Mean Opinion Score' (MOS). In some cases, the reference image is present in the experiment and is also given an MOS value. In this case, the MOS values of the original image may be subtracted from the MOS values of the test images to compute the 'Difference Mean Opinion Score' that represents the improvement of the test image over the original image [90]. MOS and DMOS scores can be quoted as a measure of noise in an image or video as perceived by a human observer.

## 8.4 Conclusion

A multibillion-dollar industry exists around the capture, processing, analysis and display of digital imagery in the twenty-first century. It is hardly surprising that the issue of image and video noise is of deep concern to the imaging industry. Substantial effort is spent on minimising all the sources of noise in the imaging chain from precise control of the impurities in the silicon used to construct image sensors to advanced algorithms for demosaicing, compression and sharpening designed to minimise visual noise. As image quality improves, consumers have in turn come to expect ever-greater performance from imaging systems, and this has driven a plethora of standardisation activities to measure, reduce and compare noise levels in consumer imaging. Manufacturers of imaging equipment wish to advance image quality standards in the hope that being able to quantify image noise enables them to show consumers that a specific piece of imaging equipment is superior to the competition. These standards need to be backed up with solid psychophysical testing on human subjects across a variety of image content to ensure respect and validity for the standard. With the spread of imaging and product testing review websites and blogs on the Internet, spurious claims to quality by manufacturers are quickly exposed.

As unique image modalities such as 3D scanning, medical imaging and Virtual and Augmented Reality technologies gain an increasing foothold in the consumer imaging market, image noise measures and standards must be extended to these domains. Given that there is still a great deal that we do not understand about the perception of image noise for 2D imagery, the task of measuring, quantifying and reducing noise in 3D imaging technologies is a difficult task for future research. Early efforts have already begun on this problem, including efforts to quantify notions of quality for the purpose of standardising the compression of light field and 3D point cloud information [75, 76]. However, there is still much more work to be done in this field, and plenty of scope to discover more about how humans perceive and experience the visual world.

## References

1. Heyman S (2015) Photos, photos everywhere, New York Times, 29 July 2015. <https://www.nytimes.com/2015/07/23/arts/international/photos-photos-everywhere.html>. Accessed 13 Mar 2018
2. Zhang B, Jenkin R, Denny P (2018) Overview. In: Proceedings of autonomous vehicles and machines 2018, IS&T international symposium on electronic imaging, Burlingame, CA, USA
3. Boncelet C (2009) Image noise models. In: Bovik A (ed) The essential guide to image processing, 2nd edn. Elsevier Science, Burlington, MA, pp 143–167
4. Fiete RD (2010) Digital sensors. In: Modeling the imaging chain of digital cameras. SPIE, Bellingham, Washington, pp 73–97
5. Freund JE (1992) Mathematical statistics, 5th edn. Prentice-Hall, Englewood Cliffs, New Jersey
6. Jain AK (1989) Fundamentals of digital image processing. Prentice-Hall, Englewood Cliffs, New Jersey
7. Barry JR, Lee EA, Messerschmitt DG (2004) Digital communication, 3rd edn. Kluwer Academic Publishers, Norwell, Massachusetts
8. Nyquist H (1992) Thermal agitation of electric charge in conductors. Phys Rev 32(1): pp 110–113
9. Tian H (2000) Noise analysis in CMOS image sensors. PhD dissertation, Stanford University, Stanford, CA, USA
10. Fiete RD (2010) Optics. In: Modeling the imaging chain of digital cameras. SPIE, Bellingham, Washington, pp 49–72
11. Cambridge in colour, understanding camera lens flare Cambridge in colour (2018) <https://www.cambridgeincolour.com/tutorials/lens-flare.htm>. Accessed 22 Jan 2018
12. Photoshop essentials (2018) How to add lens flare to an image with photoshop, Adobe Systems Inc. <https://www.photoshopessentials.com/photo-effects/how-to-add-lens-flare-to-an-image-with-photoshop/>. Accessed 22 Jan 2018
13. Wu G, Masia B, Jarabo A, Zhang Y, Wang L, Dai Q, Chai T, Liu Y (2017) Light field image processing: an overview. IEEE J Sel Top Signal Process 11(7):926–954
14. Lockwood A (2018) Aliasing in digital photography explained, picture correct. <https://www.picturecorrect.com/tips/aliasing-in-digital-photography-explained/>. Accessed 22 Jan 2018
15. Marshall GF, Stutz GE (2012) Handbook of optical and laser scanning, 2nd edn. CRC Press, Boca Raton, Florida
16. Hunt RWG (2004) The reproduction of colour, 6th edn. Wiley, Chichester, West Sussex
17. Zhen R, Stevenson RL (2015) Image demosaicing. In: Celebi E, Lecca M, Smolka B (eds) Color image and video enhancement. Springer International Publishing, Cham, pp 13–54
18. Sergej T, Mantiuk R (2014) Perceptual evaluation of demosaicing artefacts. In: Proceedings of the 2014 international conference on image analysis and recognition (ICIAR2014), Vilamoura, Algarve, Portugal
19. McCann JJ (2000) Color gamut mapping using spatial comparisons. In: Proceedings of the SPIE, color imaging: device-independent colour, color hardcopy, and graphic arts VI, San Jose, CA, USA, 2000
20. Granados M, Aydin TO, Tena JR, Lalonde JF, Theobalt C (2015) HDR image noise estimation for denoising tone mapped images. In: Proceedings of the 12th European conference on visual media production (CVMP'15), London, UK
21. Yuen M (2006) Coding artifacts and visual distortions. In: Wu HR, Rao KR (eds) Digital video image quality and perceptual coding. CRC Press, Boca Raton, Florida, pp 87–122
22. Gonzalez RC, Woods RE (1993) Digital image processing, reading. Addison-Wesley, Massachusetts
23. Roehrig H, Krupinski EA, Chawla AS, Fan J, Gandhi K, Furukawa T, Ohashi M (2003) Noise of LCD display systems. Int Congr Ser 1256(1):162–168
24. ONYX Graphics Inc. (2011) ONYX White Paper: banding issues with wide format printing ONYX Graphics Inc., Salt Lake City, UT, USA

25. International Organization for Standardization (2017) ISO/IEC 24790:2017, information technology—office equipment—measurement of image quality attributes for hardcopy output—monochrome text and graphic images, ISO
26. Naccari M, Mrak M (2014) Perceptually optimized video compression. In: Academic press library in signal processing, vol 5. Elsevier, Kidlington, Oxford, pp 155–196
27. Huang TS, Burnett JW, Deczky AG (1975) The importance of phase in image processing filters. *IEEE Trans Acoust Speech Signal Process* 23(6):529–542
28. International Organization for Standardization (2017) ISO 15739:2017, photography—electronic still-picture imaging—noise measurements, ISO
29. Engelke U, Zepernick HJ (2007) Perceptual-based quality metrics for image and video services: a survey. In: Proceedings of the 2007 next generation internet networks, Trondheim, Norway
30. Thung KH, Raveendran P (2009) A survey of image quality measures. In: Proceedings of the 2009 technical conference for technical postgraduates, Kuala Lumpur, Malaysia
31. Wang Z, Bovik AC, Sheikh HR, Simoncelli EP (2004) Image quality assessment: from error visibility to structural similarity. *IEEE Trans Image Process* 13(4):600–612
32. Meesters L, Martens J-B (2002) A single-ended blockiness measure for JPEG-coded images. *Signal Process* 82(3):369–387
33. Vlachos T (2000) Detection of blocking artifacts in compressed video. *Electron Lett* 36(13):1106–1108
34. Wu HR, Yuen M (1997) A generalized block-edge impairment metric for video coding. *IEEE Signal Process Lett* 4(11):317–320
35. Yap K-H, Guan L, Perry SW, Wong H-S (2009) Adaptive image processing: a computational intelligence perspective, 2nd edn. CRC Press, Boca Raton, Florida
36. Marziliano P, Dufaux F, Winkler S, Ebrahimi T (2002) A no-reference perceptual blur metric. In: Proceedings of the IEEE international conference on image processing, Rochester, NY, USA
37. Marichal X, Ma WY, Zhang HJ (1999) Blur determination in the compressed domain using DCT information. In: Proceedings of the 1999 IEEE international conference on image processing, Kobe, Japan, 1999
38. Caramma M, Lancini R, Marconi M (1999) Subjective quality evaluation of video sequences by using motion information. In: Proceedings of the 1999 IEEE international conference on image processing, Kobe, Japan
39. Pastrana-Vidal RR, Gicquel J-C (2006) Automatic quality assessment of video fluidity impairments using a no-reference metric. In: Proceedings of the international workshop on video processing and quality metrics for consumer electronics, Tempe, AZ, USA
40. Wang Z, Sheikh HR, Bovik AC (2002) No-reference perceptual quality assessment of JPEG compressed images. In: Proceedings of the 2002 IEEE international conference on image processing, Rochester, NY, USA
41. Cavallaro A, Winkler S (2004) Segmentation-driven perceptual quality metrics. In: Proceedings of the 2004 IEEE international conference on image processing, Singapore, Singapore
42. Pappas TN, Safranek RJ (2000) Perceptual criteria for image quality evaluation. In: Bovik AC (ed) Handbook of image and video processing. Academic Press, Burlington, MA, pp 669–684
43. Zhang X, Wandell BA (1997) A spatial extension of CIELAB for digital color-image reproduction. *J Soc Inform Display* 5(1):1938–3657
44. Daly S (1993) The visible difference predictor: an algorithm for the assessment of image fidelity. In: Watson A (ed) Digital images and human vision. MIT Press, Cambridge, Massachusetts, USA, pp 179–206
45. Lubin J (1993) The use of psychophysical data and models in the analysis of display system performance. In: Digital images and human vision. MIT Press, Cambridge, MA, USA, pp 163–178
46. Kolamin A, Yadid-Pecht O (2012) Quaternion structural similarity: a new quality index for color images. *IEEE Trans Image Process* 21(4):1526–1536
47. Mannos JL, Sakrison DJ (1974) The effects of a visual fidelity criterion on the encoding of images. *IEEE Trans Inf Theory* 20(4):525–536

48. Teo PC, Heeger DJ (1994) Perceptual image distortion. In: Proceedings of the 1994 IEEE international conference on image processing, Austin, TX, USA
49. Chandler DM, Hemami SS (2007) VSNR: a wavelet-based visual signal-to-noise ratio for natural images. *IEEE Trans Image Process* 16(9):2284–2298
50. Watson AB (1993) DCT quantization matrices visually optimized for individual images. In: Proceedings of the SPIE, human vision, visual processing and digital display IV
51. Winkler S (2005) Digital video quality—vision models and metrics. Wiley, Chichester, Est Sussex
52. Shiekh HR, Bovik AC, Cormack L (2005) No-reference quality assessment using natural scene statistics: JPEG2000. *IEEE Trans Image Process* 14(11):1918–1927
53. Wang Z, Simoncelli EP (2005) Reduced-reference image quality assessment using a wavelet-domain natural image statistic model. In: Proceedings of the SPIE, human vision and electronic imaging x, San Jose, CA, USA
54. Srivastava A, Lee AB, Simoncelli EP, Zhu SC (2003) On advances in statistical modeling of natural images. *J Math Imaging Vision* 18(1):17–33
55. Jin EW, Phillips JB, Farnard S, Belska M, Tran V, Chang E, Wang Y, Tseng B (2017) Towards the development of the IEEE P1858 CPIQ standard—a validation study. In: Proceedings of the IS&T, image quality and system performance conference XIV, Burlingame, CA, USA
56. Simoncelli EP (1997) Statistical models for images: compression, restoration and synthesis. In: Proceedings of the IEEE Asilomar conference on signals, systems and computers, Pacific Grove, California, USA
57. Buccigrossi RW, Simoncelli EP (1999) Image compression via joint statistical characterization in the wavelet domain. *IEEE Trans Image Process* 8(12):1688–1701
58. Sheskin DJ (2003) Handbook of parametric and nonparametric statistical procedures, 3rd edn. Chapman & Hall/CRC, Boca Raton, Florida
59. Wang Z, Bovik AC, Shiekh HR (2005) Structural similarity based image quality assessment. In: Wu HR, Rao KR (eds) Digital video image quality and perceptual coding. CRC Press, Boca Raton, Florida, pp 225–241
60. Wang Z, Bovik AC (2002) A universal image quality index. *IEEE Signal Process Lett* 9(3):81–84
61. Horé A, Ziou D (2010) Image quality metrics: PSNR vs SSIM. In: Proceedings of the 2010 international conference on pattern recognition, Istanbul, Turkey
62. Wang Z, Simoncelli EP, Bovik AC (2003) Multiscale structural similarity for image quality assessment. In: Conference record of the thirty-seventh Asilomar conference on signals, systems and computers, 2003, Pacific Grove, CA, USA
63. Bianco S, Ciocca G, Marini F, Schettini R (2009) Image quality assessment by preprocessing and full reference model combination. In: Proceedings of the SPIE 7242, Image Quality and System Performance VI, San Jose, CA, USA
64. Lee JH, Horiuchi T (2008) Image quality assessment for color halftone images based on color structural similarity. *IEICE Trans Fundam Electron, Commun Comput Sci* E91.A(6):1392–1399
65. Shi Y, Ding Y, Zhang R, Li J (2009) An attempt to combine structure and color for assessing image quality. In: Proceedings of the 2009 international conference on digital image processing, Bangkok, Thailand
66. Shi Y, Ding Y, Zhang R, Li J (2009) Structure and hue similarity for color image quality assessment. In: Proceedings of the 2009 international conference on electronic computer technology, Macau, China
67. Phillips JB, Eliasson H (2017) Camera image quality benchmarking, Wiley, The Wiley-IS&T Series in Imaging Science and Technology, Hoboken, New Jersey
68. International Organization for Standardization (2018) ISO—international organization for standardization, ISO. <https://www.iso.org/home.html>. Accessed 14 Jan 2018
69. Institute of Electrical and Electronics Engineers (2017) IEEE-SA—The IEEE Standards Association—Home. IEEE. <http://standards.ieee.org/>. Accessed 14 Jan 2018

70. Society of Motion Picture and Television Engineers (2018) We are SMPTE|society of motion picture and television engineers, SMPTE. <https://www.smpte.org/>. Accessed 14 Jan 2018
71. The International Telecommunication Union (2018) ITU: committed to connecting the world, ITU. <https://www.itu.int/en/Pages/default.aspx>. Accessed 14 Jan 2018
72. International Organization for Standardization (2018) ISO/TC 42—photography, ISO. <https://www.iso.org/committee/48420.html>. Accessed 14 Dec 2018
73. Institute of Electrical and Electronic Engineers (2018) IEEE SA—CPIQ—camera phone image quality, IEEE. <https://standards.ieee.org/develop/wg/CPIQ.html>. Accessed 14 Jan 2018
74. Video Quality Experts Group (2018), Video quality experts group (VQEG), ITS. <https://www.its.bldrdoc.gov/vqeg/vqeg-home.aspx>. Accessed 14 Jan 2018
75. The Joint Photographic Experts Group, JPEG (2018), JPEG. <https://jpeg.org/>. Accessed 14 Jan 2018
76. The Moving Picture Experts Group (2018) MPEG|the moving picture experts group website, MPEG. <https://mpeg.chiariglione.org/>. Accessed 14 Jan 2018
77. Institute of Electrical and Electronic Engineers (2016) IEEE Std 1858–2016 (Incorporating IEEE Std 1858-2016/Cor 1-2017)—IEEE Standard for camera phone image quality. IEEE
78. International Organization for Standardization (2009) ISO 14524:2009 Photography—electronic still-picture cameras—methods for measuring opto-electronic conversion functions (OECFs), ISO
79. Gescheider GA (1997) Psychophysics: the fundamentals, 3rd edn. Lawrence Erlbaum Associates, Mahwah, New Jersey
80. International Organization for Standardization (2012) ISO 20462-3:2012 photography psychophysical experimental methods for estimating image quality—part 3: quality ruler method, ISO
81. Video Quality Experts Group (2018) Video quality experts group (VQEG): FRTV phase 1. <https://www.its.bldrdoc.gov/vqeg/projects/frtv-phase-i/frtv-phase-i.aspx>. Accessed 14 Jan 2018
82. Video quality experts group (VQEG) (2018) Video quality experts group (VQEG): FRTV phase II. <https://www.its.bldrdoc.gov/vqeg/projects/frtv-phase-ii/frtv-phase-ii.aspx>. Accessed 14 Jan 2018
83. The International Telecommunication Union (2004) ITU-T recommendation J.144, Objective perceptual video quality measurement techniques for digital cable television in the presence of a full reference, ITU
84. The International Telecommunication Union (2004) ITU-R Rec. BT.1683, Objective perceptual video quality measurement techniques for standard definition digital broadcast television in the presence of a full reference, ITU
85. Video Quality Experts Group (VQEG) (2018), Video quality experts group (VQEG)|RRNR-TV, ITU. <https://www.its.bldrdoc.gov/vqeg/projects/rmr-tv/rmr-tv.aspx>. Accessed 14 Jan 2018
86. The International Telecommunication Union (2010) ITU-T Rec. J.249, Perceptual video quality measurement techniques for digital cable television in the presence of a reduced reference, ITU
87. The International Telecommunication Union (2010) ITU-T Rec J.340, Reference algorithm for computing peak signal to noise ratio (PSNR) of a processed video sequence with constant spatial shifts and a constant delay, ITU
88. The International Telecommunication Union (2008) ITU-T P.910 (04/08), Subjective video quality assessment methods for multimedia applications, ITU
89. The International Telecommunication Union (1998) ITU-R BT.1129 (02/98), Subjective assessment of standard definition digital television (SDTV) systems, ITU
90. The International Telecommunication Union (2012) ITU-R BT.500 (01/2012), Methodology for the subjective assessment of the quality of television pictures, ITU

# Chapter 9

## Noise Characteristics and Noise Perception



Tamara Seybold

**Abstract** Denoising is a traditional but still challenging problem in signal processing. To reduce the noise in images and videos captured by a digital sensor receives more and more attention also due to the shrinking size of today's image sensors and striving for even higher resolutions. A vast amount of research has been conducted to solve the complex problem of separating noise from the true signal. The widespread assumption of additive white Gaussian noise (AWGN) in readily processed image data, however, has led to algorithms that fail on real camera data. This shows how crucial the underlying assumptions and the considered quality metrics are to reach results that are convincing on real data and for real people. In this chapter, we will discuss the properties of real camera noise from sensor data up to human perception. First, we will address how test data is generated and review the noise characteristics of a real single sensor camera. Real camera noise is fundamentally different from AWGN: it is spatially and chromatically correlated, signal dependent, and its probability distribution is not necessarily Gaussian. Second, the challenging aspects of evaluating denoising results based on metrics will be addressed. Instead of rating an algorithm based on a metric like PSNR, which is still the metric the latest benchmarks are based on, a more meaningful metric is required. We show our results of different perception tests that investigated the visibility of spatiotemporal noise as it occurs in digital video. Including these results into a perceptual metric could enable a reliable denoising evaluation with respect to the human perception of visual quality.

### 9.1 Beyond Standard Noise Models: Noise and Denoising in a Real Camera Processing Pipeline

Digital cameras, even the highest quality products, still are limited by sensor saturation in the highlights and by noise in low-light conditions. Capturing in low-light conditions has become even more difficult with increased sensor resolution: when

---

T. Seybold (✉)  
ARRI, Türkenstr. 89, 80799 Munich, Germany  
e-mail: tseybold@arri.de

© Springer International Publishing AG, part of Springer Nature 2018  
M. Bertalmío (ed.), *Denoising of Photographic Images and Video*,  
Advances in Computer Vision and Pattern Recognition,  
[https://doi.org/10.1007/978-3-319-96029-6\\_9](https://doi.org/10.1007/978-3-319-96029-6_9)

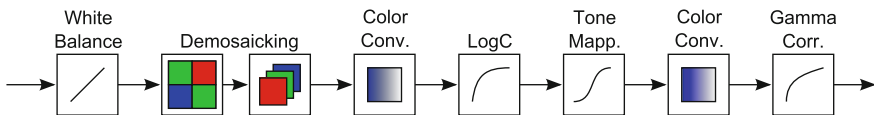
235

the physical size remains constant, a resolution increase leads to a pixel size decrease. The amount of light trapped by a smaller pixel is lower and, therefore, the signal-to-noise ratio decreases. As a high noise level decreases the visual quality of color images and can reduce the efficiency of subsequent image processing tasks, eliminating noise using algorithmic methods is necessary.

The denoising problem has been studied extensively and various methods have been developed [4, 9, 17, 41, 45, 57]. The best available denoising methods show an impressive increase in peak signal-to-noise ratio (PSNR) on most standard datasets and reach almost optimality in terms of image fidelity [5–7, 38, 40, 54]. However, this does not mean that the visual quality is optimal [22, 58]. The PSNR usually indicates a successful reduction of noise, but it does not show how unnatural and disturbing the denoising artifacts may appear to the viewer.

Most denoising methods are designed and evaluated based on a standard dataset—e.g., the Kodak dataset—and a standard noise model—usually additive white Gaussian noise (AWGN). This model does not correspond to the noise in real-world image or video data taken with a digital camera, and therefore, denoising based on this model leads to clearly suboptimal results on camera data.

To understand the difference, let us review the color image capture via a digital camera, which is the usual method of image capture nowadays. One pixel captures the irradiance; thus the sensor data corresponds linearly to the brightness at the pixel position. To capture color data, a color filter array (CFA) is used, which covers the pixels with a filter layer. Thus, the output of the sensor is a value that represents the amount of light for one color band at one-pixel position (linear sensor data,  $I_{Bayer}$ ). This linear sensor data cannot be displayed before additional steps are applied (see Fig. 9.1). Mandatory in every camera processing pipeline, we have the white balance and the so-called debayering, which generates a full-color image (linear RGB,  $I_{full-color}$ ). To transform this linear image to monitor displayable image, additional color transformation are needed. A minimal color transformation would be a color space conversion to, e.g. sRGB and a gamma transformation. This, however, is not sufficient for high image quality. We present the workflow published by Andriani et al. [2]. For this pipeline, the color transformations include first a color space conversion from camera to a wide gamut color space, a logarithmic transformation called LogC, subsequently the tone mapping and the gamma correction. The color transformations are required to transform the linear data to displayable monitor data



**Fig. 9.1** The camera processing pipeline from raw sensor data to a display domain image. The pipeline starts with the sensor output, followed by the white balance and the demosaicking. These two steps are mandatory in every camera processing. The color conversions transform the linear data to monitor gamma and color space. In many cameras, nonlinear curves (here LogC and tone mapping) additionally adapt high dynamic range data for standard monitors



adapted to the monitor's gamma and color space, but the exact type of transformation may differ. All the steps described lead to a noise characteristic that is fundamentally different from the usually assumed AWGN: the noise is signal dependent in the raw data; through debayering, it becomes spatially and chromatically correlated; and after the nonlinear color transformations, the noise distribution no longer corresponds to a Gaussian distribution. The workflow discussed here is based on the processing presented for a motion picture camera. However, as cameras for consumer market also reach higher and higher dynamic range and resolution, we expect our findings and methods also to be applicable to other signal-sensor camera data.

Different approaches can be used to tackle the problem of spatially correlated camera noise: either apply the denoising to the raw sensor data, before debayering leads to spatial correlation, or take the correlation into account. Despite the vast number of proposed denoising methods for AWGN, only a few try to employ one of these strategies.

Applying denoising to the raw sensor data requires a signal-dependent noise model. Denoising methods for images containing signal-dependent noise have been studied using two different approaches. The first is to include the signal-dependent noise characteristic in the denoising methods. The second strategy is to apply a transformation of the input signal to a signal with approximately constant variance [18, 19, 23]. Including a signal-dependent noise model, however, is not enough: at this level, the raw data is mosaicked in real single sensor camera images, so that denoising methods must be adapted accordingly. There are different approaches of denoising mosaicked data using the traditional noise model [8, 10, 24, 61, 63], which are mostly trained on a standard dataset. Results for demosaicking real camera data [2] show that algorithms trained on the standard dataset need to be adapted to signal characteristics of camera data. Hence, besides the noise model and the missing neighboring values, denoising algorithms for raw camera data have to cope with the linear signal values.

A combined study of the noise characteristics in images taken with a single sensor camera must consider all three aspects: signal dependence, debayering, and the signal domain. We will evaluate the noise characteristics with respect to the signal domain and work out the differences between the real camera noise in digital color images and the traditional noise model. To evaluate the impact of these differences, we compare the visual quality of noisy images using both noise models based on a subjective test, which enables us to discuss the impact of the noise model on human perception. Additionally, we show how the noise characteristic influences denoising results.

The structure of this chapter is as follows. First, we show the camera processing pipeline in Sect. 9.2: we start with the basic transformations that are required to present image data captured with a digital sensor correctly on a monitor (color space conversion and gamma transformation), and second present the nonlinear transformations that are used additionally for rendering images that contain a relatively high dynamic range (logarithmic transformation and time mapping). Section 9.3 discusses the camera noise in the raw domain. In Sect. 9.4, we show the difference of the noise in the raw domain to the display domain. We describe the processing of the raw images and the relevance of the processing steps to the noise characteristics.

In Sect. 9.5, we then evaluate the noise characteristics after the processing, in particular, the spatial correlation in the display domain. The visual perception of different noise types is analyzed in Sect. 9.7 based on a set of subjective tests. In Sect. 9.6, we apply denoising to a simulated video sequence with different noise types, thus we can evaluate the effect of different noise characteristics on denoising results.

## 9.2 Camera Processing Pipeline: From Sensor Data to Monitor-Ready Image Data

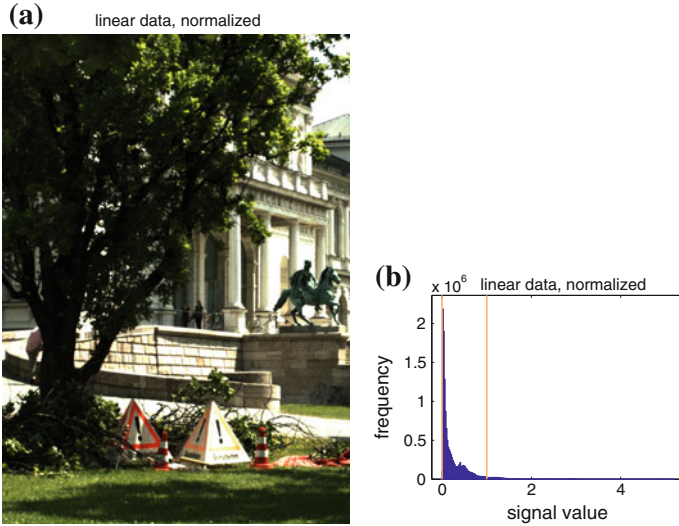
The sensor data contains only one value per pixel, which is green, red, or blue depending on the location of the pixel. The most usual CFA is the Bayer pattern. To obtain a full-color image with three color values per pixel, an interpolation method is applied, called color demosaicking or debayering.

$$I_{Bayer}(n_1, n_2) = \iint_V \int_{\lambda} L_c(\lambda, n_1, n_2) r(u, v, \lambda) h(n_1 - u, n_2 - v) d\lambda dudv + N(n_1, n_2) \quad (9.1)$$

the data captured by a digital sensor can be described as a function of the incident radiance  $r_c$ , which depends on the spectral response  $L_c$ , which is defined by the IR and UV cut-off filters, the color filters for R, G, and B and the sensitivity of the sensor's photo site. The resulting image pixel values at the spatial sampling location  $n_1, n_2$  additionally depend on the spatial response  $h$ , which is mainly determined by an optical low-pass filter and the sensor's pixel pitch. We include an additive noise term  $N$ .

The task of debayering is to reconstruct the full-color data  $I_{full-color}$  from the observed sensor data  $I_{Bayer}$ . Andriani et al. [2] presented an evaluation of a camera-optimized method compared to methods trained on artificial test images and introduced a new dataset of real images. This dataset includes a test image with a real full-color reference, which is obtained by a monochromatic sensor equipped with alternating color filters for every captured image. Thus, the above-mentioned  $I_{full-color}$  is available as a reference. The evaluation shows that the quality of the debayering depends strongly on the data it was optimized for. We, therefore, first have a more detailed look on camera data characteristics.

The sensor output  $I_{Bayer}(n_1, n_2)$  depends linearly on the incident light and cannot be displayed directly. This is especially severe when the camera output is of high dynamic range compared to what the display is capable of. Such a linear image directly displayed appears mostly black, only the highlights are visible. When the image is normalized according to the exposure, the main part of the image content can be displayed (Fig. 9.2a). The histogram is scaled such that a certain range of signal values fall inside the display values, however, highlights are clipped and black tones are densely packed, visible as a peak in Fig. 9.2b.



**Fig. 9.2** Image example (a) and histogram (b) of a linear image. After normalization to the exposure index the main part of the signal values is fitted in the signal range [0, 1], which is indicated by the yellow lines

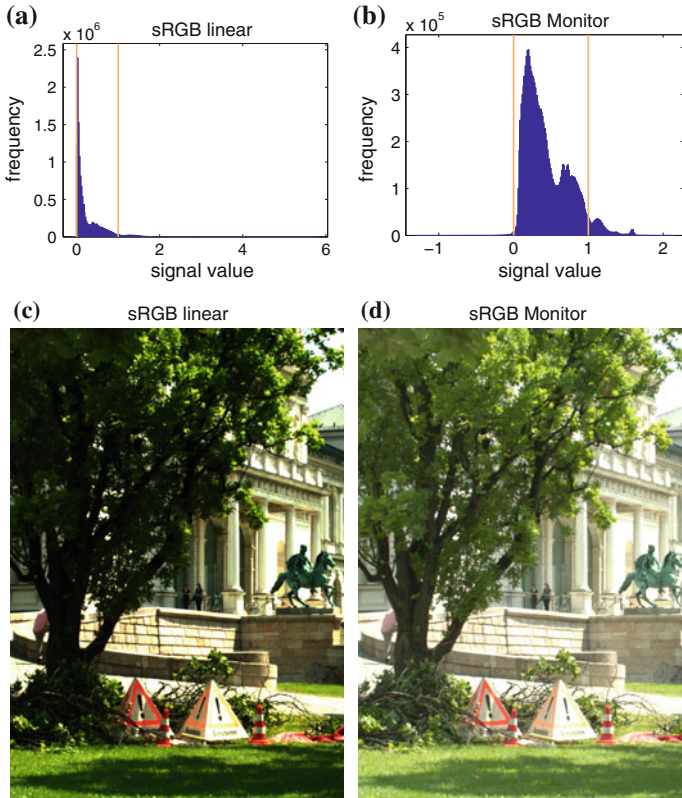
The normalized output data  $I_n$  is calculated from  $I_{raw}$  by adjusting the range from x-bit-range to [0, 1] and correction depending on the exposure index EI with a multiplying factor. This normalization scales the values such that middle gray captured using an 18%-reflectance gray card, is mapped to the RGB values (0.18, 0.18, 0.18).

$$I_n = \frac{18EI}{400} \cdot \frac{I_{raw} - 256}{65,535 - 256} \tag{9.2}$$

The next step is the reconstruction of the color values. To convert the color from camera RGB ( $R_c, G_c, B_c$ ) to the usual sRGB monitor color space a color matrix conversion is applied. First, the data is converted into a wide gamut color space ( $R_{wg}, G_{wg}, B_{wg}$ ), whose primaries are chosen to avoid clipping in all but the most extreme cases. Subsequently, the conversion from the wide gamut color space to the color space defined in the ITU Recommendation 709 [25], which has the same primaries as sRGB.

The output sRGB images are displayed in Fig. 9.3c, the histogram in Fig. 9.3a. The last step missing for a monitor image is the compensation for the nonlinear electro-optical conversion function (EOCF) of the display device (gamma transformation).

Figure 9.3d shows the final display domain image after these two basic steps described: color space conversion and gamma transformation. Although this image has been converted correctly using the basic conversion steps, in the very bright areas

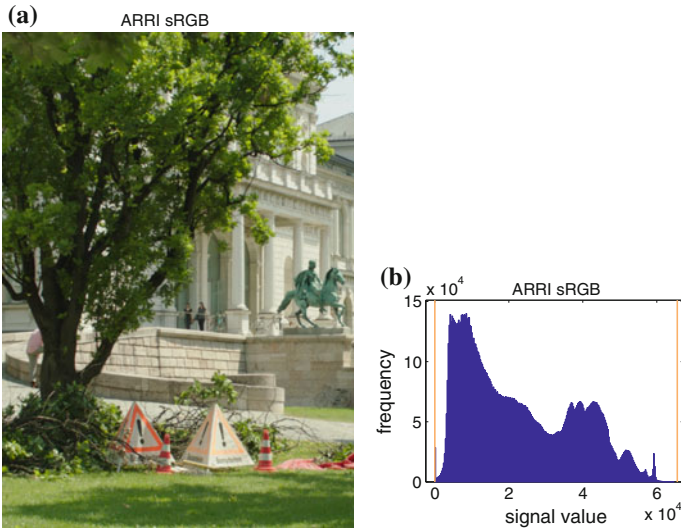


**Fig. 9.3** If the color space conversion is applied directly to the linear data, linear sRGB data is obtained (a, c). This image visualizes that the color space conversion is not enough: although the color values are corrected, the sRGB values additionally need a compensation for the monitor gamma. The result of this gamma compensation is shown in (b, d)

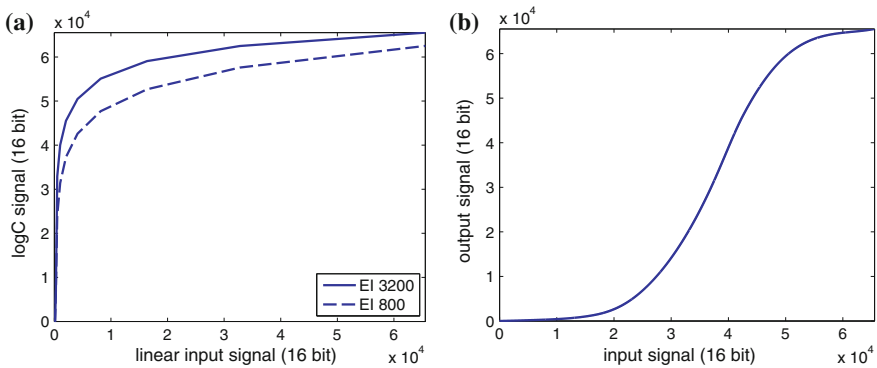
the highlights are clipped. This may also be observed in the histogram of the output image in Fig. 9.3b.

This basic processing, therefore, is not adequate for high dynamic range data delivered by today’s image sensors. Figure 9.4 shows the image when transformed using the camera processing procedure published in [2], which—additionally to the basic processing steps—includes nonlinear transformations to better convert the full image information of the higher dynamic range linear image into the low dynamic range display domain representation.

The first nonlinear curve is a logarithmic transformation, called LogC transformation, which is applied in the wide gamut color space. The LogC image is a format that can be captured with most ARRI cameras. Other camera manufacturers have similar logarithmic formats that allow preserving most image information but already provide visible image content when displayed on a monitor. The format was designed



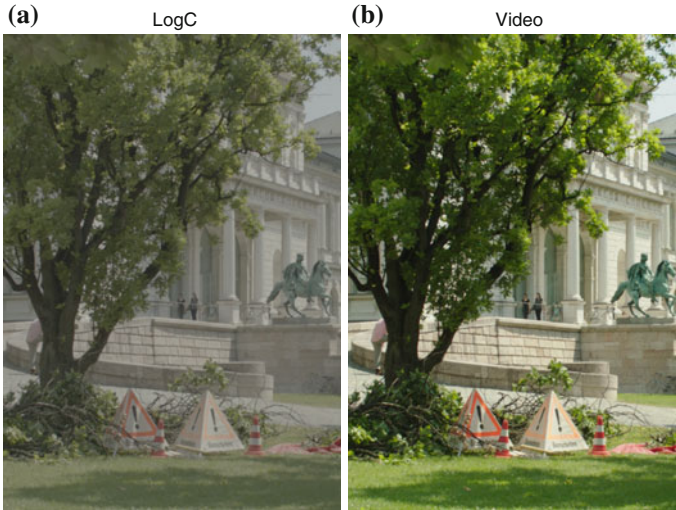
**Fig. 9.4** Monitor data, image example, and histogram. With a nonlinear tone mapping function, more pleasant shadows and more details in the highlights can be preserved



**Fig. 9.5** **a** Shows the LogC curve for two different exposure indices plotted with linear x-axes. **b** Shows the tone mapping curve

to match the characteristics of negative film: it is linear to the exposure over a wide range of signal values. The LogC curve, shown in Fig. 9.5a, hence, transforms the linear data into a representation that is linear to the exposure around the middle of the signal range. To avoid clipping and to keep the information at the borders of the signal range, the curve flattens depending on the chosen exposure index.

The second nonlinear curve is the tone mapping, applied on the LogC data. Tone mapping is long known to be an important factor in reaching high photographic image quality. It compresses the highlights and shadows and provides a steep slope in the main signal range, which leads to a high-contrast monitor image [53].



**Fig. 9.6** LogC image (a) and the final tone mapped monitor image (b)

The tone mapping curve, shown in Fig. 9.5b, has a steep slope to increase the contrast matching the display-specific range and compresses the highlights and shadows to avoid clipping.

An example of a LogC image is shown in Fig. 9.6 on the left, the resulting display domain image is shown on the right.

### 9.3 Camera Noise Characteristics

To measure the camera noise in the raw images at different signal levels, we take a series of exposures with the ARRI Alexa camera. To measure the camera noise, we use the photon transfer method [1]. Two frames  $A$  and  $B$  are used to calculate the variance as the sum of the squared differences in the active area of size  $N \times M$ .

$$\sigma^2 = \frac{1}{2NM} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} (A_{ij} - B_{ij})^2 \quad (9.3)$$

The curve in Fig. 9.7a shows the variance plotted over the respective mean value. The signal value is the digital 16-bit value of the sensor output, which represents the light intensity level. The variance of the sensor noise can be approximated by a linear curve. This result matches the results with other cameras in [55]. In Fig. 9.7b, we show the distribution at a fixed signal level. The distribution is very similar to the Gaussian distribution. That means we can well approximate the sensor noise using a Gaussian distribution with signal-dependent variance.  $x_n = x + n$  with  $n \sim \mathcal{N}(0, \sigma(x))$  and

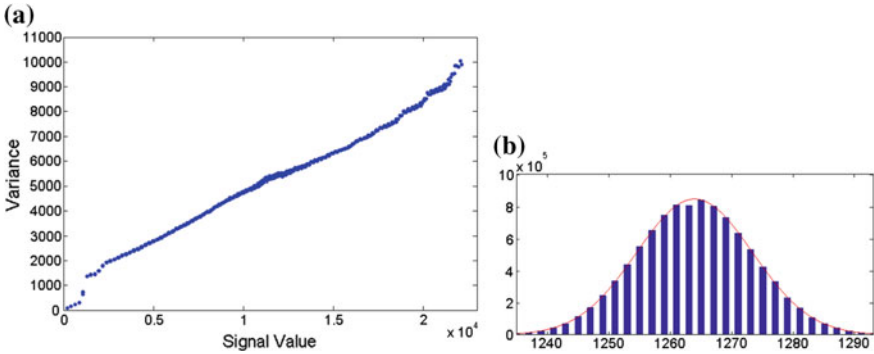


Fig. 9.7 Variance and distribution of sensor noise

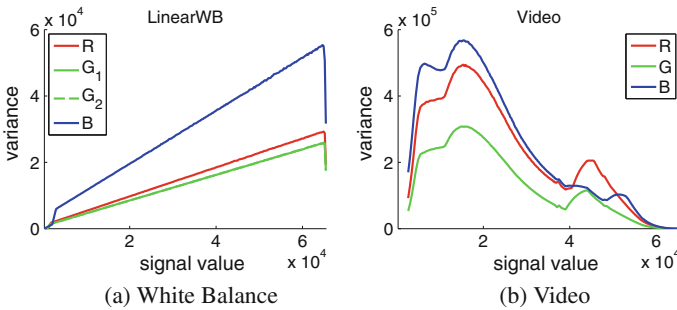


Fig. 9.8 The noise variance over mean signal value is plotted for raw data after white balancing at 3200K (left) and for display-optimized monitor data at EI 800 (right)

$\sigma(x) = \sqrt{mx + t}$  where  $m$  is the slope and  $t$  the intercept of the linear approximation of the curve in Fig. 9.7a.

As we have Bayer data at this point, the noise level, of course, depends on the signal value of one color. With different signal values of R, G, and B, the variance of neighboring pixels is quite different depending on the color in the image (Fig. 9.8).

### 9.4 Camera Noise in the Processing Pipeline

As explained in the introduction, the raw data is in the linear domain, i.e., the signal value is proportional to the amount of light collected by the sensor. When processing the sensor output, multiple steps are performed to achieve a monitor color image. The steps are

1. White Balance,
2. Debayering,
3. Color Transformations.



Applying these steps gives a displayable image, but they also influence the sensor noise.

White balance is a known gain factor  $g_c$  different for each color. It directly influences the noise  $n_c$  in different colors.

$$n_c = g_c n \sim \mathcal{N}(0, g_c \sigma(x)) \quad (9.4)$$

The debayering step creates three color values by interpolation using the pixel and the neighbor values. Therefore, a spatial and chromatic correlation of the three color channels is introduced. The spatial correlation of the noise  $n_{deb}$  in the debayered image is usually disregarded in common state-of-the-art denoising methods.

The third step, the color transformations, is a nonlinear tone mapping and color space conversion to map the linear values to displayable signals. Color transformations can strengthen the spatial and chromatic correlation.

Figure 9.10b shows the noise distribution after color transformation: We observe a more compact distribution around the mean value and longer tails. The color transformations, however, are an individual choice and this distribution can vary over the signal range. Hence, we cannot expect a Gaussian distribution in real camera data.

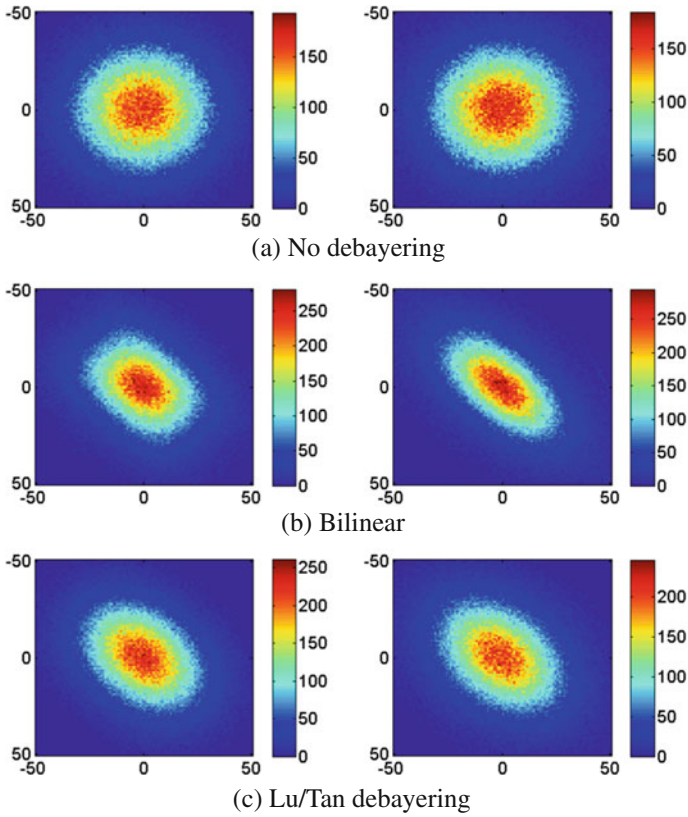
## 9.5 Local Correlation Introduced with Debayering

When looking at the noise power spectrum (R channel) of digital camera noise, Fig. 9.10a shows that real camera noise is not white: due to the spatial correlation introduced with debayering the noise is more prominent in lower noise frequencies than in high frequencies. To evaluate the influence of debayering for different methods, we first use the standard test images from Kodak. This choice is since most debayering algorithms in the literature are optimized using that test set.

The spatial correlation of the noise is evaluated after debayering a noisy and a noise-free image. The difference image contains the error introduced by the noise. We use this difference image to calculate the correlation matrix  $C$  and a scatter plot. We compare different debayering algorithms and their influence on the noise characteristics.

To visualize the distribution, we plotted 2-dimensional histograms of the noise. The difference of the noisy and the reference image is the error due to noise. In the 2D histogram, the densities of two neighboring pixels are plotted. The color represents the density; the position in the plot represents the value of the error. The resulting scatter plots for an image with AWGN without debayering are given in Fig. 9.9a. The distribution is symmetric as expected for uncorrelated noise. Figure 9.9b, c show the scatter plots of the noise after debayering using bilinear interpolation and using the debayering method proposed by Lu/Tan [39]. The scatter plots of the debayered noise are dispersed into the diagonal direction, which indicates a correlation: it is more likely for a pixel to have the same or a similar value as its neighbors (Fig. 9.10).



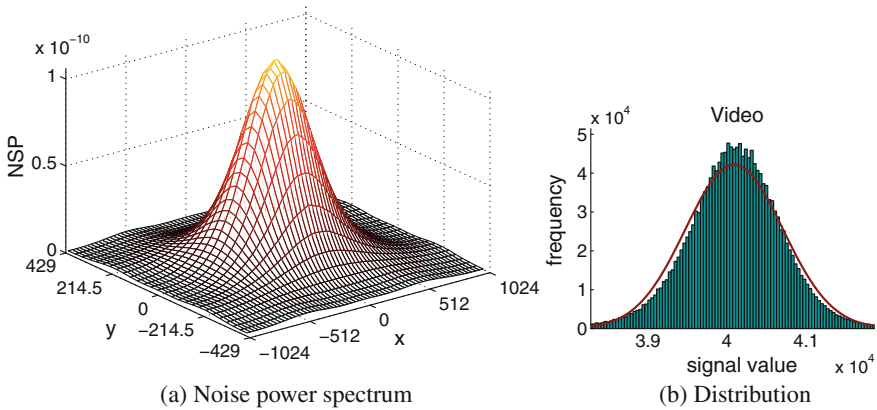


**Fig. 9.9** Scatter plot of the noise in a noisy Kodak image before (a) and after debayering (b, c); the first Kodak image was used and AWGN with  $\sigma = 20$  added. From left to right, the G and B channel. The R channel is not shown, because as the debayering methods interpolate the B and R channels the same way and the scatter plots of B and R look very similar

In Sect. 9.4, we discussed the influence of processing on the noise. An additive model for the noisy image  $I_n$  is to be the sum of the image  $I$  and the noise  $n$ . For denoising applications, it is quite usual to assume white Gaussian noise,

$$I_n = I + n; \quad n \sim \mathcal{N}(0, \sigma) \tag{9.5}$$

spatially independent with a fixed variance  $\sigma$ . This assumption is far from real camera noise, which is correlated spatially and signal dependent. We calculated the correlation matrices  $C$ , given in Table 9.1. The matrices contain the correlation between a pixel  $(i, j)$  and its neighbors; more precisely the entry  $(k, l)$  in the matrix corresponds to the correlation of the neighbor pixel  $(i + k, j + l)$ . Numbers for bilinear interpolation, DLMMSE [62], and Lu und Tan method [39] are given. The matri-



**Fig. 9.10** The noise power spectrum (R channel) of digital camera noise (left) shows that real camera noise is not white. On the right is displayed the noise distribution after color transformation: we observe a more compact distribution around the mean value and longer tails

**Table 9.1** Correlation matrices for different debayering methods, calculated using AWGN and the Kodak dataset. While in the usual uncorrelated noise model the noise values do not correlate with the neighboring values and thus lead to a matrix with only a one in the upper left matrix position, these correlation matrices show that a strong correlation to the neighboring pixels is present after debayering

$C_{bilinear} =$ $\begin{bmatrix} 1 & 0.5693 & 0.1243 \\ 0.5715 & 0.3214 & 0.07181 \\ 0.1284 & 0.07425 & 0.0191 \end{bmatrix}$	$C_{dlmmse} =$ $\begin{bmatrix} 1 & 0.2239 & -0.0066 \\ 0.375 & 0.06351 & 0.03626 \\ -0.003771 & 0.03715 & 0.03545 \end{bmatrix}$	$C_{lu} =$ $\begin{bmatrix} 1 & 0.3173 & 0.01578 \\ 0.2947 & 0.1007 & 0.04422 \\ 0.01689 & 0.04548 & 0.04233 \end{bmatrix}$
---	--	--

ces are  $3 \times 3$  because numbers outside of this region are very small. The bilinear interpolation has the strongest correlation.

We propose to approximate the noise after debayering by a multivariate Gaussian distribution with a covariance matrix  $\Sigma$ .

$$I_n \approx I + n; \quad n \sim \mathcal{N}(0, B \Sigma_d B^T) \tag{9.6}$$

with  $\Sigma = B \Sigma_d B^T$

We expect the expression to be separable with a diagonal matrix  $\Sigma_d$ , whose entries linearly depend on the corresponding pixel intensities (c.f. Fig. 9.7a) and a matrix  $B$  depending on the spatial correlation introduced by the debayering. For linear debayering methods, the above approximation is exact.

## 9.6 Influence on Denoising

To answer the question what impact the difference between the traditional independent noise and the camera noise has, we want to evaluate the effect on denoising in this section. We already showed that the spatially correlated noise is more disturbing, thus reduces the visual quality. We thus can expect that the visual quality is also reduced with debayering when rating the denoising results. To evaluate if there is also an effect on the denoising error, we use the PSNR, as it directly measures the error of the denoised image quantitatively.

The PSNR of the denoising results is shown in Table 9.2. We compare signal-dependent camera noise with a variance as in Fig. 9.7a, to Gaussian noise added in display domain. The variance of the Gaussian noise was fixed with the objective of a similar visual impression in display domain. We process the frames in two ways: without debayering (called “RGB” in Table 9.2) and with debayering (“dem.” in Table 9.2). All these cases then are denoised, either in linear domain or in display domain (sRGB). Two different denoising algorithms are used: BLS-GSM [45] and BM3D [9]. The parameters are picked to obtain the most visually pleasing results. We calculated the PSNR of each frame and took the mean over 20 frames.

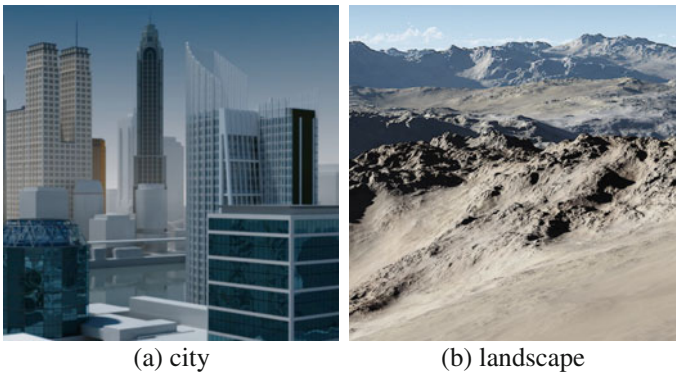
While the visual quality in the last section was shown to be lower with debayering, the PSNR of the noisy sequence shows that the debayering has a slight denoising effect, which leads to a higher PSNR of the debayered noise. However, the PSNR of denoised images is up to 8 dB lower when the debayering is included and thus the noise is correlated. Hence, denoising is significantly harder due to the spatial correlation in the debayered images. BLS-GSM denoising brings lower PSNR improvement for the signal-dependent noise, thus it shows similar to the noisy case a slightly higher PSNR with the demosaicked noise. BM3D seems to be more robust to the signal dependence, for both it leads to an improvement of above 10 dB in PSNR, but it seems more sensitive to the spatially correlated noise as the improvement of 5.54 dB for AWGN and 2.82 dB for signal-dependent noise is much smaller.

**Table 9.2** PSNR results for denoising the city sequence with BM3D [9] and BLS-GSM [45]. Two noise types are used: Gaussian noise added in display domain (AWGN) and signal-dependent camera noise added in linear domain (SD). Denoising is either performed in linear domain (lin.) or in display domain (sRGB)

	Noisy		BLS-GSM		BM3D	
	RGB	dem.	RGB	dem.	RGB	dem.
AWGN/sRGB	35.51	35.35	44.47	38.81	46.33	38.17
SD/sRGB	31.51	33.95	36.45	37.43	43.11	39.49
SD/lin	31.51	33.95	39.40	36.61	41.99	38.34

BLS-GSM works better in linear domain, we think because the noise characteristics differ less from the assumed model. In contrast, the similarity-based method, BM3D, works better in display domain. Hence, denoising methods must be tested on linear domain data explicitly and it depends on the methods if denoising in the linear domain or in the display domain works better. In total, the results show that it is very important to adapt the standard methods to the correct noise model.

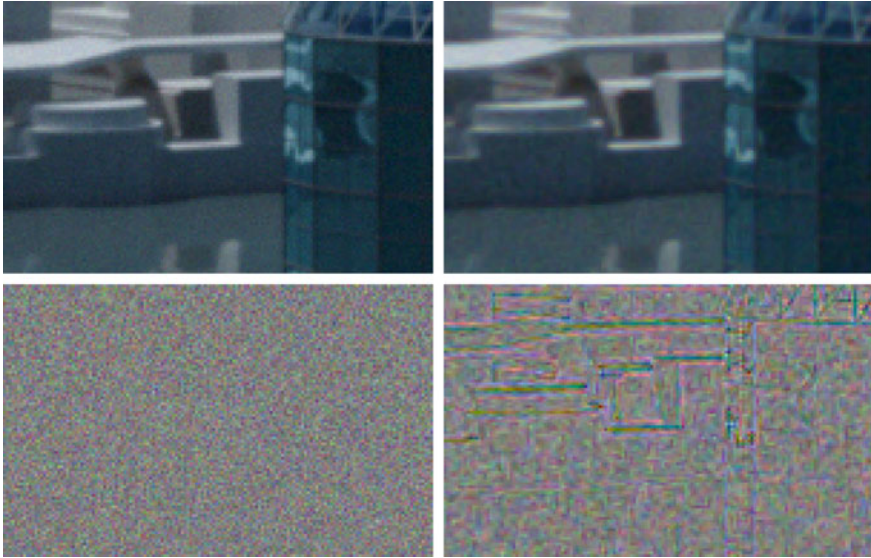
Besides the noise model, also the signal domain of the images is important to incorporate in the development of denoising algorithms. The very successful BLS-GSM method shows severe artifacts when applied to linear data of a high dynamic range scene (Fig. 9.12). This is not special for the algorithm, we obtain artifacts around highlights with many denoising methods optimized on standard datasets, because they were not optimized for linear high dynamic range data (Figs. 9.11, 9.13 and 9.14).



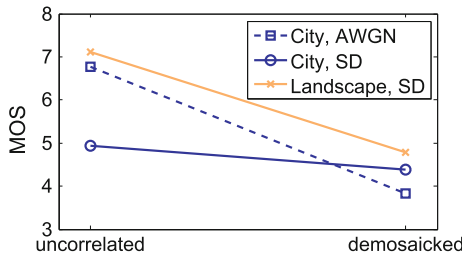
**Fig. 9.11** The computer-generated test sequences

**Fig. 9.12** The image crops of the arch sequence from [2] denoised with BLS-GSM demonstrate that denoising optimized for standard test data can lead to strong artifacts when applied on high dynamic range camera data





**Fig. 9.13** Crop of the sequence “City”. Noisy image (left) and noisy image with debayering (right). In the second row, the respective difference image  $I_d$  scaled for display ( $I_{d,scaled} = I_d \cdot 4 + 128$ )

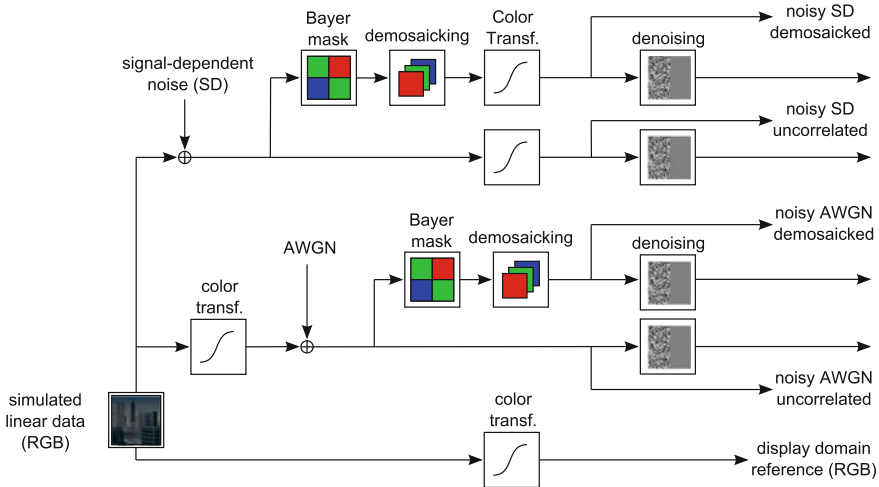


**Fig. 9.14** The MOS results for the test sequences “City” and “Landscape” using the traditional AWGN model (dashed) and the realistic signal-dependent noise (solid lines). The uncorrelated noise, processed without debayering, is shown on the left, the results with debayering on the right

### 9.7 Visual Perception of Spatially Correlated Noise

Building on the discussed characteristics of realistic camera noise in the last sections, we now study the human perception of the traditional model compared to the more realistic camera noise model. We use computer-generated video sequences combined with a simulation of the camera parameters.

The “city” sequence is a pan over a city, see Fig. 9.11a. The frames are rendered in high resolution and in linear signal domain. To incorporate the optic of a camera system, the images of the sequence are multiplied in the Fourier domain with the optical transfer function of the camera. This step takes into account the diffraction-limited



**Fig. 9.15** Processing of simulated sensor data for the test using signal-dependent noise (SD) and AWGN

lens, the optical low-pass filter and the pixel aperture, as described by Schöberl et al. [52]. The videos used in a test to compare the different noise types are simulated with the signal-dependent camera noise added before the white balance, which is a realistic noise model, and compared to the traditional model, AGWN added on standard images that are processed for display. The camera noise is added in accordance to the measured values in Sect. 9.3 using a Gaussian distribution with a signal-dependent variance defined by the linear approximation of the measurement data. This gives us simulated raw images with a reference. The processing for the different noise types is visualized in Fig. 9.15.

To compare the effect of debayering on the noise characteristic, the processing with debayering is compared to the processing without debayering. The last one is only possible with the simulated RGB values; in real raw data, the debayering cannot be omitted. A crop of the city sequence with noise and with/without debayering is shown in Fig. 9.13. The degraded video (noisy, with and without debayering) is compared to the reference. The difference of one frame from the noisy video compared to reference frame, shown in Fig. 9.13, visualizes the effect: When debayering is included the noise is structured and of coarser grain. The difference images are scaled the same way to be comparable. We also observe that the correlated noise after debayering appears more colorful. This may be caused by the lower frequency of the signal. As the maximum of the luminance contrast sensitivity is in higher frequencies, the color might be perceived stronger for a low-frequency signal.

To obtain reliable information about the human perception of the different noise characteristics, we perform a subjective test with 18 participants. We used the double stimulus impairment scale (DSIS) methodology with an undistorted reference and impaired noisy sequence according to ITU-R BT.500. The observer sees the reference

video, subsequently, the impaired video, and afterward is asked to rate the second video on an impairment scale. A discrete scale from 1 to 10, representing an impairment range of “very annoying” to “imperceptible”, was used. The task for the participants was to assess the perceived impairment of the videos. The test was performed in the ITU-R BT.500 compliant video quality evaluation laboratory at the Institute for Data Processing at Technische Universität München. For displaying the videos, a color calibrated Sony BVM-L230 reference LCD display with a screen diagonal of 23 in. was used. To get reliable results, the outliers were removed in the post processing of the subjects’ votes. Votes were removed, if they deviated more than  $2\sigma$  from the mean for a sequence. Using this criterion, 4.6% of all votes were discarded. After outlier removal, the mean opinion score (MOS) was determined for the different test videos.

The MOS provides reliable values for the subjective quality of the test videos. Four different noise models were used in our test: the usual AWGN model, AWGN with debayering, signal-dependent noise without debayering, and finally the realistic camera noise model—signal-dependent noise with debayering. Based on the MOS, we evaluate the visual quality of the noisy test sequences and analyze the main differences between the realistic camera noise and AWGN: spatial correlation introduced through debayering and signal dependence.

The spatially correlated noise is perceived as more annoying. While the MOS is different depending on image content and noise type, Fig. 9.14 shows a lower MOS for the all the demosaicked sequences compared to the sequences with uncorrelated noise. This may be due to the higher visibility of spatially correlated noise, which shows coarser grain and appears more colorful. The MOS of the city sequence with AWGN is about 3 scores lower when debayering is included. Regarding the sequences with signal-dependent noise, the MOS is 0.5 lower for the city sequence and 2.3 lower for the landscape sequence when debayering is included. We thus showed the significant effect of the noise characteristic on the visual perception of color video sequences: the spatial correlation of the noise decreases the perceived image quality.

Our test results show that the noise characteristics have a significant effect on the image quality perception. We, therefore, conclude that a more detailed study on the perception of noise in different frequency bands could be helpful to obtain a more detailed understanding of human perception.

## 9.8 Noise Perception in Videos

Based on the analysis of noise in digital videos presented in the first part of this chapter, we know that the noise of a real camera is spatially correlated due to debayering. When denoising is applied the high spatial frequencies in the image frame are usually even more suppressed and low-frequency noise remains in each image. This low-frequency noise is not visible in still images, due to the fall-off in the contrast sensitivity for low spatial frequencies. If the images are part of a video sequence, this low-frequency noise reappears as flickering in the video sequences and this effect is



very disturbing. As the noise in the low-frequency bands is very difficult to separate from real image content this is still a common quality issue in video denoising.

While temporal denoising methods can better reduce flickering, they can introduce new artifacts, especially motion artifacts, and they come with high computational cost, especially memory requirements are extremely high for the current high-resolution data (4k and beyond).

To the best of our knowledge, no study is available that gives details about the visibility of noise in video sequences depending on the spectral distribution of the digital video noise. Winkler and Süssstrunk [60] presented a subjective study examining noise visibility in still images. Besides white noise, they also used mid-frequency noise and high-frequency noise. The results show the lower noise visibility for the high-frequency noise compared to the mid-frequency noise, as it would be expected from the contrast sensitivity function. We expect that the same effect could have been shown for low-frequency noise, as the contrast sensitivity also falls off towards low frequencies. Unfortunately, this has not been the subject of study.

To obtain more detailed information about noise visibility, we study the visibility of noise for eight different frequency bands. We evaluate both the visibility of static noise, as it occurs in still images, and dynamic noise, as it is present in video data. We start with giving a short overview of the literature about spatial and temporal contrast sensitivity of human vision. We then describe our approach: We explain how we obtain the different noise patterns, which can be displayed on a standard monitor. The results of a test with 22 observers for a static noise pattern and a dynamic (spatiotemporal) noise are subsequently presented. Subsequently, we present an additional test, allowing us to compare the noise visibility for a standard frame rate (24 fps) and a higher frame rate (48 fps) and discuss all our results.

### ***9.8.1 Related Work***

Early technical achievements like movies and discharge lamps provoked early experiments on temporal effects of human vision. Temporal contrast sensitivity and explaining the effects using Fourier analysis was already studied almost 60 years ago, by De Lange [11–15], Kelly [31, 33] and Roufs [47–51]. An overview of the early experiments is given by Kelly in 1977 [35].

The first measurements for combined spatiotemporal sensitivity were presented by Robson in 1966 [46]. He determined the thresholds for four spatial frequencies and for four temporal frequencies. The lowest frequency was 1 Hz, which is considered to be equivalent to static results (Van Nes results [56] indicate thresholds a little bit above 0 Hz). While the spatial CSF measured for static patterns shows a band-pass characteristic, the sensitivity function for the same spatial frequencies, measured with spatial patterns that are temporally varying, is a low pass. The spatiotemporal contrast sensitivity is hence not separable, it shows a clearly more complicated shape than could be obtained by the product of spatial and temporal CSFs and Kelly in 1966 mentioned effects not explainable by a separable model [32]. He measured the



CSF for spatiotemporal stimuli (traveling waves) [36] and the results are similar to Robson's: for 2 Hz the band-pass shape holds, the frequencies 13.5, 17, and 23 Hz suggest a low-pass structure; Chromatic CSF curves in [34, 37].

The experiments for CSF measurements are conducted with sine waves. While this has become the standard procedure to measure CSF curves, we, in this chapter, want to evaluate the sensitivity to noise of different spatial frequencies, as it can occur in video sequences. Thus, our test setup is not directly comparable to former CSF measurements.

Some studies examined the visibility of signals when noise is added [3, 20, 21]. However, for the application of video quality assessment and denoising evaluation, it is more important to study the visibility of the noise.

Most applications that use spatiotemporal CSF models still rely on this data, e.g., the perceptual quality metric by Winkler [59]; other perception-based metrics are designed for still images and, therefore, only rely on the spatial CSF, e.g., [30, 43] and the image difference metric by Perdersen [44]. Nadenau et. al included a masking model for perception-based image compression, however, the algorithm is designed for still images and not for video [42].

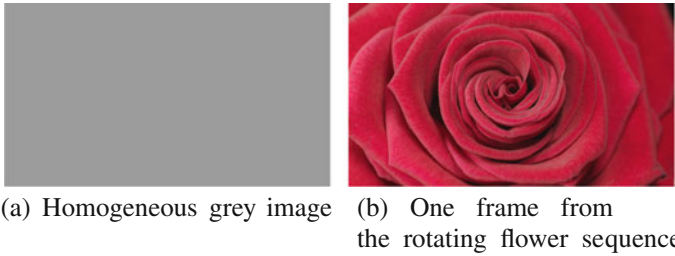
A masking experiment was conducted by Winkler and Süsstrunk; they measured noise visibility on 30 test images. The visibility of noise in natural images was evaluated for three types of noise: white noise and two band-pass noise patterns of medium- and high-frequency bands [60]. Their work provides details for noise visibility in still images, but this type of experiment is still missing for temporal noise.

## 9.8.2 Noise Visibility Test

We investigate the visibility of noise of different spatial frequency bands in still images and in video sequences (we use 24 fps video sequences). To that end, we conduct a subjective test. Since the content of the background image may have a significant impact on noise perception, we select a homogeneous gray sequence and a rotating rose sequence. Both are displayed in Fig. 9.16. We include two types of test patterns in the test: static (spatial) noise patterns and dynamic (spatiotemporal) noise patterns. Eight spatial frequency bands were used for each pattern type.

### Test Pattern Generation

The test patterns were obtained by first generating white noise and subsequent band-pass filtering, which is done by cutting the desired frequency band in the Fourier domain. To test the perception of luminance noise, we used the IPT color space and added the noise to the luminance channel (I channel). IPT is an opponent color space that was developed by Ebner and Fairchild [16] to create a space that is perceptually uniform. The transformation includes the monitor model (gamma transformation).



**Fig. 9.16** Chosen sequences for the experiment

The workflow for the noise pattern generation is described in the following.

1. First, 2-D zero-mean white noise is generated by MATLAB's `randn` function.
2. The noise is transformed to the Fourier domain and a band-pass filter is used to cut a defined band from the spectrum.
3. Each pixel row of the image is multiplied by a factor that increases logarithmically from top to bottom, to obtain noise with increasing variance (i.e., contrast).
4. A uniform gray image, with a constant pixel value of 0.6104 (in range [0–1]) is generated in sRGB space.
5. The gray image is transformed to the IPT space.
6. The zero-mean noise is finally added to one of the channels of the IPT image.
7. The noisy image is transformed back to sRGB.

The static noise patterns correspond to the noise in still images, while the dynamic noise patterns correspond to the noise in video sequences. For the dynamic noise measurements, we generated 360 static noise patterns, which are displayed as a sequence with 24 fps in the dynamic noise experiments.

For each frequency band, one sequence is generated. We did the measurements for eight frequency bands between 0.13 and 13.59 cpd. Table 9.3 shows the frequency bands of the noise test patterns.

### 9.8.2.1 Test Session

We conduct a subjective test to find out the noise level at which noise is visible in a digital video. The experiment is divided in four test parts:

- Uniform gray video with static noise patterns.
- Static rose video with static noise patterns.
- Uniform gray video with dynamic noise patterns.
- Rotating rose video with dynamic noise patterns.

The noise level in each sequence is gradually increasing for each frame and each sequence lasts 15 s. We included numbers into the frames corresponding to the duration from 1 to 15 s (Fig. 9.17).

**Table 9.3** The frequency bands used in the test are given in pixels per period and cycles per degree (cpd)

Nr. of pixels per sine period		Spatial frequency in cpd	
Range	Mean	Range	Mean
332.6–706.9	519.75	0.08–0.17	0.13
166.3–332.6	249.45	0.17–0.34	0.26
82.0–166.3	124.15	0.34–0.69	0.52
40.7–82.0	61.35	0.69–1.39	1.05
19.4–40.7	30.05	1.39–2.91	2.15
9.7–19.4	14.55	2.91–5.82	4.37
4.9–9.7	7.3	5.82–11.64	8.74
3.6–4.9	4.25	11.64–15.53	13.59



(a) First frame from a sequence (b) Last frame from a sequence

**Fig. 9.17** The visibility of noise on first and last step

Before starting the experiment, a short test for visual acuity and color blindness was conducted, for visual acuity the “tumbling E” chart and for color blindness the Ishihara plates were used. After this test, written and oral instructions were provided. Additionally, one dummy sequence was presented as a demonstration and the distance between the observers and the screen was corrected. The participants of the test watch the sequences and observe at which second the noise first becomes visible. The observers were free to choose where to look at, because this is the realistic use case for video viewing.

Each sequence is repeated once to let the participant get a more precise observation. Between those repetitions, there is a short break (3 s) during which noiseless gray sequence with the text “Repeating...” is displayed. In order to avoid leaving an afterimage the text is placed in the bottom right corner of image. In addition, there is another break lasting 10 s before every new noise type, allowing the participants to write down their observation and preventing the noisy afterimage from damaging the vision of the next sequence. Furthermore, we added a progress bar and the number of the next sequence into this break sequence to make sure the observer does not miss the beginning of the new noise type. Thus, for each frequency band, the observer will watch a test sequence of duration 43 s (10 s + 15 s + 3 s + 15 s). There are 8 frequency bands for static and dynamic noise types, a total of 32 test sequences.

According to the ITU-R recommendations for the subjective assessment of quality of television pictures [27], visual experiments should not last more than 30 min, since

the experiment can be very tiring. In addition, at the beginning of each test part about three training presentations should be introduced to stabilize the observer's opinion.

We introduce two training sequences at the beginning of each test part and each sequence is repeated once. Between the first play and the repetition, there is a short break of 3 s. After the repetition, there is a break lasting 10 s before the next noise type, allowing the participants to have enough reaction time. Considering the training sequences one test part lasts about 7 min, therefore, the complete experiment lasts approximately 28 min.

Furthermore, a random order was used for the presentations in order to eliminate contextual effects.

### 9.8.2.2 Test Setup Settings

The experiment was conducted in the video quality evaluation laboratory of the Institute for Data Processing at the Technical University of Munich in a room compliant with recommendation ITU-R BT.500 [27]. The tests were done with a professional broadcast monitor (Sony BVM-L230) set to ITU-R BT.709 color space. The monitor has a 10-bit serial digital interface (SDI). The images used for the experiment had a resolution of  $1920 \times 1080$  pixels.

According to the ITU-R recommendations BT.2022 [26], the distance between the screen and the observers should be about three times the picture height for full HD data in BT.709 [28] color space. The height of the reference display is 30 cm, therefore, the distance used in the experiment was 90 cm.

Additionally, the contrast of the display should be adjusted using a photometer. The display luminance value was  $70 \text{ cd/m}^2$ , according to the ITU-R BT.814 recommendation [29]. The ratio of background luminance behind the monitor to peak luminance of the picture was around 0.15, as recommended by ITU-R BT.500 [27].

### 9.8.3 Results

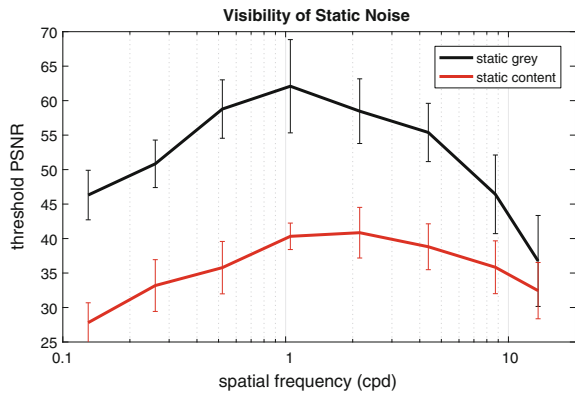
As the peak signal-to-noise ratio (PSNR) is a widely used metric in video applications, we use threshold PSNR levels to illustrate our results. We calculate the average PSNR value for each step (15 steps correspond to 15 s). After that, we equate each observer's input for each sequence with the corresponding step's PSNR value. This means, when an observer sees the noise relatively late, the threshold PSNR will be large. Therefore, the threshold PSNR represents how sensitive an observer is to the presented noise video.

Based on the data of 22 observers, we calculate the average threshold PSNR and thereby compare the visibility of static and dynamic noise on uniform and natural (content) images. We additionally plotted error bars indicating the standard deviation of the results.

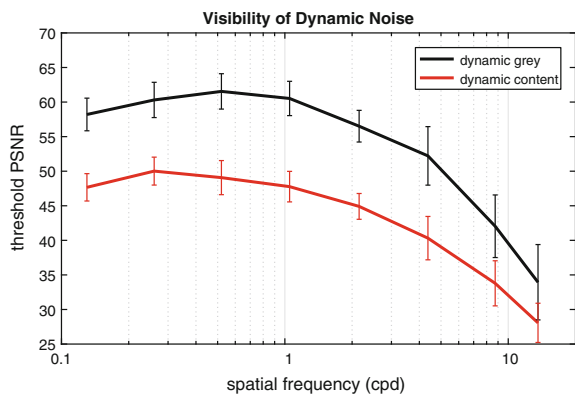
Figure 9.18 shows the curves of the PSNR threshold for the eight spatial frequencies used in the experiment. The observed sensitivity in gray images shows a peak in the mid-frequency 1.05 cpd and in content image in the mid-frequency 2.15 cpd. Both curves show band-pass characteristics. There is, however, a significant difference between both curves, in natural (content) images the threshold level of noise visibility is much lower than in the homogeneous gray videos. This discrepancy between noise visibility in natural and uniform videos can also be observed in the results for visibility of dynamic noise.

The results for the dynamic noise, shown in Fig. 9.19, are significantly different from the static noise. Both curves show a rather low-pass shape than a band-pass. The peak is at lower spatial frequencies for both sensitivity curves. We see a smooth peak at 0.52 cpd for the gray sequences and a peak at 0.26 cpd for the flower sequences.

**Fig. 9.18** Result of the test with the static noise patterns, error bars show the standard deviation of the results



**Fig. 9.19** Result of the test with the dynamic noise, which is a sequence of newly generated static noise patterns with 24 fps. Error bars show the standard deviation of the results



## 9.8.4 Discussion

### 9.8.4.1 Comparison of Static and Dynamic Noise Visibility

First, we will discuss the test results for static and dynamic noise visibility in gray and flower test parts.

The results for all four test parts are shown in Fig. 9.20. As already described, the visibility curves of dynamic and static noise are significantly different, for the gray images as well as for the rose images. The threshold level of noise visibility for the content images is much lower compared to the plain gray images. This confirms that the content is masking the noise, this means that noise is significantly less perceivable depending on the background it is applied on. We conclude that noise is more perceivable in the uniform videos than in the natural videos.

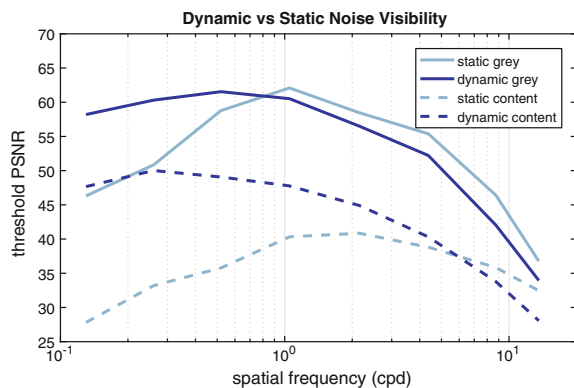
Comparing the visibility of static and dynamic noise, we observe that low-frequency noise is more perceivable when the noise is dynamic, as in video sequences. This difference is significant for the uniform image, but it is considerably larger in content videos. This means our results can be expected to be very relevant for real video processing applications.

#### HFR Experiment

In future, video frame rates might be higher than 24 or 30 fps. We, therefore, conducted an additional experiment evaluating the difference in noise visibility of 24 and 48 fps sequences.

We used the same test setup as described above. As the video frame rate could not be switched during the experiment, we displayed the complete test in 48 fps. The 24-fps content was simulated using a 48-fps sequence showing the exact same image twice. A different monitor had to be used to display 48 fps sequences in full HD resolution (EIZO CG318-4K). The monitor was calibrated before the experiment and the luminance levels were measured to meet the requirements described above. To reduce the length of the test, this time three spatial frequencies for the noise were selected (low, mid, and high). Twenty observers completed the test.

**Fig. 9.20** Comparison of test results for dynamic and static noise in gray and content sequences



**Fig. 9.21** Comparison of test results for dynamic noise in gray and content sequences displayed with 24 and 48 fps. Three frequency bands were used: “low” corresponds to 0.5 cpd, “mid” corresponds to 8.35 cpd, and “high” corresponds to 26.25 cpd

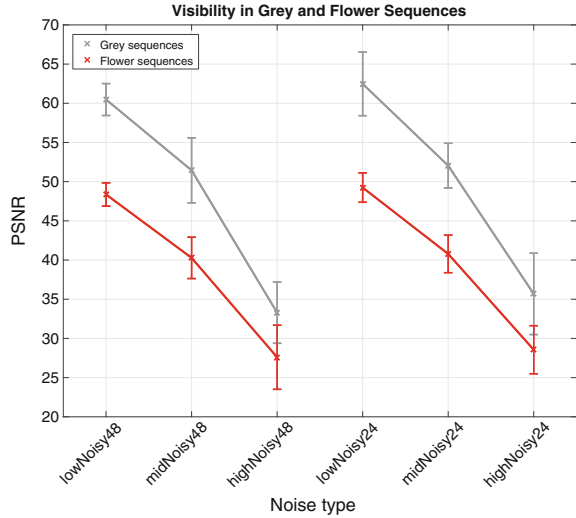


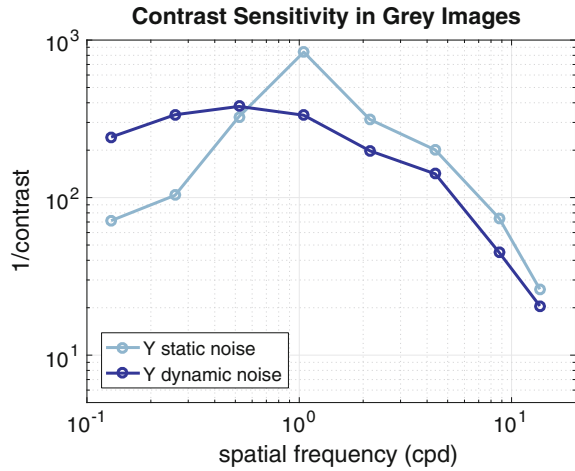
Figure 9.21 shows the results. For all three frequencies, the noise visibility is very similar for 48 and 24 fps sequences. The noise visibility hence does not decrease significantly with higher frame rates than 24 fps. As stated in the introduction, higher resolution of image sensors leads to higher noise, because less light is trapped by the photo sites. As higher frame rates require shorter exposure time, this additionally reduces the light trapped by one pixel, and hence increases noise. Therefore, we can conclude that noise will continue to limit video quality in high-resolution and high frame rate video data shot in low- light conditions.

### 9.8.4.2 Comparison of Test Results to the Contrast Sensitivity Function

Due to the clear fall-off of the temporal contrast sensitivity above 10 Hz (Robson [46] and Kelly [36]), a flickering grating displayed at 24 Hz should lead to lower sensitivity than a static pattern. However, our results for the dynamic noise show the higher sensitivity, hence the contrary. In Figs. 9.22 and 9.23, we replotted our previously shown results in linear luminance to make them better comparable to other publications. The contrast is calculated using RMS contrast, i.e., the standard deviation divided by the mean luminance.

First, we will discuss the results of the homogeneous gray test part in Fig. 9.22. The absolute values of the contrast sensitivity function in gray images with static noise are comparable to the values reported by Robson [46] and Kelly [36]. In the low-frequency range from 0.13 to 0.52 cpd, the sensitivity is higher for dynamic noise compared to static noise. This matches the severe differences we see in the visual quality of spatial denoising algorithms when they are applied to motion picture data compared to their application on still images. In the mid and high-frequency bands

**Fig. 9.22** The contrast sensitivity was calculated in XYZ color space for better comparison with other results. Plotted here, is the contrast sensitivity of luminance noise (Y-channel) results of the static and dynamic noise patterns in gray images



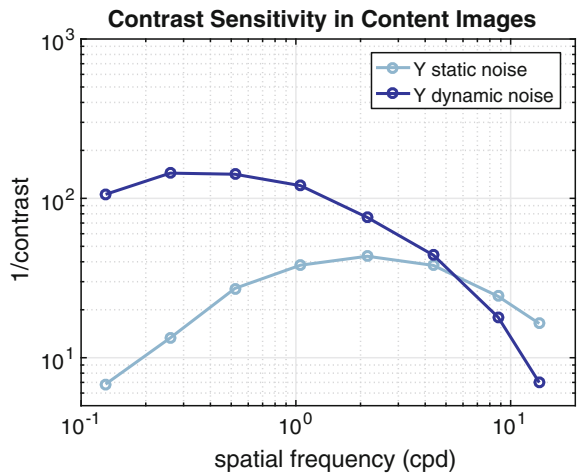
from 1.05 to 13.59 cpd the sensitivity to the dynamic noise is lower than to the spatial noise, but the difference is not large.

Whereas, gray background allows the study of noise visibility without influence of image content, the question of noise visibility in real-world video sequences is even more relevant for practical applications.

Figure 9.23 shows the contrast sensitivity in the XYZ color space for the flower test part. We observe that the absolute values of contrast sensitivity function in the flower images are lower compared to the contrast sensitivity function in the gray images.

In the low- and mid-frequency range from 0.13 to 4.37 cpd, the sensitivity is clearly higher for dynamic noise compared to static noise. As for the gray sequences, this

**Fig. 9.23** Contrast sensitivity of luminance noise (Y-channel) results of the static and dynamic noise patterns in flower images





matches the severe differences in the visual quality of spatial denoising algorithms, when they are applied to motion picture data compared to their application on still images. In the high-frequency bands around 8.74 and 13.59 cpd, the sensitivity to the dynamic noise is lower than to the spatial noise, the difference gets larger for the highest spatial frequency.

The main result, that the low-frequency band noise is clearly more visible in the dynamic noise test, matches our visual impression, which is that the dynamic noise intensity seems to increase toward lower spatial frequencies.

Our findings show the significant differences in noise visibility from still to video data, which up to now are not respected in video processing algorithms as denoising, or video quality assessment. While our findings show a tendency that might explain quality issues and help in improving some of the image processing algorithms for video data, a detailed model would be needed to include precise noise visibility information for perception-based video processing algorithms. This would require more tests and a more detailed study on spatiotemporal masking, which is beyond the scope of this paper.

## 9.9 Summary and Conclusion

Camera noise in raw data can be modeled as a Gaussian distribution with signal-dependent variance, but the camera noise characteristic in display domain images is very different from the noise in the raw domain. The noise is color channel and signal dependent in a nonlinear fashion. Additionally, spatial and chromatic correlation is introduced after the debayering step. Correlation matrices were calculated for different debayering methods using standard test images and the effect is demonstrated in frequency domain: real camera noise is not white.

To demonstrate how important it is to use a correct noise model, we evaluate the impact of the noise characteristic on the denoising performance using a computer-generated test sequence that includes the optical characteristics of the camera. Camera noise is compared to the traditional model and the influence of debayering is studied. We can conclude that the spatially correlated noise is perceived as more annoying: in our subjective test with 18 participants the visual quality was rated significantly lower for the sequences containing correlated noise compared to sequences with AWGN and equal PSNR level. This result may be explained by a lower visibility of high noise frequencies.

Denoising was applied to the same test data, and we showed that the PSNR of the denoising results is up to 8 dB lower with correlated camera noise. Hence, the noise characteristic in the image has a significant effect on both visual perception and on denoising results. To account for the correct noise model is thus very important to achieve high image quality in future research. Instead of applying denoising on RGB data, one could also think of applying it directly on the sensor data. The sensor data, however, represents digital signal values linear to the amount of light. Examples of denoising methods that show excellent results on sRGB data show, that also the

data characteristics have a significant influence and denoising methods need to be optimized for both data and noise characteristics.

Driven by the visual comparison of noise with different characteristics, especially lower frequency noise that occurs after debayering and denoising, we evaluated the noise visibility over a range of spatial frequencies. We measured the visibility of noise for static (spatial) noise, which occurs in still images, and for dynamic (temporal) noise, which occurs in video data. We obtain three main results.

First, the contrast sensitivity of spatially low-frequency noise is significantly higher when the noise is temporally varying. This can explain why algorithms designed for still images might not show high-quality results on video data, e.g., denoising algorithms for still images that do not eliminate low-frequency noise, because it is not visible in still images.

Second, we showed that the noise visibility strongly depends on the image or video content. Our results show that the noise is significantly more perceivable in uniform images than in natural images, which can be explained by masking. In addition to that, the above-mentioned difference between noise visibility of static and dynamic noise is considerably larger for our natural image content than for the gray image. That means, that the observed difference in noise visibility is important to consider for improving video processing algorithms.

Third, an additional experiment evaluated the influence of the frame rate on noise visibility by comparing 24 and 48 fps. The results show that the noise visibility does not decrease significantly for 48, compared to 24 fps. As stated in the introduction, higher resolution of image sensors lead to higher noise, because less light is trapped by the sensor. Higher frame rates require shorter exposure time, which also reduces the light trapped by one pixel, and hence increases noise. Therefore, we can conclude that noise will continue to limit video quality in high resolution and high frame rate video data shot in low-light conditions.

Further research on noise visibility and developing metrics that include this knowledge is therefore crucial to allow developing and improving denoising algorithms for high visual quality.

## References

1. (2010) EMVA 1288, standard for characterization of image sensors and cameras
2. Andriani S, Brendel H, Seybold T, Goldstone J (2013) Beyond the kodak image set: a new reference set of color image sequences. In: ICIP, pp 2289–2293. <https://doi.org/10.1109/ICIP.2013.6738472>
3. Blackwell KT (1998) The effect of white and filtered noise on contrast detection thresholds. *Vis Res* 38(2):267–280. [https://doi.org/10.1016/S0042-6989\(97\)00130-2](https://doi.org/10.1016/S0042-6989(97)00130-2)
4. Buades A, Coll B, Morel J (2005) A review of image denoising algorithms, with a new one. *Multiscale Model Simul* 4(2):490–530
5. Chatterjee P, Milanfar P (2010) Fundamental limits of image denoising: are we there yet? In: 2010 IEEE international conference on acoustics speech and signal processing (ICASSP). IEEE, pp 1358–1361

6. Chatterjee P, Milanfar P (2010) Is Denoising dead? *IEEE Trans Image Process* 19(4):895–911. <https://doi.org/10.1109/TIP.2009.2037087>
7. Chatterjee P, Milanfar P (2011) Practical bounds on image denoising: from estimation to information. *IEEE Trans Image Process* 20(5):1221–1233. <https://doi.org/10.1109/TIP.2010.2092440>
8. Condat L (2010) A simple, fast and efficient approach to denoising: joint demosaicking and denoising. In: *ICIP*, pp 905–908
9. Dabov K, Foi A, Katkovnik V, Egiazarian K (2007) Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Trans Image Process* 16(8):2080–2095. <https://doi.org/10.1109/TIP.2007.901238>
10. Danielyan A, Vehvilainen M, Foi A, Katkovnik V, Egiazarian K (2009) Cross-color BM3D filtering of noisy raw data. In: *2009 international workshop on local and non-local approximation in image processing*, pp 125–129. <https://doi.org/10.1109/LNLA.2009.5278395>
11. De Lange Dzn H (1952) Experiments on flicker and some calculations on an electrical analogue of the foveal systems. *Physica* 18(11):935–950. [https://doi.org/10.1016/S0031-8914\(52\)80230-7](https://doi.org/10.1016/S0031-8914(52)80230-7)
12. De Lange Dzn H (1954) Relationship between critical flicker-frequency and a set of low-frequency characteristics of the eye. *J Opt Soc Am* 44(5):380–388. <https://doi.org/10.1364/JOSA.44.000380>
13. De Lange Dzn H (1958) Research into the dynamic nature of the human fovea-cortex systems with intermittent and modulated light. I. Attenuation characteristics with white and colored light. *J Opt Soc Am* 48(11):777–783. <https://doi.org/10.1364/JOSA.48.000777>
14. De Lange Dzn H (1958) Research into the dynamic nature of the human fovea-cortex systems with intermittent and modulated light. II. Phase shift in brightness and delay in color perception. *J Opt Soc Am* 48(11):784–787. <https://doi.org/10.1364/JOSA.48.000784>
15. De Lange Dzn H (1961) Eye's response at flicker fusion to square-wave modulation of a test field surrounded by a large steady field of equal mean luminance. *J Opt Soc Am* 51(4):415–421. <https://doi.org/10.1364/JOSA.51.000415>
16. Ebner F, Fairchild MD (1998) Development and testing of a color space (IPT) with improved Hue uniformity. *Color Imaging Conf 1998*(1):8–13
17. Elad M, Aharon M (2006) Image denoising via learned dictionaries and sparse representation. *CVPR* 1:895–900
18. Foi A (2009) Clipped noisy images: heteroskedastic modeling and practical denoising. *Signal Process* 2609–2629
19. Foi A, Katkovnik V, Egiazarian K (2007) Signal-dependent noise removal in pointwise shape-adaptive DCT domain with locally adaptive variance. In: *EUSIPCO*
20. Fredericksen RE, Hess RF (1997) Temporal detection in human vision: dependence on stimulus energy. *J Opt Soc Am A* 14(10):2557–2569. <https://doi.org/10.1364/JOSAA.14.002557>
21. Fredericksen RE, Hess RF (1998) Estimating multiple temporal mechanisms in human vision. *Vis Res* 38(7):1023–1040. [https://doi.org/10.1016/S0042-6989\(97\)00239-3](https://doi.org/10.1016/S0042-6989(97)00239-3)
22. Girod B (1993) What's wrong with mean squared error? In: *Watson AB (ed) Digital images and human vision*. MIT Press, Cambridge, MA, USA, pp 207–220. <http://dl.acm.org/citation.cfm?id=197765.197784>
23. Giryes R, Elad M (2012) Sparsity based poisson denoising. In: *2012 IEEE 27th convention of electrical electronics engineers in Israel (IEEEI)*, pp 1–5. <https://doi.org/10.1109/IEEEI.2012.6377139>
24. Hirakawa K, Parks T (2006) Joint demosaicking and denoising. *IEEE Trans Image Process* 15(8):2146–2157. <https://doi.org/10.1109/TIP.2006.875241>. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1658081>
25. ITU: Rec. ITU-R BT.709-4 (2000) Parameter values for the HDTV standards for production and international programme exchange
26. ITU-R Recommendation BT.2022 (2012) General viewing conditions for subjective assessment of quality of SDTV and HDTV television pictures on flat panel displays. International Telecommunication Union, Geneva, Switzerland

27. ITU-R Recommendation BT.500-13 (2012) Methodology for the subjective assessment of the quality of television pictures. International Telecommunication Union, Geneva, Switzerland
28. ITU-R Recommendation BT.709-5 (2009) Parameter values for the HDTV standards for production and international programme exchange. International Telecommunication Union, Geneva, Switzerland
29. ITU-R Recommendation BT.814-1 (1992–1994) Specifications and alignment procedures for setting of brightness and contrast of displays. International Telecommunication Union
30. Ivkovic G, Sankar R (2004) An algorithm for image quality assessment. In: ICASSP '04, vol 3, pp iii–713–716. <https://doi.org/10.1109/ICASSP.2004.1326644>
31. Kelly DH (1961) Visual responses to time-dependent stimuli. I. Amplitude sensitivity measurements. *J Opt Soc Am* 51(4):422–429. <https://doi.org/10.1364/JOSA.51.000422>
32. Kelly DH (1966) Frequency doubling in visual responses. *J Opt Soc Am* 56(11):1628–1632. <https://doi.org/10.1364/JOSA.56.001628>
33. Kelly DH (1971) Theory of flicker and transient responses, II. Counterphase gratings. *J Opt Soc Am* 61(5):632–640. <https://doi.org/10.1364/JOSA.61.000632>
34. Kelly DH (1974) Spatio-temporal frequency characteristics of color-vision mechanisms. *J Opt Soc Am* 64(7):983–990. <https://doi.org/10.1364/JOSA.64.000983>
35. Kelly DH (1977) Visual contrast sensitivity. *J Mod Opt* 24(2):107–129
36. Kelly DH (1979) Motion and vision. II. Stabilized spatio-temporal threshold surface. *J Opt Soc Am* 69(10):1340–1349. <https://doi.org/10.1364/JOSA.69.001340>
37. Kelly DH (1983) Spatiotemporal variation of chromatic and achromatic contrast thresholds. *J Opt Soc Am* 73(6):742–749. <https://doi.org/10.1364/JOSA.73.000742>
38. Levin A, Nadler B (2011) Natural image denoising: optimality and inherent bounds. In: 2011 IEEE conference on computer vision and pattern recognition (CVPR), pp 2833–2840
39. Lu W, Tan YP (2003) Color filter array demosaicking: new method and performance measures. *IEEE Trans Image Process* 12(10):1194–1210. <https://doi.org/10.1109/TIP.2003.816004>
40. Lukin V, Abramov S, Ponomarenko N, Egiazarian K, Astola J (2011) Image filtering: potential efficiency and current problems. In: 2011 IEEE international conference on acoustics, speech and signal processing (ICASSP), pp 1433–1436. <https://doi.org/10.1109/ICASSP.2011.5946683>
41. Mairal J, Bach F, Ponce J, Sapiro G, Zisserman A (2009) Non-local sparse models for image restoration. In: ICCV, pp 2272–2279
42. Nadenau MJ, Reichel J, Kunt M (2003) Wavelet-based color image compression: exploiting the contrast sensitivity function. *IEEE Trans Image Process* 12(1):58–70. <https://doi.org/10.1109/TIP.2002.807358>
43. Neumann L, Matkovic K, Purgathofer W (1998) Perception based color image difference. *Comput Gr Forum* 17(3):233–241. <https://doi.org/10.1111/1467-8659.00270>. <http://dx.doi.org/10.1111/1467-8659.00270>
44. Pedersen M (2014) An image difference metric based on simulation of image detail visibility and total variation. In: Color and imaging conference. Society for Imaging Science and Technology, pp 37–42
45. Portilla J, Strela V, Wainwright MJ, Simoncelli EP (2003) Image denoising using scale mixtures of Gaussians in the wavelet domain. *IEEE Trans Image Process* 12(11):1338–1351
46. Robson JG (1966) Spatial and temporal contrast sensitivity functions of the visual system. *J Opt Soc Am* 56(8):1141–1142
47. Roufs JAJ (1972) Dynamic properties of vision-I. Experimental relationships between flicker and flash thresholds. *Vis Res* 12(2):261–278. [https://doi.org/10.1016/0042-6989\(72\)90117-4](https://doi.org/10.1016/0042-6989(72)90117-4)
48. Roufs JAJ (1972) Dynamic properties of vision-II. Theoretical relationships between flicker and flash thresholds. *Vis Res* 12(2):279–292. [https://doi.org/10.1016/0042-6989\(72\)90118-6](https://doi.org/10.1016/0042-6989(72)90118-6)
49. Roufs JAJ (1973) Dynamic properties of vision-III. Twin flashes, single flashes and flickerfusion. *Vis Res* 13(2):309–323. [https://doi.org/10.1016/0042-6989\(73\)90109-0](https://doi.org/10.1016/0042-6989(73)90109-0)
50. Roufs JAJ (1974) Dynamic properties of vision-IV: thresholds of decremental flashes, incremental flashes and doublets in relation to flicker fusion. *Vis Res* 14(9):831–851. [https://doi.org/10.1016/0042-6989\(74\)90148-5](https://doi.org/10.1016/0042-6989(74)90148-5)

51. Roufs JAJ (1974) Dynamic properties of vision-V: perception lag and reaction time in relation to flicker and flash thresholds. *Vis Res* 14(9):853–869
52. Schöberl M, Schnurrer W, Oberdörster A, Föbel S, Kaup A (2010) Dimensioning of optical birefringent anti-alias filters for digital cameras. In: ICIP, pp 4305–4308
53. Stroebel L, Compton J, Current I, Zakia R (2000) Tone reproduction. In: *Basic photographic materials and processes*. Elsevier, pp 235–255
54. Talebi H, Milanfar P (2014) Global denoising is asymptotically optimal. In: 2014 IEEE international conference on image processing (ICIP), pp 818–822. <https://doi.org/10.1109/ICIP.2014.7025164>
55. Trussell HJ, Zhang R (2012) The dominance of Poisson noise in color digital cameras. In: ICIP, pp 329–332. <https://doi.org/10.1109/ICIP.2012.6466862>
56. Van Nes FL, Koenderink JJ, Nas H, Bouman MA (1967) Spatiotemporal modulation transfer in the human eye. *J Opt Soc Am* 57(9):1082–1087. <https://doi.org/10.1364/JOSA.57.001082>
57. Vansteenkiste E, Weken D, Philips W, Kerre E (2006) Perceived image quality measurement of state-of-the-art noise reduction schemes. In: *Advanced concepts for intelligent vision systems*. Springer, Berlin, Heidelberg, pp 114–126
58. Wang Z, Bovik A (2009) Mean squared error: love it or leave it? A new look at signal fidelity measures. *IEEE Signal Process Mag* 26(1):98–117. <https://doi.org/10.1109/MSP.2008.930649>
59. Winkler S (1999) A perceptual distortion metric for digital color video. In: *Electronic imaging '99*, San Jose, California, pp 175–184
60. Winkler S, Süsstrunk S (2004) Visibility of noise in natural images. In: *Electronic imaging 2004*. International Society for Optics and Photonics, pp 121–129
61. Zhang L, Lukac R, Wu X, Zhang D (2009) PCA-Based spatially adaptive denoising of CFA images for single-sensor digital cameras. *IEEE Trans Image Process* 18(4):797–812. <https://doi.org/10.1109/TIP.2008.2011384>
62. Zhang L, Wu X (2005) Color demosaicking via directional linear minimum mean square-error estimation. *IEEE Trans Image Process* 14(12):2167–2178
63. Zhang L, Wu X, Zhang D (2007) Color reproduction from noisy CFA data of single sensor digital cameras. *IEEE Trans Image Process* 16(9):2184–2197

# Chapter 10

## Pull-Push Non-local Means with Guided and Burst Filtering Capabilities



John R. Isidoro and Peyman Milanfar

**Abstract** Non-local means filtering (NLM) has cultivated a large amount of work in the computational imaging community due to its ability to use the self-similarity of image patches in order to more accurately filter noisy images. However, non-local means filtering has a computational complexity that is the product of three different factors, namely,  $O(NPK)$ , where  $K$  is the number of filter kernel taps (e.g., search window size),  $P$  is the number of taps in the patches used for comparison, and  $N$  is number of pixels in the image. We propose a fast approximation of non-local means filtering using the multiscale methodology of the pull-push scattered data interpolation method. By using NLM with a small filter kernel to selectively propagate filtering results and noise variance estimates from fine to coarse scales and back, the process can be used to provide comparable filtering capability to brute force NLM but with algorithmic complexity that is decoupled from the kernel size,  $K$ . We demonstrate that its denoising capability is comparable to NLM with much larger filter kernels, but at a fraction of the computational cost. In addition to this, we demonstrate extensions to the approach that allows for guided filtering using a reference image as well as motion compensated multi-image burst denoising. The motion compensation technique is notably efficient and effective in this context since it reuses the multiscale patch comparison computations required by the pull-push NLM algorithm.

### 10.1 Introduction

A widely practiced way to improve the efficiency of a filtering algorithm is to use a multiscale technique. Multiscale filtering effectively computes a wide filtering kernel by using a cascade of simple filters across all scales used. Using a coarse-to-fine strategy allows for a small amount of processing at each successive level to refine the result.

---

J. R. Isidoro · P. Milanfar (✉)  
Google Research, Mountain View, USA  
e-mail: peyman.milanfar@gmail.com

© Springer International Publishing AG, part of Springer Nature 2018  
M. Bertalmío (ed.), *Denoising of Photographic Images and Video*,  
Advances in Computer Vision and Pattern Recognition,  
[https://doi.org/10.1007/978-3-319-96029-6\\_10](https://doi.org/10.1007/978-3-319-96029-6_10)

267

Commonly used multiscale approaches begin with a fine to coarse pass followed by a coarse-to-fine pass to generate the final result. This requires a pyramid for each pass, resulting in 1.66 times the complexity of the direct approach in terms of the number of pixels processed. However, since these multiscale approaches allow the use of much simpler per-level filter kernels, the kernel size becomes much less of a factor in the computation. This is especially true when a large and non-separable single scale filter is being approximated.

We present three main contributions in this work. The first is selective edge adaptive propagation of filtering results from level to level. Even though multiscale techniques have been used for decades, generally these techniques use a fixed (non-data-dependent) kernel in order to generate successively downsampled versions of the original image. If the filtering kernel weights are not input dependent, they may combine pixels in a way that is incompatible with data-dependent filtering, i.e., filtering across edges and propagating erroneous results through the solution. In contrast, our approach uses a nonlinear filter (NLM) to propagate results from pyramid level to level.

Second, the original pull-push approach [10] was used for scattered data interpolation. Here, we extend its methodology and present a data-dependent multiscale algorithm for denoising images efficiently. More specifically, we expand upon the concept of pull-push to accelerate non-local means in a way that provides comparable output to large filter kernel (e.g., large search window) NLM filtering, but with computation comparable to using a much smaller kernel.

The third contribution is efficient motion compensated burst denoising. Our extension of the algorithm to burst processing allows the patch comparison computations of pull-push NLM to be reused for motion estimation. These motion estimates are naturally propagated in a coarse-to-fine manner to offset the filter kernels to follow the motion between frames in the burst. By accounting for the motion, better matched patches can be found and denoising can be improved.

## 10.2 Related Work

Non-local means [2] is a filtering technique that extends bilateral filtering [28, 35] by using patch-based comparison instead of single pixel comparison to derive weights for filtering. Patch-based comparison allows for better characterization of image regions surrounding pixels to be filtered. This in turn allows for more accurately computed filter weights and results in better quality edge-aware denoising than bilateral filtering.

One class of methods for accelerating edge-aware filtering involves converting the filtering function into a higher dimensional space in which fast non-data-dependent separable filtering is applied to perform the bulk of the work [3, 20]. This decouples the computational complexity from the filter kernel size (e.g., search window size)  $K$ . As an example, the permutohedral lattice [1] is capable of NLM filtering, but the performance scales quadratically with the number of comparison patch samples  $P$ , i.e.,  $O(NP^2)$ .

Fast edge-aware filtering can also be performed by approximating the bilateral filter as a series of 1D separable [23] or recursive IIR [9, 37, 38] passes. These approaches have not been extended to NLM, and it is not obvious how to do so.

Multiscale techniques have been used to accelerate NLM in the original paper [2], but were limited to optimizing the patch comparisons or weight computation. Another approach [17] applies a wavelet decomposition to improve patch comparison quality for weight computation, but does not use multiscale for the filtering itself.

Nercessian et al. [19] used Laplacian pyramids and performed NLM filtering in each level as part of the reconstruction of the original image from the pyramid decomposition. The method of Zhang and Gunturk [39] is conceptually similar but uses a wavelet decomposition. She et al. [27] proposed an alternative fixed kernel pyramid decomposition with bilateral filtering at each level. Zontak et al. [40] also used multiple image scales generated with non-edge-aware filtering, but used NLM to directly fuse these pixels back into the original level. Other edge preserving multiscale decompositions exist throughout the literature, such as [7] and [8], but have not been extended to NLM filtering.

Due to the non-edge-aware filtering during pyramid creation, many multiscale techniques can potentially suffer from color bleed issues, but probably less so due to the anisotropic downscaling, and always comparing patches against pixel data from the full resolution image being filtered. The main difference between our multiscale approach and these approaches is that by using NLM as the filter in the construction of the up and down pyramid processing, we avoid combining pixels together that would not be combined under the brute force NLM approach.

Kervrann and Boulanger [14] performed successive iterations of NLM filtering and updated per-pixel variance values based on the estimated variance reduction from the sum of squared normalized weights. Much of the variance propagation machinery is similar in our approach, but their approach is not multiscale and thus processes the full-resolution image multiple times.

Another related iterative method which can be applied to NLM filtering is the SAIF method [34] which adaptively adjusts the number of iteration passes locally based on an approximation of MSE. In contrast, the pull-push framework tracks an estimate of the noise variance throughout the multiscale process to selectively control the amount of filtering on a per-pixel basis, but adds multiscale capability to efficiently process large filter footprints.

It is also possible to approximate global NLM filtering as an approximated low-rank representation of the affinity matrix [18]. This representation has successfully been used for a variety of non-local image enhancement tasks [32]. The storage and runtime cost of an algorithm which uses this technique for purely denoising [31] is a function of both the number of pixels and the rank of the approximation  $O(NR)$ . While being capable of approximating large filter footprints (as large as the image itself) the rank of the approximation needed for good quality is dependent on both the noise level and complexity of the underlying image itself whereas the performance of pull-push NLM is independent of both. It should also be noted that pull-push NLM denoising also could be used in place of standard NLM for other filtering tasks where a denoiser is used as a building block [26, 33].



An effective speedup for NLM is to reorder the loops in the algorithm and use separable filtering for the weight computation [4, 5]. This helps mostly when the patch sizes are large. This technique is solely part of the patch matching process, and as such, it can also be applied to the pull-push algorithm presented in this chapter as an additional optimization.

### 10.2.1 Overview of Pull-Push

The pull-push algorithm was originally developed as an efficient and GPU amenable scattered data interpolation method [10]. The problem of scattered data interpolation is to estimate and fill in the missing pixels in an image where only a subset of the pixels are specified. To describe the pull-push methodology, first, we establish some basic notation. Since this chapter deals with image pyramids, we use a superscript (e.g.,  $x^{[r]}$ ) to indicate pyramid level. A[0] superscript indicates the finest resolution (initial) pyramid level, with subsequent indices indicating each coarser level. Consider an initial image  $x^{[0]}$  indexed by the pixel location,  $i$ , with per-pixel weights  $w_i^{[0]}$ . A weight of 1.0 is used for pixels that are present and a weight of 0.0 for pixels that are missing.

The first step in pull-push is to build an image pyramid. Each successive level is generated recursively by a “pull” (analogous to downsampling) filter. This filter uses a per-pixel weighting scheme to selectively combine existing pixel data to estimate missing regions. In addition to this, for each pixel generated, a weight is computed which is proportional to the number of present pixels used to generate that pixel. This weight can be thought of as a coverage fraction.  $d$  is a fixed tap spatial filter for downsampling with taps indexed by tap offset  $k$ . Often a simple  $2 \times 2$  box filter is used in practice to leverage bilinear texture mapping hardware on a GPU. The expressions for the weight propagation and pixel updates are (borrowing some notation from [16])

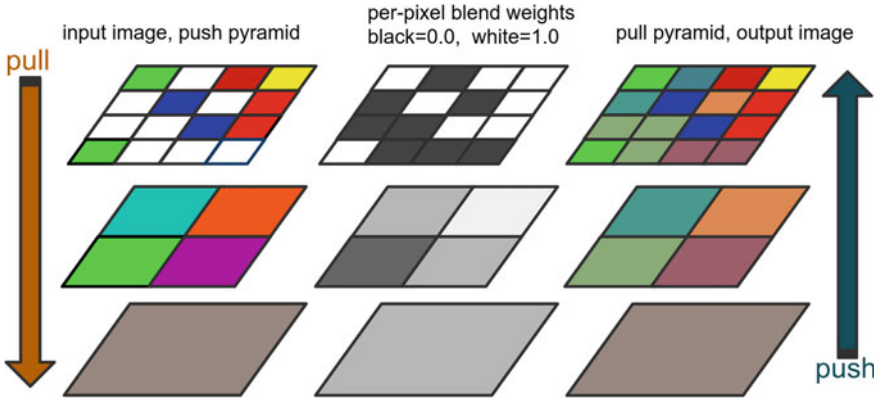
$$w_i^{[r+1]} = \sum_k d_k w_{2i+k}^{[r]}, \quad (10.1)$$

$$x_i^{[r+1]} = \sum_k \frac{d_k w_k^{[r]} x_{2i+k}^{[r]}}{w_i^{[r+1]}}. \quad (10.2)$$

After the pull stage is evaluated to the coarsest level, the push stage blends in missing pixels starting from the coarsest level and generating finer levels iteratively. Denoting  $\chi^{[r-1]}$  as the upsampled version of the next coarser pyramid level  $y^{[r]}$  and  $u$  as a fixed spatial filter<sup>1</sup> for upsampling, we have

---

<sup>1</sup>Often in practice, this is simply a bilinear upscaling kernel which can leverage bilinear texture filtering hardware on a GPU [16].



**Fig. 10.1** Diagram of how the pull-push algorithm propagates per-pixel weights to represent the pixel coverage each coarser level has. These weights are used to combine samples in successively coarser or finer levels

$$\omega_i^{[r-1]} = \sum_k u_k w_{(i/2)+k}^{[r]}, \tag{10.3}$$

$$\chi_i^{[r-1]} = \sum_k \frac{u_k w_k^{[r]} y_{(i/2)+k}^{[r]}}{\omega_i^{[r-1]}}, \tag{10.4}$$

$$y_i^{[r-1]} = x_i^{[r-1]} w_i^{[r-1]} + \chi_i^{[r]} (1 - w_i^{[r-1]}). \tag{10.5}$$

A few key observations about pull-push should be noted here. Like other multi-scale frameworks such as Laplacian pyramids, pull-push maintains the capability to much more efficiently filter over large regions than a single scale method. Since a tiny fixed size kernel is used per level, the overall complexity of the pull-push approach is not dependent on the effective filter size (Fig. 10.1).

Each pull-stage downsampled pixel can be considered as a distribution of the pixel values in the finer levels above it. The pixel value functions as a mean, and the weight function as an encoding of the strength or reliability of the mean (e.g., how many pixels were combined to represent it). Regions which have less reliable or no information incorporate more data from a wider region around them in order to form an estimate. This selective filtering and reliability (i.e., information) propagation is what makes push-pull different from existing multiscale frameworks.

### 10.2.2 Overview of NLM

Non-local means [2] is an edge-aware filtering technique that uses a patch matching score to determine filtering weights. Given an input image  $x$ , the kernel weight  $w_{i,k}$  per-pixel location  $i$  and tap location  $k$  is based on a patch neighborhood with patch taps  $p$ . Consider  $\sigma_s$  to be a smoothing parameter based on the amount of sensor noise. The weights are designed as follows:

$$\Delta_{i,k} = \sum_p \left( \frac{x_{i+p} - x_{i+k+p}}{\sigma_s} \right)^2, \quad (10.6)$$

$$w_{i,k} = \exp(-\Delta_{i,k}). \quad (10.7)$$

Normalizing by the sum of weights

$$\hat{w}_{i,k} = \frac{w_{i,k}}{\sum_k w_{i,k}}. \quad (10.8)$$

Each new filtered pixel is a weighted average of its kernel neighborhood:

$$\hat{x}_i = \sum_k \hat{w}_{i,k} x_{i+k}. \quad (10.9)$$

Although patch matching allows for more accurate weighting of samples for denoising, it comes at additional computational expense over bilateral filtering. The additional work in the per-pixel inner loop over the patch taps is multiplicative and increases the computational complexity of the algorithm from the  $O(NK)$  of a brute force bilateral implementation to  $O(NPK)$ , where  $K$  is the number of filter kernel taps,  $P$  is the number of patch samples, and  $N$  is number of pixels. Note that in the original NLM paper [2],  $K$  included all pixels in the image, but in practical implementation, this is often limited to a reasonably small local window (roughly  $5 \times 5$  to  $21 \times 21$  pixels) with a fixed number of kernel taps due to computation cost. By combining NLM with the pull-push methodology, we can reduce the computational complexity to  $O(NP)$ . Pull-push NLM [12] gives us both a large effective kernel size and computational efficiency.

### 10.3 Pull-Push NLM

Using the standard NLM formulation with patch matching based on a per-pixel i.i.d. Gaussian noise model, pixels are selectively averaged together based on their patch similarities. Considering the sensor noise for a pixel as i.i.d. Gaussian with

mean  $\mu_i$  and standard deviation  $\sigma_n$ , the estimated distribution of the filtered sample (a weighted average of Gaussian samples) can be computed from the sum of the squared normalized kernel weights [18]

$$\hat{x}_i \sim \mathcal{N} \left( \sum_k \hat{w}_{i,k} x_{i+k}, \sigma_n^2 \sum_k \hat{w}_{i,k}^2 \right). \quad (10.10)$$

The kernel weights are also a valuable source of information to distinguish structure from noise in the kernel neighborhood. The unnormalized  $w_{i,k}$  kernel weights allow for simple approximate measure of structure versus noise. If most of the weights for a patch are high, it is likely to be a flat region in the image with only additive sensor noise. If only the center tap has a high weight, the center tap is likely to be speckle noise. If a certain subset of the weights has a high value, it can often be due to matching some type of image structure such as an edge or a corner.

Considering the variance reduction formula from Eq. 10.10, the NLM kernel weights can be used to determine the reliability of a filtered sample. By storing the sum of squared normalized weights along with each corresponding NLM filtered value per pixel, we are in effect providing a simple characterization of the distribution of samples used to create the output. By propagating this information through the layers, we enable tracking a simplified model of the neighborhood statistics for successively larger regions in a multiscale manner.

One last point is that eventually, we will need to combine estimates with different variances during the pull stage. Inverse variance weighting is a common method to aggregate multiple independent observations  $y_k$  each with different variances  $\sigma_{y_k}^2$ . Given this input, inverse variance weighting is known to be the minimum variance estimate using all the uncorrelated observations [11, 13],

$$\hat{y} = \frac{\sum_k \frac{y_k}{\sigma_{y_k}^2}}{\sum_k \frac{1}{\sigma_{y_k}^2}}. \quad (10.11)$$

In Sect. 10.3.4, we show how inverse variance weighting is used to derive our reliability score.

### 10.3.1 Pull Stage: Downfuse

In order to aggregate filtering results for a neighborhood of pixels without filtering across edges, NLM filtering is used to generate subsequent coarser pyramid levels during the pull stage. Because it is not performing a scaling operation but rather computing a selective combination of pixels, we call this a downfuse pass instead of downscaling. We perform NLM once per  $2 \times 2$  block of pixels at level  $[r]$  and store the results in the next coarsest level  $[r + 1]$

$$\Delta_{i,k}^{[r]} = \sum_p \left( \frac{x_{i+p}^{[r]} - x_{i+k+p}^{[r]}}{\sigma_s^{[r]}} \right)^2, \quad (10.12)$$

$$w_{i,k}^{[r]} = \exp(-\Delta_{i,k}^{[r]}), \quad (10.13)$$

$$\widehat{w}_{i,k}^{[r]} = \frac{w_{i,k}^{[r]}}{\sum_k w_{i,k}^{[r]}}, \quad (10.14)$$

$$x_i^{[r]} = \sum_k \widehat{w}_{2i,k}^{[r-1]} x_{2i+k}^{[r-1]}. \quad (10.15)$$

The output will be the result of fusing together differing numbers of samples on a per-pixel basis. To keep track of this variance on a per-pixel basis, the pull-push approach relies on propagating reliability information to selectively combine data from level to level. We also compute reliability per-pixel for each downfused sample. One statistically motivated way to do this is

$$\rho_i^{[r]} = \frac{1}{\sum_k (\widehat{w}_{2i,k}^{[r-1]})^2}. \quad (10.16)$$

This reliability score is the inverse of the variance scale factor for weighted sum of random variables described previously. This reliability is analogous to the way blend weights are propagated in the pull-push scattered data interpolation technique. Incorporating the reliability score into NLM can be performed as weighted NLM for  $r \geq 1$

$$w_{i,k}^{[r]} = \rho_{i,k}^{[r-1]} \exp(-\Delta_{i,k}^{[r]}) \cdot S. \quad (10.17)$$

### 10.3.2 Push Stage: Upfuse

Now that a pyramid has been constructed containing filtering results, each with a reliability score, a push stage will be used in order to up-integrate the results. Similarly to the pull stage, the push stage also uses NLM to selectively combine samples, but also incorporates samples from lower pyramid levels. Analogous to the downfusing, we call the push stage the upfuse stage because of the selective filtering and combination of samples to generate the next finer level.

In order to determine the patch comparison weights to incorporate samples from a coarser level, we perform patch comparisons from one level finer. During the down-fuse stage, each NLM result sample was generated from one level finer, including the

patch comparisons to generate that sample. Therefore, it makes sense that a center tap patch used by the downfuse stage to generate a coarser sample serves as a signature for the neighborhood, and can be used for patch comparison to determine weights for the coarser sample. For NLM filtering in the upfuse stage, the tap offsets  $k$  come from two different locations. Tap offsets from the current level are denoted as  $k_f$ , and the tap offsets for samples from the coarser level will be denoted as  $k_c$ . The set of taps containing both  $k_f$  and  $k_c$  is  $k$ .

The weights for sample offsets for the different levels are

$$w_{i,k_f}^{[r]} = \exp(-\Delta_{i,k_f}^{[r]}), \quad (10.18)$$

$$w_{i,k_c}^{[r]} = \rho_{g(i+k_c)/2}^{[r+1]} \exp(-\Delta_{i,g(k_c)}^{[r]}), \quad (10.19)$$

where  $g(j)$  is the tap location in the finer level that generated the downfused sample in the coarser level for location  $j$ .

Again the weights are normalized using the sum over all weights (e.g., from both the coarse and fine levels),

$$\hat{w}_{i,k}^{[r]} = \frac{w_{i,k}^{[r]}}{\sum_k w_{i,k}^{[r]}} \quad (10.20)$$

$$y_i^{[r]} = \left( \sum_{k_f} \hat{w}_{i,k_f}^{[r]} x_{i+k_f}^{[r]} \right) + \sum_{k_c} \hat{w}_{i,k_c}^{[r]} y_{g(i+k_c)/2}^{[r+1]} \quad (10.21)$$

with the exception of the coarsest level which uses the coarsest downfused output image directly. Figure 10.2 shows an image being filtered and how filtering progresses through the downfuse and upfuse process. Coarser levels are magnified in the diagram for easier visualization.

### 10.3.3 Per-Level Patch Variance Reduction

When combining samples the modeled variance of each output pixel is computed from the NLM weights and stored. However, patch differences are the result of both noise and structural variation. Note that patch comparison is meant to measure how much actual structural variation there is between patches. We do not want to downweight important structure but rather maintain its influence. Using the per-pixel variance reduction for each pixel in the patch to patch comparison may underweight meaningfully different edge pixels in comparison to well filtered flat regions.



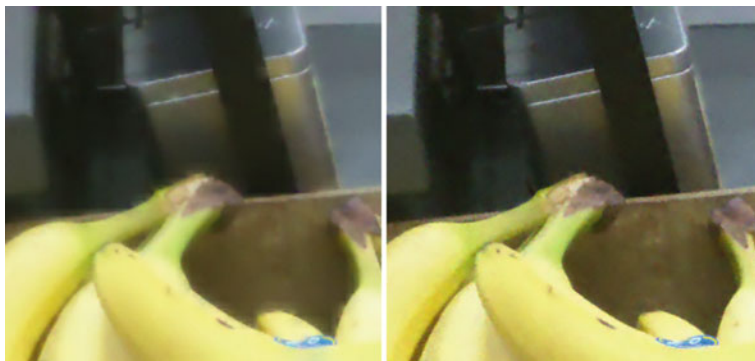
**Fig. 10.2** This series of images shows the progression of NLM pull-push filtering of an image through the downfuse and upfuse stages. Notice how the noise is reduced throughout the stages

Therefore, we do not use the per-pixel variance reweighting inside the patch comparison function itself  $\Delta_{i,k}$ .

However, from our experiments simply using the same patch sigma from level to level usually results in a loss in mid-frequency detail caused by low contrast structural edges being smoothed away. In order to account for this, the overall patch comparison sigma is multiplied by a sigma reduction factor for each level down the pyramid. We found that in practice setting the sigma reduction factor to 0.5 provided the best results visually in terms of trading off detail for noise removal. Intuitively, the 0.5 factor this makes sense because in the pyramids we use, four pixels in a pyramid level map to a single pixel in the level below, and the sigma reduction factor of 0.5 corresponds exactly to the theoretical variance reduction of averaging 4 Gaussian i.i.d. samples.

### 10.3.4 Selective Up-Integration Using Reliability

During the upfuse process, it is important to prevent incorporating unreliable data from lower pyramid levels. Due to local image structure, filtered results may not be reliable or useful if too few samples were combined to create the result. Furthermore, but if few samples were combined, it signifies the region may contain some form of local structure, and directly using the coarser results may introduce blocky under-sampling artifacts. To prevent this, we only combine samples for which the reliability score is above a given threshold. In practice, we found a threshold corresponding to



**Fig. 10.3** The left image shows the effect of pull-push filtering without reliability downweighting. Near the edge of the banana, the upfused samples contained a high amount of structure and resulted in color smearing artifacts near the edges of the bananas. The right side image shows the same region filtered using the reliability adjustment to mitigate the artifacts

at least three samples being combined  $\rho_{thresh} = 3$  is effective to eliminate artifacts. Figure 10.3 shows how reliability downweighting alleviates artifacts. In order to prevent artifacts from a hard thresholding and to provide a smooth transition based on the reliability, we subtract and clamp in order to implement this soft threshold.

$$\hat{\rho}_i = \min(\rho_i - \rho_{thresh}, 0) \quad (10.22)$$

## 10.4 Joint Pull-Push Non-local Means

Joint (or guided) bilateral filters have been successful in a variety of applications where one high-quality “reference” image,  $v$ , is used to guide the filtering of the input image,  $x$  [6, 15, 22]. The reference image is often higher in resolution and/or lower in noise than the image to be filtered. A common use case is flash photography, where the frame taken with the flash on is used to refine the “natural light” image taken with the flash off. Other potential use cases are refining depth maps with an input image, or refining images from additional modalities such as infrared or other non-visible spectra.

The joint bilateral filter (sometimes also referred to as guided or cross bilateral filtering) [6] and [22] uses a separate reference image to derive filtering weights for the image to be filtered. The idea can be extended to standard non-local means filtering as well, where all pixels for patch comparison and weight computation are taken from the reference image instead of the input image.

However, in the pull-push NLM methodology, since filtering weights are derived from subsequent levels of the pull-push pyramid, the way to do this is not immediately



obvious. From the reference image, a Gaussian pyramid is built with each level being half the resolution of the previous level.

$$\hat{v}^{[r+1]} = G(v^{[r]}) \quad (10.23)$$

The function  $G$  performs standard separable Gaussian filtering on the image and downsamples the result to one half of its resolution. The separable kernel used in  $x$  and  $y$  for filtering is  $[1/6, 4/6, 1/6]$ . Since each pixel is the average of a small neighborhood of pixels from the level above, the noise variance for each pixel of each level can be computed directly from the noise variance of source pixels using Eq. 10.10. This particular kernel results in each pixel having one half the noise standard deviation as the pixels in the level above. As is used in non-joint pull-push NLM, the per-level sigma reduction factor from Sect. 10.3.3 is used to account for the reduction in sigma from level to level. Again, a per-sigma reduction factor of 0.5 is theoretically motivated and works well in practice.

The images from this pyramid will be used for patch comparisons in order to compute the weights. The new patch comparison function for joint filtering simply uses the reference images in the place of the input images for the given pyramid level.

$$\Delta_{i,k}^{[r]} = \sum_p \left( \frac{v_{i+p}^{[r]} - v_{i+k+p}^{[r]}}{\sigma_s^{[r]}} \right)^2 \quad (10.24)$$

Other than this, the joint pull-push NLM algorithm is identical to the non-joint version.

To get the best results from this algorithm, the optimal value for the smoothing parameter  $\sigma_s^{[r]}$  depends on both the noise level in the reference image as well as the noise level of the image to be filtered. In practice, we found that the optimal for  $\sigma_s^{[r]}$  for joint pull-push NLM is roughly the average of the optimal  $\sigma_s^{[r]}$  values for denoising the reference and input images independently.

## 10.5 Burst Processing

Most modern cameras allow for burst capture of a sequence of images taken with shorter exposures in order to compensate for both motion and sensor noise. While the shorter exposure times may more accurately capture moving objects (reduction of motion blur), it will also increase the noise in each image of the burst. Combining multiple frames together into a single frame will be necessary to reduce this noise.

The frames to be fused together may contain some motion, and higher quality results can be obtained by taking into account this motion. A base frame is chosen from the sequence as the reference frame. The other burst images will be fused into the base frame in order to enhance it. In most cases, the middle frame of the sequence

is chosen so that motion across frames would likely have the smallest deviation from the base frame.

This section describes two methods to handle burst fusion, a simple scheme for dealing with small motions, and a more complex scheme which involves rough estimation of the motion to guide the search for matching patches across frames.

### ***10.5.1 Processing of Burst Frames Without Explicit Motion Compensation***

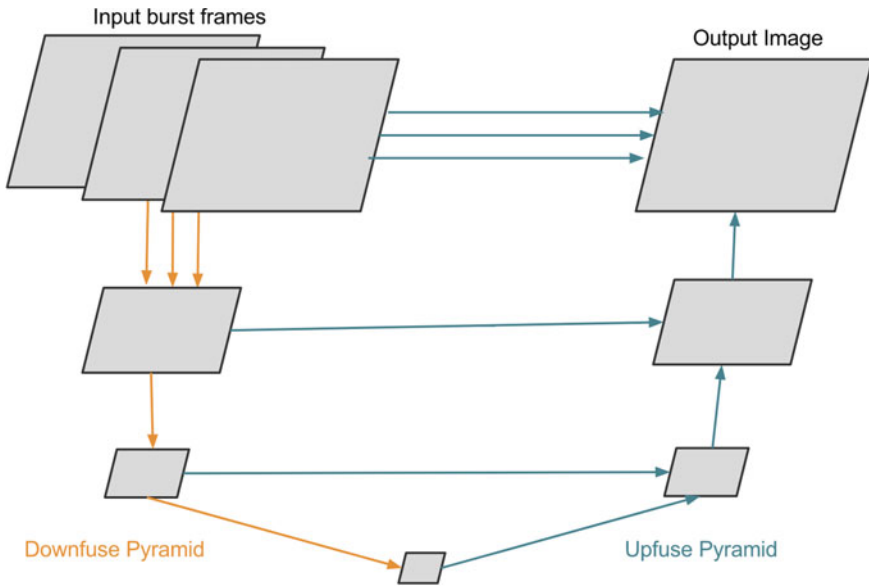
One way to incorporate burst processing into standard non-local means is to not estimate the motion, and allow the non-local means algorithm to use any matching patches within the kernel footprint in the burst frames as well [24, 30].

The extension to the pull-push NLM is very similar to the existing approach, but with two small differences. When generating the first downfused image, the NLM kernel footprint is extended to encompass the time dimension across all input burst frames.

The second difference is in the final upfuse step. In this step, all the burst images are fused along with the previous upfused result in order to generate the final image. Again the pixel footprint is extended across the burst images, allowing for denoising across frames. Figure 10.4 shows how these two differences allow the burst frames to be incorporated into the pull-push framework.

In this scheme, the kernel footprint is in the same location across the burst frames, so motions that are within the kernel size can benefit from filtering across multiple frames. For example, if an object moves one pixel from its position in the base frame, the NLM algorithm performs patch comparisons with all pixels within the kernel footprint, and there will be a kernel offset which corresponds to the object that has moved. Since the kernel footprint has the same spatial location from frame to frame, motions which extend outside the extent of the kernel will not contribute to the denoising. For example, the object has moved far enough, there will be no matching patches within the kernel footprint, and samples from the region of motion will not be averaged into the final result. This limits the capability of the algorithm to denoise across frames.

Despite the limited ability to denoise in large motion situations, the non-motion compensated pull-push NLM works well in practice on stable scenes, but faltered for large motions. Ideally what we want is for the kernel footprint to track the object motion from frame to frame in the burst. The hope is that for moving objects, we can pull patches from the same object across frames, and get better denoising. The interesting thing about this is that with some small modifications, the multiscale patch matching already performed by pull-push NLM can be leveraged for per-pixel motion estimation.

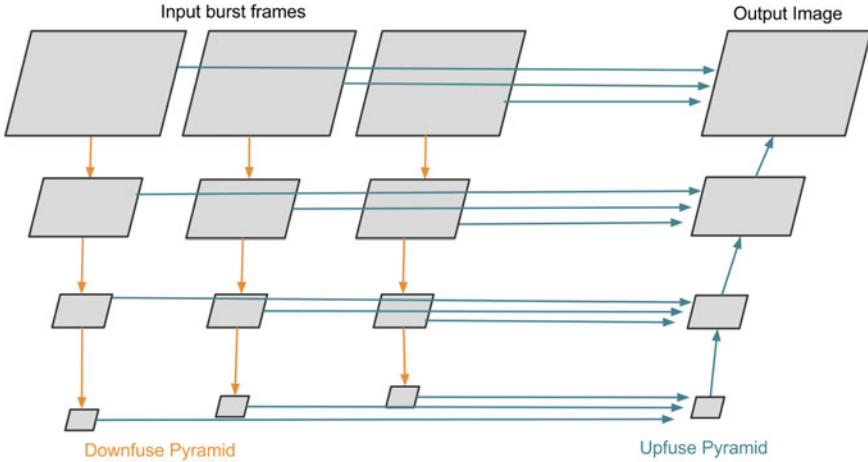


**Fig. 10.4** Flow diagram showing pull-push NLM burst processing without motion compensation. Only the first step of the downfuse and final step of the upfuse use multiple frames as inputs

### 10.5.2 Simultaneous NLM and Motion Compensation Using the Pull-Push Framework

A straightforward method for motion estimation is to perform patch matching over a neighborhood and use the offset of the best matching patch. A detailed overview of related patch matching-based motion estimation techniques is available in [21]. Since we are already doing patch to patch comparisons when performing NLM, most of the necessary computation has already been performed. In addition, since the pull-push methodology performs these patch matching comparisons on multiple scales, the resulting estimated per-pixel motion can be propagated from the coarsest to fine levels during the upfuse stage. Since the effective image area of each pixel doubles in size as we go down one level in the pyramid, every level of the pyramid doubles the amount of motion capable of being estimated. A coarse-to-fine strategy of propagating motion estimation results from level to level, a 5-level pyramid can handle roughly  $2^5 = 32$  pixels of motion. From level to level, the estimated per-burst-image motion field is also used as a per-pixel offset that is applied to the kernel in the hope of finding pixels that better patch match and be able to incorporate more pixels into the final estimate.

Since we do not know the motion to begin with, a downfuse pyramid is computed independently for each burst image. Figure 10.5 shows the separate downfuse pyramids and how it differs from the strategy of the non-motion compensated case



**Fig. 10.5** Pull-Push NLM burst processing with motion compensation. Each downfuse pyramid is computed separately, after this during each stage of the upfuse multiple frames

Fig. 10.4. Logically, for the base frame itself, the motion field image is always set to zero because the base frame should not move with respect to itself. Also, the motion field image used as input for the lowest pyramid level is initialized to zero,

$$m_{i,b}^{[R]} = 0, \tag{10.25}$$

where  $b$  is the index of the burst frame. The number of pyramid levels is  $R$ . If some external source of motion estimation is available the algorithm could be initialized with that instead.

The motion field image guides the patch matching. Since it contains an estimate of where an object has moved to from the previous level, it can be used as a per-pixel per-burst-image offset for the kernel itself. This way patches are more likely be taken from the new position of objects that moved from frame to frame,

$$\widehat{k}_{i,b}^{[r]} = k + 2m_{i/2,b}^{[r+1]}. \tag{10.26}$$

The  $\widehat{k}_{i,b}^{[r-1]}$  are used in the place of  $k_f$  in Eq. 10.21. Motion estimation is refined through each step of the upfuse. During the NLM patch matching, we keep track of the kernel offset  $k$  that produced the lowest overall patch delta  $\Delta_{i,k}$  on a per-pixel basis. We define these kernel offsets as  $\vec{k}_{i,b}^{[r]}$ .

The motion field  $m_{i,b}^{[r]}$  is refined from level to level using the estimate from the previous level.

$$m_{i,b}^{[r-1]} = 2m_{i/2,b}^{[r]} + \vec{k}_{i,b}^{[r]} \tag{10.27}$$

One consideration that should be mentioned is that the ultimate goal of this approach is better denoising through motion compensation, not computing accurate motion estimation. As long as we find better matched patches to fuse pixels from, we gain better denoising capability. For example, in flat regions, it may not be possible to estimate motion accurately since all kernel taps have well-matched patches. But, since the patches all match well, they will contribute to the final fused result and improve the denoising. So having accurate motion vectors is not a requirement, all we are trying to do is to find the best offset to pull samples from on a per-pixel basis.

We have seen improvements in accuracy for motion estimation by combining this approach with the joint pull-push approach from the previous section. In this case, a Gaussian pyramid is generated for each burst frame solely to be used as the reference images for patch comparison. This makes intuitive sense since the selective kernel of NLM reweights taps from different offsets, which can bias the motion estimation. The Gaussian kernel is radially symmetric and does not suffer from this shortcoming. There is no directional bias for any given direction.

Burst photography can be combined with joint filtering as well. One way in which to obtain the inputs for joint burst photography is to capture the burst with a synchronized strobe light where pairs of reference and burst frames are taken in rapid succession [29]. This falls naturally out of the motion compensated burst processed framework, the only difference is that the Gaussian pyramids for patch comparison are generated from the reference images.

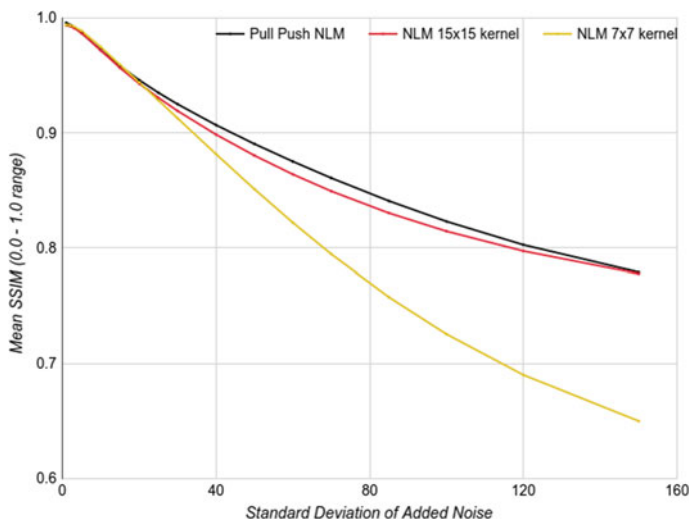
## 10.6 Experimental Results

### 10.6.1 Pull-Push NLM Experiments

The experiments in this section compare pull-push NLM to the standard NLM algorithm. These experiments were run on a 6 Core Xeon CPU using Halide [25] to enable vectorization and parallelization across cores. The input imagery is a set of 143 different real-world color 12 MP images. In order to best compare algorithm performance, both the pull-push NLM and standard NLM implementations use 32-bit floating point math and the same  $3 \times 3$  patch comparison size. Other than using standard Halide vectorization and parallelization, no additional optimizations were used for this comparison.

The comparison was performed on a collection of clean real-world images by adding synthetic Gaussian noise across a variety of different noise levels. We use SSIM [36] to quantify the difference between the noiseless base image and the processed denoised image. The standard recommended values from [36] were used for all parameters. For each algorithm, the input  $\sigma_s$  parameter resulting in the highest SSIM score was used on a per-image basis.

We found that the pull-push NLM approach runs in a little less time than NLM using a  $7 \times 7$  filter kernel (e.g., search window) (320 vs. 350 ms per MP). Comparison



**Fig. 10.6** Denoising performance of pull-push NLM versus standard NLM for a range of synthetic noise levels

with the  $7 \times 7$  NLM can be considered as a quality comparison for equivalent runtime. In order to compare pull-push NLM versus large kernel NLM we use a  $15 \times 15$  filter kernel (1660 ms per MP) for the standard NLM algorithm. Generally, the  $15 \times 15$  kernel NLM has comparable quality to the pull-push NLM results. Note that all of these timings can be improved by an order of magnitude with standard NLM optimizations such as those described in [24].

The results are summarized in Fig. 10.6. Trials were run on a wide range of synthetic noise levels ranging in standard deviation from mild noise ( $\sigma_n = 1$ ) to extreme noise ( $\sigma_n = 150$ ). Each data point on the graph represents the mean best-case SSIM for the given algorithm across the 143 images. For noise  $\sigma_n \leq 15$ , the three approaches resulted in very similar SSIM scores. However as  $\sigma_n$  increased, the pull-push NLM approach performed better than  $7 \times 7$  NLM with the difference widening as noise increased. Even against NLM using a  $15 \times 15$  kernel (and taking five times as much time to process the same image), the pull-push approach provided better quality results from  $\sigma_n$  ranging from 20 to 125.

From the experiments, we found the pull-push NLM algorithm performed best when using five pyramid levels. Using additional pyramid levels has a comparatively negligible cost, but offered negligible additional denoising. This is due to the fact that a 5 level pyramid offers roughly up to a 31-pixel filter radius ( $63 \times 63$ ) and can potentially combine thousands of samples, which is sufficient for our denoising task.

Figures 10.7, 10.8, and 10.9 show a visual comparison of the results and the associated SSIM scores. Figure 10.10 visually shows the results on imagery with realworld noise.



**Fig. 10.7** From left to right, image with additive Gaussian noise with  $\sigma_n = 85$  (SSIM:0.065),  $7 \times 7$  NLM (SSIM:0.799),  $15 \times 15$  NLM (SSIM:0.918), and pull-push NLM (SSIM:0.935)



**Fig. 10.8** From left to right, image with additive Gaussian noise with  $\sigma_n = 60$  (SSIM:0.132),  $7 \times 7$  NLM (SSIM:0.873),  $15 \times 15$  NLM (SSIM:0.927), and pull-push NLM (SSIM:0.935)

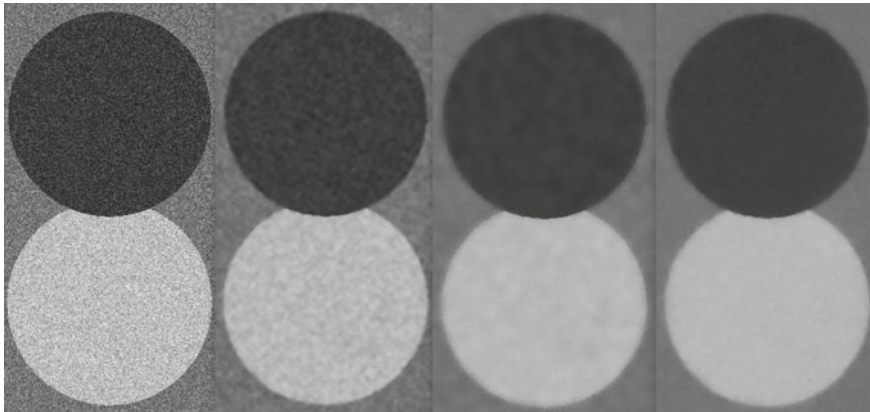
### 10.6.2 Joint Pull-Push NLM Experiments

In order to show how joint pull-push NLM filtering works, we compare it to the standard pull-push NLM algorithm and show how the addition of a reference image can improve the results.

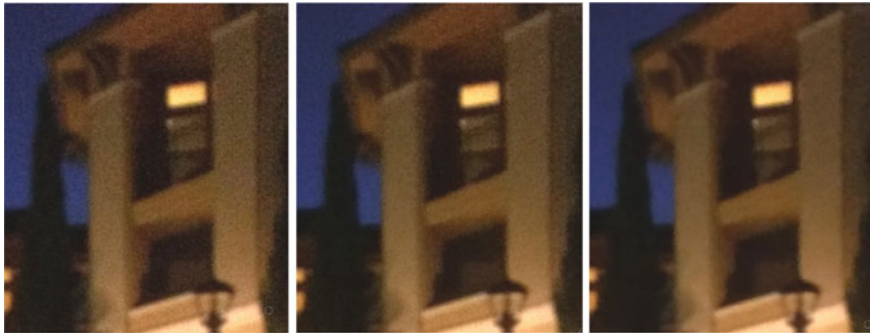
The reference image used for joint filtering is just the luminance channel of the original image. The noisy image to be denoised is the color image with synthetic Gaussian noise ( $\sigma_n = 30$ ).

Figure 10.11 shows the input images and results. For the standard pull-push NLM, the  $\sigma_s$  parameter resulting in the best reconstruction (SSIM:0.765) was  $\sigma_s = 27$ . For joint pull-push NLM, the  $\sigma_s$  parameter resulting in the best reconstruction





**Fig. 10.9** From left to right, synthetic three-color image with additive Gaussian noise with  $\sigma_n = 60$  (SSIM:0.082),  $7 \times 7$  NLM (SSIM:0.834),  $15 \times 15$  NLM (SSIM:0.945), and pull-push NLM (SSIM:0.973)



**Fig. 10.10** Real-world example: The leftmost image is an input burst frame taken in low light, and the center image shows NLM filtering using a  $7 \times 7$  kernel. The rightmost frame shows the results of the pull-push NLM approach

(SSIM:0.781) was  $\sigma_s = 14$ . The reference image has less noise than the input images and is used to derive the weights for combining similar samples for denoising. Because of this, the  $\sigma_s$  which gives the best result is a function of the noise standard deviation of both the reference image and the image to filter.

### 10.6.3 Burst Processing Experiments

To demonstrate burst processing for pull-push NLM, we show results in a synthetic sequence with both background and foreground motion. The background is moved to the up and left direction for each frame in the burst while an overlay image (roughly



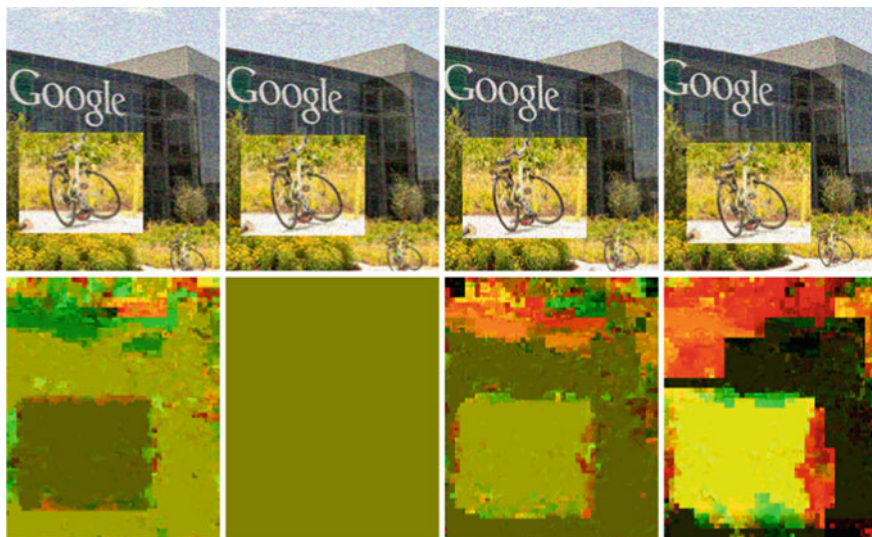


**Fig. 10.11** The upper left image shows a highly noisy image with AWGN  $\sigma_n = 30$  (SSIM:0.488). The upper right image shows a noiseless reference frame. The lower left image shows the best result using pull-push NLM (SSIM:0.765). The lower right image shows the result of joint pull-push NLM (SSIM:0.784). Notice the improvement in quality around the wheels and shadow of the bicycle

1/4 the size of the image) is moved down and to the right. Figure 10.12 shows the estimated motion field from several frames of this burst. The red and green channels display the per-pixel motion in x and y. The center intensity value signifies no motion. The second image is considered the base frame that the motion of other frames are estimated with respect to, and by definition has no motion. You can see regions with texture are mostly well tracked. Flat regions like the sky are unable to be well estimated. However, since the region is flat the motion offsets are not necessary, since useful patches for denoising will still be found.

Figure 10.13 shows a comparison of single frame pull-push NLM processing and the two burst-based pull-push NLM approaches.

Single frame pull-push NLM on the base frame does a decent job of denoising, but in highly textured regions, denoising is limited. The best quality (SSIM:0.844) was achieved using a pull-push sigma parameter,  $\sigma_s = 20$ .



**Fig. 10.12** The upper row image shows four images out of a burst of nine images with AWGN  $\sigma_n = 20$  (SSIM:0.574). The background is scrolling left and upward while a sprite overlay (the large bicycle) is being moved down and to the right. The lower row shows the estimated motion compensation field

Using the straightforward approach to burst processing provides better results (SSIM:0.868), but in highly textured regions, it is only able to incorporate nearby frames due to limited motion compensation capability. Since more patches are available from additional frames, the best quality was achieved using setting  $\sigma_s = 16$ . Because of the lower sigma, more low contrast detail can be preserved as can be seen in the windows of the building.

Using motion compensated burst processing allows for even better quality (SSIM:0.900), since textured regions in all frames are able to contribute to denoising. The best result was achieved using  $\sigma_s = 15$ . Lower  $\sigma_s$  indicates that having more useful candidate patches allows for more selectivity in which patches to incorporate, which results in even better detail preservation while denoising.

The next experiment uses a collection of luminance reference images in addition to the burst images from the last experiment as input to the joint burst pull-push NLM method. Note that we also use our motion compensation in conjunction with the joint burst method.

Results can be seen in Figs. 10.14, 10.15, 10.16 and 10.17. The joint burst method improves the SSIM score even further with the best quality result being (SSIM:0.940). The increased accuracy of the reference images using in the joint method versus the noisy input images results in better quality motion estimation. Not only that but also



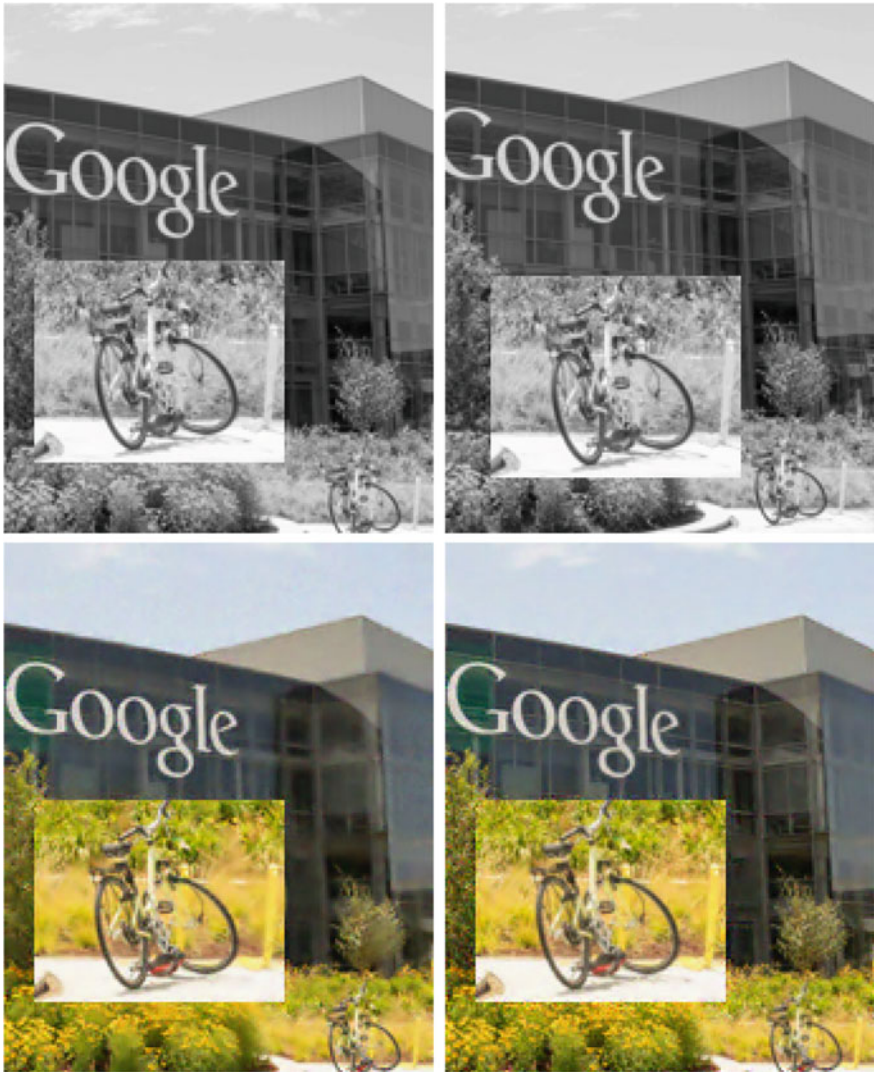
**Fig. 10.13** Burst pull-push NLM: The upper left image shows the noiseless base frame of the nine frame burst from the previous figure. The upper right image shows the result of single frame pull-push NLM on this frame (SSIM:0.844). Using the simple approach to burst processing (lower left) provides better results (SSIM:0.868), but in highly textured regions is only able to incorporate nearby frames due to limited motion compensation. Using the motion compensated burst processing (lower right) allows for the best quality result (SSIM:0.900)





**Fig. 10.14** Real-world example of burst pull-push NLM: The top image is one of a collection of seven input burst frames. The center image uses single frame pull-push NLM denoising with the best result obtained using  $\sigma_s = 11$ . The bottom frame uses burst pull-push NLM with motion compensation and is able to retain more detail with the same level of denoising. The best result was obtained with  $\sigma_s = 7$

compared to the ground truth allows for a much tighter and selective sigma. Not only that but also the reference images allow for better quality motion estimation which also improves the results. These two factors allowed for a more selective  $\sigma_s$  parameter to be used. The peak quality result of joint burst pull-push NLM in terms of SSIM (SSIM:0.940) was achieved by setting  $\sigma_s = 5$ .



**Fig. 10.15** Joint burst pull-push NLM: The following shows a proof of concept for joint burst pull-push NLM. A noiseless luminance-only reference burst sequence is used as additional input to the noisy burst sequence in the previous example. The top row shows two frames from the reference burst (the base frame and the last frame used.) The lower left image shows the previous motion compensated burst result (SSIM:0.900). The lower right image shows the result using the reference images and the joint burst pull-push NLM method. The resulting image has better quality (SSIM:0.940). Notice how more detail can be seen in low contrast areas such as the window frames



**Fig. 10.16** This figure shows the results with much higher added synthetic noise level  $\sigma_n=60$ . The first row shows, the noisy base frame (SSIM:0.239), and non-motion compensated pull-push burst NLM (SSIM:0.619). The second row shows motion compensated pull-push burst NLM (SSIM:0.675) and motion compensated joint burst pull-push NLM (SSIM:0.805)





**Fig. 10.17** This figure shows the results on a real-world strobe burst. Four image pairs were captured, each with and without flash. The first row shows one of the pairs, and the non-flash image was taken under low light and has high noise. The next row shows the result of burst pull-push NLM which does a good job of denoising. The final row shows the improvement gained by using the reference frames with joint burst pull-push NLM processing. The amount of denoising is comparable, but the final result is sharper and more accurate

## 10.7 Conclusion

We have presented a framework to perform NLM denoising with large spatial kernels efficiently using a pull-push NLM algorithm. We are able to achieve much better filtering results than the brute force NLM algorithm for a given computational budget. In addition, we describe novel extensions to our approach for both joint and burst processing. The pull-push approach is especially well suited to burst processing since the multiscale NLM patch comparison computation is reusable for multiscale motion compensation. This synergy improves the quality and efficiency of the implementation significantly.

One minor limitation of the proposed algorithm is that repetitive patterns within an image may not receive as much filtering as in a brute force NLM algorithm with a large kernel size. However, pull-push is capable of better noise reduction in smooth regions where the sensor noise tends to be the most noticeable.

## References

1. Adams A, Baek J, Davis MA (2010) Fast high-dimensional filtering using the permutohedral lattice. *Comput Graph Forum* 29–2:753–762
2. Buades A, Coll B, Morel JM (2005) A review of image denoising algorithms, with a new one. *Multiscale Model Simul (SIAM Interdisciplinary Journal)* 4(2):490–530
3. Chen J, Paris S, Durand F (2007) Real-time edge-aware imageprocessing with the bilateral grid. *ACM Trans Graph* 26(3)
4. Condat L (2010) A simple trick to speed up and improve the non-local means, research Report hal-00512801
5. Darbon J, Cunha A, Chan TF, Osher S, Jensen GJ (2008) Fast nonlocal filtering applied to electron cryomicroscopy. In: *International symposium on biomedical imaging (ISBI)*. IEEE, pp 1331–1334
6. Eisemann E, Durand F (2004) Flash photography enhancement via intrinsic relighting. *ACM Trans Graph* 23(3):673–678
7. Farbman Z, Fattal R, Lischinski D, Szeliski R (2008) Edge-preserving decompositions for multi-scale tone and detail manipulation. *ACM Trans Graph* 27(3)
8. Fattal R (2009) Edge-avoiding wavelets and their applications. *ACM Trans Graph* 28(3):1–10
9. Gastal ESL, Oliveira MM (2011) Domain transform for edge-aware image and video processing. *ACM Trans Graph* 30(4):69:1–69:12
10. Gortler SJ, Grzeszczuk R, Szeliski R, Cohen MF (1996) The lumigraph. In: *Proceedings of SIGGRAPH 96*. ACM, pp 43–54
11. Hartung J, Knapp G, Sinha BK (2008) *Statistical meta-analysis with applications*. Wiley. ISBN 978-0-470-29089-7
12. Isidoro JR, Milanfar P (2016) A pull-push method for fast non-local means filtering. In: *2016 IEEE international conference on image processing (ICIP)*, pp 1968–1972
13. Kay SM (1993) *Fundamentals of statistical signal processing—estimation theory*. In: *Signal processing series*. PTR Prentice-Hall, Englewood Cliffs, N.J
14. Kervrann C, Boulanger J (2008) Local adaptivity to variable smoothness for exemplar-based image regularization and representation. *Int J Comput Vis* 79(1):45–69
15. Kopf J, Cohen MF, Lischinski D, Uyttendaele M (2007) Joint bilateral upsampling. *ACM Trans Graph* 26(3)
16. Kraus M (2009) The pull-push algorithm revisited. In: *Proceedings GRAPP 2009*



17. Liu X, Feng X, Han Y (2013) Multiscale nonlocal means for image denoising. In: 2013 international conference on wavelet analysis and pattern recognition (ICWAPR)
18. Milanfar P (2013) A tour of modern image filtering. *IEEE Signal Process Mag* 30(1):106–128
19. Nercessian S, Panetta KA, Aghaian SS (2012) A multi-scale non-local means algorithm for image de-noising. *Proc SPIE* 8406:84,060J–84,060J–10
20. Paris S, Durand F (2009) A fast approximation of the bilateral filter using a signal processing approach. *Int J Comput Vis* 81(1):24–52
21. Pesquet-Popescu B, Cagnazzo M, Dufaux F (2016) Motion estimation techniques. TELECOM ParisTech, pp 33–34
22. Petschnigg G, Agrawala M, Hoppe H, Szeliski R, Cohen M, Toyama K (2004) Digital photography with flash and no-flash image pairs. *ACM Trans Graph (Proc SIGGRAPH)*, pp 664–672
23. Pham TQ, Vliet LJ (2005) Separable bilateral filtering for fast video preprocessing. In: In IEEE international conference on multimedia and Expo, CD1–4. IEEE, pp 1–4
24. Protter M, Elad M, Takeda H, Milanfar P (2009) Generalizing the non-local-means to super-resolution reconstruction. *IEEE Trans Image Process* 36
25. Ragan-Kelley J, Barnes C, Adams A, Paris S, Durand F, Amarasinghe S (2013) Halide: a language and compiler for optimizing parallelism, locality, and recomputation in image processing pipelines. *SIGPLAN Not* 48(6):519–530
26. Romano Y, Elad M, Milanfar P (2017) The little engine that could: regularization by denoising (red). *SIAM J Imag Sci* 10(4):1804–1844
27. She Q, ZLu, Li W, Liao Q (2014) Multigrid bilateral filtering. *IEICE Trans Inf Syst* 2748–2759
28. Smith SM, Brady JM (1997) Susan—a new approach to low level image processing. *Int J Comput Vis* 23(1):45–78
29. Suominen O, Gotchev A (2015) Preserving natural scene lighting by strobe-lit video. In: *Image processing: algorithms and systems XIII*, SPIE, vol 9399
30. Takeda H, Milanfar P, Protter M, Elad M (2009) Super-resolution without explicit subpixel motion estimation. *IEEE Trans Image Process* 18(9):1958–1975
31. Talebi H, Milanfar P (2014) Global image denoising. *IEEE Trans Image Process* 23(2):755–768
32. Talebi H, Milanfar P (2014) Nonlocal image editing. *IEEE Trans Image Process* 23(10):4460–4473
33. Talebi H, Milanfar P (2016) Fast multi-layer Laplacian enhancement. *IEEE Trans Comput Imag*
34. Talebi H, Zhu X, Milanfar P (2013) How to SAIF-ly boost denoising performance. *IEEE Trans Image Process* 22(4):1470–1485
35. Tomasi C, Manduchi R (1998) Bilateral filtering for gray and color images. *ICCV*. Bombay, India, pp 836–846
36. Wang Z, Bovik AC, Sheikh HR, Simoncelli EP (2004) Image quality assessment: from error visibility to structural similarity. *IEEE Trans Image Process* 13(4):600–612
37. Wong TS, Milanfar P (2016) Turbo denoising for mobile photographic applications. In: 2016 IEEE international conference on image processing, ICIP 2016, pp 988–992
38. Yang Q (2012) Recursive bilateral filtering. In: *ECCV*, pp 399–413
39. Zhang M, Gunturk BK (2008) Multiresolution bilateral filtering for image denoising. *IEEE Trans Image Process* 17(12):2324–2333
40. Zontak M, Mosseri I, Irani M (2013) Separating signal from noise using patch recurrence across scales. In: *IEEE conference on computer vision and pattern recognition*, pp 1195–1202

# Chapter 11

## Three Approaches to Improve Denoising Results that Do Not Involve Developing New Denoising Methods



Gabriela Ghimpeteanu, Thomas Batard, Stacey Levine  
and Marcelo Bertalmío

**Abstract** Image denoising has been a topic extensively investigated over the last three decades and, as repeatedly shown in this book, denoising algorithms have become incredibly good, so much so that many researchers have started questioning the need to further pursue this line of research. In this chapter, we argue that there is indeed room for improvement of denoising results, and we propose three different avenues to explore, none of which requires the development of new denoising methods. First, we describe how it can be better to denoise a transform of the noisy image rather than denoise the noisy image directly. We mention several possible transforms, and an open problem is to find a transform that is optimal for denoising, according to a proper image quality metric. Next, we point out the importance of having a proper noise model for JPEG pictures, so that a variance stabilization transform can be developed that transforms noise in JPEG images into additive white Gaussian noise, enabling existing denoising methods to be properly applied to the JPEG case. Finally, we highlight the fact that while virtually all denoising methods are optimized and validated in terms of the PSNR or SSIM measures, these metrics are not well correlated with perceived image quality, and therefore, it could be best

---

G. Ghimpeteanu (✉) · M. Bertalmío  
Universidad Pompeu Fabra, Barcelona, Spain  
e-mail: gabriela.ghimpeteanu@upf.edu

M. Bertalmío  
e-mail: marcelo.bertalmio@upf.edu

T. Batard  
TU Kaiserslautern, Kaiserslautern, Germany  
e-mail: tomas.batard@gmail.com

S. Levine  
Duquesne University, Pittsburgh, PA, USA  
e-mail: sel@mathcs.duq.edu

to optimize the parameter values of denoising methods according to subjective testing. A remaining challenge is to develop perceptually based image quality metrics that match observer preference.

## 11.1 Introduction

In this chapter we propose, in order to improve denoising results, to explore three different avenues that do not require the development of new denoising methods.

First, we review recent works that improve the performance of denoising algorithms by applying them to transforms of image data instead of applying them directly to the noisy image. An open challenge is then to find a transform that is optimal for the denoising problem, according to a proper image quality metric.

Second, we show how not only the performance but also the ranking of denoising algorithms is different in the real noise scenario than when working under the common assumption that noise is additive white Gaussian (AWG) of known variance which is fixed and independent from the image values. A second way to improve denoising results would then be to develop a noise model for JPEG pictures and a corresponding variance stabilization transformation, so that existing denoising methods that assume AWG noise can be properly applied to the JPEG case.

Finally, we note that although the PSNR and SSIM error measures are not correlated with perceived image quality, virtually all denoising methods are optimized and validated in terms of these metrics. This suggests a third approach to improve denoising results, that of developing a perceptually based image quality metric that matches observer preference, or using subjective experiments to select the optimal parameter values for denoising methods.

## 11.2 Denoise a Transform of an Image Instead of the Image Itself

There are often benefits to processing a linear or nonlinear transform of an image rather than processing the observed image data directly. In the context of image denoising, this has traditionally taken the form of thresholding Fourier or wavelet coefficients, which then has a denoising effect on the underlying image data. The line of research we follow here is different in that we specifically *apply an image denoising method* to a transform of an image which in turn has a denoising effect on the original image data. The latter differs from the former in several ways. First, the choice of processing applied to the transform is different, the latter being one that is borrowed from algorithms for denoising the image data directly while the former, such as thresholding, is not. Second, the latter may require a specially designed mechanism for reconstructing the denoised image, particularly for nonlinear transforms.

The first instance we know of applying an image denoising technique to an image transform is the work of Lysaker et al. [30] in which the authors propose using a constrained total variation (TV) minimization problem to denoise the unit normal vector field of a noisy image surface. The denoised image is then reconstructed using a variational approach whose solution has a unit normal field that matches the results of this constrained TV problem. Similar denoising strategies are presented in [41] and [20] where the unit tangent field to an image surface is denoised, allowing for a mathematically sound model. The approach in [30] is also directly related to the Bregman iterative algorithm of Osher et al. [38].

A similar approach was proposed by Bertalmío and Levine [8]. Justified by the fact that the curvature of the level lines of a gray-level image has a higher SNR along likely edges than the image itself (in the context of AWG noise), they demonstrate that reconstructing an image from its denoised level line curvature is consistently more effective than denoising the image directly. This is confirmed by experiments with four denoising methods: TV denoising performed through both gradient descent [43] and the Bregman iterative algorithm [38], orientation matching using smoothed unit tangents [20], nonlocal means (NLM) [11] and block-matching and 3D filtering (BM3D) [14]. A variational method was successful for the reconstruction step.

Batard and Berthier [5] introduced a moving frame approach in the process of describing a Fourier theory for  $n$ -channel images which takes into account the local geometry of an image. Their orthonormal moving frame in  $\mathbb{R}^{n+2}$ , defined over the image domain, consists of two vector fields that are tangent to the image graph and  $n$  components that are normal to the surface. Their idea was to construct the components of an image in this moving frame, compute the standard 2D Fourier transform of each of the  $n+2$  components, apply a different Gaussian kernel to each one, and finally project back. By applying Euclidean heat diffusion to each component, the output is a filtered image that retains its local geometry after the diffusion step.

Batard and Bertalmío [6, 7] used this moving frame approach for image denoising. Instead of directly applying a denoising method to an image, they proposed to apply it to its components in the moving frame described above. The authors used the vectorial extension of the total variation-based denoising method of Rudin et al. [43] proposed by Blomgren and Chan [9], and the vectorial total variation (VTV) denoising method of Bresson and Chan [10]. In both cases, this strategy produced better results in terms of PSNR than denoising the image directly with these approaches.

In this section, we revisit the work of [16] and detail how to improve the result of a denoising method by denoising the components of an image in a moving frame, as in [5–7], instead of denoising the original noisy image directly. We note a theoretical analysis of why with this approach we can expect cleaner results with better preserved details, regardless of denoising algorithm. We validate the consistency of this approach by showing how the moving frame strategy brings an improvement in terms of PSNR and SSIM for three different noise removal techniques: a local variational method (VTV, [10]), a patch-based method (NLM [11]), and a method combining patch based processing with filtering in the spectral domain (BM3D [14]). The approach in [16] has the advantage of simplicity in the reconstruction step which is expressed as a matrix transform, in comparison to the similar curvature-based strat-

egy in [8] or the vector fields smoothing techniques of [20, 30, 38, 41] which require solving a second- or third-order PDE evolution equation for the reconstruction step.

### 11.2.1 Image Decomposition in a Moving Frame

We denote a gray-level image by  $I: \Omega \subset \mathbb{R}^2 \rightarrow \mathbb{R}$ , and the standard coordinate system of  $\mathbb{R}^2$  by  $(x, y)$ .  $I_x$  and  $I_y$  are the derivatives of  $I$  with respect to  $x$  and  $y$ , respectively, and  $\nabla I$  is the gradient of  $I$ . We construct an image decomposition model for  $I$  comprised of two steps. First, construct an orthonormal moving frame  $(Z_1, Z_2, N)$  of  $(\mathbb{R}^3, \|\cdot\|_2)$  over  $\Omega$  that takes into account the local geometry of  $I$ . Next, calculate the components  $(J^1, J^2, J^3)$  of the  $\mathbb{R}^3$ -valued function  $(0, 0, I)$  in that moving frame.

We use  $\mu I$  (a scaled version of  $I$ , with  $\mu \in ]0, 1]$ ), and its graph, given by the surface  $S$  in  $\mathbb{R}^3$  parametrized by

$$\psi : (x, y) \mapsto (x, y, \mu I(x, y)). \tag{11.1}$$

We construct an orthonormal moving frame  $(Z_1, Z_2, N)$  by choosing  $Z_1$  to be the vector field tangent to the surface  $S$  that points in the direction of the steepest slope at each point of  $S$ , and  $Z_2$  to be the vector field tangent to  $S$  that points in the direction of the lowest slope at each point of  $S$ . To complete the orthonormal frame, the component  $N$  is the unit normal to the surface.

This can be realized by considering the gradient of  $\mu I$ ,  $z_1 = (\mu I_x, \mu I_y)^T$ , and the vector indicating the direction of the level lines of  $\mu I$ ,  $z_2 = (-\mu I_y, \mu I_x)^T$ . On smooth regions of  $I$  where  $I_x(x, y) = I_y(x, y) = 0$ , we fix  $z_1 = (1, 0)^T$  and  $z_2 = (0, 1)^T$ . Then  $Z_1$  and  $Z_2$  are defined as

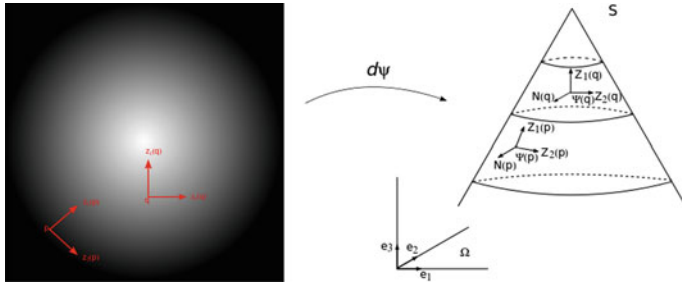
$$Z_i = \frac{d\psi(z_i)}{\|d\psi(z_i)\|_2}, \quad i = 1, 2, \tag{11.2}$$

where  $\psi$  maps vector fields from  $\Omega$  to tangent vector fields of  $S$ , and  $d\psi$  denotes its differential. The unit normal  $N$  is given by the vectorial product between  $Z_1$  and  $Z_2$ .

More explicitly, the coordinates of the vector fields  $Z_1, Z_2, N$  are given by the first, second, and third columns of the matrix field

$$P = \begin{pmatrix} \frac{I_x}{\sqrt{|\nabla I|^2(1 + \mu^2|\nabla I|^2)}} & \frac{-I_y}{|\nabla I|} & \frac{-\mu I_x}{\sqrt{1 + \mu^2|\nabla I|^2}} \\ \frac{I_y}{\sqrt{|\nabla I|^2(1 + \mu^2|\nabla I|^2)}} & \frac{I_x}{|\nabla I|} & \frac{-\mu I_y}{\sqrt{1 + \mu^2|\nabla I|^2}} \\ \frac{\mu|\nabla I|^2}{\sqrt{|\nabla I|^2(1 + \mu^2|\nabla I|^2)}} & 0 & \frac{1}{\sqrt{1 + \mu^2|\nabla I|^2}} \end{pmatrix} \tag{11.3}$$

Figure 11.1 shows the moving frames  $(z_1, z_2)$  and  $(Z_1, Z_2, N)$  associated to a simple image. On the left, the figure illustrates the moving frame  $(z_1, z_2)$  at the points  $p$  and



**Fig. 11.1** Moving frame encoding the local geometry of a gray-level image. Left: original gray-level image and a moving frame  $(z_1, z_2)$  indicating the direction of the gradient and the level line of the image at two points  $p$  and  $q$  of the image domain  $\Omega$ . Right: the orthonormal moving frame  $(Z_1, Z_2, N)$  of  $(\mathbb{R}^3, \|\cdot\|_2)$  over  $\Omega$  indicating the direction of the steepest and lowest slopes of the surface  $S$ , for some smoothing parameter  $\mu$ , at the points  $\psi(p)$  and  $\psi(q)$

$q$  of the domain  $\Omega$ , and on the right, it shows the induced moving frame  $(Z_1, Z_2, N)$  constructed on the surface  $S$  at the points  $\psi(p)$  and  $\psi(q)$ .

If  $(e_1, e_2, e_3)$  represents the orthonormal frame of  $(\mathbb{R}^3, \|\cdot\|_2)$ , where  $e_1 = (1, 0, 0)$ ,  $e_2 = (0, 1, 0)$ , and  $e_3 = (0, 0, 1)$ , then the matrix  $P$  in (11.3) can be seen as the frame change field from  $(e_1, e_2, e_3)$  to  $(Z_1, Z_2, N)$ . More precisely, the components of the  $\mathbb{R}^3$ -valued function  $(0, 0, I)$  in the new frame are given by  $(J^1, J^2, J^3)$ , where

$$(J^1, J^2, J^3)^T = P^{-1}(0, 0, I)^T. \tag{11.4}$$

From (11.3) and (11.4), we see that these components can be explicitly expressed as

$$J^1(I) = \frac{\mu I |\nabla I|}{\sqrt{1 + \mu^2 |\nabla I|^2}}, \quad J^2(I) = 0, \quad \text{and} \quad J^3(I) = \frac{I}{\sqrt{1 + \mu^2 |\nabla I|^2}}. \tag{11.5}$$

Figure 11.2 illustrates the gray-level test image “castle” and its components  $J^1$  and  $J^3$  computed for  $\mu = 0.05$ . The component  $J^1$  contains edge and texture information due to its inclusion of the norm of the image gradient. The component  $J^3$  better reproduces the original image, from which the norm of the gradient has been diminished.

The multichannel case involves embedding an  $n$ -channel image into  $\mathbb{R}^{n+2}$ , then using similarly motivated choices for  $Z_1$  and  $Z_2$  with extra care taken in the construction of the remaining normal vector fields. Details can be found in [16].

An important role is given to the parameter  $\mu$ . It represents a smoothing parameter for the moving frame associated to the image  $I$ . In the next sections, we analyze just how crucial the value of this parameter is for image denoising.



**Fig. 11.2** From left to right: gray-level image “castle”, component  $J^1$ , and component  $J^3$

### 11.2.2 Application to Image Denoising

We propose denoising the components of an image in the moving frame described above instead of denoising the image directly. The effect on image denoising using these two approaches can be compared by performing the following experiment:

1. Denoise  $I$  with some method and denote the output image  $I_{den}$ .
2. Compute the components of  $I$  in the moving frame. Apply the same denoising method to the components to obtain the processed components. Then, apply the inverse frame change matrix field to the processed components, obtaining a reconstructed image denoted by  $I_{denMF}$ . In the grayscale case, this can be explicitly expressed as

$$I_{denMF} = P_{13}(I)J^1(I)_{den} + P_{33}(I)J^3(I)_{den}. \quad (11.6)$$

3. Compare  $I_{den}$  and  $I_{denMF}$  using PSNR, SSIM or any image quality metric.

The extension from single to multichannel denoising algorithms is not always straightforward. Depending on the denoising method, there are several ways to do this, given by the use of different color spaces and the manner in which to apply the algorithm (channel-wise, only to selected channels, or vectorially). Details can be found in [16].

It is interesting to note that the denoising approach introduced above can in fact be used with any moving frame. Several choices were previously analyzed in [7], showing a similar output quality is attained when  $Z_1, Z_2$  are any randomly chosen, but orthonormal, vector fields in the tangent planes of the surface parametrized by (11.1) for gray-level images. However, when  $Z_1, Z_2$  are not in the tangent space, the results are of low quality.

**Table 11.1** Average values of the PSNR for the components  $J^1, J^3$  and the image  $I$  over the Kodak database for different noise levels and values of the parameter  $\mu$

Noise level	Function	$\mu = 1$	$\mu = 0.1$	$\mu = 0.01$	$\mu = 0.005$	$\mu = 0.001$	$\mu = 0.0001$
$\sigma = 5$	Component $J^1$	20.51	20.09	34.37	37.84	40.17	40.31
	Component $J^3$	18.56	26.02	34.24	34.22	34.19	34.19
	Image $I$	34.19	34.19	34.19	34.19	34.19	34.19
$\sigma = 10$	Component $J^1$	19.34	15.96	28.21	31.51	33.84	33.97
	Component $J^3$	16.94	19.84	28.27	28.24	28.21	28.21
	Image $I$	28.21	28.21	28.21	28.21	28.21	28.21
$\sigma = 15$	Component $J^1$	18.32	14.16	24.44	27.79	30.09	30.22
	Component $J^3$	16.32	16.93	24.80	24.77	24.73	24.73
	Image $I$	24.73	24.73	24.73	24.73	24.73	24.73
$\sigma = 20$	Component $J^1$	17.37	13.10	21.86	25.12	27.38	27.51
	Component $J^3$	15.98	15.22	22.38	22.33	22.28	22.27
	Image $I$	22.27	22.27	22.27	22.27	22.27	22.27
$\sigma = 25$	Component $J^1$	16.47	12.36	19.89	23.03	25.25	25.38
	Component $J^3$	15.77	14.10	20.50	20.44	20.37	20.37
	Image $I$	20.37	20.37	20.37	20.37	20.37	20.37

### 11.2.3 The Noise Level Is Higher on the Intensity Values of a Gray-Level Image Than on Its Components in a Well-Chosen Moving Frame

In this section, we study how for carefully selected  $\mu$  values, the components  $J^1(I)$  and  $J^3(I)$  of a gray-level image  $I$  in the moving frame (11.3), given by (11.4) and (11.5), are less degraded by AWG image noise than the image itself.

Assume  $I = a + n$  is a gray-level image obtained by adding to the image  $a$  Gaussian noise  $n$  of mean zero and standard deviation  $\sigma$ . Before delving into a more formal analysis, we consider an experiment in which we calculate the PSNR values of the components  $J^1(I)$  and  $J^3(I)$  of the images from the Kodak database [2], for noise levels  $\sigma = 5, 10, 15, 20, 25$  and  $\mu = 1.0, 0.1, 0.01, 0.005, 0.001, 0.0001$ . The results of this experiment are reported in Table 11.1.

Notice that the PSNR values of the components are consistently larger than the PSNR of the image for sufficiently small  $\mu$ . Specifically, the components are generally “less noisy” when  $\mu \in ]0, 0.005]$  for all noise levels considered. For  $\sigma = 5, 10$ , the upper bound of 0.005 can be increased to 0.01.

While the values in Table 11.1 were computed across the entire image, we can more formally study this behavior by considering locations of likely image contours and homogeneous regions separately.



### 11.2.3.1 Edges

As in [8], we attain the following conclusion comparing the PSNR of a grayscale image and its moving frame components along likely image contours.

**Proposition 1** *Using central differences for the approximation of  $\nabla I$  and  $\mu > 0$ , at locations in the image domain where  $|\nabla a| \gg |\nabla n|$ , likely edges of  $I$ ,*

$$PSNR(J^1(I)) \geq PSNR(I) \text{ and } PSNR(J^3(I)) > PSNR(I).$$

The proof, detailed in [16], uses the fact that the explicit representations for the moving frame components  $J^1(I)$  and  $J^3(I)$  given in (11.5) make it possible to approximate their noise as additive along likely contours, leading to the estimates

$$PSNR(J^1(I)) = 20 \log_{10} \left( 255 \frac{127.5\sqrt{2}\mu}{\sqrt{1 + 2\mu^2(127.5)^2}} \times \frac{\sqrt{1 + \mu^2|\nabla I|^2}}{\mu|\nabla I|\sigma} \right) \tag{11.7}$$

$$\geq 20 \log_{10} \left( \frac{255}{\sigma} \right) = PSNR(I) \tag{11.8}$$

and

$$PSNR(J^3(I)) = 20 \log_{10} \left( 255 \frac{\sqrt{1 + \mu^2|\nabla I|^2}}{\sigma} \right) \tag{11.9}$$

$$> 20 \log_{10} \left( \frac{255}{\sigma} \right) = PSNR(I). \tag{11.10}$$

To better understand the role of  $\mu$  along likely contours of  $I$ , (11.9) indicates that  $PSNR(J^3(I))$  is a strictly increasing function of  $\mu$ , tending to  $+\infty$  as  $\mu \rightarrow +\infty$ , and to  $PSNR(I)$  as  $\mu \rightarrow 0$ . On the other hand, (11.7) indicates that  $PSNR(J^1(I))$  is a decreasing function of  $\mu$ , tending to  $PSNR(I)$  as  $\mu \rightarrow +\infty$ , and tending to  $20 \log_{10} \left( \frac{255 \times 127.5 \sqrt{2}}{|\nabla I| \sigma} \right)$  when  $\mu \rightarrow 0$ . Thus, we infer that along image contours, the larger the value of  $\mu$ , the better the estimation of the clean component  $J^3(a)$ , while the smaller the value of  $\mu$ , the better the estimation of the clean component  $J^1(a)$ .

Furthermore, since  $|\nabla I| \approx |\nabla a|$  at contours, we obtain (see (11.3))

$$P_{31}(I) \approx P_{31}(a) \text{ and } P_{33}(I) \approx P_{33}(a). \tag{11.11}$$

Thus (11.6) and (11.11) imply that, along likely contours,

$$\begin{aligned} I_{denMF} &= P_{13}(I)J^1(I)_{den} + P_{33}(I)J^3(I)_{den} \\ &\approx P_{13}(a)J^1(I)_{den} + P_{33}(a)J^3(I)_{den}. \end{aligned} \tag{11.12}$$

From Proposition 1, we deduce that  $J^1(I)_{den}$  and  $J^3(I)_{den}$  are better estimates of  $J^1(a)$  and  $J^3(a)$  than  $I_{den}$  is of  $a$  at these locations. Therefore, from (11.12) and the fact that

$$a = P_{13}(a)J^1(a) + P_{33}(a)J^3(a)$$

we conclude that at likely image contours,  $I_{denMF}$  is a better approximation of  $a$  than  $I_{den}$ . Thus, the value for the parameter  $\mu$  that gives a better reconstruction of image contours in the clean image is strictly positive, since for  $\mu = 0$  we get  $I_{denMF} = I_{den}$ .

### 11.2.3.2 Homogeneous Regions

In this section, we consider the case of homogeneous or slowly varying regions, where  $|\nabla a| \ll |\nabla n|$ . At these locations, we obtain

$$J^1(I) \approx \frac{\mu I |\nabla n|}{\sqrt{1 + \mu^2 |\nabla n|^2}} \quad \text{and} \quad J^3(I) \approx \frac{I}{\sqrt{1 + \mu^2 |\nabla n|^2}}. \quad (11.13)$$

Note that for  $\mu > 0$ , the range, and therefore the variations, of  $J^1(I)$  and  $J^3(I)$  is diminished compared to those of  $I$ .

Moreover, if  $|\nabla a| \ll |\nabla n|$  then

$$\begin{aligned} I_{denMF} &= P_{13}(I)J^1(I)_{den} + P_{33}(I)J^3(I)_{den} \\ &\approx \frac{\mu |\nabla n|}{\sqrt{1 + \mu^2 |\nabla n|^2}} J^1(I)_{den} + \frac{1}{\sqrt{1 + \mu^2 |\nabla n|^2}} J^3(I)_{den}. \end{aligned}$$

Therefore, as  $|\nabla n|$  increases,  $J^1(I)_{den}$  has a larger weight in the reconstruction. This is advantageous, as the results from Table 11.1 suggest that for small values of  $\mu > 0$ , across the entire image  $I$ , including edges, textures, and homogeneous regions, we see a clear trend that

$$PSNR(J^1(I)) > PSNR(I) \quad \text{and} \quad PSNR(J^3(I)) \approx PSNR(I). \quad (11.14)$$

While a formal proof is not trivial, it is reasonable to infer that the proposed moving frame approach should be successful for a carefully chosen  $\mu$  value in homogeneous areas as well. Combining this argument with Proposition 1, one can expect that across the entire image,  $I_{denMF}$  should be at least as good as, if not better than,  $I_{den}$ .

## 11.2.4 Experiments

We report results comparing  $\mu = 0$  and  $\mu > 0$  for three denoising methods, VTV, NLM and BM3D. Although automating  $\mu$  could be a challenge, we found that for the nonlocal algorithms NLM and BM3D a fairly consistent value of  $\mu = 0.001$  gave

**Table 11.2** The gray-level case: variational approach

Approach\noise variance	5	10	15	20	25
PSNR of $I_{den} = ROF(I)$	35.39	31.51	29.48	28.15	27.17
PSNR of $I_{denMF} = P_{13}(I)ROF(J^1(I)) + P_{33}(I)ROF(J^3(I))$	<b>36.36</b>	<b>32.23</b>	<b>30.04</b>	<b>28.60</b>	<b>27.49</b>
SSIM index of $I_{den} = ROF(I)$	93.76	87.07	81.91	77.63	74.12
SSIM index of $I_{denMF} = P_{13}(I)ROF(J^1(I)) + P_{33}(I)ROF(J^3(I))$	<b>94.61</b>	<b>88.37</b>	<b>83.22</b>	<b>78.71</b>	<b>74.78</b>
$\mu$ value used to compute $I_{denMF}$	0.008	0.005	0.005	0.004	0.004

the best results, independent of image content, noise level, and the measure chosen for the denoising evaluation. This is not the case for the local method VTV, for which the optimal  $\mu$  value was highly related to the noise level. The results for VTV were obtained with noise-dependent, optimal values of  $\mu$  (found experimentally), while those for NLM, BM3D all used  $\mu = 0.001$ .

Tables 11.2 and 11.3 summarize the average PSNR and SSIM values of comparing the final image denoising results when the same denoising algorithm is applied to an image directly to obtain  $I_{den}$ , Eq. (11.6) with  $\mu = 0$ , as opposed to its moving frame components to obtain  $I_{denMF}$ , Eq. (11.6) with  $\mu > 0$ . The results are averaged across the grayscale versions of all images in the Kodak database; analogous PSNR and SSIM results for the Kodak database color images are reported in [16] with similarly chosen values of  $\mu$ . It is important to note that while the differences in these image quality metrics diminish as the more powerful denoising techniques are applied, e.g., BM3D, there is still a consistent increase across all noise levels, and this level of increase reflects previously reported mean squared error optimality bounds [28, 29]. Furthermore, while the differences in the image quality metrics may be leveling off, the difference in the resulting image details when comparing these approaches is notable, with more accurate details preserved in the result of denoising the geometrically motivated moving frame components. Visual examples comparing all three denoising algorithms, VTV, NLM and BM3D, can be found in Fig. 11.3.

### 11.2.5 Research Avenue to Explore

We have just seen that it often makes sense to denoise the transform of an image such as its moving frame components, level line curvature, or unit normal vector field. Still, it would be interesting to find a transform that is **optimal** for image denoising. The optimality should be evaluated according to some criterion (not necessarily higher PSNR), and the noise model should carefully reflect the image acquisition model as well; the analysis in this section has assumed AWG noise for simplicity, but the importance of using the correct noise model cannot be overstated, as we detail in the following section.

**Table 11.3** The gray-level case: patch-based approaches ( $\mu = 0.001$ )

Approach\noise variance	5	10	15	20	25
PSNR of $I_{den} = NLM(I)$	37.41	33.38	31.05	30.04	28.91
PSNR of $I_{denMF} = P_{13}(I)NLM(J^1(I)) + P_{33}(I)NLM(J^3(I))$	<b>37.52</b>	<b>33.59</b>	<b>31.57</b>	<b>30.12</b>	<b>29.00</b>
SSIM index of $I_{den} = NLM(I)$	94.96	88.71	82.17	80.34	75.94
SSIM index of $I_{denMF} = P_{13}(I)NLM(J^1(I)) + P_{33}(I)NLM(J^3(I))$	<b>95.11</b>	<b>89.54</b>	<b>85.37</b>	<b>81.03</b>	<b>76.95</b>
PSNR of $I_{den} = BM3D(I)$	38.23	34.34	32.26	30.89	29.88
PSNR of $I_{denMF} = P_{13}(I)BM3D(J^1(I)) + P_{33}(I)BM3D(J^3(I))$	<b>38.25</b>	<b>34.38</b>	<b>32.31</b>	<b>30.93</b>	<b>29.92</b>
SSIM index of $I_{den} = BM3D(I)$	95.71	91.38	87.52	84.19	81.32
SSIM index of $I_{denMF} = P_{13}(I)BM3D(J^1(I)) + P_{33}(I)BM3D(J^3(I))$	<b>95.74</b>	<b>91.49</b>	<b>87.71</b>	<b>84.38</b>	<b>81.44</b>

### 11.3 Have a Proper Noise Model

A key aspect that is overlooked when suggesting that denoising is an almost solved problem is the following: *most denoising methods in the literature are based on modeling noise as being additive and independent from the image data.* In fact, validations and comparisons are normally performed by taking clean photographs as ground truth, creating noisy versions with AWG noise of known variance (fixed and independent from the image values), applying denoising algorithms to them, and comparing each denoised result with the corresponding clean ground truth image using an objective metric such as PSNR. *Everything is dependent on this AWG assumption:* the design of the denoising algorithms, their ranking according to the quality of the outputs, even the computation of the optimality bounds that suggest that state-of-the-art algorithms are close to optimal.

It is well known that noise in regular output images, in JPEG format, is not AWG, but despite this fact, the vast majority of the denoising literature implicitly assumes that this difference should not have an impact on how we address the denoising problem, nor on how results are validated or methods compared. A few exceptions in the literature are, for instance:

- Nam et al. [37] propose a quite complex cross-channel noise model for JPEG images, and a neural network to estimate the noise model parameters per camera model and per ISO sensitivity value; this model can then be used to improve denoising results.



**Fig. 11.3** Row 1:  $VTV$  result for image 13 [2] with  $\sigma = 15$ . Left: noisy image,  $I$ . Middle: results of denoising  $I$  directly,  $I_{den} = VTV(I)$ , PSNR=27.00. Right: moving frame result (11.6),  $I_{denMF}$  with  $J^i(I)_{den} = VTV(J^i(I))$ , PSNR=27.56. Row 2:  $NLM$  results for image 15 [2] with  $\sigma = 20$ . Left:  $I$ . Middle:  $I_{den} = NLM(I)$ , PSNR=29.29. Right:  $I_{denMF}$  with  $J^i(I)_{den} = NLM(J^i(I))$ , PSNR=29.70. Row 3:  $BM3D$  result for image 24 [2] with  $\sigma = 20$ . Left:  $I$ . Middle:  $I_{den} = BM3D(I)$ , PSNR=31.26. Right:  $I_{denMF}$  with  $J^i(I)_{den} = BM3D(J^i(I))$ , PSNR=31.40. The PSNR is computed across the entire image, but the images included here are zoomed-in to better visualize details. Full resolution results can be found in [16]

- The Noise Clinic of Lebrun et al. [25] adapts the nonlocal Bayes approach [26] (that assumes AWG noise) to signal-, scale-, and frequency-dependent noise, which requires an estimate of the covariance matrix of the noise; the authors state that inaccuracies in the estimate of this covariance matrix can introduce artifacts.
- Seybold et al. [44] show how the performance of denoising methods decays drastically when using realistic noise models instead of AWG noise.
- Plötz and Roth [40] compared the BM3D denoising method to several state-of-the-art algorithms for AWG noise images. They concluded that when applied to real noise (for both RAW images and camera outputs), BM3D outperforms (visually and in terms of PSNR) several methods that under the AWG assumption were supposed to perform better.

In this section, we will stress the importance of having a proper noise model for the images one intends to denoise, extending the work presented in [17]. First, we introduce an image database consisting of clean and noisy photo pairs, in formats corresponding to the sensor image (12-bit RAW) and the nonlinear and

uncompressed camera output (8-bit PNG). Next, we use the database to show how both the performance and the ranking of denoising methods are considerably different in the realistic noise scenario as compared with the AWG noise case. Finally, we show how a simple local denoising method, applied to the RAW input, can outperform nonlocal methods applied to the camera output. The local method can also perform as well as low-complexity versions of the nonlocal algorithms applied to the RAW image, but with a much lower computational cost, suggesting the possibility for local denoising to replace more elaborate denoising methods for in-camera implementations.

### 11.3.1 *An Image Database with Clean and Noisy Versions of Photos in RAW and Nonlinear Camera Output Versions*

In order to evaluate and compare denoising methods, we created a test image set containing clean and noisy image pairs with both RAW and nonlinear output versions of each image. Using a Nikon D3100 camera, optimizing exposure, and setting ISO to 100 so as to minimize noise, we capture the twenty “clean” reference RAW images shown in Fig. 11.4. We add noise to each image and then both the clean original and the noisy version are passed through the basic image processing pipeline of a digital camera producing the nonlinear camera outputs. In this way, we have a noisy picture and the corresponding clean reference, which allows us to evaluate denoising results using objective image quality metrics like PSNR.

Recently, Plötz and Roth [40] also proposed a database of real noise photographs and their corresponding ground truth. Each pair in their database has a reference image and a ground truth which are generated from a series of images of the same scene, with different ISO values and exposure times. The reference is chosen to be the photograph taken with low ISO that shows almost no noise, and the ground truth

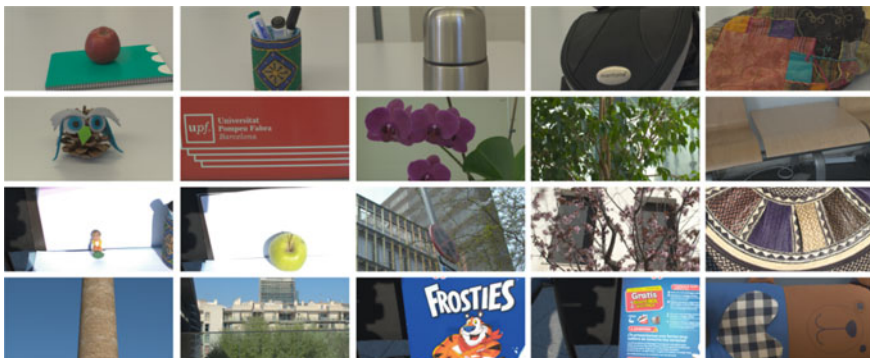


Fig. 11.4 Our image test set

is generated by a post-processing step that corrects for differences in illumination and minor displacements (of objects in the scene or camera shake) between several exposures. Our database has a different approach than the one proposed in [40], in that it includes an estimated RAW noise model so that new noisy images can be generated, with arbitrary noise levels. It also allows the user to introduce new clean images from which additional clean/noisy pairs (in both RAW and nonlinear output versions) can be added to the database. This database is publicly available [1] so that researchers can develop and test denoising algorithms for real-world scenarios.

The creation of our database involves the following components:

- *Simulating the camera processing pipeline.* This pipeline is necessary for processing the image from its RAW format to the camera output (as camera makers do not make public the specific steps each model performs).
- *Creating a signal-dependent noise model estimated at the RAW level.*
- *Validating the results.* A noisy RAW image should have the same appearance as a clean RAW image to which noise has been added according to our estimated model.

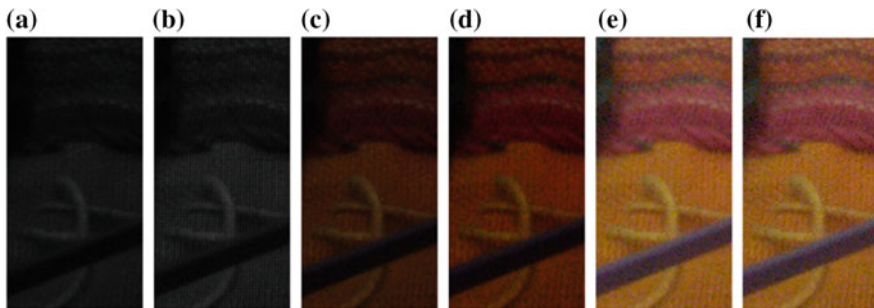
We detail each of these stages in the following subsections.

### 11.3.1.1 Simulate the Camera Processing Pipeline

The following is a concise enumeration of the basic steps of the image processing pipeline, common to digital cameras.

1. **Capture.** A photo is saved in the RAW format as a 12-bit depth image, obtaining the CFA (color filter array) RAW data with a Bayern mosaic pattern. An image example is illustrated in Fig. 11.5a.
2. **White balance.** This process guarantees that the image has no color cast. For neutral colors to keep the correct appearance, a scaling of all intensity values from the RAW file is performed. Figure 11.5b depicts the white balanced image example.
3. **Demosaicking.** The camera sensors produce an image in which for each pixel we only get one of the image channel intensity values (either red or green or blue); demosaicking is an interpolation process that estimates the other two missing values, as exemplified in Fig. 11.5c. For our image processing pipeline, we chose the local demosaicking algorithm proposed by Malvar et al. [34], which is based on bilinear interpolation and further refined by using the correlation among the RGB channels, with Laplacian cross-channel corrections.
4. **Color correction.** This process makes the conversion from the camera color space to sRGB (standard RGB) color space, as illustrated in Fig. 11.5d.
5. **Gamma correction.** In this step, the (normalized) image values are raised to the standard power of  $1/2.2$ . An image example is shown in Fig. 11.5e. This step assures an optimized encoding that models the nonlinear human perception of luminance: more sensitive to details in darker areas.





**Fig. 11.5** Image example to illustrate the camera processing pipeline. From left to right: RAW original image (a), result after applying white balance (b), demosaicking (c), color correction (d), gamma correction (e), and quantizing (f)

6. **Quantization.** This final step (for our purposes) of the pipeline quantizes the image from 12-bit depth to 8-bit depth, outputting an RGB image ready for display, as in Fig. 11.5f.

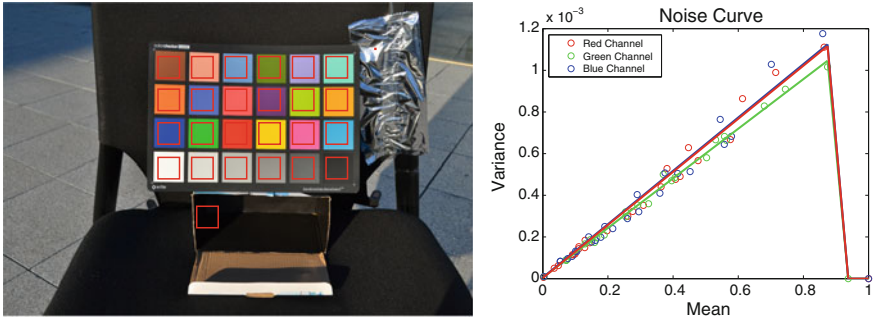
### 11.3.1.2 The Noise Model

For constructing a realistic signal-dependent noise model, estimated on the RAW image, we follow the line of experiments of [19, 44, 46]. We analyze a RAW Colorchecker photograph by segmenting and extracting all of its 24 homogeneous color patches, and computing the noise variance in each color square and for each RAW color channel. Tests on several RAW images of the Colorchecker, taken with different camera settings, allow us to conclude, as in [44] and [46], that the variance as a function of the mean can be fitted by an increasing linear function, indicating that the noise is signal-dependent.

Our noise estimation setup, exemplified by the left side of Fig. 11.6, also includes two objects associated to two extreme cases that the Colorchecker does not take into account: a cardboard box with the interior in shadow and painted with black matte paint, and an aluminum foil that receives direct light and creates specular highlights. The noise variance is then computed from crops from the black box as well as the area with specular highlights. All of the patches from which we compute the variance are marked in red in the left image in Fig. 11.6. Finally, we estimate a noise model from the 26 mean and variance pairs. An example of the channel-wise variance plot as a function of mean pixel value is shown in right side of Fig. 11.6.

To add noise to a clean RAW image, we add to each pixel in the clean RAW image white Gaussian noise with the local variance given by the variance plot value corresponding to the pixel's intensity. Afterward, we apply the rest of the camera pipeline: white balance, demosaicking, color correction, gamma correction and quantization to 8-bit depth.





**Fig. 11.6** The Colorchecker setup: an image of our noise estimation setup captured with ISO 800, with marked regions used for estimating the noise model (left). Plot (example for one set of fixed camera parameters) of variance as function of the mean, for a RAW image scaled between 0 and 1 (right). Values extracted from a RAW Colorchecker image taken with ISO 3200. Dots show the real values obtained for each color square and each channel, while the continuous lines show the fitted linear functions

### 11.3.1.3 Validation of the Noise Model

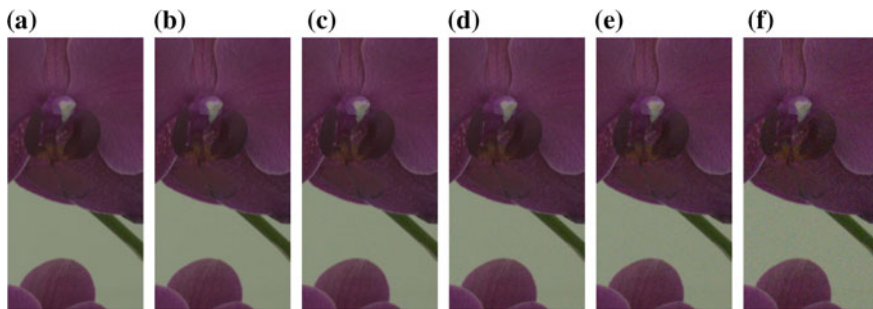
The ISO speed (or ISO sensitivity) estimates the camera sensitivity to light: the higher the value, the higher the sensitivity. The camera transforms the light captured by the sensors into an electrical signal, and increasing the ISO means amplifying the electrical signal before the signal conversion from analog to digital. For example, when increasing the ISO value from 100 to 200, the original electrical signal is doubled. Amplifying the electrical signal better preserves image details. However, this comes with the cost of amplified noise: the higher the ISO speed, the higher the noise level. This justifies our choice of noise levels: we associate one to each possible ISO value. Table 11.4 shows the average standard deviation computed over our test set on the output images for the ISO levels given by our camera. Notice that the highest noise level associated to ISO 3200 produces a relatively small standard deviation.

Figure 11.7 illustrates a crop from an image example from our database, for which the original image and noisy images were created with the realistic noise model in Sect. 11.3.1.2 and then followed the camera processing pipeline in Sect. 11.3.1.1.

Figure 11.8 contains a validation of our noise model, comparing several image examples taken with ISO 3200, which produces the highest noise level.

**Table 11.4** Average noise levels given by different ISO noise curves on our test set

ISO sensitivity	100	400	800	1600	3200
$\sigma$	2.57	3.4	4.24	5.82	8.39



**Fig. 11.7** Crops from a noisy image example from our test set. From left to right: **a** original image, **b–f** synthesized noisy images obtained with the noise curve associated to: **b** ISO 100 ( $\sigma = 2.42$ ), **c** ISO 400 ( $\sigma = 3.17$ ), **d** ISO 800 ( $\sigma = 3.95$ ), **e** ISO 1600 ( $\sigma = 5.43$ ), and **f** ISO 3200 ( $\sigma = 7.98$ )

### 11.3.2 Ranking Denoising Algorithms: AWG Noise Versus Realistic Noise Model

In this section, we present an experiment that shows how essential the noise model is for evaluating denoising methods. For this, we compare three denoising methods applied to camera output images created with two different noise models. We use the patch-based NLM and BM3D denoising methods, implemented with their publicly available IPOL code [12] and [27], and the local VTV-based denoising method.

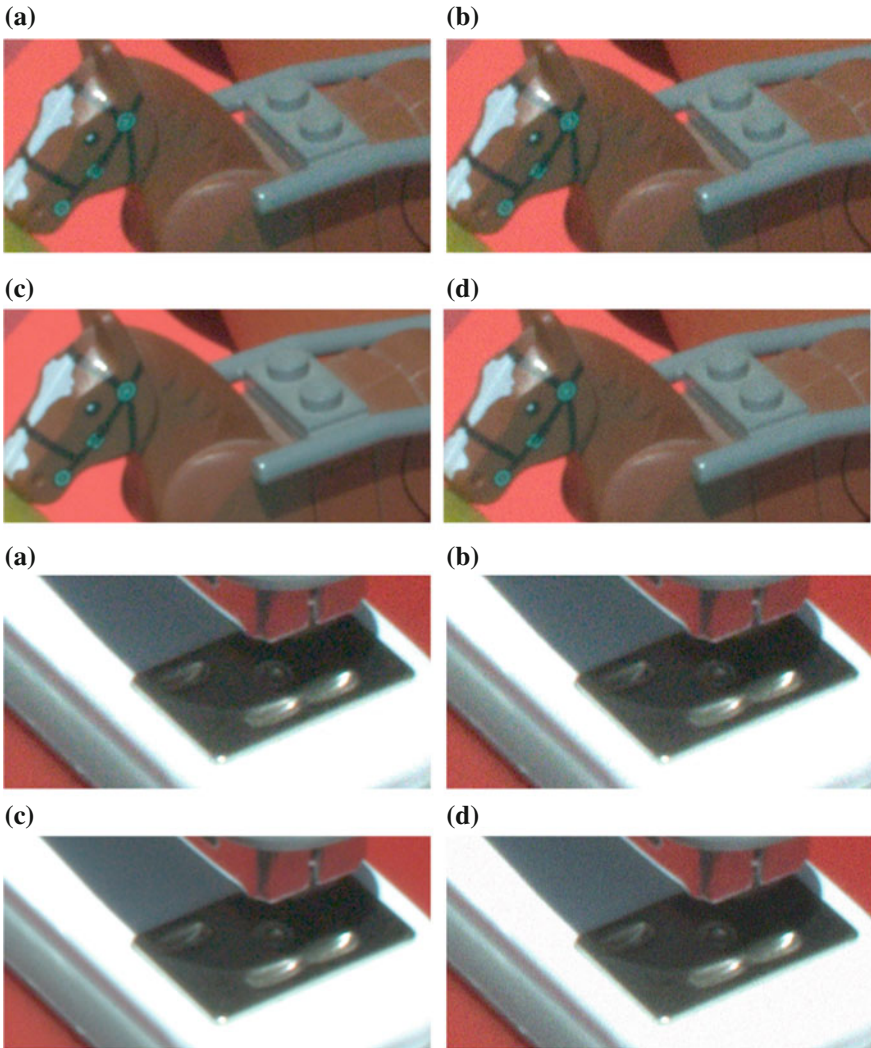
This latter method, proposed by Blomgren and Chan [9], is a vectorial extension of the channel-wise TV-based denoising and consists of replacing the gradient operator acting on each channel by the Jacobian operator acting on the whole image:

$$I_{t+dt} = I_t - dt \mathcal{J}^* \left( \frac{\mathcal{J}(I)}{\sqrt{\|\mathcal{J}(I)\|^2 + \varepsilon}} \right), I_{t=0} = I_0, \quad (11.15)$$

where  $\mathcal{J}$  and  $\mathcal{J}^*$  are, respectively, the Jacobian operator and its adjoint, and  $\varepsilon$  is a small positive constant used to avoid division by 0. We stop the iterative procedure after a fixed number of iterations.

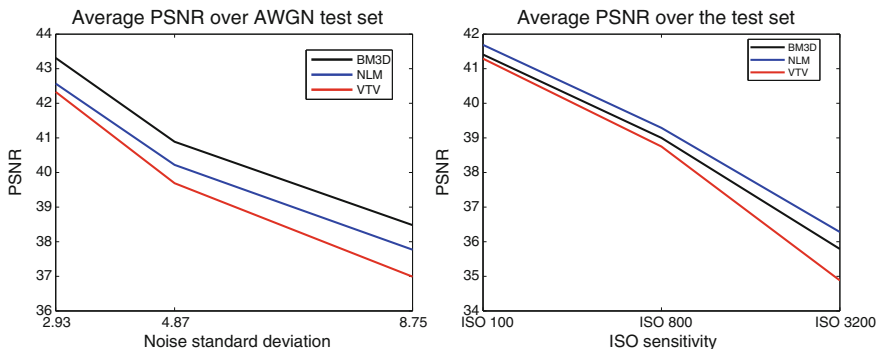
The parameters of these algorithms are the standard deviation of the noise, in the case of NLM and BM3D, and the number of iterations for the VTV-based denoising. We experimentally optimize these parameters for each image and noise level of the database, choosing the ones that maximize the PSNR values of the denoised results. We compute the denoising results under the following two noise models:

1. Starting with a clean RAW image  $I_{cleanRAW}$ , add Gaussian noise, with variance given by the associated noise variance plot as detailed in Sect. 11.3.1.2, to obtain a noisy image  $I_{noisyRAW}$ . For  $I_{cleanRAW}$  and  $I_{noisyRAW}$ , apply white balance, demosaicking, color correction, gamma correction, and quantize to 8 bit to obtain the camera outputs  $I_{clean}$  and  $I_{noisy}$ . Apply NLM, BM3D and VTV-based denoising on  $I_{noisy}$  to obtain the denoised images  $I_{NLM}$ ,  $I_{BM3D}$  and  $I_{VTV}$ .



**Fig. 11.8** Comparing a real noise photograph (a) and synthesized noisy images obtained by adding: **b** Gaussian noise with variance given by our realistic noise model to the RAW image, **c** Gaussian noise of constant variance to the RAW image, and **d** Gaussian noise of constant variance to the camera output

2. Starting with the clean camera output image  $I_{clean}$ , add white Gaussian noise (AWG), with fixed variance described in the next paragraph, to obtain a noisy image  $I_{awgnoisy}$ . Apply NLM, BM3D and VTV-based denoising methods on  $I_{awgnoisy}$  to obtain the denoised images  $I_{awgNLM}$ ,  $I_{awgBM3D}$ , and  $I_{awgVTV}$ .



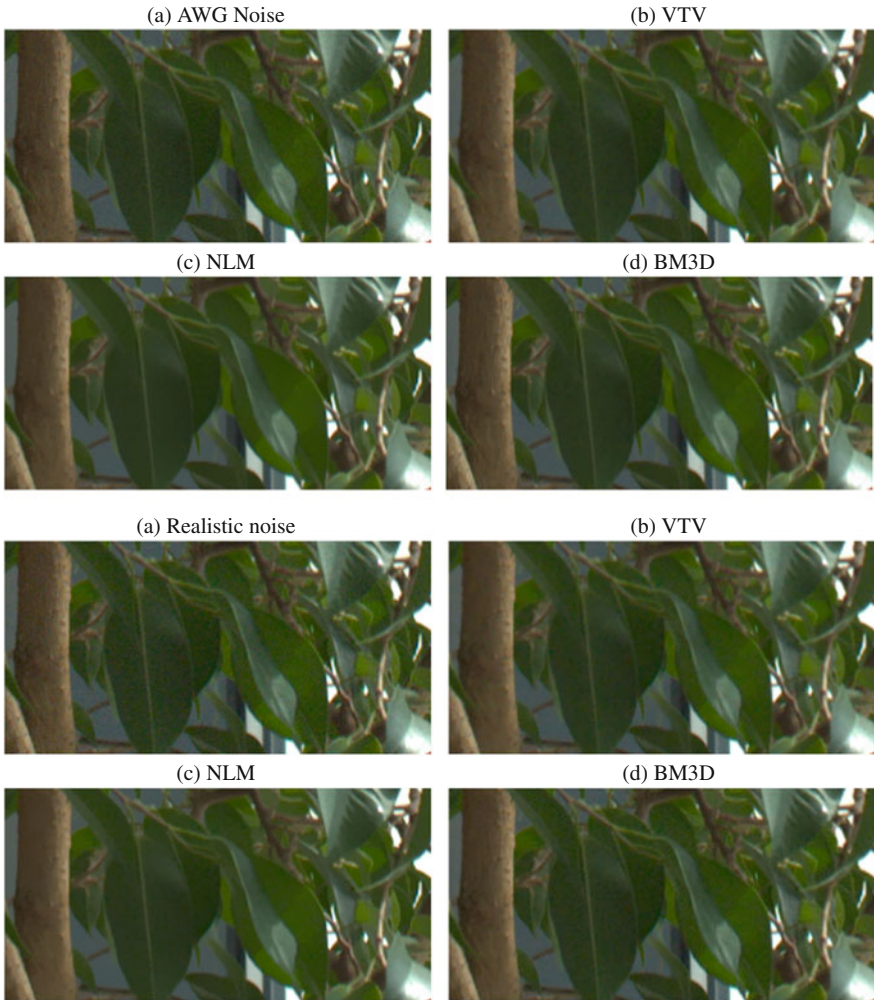
**Fig. 11.9** Comparison of BM3D, NLM, and VTV applied to the camera output, under two noise models. Average PSNR value plots of denoising applied to noisy images created with additive white Gaussian noise (left) and our realistic noise model (right)

The reference clean image  $I_{clean}$  serves as a ground truth for both experiments. We tested the denoising methods on our test set, and the PSNR results are shown in Fig. 11.9. The left shows a plot of PSNR as a function of the average noise standard deviation computed in the 8-bit depth noisy images with AWG noise over the database. The right shows PSNR as a function of ISO sensitivity for images degraded using our realistic noise model. Comparable noise levels are used for both experiments, as seen from the average standard deviation values shown in Table 11.4.

Notice how the ranking of the denoising methods is different with realistic noise than with AWG noise. This justifies the use of a realistic noise model for image denoising. There is also a large drop in the PSNR value for each denoising method from denoising the AWG noise images to realistic noise images, consistent with what was reported by Seybold et al. [44]. The local denoising method applied to the camera output gives worse results in terms of PSNR than the nonlocal patch-based methods, for both noise models. Figure 11.10 illustrates this behavior. Denoising an AWG noise image with BM3D gives an excellent output, while for the same original image but with realistic noise, BM3D produces blocking artifacts on the leaf in the shadow.

### 11.3.3 Comparing Local Denoising on DRAW Versus Nonlocal Denoising on Camera Output

In this section, we illustrate the power of processing the RAW data as opposed to the camera output by showing how a local denoising method, applied to the demosaicked RAW (DRAW) image, can outperform nonlocal methods applied to the camera output.



**Fig. 11.10** Comparison of VTV, NLM, and BM3D denoising methods under AWG on camera output and a realistic noise model. Rows 1–2. **a** Crop from AWG noise image “image9” with  $\sigma = 4.96$ . **b** VTV, PSNR=37.66. **c** NLM, PSNR=38.37. **d** BM3D, PSNR=38.92. Rows 3–4. **a** Crop from realistic noise image “image9” with  $\sigma = 5.67$  and ISO 800. **b** VTV, PSNR=35.39. **c** NLM, PSNR=35.89. **d** BM3D, PSNR=35.72

### 11.3.3.1 Adapting a TV-based Denoising Method to the Signal-Dependent Noise Model

TV-based denoising methods were proposed in the context of images corrupted by signal-independent AWG noise. However, our database images suggest realistic signal-dependent noise is a more accurate model.

Two reasonable approaches are discussed in [33] for removing signal-dependent noise. The first approach is to *adapt an existing denoising method* to treat specific noise model properties. For example, Luisier et al. propose a methodology to adapt transform-domain thresholding algorithms for the mixed Poisson–Gaussian noise model [31]. The second approach is to *create a variance stabilizing transformation (VST) for the particular noise model*. Applying the VST to an image removes signal-dependency and the noise variance becomes constant over the entire image. Then one can use a denoising algorithm created to eliminate Gaussian noise with constant variance. After denoising, one needs to apply the inverse VST. The advantage of the second technique is that denoising images corrupted by AWG noise is an extremely popular topic that has produced many algorithms over the last decades.

Donoho [15] was the first to propose applying the Anscombe transform [4] as a VST. As described above, a denoising algorithm created to eliminate AWG noise with constant variance can then be applied, followed by the inverse VST. Mäkitalo and Foi [32] also used the Anscombe transform to remove the signal-dependency, but emphasized the importance of applying a suitable inverse. Following this approach, we apply the Anscombe transform  $f_{Anscombe}$  to the demosaicked noisy RAW image  $I_{noisyDRAW}$ :

$$f_{Anscombe}(I_{noisyDRAW}) = 2\sqrt{I_{noisyDRAW} + \frac{3}{8}} \tag{11.16}$$

and denoise the image  $f_{Anscombe}(I_{noisyDRAW})$  with VTV, instead of denoising  $I_{noisyDRAW}$ ; this intermediate result is denoted  $D$ . Then, we apply the closed-form approximation of the exact unbiased inverse Anscombe transform proposed by Mäkitalo and Foi [32] to  $D$ :

$$f_{Anscombe}^{-1}(D) = \frac{1}{4}D^2 + \frac{1}{4}\sqrt{\frac{3}{2}}D^{-1} - \frac{11}{8}D^{-2} + \frac{5}{8}\sqrt{\frac{3}{2}}D^{-3} - \frac{1}{8}.$$

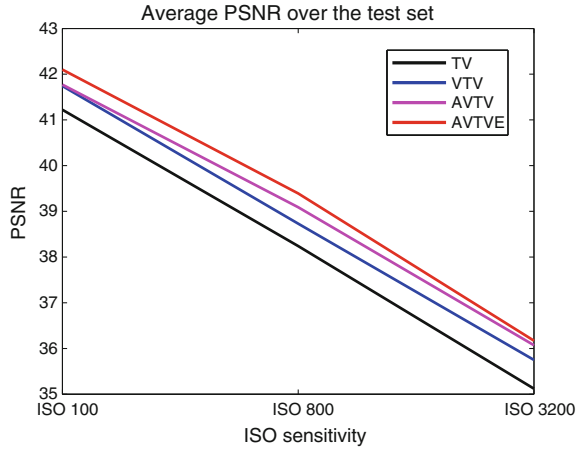
We denote the resulting image by  $I_{denDRAW}$  and refer to this method combining VTV with the Anscombe transform as AVTV. Figure 11.11 illustrates an improvement in the average PSNR value of image results obtained by applying the Anscombe transform before denoising with the VTV-based procedure described by (11.15).

### 11.3.3.2 Refine the Denoising Output by Recovering Lost Details

Even the best denoising algorithms can benefit from the so-called “boosting” techniques [42]. One boosting mechanism involves adding content from the residual (difference between the noisy and denoised image) back to the denoised image. This is justified by the fact that denoising is an imperfect process that eliminates not only noise but small details as well. But while signal leftovers can be retained in the residual, the opposite is also true: noise is retained in the denoised image. An alternate



**Fig. 11.11** Evolution of our TV-based local denoising experiments, under the proposed realistic noise model: average PSNR values computed over our image test set



boosting mechanism aims to eliminate the noise retained in the denoised image, but this produces oversmoothed images as a result.

The biggest challenge for any denoising method, especially for a local one, is to make the distinction between noise and details. As a boosting technique, we propose adding back selected useful information to the denoised image, determined by

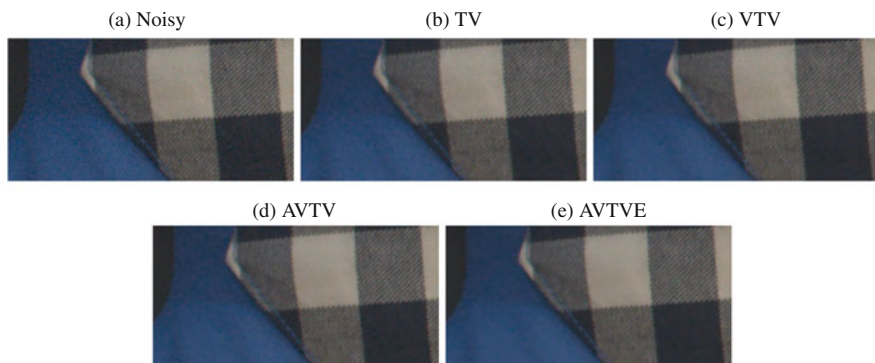
$$I_{AVTVE} = (1 - a)I_{noisyDRAW} + aI_{denDRAW}, \tag{11.17}$$

where the weight function  $a$  should ideally include information related to the local image content. The essential part here is the criteria for differentiating between noise and details we want to recover from the residual.

The function  $a$  we consider is an indicator of the local information in the luminance channel of the denoised image. In smooth areas we want to keep the pixel intensity values of the denoised image intact, so the value of  $a$  should be large; on the other hand, along image contours we want to partially recover some of the details of the original noisy image, so the value of  $a$  should be small (positive and close to 0). Therefore, we propose estimating  $a$  using a local edge indicator, like the Charbonnier diffusivity function [13]

$$f_{Charbonnier} = \frac{1}{\sqrt{1 + \frac{\|\nabla L(I_{denDRAW})\|^2}{\lambda}}}, \tag{11.18}$$

where  $L$  denotes the luminance component, and  $\lambda > 0$  is a contrast parameter related to edge localization. The image  $I_{denDRAW}$  is obtained by finding the number of iterations in the iterative scheme introduced in (11.15) that maximizes the PSNR index computed after color correction, gamma correction, and the quantization step. We give a higher weight to the denoised image than to the noisy one, by choosing:



**Fig. 11.12** Comparison of our local TV-based denoising methods applied to the demosaicked RAW, under the proposed realistic noise model. **a** Crop from noisy image “image20” with  $\sigma = 4.84$  and ISO 800. **b** TV, PSNR=37.14. **c** VTV, PSNR=37.38. **d** AVTV, PSNR=37.45. **e** AVTVE, PSNR=37.73

$$a = \frac{1 + f_{\text{Charbonnier}}}{2}. \quad (11.19)$$

Experiments show that both the step of applying the Anscombe transform before denoising and its inverse after, and the step of refinement described by (11.17), bring an improvement (both in terms of PSNR and visually) compared to only denoising with the iterative scheme introduced in (11.15), as seen in Figs. 11.11 and 11.12. These experiments also illustrate that the vectorial VTV-based denoising approach improves the channel-wise TV-based denoising strategy that was used in [17].

### 11.3.3.3 The Comparison

The experiment in this section is intended to mimic a realistic scenario. Nonlocal patch-based denoising methods are too complex to be implemented in-camera without essential simplifications. Therefore, we compare our local denoising approach AVTVE applied to the demosaicked RAW noisy image, with two nonlocal patch-based methods (NLM and BM3D) applied at the end of the noisy image processing chain, following these steps:

1. We take a clean RAW image  $I_{\text{cleanRAW}}$  and add Gaussian noise, with variance given by the associated noise variance plot as detailed in Sect. 11.3.1.2, to obtain a noisy image  $I_{\text{noisyRAW}}$ .
2. For  $I_{\text{cleanRAW}}$  and  $I_{\text{noisyRAW}}$ , apply white balance, demosaicking, color correction, gamma correction, and quantize to 8 bit to obtain the camera outputs  $I_{\text{clean}}$  and  $I_{\text{noisy}}$ .

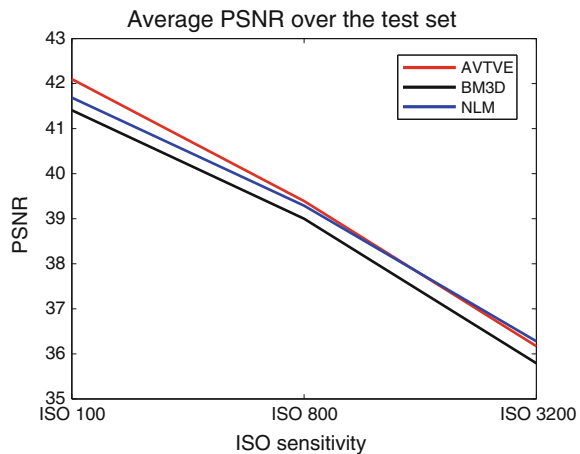


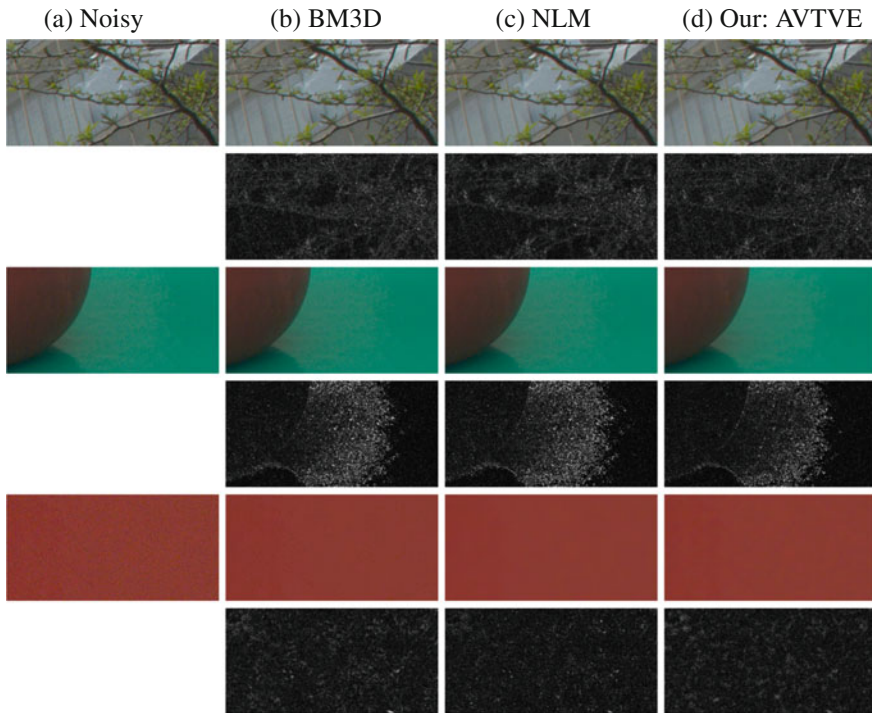
3. Apply nonlocal patch-based denoising methods (NLM and BM3D) to  $I_{noisy}$  to obtain the denoised image ( $I_{NLM}$  and  $I_{BM3D}$ ), optimizing the denoising parameters so as to maximize the PSNR. The reference clean image  $I_{clean}$  serves as a ground truth.
4. Apply white balance and demosaicking to  $I_{noisyRAW}$  to obtain  $I_{noisyDRAW}$ . Then denoise with our local method AVTVE, followed by color correction, gamma correction, and quantization to 8 bit, to output our denoised image  $I_{AVTVE}$ . The denoising parameters, described in the following, are optimized so as to maximize the PSNR of  $I_{AVTVE}$ .
5. Evaluate the images  $I_{BM3D}$ ,  $I_{NLM}$ , and  $I_{AVTVE}$ , visually and with respect to PSNR.

The plot of Fig. 11.13 shows the average PSNR values over our proposed image dataset, for each noise level given by the considered ISO sensitivity and each denoising strategy aforementioned. We can see that our denoising method AVTVE produces better results in terms of PSNR than the BM3D and NLM denoising methods, for almost all ISO levels. However, for the highest noise level associated to ISO 3200, NLM is the best in terms of PSNR, while our method is second.

A visual comparison is illustrated in Fig. 11.14, where the images are denoised with the optimal parameters described above. For a better comparison, the difference between the clean and denoised images for each method is included in Rows 2, 4, and 6. Ideally, the difference image should be completely black, as the difference is given by lost details or artifacts introduced by the denoising method. The close-up images in the first two rows demonstrate that for small noise levels, the denoising methods BM3D, NLM and AVTVE produce comparable results. All three algorithms preserve the apple in the example in the third and fourth rows, although the AVTVE result has a cleaner appearance while the NLM and BM3D denoised images exhibit small blocking artifacts. An image with the highest noise level was investigated in the bottom two rows, where the BM3D output reveals strong blocking artifacts in the homogeneous area, while the AVTVE and NLM results have a cleaner appearance.

**Fig. 11.13** Comparison between the local denoising method AVTVE applied to the demosaicked RAW to the NLM and BM3D denoising algorithms applied to the camera output, under the proposed realistic noise model. Average PSNR values computed over our image test set





**Fig. 11.14** Comparison of the local denoising method AVTVE applied to the demosaicked RAW to the BM3D and NLM denoising methods applied to the camera output, under the proposed realistic noise model. Row 1: crop from noisy image “image13” with  $\sigma = 4.11$  and ISO 800, BM3D result with PSNR=36.79, NLM result with PSNR=36.97, AVTVE result with PSNR=37.11. Row 2: difference images for crops of Row 1, scaled for visualization with the scaling factor 7. Row 3: crop from noisy image “image1” with  $\sigma = 4.55$  and ISO 100, BM3D result with PSNR=36.35, NLM result with PSNR=35.94, AVTVE result with PSNR=37.95. Row 4: difference images for crops of Row 3, scaled for visualization with the scaling factor 7. Row 5: crop from noisy image “image7” with  $\sigma = 9.14$  and ISO 3200, BM3D result with PSNR=38.93, NLM result with PSNR=40.11, AVTVE result with PSNR=39.70. Row 6: difference images for crops of Row 5, scaled for visualization with the scaling factor 7

Table 11.5 shows the average running time for the AVTVE, NLM, and BM3D methods for a  $1000 \times 2000$  color image from our test set, on a i7-4770 CPU with 3.4GHz and 8 cores. At a fraction of the running time of NLM and BM3D, the AVTVE method, although not optimized for speed, produces results that are comparable or better both visually and in terms of PSNR.

**Table 11.5** Average running time (s) for one  $1000 \times 2000$  color test image of *AVTVE*, *BM3D*, and *NLM*, for different noise levels given by different ISO settings

ISO sensitivity	100	400	800	1600	3200
AVTVE	1.5	2.0	2.5	3.5	5.0
BM3D	31	31	31	31	31
NLM	19	19	19	19	19

### 11.3.3.4 Compare Local with Low-Complexity Nonlocal Denoising Applied at the Same Stage of the Image Processing Chain

As nonlocal methods have a high complexity and cannot be implemented in-camera unless some simplifications are done, we consider a reduced-complexity version of the BM3D method, obtained by tuning several parameters such that the method reaches the lowest complexity while producing a reliable image output. We fix the patch size to  $8 \times 8$  and search for similar patches in a small window of size  $10 \times 10$ . As we did for the local method, we optimize the default parameter for each image and noise level of the test set, choosing the one that maximizes the PSNR value of the denoised result.

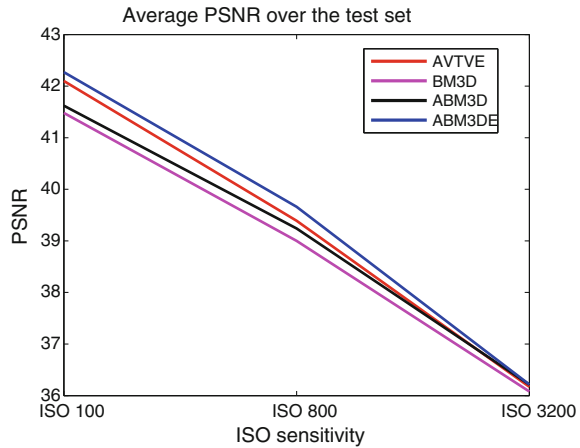
As the BM3D method is designed to treat Gaussian noise, we apply the Anscombe transform introduced in (11.16) before denoising and its inverse after, like we did for our denoising method. We denote this approach by ABM3D. We also consider the refinement step described by (11.17), producing a result denoted ABM3DE. Both the steps of applying the Anscombe transform and the refinement step produce image results that have a higher PSNR value compared to the BM3D output, as seen in the plot of Fig. 11.15. Table 11.6 shows the average running time for the AVTVE, ABM3D and ABM3DE methods for one  $1000 \times 2000$  color image from our test set, on the same machine. For computing the running time of the BM3D algorithm, we use the fast C++ implementation available online [27], while we point out again that our current implementation of the AVTVE algorithm is not optimized for speed.

The plot in Fig. 11.15 shows the average PSNR value over our proposed image dataset, for each noise level given by the considered ISO sensitivity and each denoising strategy aforementioned. For almost all noise levels, our method produces images that are better than BM3D and ABM3D in terms of PSNR. For a higher running time, ABM3DE gives the best PSNR value. Notice that for the highest noise level, all methods produce denoised images with a very similar PSNR value.

### 11.3.4 Research Avenue to Explore

It is clear that many powerful algorithms exist for removing AWG noise from images, and they can be used also to handle RAW pictures after they have been processed with a VST like the Anscombe transform. However, denoising is still a challenging

**Fig. 11.15** Comparison between the local denoising method AVTVE and the low-complexity BM3D, ABM3D, and ABM3DE applied on the demosaicked RAW, under the proposed realistic noise model. Average PSNR values computed over our image test set



**Table 11.6** Average running time (s) for one  $1000 \times 2000$  color test image of AVTVE, ABM3D, and ABM3DE, for different noise levels given by different ISO settings

ISO sensitivity	100	400	800	1600	3200
AVTVE	1.5	2.0	2.5	3.5	5.0
ABM3D	5.2	5.2	5.2	5.2	5.2
ABM3DE	10.4	10.4	10.4	10.4	10.4

task for regular camera output images, where the noise model is extremely complex. There is a need to develop a noise model for JPEG images from which a VST for JPEG noise can be derived, allowing then regular denoising methods that assume AWG noise to be applied to the JPEG case as well. An alternative would be to develop a noise model for JPEG images whose parameters can be estimated from the image itself, and develop new denoising methods adapted to this model.

### 11.4 Optimize Denoising Methods According to Perceived Quality of Results

In film photography, noise is called “film grain” as it is due to the presence of minuscule grains of silver. When subtle, people actually prefer its presence [3] as it improves image appearance. This is a phenomenon due to visual perception: a small amount of noise makes the image look sharper and appear to have higher resolution.

However, too much noise, or noise that is not uniform but highly localized, will make the image unpleasant. This is the case of digital image noise, which is not uniform but image-dependent, appearing more pronounced in dark areas and shadows.

The noise level introduced by common consumer cameras is surprisingly small compared to the level of AWG noise added to clean images in academic works

in order to create synthetic noisy images, as is traditionally used in testing image denoising algorithms. This fact can be concluded from Table 11.4 that includes the average standard deviation computed over realistic noise images, for the ISO levels given by our camera, as in our experiments described in Sect. 11.3.1.2. It can be seen that even the highest noise level corresponding to ISO 3200 gives a small standard deviation, on average. This fact is confirmed in [40], where the authors claim that using noise standard deviations of at least  $\sigma = 10$  for synthetic AWG noisy images is “mostly a historical artefact”.

Photographers often add a small amount of noise to studio images taken at low ISO, due to the fact that a photo that is “too perfect” can be perceived as fake [3]. To emulate grain noise, several layers of Gaussian noise with different variance are added. Photographers always add the noise after the sharpening step and the added noise is achromatic. While the luminance noise in a digital sensor has some sort of similarity to grain, the chroma noise (given by variations in colors) is not something we like to see in a photograph.

In [23] researchers worked with a professional photographer to learn specific ways in which a digital image can be aesthetically improved by adding noise: masking actual noise and banding artifacts in the original, improving the appearance of blown highlights, increasing the perceived resolution. The photographer introduced several different noise layers for each image example resulting in a higher noise level in midtones and a lower one in shadows and highlights, with almost no noise toward 0 and 255. Regarding the noise distribution, while for midtones a Gaussian distribution is a good choice, for highlights the histograms are skewed and the best candidate is the chi-squared distribution characterized by an asymmetrical shape.

Johnson and Fairchild [21] examined some of the ingredients that influence image sharpness perception. The highest image score was achieved by the highest resolution image to which noise with  $\sigma = 10$  was added, and processed with both contrast enhancement and increased sharpness. Regarding the noise factor, the conclusion was that additive uniform noise applied independently in each color channel increases the perceived sharpness only up to a point, after which it decreases. Interestingly, adding noise can also mask a reduction in image resolution; images with 300 ppi and 150 ppi were evaluated as having similar perceived sharpness when the lower resolution photo had added noise and increased contrast. Kurihara et al. [24] concluded that when noise is added to edges, sharpness decreases, while when added to texture, it increases up to a point, decreasing afterward. Kayargadde and Martens [22] investigated the connection between the noise level and blur, finding that adding noise to a sharp image causes it to be perceived as more blurred, while adding noise to a blurred one makes it appear a little less blurred.

None of these aspects of perceived image quality are considered when evaluating denoising results using existing image quality metrics. The PSNR measure is known to have problems in indicating the perceived image quality, and although SSIM [47] is designed to take into account perceived errors, it is not well correlated to human preference [39]. Even though their limitations are known, PSNR and SSIM are still the most popular measures for image denoising. There are many more image quality metrics, as, for example, the visual information fidelity (VIF) measure [45],

the Blind/Referenceless Image Spatial Quality Evaluator (BRISQUE) [35], or the naturalness image quality evaluator (NIQE) [36]. However, the vast majority of metrics are based on measuring differences between the denoised result and the clean ground truth, which does not necessarily correlate with the perceived image quality of the denoised result, as we will demonstrate in the next subsections. This section is extending the work described in [18].

### 11.4.1 Experiments

We compare three denoising methods: we introduce a local curvature smoothing (CS) algorithm and compare it with the nonlocal patch-based methods NLM [11] and BM3D [14].

For the CS algorithm, we start with the original noisy image  $I^0 = I_0$ , compute channel-wise its regularized level line curvature  $\kappa_{\varepsilon_2}(I_0)$  (which is just the usual level line curvature computed with an added value  $\varepsilon_2$  in the denominator: as this value increases, the curvature becomes smoother), and iterate the following equation  $N$  times:

$$I^{n+1} = I^n + \Delta t \left[ \nabla^- \cdot \left( \frac{\nabla^+ I^n}{\sqrt{\|\nabla^+ I^n\|^2 + \varepsilon_1}} \right) - \kappa_{\varepsilon_2}(I_0) \right], \quad (11.20)$$

where  $\nabla^+$  and  $\nabla^-$  are the forward and backward spatial difference operator. Due to the fact that the curvature can be estimated for each pixel with a  $3 \times 3$  stencil around it, the proposed method is local. We fix the parameters:  $\varepsilon_1 = 10^{-6}$  (very small for a good approximation of  $\kappa(I)$ ),  $\Delta t = 0.002$  and  $N = 30$ . The CS method has only one parameter, the regularizing value  $\varepsilon_2$ , and how it is chosen will be described later.

We perform this comparison on two image databases: images from the Kodak database [2] with AWG noise and photographs taken by us with the noise model proposed in Sect. 11.3.1.2. The evaluation is done using subjective testing as well as the objective PSNR and SSIM metrics.

#### 11.4.1.1 AWG Noise Case

The subjective evaluation involved 17 participants (all with normal or corrected to normal vision). Subjects sat in a well-lit office environment at approximately 64 cm from the display and were presented with four versions of an image: the original at the top, and the three denoising results (CS, NLM, and BM3D) in some random order at the bottom. The observers were asked to look at the original image and then indicate which of the three provided denoised images they preferred (Figs. 11.16 and 11.17).

We picked randomly three images from the Kodak database (“kodim1”, “kodim3”, and “kodim13”) and created three noise levels by adding Gaussian noise with



Fig. 11.16 Test images: crops from Kodak images

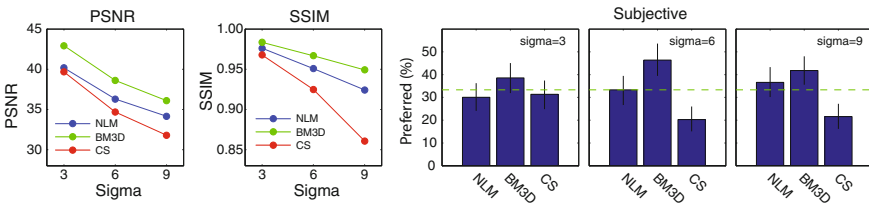


Fig. 11.17 From left to right: average PSNR and SSIM computed for three images from the Kodak database, and results of psychophysical experiment for comparing our proposed local denoising method to BM3D and NLM

Table 11.7 Optimized parameter value  $\epsilon_2$  as a function of  $\sigma$

$\sigma$	3	6	9
$\epsilon_2$	0.00032	0.003	0.00608

$\sigma = 3, 6, 9$ . As commented above, although these  $\sigma$  values seem low, they are common noise levels in photography. The denoising algorithms NLM [12] and BM3D [27] take as input the value of  $\sigma$ . We find the value of  $\epsilon_2$  of our method with a subjective methodology; we ask participants to adjust  $\epsilon_2$  via key presses for finding their preferred image result. We average over subjects and images to get one value of  $\epsilon_2$ , as shown in Table 11.7, for each noise level.

We compute the values of PSNR and SSIM for each denoising method, as we have the clean ground truth. We also conduct a user preference test (using the procedure described above). We crop the images to be able to simultaneously show all of them at the native resolution and avoid resizing. The results in Fig. 11.17 indicate that the SSIM metric gives a reasonable approximation of the subjective scores, predicting that the differences between the algorithms are small for a low noise level and that for high noise levels, the local CS method gives a poor performance. However, both PSNR and SSIM are poor at predicting the preferred algorithm on an image-by-image basis, as Fig. 11.18 shows. To evaluate the metric performance we compute an upper





Fig. 11.18 Visual comparison for one test crop from image “kodim3” and user preferences

bound by randomly dividing the subjective data into two subject groups (A and B). We then compute a percentage correct score, for each image. The score is 100% if the order is entirely correct, 33% for only getting the order of one correct or 0% for a complete failure. The result is that, on average, group A is able to predict the data from group B 64% of the time. However, both the SSIM and PSNR achieve a score of less than 46%, with a baseline score of 33%.

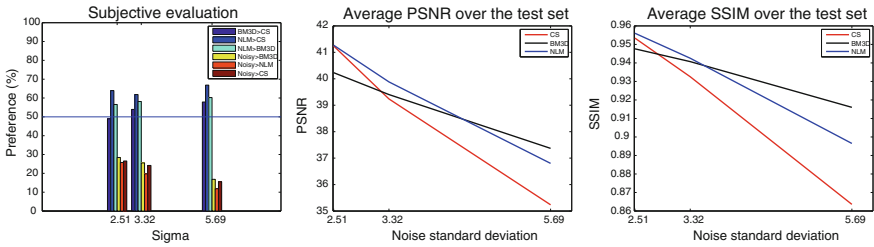
### 11.4.1.2 Realistic Noise Case

We conduct a user preference test on realistic noise removal, with results given by denoising with CS, NLM and BM3D, as well as no denoising at all. The subjective evaluation involved 19 participants (all with normal or corrected to normal vision) that sat in a well-lit office environment at approximately 64 cm from the display. We used the 20 images and three noise levels given by ISO 100, 400 and 1600 from our proposed test set following the realistic noise model proposed in Sect. 11.3.1.2, cropped to allow a simultaneous display of two images at their native resolution. At the first stage, we find the values for  $\sigma$  (the parameter for NLM and BM3D) and  $\varepsilon_2$  (parameter for CS) through user tests. Subjects are presented with a sequence of 51 versions of the same image denoised with different values of the pertinent parameter, from a minimum (no denoising) to a maximum (the image is fully denoised but blurry), and the observer chooses the one he/she prefers. At the second stage, subjects are asked to select their preferred image between two versions of it displayed simultaneously, which can be either the original noisy or the result of the preferred output of the denoising methods NLM, BM3D and CS obtained at the previous stage. The results of the psychophysical experiment are shown in Fig. 11.19.

On the left-hand side of the figure, we plot the user preference averaged across all images, for each noise level considered. People preferred NLM for all noise levels. They slightly preferred CS over BM3D, for a small noise level, while the results were comparable for a medium noise level. As the noise increased, for ISO 1600, people preferred NLM even more compared to the BM3D and CS algorithms.

Regarding the comparison between the original noisy and the denoised images, observers voted for applying a denoising method over not applying any method. However, the original noisy image was preferred surprisingly often (around 20%), especially when compared to the BM3D output. Averaging over all subjects, there is no image for which no denoising was preferred over all denoising methods. Therefore, choosing the noisy image penalized the denoised image, it was not a vote of





**Fig. 11.19** A two by two comparison between the three denoising algorithms and the original noisy image

appreciation for the noisy image. There is only one image (Image 18, cartoon-like) for which the noisy original did not get any vote from any observer, for all noise levels considered. This might indicate that we do not like noise in cartoon-like images.

At the first stage, observers could choose between the noisy original and 50 denoised images with increasing parameter values: only one observer chose the noisy image in several cases. At the second stage, of comparing two by two each of the denoising methods and no denoising, all subjects gave at least one vote to one original noisy image. Therefore, when the noisy stimulus was shown next to the denoised one, the preference for the noisy image increased.

For each denoising method, we computed the average of PSNR and SSIM for all images that people chose as their preferred one. We compared these objective metrics result to the subjective one, included in Fig. 11.19. For a small noise level, in the case of the BM3D denoising method, people preferred images denoised with parameters giving a surprisingly low PSNR value. The PSNR value also estimated a large difference in quality between the outputs of CS and BM3D, while people perceived it as small. While the PSNR and SSIM index ranked the NLM and CS methods similarly, people preferred the former one. In the case of the second noise level considered, the objective measures were better correlated to human preference. However, the SSIM index estimated that the image quality of the BM3D output is much higher than that of the CS algorithm, while subjects only perceived a small difference between the two methods. In the case of the highest noise level considered, the objective measures and the subjective preference ranked differently the denoising methods. While the BM3D method gave images with the highest PSNR and SSIM values, people chose NLM results as their favorite. Both the PSNR and SSIM metrics estimated a large difference in quality between the outputs of BM3D and CS, while people perceived it as small.

### 11.4.2 Research Avenue to Explore

We have seen examples that once more show how the use of popular metrics like PSNR or SSIM is problematic in the evaluation of different denoising outputs, since they are not well correlated with personal preference. Therefore, there is a need

to create an image metric for image denoising that is based on human perception, according to which noise removal algorithms can be optimized. Alternatively, until such a metric is developed, it would improve results if denoising methods are tuned and ranked via user tests based on perceived appearance.

## 11.5 Conclusion

In this chapter, we have pointed out several avenues to pursue in order to improve denoising results that do not entail developing new denoising algorithms. First, we described how it can be better to denoise a transform of the noisy image rather than denoise the noisy image directly. We mention several possible transforms, and an open problem is to find one that is optimal for denoising, according to a proper image quality metric. Next, we pointed out the importance of having a proper noise model for JPEG pictures, so that a VST can be developed that transforms noise in JPEG images into AWG noise, enabling existing denoising methods to be properly applied to the JPEG case. Finally, we highlighted the fact that while virtually all denoising methods are optimized and validated in terms of the PSNR or SSIM measures, these metrics are not well correlated with perceived image quality and therefore it could be better to optimize the parameter values of denoising methods according to subjective testing. A remaining challenge is to develop perceptually based image quality metrics that match observer preference.

**Acknowledgements** This work has received funding from NSF-DMS # 1320829, from the European Research Council (ERC) under Starting Grant ref. 306337, from the European Union's Horizon 2020 research and innovation program under grant agreement number 761544 (project HDR4EU) and under grant agreement number 780470 (project SAUCE), by the Spanish government and FEDER Fund, grant ref. TIN2015-71537-P (MINECO/FEDER,UE), and by the Icrea Academia Award.

## References

1. <http://ip4ec.upf.edu/realisticnoise>
2. <http://r0k.us/graphics/kodak/>
3. <https://photography.tutsplus.com>
4. Anscombe FJ (1948) The transformation of Poisson, binomial and negative-binomial data. *Biometrika* 35(3):246–254
5. Batard T, Berthier M, Spinor Fourier transform for image processing. *IEEE J Sel Topics Signal Process* 7(4):605–613 (2013)
6. Batard T, Bertalmío M (2013) Generalized gradient on vector bundle-application to image denoising. In: *Lecture Notes Computer Science*, vol 7893, pp 12–23
7. Batard T, Bertalmío M (2014) On covariant derivatives and their applications to image regularization. *SIAM J Imag Sci* 7(4):2393–2422
8. Bertalmío M, Levine S (2014) Denoising an image by denoising its curvature image. *SIAM J Imag Sci* 7(1):187–201

9. Blomgren P, Chan TF (1998) Color TV: total variation methods for restoration of vector-valued images. *IEEE Trans Image Process* 7(3):304–309
10. Bresson X, Chan TF (2008) Fast dual minimization of the vectorial total variation norm and applications to color image processing. *Inverse Probl Imag* 2(4):455–484
11. Buades A, Coll B, Morel J-M (2005) A non-local algorithm for image denoising. In: *Proceedings IEEE international conference on computer vision and pattern recognition*, vol 2, pp 60–65
12. Buades A, Coll B, Morel J-M (2011) Non-local means denoising. *Image Process On Line* 1
13. Charbonnier P, Blanc-Féraud L, Aubert G, Barlaud M (1994) Two deterministic half-quadratic regularization algorithms for computed imaging. In: *Proceedings of IEEE international conference on image processing*, vol 2. IEEE Computer Society Press, Los Alamitos, pp 168–172
14. Dabov K, Foi A, Katkovnik V, Egiazarian K (2007) Image denoising by sparse 3D transform-domain collaborative filtering. *IEEE Trans Image Process* 16(8):2080–2095
15. Donoho DL (1993) Nonlinear wavelet methods for recovery of signals, densities, and spectra from indirect and noisy data. In: *Proceedings of symposia in applied mathematics*, pp 173–205
16. Ghimpeteanu G, Batard T, Bertalmío M, Levine S (2016) A decomposition framework for image denoising algorithms. *IEEE Trans Image Process* 25(1):388–399
17. Ghimpeteanu G, Batard T, Seybold T, Bertalmío M (2016) Local denoising applied to RAW images may outperform non-local patch-based methods applied to the camera output. In: *IS&T electronic imaging conference*
18. Ghimpeteanu G, Kane D, Batard T, Levine S, Bertalmío M (2016) Local denoising based on curvature smoothing can visually outperform non-local methods on photographs with actual noise. In: *Proceedings of the IEEE international conference on image processing ICIP*
19. Healey GE, Kondepudy R (1994) Radiometric CCD camera calibration and noise estimation. *IEEE Trans PAMI* 16(3):267–276
20. Hahn J, Tai X-C, Borok S, Bruckstein AM (2011) Orientation-matching minimization for image denoising and inpainting. *Int J Comput Vis* 92(3):308–324
21. Johnson GM, Fairchild MD (2000) Sharpness rules. In: *Proceedings IS&T/SID eighth color imaging conference*, pp 24–30
22. Kayargadde V, Martens JB (1996) Perceptual characterization of images degraded by blur and noise: experiments. *J Opt Soc Am* 13(6):1166–1177
23. Kurihara T, Manabe Y, Aoki N, Kobayashi H (2008) Digital image improvement by adding noise: an example by a professional photographer. *Image Qual Syst Perf V* 6808:1–10
24. Kurihara T, Aoki N, Kobayashi H (2009) Analysis of sharpness increase by image noise. *Human Vis Electron Imag XIV* 7240:1–9
25. Lebrun M, Colom M, Morel JM (2014) The noise clinic: a universal blind denoising algorithm. In: *IEEE international conference on image processing*, pp 2674–2678
26. Lebrun M, Buades A, Morel JM (2013) A nonlocal Bayesian image denoising algorithm. *SIAM J Imag Sci* 6(31):1665–1688
27. Lebrun M (2012) An analysis and implementation of the BM3D image denoising method. *Image Process On Line* 2:175–213
28. Levin A, Nadler B (2011) Natural image denoising: optimality and inherent bounds. In: *Proceedings IEEE international conference on computer vision and pattern recognition* 2:2833–2840
29. Levin A, Nadler B, Durand F, Freeman WT (2012) Patch complexity, finite pixel correlations and optimal denoising. MIT-Computer Science and Artificial Intelligence Laboratory
30. Lysaker M, Osher S, Tai XC (2004) Noise removal using smoothed normals and surface fitting. *IEEE Trans Image Process* 13(10):1345–1357
31. Luisier F, Blu T, Unser M (2011) Image denoising in mixed Poisson-Gaussian noise. *IEEE Trans Image Process* 20(3):696–708
32. Mäkitalo M, Foi A (2011) A closed-form approximation of the exact unbiased inverse of the Anscombe variance-stabilizing transformation. *IEEE Trans Image Process* 20(9):2697–2698
33. Mäkitalo M, Foi A (2013) Optimal inversion of the generalized Anscombe transformation for Poisson-Gaussian noise. *IEEE Trans Image Process* 22(1):91–103

34. Malvar HS, He L, Cutler R (2004) High-quality linear interpolation for demosaicing of Bayer-patterned color images. In: IEEE international conference on acoustics, speech and signal processing, pp 485–488
35. Mittal A, Moorthy AK, Bovik AC (2011) No-reference image quality assessment in the spatial domain. *IEEE Trans Image Process* 21:4695–4708
36. Mittal A, Soundararajan R, Bovik AC (2013) Making a completely blind image quality analyzer. *IEEE Signal Process Lett* 22(3):209–212
37. Nam S, Hwang Y, Matsushita Y, Kim SJ (2016) A holistic approach to cross-channel image noise modeling and its application to image denoising. In: IEEE conference on computer vision and pattern recognition (CVPR), pp 1683–1991
38. Osher S, Burger M, Goldfarb D, Xu J, Yin W (2005) An iterative regularization method for total variation-based image restoration. *Multiscale Model Simul* 4(2):460–489
39. Pambrun JF, Noumeir R (2015) Limitations of the SSIM quality metric in the context of diagnostic imaging. In: Proceedings of the IEEE international conference on image processing ICIP
40. Plötz T, Roth S (2017) Benchmarking denoising algorithms with real photographs. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)
41. Rahman T, Tai X-C, Osher S (2007) A tv-stokes denoising algorithm. In: Lecture notes on computer science, vol 4485, pp 473–483
42. Romano Y, Elad M (2015) Boosting of image denoising techniques. *SIAM J Imag Sci* 8(2):1187–1219
43. Rudin LI, Osher S, Fatemi E (1992) Nonlinear total variation based noise removal algorithms. *Phys D Nonlinear Phen* 60(1–4):259–268
44. Seybold T, Cakmak Ö, Keimel C, Stechele W (2014) Noise characteristics of a single sensor camera in digital color image processing. In: Color and imaging conference, pp 53–58
45. Sheikh HR, Bovik AC (2006) Image information and visual quality. *IEEE Trans Image Process* 15(2):430–444
46. Trussell HJ, Zhang R (2012) The dominance of Poisson noise in color digital cameras. In: IEEE international conference in image processing, pp 329–332
47. Wang Z, Bovik AC (2002) A universal image quality index. *IEEE Signal Process Lett* 9:81–84

# Index

## A

ADAM optimizer, 56  
Additive White Gaussian Noise (AWGN), 1, 4, 38, 94, 235, 236  
Aliasing, 211  
Alternating Directions Method of Multipliers (ADMM), 70, 79, 81–83, 85, 87, 115

## B

Besov-space functions, 38  
Block-Matching 3D (BM3D), 42, 51, 52, 54, 56, 57, 59, 70–74, 77, 96, 102, 104, 107, 109–112, 114, 151, 152, 154, 177, 179, 247, 248, 297, 304, 306, 311–313, 317–321, 323–326

## C

Camera noise, 235, 237, 242–247, 249–251, 261  
Camera processing, 235, 236, 240, 308, 309  
Camera processing pipeline, 236, 237, 308–310  
Censored distribution, 1, 17, 18  
Central-Limit Theorem (CLT), 1  
Clipping, 16–18, 20–22, 25, 239, 241, 242  
Clustering-based Sparse Representation (CSR), 42  
Colored noise, 23, 26, 31, 32, 34, 200  
Column noise, 1, 33  
Compression, 40, 66, 101, 111, 120, 176, 180, 192, 195, 196, 202, 209, 214, 215, 218, 221, 225, 226, 227, 228, 230, 253  
Convolutional neural network, 39, 48, 49, 53, 59, 93, 94, 96–98, 105, 148, 153

Correlated noise, 1, 3, 26–28, 30, 175, 178, 195, 247, 249–251, 261  
Cross-talk, 1, 3, 26

## D

Data-driven, 39, 63–65, 71  
Deep Convolutional Neural Network (DCNN), 54  
Deep learning, 37, 39, 48, 54, 56, 57, 59, 63, 74, 94, 105, 114, 148  
Deep neural network, 59, 66, 73, 74, 97, 153  
Demosaicking, 318  
Denoising, 37–40, 42, 46, 48, 52–57, 59, 60, 63–65, 67, 69–78, 83–87, 93–101, 103–106, 108–116, 119, 120, 125–129, 132, 133, 136–138, 140–142, 144, 145, 147, 151–153, 155, 157, 159, 162–167, 175–180, 182–185, 187, 188, 192–195, 197, 199–202, 235–238, 244, 245, 247, 248, 251–253, 259, 261, 262, 267–269, 272, 278, 279, 282, 283, 285–287, 289, 292, 293, 295–297, 299, 300, 303–308, 311–327  
Dictionary learning, 40–42, 71, 72  
Directional Pyramids, 169  
Discontinuities, 67–69, 183  
Discrete Cosine Transform (DCT), 32, 40

## E

Edge-aware filtering, 268, 272

## F

Fixed-pattern noise, 1, 212  
Flicker noise, 211, 251, 252, 259

**G**

- Gaussian Mixture Models, 129, 134, 143, 144, 146, 148
- Gaussian noise, 1, 3, 15, 40, 51, 56, 73, 74, 78, 79, 83, 84, 99, 126, 127, 157, 188, 209, 210, 214, 224, 225, 226, 227, 245, 247, 272, 282, 284, 285, 315, 317, 322
- Gaussian priors, 125, 128, 130, 132, 133, 139, 142
- Gaussian Scalar Mixture (GSM), 46
- Generative Adversarial Network (GAN), 54, 59, 75, 120
- Gradient descent, 76, 79, 83, 84, 87, 95, 99, 100, 297

**H**

- Heavy noise removal, 37, 48, 59
- Heavy-tail distribution, 38
- Heteroskedastic Gaussian noise, 16
- Heteroskedasticity, 1, 14, 15, 16, 18, 23, 24
- HQ splitting, 79–81, 87
- Human Visual System (HVS), 73, 218–220

**I**

- Image denoising, 37–39, 42, 46, 48, 51, 56, 59, 63, 83, 88, 93, 95–98, 102, 106, 114, 119, 126, 175, 179, 202, 297, 300, 313, 327
- Image quality metrics, 221, 225, 227, 228, 296, 304, 307, 321, 322, 327
- Image restoration, 35, 39, 54, 74, 76, 93, 97, 101, 106, 114, 115, 120, 125, 148
- Image sequence denoising, 175, 179, 195
- Imaging pipeline, 111, 207
- Impulse noise, 212
- Imsgr denoising, 64
- Industry application, 221
- Internal and External image statistics, 156, 167

**K**

- Karhunen–Loeve Transform (KLT), 41
- K-means, 24, 41–43, 141, 142, 144, 145
- K-Singular Value Decomposition (K-SVD), 41–43, 152
- K-SVD Algorithm, 72

**L**

- Learning-based denoising methods, 72
- Lens flare, 210
- Local denoising, 160, 307, 313
- Local sparsity, 39, 40
- Local statistics, 24, 46
- Low-rank approximation, 44, 45

**M**

- MAP estimation, 65
- Mean–variance curve, 23, 24, 34
- Metrics, 217–223, 225, 228, 229, 235, 253, 262, 295, 323, 326
- Model-based, 37, 56, 57, 93, 94, 97, 114, 115
- Motion compensation, 175, 176, 179, 180, 188, 201, 267, 279–282, 287–289, 293
- MSE, 56, 217, 218, 221–223, 225–227, 269
- Multiplicative noise, 15
- Multiscale filtering, 267
- Multi-scale patch recurrence, 152
- Mumford–Shah regularizer, 69

**N**

- Natural image statistics, 66, 70, 221
- Network architecture, 64, 96, 104
- Noise characteristics, 66, 195, 235, 237, 238, 244, 250, 262
- Noise estimation, 14, 22, 25, 175, 178, 192, 196, 202, 309
- Noise measurement, 196, 227, 228, 254
- Noise models, 6, 26, 34, 176, 178, 195, 208, 213, 237, 251, 308, 311, 313
- Noise perception, 253
- Noise removal, 106, 119, 176, 177, 193, 195, 197, 198, 201, 276, 297, 327
- Noise visibility, 252, 253, 257, 258, 260–262
- Nonconvex regularizers, 68
- Non-local image processing, 269
- Non-Local Means (NLM), 153, 297, 304, 311, 312, 318
- Non-local regularization, 37, 39, 43, 77
- Nonlocal sparsity, 39, 59

**O**

- One-parameter family of distributions, 9, 10, 12, 18

**P**

- Patch-based methods, 126, 148, 152, 177, 195, 313
- PatchSNR, 151, 160, 166, 167
- Photo-Response Nonuniformity (PRNU), 34
- Poisson distribution, 2, 11–14, 66, 210
- Poisson–Gaussian noise, 12, 16, 25, 188, 315
- Poisson noise, 11, 119, 198
- Potts model, 69
- Power Spectral Density (PSD), 4, 23, 26, 34
- Primal-dual algorithm, 84, 87
- Principal component analysis, 175
- Prior, 22, 30, 46–48, 59, 66, 70, 93–96, 99, 101, 111, 114–117, 125, 128, 129, 135, 138–140, 147, 184, 212, 213, 218

- Proximal method, 76, 77, 79, 83, 88, 95, 96  
 PSD estimation, 31  
 PSNR, 52, 56, 87, 99, 102, 107, 141, 151, 153, 217, 218, 221–227, 229, 235, 236, 247, 256, 261, 296, 297, 301, 302, 304, 305, 311, 313, 315, 318, 320, 322, 325  
 Pull-push filtering, 276, 277
- R**  
 Raw data, 1, 176, 202, 237, 243, 250, 313  
 Raw image, 2, 5, 6, 26, 178, 242, 307–309, 312, 315  
 Readout noise, 1, 26  
 Real noise, 106, 109, 119, 176, 193, 306, 307, 312  
 Regularization, 42, 63, 64, 66–68, 70, 71, 75–79, 88, 94, 98, 115, 136, 145, 181, 183  
 RMSE, 159, 164, 182, 187, 188  
 Row noise, 1
- S**  
 Salt-and-pepper noise, 78, 79, 83, 84  
 Saturation, 213, 235  
 Shot noise, 2, 111, 126, 210  
 Signal-dependent noise, 34, 237, 247, 251, 308, 314  
 Signal-independent noise, 25  
 Simultaneous Sparse Coding (SSC), 39, 47, 57  
 Sparsity-based denoising, 38  
 SSIM, 51, 52, 57, 217, 221, 223–227, 282, 290, 291, 295, 297, 304, 322, 324, 326, 327
- Standard deviation function, 15, 19, 25, 34  
 Standards, 7, 9, 11, 14, 18, 20, 22, 23, 45, 46, 83, 118, 179, 184, 192, 196, 198, 201, 216, 227–230, 236, 244, 248, 252, 261, 269, 272, 277, 278, 282–284, 310, 313  
 Statistical inference, 138
- T**  
 Thermal noise, 2, 210, 211  
 Tone-mapping, 66, 197, 213, 215, 236, 241, 244  
 Total Generalized Variation (TGV), 67, 68  
 Total Variation (TV), 66–69, 70, 71, 83–85, 87, 177, 297, 311, 314, 316, 317, 297  
 Trainable Nonlinear Reaction–Diffusion (TNRD), 95  
 Transform image, 176–178, 198  
 Transmission, 209, 214–217, 229  
 Truncated quadratic regularizer, 69  
 TV-q model, 70
- V**  
 Variable splitting, 114, 115, 120  
 Variance function, 31  
 Variational method, 297  
 Vectorial Total Variation (VTV), 297  
 Video processing, 202, 207, 258, 261, 262
- W**  
 Wavelet transform, 40, 96, 195