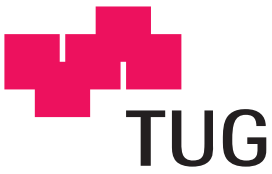


Yunjin Chen

Learning fast and effective image restoration models

DOCTORAL THESIS

to achieve the university degree of
Doktor der technischen Wissenschaften

submitted to

Graz University of Technology

Supervisor

Prof. Dr. Thomas Pock
Institute for Computer Graphics and Vision

Prof. Dr. Stefan Roth
Department of Computer Science, TU Darmstadt

Graz, Austria, November 2014

To my parents Xiaohua and Yilan

There is no royal road to science, and only those who do not dread the fatiguing climb of its steep paths have a chance of gaining its luminous summits.

Karl Marx

Abstract

Up to now, image restoration remains an active research topic, and many new approaches are constantly emerging. However, many newly proposed algorithms achieve the state-of-the-art performance, at the expense of computation time. The goal of this thesis is to develop effective image restoration approaches with both high computational efficiency and recovery quality. To that end, we focus on variational models and some related models derived from them, e.g., nonlinear diffusion processes, due to their effectiveness for many generally ill-posed computer vision problems.

Motivated by statistical inference methods, variational methods are among the most successful methods to solve image restoration problems. Variational methods aim to minimize an energy functional which is designed to appropriately describe a specific image restoration problem. Typically, it involves an image prior term (also known as regularizer) and a data fidelity term, which is derived from the observation model. The performance of a variational image restoration model heavily depends on the regularization term. The development of better image regularization techniques has received intensive attention in the past two decades. In this thesis, we concentrate on the so-called Fields of Experts (FoE) image prior model (a trainable filter-based higher order MRFs model), which was firstly proposed by Roth and Black in 2005. We prefer the FoE image prior model for two main reasons. (1) It has been widely investigated in many classic image restoration problems due to its high effectiveness, which is attributed to the explicit modeling of the heavily-tailed statistical properties of natural images. (2) The resulting variational model has the additional advantage of high computational efficiency, as it is a local model, and only involves 2D convolution operations of a set of linear filters, alluding to the fact that it is well-suited for parallel computation such as GPU.

We review existing algorithms for the training of the FoE model, and propose a refined loss-specific training scheme. The loss-specific training scheme naturally leads to a bi-level optimization problem. We make use of techniques from bi-level optimization to solve it, where we found it important to we solve the lower-level problem in the bi-level framework

with very high accuracy. It turns out that our refined training algorithm helps us to arrive a better learned FoE model, which can significantly boost the performance of previous models. We demonstrate that this seemingly negligible modification is very beneficial for the bi-level learning.

We build the link between the FoE model and a recently proposed analysis operator learning model, which can be seen as the counterpart of the well-known synthesis sparse representation based models such as K-SVD. We hold the opinion that the commonly addressed analysis prior model, which is usually designed based on image patch rather than the whole image, is a simplified version of the FoE model, at least in the context of image processing. We argue that for the analysis prior modeling, we should turn to the image-based modeling framework - FoE. Numerical experiments also demonstrate that the image-based analysis model (e.g., the FoE model) works better than the patch-based ones.

We apply the learned image regularizers (e.g., the FoE models) to a variety of classical image restoration problems, including (1) noise reduction with different noise types such as additive Gaussian noise, Nakagami multiplicative noise and impulse noise, (2) JPEG blocky artifacts suppression, (3) image super resolution from a single image or multiple low resolution frames, (4) image deconvolution for blurring images corrupted by certain linear kernels, and (5) image inpainting. The resulting variational models naturally lead to demanding non-convex optimization problems. We develop a highly efficient non-convex optimization algorithm, called iPiano to solve all the related problems. For all the investigated applications, the resulting variational models with the learned image regularizers can achieve very good recovery performance on par with state-of-the-art algorithms. Moreover, the resulting variational models have an additional advantage of simplicity, and are well-suited for GPU computation. Though the CPU implementation of our model is slower than some highly-engineered methods, such as BM3D, the GPU implementation is at least one order of magnitude faster than those strong competitors.

Motivated by the natural link between the energy functional minimization model and nonlinear diffusion models, we propose to learn optimized nonlinear diffusion models derived from the FoE model regularized energy functional. For this new training model, we obtain additional capacity to train the penalty functions associated with the FoE model, which are fixed to a specific one in the variational framework. Numerical experiments show that due to this additional freedom to tune the penalty functions, we can achieve significantly better results compared to our previous variational models.

We investigate two representative image restoration problems, namely Gaussian de-

noising (a standard test bed to evaluate newly proposed image restoration models) and JPEG deblocking (a non-smooth problem), which is used to demonstrate the applicability of our proposed training model for non-smooth problems. We train two specific nonlinear diffusion processes for these two problems. It turns out that we arrive at surprisingly good results which are on par with or even surpass the recent state-of-the-arts, within 5 diffusion steps. As the resulting diffusion processes only involve few steps (≤ 5 steps), they are extremely fast. Therefore, the CPU implementation based on Matlab is already faster than some highly-engineered methods, such as BM3D. In addition, the GPU implementation clearly accelerates the diffusion procedure. We find that with the GPU implementation, our approach successfully accomplishes the exploited image restoration problems in less than 1s for the images of size up to 3K $\times$ 3K.

Keywords. Fields of Experts, non-convex optimization, bi-level optimization, reaction diffusion, nonlinear diffusion, image denoising, image despeckling, image deblocking, image super resolution, image deconvolution, image inpainting

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

The text document uploaded to TUGRAZonline is identical to the presented doctoral thesis.

Place	Date	Signature

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Dissertation identisch.

Ort	Datum	Unterschrift

Acknowledgments

I would like to thank all those people who contributed to this thesis in various ways and made the last years an unforgettable experience for me.

First and foremost, I would like to express my sincere gratitude and appreciation to my supervisor, Prof. Thomas Pock for his valuable, indefatigable and inspiring support, for his effective guidance on carrying out scientific research in the exciting field of computer vision, for providing an excellent and professionally challenging working environment. This thesis would have not been possible without his tremendous support, patience and unsurpassed knowledge. He has been a great mentor to supervise me whilst allow me the room to work in my own way. If I were a boat drifting in the ocean, he was the navigation system that tells me where I am, and what direction to go. I am extremely thankful to him. Meanwhile, I am also grateful to Prof. Stefan Roth from TU Darmstadt for agreeing to serve as the second thesis supervisor, as well as his seminal work on the Fields of Experts (FoE) model, which is the foundation stone of my thesis.

Now I want to express my special thanks to Prof. Horst Bischof, who approved my application to be a PhD student in the ICG. Horst gave me an opportunity of great value to start my scientific career in the field of computer vision. I will never forget his encouragements during the beginning period of my study in the ICG. I am grateful to him for choosing me four years ago. I found myself lucky to gain his trust in me.

I wish to thank all my present and former colleagues and scientists from the ICG, as well as all the co-authors from other universities, who helped me in many ways to complete my thesis. I owe many heartfelt thanks to the ICG's vision group members: Martin Köstinger, Martin Lenz, René Ranftl, Stefan Heber, Thomas Mauthner, David Ferstl, Christian Reinbacher, Inayatullah Khan, Gottfried Graber, Markus Unger and Wei Yu. Special thanks to René Ranftl for interesting and fruitful discussions about the research work, as well as his patience to help me to deal with many annoying GPU programming problems. Meanwhile, many thanks to Thomas Mauthner for his encouragements when I

was suffering from depression.

Furthermore, I would like to acknowledge the system administrator - Andreas Wurm, who provided many technical support. Thanks to the ladies in the secretary office: Nicole Eichberger, Renate Hönel, Karin Maier, Eva-Maria Christina Fuchs for providing help in the office administrative works.

I am glad to know that there are people I can always count on when times are rough: my family and my Chinese friends in Graz. My greatest, deepest and most special thanks goes to my parents, Xiaohua and Yilan, my elder sisters Meirong and Haiyan, who made finishing this thesis possible with their unconditional love, endless patience and continuous support. I must also pay thanks to my Chinese friends in Graz: Huanghuang Yan, Qiang Chen, Jianhua Tong, Jianfeng Huang and Ge Jin, who play with me in leisure time. My gratitude is beyond words, but thank you for believing in me and encouraging me on this journey.

Finally, I must also thank the financial support from the CSC-FWF Scholarships, which is jointly run by the China Scholarship Council (CSC) and the Austrian Science Fund (FWF). In addition, I also acknowledge support from FWF under the START project BIVISION, No. Y729.

Contents

List of Figures

List of Tables

Preliminaries: notations and matrix calculus

0.1 Notations

In this thesis, a two-dimensional image u of size $m \times n$ is also represented as a long one-dimensional column vector $u \in \mathbb{R}^N$ with $N = m \times n$, according to the scanning manner shown in Figure 1. We will consider this representation frequently in our work, especially for derivations related to matrix calculus.

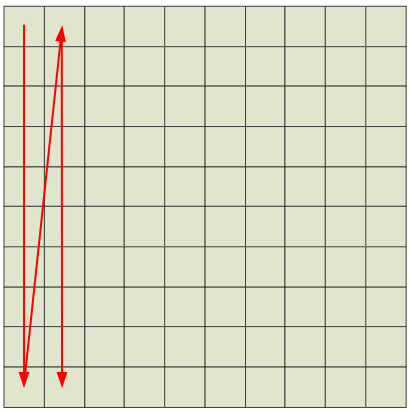


Figure 1: Vectorizing a 2D image into a long one-dimensional column vector

In this thesis, lower case letters such as "u, v, f" etc., denote scalar and vector variables, while capital letters such as "K, A, D, U" etc., denote matrix. We make use of convolution technique frequently in this thesis. The 2D convolution result of an image u with a linear filter $k \in \mathbb{R}^{r \times r}$ is written as $u * k$. The convolution result is also equivalent to two matrix-

vector product based formulations:

$$\underbrace{k * u}_{①} \Longleftrightarrow \underbrace{Ku}_{②} \Longleftrightarrow \underbrace{Uk}_{③},$$

where in the formulation ① both u and k are two-dimensional matrix, in the formulation ②, K is a highly sparse matrix $K \in \mathbb{R}^{N \times N}$ and u is a $N \times 1$ column vector, while in the formulation of ③, matrix $U \in \mathbb{R}^{N \times R}$ is constructed from image u and k is a $R \times 1$ column vector with $R = r \times r$. In this thesis, we will make use of the above equivalence frequently.

0.2 Matrix calculus

We will make heavy use of matrix calculus for our derivations. Usually, there exist two notational conventions that are used in the various fields, namely numerator layout and denominator layout. The fundamental issue is the way they lay out the gradient of scalar y with respect to a vector x. In general, there are two kinds of people in this world: those who think the gradient is a row vector, and those who think it is a column vector.

If we choose numerator layout for $\frac{\partial y}{\partial x}$, the gradient $\frac{\partial y}{\partial x}$ is represented as a row vector, while if we choose denominator layout for $\frac{\partial y}{\partial x}$, the gradient $\frac{\partial y}{\partial x}$ is represented as a column vector. In our work, we make use of the denominator layout for our derivations. We collect some useful formulas of matrix calculus that often appear in this thesis.

Let x and y be vectors of order n and m respectively:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix},$$

where each component y_i may be a function of all the x_j, a fact represented by saying that y is a function of x, or

$$y = y(x)\,.$$

If $n = 1$, x reduces to a scalar. If $m = 1$, y reduces to a scalar.

Derivative of a vector with respect to a vector

The derivative of the vector y with respect to vector x is the $n \times m$ matrix

$$\frac{\partial y}{\partial x} \stackrel{\text{def}}{=} \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_2}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_1} \\ \frac{\partial y_1}{\partial x_2} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_m}{\partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_1}{\partial x_n} & \frac{\partial y_2}{\partial x_n} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}$$

Derivative of a scalar with respect to a vector

If y is a scalar,

$$\frac{\partial y}{\partial x} \stackrel{\text{def}}{=} \begin{bmatrix} \frac{\partial y}{\partial x_1} \\ \frac{\partial y}{\partial x_2} \\ \vdots \\ \frac{\partial y}{\partial x_n} \end{bmatrix}$$

Derivative of a vector with respect to a scalar

$$\frac{\partial y}{\partial x} \stackrel{\text{def}}{=} \begin{bmatrix} \frac{\partial y_1}{\partial x} & \frac{\partial y_2}{\partial x} & \cdots & \frac{\partial y_m}{\partial x} \end{bmatrix}$$

Some useful vector derivative formulas

y	$\frac{\partial y}{\partial x}$
Ax	$A^\top$
$x^\top x$	$2x$
$x^\top Ax$	$Ax + A^\top x$

The chain rule for vector functions

Assume three vectors x, y, z have the following relationship

$$z \to y \to x \,,$$

i.e., z is a function of y, which is in turn a function of x. Then the gradient $\frac{\partial z}{\partial x}$ is obtained by the following chain rule

$$\frac{\partial z}{\partial x} = \frac{\partial y}{\partial x} \cdot \frac{\partial z}{\partial y} \,,$$

where we have to make use of this exact multiplication order. If we have an additional function w, which is a function of z, then the gradient $\frac{\partial w}{\partial x}$ is given as

$$\frac{\partial w}{\partial x} = \frac{\partial y}{\partial x} \cdot \frac{\partial z}{\partial y} \cdot \frac{\partial w}{\partial z} \,.$$

Chapter 1

Introduction

Contents

1.1 Image restoration problems

Image restoration is a class of low level computer vision problems. The goal of image restoration is to infer a clean image from the noisy (degraded) observation. Typical image restoration problems comprise the following tasks

- Image denoising for images corrupted by different types of noise, such as additive white Gaussian noise, multiplicative noise, salt-and-pepper noise (impulse noise) or Poisson noise.
- Block artifact reduction for images compressed by the JPEG compression algorithm.
- Image super resolution, which resolves a high resolution image from a single low resolution image or multiple frames.
- Image deconvolution for blurring images corrupted by some linear kernels.

- Image inpainting, the object of which is to reconstruct the underlying image only with a portion of sampling points.

A typical image degradation process can be formulated as

$$f = Hu + n\,, \tag{1.1}$$

where f is the observed noisy image, u is the underlying true image, H is a probably known linear operator (e.g., blurring kernel or down-sampling operator) and $n \sim \mathcal{N}(0, \sigma)$ is the zero-mean white Gaussian noise of known variance σ^2. Figure 1.1 shows some examples of degraded images generated by Equation (1.1).

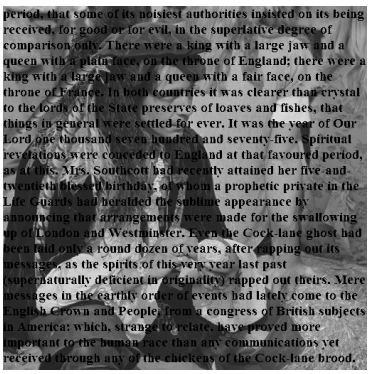
(a) Clean image

(b) Gaussian noise $\sigma = 25$

(c) Motion blur and $\sigma = 5$ Gaussian noise

(d) Down-sampled image with a factor 3

(e) 40% random sampling pixels



(f) Partially occluded image with overlaying text

Figure 1.1: Examples of degraded images generated by the process of Equation (1.1).

(a) JPEG compressed image with quality $q = 10$

(b) Nakagami multiplicative noise with $L = 8$

(c) Poisson noise with peak value peak $= 40$

Figure 1.2: Additional examples of degraded images.

Some other degraded images, which can not be casted with the formulation of Equation (1.1), are shown in Figure 1.2, including JPEG compressed image, noisy image corrupted by Nakagami distribution based multiplicative noise or Poisson noise.

The image restoration problem is stated as: given the observed noisy image f with the probably known parameters of the image degradation model (H, σ), estimate the underlying clean image u. As we know, the image restoration problems are mathematically ill-posed inverse problems in that existence, uniqueness, and stability of solutions cannot be guaranteed in the absence of additional constraints. Therefore, it is generally not possible to directly compute the solution of Equation (1.1). A potential approach is to constrict the space of possible solutions to physically meaningful ones. In this regard, constraints such as smoothness have been useful expressions of generic, a priori information about possible solutions.

1.2 Variational formulation for solving image restoration problems

An effective framework to incorporate the image prior information is to make use of the Bayesian inference theory. From the perspective of probability, we want to select the hypothesis having the highest probability, i.e., we are facing the following optimization problem

$$u^* = \arg\max_u P(u|f)\,, \tag{1.2}$$

where $P(u|f)$ is the posterior probability of the a certain hypothesis u given the observation f. This principle is known as maximum a posteriori (MAP) estimation. According to the Bayesian inference theory, we know that the conditional probability $P(u|f)$ can be written as

$$P(u|f) = \frac{P(f|u)P(u)}{P(f)}\,, \tag{1.3}$$

where $P(u)$ is the prior probability of u (i.e., a prior knowledge or image prior) and $P(f|u)$ is the conditional probability of the observed image f given the true image u (also known as likelihood or data model), which indicates the compatibility of the evidence with the given hypothesis. The expression $P(f)$ is called partition function or normalization factor that guarantees that the posterior probability sum to one. In the optimization problem Equation (1.2), $P(f)$ is a constant, which can be omitted during minimization process.

From the Gibbs distribution of image prior model [53], a typical image prior model $P(u)$ is formulated as

$$P(u) = \frac{1}{Z} e^{-\phi(Tu)}\,, \tag{1.4}$$

where ϕ is called penalty function and T is some linear operator applied to the image u, e.g., derivatives filters (first, second, or higher order), wavelets transform, curvelets etc.. Z is the so-called partition function that guarantees that the equation $\sum_u P(u) = 1$ holds.

As in Equation (1.1), we are dealing with independently distributed additive Gaussian noise, the data model is straightforward to model using the following pixel-wise product of normal distributions

$$P(f|u) = \prod_{p=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\left((Hu)_p - f_i\right)^2}{2\sigma^2}} = \frac{1}{C} e^{-\frac{\|Hu-f\|_2^2}{2\sigma^2}}\,, \tag{1.5}$$

where N is the pixel number of the image f, C is a constant, $\|\cdot\|_2$ denotes standard ℓ_2-norm and σ^2 is the noise variance.

Substituting Equation (1.5) and Equation (1.4) into Equation (1.3), ignoring the constants, we arrive at

$$P(u|f) \propto e^{-\left(\phi(Tu) + \frac{\|Hu-f\|_2^2}{2\sigma^2}\right)}\,. \tag{1.6}$$

It is clear that maximizing the posterior probability $P(u|f)$ is equivalent to minimizing the following energy functional

$$E(u) = \phi(Tu) + \frac{\|Hu - f\|_2^2}{2\sigma^2}\,. \tag{1.7}$$

This leads us to the following variational formulation of our image restoration problem

$$\arg\min_{u\in\mathbb{R}^N} E(u) = \phi(Tu) + \frac{\lambda}{2}\|Hu - f\|_2^2\,, \tag{1.8}$$

where the first term is known as the regularization term (derived from the image prior model), and the second term is known as the data fidelity term (derived from the likelihood). λ is a free parameter to tune, which provides a trade-off between regularization and data fitting.

1.3 Regularization techniques for image restoration

Through regularization, the original ill-posed inverse problem Equation (1.1) can be reformulated as a well-posed variational model Equation (1.8) whose solution is computable. Historically, the first regularization technique coming into consideration is the Tikhonov model [126, 127], which employs quadratic penalty functionals, imposing global smoothness constraints on possible solutions. That is to say, the Tikhonov regularization involves the penalty function $\phi(z) = z^2$, and the corresponding variational model is given as

$$\arg\min_{u\in\mathbb{R}^N} E(u) = \|Tu\|_2^2 + \frac{\lambda}{2}\|Hu - f\|_2^2\,, \tag{1.9}$$

for which, in general a closed-form solution exists. If one selects the linear operator T as the first order finite differentiation operator ∇, the optimal solution u^* of the optimization problem Equation (1.9) can be achieved according to the optimality condition

$$\left.\frac{\partial E}{\partial u}\right|_{u=u^*} = 0\,, \tag{1.10}$$

where $\frac{\partial E}{\partial u}$ is given as

$$\frac{\partial E}{\partial u} = \nabla^\top\nabla u + \lambda H^\top(Hu - f)\,. \tag{1.11}$$

For the case of image denoising, $H = \mathbf{I}$ with $\mathbf{I} \in \mathbb{R}^{N\times N}$ the identity matrix, the minimizer is given as

$$u^* = \lambda\left(\nabla^\top\nabla + \lambda\mathbf{I}\right)^{-1}\cdot f\,. \tag{1.12}$$

Figure 1.3(b) shows an example of image restoration result obtained by the Tikhonov model for Gaussian noise corrupted image.

From Figure 1.3(b), one can see that the denoising result generated by the Tikhonov

(a) Gaussian noise $\sigma = 25$, PSNR: 20.18

(b) Tikhonov model ($\lambda = 0.1$), PSNR: 24.52

(c) ROF model ($\lambda = 0.045$), PSNR: 28.61

Figure 1.3: Image restoration using the Tikhonov model Equation (1.9) and the ROF model Equation (1.13), respectively.

model is very blurry, i.e., all the edges in the image are smoothed out. This result is not surprising, as the Tikhonov regularizations impose global smoothness on the underlying image. Therefore, discontinuities (e.g., edges) present great difficulties to standard Tikhonov regularization, as their reconstruction requires a more precise spatial control over the smoothing properties.

In summary, the Tikhonov regularizations solve the ill-posedness of the image restoration problems, but the performance is not satisfactory, and needs to be improved via better regularization techniques.

In 1992, Rudin-Osher-Fatemi (ROF) [114] made a big step forward in this research direction by proposing the total variation (TV) based image denoising model, in which the regularization term is defined as total variation norm of image u, which has been well-known from measure theory [45]. The resulting model is well-known as the ROF model. The discrete ROF model for image denoising is given as

$$\arg\min_{u\in\mathbb{R}^N} \|\nabla u\|_{2,1} + \frac{\lambda}{2}\|u-f\|_2^2\,, \tag{1.13}$$

where $\|\nabla u\|_{2,1}$ denotes the discrete version of the isotropic total variation norm defined as

$$\|\nabla u\|_{2,1} = \sum_{p=1}^{N} \sqrt{(\nabla_x u)_p^2 + (\nabla_y u)_p^2}\,, \tag{1.14}$$

where ∇_x and ∇_y denote the linear operators computing the gradient in x-direction and y-direction, respectively. Comparing the ROF model Equation (1.13) to the Tikhonov regularized denoising model Equation (1.9), the only difference lies in that we replace the quadratic penalty function with the ℓ_1 norm. It turns out that this seemingly small modification has brought significant improvement for the image denoising problem, an illustrative example is shown in Figure 1.3. The corresponding minimization problem is solve efficiently via the recently proposed primal-dual algorithm [21].

The main advantage of the TV regularization is that it disfavors small oscillations such as noise but allows for sharp discontinuities such as edges. This ability to preserve edges can also be explained by means of robust statistics [66]. The ℓ_1 penalty function employed in the TV regularization is a robust potential function, which is not sensitive to the outliers and allows existence of outliers. In the context of image processing, outliers signify large magnitude of the image gradient, i.e., image edges. In contrast, the quadratic penalty function exploited in the Tikhonov regularizations penalize the outliers too much, and therefore, the outliers are smoothed out, i.e., image edges are blurred. From Figure 1.3, one can see the difference between two regularization techniques. Figure 1.3(c) demonstrates that the ROF model can effectively remove the noise, while simultaneously preserve the image edges.

The promising edge-preserving property of the ROF model demonstrates the effectiveness of TV regularization. Since then, the success of the ROF model has inspired tremendous works based on the TV regularization, which is still widely used nowadays. No doubt the TV regularization is one of the most popular regularization technique up to now.

Despite of the effectiveness of the TV regularization, it has a main drawback that the TV regularized variational models suffer from the so-called "stair-case" artifacts, i.e., the TV-based models favor minimizer with piece-wise constant regions. In order to ameliorate the undesirable stair-casing effect, a lot of efforts have been devoted to address this issue. An effective approach is to introduce higher order derivatives into the regularization term. Most of the work concentrate on exploiting higher-order derivatives, e.g., fourth-order derivatives, see [10, 23, 78, 88, 139] for instance.

There are several typical variants of fourth order derivative based regularizers, which

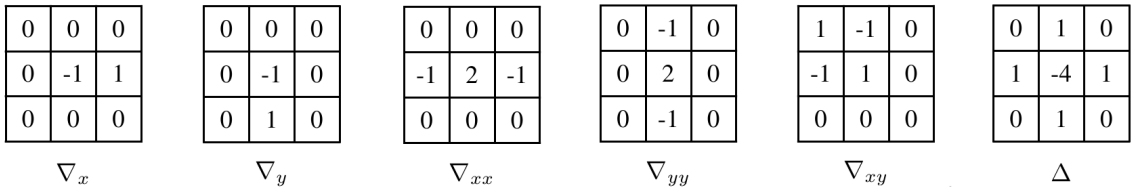


Figure 1.4: Discrete first- and second-order derivatives, interpreted as 2D linear filters for image processing.

replace the standard TV norm using the following energy functionals

$$\begin{cases} \mathcal{R}_1(u) = \|\Delta u\|_1 = \sum\limits_{i=1}^{N} |(\Delta u)_p| \\ \mathcal{R}_2(u) = \sum\limits_{p=1}^{N} |(\nabla_{xx} u)_p| + |(\nabla_{yy} u)_p| \\ \mathcal{R}_3(u) = \sum\limits_{p=1}^{N} \sqrt{(\nabla_{xx} u)_p^2 + (\nabla_{yy} u)_p^2 + (\nabla_{xy} u)_p^2 + (\nabla_{yx} u)_p^2}, \end{cases} \tag{1.15}$$

where $\Delta \in \mathbb{R}^{N\times N}$ is the Laplacian operator defined as $\Delta = -\nabla^\top \nabla$, and $\nabla_{xx} \in \mathbb{R}^{N\times N}$, $\nabla_{yy} \in \mathbb{R}^{N\times N}$, $\nabla_{xy} \in \mathbb{R}^{N\times N}$ and $\nabla_{yx} \in \mathbb{R}^{N\times N}$ correspond to the second-order derivatives. In the discrete case, all of these linear operators can be interpreted as linear filters, which are shown in Figure 1.4.

Numerical results demonstrate that the variational models involving higher-order derivatives can remove the staircase artifacts to a remarkable degree. Some researcher also consider hybrid models, which make a linear combination of the lower- and higher-order derivatives to better preserve the discontinuities along edges, and meanwhile, recover smooth regions, see [80, 89, 102] for an example. A notable work of considering the higher-order derivative information is the so-called TGV (Total generalized variation) model [18] proposed by Bredies et al., which has become a very popular convex regularizer in the imaging community.

1.4 Relations between regularization and nonlinear diffusion filtering

As revealed in many previous works [19, 48, 118, 122, 133], there exist closely relations between the regularization based methods and the nonlinear diffusion processes. In the nonlinear diffusion framework, natural relations between biased diffusion and regularization theory exist via the Euler equation for the regularization functional. This Euler

equation can be regarded as the steady-state of a suitable nonlinear diffusion process with a bias term, called reaction diffusion [95]. It is shown in [118] that regularization may be regarded as diffusion filtering with an implicit time discretization where one single step is used. Thus, iterated regularization with small regularization parameters approximates a diffusion process.

In a more general sense, if we consider the steepest descent method to solve the regularization involved energy functional, it naturally leads to a diffusion process, i.e.,

$$u^{t+1} = u^t - \Delta t \cdot \left.\frac{\partial E}{\partial u}\right|_{u^t} \Longleftrightarrow \frac{\partial u}{\partial t} = \frac{u^{t+1} - u^t}{\Delta t} = -\left.\frac{\partial E}{\partial u}\right|_{u^t}, \tag{1.16}$$

Therefore, any energy functional can lead to a nonlinear diffusion process, and any nonlinear diffusion process may correspond to certain regularization functional.

Let us take for instance, the well-know Pernona-Malik nonlinear diffusion filtering model [104], which reads as

$$\frac{\partial u}{\partial t} = \nabla \cdot (g(|\nabla u|)\nabla u) \,, \tag{1.17}$$

in the continuous form, where $g(z)$ is the diffusivity function given as

$$g(z) = \frac{1}{1 + az^2} \,, \tag{1.18}$$

$u(t = 0) = u_0 = f$, Neumann boundary conditions. The corresponding discrete version is given as

$$\frac{u^{t+1} - u^t}{\Delta t} = -\sum_{i=\{x,y\}} \nabla_i^\top \mathcal{D}(u^t) \nabla_i u^t \,, \tag{1.19}$$

where matrices ∇_x and $\nabla_y \in \mathbb{R}^{N\times N}$ extract the gradients in x-direction and y-direction, respectively. $\mathcal{D}(u) \in \mathbb{R}^{N\times N}$ is defined as a diagonal matrix

$$\mathcal{D}(u) = \operatorname{diag}\left(g\left(\sqrt{(\nabla_x u)_p^2 + (\nabla_y u)_p^2}\right)\right)_{p=1,\cdots,N} .$$

Therefore, the regularization functional associated with the nonlinear diffusion process (1.19) is given as

$$E(u) = \sum_{p=1}^{N} \phi\left(\sqrt{(\nabla_x u)_p^2 + (\nabla_y u)_p^2}\right) \,, \tag{1.20}$$

where $\phi(z) = \frac{1}{2a} \cdot \log(1 + az^2)$. Note the relation between function ϕ and g, which reads

as

$$\phi'(z) = zg(z)\,,$$

where the function $\Phi(z) = zg(z)$ is the so-called flux/influence function.

It is clear that the steady-state of the diffusion process (1.19), i.e., the minimizer of the energy functional (1.20) is the trivial solution $u = c$ (c is a constant). In order to achieve a meaningful solution, the diffusion process should be stopped at certain optimal time t^*, or equivalently by adding a reaction term to force the solution close to the initialization u_0, which naturally leads to the reaction diffusion process, given as

$$\frac{\partial u}{\partial t} = -\sum_{i=\{x,y\}} \nabla_i^\top \mathcal{D}(u)\nabla_i u + \lambda(f-u)\,, \tag{1.21}$$

where $f - u$ defines the reaction force, and λ controls the influence of the reaction force. Therefore, searching the optimal stopping time t^* in the original diffusion process (1.19) becomes finding an optimal parameter λ^* in the reaction diffusion model (1.21). The corresponding energy functional with respect to the reaction diffusion process (1.21) is given as

$$\arg\min_u E(u) = \sum_{p=1}^{N} \phi\left(\sqrt{(\nabla_x u)_p^2 + (\nabla_y u)_p^2}\right) + \frac{\lambda}{2}\|u-f\|_2^2\,. \tag{1.22}$$

Concerning the TV regularized energy functional (1.13), the corresponding diffusion process is given as

$$\frac{\partial u}{\partial t} = -\sum_{i=\{x,y\}} \nabla_i^\top \mathcal{D}(u)\nabla_i u + \lambda(f-u)\,, \tag{1.23}$$

with $g(z) = \frac{1}{\sqrt{z^2+\varepsilon^2}}$ and $\phi(z) = \sqrt{z^2+\varepsilon^2}$ with $\varepsilon > 0$ smoothing parameter, corresponding to the smoothed version of the TV regularization, which has been investigated in many previous work [19, 48, 122].

The effectiveness of the Pernona-Malik nonlinear diffusion process inspired many works to investigate nonlinear diffusion processes involving diffusivity functions of the similar form to (1.18), which correspond to non-convex penalty functions. In the following section, we will discuss non-convex penalty functions.

1.5 From convex regularizations to non-convex regularizations

As we have seen in the last section that all the regularization techniques are convex. On the other hand, a considerable efforts, in the meanwhile have been made to investigate non-convex penalty function, instead of the convex ℓ_1-norm in the TV-norm. Since the pioneering work of Geman and Geman [53], different non-convex penalty functions ϕ have been considered in either a statistical or variational framework, representative works include [61, 71, 74, 96, 104, 116]. Experimental results show that the associated minimizers provide solutions with neat edges and well-smoothed homogeneous regions.

Several typical non-convex penalty functions, which have been intensively investigated are shown in Table 1.1. There are also some non-parameterized penalty functions, which are derived from image statistic [117] or learned training samples [70].

Non-convex penalty functions	
$\phi(z) = \|z\|^p,\ 0 < p < 1$, [71, 116]	$\phi(z) = 1 - e^{-az^2}$, [74, 104]
$\phi(z) = \log(1 + az^2)$, [61, 104]	$\phi(z) = \frac{a\|z\|}{1+a\|z\|}$, [96]
$\phi(z) = \log(1 + a\|z\|)$, [20]	$\phi(z) = \frac{az^2}{1+az^2}$, [51]

Table 1.1: Commonly used non-convex penalty functions.

Recently, researchers propose to combine the non-convex penalty function with the higher-order derivatives [100] or other linear filters e.g., DCT (Discrete cosine transform) filters [73]. Especially, [73] exhibits very encouraging results for image denoising problems, which are significantly better than all the previous variational models, and are also strongly competitive to state-of-the-art denoising algorithms.

The usage of non-convex penalty function is also motivated by the observation of the statistical properties of natural images. In [65], it was shown that the gradients of natural images typically exhibit heavy-tailed distribution, which is clearly non-Gaussian. Therefore, convex penalty functions cannot capture this statistical property, and a non-convex penalty function is required. Figure 1.5 shows the negative log PDF (probability density function) of the first-order derivative ∇_x applied to natural images. From Figure 1.5, one can see that even thought the commonly used ℓ_1 penalty function can provide much better fitting than the quadratic penalty function derived from the Gaussian distribution, it is quite far away from the true PDF; in contrast, the non-convex penalty function $|z|^{\frac{1}{2}}$ can provide the best fitting to the heavy-tailed shape of the true density function.

In a recent work [73], Kunisch and Pock exploit a non-convex regularization based

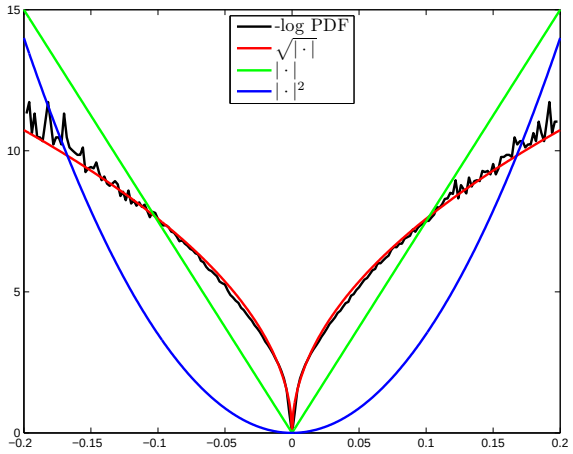


Figure 1.5: Negative log probability density function (PDF) of the filter response of the first-order derivative filter ∇_x applied to natural images. Note that the non-convex penalty function $|z|^{\frac{1}{2}}$ function provides the best fitting to the heavy tailed shape of the true density function.

variational model for image denoising, which involves the non-convex penalty function $|z|^{\frac{1}{2}}$ and the DCT filters. The associated variational model is given as

$$\arg\min_{u\in\mathbb{R}^N} \sum_{i=1}^{N_k} \alpha_i \|K_i u\|_{\frac{1}{2}}^{\frac{1}{2}} + \frac{1}{2}\|u-f\|_2^2\,, \tag{1.24}$$

where $\|K_i u\|_{\frac{1}{2}}^{\frac{1}{2}} = \sum_{p=1}^{N} |K_i u|_p^{\frac{1}{2}}$ and $K_i \in \mathbb{R}^{N\times N}$ are the matrix-form of the $\text{DCT}_{5\times 5}$ filters shown in Figure 1.6, i.e., K_i is a sparse matrix implemented such that $K_i u \Longleftrightarrow k_i * u$ (k_i is a 2D filter kernel and $*$ denotes the 2D convolution operation). The weights α_i of this model is optimized via a bi-level optimization approach. With the optimized parameters α_i, this model can obtain a denoising result presented in Figure 1.7. The corresponding non-convex non-smooth minimization problem is solved with an iterated ℓ_1 algorithm [98].

From Figure 1.7, one can see that compared to the ROF model, Equation (1.24) generates much more natural denoising result without stair-casing artifacts. Moreover, it also presents strongly competitive performance relative to a state-of-the-art image denoising algorithm - BM3D [36].

In summary, we have shown that an effective image regularizer should possess two important properties:

- *Higher-order linear filters, i.e., filters with larger influence size relative to pair-wise filters;*

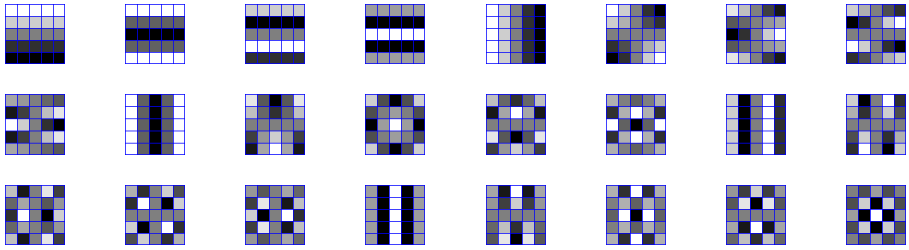


Figure 1.6: $\mathrm{DCT}_{5\times5}$ linear filters.

(a) ROF model ($\lambda = 0.045$), PSNR: 28.61

(b) Non-convex model (1.24), PSNR: 29.25

(c) BM3D [36], PSNR: 29.62

Figure 1.7: Image restoration using the ROF model Equation (1.13), non-convex regularization based model Equation (1.24), and a state-of-the-art denoising algorithm - BM3D, respectively.

- *Non-convex penalty functions, which will prove to be the critical factor for an effective regularizer, latter in Section 2.5.*

Based on the existing work of image regularization, especially the work in [73], a natural question arises:

Can we go further? Namely, Can we find or define better image regularizers for image restoration problems?

The answer to this question leads us to the contributions of this thesis.

1.6 Contributions of the thesis

Then contents of this thesis is based on the work presented in [25–30, 47, 97], which is the result of joint work with Prof. Thomas Pock, Peter Ochs, René Ranftl, Wensen Feng, Wei Yu and Prof. Horst Bischof. The main contributions of this thesis lie in that we

have trained two classes of effective, as well as highly efficient image restoration models, namely:

- **variational models with optimal image regularizers, which are trained via bi-level optimization;**
- **effective reaction diffusion approaches, which are also optimized from training samples using loss minimization.**

1.6.1 Learning better image regularizers

As previously shown, the non-convex image regularizer in Equation (1.24) employs fixed $\text{DCT}_{5\times5}$ filters, and only the weights of corresponding filters are optimized from training samples. Intuitively, if we also optimize the linear filters in the training, this process should have the potential to yield superior results, as the regularizer has more freedom to tune. Therefore, in this thesis we consider the following image regularization model

$$\mathcal{R}(u) = \sum_{i=1}^{N_k} \alpha_i \left(\sum_{p=1}^{N} \phi_i \left((K_i u)_p \right) \right) = \sum_{i=1}^{N_k} \alpha_i \left(\sum_{p=1}^{N} \phi_i \left((k_i * u)_p \right) \right) , \tag{1.25}$$

where k_i are a set of linear filters with associated weights α_i, both of which will be trained simultaneously in our work. ϕ_i are non-convex penalty functions, which are chosen from some candidate functions, and then fixed in the training.

One can see the investigated image regularization naturally leads to a widely used image prior model, called Fields of Experts (FoE), which was proposed by Roth and Black [111]. The FoE model is a filter-based higher-order Markov Random Fields (MRF) model. Our goal of our work is to learn a better FoE image prior model by using bi-level optimization.

Interestingly, we also find that the regularization model (1.25) closely links to the so-called co-sparse analysis model [43, 60, 105, 112, 113], which is a recently introduced sparse representation model for signal processing. In Section 2.2, we provide our insights into the analysis operator learning problem. We show that the analysis prior model, which is usually designed based on image patch rather than the whole image, is a simplified version of the FoE model. We argue that for the analysis prior modeling, we should turn to the image-based modeling framework - FoE, at least in the circumstances of image processing. Therefore, we claim that the so-called analysis prior model is actually equivalent to the filter-based MRF model, and we propose to exploit the framework of FoE model to define

the analysis prior model.

There are some previous work to train the parameters of the FoE model. In Section 2.1, we review existing training approaches for the FoE model. Broadly speaking, recent years have seen the emergence of two main approaches for learning the parameters in the FoE model: (1) probabilistic learning using sampling-based algorithms [52, 111, 119] and (2) loss-specific training based on MAP estimate [8, 40, 115]. After investigating existing training approaches, we find that the performance of the loss-specific training has been significantly underestimated in existing works. Therefore, we are motivated to revisit this approach. We propose a refined training algorithm for the loss-specific training scheme. Finally, numerical results demonstrate that we can obtain a substantial gain in the performance, i.e., our refined training strategy pays off.

With our refined training algorithm, we train a variety of image regularization models with different configurations, including different penalty functions, filter size and number of filters. The training is mainly based on the Gaussian denoising problem. We comprehensively investigate the influence of different aspects of the regularization model, and our results suggest that the penalty function is the most important factor.

As the image regularization model investigated in this thesis is non-convex, the resulting variational models impose hard non-convex optimization problems, and therefore, an efficient optimization algorithm is required from a practical point of view. To that end, we propose the so-called iPiano algorithm to efficiently solve the corresponding problems. The proposed iPiano algorithm is based on a forward-backward splitting scheme with an inertial force term, and it is applicable for a class of non-convex minimization problem, which is composed of a smooth (possibly nonconvex) function and a convex (possibly non-smooth) function. We present some overall aspects of the iPiano algorithm in Section 2.3.3.

1.6.2 Applications of the learned image regularizers

Once we have trained an image regularizer, we apply it to various of classical image restoration problems, such as image denoising with different noise types, deblocking for JPEG compressed images, image super resolution, non-blind image deconvolution and image inpainting. Even though the image regularization model in our work is discriminatively trained based on the Gaussian denoising problem, it turns out that it well generalizes to other image restoration problems, not just the image denoising problem. Therefore, the discriminatively trained image regularizer can be treated as a generative model or an

image prior model, which is applicable for any image restoration problem *.

In Section 3.1, we mainly focus on the image denoising task with different noise types at different noise levels, including commonly investigated additive white Gaussian noise, multiplicative noise and impulse noise. The obtained denoising results are carefully compared to current state-of-the-art algorithms. Numerical results demonstrate that our trained models are on par with highly specialized image denoising algorithms.

In Section 3.2, we focus on image deblocking for JPEG compressed images. To that end, we propose a novel variational model for reducing blocking artifacts, which combines our trained image regularizer with the indicator function of the quantization constraint set (QCS). This new model leads to a generally hard non-convex non-smooth optimization problem, and we make use of our proposed iPiano algorithm to solve it efficiently. We first review some related work for image deblocking, and then introduce our algorithm. Again, the deblocking results are compared with related work to demonstrate the effectiveness of our proposed variational model.

In Section 3.3, experiments of other image restoration problems are presented. For image super resolution, we first consider the commonly addressed single image super resolution task, then we address the case of multiple frames. Regarding the task of multi-frame image super resolution, we also make use of a variational method, which incorporates additional warping information, derived from the corresponding optical flow estimation. Then, we turn to the problems of image inpainting and non-blind image deconvolution. We present results of typical cases exploited in the literature, and comparisons are made.

It is clear that all the variational models involving the trained image regularizers correspond to non-convex optimization problems, sometimes even non-smooth. Fortunately, these minimization problems can be efficiently solved by the proposed iPiano algorithm. Moreover, the trained image regularization models come along with the additional advantage of simplicity, as it only involves 2D image convolution operations with a few linear filters. Therefore, our model is well suited to GPU parallel computation. For some image restoration problems, we also implement the GPU version of our model, which additionally accelerates the inference procedure.

In summary, the variational models, that incorporate our trained image regularizers, can effectively solve the image restoration problems, meanwhile possessing the property of high efficiency.

*Note that in principle it will lead to probable improvements if one train a specialized FoE prior model for a specific data term.

1.6.3 Learning effective reaction diffusion processes

Although the variational image restoration model based on the trained regularizer can be efficiently solved using the iPiano algorithm, the inference process is not easy in that it imposes a generally demanding non-convex minimization problem, and usually couples of hundreds of iterations are required to converge to a stationary solution. Besides, there are some drawbacks in the loss-specific training of the image regularizer in the variational model. The loss-specific training of the image regularizer is formulated as a bi-level optimization problem, which is solved by gradient-based algorithms. However, the computation of the gradients in this problem is quite expensive, because it needs to calculate the inverse matrix of a huge matrix, and the minimizer (with high accuracy) of corresponding variational model is also required.

Bearing these drawbacks in mind, we are therefore motivated to think about a possibility:

Starting from the energy functional in the variational model, is it possible to design a process, which only involves few steps (e.g., < 10 *steps), while can reach a result still owning good performance?*

As describe in Section 1.4, an energy functional minimization problem can be related to a diffusion process. Therefore, the FoE model regularized energy functional is also related the following diffusion process (given as gradient descent steps)

$$u^{t+1} = u^t - \left.\frac{\partial E}{\partial u}\right|_{u^t}, \tag{1.26}$$

where $\left.\frac{\partial E}{\partial u}\right|_{u^t}$ is the gradient of the functional E at point u^t, which is defined by the FoE image prior regularized functional. Now the new goal is to train the parameters (e.g., the FoE image regularization model) in each gradient descent step, such that the output of this process is optimized.

We find that the above gradient descent process (1.26) naturally leads to a nonlinear reaction diffusion process with higher-order filters. In Section 4.1, we review the relation to the commonly investigated nonlinear diffusion process. Then we propose a trainable nonlinear diffusion model, which is parameterized by the linear filters in the FoE model and influence functions controlling the diffusion behavior. We train two specific diffusion processes for the image denoising problem and image deblocking task, respectively.

We still make use of the loss-minimization training scheme to optimize the proposed diffusion process. It turns out that the trained nonlinear diffusion processes surprisingly

exhibit superior performance even compared to recent state-of-the-arts, meanwhile with extremely high efficiency.

Chapter 2

Learning optimized FoE models using loss-specific minimization

Contents

2.1 Revisiting loss-specific training of filter-based MRFs

Recall that the image regularization model to be exploited in this thesis is defined as the following FoE image prior model (filter-based MRF)

$$\mathcal{R}(u) = \sum_{i=1}^{N_k} \sum_{p=1}^{N} \alpha_i \phi_i \left((k_i * u)_p \right) , \tag{2.1}$$

where the filters k_i with associated weights α_i are to be trained from samples, and the penalty function is chosen from some candidate functions, which will be then fixed in the training process.

In the following subsections, we first introduce the well-known image prior model called Fields of Experts (FoE), which was proposed by Roth and Black in 2009 [111].

The FoE image prior model is trained from samples. Up to now there already exist a few previous works to train the parameters of the FoE model. Then, we review existing training approaches, and finally show the motivation to revisit a specific training scheme, namely, loss-specific (or loss minimization) training approach, which will be utilized in this thesis.

2.1.1 The Fields of Experts (FoE) model

In 2005, the FoE image prior model was first proposed in [110]. It has attracted many research attention until now due to its effectiveness for many image restoration problems. The FoE image prior model is defined by a set of linear filters $\{k_i\}_{i=1}^{N_k}$ (N_k is the number of filters) and potential functions ρ_i. The FoE model defines the probability density of a full image u, which is formally given as

$$P(u) = \frac{1}{Z(\Theta)} \prod_{i=1}^{N_k} \prod_{p=1}^{N} \rho_i((k_i * u)_p; \alpha_i), \quad \Theta = \{\theta_1, \cdots, \theta_{N_k}\}, \tag{2.2}$$

where $\theta_i = \{k_i, \alpha_i\}$ are the model parameters, ρ_i is the so-called potential functions or experts, $Z(\Theta)$ is the normalization constant. According to the Gibbs distribution, the probability density $P(u)$ can be rewritten as $P(u) = \frac{1}{Z(\Theta)} \exp(-E_{\text{FoE}}(u, \Theta))$ with

$$E_{\text{FoE}}(u, \Theta) = -\sum_{i=1}^{N_k} \sum_{p=1}^{N} \log \rho_i((k_i * u)_p; \alpha_i). \tag{2.3}$$

Comparing Equation (2.1) and Equation (2.3), one can see that they are exactly the same if we choose the penalty function according the rule

$$\alpha_i \phi_i(z) = -\log \rho_i(z; \alpha_i).$$

Based on the observation that responses of mean-zero linear filters typically exhibit heavy-tailed distributions [65] on natural images, three typical types of potential functions have been investigated, including the Student-t distribution (ST), generalized Laplace

model	potential	training	inference	PSNR
5×5 FoE	ST&Lap.	contrastive divergence	MAP, CG	27.77[111]
3×3 FoE	GSMs	contrastive divergence	Gibbs sampling	27.95[119]
5×5 FoE	GSMs	persistent contrastive divergence	Gibbs sampling	28.40[52]
5×5 FoE	ST	loss-specific(truncated optimization)	MAP, GD	28.24[8]
5×5 FoE	ST	loss-specific(truncated optimization)	MAP, L-BFGS [86]	28.39[40]
5×5 FoE	ST	loss-specific(implicit differentiation)	MAP, CG	27.86[115]

Table 2.1: Summary of various typical MRF-based systems and the average denoising results on 68 test images [111] with $\sigma = 25$

distribution (GLP) and Gaussian scale mixtures (GSMs) function.

$$\begin{cases} \rho(z;\alpha_i) = (1+z^2)^{-\alpha_i}, \alpha_i > 0 & \text{(ST)} \\ \rho(z;\alpha_i) = e^{-|z|^{\alpha_i}}, 0 < \alpha_i < 1 & \text{(GLP)} \\ \rho(z;\alpha_i) = \sum\limits_{j=1}^{J} \alpha_{ij} \cdot \mathcal{N}(z;0,\gamma_i^2/s_j), \alpha_{ij} \geq 0 & \text{(GSMs)}, \end{cases} \tag{2.4}$$

where $\mathcal{N}(\cdot)$ denotes the mean-zero Gaussian function, and α_{ij} are the normalized weights of the Gaussian component with scale s_j and base variance γ_i^2. Therefore, for the ST and GLP distributions, we can obtain the corresponding penalty functions, namely,

$$\begin{cases} \rho_i(z;p_i) = (1+z^2)^{-\alpha_i} \Longleftrightarrow \phi_i = \alpha_i \log(1+z^2) \\ \rho_i(z;p_i) = e^{-|z|^{\alpha_i}} \qquad \Longleftrightarrow \phi_i = |z|^{\alpha_i} \ (L_p \text{ norm}). \end{cases} \tag{2.5}$$

Now we can see that the penalty functions ϕ in (2.5) are commonly used non-convex functions already mentioned in the last chapter. The goal of this chapter is to train the FoE image prior model for image restoration problems such that it performs in some optimal way.

2.1.2 Related works and motivation to revisit loss-specific training scheme

In recent years several approaches for learning the parameters of the FoE model have emerged [8, 40, 52, 111, 115, 119]. Table 2.1.2 gives a summary of several typical methods and the corresponding average denoising PSNR results based on 68 test images from Berkeley database with $\sigma = 25$ Gaussian noise.

In general, existing training approaches fall into two main types: (1) probabilistic train-

ing using sampling-based algorithms, such as (persistent) contrastive divergence ((P)CD); (2) loss-specific training based on the MAP estimation. Roth and Black [111] first introduced the concept of FoE and proposed an approach to learn the parameters of FoE model which uses a sampling strategy and the idea of CD to estimate the expectation value over the model distribution. Schmidt et al. [119] improved the performance of their previous FoE model [111] by changing (1) the potential function from ST and Laplace to GSMs and (2) the inference method from MAP estimate to Bayesian minimum mean squared error estimate (MMSE). The same authors present their latest results in [52], where they achieve significant improvements by employing an improved learning scheme called PCD instead of previous CD.

In this thesis, we are dedicated to MAP estimation for the usage of the FoE image prior model, due to its preferable computational efficiency relative to the MMSE estimation. We also prefer the loss-specific training based on MAP estimation, as it is stated in a previous work [115] that:

if the intent is to use MAP estimates and evaluate the estimates using some criterion, like PSNR, obviously, a better strategy is to find the parameters such that the quality of MAP estimates are directly optimized. Therefore, if the goal is to maximize PSNR, maximizing a likelihood (probabilistic training) and then hoping that it leads to MAP estimates with good PSNR values is not the best strategy. Instead, one should choose the parameters such that the MAP estimates have the highest PSNR possible.

Samuel and Tappen [115] present a novel loss-specific training approach to learn MRF parameters under the framework of bi-level optimization [31]. In their approach they train the MRF model by optimizing the parameters such that the minimum energy solution of the MRF model is as similar as possible to the ground-truth. They use a plain gradient-descent technique to optimize the parameters, where the essence of this learning scheme - the gradients, are calculated by using implicit differentiation technique. Domke [40] and Barbu [8] propose two similar approaches for the training of MRF model parameters also under the framework of bi-level optimization. Their methods are some variants of standard bi-level optimization method [115]. In the modified setting, the MRF model is trained in terms of results after optimization is truncated to a fixed number of iterations, i.e., they do not solve the energy minimization problem exactly; instead, they just run some specific optimization algorithm for a fixed number of steps (10 in [40] and 4 in [8]). The main reason why they use the strategy of truncated optimization is due to its lower computational expense compared to standard "full" training.

	BM3D [36]	GMM-EPLL [143]	LSSC [91]
$\sigma = 25$	28.56	28.68	28.70

Table 2.2: Average denoising results of the current state-of-the-art methods for 68 test images ($\sigma = 25$)

Even though recent years have seen the introduction of various new approaches for training the MRF models, unfortunately, taking image denoising task for instance, the performance of MRF-based systems is still far away from the state-of-the-art methods. Table 2.2 presents the denoising results of state-of-the-art methods: (1) patch-average method, BM3D [36]; (2) non-local sparse coding method, LSSC [91]; (3) expected patch log likelihood method using GMM prior, GMM-EPLL [143]. It is clear that these methods outperform the MRF-based models in Table 2.1.2. This situation challenges the theoretically sound MRF approaches. However, as described below, we can find some clues to improve the MRF model, after having a closer look at the current loss-specific training approaches.

Analysis: The loss-specific training criterion is formally expressed as the following bi-level optimization problem

$$\begin{cases} \arg\min_{\vartheta} L(u^*(\vartheta), g) \\ \text{subject to } u^*(\vartheta) = \arg\min_{u} E(u, f, \vartheta). \end{cases} \tag{2.6}$$

In this model, given the observation f and ground-truth g, our goal is to find the optimal parameters ϑ to minimize the loss function $L(u^*(\vartheta), g)$, which is called the upper-level problem in the bi-level framework. The MRF model is defined by the energy minimization problem $E(u, f, \vartheta)$, which is called the lower-level problem.

The essential point to solve this bi-level optimization problem is to calculate the gradient of the loss function $L(u^*(\vartheta), g)$ with respect to the parameters ϑ. As aforementioned, [115] employs the implicit differentiation technique to calculate the gradients explicitly; in contrast, [40] and [8] make use of an approximation approach based on truncated optimization. All of them use the same ST-distribution as potential function; however, the latter two approaches surprisingly obtain much better performance than the former, as can be seen in Table 2.1.2.

Since Samuel and Tappen use a "full" fitting training scheme, they should achieve better results compared to the approximation approaches, but actually they fail in practice.

Therefore, we argue that there should be something imperfect in their training scheme. We could probably achieve promising improvements by refining this "full" fitting training scheme.

Another instance to prove that the ST-distribution based MRF model should not be so inferior is the work in a recent paper [73], where a Laplace distribution based MRF model is also optimized under the framework bi-level optimization. Even though they only optimize the weights of some fixed filters such as DCT (Discrete cosine transform) filters, their convex ℓ_1 model, which is a worse fitting of heavy-tailed distribution compared to the ST-distribution, has already led to comparable results and their non-convex $\ell_{\frac{1}{2}}$ leads to significantly better results compared to [115].

Motivations: Motivated by the above observations, we think it is necessary and worthwhile to restudy the loss-specific training scheme and we expect that we can achieve significant improvements. In our work, we do not make any modifications to the training model used in [115] - we use exactly the same model capacity, potential function and training images. The only difference is the training algorithm. We exploit a refined training algorithm, where we solve the lower-level problem in the loss-specific training with very high accuracy and make use of a more efficient quasi-Newton's method for model parameters optimization. We conduct a series of playback experiments and show that the performance of loss-specific training is indeed underestimated in previous work [115]. We argue that the the critical reason is that they do not solve the lower-level problem to sufficient accuracy. We also demonstrate that solving the lower-level problem with higher accuracy is indeed beneficial. This argument about the loss-specific training scheme is the major contribution of this section.

In addition, we show that our trained model can obtain further improvement by increasing the model size. It turns out that for image denoising task, our optimized MRF (opt-MRF) model of size 7×7 has achieved the best result among existing MRF-based systems and been on par with state-of-the-art methods. Due to the simplicity of our model, it is easy to implement the inference algorithm on parallel computation units, e.g., GPUs. Numerical results show that our GPU-based implementation can perform image denoising in near real-time with state-of-the-art performance.

2.1.3 Basic training model

Our training model makes use of the bi-level optimization framework, and is based on the image denoising task. For image denoising, the FoE image prior based variational model

is expressed as

$$\arg\min_u E(u) = \sum_{i=1}^{N_k} \alpha_i \sum_{p=1}^{N} \phi((K_i u)_p) + \frac{\lambda}{2}\|u - f\|_2^2. \tag{2.7}$$

where N_k is the number of filters, N is the number of pixels in image u, K_i is an $N \times N$ highly sparse matrix, which makes the convolution of the filter k_i with a two-dimensional image u equivalent to the product result of the matrix K_i with the vectorization form of u, i.e., $k_i * u \Leftrightarrow K_i u$. Moreover, $\alpha_i \geq 0$ is the associated weight for filter K_i, and λ defines the trade-off between the prior term and data fitting term. $\phi(\cdot)$ denotes the penalty function which can be chosen from some candidate functions. Up to now, we do not specify the penalty function. Note that this variational model defines the lower-level problem in the bi-level framework.

The loss function $L(u^*, g)$ (upper-level problem) is defined to measure the difference between the optimal solution of energy function and the ground-truth. In this paper, we make use of the same loss function as in [115],

$$L(u^*, g) = \frac{1}{2}\|u^* - g\|_2^2, \tag{2.8}$$

where g is the ground truth image and u^* is the optimal solution of the variational model (2.7). This loss function is related to the PSNR quality measure. Note that as shown in [68], other quality measures, such as structural similarity (SSIM) and mean absolute error (MAE) can be chosen to define the loss function. We currently only consider the quadratic loss function due to its simplicity. Extension to other loss function is subject to future work.

Given the training samples $\{f_s, g_s\}_{s=1}^S$, where g_s and f_s are the s^{th} clean image and the associated noisy version respectively, our aim is to learn an optimal MRF parameter $\vartheta = (\alpha, k)$ (we group the linear filters k_i and weights α_i into a single vector ϑ), to minimize the overall loss function. Therefore, the learning model is formally formulated as the following bi-level optimization problem

$$\begin{cases} \min\limits_{\alpha \geq 0, k} L(u^*(\alpha, k)) = \sum\limits_{s=1}^{S} \frac{1}{2}\|u_s^*(\alpha, k) - g_s\|_2^2 \\ \text{where } u_s^*(\alpha, k) = \arg\min\limits_u \sum\limits_{i=1}^{N_k} \alpha_i \phi(k_i * u) + \frac{1}{2}\|u - f_s\|_2^2, \end{cases} \tag{2.9}$$

where $\phi(k_i * u) = \sum\limits_{p=1}^{N} \phi((k_i * u)_p)$. We eliminate λ for simplicity, since it can be incorporated

into weights α.

2.2 Link to analysis operator learning problem

In the literature of sparse representation for image processing, a new sparsity based concept has been introduced in recent years, which is so-called co-sparse analysis model (or co-sparsity) [60, 105, 113]. Many algorithms are proposed to train the so-called analysis operator in the co-sparse analysis model. However, we find some interesting relations between the the co-sparse analysis model and the FoE image prior model.

In this section, we give new insights into the co-sparse analysis model from the view of filter-based Markov Random Fields (MRF) models. We hold the opinion that for image processing the cos-parse analysis model is equivalent to the filter-based MRF model, which is also known as Field of Experts (FoE). Therefore, we advocate treating analysis operator learning the same as the filters learning for the FoE model. In this section, we establish the connection between the co-sparse analysis model and the FoE model.

2.2.1 Background of the co-sparse analysis model

2.2.1.1 Related notations

In this section, our model presents a global prior over the entire image instead of small image patches. In order to distinguish between a small patch and an entire image, we represent a square patch (patch size: $\sqrt{m} \times \sqrt{m}$) by $x \in \mathbb{R}^m$, and an image (image size: $M \times N$, with $m \ll M, m \ll N$) by $u \in \mathbb{R}^{MN}$. We denote the patch-based synthesis dictionary and analysis operator by $D \in \mathbb{R}^{m \times n}$ and $A \in \mathbb{R}^{n \times m}$ with $m \leq n$, respectively. Furthermore, when the analysis operator A is applied to the entire image u, we use the common sliding-window fashion to compute the coefficients Ax for all MN patches in the image. This result is equivalent to a multiplication of a highly sparse matrix $\mathcal{A} \in \mathbb{R}^{(n \times MN) \times MN}$ with the image u, i.e., $\mathcal{A}u$. We can group $\mathcal{A}$ to n separable sparse matrices $\{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$, where $\mathcal{A}_i \in \mathbb{R}^{MN \times MN}$ is associated with the i^{th} row of A (A_i). If we consider A_i as a 2-D filter ($\sqrt{m} \times \sqrt{m}$), we have: $\mathcal{A}_i u$ is equivalent to the result of convolving image u with filter A_i.

2.2.1.2 Patch-based synthesis and analysis models

Historically, sparse representations refer to the so-called sparse synthesis model. In the synthesis-based models, a signal $x \in \mathbb{R}^m$ (a patch) is called sparse over a given dictionary

$D \in \mathbb{R}^{m \times n}$ with $m \leq n$, when it can be composed of a linear combination of only a few atoms from dictionary D. This is formulated as the following minimization problem:

$$x^* = D\alpha^*; \alpha^* = \arg\min_{\alpha \in \mathbb{R}^n} \varphi(\alpha) + \frac{\lambda}{2}\|D\alpha - f\|_2^2, \tag{2.10}$$

where $f \in \mathbb{R}^m$ is the observed patch, $\alpha \in \mathbb{R}^n$ is the coefficient vector and φ is the penalty function. In order to induce sparsity in the representation, typical penalty functions include ℓ_p norms with $p \in \{0, 1\}$ or logarithmic functions such as $\log(1 + |z|^p)$, with $p \in \{1, 2\}$. The synthesis model has been intensively studied in the past decade, including global and specialized dictionary learning algorithms and applications to various image processing tasks, see [3, 42, 76, 90, 92] for examples.

However, there is another viewpoint to consider sparse representations, which is the so-called co-sparse analysis model [43]. The objective of a co-sparse model is to pursue a linear operator $A \in \mathbb{R}^{n \times m}$, such that the resulting coefficient vector $Ax \in \mathbb{R}^n$ is expected to be sparse. In the framework of MAP inference, the co-sparse analysis model is given as the following minimization problem:

$$x^* = \arg\min_{x \in \mathbb{R}^m} \varphi(Ax) + \frac{\lambda}{2}\|x - f\|_2^2, \tag{2.11}$$

where A is the so-called analysis operator, φ is again a sparsity promoting function as mentioned above and $f \in \mathbb{R}^m$ is the observed patch. Note that both the analysis model and the synthesis model become equivalent if D is invertible. However, the analysis model is much less investigated compared to the well-known synthesis model, but it has been gaining more and more attention in recent years [60, 105, 113].

2.2.1.3 Patch-based analysis operator learning

In the case of the synthesis model, the learning of an optimized dictionary has become ubiquitous. However, in analysis-based models, fixed operators inspired from variational methods such as the discrete total variation have been used for a long time. It is only recently that people started to develop customized algorithms to learn in some sense optimal analysis operators.

Existing algorithms mainly concentrate on the patch-based training strategy. Given a set of S training samples $Y = [y_1, \ldots, y_S] \in \mathbb{R}^{m \times S}$, where depending of the training procedure, each sample is a noisy version or a clean version of an image patch. For the noisy version, $y_i = x_i + n_i$, where n_i is an additive zero-mean white Gaussian noise vector

and x_i is the clean signal.

For the noise-free training, the goal of the analysis operator learning is to find a linear operator $A \in \mathbb{R}^{n\times m}$ with $m \leq n$, such that the coefficient vector Ay_i is as sparse as possible. This strategy can be formally expressed as the following optimization problem.

$$A^\star = \arg\min_A \varphi(AY). \tag{2.12}$$

For the noise aware case, the objective is to learn an optimal analysis operator A, which enforces the coefficient vector Ax_i to be sparse, while $\|x_i - y_i\|_2^2 \leq \varepsilon$ for each training sample (ε is an error tolerance, which is derived from the noise level). This requires solving a problem of the form

$$\begin{aligned} \{A^\star, X\} &= \arg\min_{A,X} \varphi(AX), \\ &\text{subject to } \|x_i - y_i\|_2^2 \leq \varepsilon. \end{aligned} \tag{2.13}$$

Using a Lagrange multiplier $\lambda > 0$ this can be equivalently expressed as

$$\{A^\star, X\} = \arg\min_{A,X} \varphi(AX) + \frac{\lambda}{2}\|X - Y\|_F^2, \tag{2.14}$$

where φ is again a sparsity promoting function and $\|\cdot\|_F$ denotes the Frobenius norm.

Unfortunately, the above optimization problems suffer from the problem of trivial solutions. Indeed, if no constraints are imposed on A, it is easy to see that the trivial solution $A \equiv 0$ is the global minimizer of (2.12), (2.13) and (2.14). A possible solution to exclude the trivial solution is to impose additional assumptions on A, i.e., restricting the solution set to an admissible set $\mathcal{C}$. The following constraints have been investigated in [137] and [60]:

(i) row norm constraints. All the rows of A have the same norm, i.e., $\|A_i\|_2 = c$ for the i^{th} row of operator A.

(ii) row norm + full rank constraints. The analysis operator A has full rank, i.e., $rk(A) = m$.

(iii) tight frame constraints. The admissible set of this constraint is the set of tight frame in $\mathbb{R}^{n\times m}$, i.e., $A^\top A = \mathbf{I}_m$, where $\mathbf{I}_m$ is the identity operator in $\mathbb{R}^m$.

As pointed out in [137], each individual constraint presented above does not lead to satisfactoy results. Therefore, in [137, 138] a constraint called the Uniform Normalized

Tight Frame (UNTF) was proposed, which is a combination of the unit row norm and the tight frame constraint. The authors of [60] employed a constraint combining the unit row norm and the full rank constraint with an additional consideration that the analysis operator A doesn't have trivially linear dependent rows, i.e., $A_i \neq A_j$ for $i \neq j$.

In [137], Yaghoobi et al. employed the convex ℓ_1-norm, i.e., $\varphi(AY) = \|AY\|_1$, as sparsity promoting function and the UNTF constraint to solve problem (2.12). In [138], the same authors proposed an extension of their previous algorithm that simultaneously learns the analysis operator and denoises the training samples. The improved algorithm solves the problem (2.14) by alternating between updating the analysis operator and denoising the training samples. They gave some preliminary image denoising results by applying the learned operator to natural face images.

In [60], Hawe et al. exploited the above constraints - full rank matrices with normalized rows, and a non-convex sparsity measurement function called the mixed (p, q)-pseudo-norm to minimize problem (2.12). They employed a conjugate gradient method on manifolds to solve this optimization problem. Their experimental results for classical image restoration problems show competitive performance compared to state-of-the-art techniques.

Rubinstein et al. [112] presented an adaption of the widely known K-SVD dictionary learning method [42] to solve the problem (2.13) directly based on the ℓ_0 quasi-norm, i.e., $\varphi(AY) = \|AY\|_0$. Unfortunately, there are only synthetic experiments and examples based on piece-wise constant images considered in their work. The same authors presented some preliminary results for natural image denoising in their later work [113]. However, it turns out that the performance of the learned analysis operator is inferior to the synthesis model [42].

Ophir et al. [101] proposed a simple analysis operator learning algorithm, where analysis "atoms" are learned sequentially by identifying directions that are orthogonal to a subset of the training data.

Apart from the above analysis operator learning algorithms, Peyré and Fadili proposed an attractive learning approach in [105]. They considered the analysis operator from a particular viewpoint. They interpreted the behavior of the analysis operator as a convolution with some finite impulse response filters. Keeping this idea in mind, they formulated the analysis operator learning as a bi-level programming problem [31] which was solved using a gradient descent algorithm. However, their work only considered a simple case - one filter and 1D signals. Following this direction, a preliminary attempt to apply this idea to 2D image processing was done in [26].

2.2.1.4 Comments to existing co-sparse analysis model

Among the existing algorithms for analysis operator learning, only few prior works have been evaluated based on natural images [60, 113, 138]. Moreover, most of these algorithms have to impose a non-convex constraint on the analysis operator A, making the corresponding optimization problems hard to solve. Thus a question arises: Is it possible to introduce a more principled technique to learn optimized analysis operators without the need to impose additional constraints on the operators?

In this thesis, we give an answer to this question. First, we extend the patch-based analysis model to a global image regularization term, which allows to consider also more general inverse problems such as image deconvolution and image inpainting. Then, we show that this model is equivalent to higher-order filter-based MRF models such as the FoE model [111]. Motivated by this observation, we apply a loss-function based training scheme [115] and show that this approach excludes the trivial solution of the analysis operator learning problem without imposing any additional constraints. Furthermore, we carefully investigate the effect of different aspects of the analysis based model. We show that the choice of the sparsity promoting function is the most important aspect.

2.2.2 Insights into analysis based models

In this section, we first show the equivalence between the patch-based analysis model and filter-based probabilistic image patch modeling - Product of Experts (PoE) [63, 135]. Then we extend the patch-based analysis model to the image-based model and show connections to higher order MRFs [111].

2.2.2.1 Equivalence between the patch-based analysis model and the PoE model

The patch-based analysis model in (2.11) focuses on modeling small image patches, which is formulated as a matrix-vector multiplication (Ax). This procedure can be interpreted as projecting a signal x (an image patch) onto a set of linear components $\{A_i\}_{i=1}^n$, where each component A_i is a row of the matrix A. Note that projecting an image patch onto a linear component ($A_i x$) is equivalent to filtering the patch with a linear filter given by A_i.

The PoE model provides a prior distribution on small image patches by taking the product of several expert distributions, where each expert works on a linear filter and the expert function. The PoE model is formally written as $p(x) = \frac{1}{Z(\Theta)}\exp(-E_{PoE}(x, \Theta))$

with

$$E_{\text{PoE}}(x, \Theta) = -\sum_{i=1}^{n} \log\rho_i(A_i x), \tag{2.15}$$

where ρ_i is the potential function, $Z(\Theta)$ is the normalization and Θ are the parameters of this model.

Comparing the analysis prior given in (2.11), $\varphi(Ax) = \sum_{i=1}^{n} \varphi_i(A_i x)$ with the above PoE model, we can see they are actually the same model if we choose the penalty function as $\varphi_i = -\log\rho_i$. In this case, if we consider the analysis operator learning problem based on the strategy which focuses on the modeling of small image patches rather than defining a prior model over an entire image, the learning problem is tantamount to learning filters in the PoE model.

2.2.2.2 From patch-based to image-based model

Patch-based models are only valid for the reconstruction of a single patch. When they are applied to full image recovery, a common strategy is patch averaging [42]. All the patches in the entire image are treated independently, reconstructed individually and then integrated to form the final reconstruction result by averaging the overlapping regions. While this method is simple and intuitive, it clearly ignores the coherence between overlapping patches, and thus misses global support during image reconstruction. To overcome these drawbacks, an extension to the whole image is necessary where patches are not treated independently but each of them is a part of the image.

A promising direction to formulate an image-based model is to make use of the formalism of higher-order MRFs which enforce coherence across patches. The basic idea is to modify the patch-based analysis model in (2.11) such that all possible patches in the entire image and the corresponding coefficient vectors Ax are considered at once. This leads to an image-based prior model of the form:

$$E_{\text{prior}}(u) = \sum_{p=1}^{N} \varphi(AP_p u), \tag{2.16}$$

where u is an image of size $M \times N$, $N = N \times M$, $\varphi(AP_p u) = \sum_{i=1}^{n} \varphi((AP_p u)_i)$ and $P_p \in \mathbb{R}^{m \times N}$ is a sampling matrix extracting the patch at pixel p in image u. For the patches at the image boundaries, we extract patches by using symmetric boundary conditions.

A key characteristic of the model (2.16) is that it explicitly models the overlapping of

image patches, which are highly correlated. Intuitively, it is a better strategy for image modeling compared to the patch averaging approach. In Subsection 3.1.1 we will provide experimental results to support this claim.

2.2.2.3 Equivalence between the image-based analysis model and the FoE model

If we consider in (2.16) each row of A (A_i) as a 2-D filter ($\sqrt{m} \times \sqrt{m}$), we can rewrite this term as

$$E_{\text{prior}}(u) = \sum_{p=1}^{N} \sum_{i=1}^{n} \varphi(\langle A_i, u_p \rangle) = \sum_{i=1}^{n} \sum_{p=1}^{N} \varphi((A_i * u)_p) \,, \tag{2.17}$$

where $\langle A_i, u_p \rangle$ denotes the inner product of the patch at pixel p with filter A_i [*]. Recall that the exploited image regularization model is defined as

$$\mathcal{R}(u) = \sum_{i=1}^{N_k} \sum_{p=1}^{N} \alpha_i \phi_i \left((k_i * u)_p \right) \,. \tag{2.18}$$

Comparing the image-based analysis model and the the exploited image regularization model (e.g., the FoE image prior model), it is easy to see the equivalence of these two models:

- Each row of the analysis operator A corresponds to a linear filter in the FoE model;
- The sparsity promoting function φ corresponds to the penalty function ϕ in the FoE model ;
- The row number of the analysis operator A corresponds to number of filters in the FoE model.

Therefore, it becomes quite clear that the analysis co-sparse model is actually equivalent to the well-known FoE model, and we think it is better to consider the analysis operator learning problem in the framework of FoE model.

In conclusion, the FoE model can be treated as an extension of the co-sparse analysis model from a patch-based formulation to an image-based formulation. It comes along with the advantage of inherently capturing the coherence between overlapping patches which has to be enforced explicitly in patch-based models. As we will see in the next section, the image-based model also allows to learn optimized analysis operators without the need for additional constraints.

[*] It can also be seen as the result of convolving the patch at pixel p with filter A_i.

2.2.3 Loss-specific analysis operator learning model

Given the training samples $\{f_s, g_s\}_{s=1}^{S}$, where g_s and f_s are the s^{th} clean image and the associated noisy version respectively, our aim is to learn an optimal analysis operator or a set of filters which are defined by parameters $\vartheta = (\alpha, k)$ (we group the filters k_i and weights α_i into a single vector ϑ), such that the overall loss function for all samples is as small as possible. Therefore, our learning model is formulated as the following bi-level optimization problem:

$$\begin{cases} \min\limits_{\alpha \geq 0, A} L(u^*(\alpha, A)) = \sum\limits_{s=1}^{S} \frac{1}{2} \|u_s^*(\alpha, A) - g_s\|_2^2 \\ \text{where } u_s^*(\alpha, A) = \arg\min\limits_{u} \sum\limits_{i=1}^{n} \alpha_i \phi(\mathcal{A}_i u) + \frac{1}{2}\|u - f_s\|_2^2. \end{cases} \tag{2.19}$$

We eliminate λ for simplicity since it can be incorporated into the weights α. Our analysis operator training model has two advantages over existing analysis operator learning algorithms.

(a) *It is completely unconstrained with respect to the analysis operator A.* Normally, existing approaches such as [60, 137, 138] have to impose some non-convex constraints over the analysis operator. On the one hand, this makes the corresponding optimization problem difficult to solve, and on the other hand it decreases the probability of learning a meaningful analysis operator, because as indicated in [137], there is no evidence to prove that the introduced constraints are the most suitable choices. The reason why constraints are indispensable for these approaches lies in the need to exclude the trivial solution $A = 0$. However, looking back at our training model, this trivial solution can be avoided naturally. If $A = 0$, the optimal solution of the lower-level problem in (2.21) is certainly $u_s^* = f_s$, which makes the loss function still large; thus this trivial solution is not acceptable in that the goal of our model is to minimize the loss function. Therefore, the optimal operator A must comprise some meaningful filters such that the minimizer of the lower-level problem is close to the ground-truth.

(b) *The learned analysis operator inherently captures the properties of overlapping patches.* In [42, 112, 113, 137], their approaches present a patch-based prior, and thus for global reconstruction of an entire image, the common strategy consists of two stages: (i) extract overlapping patches, reconstruct them individually by synthesis-prior or analysis-prior based model, and (ii) form the entire image by

averaging the final reconstruction results in the overlapping regions. This strategy clearly misses global support during the reconstruction process; however our approach can overcome these drawbacks.

In the work of [60], the authors employ the patch-based model to train the analysis operator, but use it in the manner of an image based model. Clearly, if the final intent is to use the analysis operator in an image-based model, a better strategy is to train it also in the same framework.

2.3 Solving the loss-specific problem

2.3.1 Gradients computation

In this subsection, we consider the bi-level optimization problem from a general point of view. For convenience, we only consider the case of a single training sample and we show how to extend the framework to multiple training samples in the end.

First of all, in our training we consider linear filters constructed from certain basis filters to preserve some special properties, e.g., zero-mean filters. That is to say, we employ some specific basis filters $\{B_1, \cdots, B_J\}$, and the filters to train are expressed as the following linear combination

$$K_i = \sum_{j=1}^{J} \beta_{ij} B_j \,. \tag{2.20}$$

Therefore, the filter K_i is defined by the coefficients β_{ij}. In the following derivation, the FoE model is parameterized by a vector $\vartheta = (\alpha, \beta)$. Recall that the FoE image prior training model using the loss-specific learning scheme is formulated as

$$\begin{cases} \min\limits_{\alpha \geq 0, \beta} L(u^*(\alpha, \beta)) = \sum\limits_{s=1}^{S} \frac{1}{2} \|u_s^*(\alpha, \beta) - g_s\|_2^2 \\ \text{where } u_s^*(\alpha, \beta) = \arg\min\limits_{u} \sum\limits_{i=1}^{N_k} \alpha_i \phi(K_i u) + \frac{1}{2} \|u - f_s\|_2^2, \end{cases} \tag{2.21}$$

According to the optimality condition, the solution of the lower-level problem in (2.21) is given by u^*, such that $\nabla_u E(u^*) = 0$. Therefore, we can rewrite problem (2.21) as

following constrained optimization problem

$$\begin{cases} \min\limits_{\alpha\geq 0,\beta} L(u(\alpha,\beta)) = \frac{1}{2}\|u(\alpha,\beta)-g\|_2^2 \\ \text{subject to } \nabla_u E(u) = \sum\limits_{i=1}^{N_k} \alpha_i K_i^\top \phi'(K_i u) + u - f = 0, \end{cases} \tag{2.22}$$

where $\phi'(K_i u) = (\phi'((K_i u)_1), \cdots, \phi'((K_i u)_p))^\top \in \mathbb{R}^N$. Now we can introduce Lagrange multipliers and study the Lagrange function

$$\mathcal{L}(u,\alpha,\beta,p,\mu) = \frac{1}{2}\|u-g\|_2^2 + \langle -\alpha,\mu\rangle + \langle \sum_{i=1}^{N_k} \alpha_i K_i^\top \phi'(K_i u) + u - f, p\rangle, \tag{2.23}$$

where $\mu \in \mathbb{R}^{N_k}$ and $p \in \mathbb{R}^N$ are the Lagrange multipliers associated to the inequality constraint $\alpha \geq 0$ and the equality constraint in (2.22), respectively. Here $\langle\cdot,\cdot\rangle$ denotes the standard inner product. Taking into account the inequality constraint $\alpha \geq 0$, the first order necessary condition for optimality is given by

$$G(u,\alpha,\beta,p,\mu) = 0, \tag{2.24}$$

where

$$G(u,\alpha,\beta,p,\mu) = \begin{pmatrix} (\sum\limits_{i=1}^{N_k} \alpha_i K_i^\top \mathcal{D}_i K_i + \mathcal{I})p + u - g \\ (\langle K_i^\top \phi'(K_i u), p\rangle)_{N_k\times 1} - \mu \\ (\langle B_j^\top \phi'(K_i u) + K_i^\top \mathcal{D}_i B_j u, p\rangle)_{n\times 1} \\ \sum\limits_{i=1}^{N_k} \alpha_i K_i^\top \phi'(K_i u) + x - f \\ \mu - \max(0, \mu - c\alpha) \end{pmatrix}.$$

Wherein $\mathcal{D}_i(K_i u) = \operatorname{diag}(\phi''((K_i u)_1), \cdots, \phi''((K_i u)_p)) \in \mathbb{R}^{N\times N}$, $(\langle\cdot,p\rangle)_{N\times 1} = (\langle(\cdot)_1,p\rangle, \cdots, \langle(\cdot)_r,p\rangle)^\top$, in the third formulation $n = N_k \times J$. Note that the last formulation is derived from the optimality condition for the inequality constraint $\alpha \geq 0$, which is expressed as $\alpha \geq 0, \mu \geq 0, \langle\alpha,\mu\rangle = 0$. It is easy to check that these three conditions are equivalent to $\mu - \max(0, \mu - c\alpha) = 0$ with c to be any positive scalar and max operates coordinate-wise.

Generally, we can continue to calculate the generalized Jacobian of G, i.e., the Hessian matrix of Lagrange function, with which we can then employ a Newton's method to solve the necessary optimality system (2.24)[73]. However, for this problem calculating the

Jacobian of G is computationally intensive; thus in this paper we do not consider it and only make use of the first derivatives.

Since what we are interested in is the MRF parameters $\vartheta = \{\alpha, \beta\}$, we can reduce unnecessary variables in (2.24). By solving for p and u in (2.24), and substituting them into the second and the third formulation, we arrive at the gradients of loss function with respect to parameters ϑ

$$\begin{cases} \nabla_{\beta_{ij}} L = -(B_j^\top \phi'(K_i u) + K_i^\top \mathcal{D}_i B_j u)^\top (H_E(u))^{-1} (u - g) \\ \nabla_{\alpha_i} L = -(K_i^\top \phi'(K_i u))^\top (H_E(u))^{-1} (u - g) \\ \text{where } \nabla_u E(u) = \sum_{i=1}^{N_k} \alpha_i K_i^\top \phi'(K_i u) + u - f = 0. \end{cases} \tag{2.25}$$

In (2.25), $H_E(u)$ denotes the Hessian matrix of $E(u)$,

$$H_E(u) = \sum_{i=1}^{N_k} \alpha_i K_i^\top \mathcal{D}_i K_i + \mathcal{I}. \tag{2.26}$$

In (2.25), we also eliminate the Lagrange multiplier μ associated to the inequality constraint $\alpha \geq 0$, as we utilize a quasi-Newton's method for optimization, which can easily handle this type of box constraints. We can see that (2.25) is equivalent to the results presented in previous work [115] using implicit differentiation.

Considering the case of S training samples, in fact it turns out that the derivatives of the overall loss function in (2.21) with respect to the parameters ϑ are just the sum of (2.25) over the training dataset.

2.3.2 Bi-level learning algorithm

In (2.25), we have collected all the necessary information to compute the gradients of the loss function with respect to the parameters ϑ, so we can now employ gradient descent based algorithms, e.g., the steepest descent method, for optimization. Although this type of algorithm is very easy to implement, it is not effective. In this thesis, we resort to a more efficient non-linear optimization method - the Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) quasi-Newton's method [86]. We summarize our bi-level learning scheme in Algorithm 1.

In our work, step (ii) in Algorithm 1 is accomplished efficiently by using our proposed iPiano algorithm. We solve this minimization problem to a very high accuracy with $\|\nabla_u E(u^*)\|_2 \leq 10^{-3}$ (gray-values in the range [0 255]), i.e., we use a more conservative

Algorithm 1 Bi-level learning algorithm for analysis operator training

(i) Given training samples $\{f_s, g_s\}_{s=1}^S$, initialization of parameters $\vartheta^0 = \{\alpha^0, \beta^0\}$, let $l = 0$,

(ii) For each training sample, solve for $u_s^*(\vartheta_l)$

$$\sum_{i=1}^{n} \alpha_i^l K_i^\top \phi'(K_i u_s^*) + u_s^* - f_s = 0, \text{where } K_i = \sum_{j=1}^{J} \beta_{ij}^l B_j.$$

(iii) Compute $\nabla_\vartheta L(u^*(\vartheta))$ at ϑ^l via (2.25),

(iv) Update parameters $\vartheta = \{\alpha, \beta\}$ by using a quasi-Newton's method, let $l = l + 1$, and goto (ii).

convergence criterion in this inner loop than previous work [115]. The training algorithm is terminated when the relative change of the loss is less than a tolerance, e.g., $tol = 10^{-5}$, a maximum number of iterations e.g., $maxiter = 500$ is reached or L-BFGS can not find a feasible step to decrease the loss.

2.3.3 The iPiano algorithm to solve the lower level problem

As shown in the aforementioned subsections, we need to solve the lower-level problem in the training. One can see that the corresponding minimization problems poses a generally demanding non-convex optimization problem, which typically takes thousands of iterations with standard gradient descent algorithms, e.g., 5000 iterations in [111]. Obviously, it is not good to employ such a slow algorithm to solve the lower-level problem in the training. Therefore, a computationally efficient algorithm is urgently necessary.

In this subsection, we present our proposed iPiano (Inertial Proximal Algorithm for Non-convex Optimization) algorithm for a class of non-convex problems. The iPiano algorithm is to solve a minimization problem composed of a differentiable (possibly non-convex) and a convex (possibly non-differentiable) function. This algorithm combines forward-backward splitting with an inertial force. As a rigorous convergence analysis of the proposed algorithm is beyond the scope of this thesis, we only present the overall aspects of the iPiano algorithm in this subsection, and the convergence details can be found in the full paper [97].

2.3.3.1 Introduction

The gradient method is certainly one of the most fundamental but also one of the most simple algorithms to solve smooth convex optimization problems. In the last decades, the gradient method has been modified in many ways. One of those improvements is to consider so-called multi-step schemes [94, 108]. It has been shown that such schemes significantly boost the performance of the plain gradient method. Triggered by practical problems in signal processing, image processing and machine learning, there has been an increased interest in so-called composite objective functions, where the objective function is given by the sum of a smooth function and a non-smooth function with an easy to compute proximal map. This initiated the development of the so-called proximal gradient or forward-backward method [84], that combines explicit (forward) gradient steps w.r.t. the smooth part with proximal (backward) steps w.r.t. the non-smooth part.

In our work, we combine the concepts of multi-step schemes and the proximal gradient method to efficiently solve a certain class of *non-convex*, non-smooth optimization problems. Although, the transfer of knowledge from convex optimization to non-convex problems is very challenging, it aspires to find efficient algorithms for certain non-convex problems. Therefore, we consider the subclass of non-convex problems

$$\min_{x \in \mathbb{R}^N} \; F(x) + G(x)\,,$$

where G is a *convex (possibly non-smooth)* and F is a *smooth (possibly non-convex)* function. The sum $F + G$ comprises non-smooth, non-convex functions. Despite the non-convexity, the structure of F being smooth and G being convex makes the forward-backward splitting algorithm well-defined. Inspired by the heavy ball algorithm [141], an inertial force is incorporated into the design of our algorithm, which we termed *iPiano*. Informally, the update scheme of the algorithm that will be analyzed is

$$x^{n+1} = (I + \tau \partial G)^{-1}(x^n - \tau \nabla F(x^n) + \mu(x^n - x^{n-1}))\,,$$

where τ and μ are the step size parameters. The term $x^n - \tau \nabla F(x^n)$ is referred as *forward step*, $\mu(x^n - x^{n-1})$ as *inertial term*, and $(I + \tau \partial G)^{-1}$ as backward or *proximal step*.

Setting $\mu = 0$ results in the forward-backward splitting algorithm, which has the nice property that in each iteration the function value decreases. Our convergence analysis reveals that the additional inertial term prevents our algorithm from monotonically decreasing the function values. Although this may look like a limitation on first glance,

demanding monotonically decreasing function values anyway is too strict as it does not allow for provably optimal schemes. We refer to a statement of Nesterov [94]: "In convex optimization the optimal methods never rely on relaxation. Firstly, for some problem classes this property is too expensive. Secondly, the schemes and efficiency estimates of optimal methods are derived from some global topological properties of convex functions"[†]. The negative side of better efficiency estimates of an algorithm is usually the convergence analysis. This is even true for convex functions. In case of non-convex and non-smooth functions, this problem becomes even more severe.

For $g \equiv 0$ the proximal step is the identity and the update scheme is usually referred as *Heavy-ball method* [141]. This reduced iterative scheme is an explicit finite differences discretization of the so-called *Heavy-ball with friction* dynamical system

$$\ddot{x}(t) + \gamma \dot{x}(t) + \nabla F(x(t)) = 0\,.$$

It arises when Newton's law is applied to a heavy material point subject to a constant friction $\gamma > 0$ (of the velocity $\dot{x}(t)$) and a profile defined by F in the gravity potential. This explains the naming "Heavy-ball method" and the interpretation of $\mu(x^n - x^{n-1})$ as inertial force. A more detailed mechanical interpretation reads as follows.

2.3.3.2 The heavy ball with friction method

When we consider a non-linear oscillator with damping

$$\ddot{x}(t) + \gamma \dot{x}(t) + \nabla F(x(t)) = 0\,, \tag{2.27}$$

where γ is a positive real number ($\gamma > 0$). This system modelizes the motion of a heavy material point $M(t) = (x(t), F(x(t)))$ sliding on a profile defined by F. The damping term $\gamma \dot{x}(t)$ corresponds to a viscous mechanical friction. Because of this mechanical interpretation, this system is referred as the "heavy ball with friction" dynamical system.

Concerning the discretized version of the dynamical system (2.27), it naturally leads to

$$\frac{(x_{t+1} - x_t) - (x_t - x_{t-1})}{2\Delta t} + \gamma \frac{x_{t+1} - x_t}{\Delta t} + \nabla F(x_t) = 0\,,$$

[†]Relaxation is to be interpreted as the property of monotonically decreasing function values in this context. Topological properties should be associated with geometrical properties.

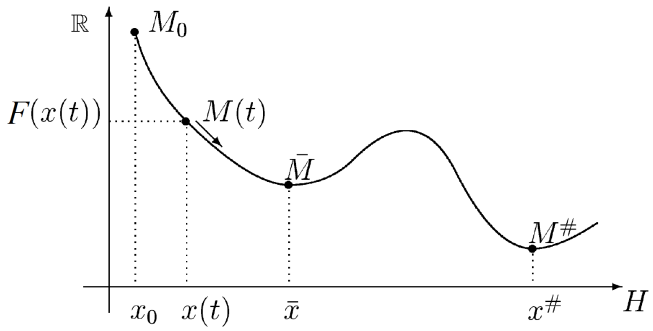


Figure 2.1: An illustrative example of the heavy ball method

with which we then have

$$x_{t+1} = x_t - \tau \nabla F(x_t) + \mu(x_t - x_{t-1}),$$

with $\tau = \frac{2\Delta t}{1+2\gamma}$ and $\mu = \frac{1}{1+2\gamma}$. This will help us to derive a numerical algorithm to compute the local minima of F.

As an illustration, let us consider the following situation (see Figure 2.1) presented in [6] and the corresponding Cauchy problem for (2.27):

$$\begin{cases} \ddot{x}(t) + \gamma \dot{x}(t) + \nabla F(x(t)) = 0 \\ x(0) = x_0, \ \dot{x}(0) = \dot{x}_0 \end{cases} \tag{2.28}$$

For this dynamical system, the stationary state is determined by (1) the initial data, namely the initial position x_0 and the initial velocity $\dot{x}_0$, and (2) the friction parameter γ, which allow us to reach asymptotically several local minima of F. For example, when starting from x_0, one can asymptotically reach $\bar{x}$ or $x^{\#}$, depending on the velocity $\dot{x}_0$. As one can additionally play with the initial velocity $\dot{x}_0$, even when starting from $\bar{x}$ with an initial velocity which is large enough, the material point can escape from the attraction domain of $\bar{x}$, and converge to $x^{\#}$. In contrast, this will never happen for the steepest descent method (also called gradient descent method), which is a first order dynamical system (in time)[‡] and does not contain an acceleration term.

[‡]Note that the heavy ball with friction system is a second order (in time) dissipative dynamical system.

The dynamical system associated with the steepest descent method is given as

$$\begin{cases} \dot{x}(t) + \nabla F(x(t)) = 0 \\ x(0) = x_0 \end{cases} \tag{2.29}$$

The above dynamical system (2.29) modelizes the motion of a drop of water sliding on the profile represented by F. In general, when starting from the initial point x_0, the trajectory asymptotically converges to $\bar{x}$, and it will never have the chance to climb over the "hill" and to reach $x^{\#}$. The dynamical system (2.29) is a first order (in time) system, the trajectory of the point is completely determined by its initial position. However, in the second order system (i.e., the heavy ball with friction system (2.28)), we have additional flexibility and possibility of control, which will help us to overcome spurious stationary points, see the next subsection for an example.

2.3.3.3 The proposed algorithm - iPiano

We consider a structured non-smooth non-convex optimization problem with a proper lower semi-continuous extended valued function $H\colon \mathbb{R}^N \to \mathbb{R} \cup \{+\infty\}$, $N \geq 1$:

$$\min_{x \in \mathbb{R}^N} H(x) = \min_{x \in \mathbb{R}^N} F(x) + G(x)\,, \tag{2.30}$$

which is composed of a C^1-smooth (possibly non-convex) function $F\colon \mathbb{R}^N \to \mathbb{R}$ with L-Lipschitz continuous gradient on $\operatorname{dom} G$, $L > 0$, and a convex (possibly non-smooth) function $G\colon \mathbb{R}^N \to \mathbb{R} \cup \{+\infty\}$. Furthermore, we require H to be coercive, i.e., $\|x\|_2 \to +\infty$ implies $H(x) \to +\infty$, and bounded from below by some value $\underline{H} > -\infty$.

The proposed algorithm seeks for a critical point $x^* \in \operatorname{dom} H$ of H, which is characterized by the necessary first-order optimality condition $0 \in \partial H(x^*)$. In our case, this is equivalent to

$$-\nabla F(x^*) \in \partial G(x^*)\,.$$

In our work, we propose an algorithm, iPiano, with the generic formulation in Algorithm 2. It is a forward-backward splitting algorithm incorporating an inertial force. In the forward step, τ_n determines the step size in the direction of the gradient of the differentiable function F. The step in gradient direction is aggregated with the inertial force from the previous iteration weighted by μ_n. Then, the backward step is the solution of

the proximity operator for the function G with the weight τ_n.

Algorithm 2 inertial **p**rox**i**mal **a**lgorithm for **n**on-convex **o**ptimization (iPiano)

- Initialization: Choose a starting point $x^0 \in \operatorname{dom} h$ and set $x^{-1} = x^0$. Moreover, define sequences of step size parameter $(\tau_n)_{n=0}^{\infty}$ and $(\mu_n)_{n=0}^{\infty}$.
- Iterations ($n \geq 0$): Update

$$x^{n+1} = (I + \tau_n \partial G)^{-1}(x^n - \tau_n \nabla F(x^n) + \mu_n(x^n - x^{n-1}))\,. \tag{2.31}$$

In order to make the algorithm specific and convergent, the step size parameters must be chosen appropriately. There are several strategies to choose appropriate step size parameters. In this section, we only present the backtracking method, which is used to solve the non-convex optimization problems in our work.

2.3.3.4 Backtracking based iPiano

The case where we have only limited knowledge about the objective function occurs more frequently. It can be very challenging to estimate the Lipschitz constant of ∇F beforehand. Using backtracking the Lipschitz constant can be estimated automatically. The Lipschitz constant at iteration n to $n+1$ must satisfy the following inequality

$$F(x^{n+1}) \leq F(x^n) + \left\langle \nabla F(x^n), x^{n+1} - x^n \right\rangle + \frac{L_n}{2}\|x^{n+1} - x^n\|_2^2\,. \tag{2.32}$$

Although, there are different strategies to determine L_n, the most common one is by defining an increment variable $\eta > 1$ and looking for $L_n \in \{L_{n-1}, \eta L_{n-1}, \eta^2 L_{n-1}, \ldots\}$ minimal satisfying (2.32). Sometimes, it is also feasible to decrease the estimated Lipschitz constant after a few iterations. A possible strategy is as follows: if $L_n = L_{n-1}$, then search for the minimal $L_n \in \{\eta^{-1} L_{n-1}, \eta^{-2} L_{n-1}, \ldots\}$ satisfying (2.32).

In Algorithm 3 we propose the iPiano algorithm with variable step sizes, which is employed to solve the non-convex minimization problems in our work. In practice, we make use of the following parameter settings:

$$L_{-1} = 1, \eta = 1.2, \mu = 0.8, \tau = 1.99(1-\mu)/L_n\,.$$

In order to make use of possible larger step sizes in practice, we use a following trick: when the inequality (2.32) is fulfilled, we decrease the evaluated Lipschitz constant L_n slightly

by setting $L_n = L_n/1.05$.

Algorithm 3 non-monotone **i**nertial **p**rox**i**mal **a**lgorithm for **n**on-convex **o**ptimization with backtracking (nmiPiano)

- Initialization: Choose $\beta \in [0, 1)$, $L_{-1} > 0$, $\eta > 1$, and $x^0 \in \operatorname{dom} h$ and set $x^{-1} = x^0$.
- Iterations ($n \geq 0$): Update x^n as follows:

$$x^{n+1} = (I + \tau_n \partial G)^{-1}(x^n - \tau_n \nabla F(x^n) + \mu(x^n - x^{n-1}))\,, \tag{2.33}$$

  where $L_n \in \{L_{n-1}, \eta L_{n-1}, \eta^2 L_{n-1}, \ldots\}$ is minimal satisfying

$$F(x^{n+1}) \leq F(x^n) + \left\langle \nabla F(x^n), x^{n+1} - x^n \right\rangle + \frac{L_n}{2}\|x^{n+1} - x^n\|_2^2 \tag{2.34}$$

  and $\tau_n < 2(1 - \mu)/L_n$.

2.3.3.5 Ability to overcome spurious stationary points

Let us present some of the qualitative properties of the proposed algorithm. For this, we consider to minimize the following simple problem

$$\min_{x \in \mathbb{R}^N} H(x) := F(x) + G(x)\,, \quad F(x) = \frac{1}{2}\sum_{i=1}^{N} \log(1 + \gamma(x_i - u_i^0)^2)\,, \quad G(x) = \lambda\|x\|_1\,, \tag{2.35}$$

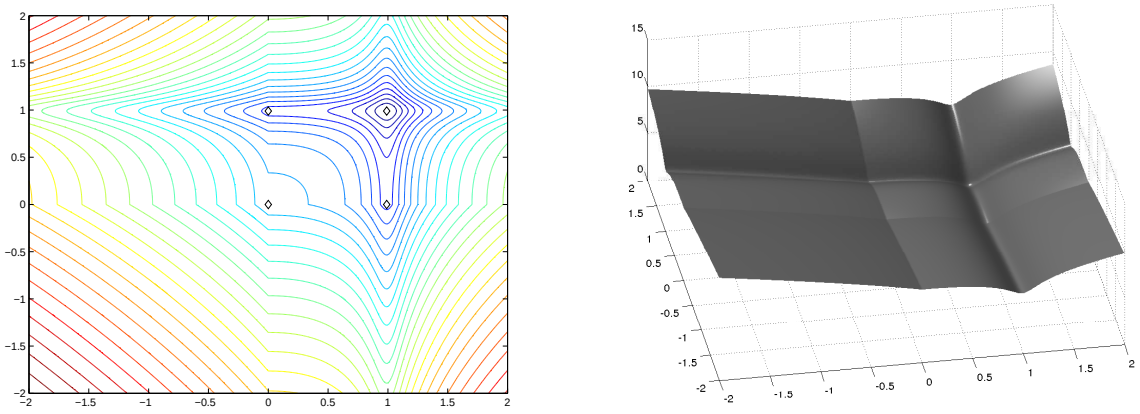
where x is the unknown vector, u^0 is some given vector, and $\lambda, \gamma > 0$ are some free parameters. A contour plot and the energy landscape of H in the case of $N = 2$, $\lambda = 1$, $\gamma = 100$, and $u^0 = (1, 1)^\top$ is depicted in Figure 2.2. It turns out that the function H has four stationary points, i.e. points $\bar{x}$, such that $0 \in \nabla F(\bar{x}) + \partial G(\bar{x})$. These points are marked by small black diamonds.

Clearly the function F is non-convex but has a Lipschitz continuous gradient with components

$$\nabla F(x)_i = \gamma \frac{x_i - u_i^0}{1 + \gamma(x_i - u_i^0)^2}$$

The Lipschitz constant of ∇F is easily computed as $L = \gamma$. The function G is non-smooth but convex and the proximal operator with respect to G is given by the well-known shrinkage operator

$$(I + \tau \partial G)^{-1}(y) = \max(0, |y| - \tau\lambda) \cdot \operatorname{sgn}(y)\,, \tag{2.36}$$



(a) Contour plot of $h(x)$

(b) Energy landscape of $h(x)$

Figure 2.2: Contours plot (left) and energy landscape (right) of the non-convex function h shown in (2.35). The four diamonds mark stationary points of the function h.

where all operations are understood component-wise. Let us test the performance of the proposed algorithm on the example shown in Figure 2.2. We set $\tau = 2(1-\mu)/L$. Figure 2.3 shows the results of using the iPiano algorithm for different settings of the extrapolation factor μ. We observe that iPiano with $\mu = 0$ is strongly attracted by the closest stationary points while switching on the inertial term can help to overcome the spurious stationary points. The reason for this desired property is that while the gradient might vanish at some points, the inertial term $\mu(x^n - x^{n-1})$ is still strong enough to drive the sequence out of the stationary region.

Clearly, there is no guarantee that iPiano always avoids spurious stationary points. iPiano has in general no chance to find the global optimum. However, our numerical experiments suggest that in many cases, iPiano finds lower energies than the respective algorithm without inertial term. A similar observation about the Heavy-ball method is described in [11].

2.4 Refined training scheme

In Section 2.1, we have shown that the performance of loss-specific training scheme has been underestimated in the previous work [115], and it is necessary to revisit it. In this section, we revisit the loss-specific training approach for the FoE image prior model learning problem, and propose a refined training algorithm. With our refined training

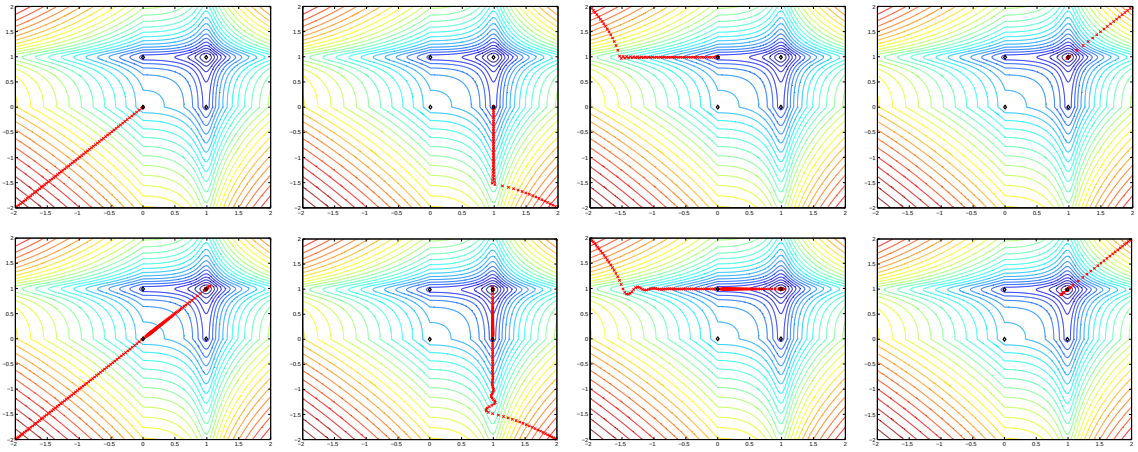
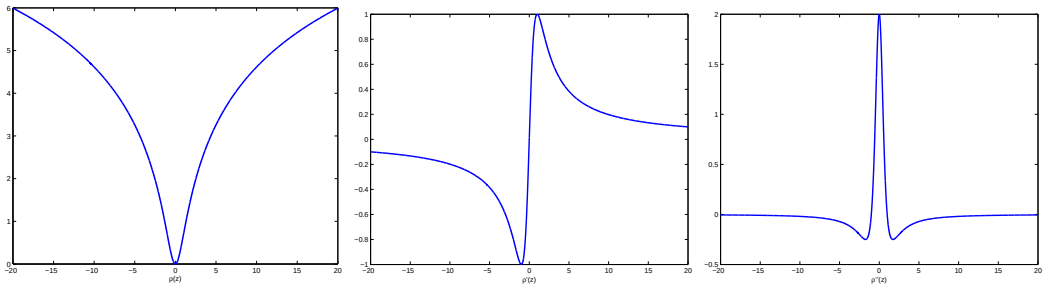


Figure 2.3: The first row shows the result of the iPiano algorithm for four different starting points when using $\mu = 0$, the second row shows the results when using $\mu = 0.75$. While the algorithm without inertial term gets stuck into unwanted local stationary points in three of four cases, the algorithm with inertial term always succeeds to converge to the global optimum.



(a) Penalty function ϕ (b) 1st-order derivative ϕ' (c) 2nd-order derivative ϕ''

Figure 2.4: The Lorentzian penalty function $\phi(z) = \log(1 + z^2)$ and its derivatives.

algorithm, we achieve a FoE prior model, which can significantly boost the performance of the variational model.

As it is a revising study, we conducted a playback experiment using the same training configurations, i.e.,

(1) The same penalty function $\phi(z) = \log(1 + z^2)$, as shown in Figure 2.4;

(2) The same model capacity - 24 filters of size 5×5;

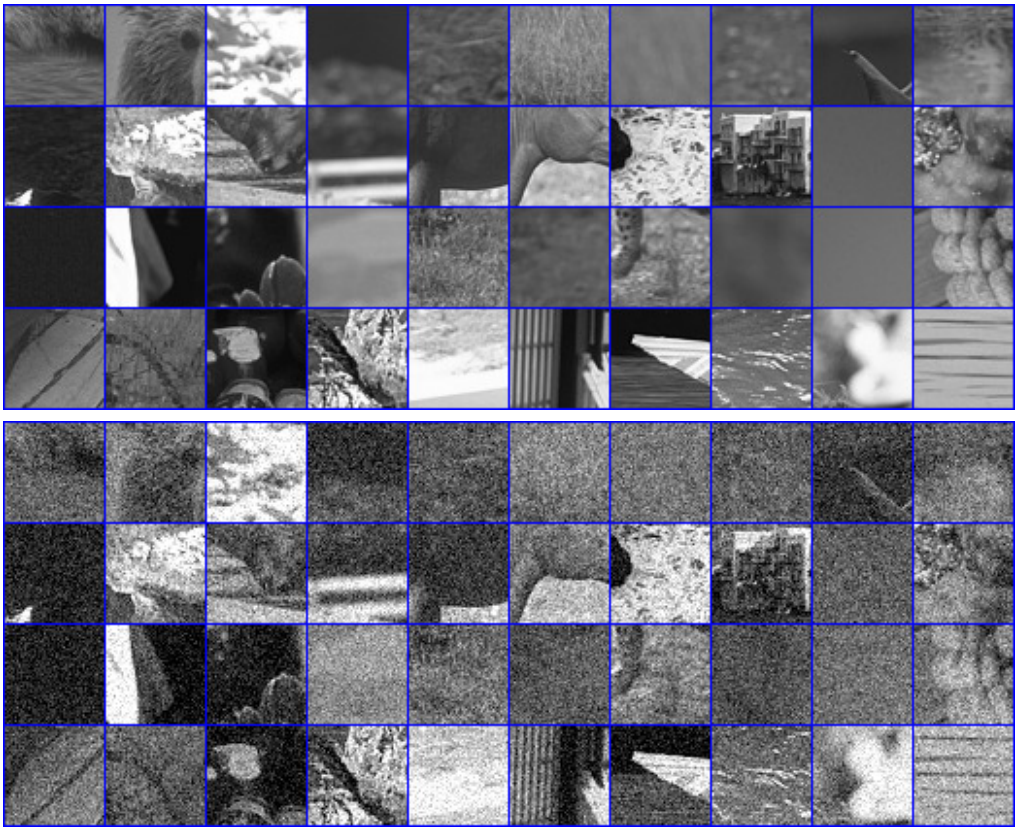


Figure 2.5: Subset of the ground truth and the noisy data with noise level $\sigma = 25$.

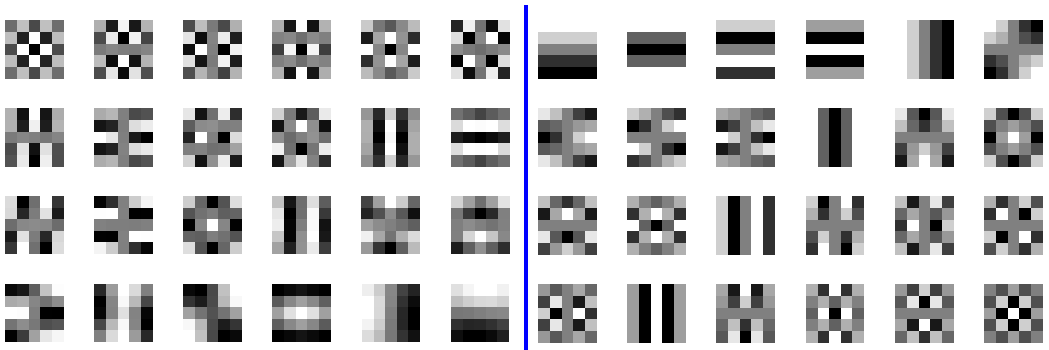
(3) The same basis filters - "inverse" whitened PCA filters [§], shown in Figure 2.6(a);

(3) The same 40 images for training and 68 images for performance evaluation.

We randomly sampled four 51×51 patches from each training image, resulting in a total of 160 training samples. We then generated the noisy versions by adding Gaussian noise with standard deviation $\sigma = 25$. Figure 2.5 shows an exemplary subset of the training data together with the noisy version.

The major difference between our training experiment and previous one is the training algorithm. In our refined training scheme, we employed (1) our proposed iPiano algorithm to solve the lower-level problem with very high accuracy, and (2) L-BFGS to optimize the model parameters, but in contrast, Samuel and Tappen used non-linear conjugate gradient and plain gradient descent algorithm, respectively. In our refined training algorithm, we used the normalized norm of the gradient, i.e., $\frac{\|\nabla_x E(x^*)\|_2}{\sqrt{N}} \leq \varepsilon_l$ (N is the pixel number of

[§]The PCA filters are generated from natural image patches of size 5×5.



(a) Reduced $\text{PCA}_{5\times5}$ filters (b) Reduced $\text{DCT}_{5\times5}$ filters

Figure 2.6: Two exploited basis filters.

the training patch) as the stopping criterion for solving the lower-level problem. In our training experiment, we set $\varepsilon_l = 10^{-5}$ (gray-value in range [0 255]), which implies a very accurate solution.

Based on this training configuration, we learned 24 filters of size 5×5, then we applied them to the task of image denoising to estimate the inference performance using the same 68 test images. Finally, we achieved an average PSNR value of 28.51dB for noise level $\sigma = 25$, which is significantly superior to previous result of 27.86dB in [115].

We believe that the major reason lies in our refined training algorithm where we solve the lower-level problem with very high accuracy.

To make this argument more clear, we need to eliminate the possibility of training dataset, because we did not exploit exactly the same training dataset as previous work (unfortunately we do not have their dataset at hand). Since the training patches were randomly selected, we could run the training experiment multiple times by using different training dataset. Finally, we found that the deviation of test PSNR values based on 68 test images is within 0.02dB, which is negligible. Therefore, it is clear that training dataset is not the reason for this improvement, and the only remaining reason is our refined training scheme.

The influence of ε_l: To investigate the influence of the solution accuracy of the lower-level problem ε_l more detailedly, we conducted a series of training and testing experiments by setting ε_l to different magnitudes. Based on a fixed training dataset (160 patches of size 51×51) and 68 test images, we got the performance curves with respect to the solution accuracy ε_l, as shown in Figure 2.7. From Figure 2.7, we can clearly see that it is indeed

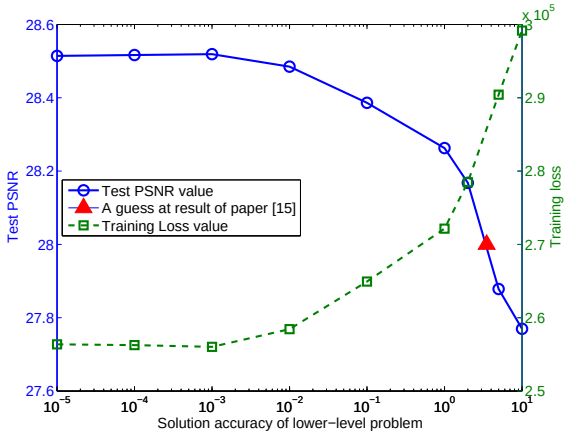


Figure 2.7: Performance curves (test PSNR value and training loss value) *vs.* the solution accuracy of the lower-level problem ε_l. It is clear that solving the lower-level problem with higher accuracy is beneficial.

the high solution accuracy that helps us to achieve the above significant improvement.

Our finding suggests that solving the lower-level problem with higher accuracy pays off. This finding is the main contribution of our refined loss-specific training algorithm. We also make a guess how accurate Samuel and Tappen solve the lower-level problem according to their result and our performance curve, which is marked by a red triangle in Figure 2.7. The argument that higher solution accuracy of the lower-level problem is helpful is explicable, the reason reads as follows.

The key aspect of our approach is to calculate the gradients of the loss function with respect to the parameters ϑ. According to (2.25), there is a precondition to obtain accurate gradients: both the lower-level problem and the inverse matrix of Hessian matrix H_E must be solved with high accuracy, i.e., we need to calculate a u^* such that $\nabla_u E(u^*) = 0$ and compute $(H_E)^{-1}$ explicitly. Since the Hessian matrix H_E is highly sparse, we can solve the linear system $H_E u = b$ efficiently with very high accuracy (we use the "backslash" operator in Matlab). However, for the lower-level problem, in practice we can only solve it to finite accuracy by using certain algorithms, i.e., $\frac{\|\nabla_u E(u^*)\|_2}{\sqrt{N}} \leq \varepsilon_l$. If the lower-level problem is not solved to sufficient accuracy, the gradients $\nabla_\vartheta L$ are certainly inaccurate which will probably affect the training performance. This has been demonstrated in our experiments.

Therefore, for the bi-level training framework, it is necessary to solve the lower-level problem as accurately as possible, e.g., in our training we solved it to a very high accuracy

with $\varepsilon_l = 10^{-5}$, and experimental results demonstrate that it pays off.

The influence of basis: In our playback experiments, we used the "inverse" whitened PCA basis to keep consistent with previous work. However, we argue that the DCT basis is a better choice, because meaningful filters should be mean-zero according to the findings in [65], which is guaranteed by DCT basis without the constant basis vector, shown in Figure 2.6(b). Therefore, we will exploit the DCT filters excluding the filter with uniform entries from now on. Using this modified DCT basis, we retrained our model and we got a test PSNR result of 28.54dB.

The influence of training dataset: To verify whether larger training dataset is beneficial, we retrained our model by using (1) 200 samples of size 64×64 and (2) 200 samples of size 100×100, which is about two times and four times larger than our previous dataset, respectively. Finally, we got a test PSNR result of 28.56dB for both cases. As shown before, the influence of training dataset is marginal.

Based on 200 samples of size 64×64, the trained filters together with the associated weights are shown in Figure 2.8, with which we can achieve an average PSNR result of 28.56dB over the test data set of 68 natural images.

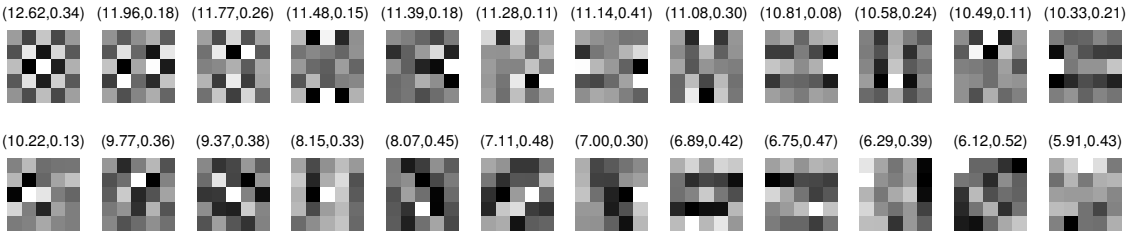


Figure 2.8: 24 learned filters (5×5). The first number in the bracket is the weight α_i and the second one is the norm of the filter.

2.5 More training experiments

Employing our refined loss-specific training scheme, we can train the FoE image prior model with different configurations to investigate its properties. We conducted our training experiments using the training images from the BSDS300 image segmentation database [4]. We used the whole 200 training images, and randomly sampled one 64×64 patch from each training image, resulting in a total of 200 training samples. We then generated the noisy versions by adding Gaussian noise with standard deviation $\sigma = 25$.

In order to evaluate the performance of the learned analysis operators, we applied them to image denoising experiments over a validation dataset consisting of 68 images from Berkeley database [4]. This is a common denoising test dataset for natural images, which was selected by Roth and Black [111]. The performance of an image denoising algorithm varies greatly for different image contents. We therefore consider the average performance over the whole test dataset as performance measurement.

2.5.1 Penalty functions

In order to investigate the importance of the penalty function, in this thesis, we consider three penalty functions with different properties: (1) the ℓ_1 norm, which is a well-known convex sparsity promoting function and has been successfully applied to a number of problems in image restoration [41], (2) the log-sum penalty suggested in [20], $\log(1+|z|)$, which is a non-convex function and can enhance sparsity and (3) the smooth non-convex function $\log(1+z^2)$, which is derived from the student-t distribution and has been employed as the penalty function for sparse representation [125], as well as in the original work of the FoE model [111].

As our training model needs differentiable penalty functions, we have to use a small parameter ε to regularize the absolute function $|z|$. The penalty functions and their associated derivatives are given by

$$\begin{cases} \phi(z) = \sqrt{z^2+\varepsilon^2} \\ \phi'(z) = z/\sqrt{z^2+\varepsilon^2} \\ \phi''(z) = \varepsilon^2/(z^2+\varepsilon^2)^{3/2} \end{cases} \qquad \begin{cases} \phi(z) = \log(1+z^2) \\ \phi'(z) = 2z/(1+z^2) \\ \phi''(z) = 2(1-z^2)/(1+z^2)^2, \end{cases} \tag{2.37}$$

and

$$\begin{cases} \phi(z) = \log(1-\varepsilon+\sqrt{z^2+\varepsilon^2}) \\ \phi'(z) = \frac{z}{\sqrt{z^2+\varepsilon^2}(1-\varepsilon+\sqrt{z^2+\varepsilon^2})} \\ \phi''(z) = \frac{\varepsilon^2(1-\varepsilon)+(\varepsilon^2-z^2)\sqrt{z^2+\varepsilon^2}}{(z^2+\varepsilon^2)^{3/2}(1-\varepsilon+\sqrt{z^2+\varepsilon^2})^2}. \end{cases}$$

2.5.2 Training experiments

First of all, we focused training on filters of dimension 7×7, since our approach allowed us to train larger filters than those trained in [52, 119], and normally larger filters can involve more information of the neighborhood.

We started with a preliminary training experiment based on the penalty function

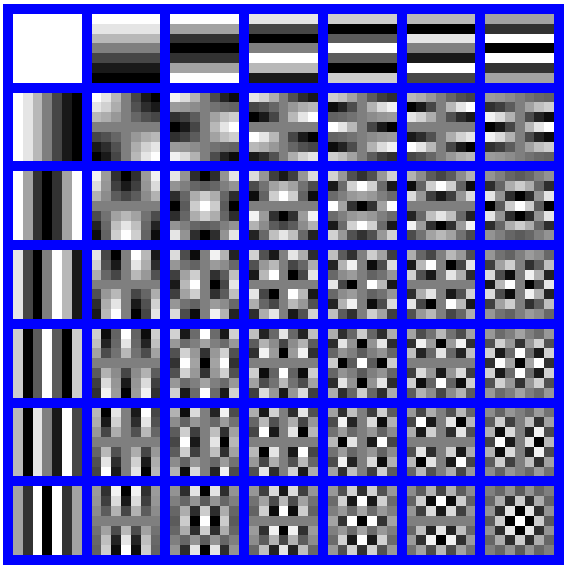


Figure 2.9: The $\mathrm{DCT}_{7\times 7}$ basis

$\log(1+z^2)$. We intended to learn an analysis operator $A \in \mathbb{R}^{48\times 49}$, i.e, 48 filters with dimension 7×7, and each filter is expressed as a linear combination of the DCT-7 basis, which is shown in Figure 2.9. In principle, we can use any basis such as the identity [¶], PCA or ICA basis; however as described below, we need zero-mean filters, which is guaranteed by the DCT filters after excluding the filter with constant entries.

For the preliminary experiment, we initialized the analysis operator using 48 random filters having unified norms and weights, which are 0.01 and 1, respectively. Finally training result shows that all the coefficients with respect to the first atom of DCT-7 (an atom with constant entries) are approximately equal to zero, implying that the first atom isn't necessary to construct the filters. Therefore the learned filters are undoubtedly zero-mean because all the remaining atoms are zero-mean; this makes the analysis prior based model Equation (2.7) invariant to constant functions. This result is coherent with the findings in the work [65] that meaningful filters should be zero-mean. Hence we explicitly exclude the first atom in DCT-7 to speed up the training process for the sequent experiments.

We then conducted training experiments based on three different penalty functions. In this paper, the regularization parameter ε in (2.37) was set to $\varepsilon = 10^{-2}$. Smaller ε implies a better fitting to the absolute function, but it makes the lower-level problem harder to

[¶]The identity basis contains filters formulated by unit vectors like $(1,0,0,\cdots),(0,1,0,\cdots),\cdots$

Final training results and the average denoising PSNR results on 68 test images

$\phi(z)$	$\|z\|$	$\log(1+\|z\|)$	$\log(1+z^2)$	$\log(1+z^2)$	$\log(1+z^2)$	$\log(1+z^2)$	$\log(1+z^2)$	$\log(1+z^2)$
fsz	7×7	7×7	7×7	3×3	5×5	7×7	9×9	direct DCT-7
N_k	48	48	48	8	24	98	80	48
$Train$	440,350	389,860	388,053	437,788	396,250	386,270	384,788	407,556
$Test$	28.04	28.64	28.66	28.13	28.56	28.68	28.70	28.47

Table 2.3: Summary of the final training loss values for different model capacities and the corresponding average denoising PSNR results based on 68 test images with $\sigma = 25$ Gaussian noise. In the table, fsz denotes the filter size, N_k is the number of filters, $Train$ means the final loss value in the training and $Test$ signifies the average PSNR value in the test.

solve and the training algorithm fail.

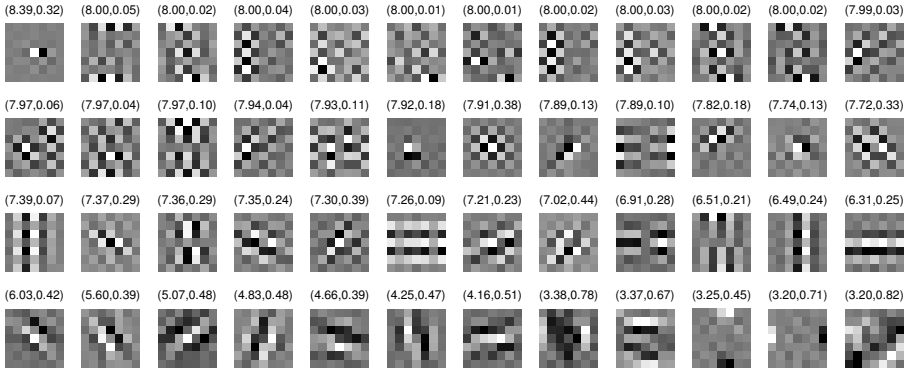
As in the preliminary experiment, we also learned 48 filters. We initialized the filters using the reduced DCT-7 basis with unified norms and weights. The optimal analysis operators learned by using three different penalty functions are shown in Figure 2.10. The final loss function values (normalized by the number of training images) of these three experiments are presented in Table 2.3 (first three columns), together with the average denoising PSNR results based on 68 test images with $\sigma = 25$ Gaussian noise.

As shown in Figure 2.10, the learned filters present some special structures. We can find high-frequency filters as well as derivative filters including the first derivatives along different directions, the second and higher-order derivatives. These filters make the analysis prior based model (2.7) a higher-order model which is able to capture the structures in natural images that cannot be captured by using only the first derivatives as in the total variation based methods.

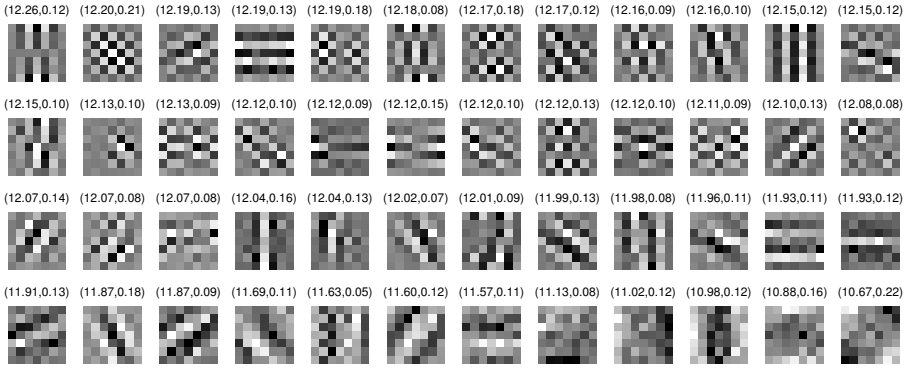
In our training model, the size and the number of filters are free parameters; thus we can train filters of various sizes and numbers. Our current implementation is an unoptimized Matlab code. The training time for 48 filters of size 7×7 was approximately 24 hours on a server (Intel X5675, 3.07GHz), 98 filters of size 7×7 took about 80 hours. However, the training time for larger filter size 9×9 was much longer; it took about 20 days. Fortunately, the training procedure is off-line; thus the training time does not matter too much in practice.

2.5.3 The influence of the penalty function

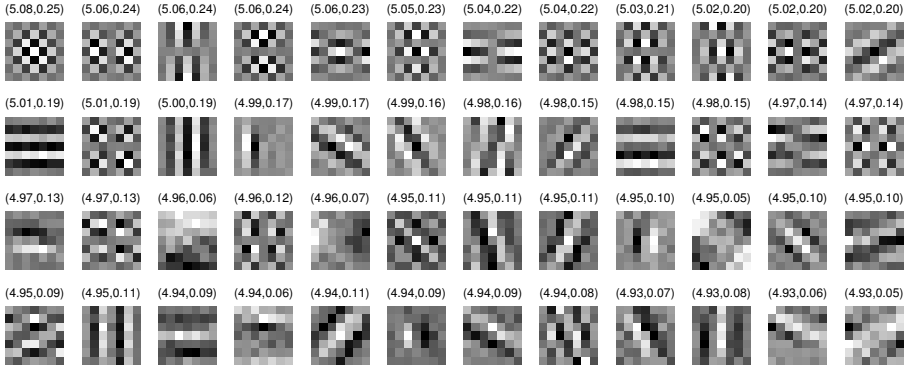
From Table 2.3, we can see that the results obtained by two non-convex penalty functions, $\log(1+|z|)$ and $\log(1+z^2)$ are very similar; however, there is a great improvement compared



(a) filters learned by using the penalty function $\log(1 + z^2)$



(b) filters learned by using the penalty function $\log(1 + |z|)$



(c) filters learned by using the penalty function $|z|$

Figure 2.10: 48 filters of size 7×7 learned by using different penalty functions. Each filter is shown with the corresponding norm and weight. The first number in the bracket is the weight α_i and the second one is the norm of the filter.

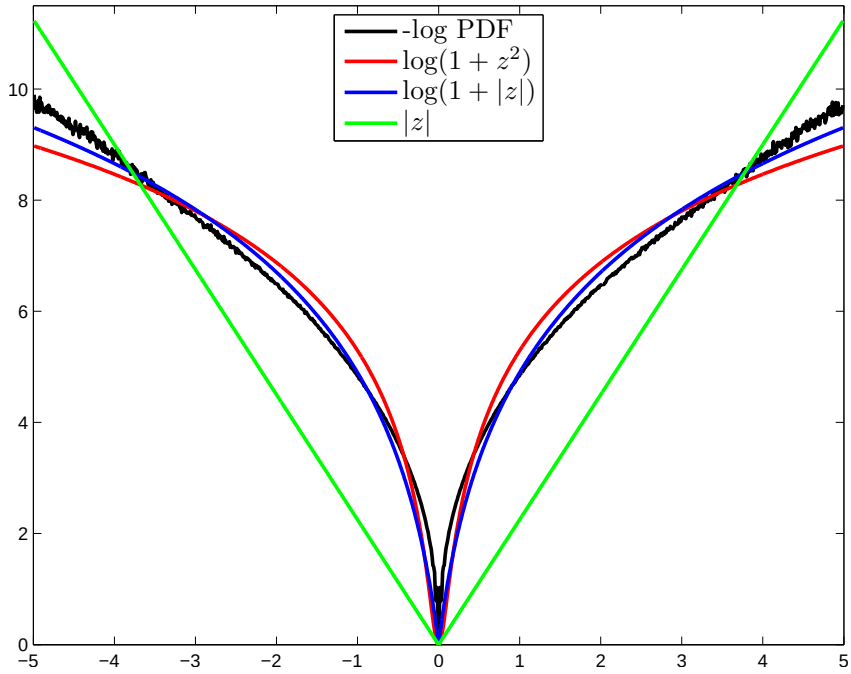


Figure 2.11: Negative log probability density function (PDF) of the filter response of a learned 7×7 filter applied to natural images. Note that non-convex functions $\log(1 + |z|)$ and $\log(1 + z^2)$ provide much better fits to the heavy tailed shape of the true density function, compared to the convex function $|z|$.

to the convex ℓ_1 penalty function. The reason is as follows.

It is well known that the probability density function (PDF) of the responses of zero-mean linear filters (e.g., DCT filters) applied to natural images exhibit heavy tailed distributions [65]. Figure 2.11 shows the negative log PDF of the first filter in Figure 2.10(a) applied to natural images together with different model fits. We can clearly see that two non-convex functions, $\log(1+|z|)$ and $\log(1+z^2)$ both provide an almost perfect fit to the heavy tailed shape of the true density function. The convex function $|z|$ presents a much worse fitting, however. Therefore, a suitable penalty function is crucial for the analysis-prior based model. In general, in order to model the heavy tailed shape of the true PDF, a non-convex function is required.

In order to further investigate how important the non-convex penalty function is for the analysis prior based model, we considered an analysis model consisting of 48 fixed and predefined filters (DCT-7 filters excluding the filter with uniform entries) and making use of the $\log(1 + z^2)$ penalty function. We only optimized the norm and weight of each filter

by using our bi-level training algorithm. The training loss value and the denoising test result of this model are shown in Table 2.3 (the sixth column entitled "direct DCT-7"). The image denoising test result is surprisingly good, even though this analysis model only utilizes a predefined analysis operator DCT-7. We will see in Table 3.1 of Section 3.1 that the performance of this model is already on par with the currently best analysis operator learning model - GOAL [60], which involves much more carefully trained filters. We believe that the superiority of our model lies in the non-convex penalty function $\log(1 + z^2)$.

2.5.4 The influence of the number of filters

In previous work of analysis operator learning [60, 112, 137], the authors were interested in the over-complete case, where the number of filters is larger than the dimension of filters. Clearly our learned analysis operator in Figure 2.10 is under-complete. In order to investigate the influence of the over-complete property, we also conducted a training experiment for the over-complete case ($A \in \mathbb{R}^{98\times 49}$) based on the penalty function $\log(1 + z^2)$. We initialized the analysis operator A using 98 random zero-mean filters. The performance of this over-complete case is presented in Table 2.3.

From Table 2.3, one can see that the improvement achieved by over-complete analysis operator is marginal. Therefore for the analysis model, under-complete operators already work sufficiently well. An increase of the number of filters can not bring significant improvements, while it will clearly increase the training time and inference time.

2.5.5 The influence of filter size

Intuitively the size of filters should be an important factor for the analysis model. In order to investigate the influence of filter size, we conducted training experiments for several different analysis models, where the filter size varies from 3×3 to 9×9. The training and evaluation results of these models are presented in Table 2.3 and Figure 2.12.

One can see that increasing the filter size yields some improvements. However, the performance is close to saturation when the filter size is increasing to 9×9. The improvement brought by increasing the filter size to 9×9 is negligible. This implies that we can not expect large improvements by increasing the filter size to 11×11 or even larger.

2.5.6 The robustness of our training scheme

As our training model (2.21) is a non-convex optimization problem, we can only find stationary points. Thus, a natural question about the initialization arises. We did have

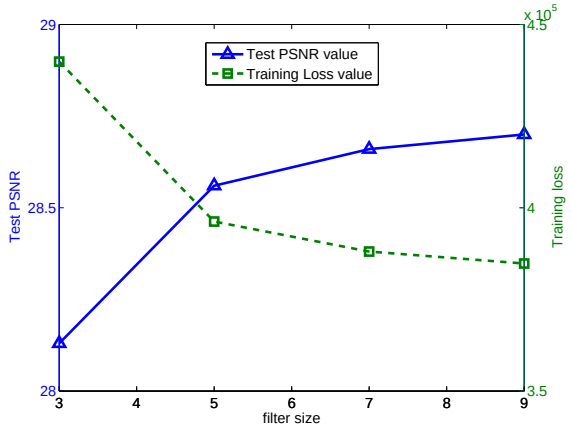


Figure 2.12: Performance curves (test PSNR value and training loss value) *vs.* the filter size. One can see that generally larger filter size can yield some improvements, but the performance is close to saturation when the filter size is increasing to 9×9.

experiments for different initializations, such as random initialization. The final learned analysis operators are surely different, but all of them corresponds to almost the same training loss function, which is the goal of our optimization problem. In addition, these operators perform similarly in the evaluation experiments.

Another issue about the robustness of our training scheme is the influence of the training dataset. Since the training patches were randomly selected, we could run the training experiment multiple times by using a different training dataset. Finally, we only found a negligible difference between the test PSNR values based on 68 test images.

In summary, even though our training is built on a relative small training data set (200 samples of size 64×64, roughly 0.8 million pixels), the trained analysis prior model (i.e., FoE image prior model or image regularizer) works quite well and it is reliable.

2.6 Discussion

In this chapter, we revisited the loss-specific training approach for the FoE image prior model. This training strategy is defined as a bi-level optimization problem, and was first proposed by Samuel and Tappen in [115]. We were motivated to revisit it by using a refined training algorithm, where we solved the lower-level problem with higher accuracy. It turns out that we have achieved significant improvements relative to the previous work in [115], i.e., the performance of the loss-specific training was indeed undervalued in previous work. We argued that the major reason lies in the solution accuracy of the lower-level problem

in the bi-level framework, and we have demonstrated that solving the lower-level problem with higher accuracy is beneficial.

We also have built the link between the investigated FoE image prior model and the recently proposed analysis operator learning model. We expressed our insights into the analysis prior model. We proposed to go beyond existing patch-based models, and to exploit the framework of the FoE model to define a image prior over the entire image, rather than image patches. Then we pointed out that the image based analysis model is equivalent to the FoE model. Based on this conclusion, we have introduced a bi-level training approach (i.e., loss-specific training) for analysis operator learning. By using our training framework, we have carefully investigated the effect of different aspects of the analysis prior model including the filter size, the number of filters and the penalty function. Numerical results have demonstrated that the penalty function is the most important factor for this model.

In order to efficiently solve the lower-level problem, which is a non-convex optimization problem, in the bi-level learning framework, we have proposed a fast non-convex optimization algorithm called iPiano. The iPiano algorithm is applicable for a class of non-convex problems, which are composed of a differentiable (possibly non-convex) and a convex (possibly non-differentiable) function. The algorithm combines forward-backward splitting with an inertial force called heavy ball method.

For future work, focusing on generic prior of natural images, we expect that our learned analysis model could be improved potentially in two aspects: (1) consider more flexible penalty function. In our current model, the penalty function is fixed the same for every filter. If we free the shape of the penalty function, our model will possess more freedom for optimization, which might increase the performance. A feasible way to consider alterable penalty function is to make use of GSMs prior [119]. (2) make use of larger training dataset. Our training is conducted based on 200 training samples, which is only a very small part of the natural images. Consequently, the learned filters may get over-fitting to this training dataset. However, our current training scheme is not available for large training dataset, e.g., $\sim 10^6$, because it needs to solve the lower-level problem for each training sample. Feasible methods may include making use of stochastic optimization.

In the next chapter, we will apply the learned FoE image regularizations to various image restoration problems, beyond the task of Gaussian denoising, to evaluate the performance of the learned image priors.

Chapter 3

Applications of the trained image regularizers

Contents

In the last chapter, we presented our refined loss-specific training scheme to learn an image regularizer (i.e., an image prior model). An important question for a learned prior model is how well it generalizes. To evaluate this, we directly applied the learned image regularizers trained based on image Gaussian denoising task to various image restoration problems such as image deconvolution, inpainting, super-resolution, JPEG artifacts suppression, as well as denoising task.

To start with, we first express the image restoration model by using the learned image regularizer, which is formulated as:

$$u^* = \arg\min_u E(u) = \sum_{i=1}^{N_k} \alpha_i \phi(K_i u) + D(u, f)\,, \tag{3.1}$$

where the first term is the learned image regularizer, $D(u, f)$ denotes the data fidelity term, which varies for different image restoration problems.

This chapter is dedicated to various representative image restoration applications by using the above variational model, involving our learned image regularizers.

3.1 Image denoising

Image denoising is an important pre-processing step for many vision applications. In this section, we consider the image denoising task for different noise types, including additive white Gaussian noise, impulse noise, and multiplicative noise (speckle noise) by using our trained variational models. In order to evaluate the performance of the learned models, we also comprehensively compare our results to recent state-or-the-art image denoising algorithms, in terms of the denoising performance and the run time.

3.1.1 Gaussian noise reduction

Gaussian denoising aims to reconstruct the underlying clean image u from its noisy observation $f = u + n$, where n is assumed to be additive white Gaussian noise with zero mean and variance σ^2. The variational model with the learned image regularizer for the Gaussian denoising problem is formulated as

$$u^* = \arg\min_u E(u) = \sum_{i=1}^{N_k} \alpha_i \phi(K_i u) + \frac{\lambda}{2}\|u - f\|_2^2 , \tag{3.2}$$

where λ is a free parameter to be tuned for different noise levels.

3.1.1.1 Solving the corresponding minimization problems

A first question about the variational model Equation 3.2 is how to solve it efficiently.

Variational model with convex penalty function $|z|$: For the case of convex penalty function $\phi(z) = |z|$, we make use of the first-order primal-dual algorithm proposed in [22] with the preconditioning technique described in [107]. In this case, the problem Equation 3.2 can be rewritten as

$$\arg\min_u \|Ku\|_1 + \frac{\lambda}{2}\|u - f\|_2^2 , \tag{3.3}$$

where the matrix K is stacked in the form

$$K = \begin{bmatrix} \alpha_1 K_1 \\ \alpha_2 K_2 \\ \vdots \\ \alpha_{N_k} K_{N_k} \end{bmatrix} .$$

The saddle-point formulation of the problem 3.3 is given as

$$\min_u \max_p \langle Ku, p\rangle + \frac{\lambda}{2}\|u - f\|_2^2 - \delta_P(p)\,, \tag{3.4}$$

where $\delta_P(p)$ is the convex conjugate of the function $\|z\|$, which is given by the indicator function over the the convex interval $P = [-1, 1]$, namely, the indicator function $\delta_P(p)$ is defined as

$$\delta_P(p) = \begin{cases} 0 & \text{if } |p_i| \leq 1\,, \\ +\infty & \text{else where}\,. \end{cases} \tag{3.5}$$

Now it is straightforward to exploit the primal-dual algorithm to solve the saddle-point problem 3.4.

In order to apply the primal-dual algorithm, we need to solve two proximal mapping subproblems $(\mathbf{I} + \mu\partial F^*)^{-1}$ and $(\mathbf{I} + \tau\partial G)^{-1}$. By casting Equation 3.4 in the form of the general saddle-point problem, we see that $F^*(p) = \delta_P(p)$ and $G(u) = \frac{\lambda}{2}\|u - f\|_2^2$. Since F^* is the indicator function of a convex set, the proximal mapping operator reduces to point-wise projection onto the interval $P = [-1, 1]$

$$p = (\mathbf{I} + \mu\partial F^*)^{-1}(\hat{p}) = \arg\min_p \frac{\|p - \hat{p}\|_2^2}{2\mu} + \delta_P(p) \Longleftrightarrow p_q = \max(-1, \min(1, \hat{p}_q))\,. \tag{3.6}$$

The proximal mapping operator with respect to G poses simple point-wise quadratic problems. The solution is trivially given by

$$u = (\mathbf{I} + \tau\partial G)^{-1}(\hat{u}) = \arg\min_u \frac{\|u - \hat{u}\|_2^2}{2\tau} + \frac{\lambda}{2}\|u - f\|_2^2 \Longleftrightarrow u_q = \frac{\hat{u}_q + \tau\lambda f_q}{1 + \tau\lambda}\,. \tag{3.7}$$

Variational models with non-convex penalty functions: For the case of non-convex penalty functions, e.g., $\phi(z) = |z|^{\frac{1}{2}}$, $\phi(z) = \log(1 + |z|)$ and $\phi(z) = \log(1 + z^2)$, we make use of our proposed iPiano algorithm for optimization*.

As the iPiano algorithm is only applicable for smooth problems, we need to smooth the first two penalty functions with a small parameter ε. The smoothed version is given as $\phi(z) = (z^2 + \varepsilon^2)^{\frac{1}{4}}$ and $\phi(z) = \log(1 - \varepsilon + \sqrt{z^2 + \varepsilon^2})$, respectively.

Casting the problem 3.2 in the general form of the iPiano algorithm, we see that $F(u) = \sum_{i=1}^{N_k} \alpha_i \phi(K_i u)$ and $G(u) = \frac{\lambda}{2}\|u - f\|_2^2$. In order to make use of iPiano algorithm,

*Note that, for the smooth problem, e.g., $\phi(z) = \log(1 + z^2)$ regularized model, we find that the general optimization algorithm - LBFGS also works quite well. In addition, all the minimization problems, including non-smooth cases, can be solved by using a very recently proposed iteratively reweighted convex algorithm [99].

we need to calculate $\nabla_u F(u)$, which is given by

$$\nabla_u F(u) = \sum_{i=1}^{N_k} \alpha_i K_i^\top \phi'(K_i u)\,. \tag{3.8}$$

We find that explicitly constructing the huge matrix K_i and its transpose $K_i^\top$ is very time consuming in practice. In order to speed up the inference algorithm, we consider the filtering technique. We know $K_i u$ is equivalent to the filtering process $(k_i * u)$. In the case of symmetric boundary condition exploited in our work, $K_i^\top v$ can be interpreted as a convolution operation of image v with a kernel $\bar{k}_i$, which is obtained by mirroring k_i around its center point, only in the central region. In order to exactly calculate $K_i^\top v$, we need to carefully handle boundaries additionally. We implemented the computation of $K_i^\top v$ with C programming, see the Appendix A.

Therefore, with the filtering technique, $\nabla_u F(u)$ is accomplished by

$$\nabla_u F(u) = \sum_{i=1}^{N_k} \alpha_i \bar{k}_i * \phi'(k_i * u)\,. \tag{3.9}$$

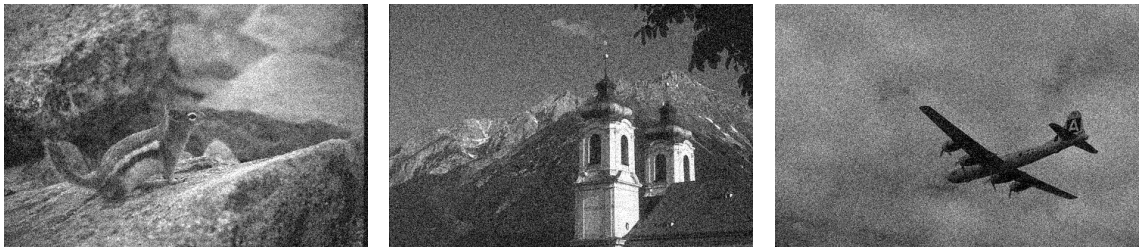
Note that $k_i * u$ is accomplished with symmetric boundary condition, and $\bar{k}_i * v$ is also accomplished with symmetric boundary condition, but additionally with careful consideration of the boundaries.

In the iPiano algorithm, we also need to compute the proximal mapping operation with respect to function G, which is the same as Equation 3.7. Then it is straightforward to employ the iPiano algorithm to solve the non-convex optimization problem Equation in 3.2.

3.1.1.2 Details of the evaluation experiments

As we know, image denoising performance of a specific method varies greatly for different image contents, in order to make a fair comparison, we conducted denoising experiments over a standard test dataset - 68 Berkeley test images identified by Roth and Black [111]. we used exactly the same noisy version of each test image for different methods and different test images were added with distinct noise realizations. All results were computed per image and then averaged over the test dataset.

We considered image denoising testing for various noise level $\sigma = \{15, 25, 50\}$. For noise levels other than $\sigma = 25$, we need to tune the parameter λ in (3.2). An empirical



(a) noisy (20.17) (b) noisy (20.17) (c) noisy (20.17)

(d) $\log(1+z^2)$ (29.31) (e) $\log(1+z^2)$ (30.40) (f) $\log(1+z^2)$ (36.84)

(g) $\log(1+|z|)$ (29.34) (h) $\log(1+|z|)$ (30.28) (i) $\log(1+|z|)$ (36.40)

(j) $|z|$ (29.03) (k) $|z|$ (29.50) (l) $|z|$ (34.24)

Figure 3.1: Comparison of denoising results obtained by three different penalty functions for noise level $\sigma = 25$. The numbers shown in the brackets refer to PSNR values with respect to the clean images.

choice of λ is: $\sigma = 15$, $\lambda = 25/\sigma \times 1.15$; $\sigma = 50$, $\lambda = 25/\sigma \times 0.8$.

σ	FoE	GOAL	$\log(1+z^2)$ 7×7, 48	$\log(1+\lvert z\rvert)$ 7×7, 48	$\lvert z\rvert$ 7×7, 48	$\log(1+z^2)$ 7×7, 98	$\log(1+z^2)$ 9×9, 80	$\log(1+z^2)$ direct DCT-7
15	30.99	31.03	31.18	31.18	30.45	**31.22**	**31.22**	30.92
25	28.40	28.45	**28.66**	28.64	28.04	**28.68**	**28.70**	28.47
50	25.35	25.44	**25.70**	25.58	25.12	**25.71**	**25.76**	25.58

Table 3.1: Summary of denoising experiments results (average PSNR values) of analysis prior based models on different noise levels.

3.1.1.3 Comparison of three different penalty functions

Table 3.1 shows the summary of denoising results achieved by different penalty functions. One can clearly see that two non-convex penalty functions lead to similarly good results and they significantly outperform the results of the convex function $|z|$. In additional, we can also see that the over-complete operator can not improve the performance too much and larger filters (9×9) can only achieve slightly better performance. Both of these two models are more time consuming than the model of 48 filters for inference; therefore, the learned model based on 48 filters of size 7×7 offers the best trade-off between computational cost and performance.

In the following testing experiments, we only consider the model of 48 filters and penalty function $\log(1 + z^2)$. We prefer the penalty function $\log(1 + z^2)$, in that it is completely smooth, making the corresponding minimization problem easier to solve. We present three denoising examples obtained by three different penalty functions in Figure 3.1.

3.1.1.4 Comparison to other analysis models

Our learned model is an analysis prior-based model, also a MRF-based system. In order to rank our model among other analysis models, we compared its performance with existing analysis models, including typical FoE models [8, 40, 52, 111, 115, 119] and the currently published best analysis operator model - GOAL [60]. For the GOAL method, we made use of the $l_{0.4}$-norm penalty function $|z|^{0.4}$, together with the learned analysis operator $\Omega \in \mathbb{R}^{98\times49}$ provided by the authors. We also utilized the iPiano algorithm to solve the corresponding minimization problem. We present image denoising results of considered approaches over 68 test images with noise level $\sigma = 25$ in Table 3.2. One can see that our model based on the penalty function $\log(1 + z^2)$ (48 learned filters, 7×7) has clearly achieved the best performance among all the related approaches. The comparison with the

model	potential	training	PSNR
5×5 FoE	ST&GLP.	contrastive divergence	27.77[111]
3×3 FoE	GSMs	contrastive divergence	27.95[119]
5×5 FoE	GSMs	persistent contrastive divergence	28.40[52]
5×5 FoE	ST	bi-level (truncated optimization)	28.24[8]
5×5 FoE	ST	bi-level (truncated optimization)	28.39[40]
5×5 FoE	ST	bi-level (implicit differentiation)	27.86[115]
7×7 GOAL	GLP	geometric conjugate gradient	28.45[60]
7×7 FoE	ST	bi-level (implicit differentiation)	**28.66**

Table 3.2: Summary of various analysis models and the average denoising results on 68 test images with $\sigma = 25$. We highlighted our result, as it is the best one.

best FoE [52] model and the latest analysis model GOAL for other noise levels is shown in Table 3.1. For all the noise levels, our trained analysis model outperforms both of them significantly.

An interesting result in Table 3.1 is that the performance of the direct DCT-7 model, which only utilizes a predefined analysis operator DCT-7 (48 filters of size 7×7), is already on par with the GOAL model, which involves much more carefully trained filters (98 filters of size 7×7). For this direct DCT-7 model, we used the $\log(1 + z^2)$ penalty function, and only optimized the norms and weights of the filters using our bi-level training algorithm. This result demonstrates the importance of non-convex penalty function and the effectiveness of our bi-level training scheme.

As the analysis operator of GOAL model is trained using a patch-based model, we can also use it in the manner of patch-averaging to conduct image denoising like K-SVD [42]. We embedded the learned analysis operator Ω into the patch-based analysis model (2.11), and used it to denoise each patch extracted from an image. We also considered overlapped windows and averaged the results in the overlapping regions to form the final denoised image. Just as expected, we got a much inferior result (average PSNR 28.25 over 68 test images) to the model formulated under the FoE framework (average PSNR 28.45).

3.1.1.5 Comparison to state-of-the-art methods

As mentioned in the preceding subsection, our opt-MRF model based on (1) the smooth penalty function $\log(1+z^2)$, and (2) 48 filters of size 7×7 offers the best trade-off between computational cost and performance; therefore, from now on, we only consider the opt-MRF with this specific configuration.

σ	KSVD	FoE	GOAL	BM3D	LSSC	EPLL	$\log(1+z^2)$ 7×7, 48	$\log(1+\lvert z\rvert)$ 7×7, 48	$\lvert z\rvert$ 7×7, 48
15	30.87	30.99	31.03	31.08	**31.27**	31.19	31.18	31.18	30.45
25	28.28	28.40	28.45	28.56	**28.70**	**28.68**	**28.66**	28.64	28.04
50	25.17	25.35	25.44	25.62	**25.72**	**25.67**	**25.70**	25.58	25.12

Table 3.3: Summary of denoising experiments results (average PSNR values) of our opt-MRF models (48 filters of size 7×7, different penalty functions) and state-of-the-art image denoising algorithms. We highlighted the state-of-the-art results.

In order to evaluate how well our analysis models work for denoising task, we compared their performance with leading image denoising methods, including three state-of-the-art methods: (1) BM3D [36]; (2) LSSC [91]; (3) GMM-EPLL [143] along with three leading generic methods: (4) a MRF-based approach, FoE [52]; (5) a synthesis sparse representation based method, KSVD [42] trained on natural image patches; and (6) the currently published best analysis operator learning method, GOAL [60]. All implementations were downloaded from the corresponding authors' homepages. We conducted denoising experiments over 68 Berkeley test images with various noise levels $\sigma = \{15, 25, 50\}$. All results were computed per image and then averaged over the number of images.

Table 3.3 shows the summary of results. One can see that our trained model based on the penalty function $\log(1+z^2)$ (48 learned filters, 7×7) outperforms three leading generic methods and is on par with three state-of-the-art methods for any noise level. To the best of our knowledge, this is the first time that a MRF model based on generic priors of natural images has achieved such clear state-of-the-art performance. Figure 3.2 gives a detailed comparison result between our learned analysis model and three state-of-the-art methods over 68 test images for $\sigma = 25$. We can see that all the points surround the diagonal line "$y = x$" closely, i.e., all considered methods achieve very similar results. Therefore, it is clear that our learned analysis models based on non-convex penalty function is state-of-the-art.

We present several denoising examples of the considered denoising methods in Figure 3.3 - Figure 3.6 for the noise level of $\sigma = 25$. As observed in a recent paper [68], state-of-the-art denoising methods possess complementary strengths and failure modes, namely, (1) FoE, having many visibly still noisy regions in the result image; (2) K-SVD, also having visibly still noisy regions and over-smoothing in some parts; (3) BM3D and LSSC, having some artifacts or patterns nonexistent in the clean image; (4) GMM-EPLL, undesired spots in the smooth regions such as the sky or water surface; and (5) our opt-

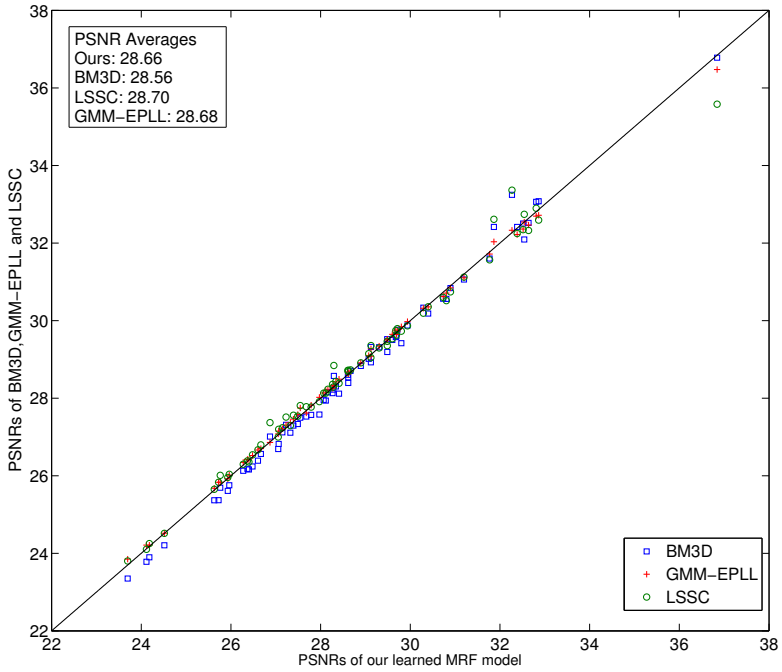
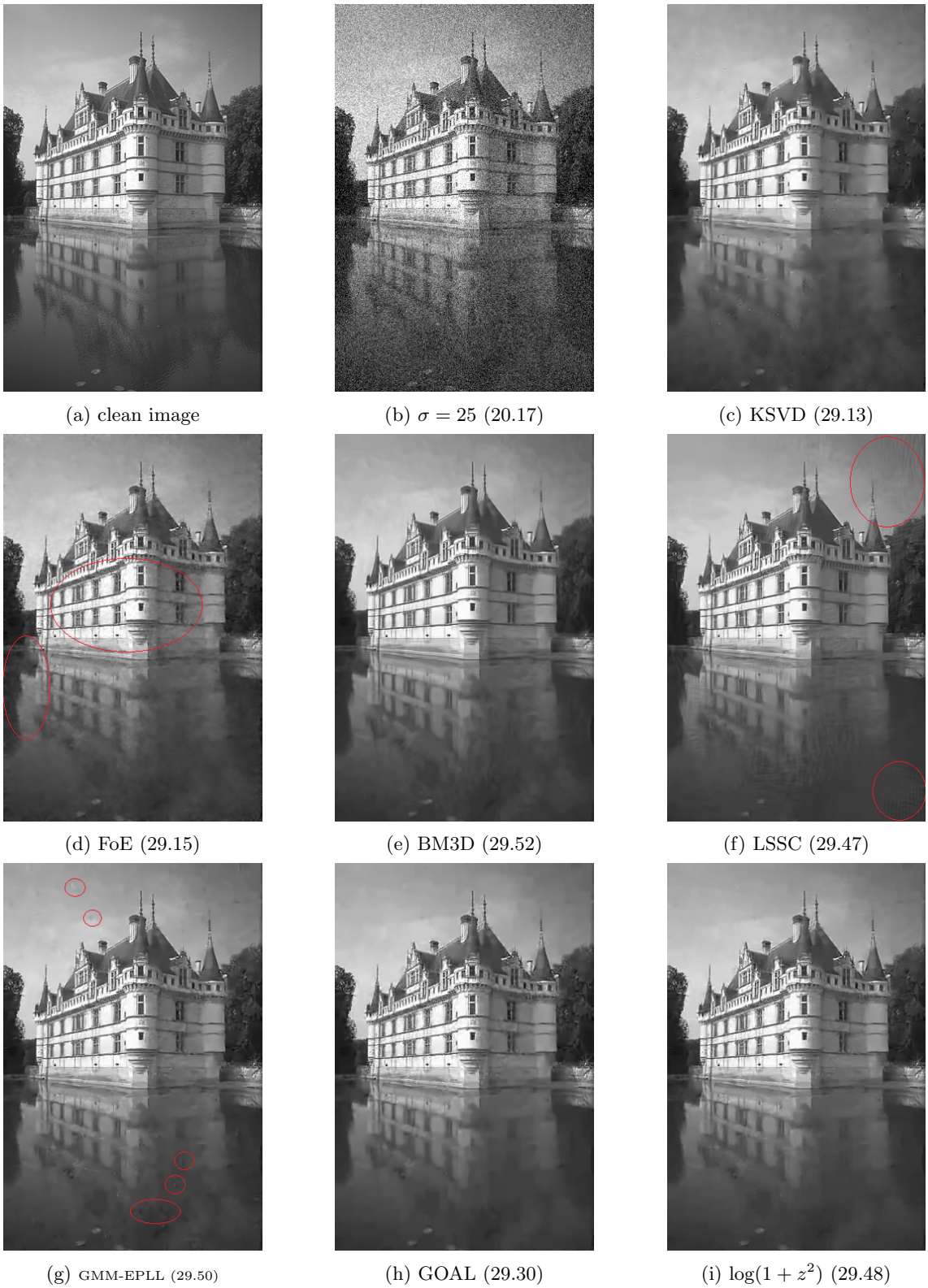


Figure 3.2: Scatter plot of the PSNRs over 68 Berkeley images produced by our learned $\log(1 + z^2)$-based analysis model, BM3D, GMM-EPLL and LSSC. A point above the diagonal line means performance is better than our model.

MRF model and the analysis operator based model - GOAL, over-smoothing in highly textured regions for some images. We highlight these typical failure modes by **red ellipses** in some denoised images.

It is worth to note that from the comparison in Figure 3.2, we see that even though our learned opt-MRF model outperforms the BM3D algorithm in terms of average PSNR value, there are two test images for which the BM3D algorithm performs much better. When we inspect these two special test images, we find that they are images with significant redundancy of local patterns. The denoising results of one of them is shown in Figure 3.7, where we can see that nonlocal methods, e.g., the BM3D and LSSC algorithm work much better than the remaining local algorithms. In fact, this result is not surprising because nonlocal models explicitly exploit the nonlocal self-similarity across the image, and therefore they can benefit from the repeated local patterns.

We also present some illustrative examples of image denoising at other noise levels, e.g., $\sigma = 15$ and $\sigma = 50$ in Figure 3.8 - Figure 3.10.

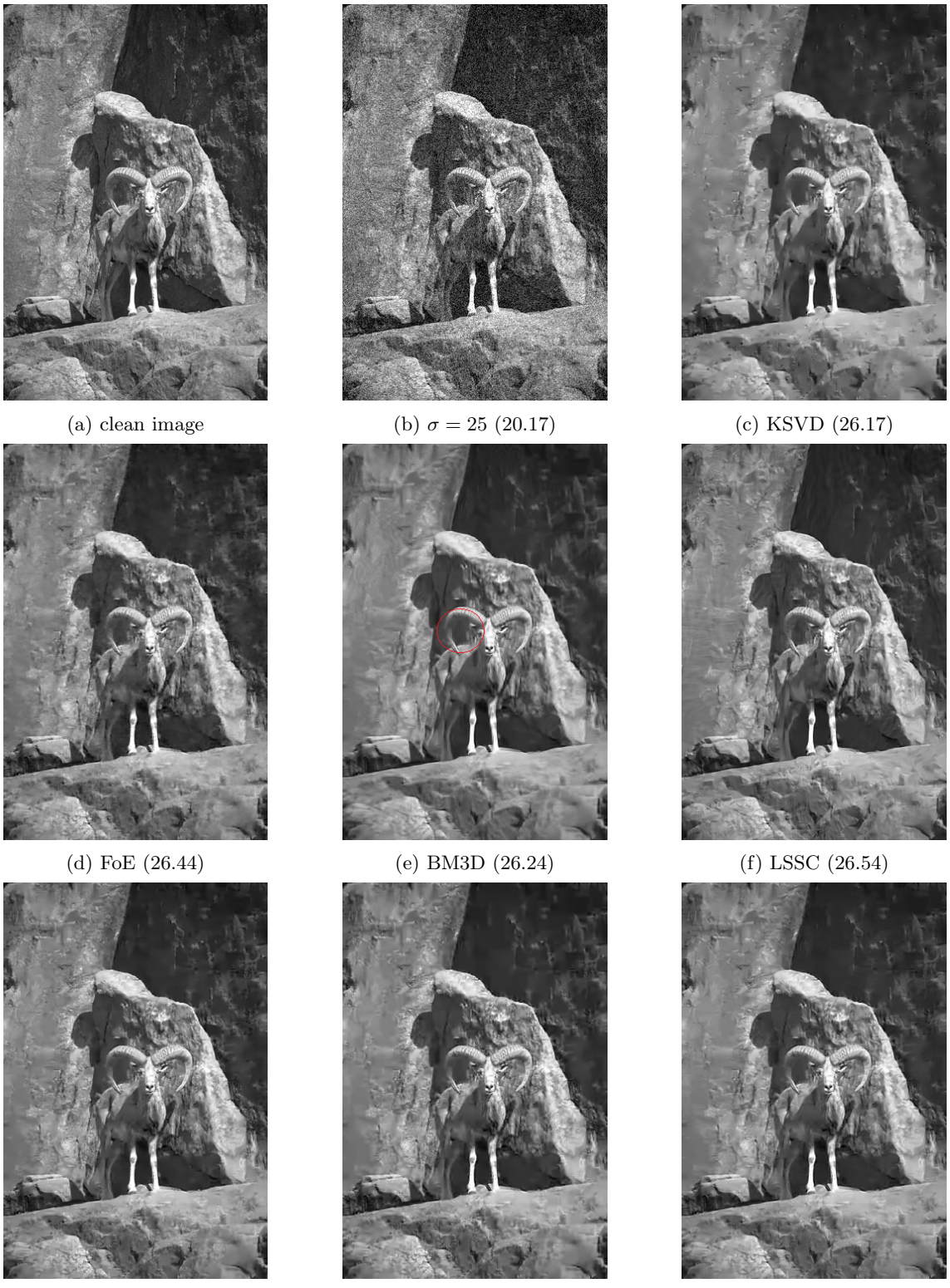


(a) clean image (b) $\sigma = 25$ (20.17) (c) KSVD (29.13)

(d) FoE (29.15) (e) BM3D (29.52) (f) LSSC (29.47)

(g) GMM-EPLL (29.50) (h) GOAL (29.30) (i) $\log(1 + z^2)$ (29.48)

Figure 3.3: Denoising results for the test image "water-castle" at noise level $\sigma = 25$. Red ellipses highlight the typical failure modes of considered approaches.



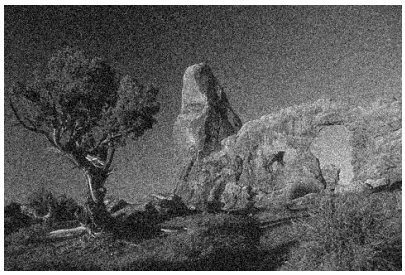
(a) clean image (b) $\sigma = 25$ (20.17) (c) KSVD (26.17)

(d) FoE (26.44) (e) BM3D (26.24) (f) LSSC (26.54)

(g) GMM-EPLL (26.52) (h) GOAL (26.29) (i) $\log(1 + z^2)$ (26.48)

Figure 3.4: Denoising results for the test image "goat" at noise level $\sigma = 25$. Red ellipses highlight the typical failure modes of considered approaches.

(a) clean image



(b) $\sigma = 25$ (20.17)

(c) KSVD (27.62)

(d) FoE (27.82)

(e) BM3D (27.58)

(f) LSSC (27.91)

(g) GMM-EPLL (28.02)

(h) GOAL (27.75)

(i) $\log(1 + z^2)$ (27.97)

Figure 3.5: Denoising results for a test image at noise level $\sigma = 25$. Red ellipses highlight the typical failure modes of considered approaches.

(a) clean image

(b) $\sigma = 25$ (20.17)

(c) KSVD (36.19)

(d) FoE (35.67)

(e) BM3D (36.78)

(f) LSSC (35.58)

(g) GMM-EPLL (36.47)

(h) GOAL (36.45)

(i) $\log(1 + z^2)$ (36.84)

Figure 3.6: Denoising results for the test image "airplane" at noise level $\sigma = 25$. Red ellipses highlight the typical failure modes of considered approaches.

(a) clean image (b) $\sigma = 25$ (20.17) (c) KSVD (32.20)

(d) FoE (31.44) (e) BM3D (**33.24**) (f) LSSC (**33.36**)

(g) GMM-EPLL (32.33) (h) GOAL (31.82) (i) $\log(1 + z^2)$ (32.27)

Figure 3.7: A special test image with a lot of repeated local patterns, e.g., the t-shirt region, for which the nonlocal models (e.g., BM3D and LSSC) generally perform better than the local models (e.g., ours), as they can benefit from this kind of nonlocal self-similarity.

(a) clean image

(b) $\sigma = 15$ (24.61)

(c) KSVD (31.37)

(d) FoE (31.73)

(e) BM3D (31.81)

(f) LSSC (31.85)

(g) GMM-EPLL (31.81)

(h) GOAL (31.80)

(i) $\log(1 + z^2)$ (31.77)

Figure 3.8: Denoising results for test image "squirrel" for noise level $\sigma = 15$.

(a) clean image

(b) $\sigma = 15$ (24.61)

(c) KSVD (31.61)

(d) FoE (31.72)

(e) BM3D (31.75)

(f) LSSC (31.96)

(g) GMM-EPLL (31.99)

(h) GOAL (31.69)

(i) $\log(1 + z^2)$ (31.88)

Figure 3.9: Denoising results for test image "elephant" for noise level $\sigma = 15$.

(a) clean image

(b) $\sigma = 50$ (14.15)

(c) KSVD (23.94)

(d) FoE (23.87)

(e) BM3D (25.03)

(f) LSSC (25.23)

(g) GMM-EPLL (24.73)

(h) GOAL (24.32)

(i) $\log(1 + z^2)$ (24.78)

Figure 3.10: Denoising results for test image "AD" for noise level $\sigma = 50$.

	KSVD	FoE	BM3D	LSSC	EPLL	GOAL	ours
T(s)	30	1600	4.3	700	99	112	12 (**0.138**)
psnr	28.28	28.40	28.56	28.70	28.68	28.45	28.66

Table 3.4: Typical run time of the denoising methods for a 481×321 image ($\sigma = 25$) on a server (Intel X5675, 3.07GHz). The highlighted number is the run time of GPU implementation.

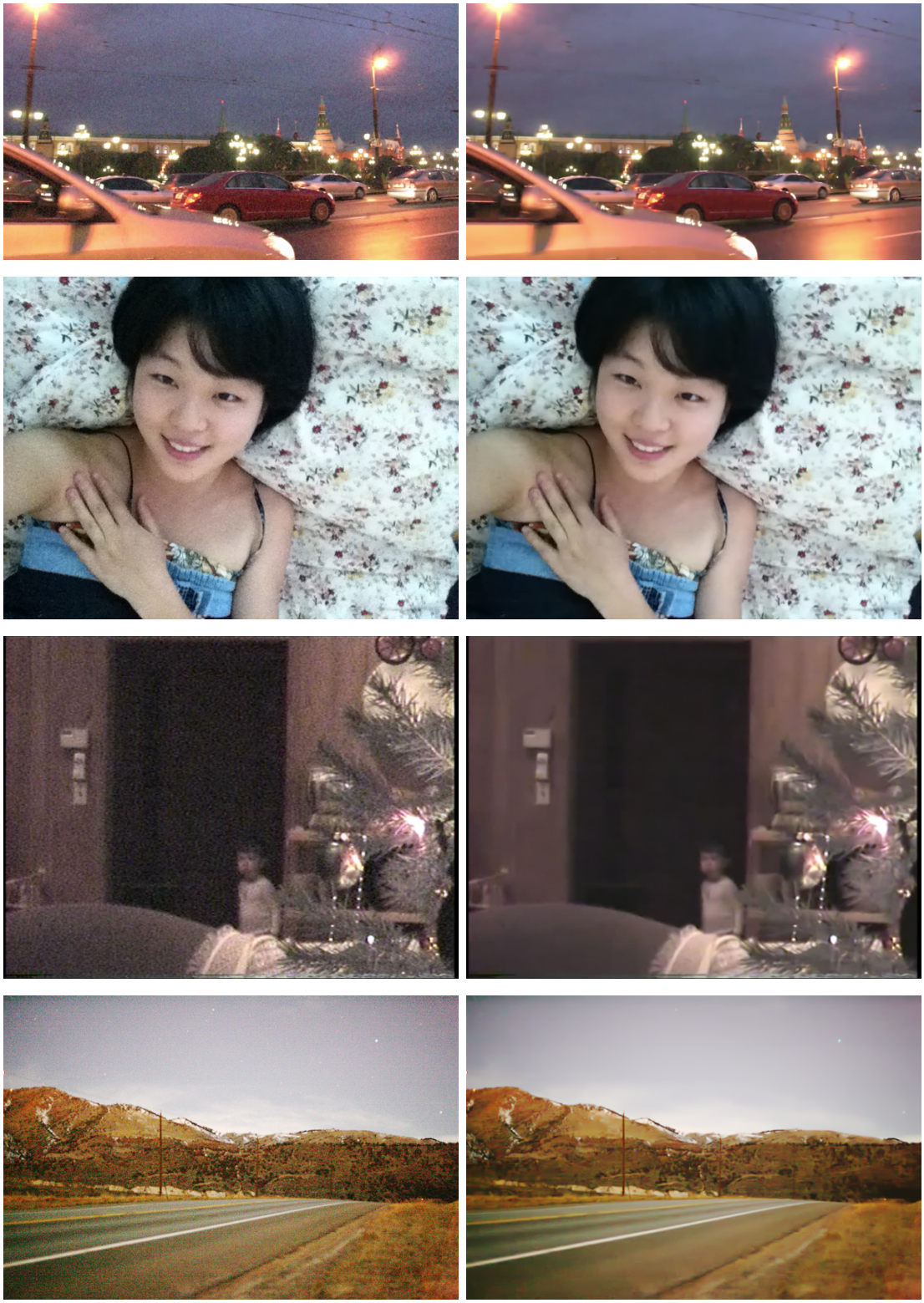
3.1.1.6 Comparison of run time

Our opt-MRF model has the advantage of simplicity, since it is a local (without block matching), only involves 48 filters (relatively few filters) and uses MAP estimate for inference (easy to implement). Our model only contains the operation of convolution of some filters with an image; therefore it is well-suited to GPU parallel computation. Our GPU implementation using CUDA based on NVIDIA Geforce GTX 780Ti accelerates the inference procedure significantly; for a denoising task with $\sigma = 25$, typically it takes 0.21s for image size 512×512, 0.138s for 481×321 and 0.078s for 256×256, i.e., using our GPU based implementation, image denoising can be conducted in near real-time at 12.8fps for 256×256 image sequence with clear state-of-the-art performance.

In contrast, GSM-EPLL is more involved (decomposing 200 64×64 covariance matrices into its eigenvectors, one can get 12,800 filters), and the inference is more time consuming; BM3D is an non-local (need for block-matching) specialized denoising approach, which is also more involved (3D collaborative filtering); LSSC is an image based method (training dictionary on-line based on noisy image itself, thus time consuming) and also more involved (e.g., 512 9×9 dictionary atoms); K-SVD, trained on natural image patches, is also more involved (e.g., 256 8×8 dictionary atoms); the inference of FoE model using MMSE estimate is quite slow because of Gibbs sampling.

In Table 3.4, we show the average run time of the considered denoising methods on 481×321 images for the case of noise level $\sigma = 25$. In this case, the iPiano algorithm typically takes 40 iterations to arrive at the solution. For heavier noise levels, e.g., $\sigma = 50$, it generally requires more iterations, typically $\sim$100 iterations. Therefore, the corresponding computation will linearly increase.

Considering the speed and quality of our model, it is a perfect choice of the base methods in the image restoration framework proposed in [68], which leverages advantages of existing methods.



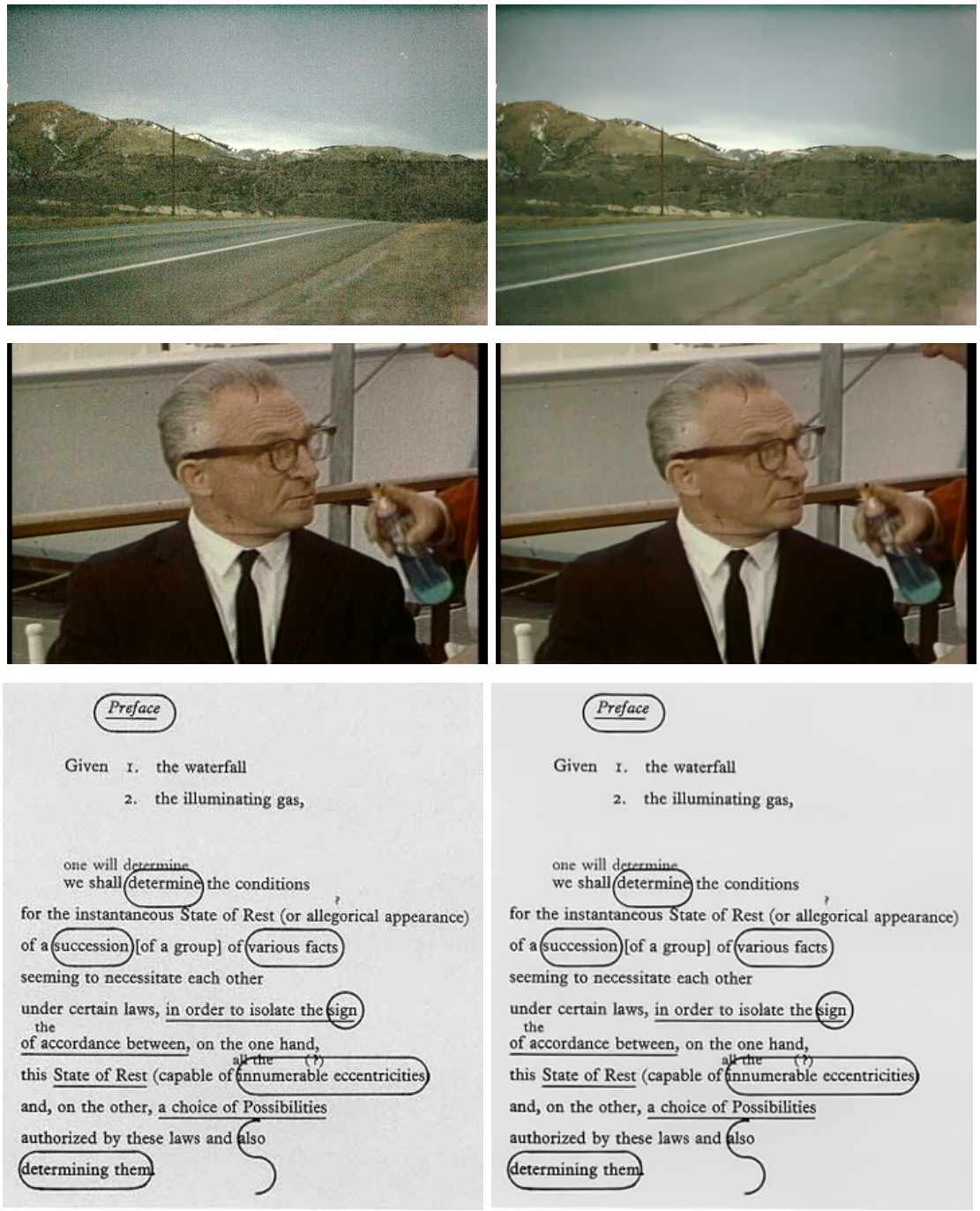


Figure 3.11: Denoising results of realistic noisy images using our opt-MRF model. **Left**: noisy images; **Right**: denoised images.

3.1.1.7 Realistic noise removal experiments

In this subsection, we present several image denoising experiments for realistic noise. Normally, images taken under the bad lighting condition will contain some noise, especially for images captured by a cell phone camera. Besides, images from old films are also rather noisy; images may become noisy (degraded) after recoding or transmission. We collected several realistic noisy (degraded) images, and tried to denoise them using our opt-MRF model, see Figure 3.11 for some examples. For color image, we process R, G, B channels separately. Visual inspection shows that our opt-MRF model also works very well for realistic noise removal application.

3.1.1.8 Discussion

In this subsection, we have evaluated the performance of the learned image regularizations, which are based on three different penalty functions, over the standard Gaussian denoising problem. For the convex penalty regularized variational model, we employed the primal-dual algorithm for optimization, and for the non-convex penalty regularized model, the iPiaon algorithm was exploited. Numerical experiments have demonstrated that the image regularizers based non-convex penalty functions clearly outperform those models based on convex penalty functions. In addition, non-convex penalty functions regularized variational models have achieved the best performance among the MRFs systems, and have been on par with state-of-the-art denoising algorithms.

The resulting variational model involving the learned image regularizations have an additional advantage of simplicity, as it only contains convolution operations of an image with a set of linear filters. Therefore, it is well-suited for GPU parallel computation. The GPU implementation based on CUDA programming significantly accelerates the inference procedure, with which we can conduct the image denoising task in near real time at 12.8fps for image size 256×256, meanwhile with state-of-the-art performance.

3.1.2 Impulse noise

Salt-and-pepper noise (also known as impulse noise) is a form of noise sometimes seen on images. It presents itself as sparsely occurring white and black pixels, i.e., strong outliers. In order to deal with strong outliers, a robust data term, which is less sensitive to the outliers, is typically required. In practice, the ℓ_1 data term seems a good choice. As a consequence, the resulting variational model based on our learned image regularizers is

defined as

$$u^* = \arg\min_u E(u) = \sum_{i=1}^{N_k} \alpha_i \phi(K_i u) + \lambda \|u - f\|_1 \,, \tag{3.10}$$

which also can be solved by the iPiano algorithm. We refer this model as the MRF-ℓ_1 model.

Casting the minimization problem of Equation 3.10 in the general form of the iPiano algorithm, we see that $F(u) = \sum_{i=1}^{N_k} \alpha_i \phi(K_i u)$ and $G(u) = \lambda \|u - f\|_1$. The gradient of $F(u)$ with respect to u is given as in the preceding subsection. The proximal mapping operator with respect to G poses simple point-wise soft shrinkage operation, which is given as

$$u = (\mathbf{I} + \tau \partial G)^{-1}(\hat{u}) \Longleftrightarrow u_q = f_q + \max(0, |\hat{u}_q - f_q| - \tau\lambda) \cdot \operatorname{sign}(\hat{u}_q - f_q) \,. \tag{3.11}$$

We directly apply the learned Student-t regularized image prior model (48 filters of size 7×7) to the impulse denoising problem. A denoising example is shown in Figure 3.12 together with the result of median filtering method. From Figure 3.12, one can see that the MRF-ℓ_1 model generally works quite well for the impulse denoising problem. However, we also find that the MRF-ℓ_1 model tends to generate somehow over smoothing results in the highly textured regions, i.e., the filters trained in the presence of Gaussian noise does not generalize so well for the impulse noise.

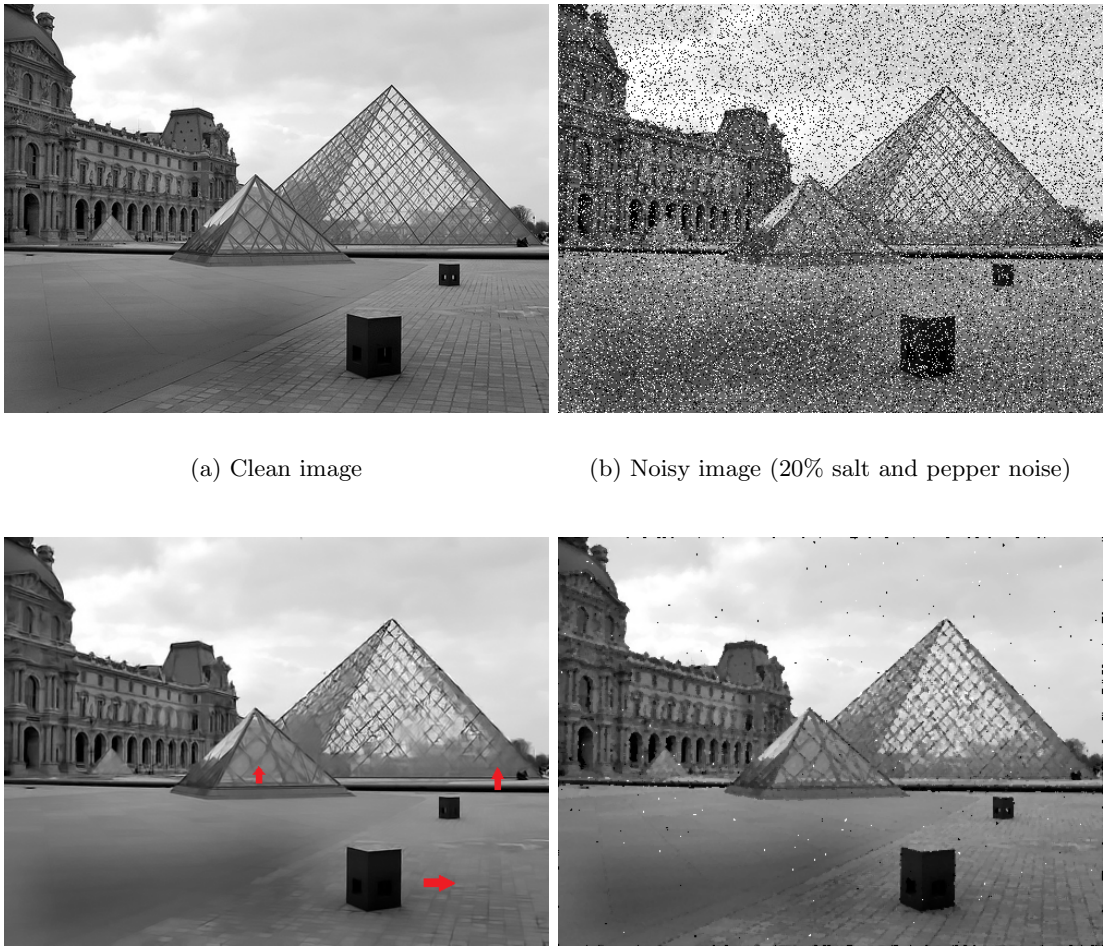
In order to alleviate these imperfection, we propose to train the filters directly based on the MRF-ℓ_1 model, i.e., we retrain the image regularizer by using the following training model directly based on the training samples corrupted by salt and pepper noise.

$$\begin{cases} \min\limits_{\alpha \geq 0, k} L(u^*(\alpha, k)) = \sum\limits_{s=1}^{S} \frac{1}{2} \|u_s^*(\alpha, k) - g_s\|_2^2 \\ \text{where } u_s^*(\alpha, k) = \arg\min\limits_u \sum\limits_{i=1}^{N_k} \alpha_i \phi(K_i u) + \|u - f_s\|_1, \end{cases} \tag{3.12}$$

where the penalty function is given as $\phi(z) = \log(1 + z^2)$. As in our training scheme, we need a smooth lower-level problem, and therefore, we make use of a small parameter to smooth the ℓ_1 data term. The revised lower-level problem is given as

$$\arg\min_u \sum_{i=1}^{N_k} \alpha_i \phi(K_i u) + \|u - f\|_{1,\varepsilon}, \tag{3.13}$$

where $\psi(z) = |z|_{1,\varepsilon} = \sqrt{z^2 + \varepsilon^2}$ with $\varepsilon = 10^{-2}$ in our experiment. For this revised lower-level problem, the iPiano algorithm is not the best choice, as it is not easy to compute



(a) Clean image

(b) Noisy image (20% salt and pepper noise)

(c) MRF-ℓ_1 model (25.70)

(d) Median filtering (22.89)

Figure 3.12: Image denoising in the case of impulse noise for an image corrupted by 20% salt and pepper noise by using (1) our MRF-ℓ_1 model incorporating filters trained in the case of Gaussian noise and (2) median filtering. Note that the MRF-ℓ_1 model leads to somehow over smoothing results in the highly textured regions as indicated in the image.

the solution for the proximal mapping operator with respect to the smoothed ℓ_1 function. Therefore, we exploit the L-BFGS algorithm to solve this problem as it is a smooth optimization problem. For the L-BFGS algorithm, we need to calculate the gradient of energy functional, which is given as

$$\frac{\partial E}{\partial u} = \sum_{i=1}^{N_k} \alpha_i K_i^\top \phi'(K_i u) + \psi'(u - f), \tag{3.14}$$

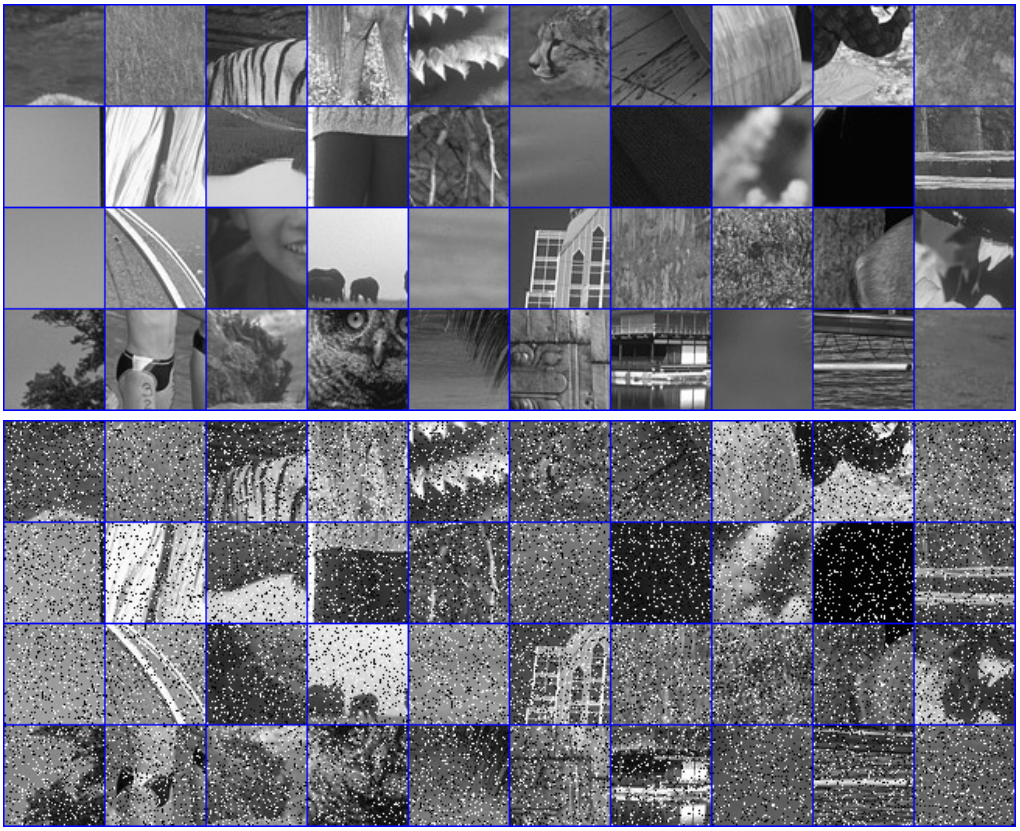


Figure 3.13: Subset of the training image pairs: the ground truth and the noisy image corrupted by 25% salt and pepper noise.

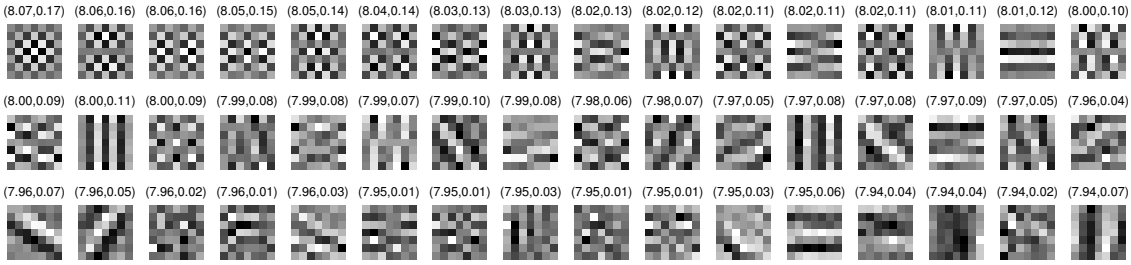
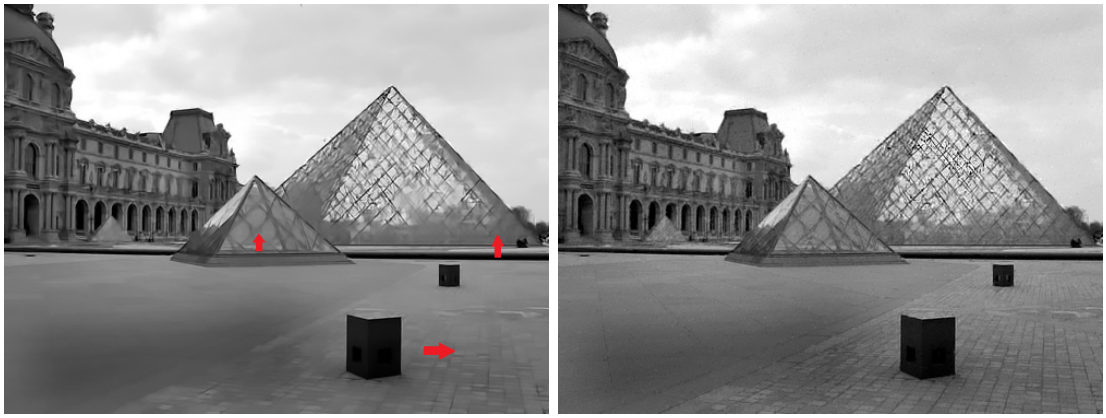


Figure 3.14: 48 learned filters of 7×7 in the case of ℓ_1 data term.

where $\psi'(u-f) = (\psi'((u-f)_1, \cdots, (u-f)_p))^\top \in \mathbb{R}^N$ with $\psi'(z) = \frac{z}{\sqrt{z^2+\varepsilon^2}}$. In this case, the Hessian matrix related to the energy functional E, which is required in the training



(a) With Gaussian filters (25.70) (b) With newly trained filters (25.65)

Figure 3.15: A performance comparison of the MRF-ℓ_1 model with different image regularization models for the impulse noise removal problem. The left one is obtained by using the filters trained in the case of Gaussian noise; the right one is generated by using the filters directly trained in the case of impulse noise. We see that even though both methods achieve similar PSNR values, the model with the specialized filters can better preserve the image structures, e.g., line-like structures.

process, is given as

$$H_E(u) = \sum_{i=1}^{N_k} \alpha_i K_i^\top \mathcal{D}_i K_i + \psi''(u - f), \tag{3.15}$$

where $\psi''(u-f) = \operatorname{diag}\left(\psi''((u-f)_1, \cdots, (u-f)_p)\right) \in \mathbb{R}^{N\times N}$ with $\psi''(z) = \varepsilon^2/(z^2+\varepsilon^2)^{3/2}$.

We set up the training dataset with the same clean samples in the training of Gaussian noise, but we replaced the noisy samples with 25% impulse noise corrupted images. An exemplary subset of the training samples are shown in Figure 3.13.

We still concentrated on the model capacity of 48 filters of size 7×7. The learned filters are shown in Figure 3.14. Then we applied this newly trained image regularizer to the impulse noise removal problem. The result with this new image regularization model is shown in Figure 3.15. We see that even though both methods achieve similar PSNR values, the model with the specialized filters can better preserve the image structures, e.g., line-like structures. Therefore, for the case of impulse noise removal problem, the specialized training helps to alleviate the over-smoothing problem in the MRF-ℓ_1 model with filters trained in the case of Gaussian noise.

3.1.3 Multiplicative noise reduction (despeckling)

In this subsection we propose a novel variational model for multiplicative noise reduction based on the FoE image prior model. The resulting model corresponds to a non-convex minimization problem, which can be efficiently solved by our proposed non-convex optimization algorithm - iPiano. Experimental results based on synthetic speckle noise and real synthetic aperture radar (SAR) images suggest that the performance of our proposed method is on par with the best published despeckling algorithm. Besides, our proposed model comes along with an additional advantage, that the inference is extremely efficient. Our GPU based implementation takes less than 1s to produce state-of-the-art despeckling performance.

3.1.3.1 Introduction

Images generated by coherent imaging modalities, e.g., synthetic aperture radar (SAR), ultrasound and laser imaging, inevitably come with multiplicative noise (also known as speckle), due to the coherent nature of the scattering phenomena. The presence of this noise prevents us from interpreting valuable information of images, such as textures, edges and point target, and therefore speckle reduction is often a necessary preprocessing step for successful use of classical image processing algorithms involving image segmentation and classification. The topic of speckle noise reduction (despeckling) has attracted a lot of research attention since early 1980s [72, 77]. At present it has been extensively studied. Roughly speaking, the major despeckling techniques fall into four categories: filtering based methods in (1) spatial domain; or (2) a transform domain, e.g., wavelet domain; (3) nonlocal filtering and (4) variational methods.

Early filtering techniques in the spatial domain are developed under the minimum mean square error (MMSE) criterion [77], and then progress to more sophisticated and promising maximum a posterior (MAP) approaches [72]. Recently, the bilateral filtering has also been modified for despeckling [81]. The emergence of wavelet transform in the early of 1990s, opened the way to a new generation of despeckling techniques. There were intensive studies of wavelet based despeckling approaches, see, for instance [5, 12] and references therein.

Nonlocal approaches, which take the advantage of self-similarity commonly present in natural as well as SAR images, have been already introduced to SAR despeckling [34, 37, 103]. By taking into account the peculiar features of multiplicative noise, the so-called SAR-BM3D algorithm [103], which is a SAR-oriented version of the well-known

BM3D algorithm [36], exhibits the best published despeckling performance at present.

The last class of methods are variational ones, which minimize some appropriate energy functionals, consisting of a regularizer (also called image prior) and a data fitting term. Up to now, the well-known total variation (TV) has been widely used as a regularizer [7, 121, 140], and the total generalized variation (TGV) regularizer [18] also has been investigated in a recent work [46].

Motivated by the results obtained by the FoE image prior regularized models in the preceding subsection for the Gaussian denoising and impulse noise removal problems, it is interesting to investigate the FoE prior based model for despeckling. In this subsection, we propose a FoE prior based variational approach for speckle removal. Again we use the iPiano algorithm to solve the corresponding minimization problem. Our proposed method obtains strongly competitive despeckling performance w.r.t. the state-of-the-art method - SAR-BM3D [103], meanwhile, preserve the property of high computational efficiency.

3.1.3.2 The variational model for despeckling

In our work, we propose FoE prior based variational models for speckle noise removal and in particular for SAR images. We apply an efficient algorithm to solve the corresponding optimization problems.

Given an observation image f corrupted by multiplicative noise, the FoE prior based despeckling model is defined as the following energy minimization problem

$$\hat{u} = \arg\min_{u} E(u, f) = E_{\mathrm{FoE}}(u) + D(u, f)\,, \tag{3.16}$$

where u is the underlying unknown image, the first term is the FoE prior model, and the second part is the data fidelity term, derived from the multiplicative noise model.

The FoE prior utilized in our despeckling model: For the image despeckling problem, we directly used the filters trained in the context of Gaussian denoising problem. We exploited the FoE model with 48 filters of size 7×7 and the Lorentzian penalty function trained based on the Gaussian noise level of $\sigma = 15$. The filters are shown in Figure 3.16.

Modeling of the data term: Let f be the observed SAR image amplitude, which follows a Nakagami distribution depending on the underlying true image amplitude u, the

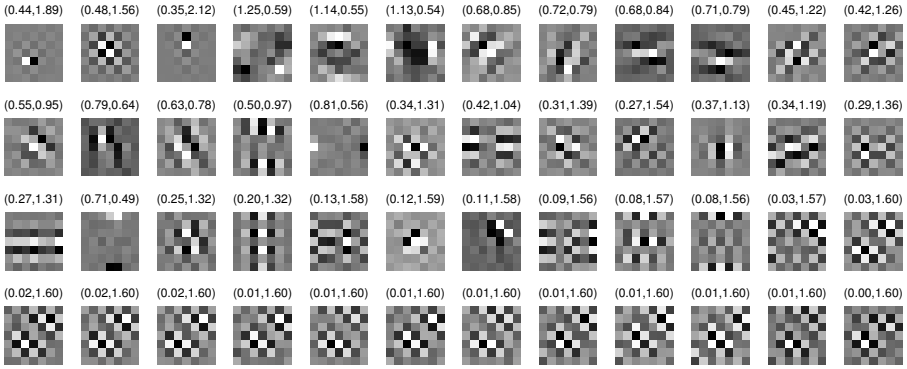


Figure 3.16: 48 learned filters of size 7×7 exploited in our despeckling model. The first number in the bracket is the norm of the filter and the second one is the weight α_i.

square root of the reflectivity [55]

$$p(f|u) = \frac{2L^L}{\Gamma(L)u^{2L}} f^{2L-1} \exp\left(-\frac{Lf^2}{u^2}\right), \tag{3.17}$$

with L the number of looks of the image (i.e., number of independent values averaged) and Γ is the classical Gamma function.

According to the Gibbs function, this likelihood leads to the following energy term via $E = -\log p(f|u)$

$$D(u, f) = L \cdot (2\log u + \frac{f^2}{u^2}). \tag{3.18}$$

Combining this data term with the FoE prior model, we arrive at the following variational model

$$\arg\min_{u>0} \sum_{i=1}^{N_k} \alpha_i \phi(k_i * u) + \frac{\lambda}{2} \left\langle 2\log u + \frac{f^2}{u^2}, 1 \right\rangle, \tag{3.19}$$

where $\langle \cdot, \cdot \rangle$ denotes the common inner product. Note that the data term is not convex w.r.t. $u > 0$, which will generally make the corresponding optimization problem harder to solve.

There exists an alternative method to define a convex data term by using the classical Csiszár I-divergence model [35]. Although the I-divergence data fitting term typically used in the context of Poisson noise, this seemingly inappropriate data term performs very well in the TV and TGV regularized variational models for despeckling [46, 121]. Therefore, we also study this variant for data term modeling in this work. Following previous works of modeling the SAR image intensity [46, 121], the I-divergence based data term for the

amplitude model is given by

$$D(u,f) = \frac{\lambda}{2}(u^2 - 2f^2 \log u)\,, \tag{3.20}$$

which is strongly convex w.r.t. u for $u > 0$. Then the variational model involving this convex data term is given by

$$\arg\min_{u>0} \sum_{i=1}^{N_k} \alpha_i \phi(k_i * u) + \frac{\lambda}{2} \left\langle u^2 - 2f^2 \log u, 1 \right\rangle\,. \tag{3.21}$$

Solving the variational despeckling models: Due to the non-convexity of the prior term, the proposed variational models impose generally very hard optimization problems, especially for the model (3.19) with a non-convex data term. In this work, we resort to our proposed non-convex optimization algorithm - iPiano [97] to solve them.

The iPiano algorithm is designed for a structured non-smooth non-convex optimization problem, which is composed of a smooth (possibly non-convex) function F and a convex (possibly non-smooth) function G:

$$\arg\min_{u} H(u) = F(u) + G(u)\,. \tag{3.22}$$

Recall that the iPiano algorithm is an inertial force enhanced forward-backward splitting algorithm with the following basic update rule

$$u^{n+1} = (I + \tau \partial G)^{-1} \left(u^n - \tau \nabla F(u^n) + \mu(u^n - u^{n-1})\right), \tag{3.23}$$

where τ and μ are the step size parameters.

For the model (3.19) with a non-convex data term, we can convert the data term to a convex function via commonly used logarithmic transformation (i.e., $w = \log u$). Therefore, the minimization problem is rewritten as

$$\arg\min_{w} \sum_{i=1}^{N_k} \alpha_i \phi(k_i * e^w) + \frac{\lambda}{2} \left\langle 2w + f^2 e^{-2w}, 1 \right\rangle\,, \tag{3.24}$$

with $u = e^w$. Casting (3.24) in the form of (3.22), we see that $F(w) = \sum_{i=1}^{N_k} \alpha_i \phi(k_i * e^w)$ and $G(w) = \frac{\lambda}{2} \left\langle 2w + f^2 e^{-2w}, 1 \right\rangle$. In order to use the iPiano algorithm, we need to calculate

the gradient of F and the proximal map w.r.t. G. It is easy to check that

$$\nabla_w F = \sum_{i=1}^{N_k} \alpha_i W K_i^\top \phi'(K_i e^w)\,,$$

where K_i is an $N \times N$ highly sparse matrix, implemented as 2D convolution of the image u with filter kernel k_i, i.e.,, $K_i u \Leftrightarrow k_i * u$, $\phi'(K_i u) = (\phi'((K_i u)_1), \cdots, \phi'((K_i u)_N))^\top \in \mathbb{R}^N$, with $\phi'(x) = 2x/(1+x^2)$, and $W = \mathrm{diag}(e^w)$.

The proximal map w.r.t. G is given as the following minimization problem

$$(I + \tau \partial G)^{-1}(\hat{w}) = \arg\min_w \frac{\|w - \hat{w}\|_2^2}{2} + \frac{\tau\lambda}{2} \left\langle 2w + f^2 e^{-2w}, 1 \right\rangle . \tag{3.25}$$

Instead of using a direct solver for (3.25) in terms of the Lambert W function [32], we utilized Newton's method. We found that this scheme has a quite fast convergence (less than 10 iterations).

For the variational model (3.21), $F(u) = \sum_{i=1}^{N_k} \alpha_i \phi(k_i * u)$, $G(u) = \frac{\lambda}{2} \left\langle u^2 - 2f^2 \log u, 1 \right\rangle$. Then we have

$$\nabla_u F = \sum_{i=1}^{N_k} \alpha_i K_i^\top \phi'(K_i u)\,, \tag{3.26}$$

and the proximal map w.r.t. G is given by the following point-wise calculation

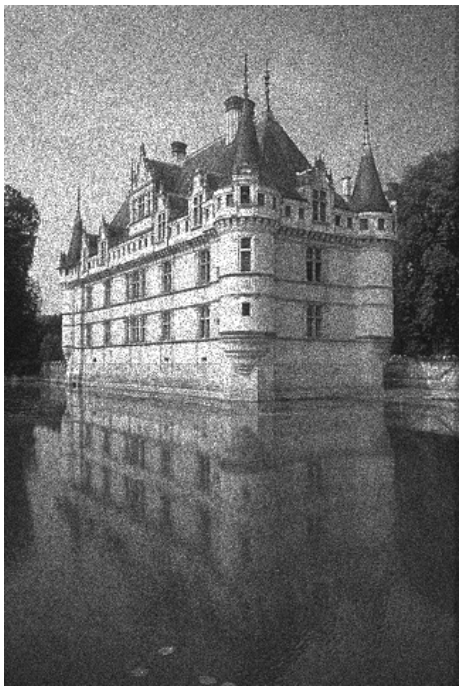
$$(I + \tau \partial G)^{-1}(\hat{u}) \Leftrightarrow u_p = \frac{\hat{u}_p + \sqrt{\hat{u}_p^2 + 4(1+\tau\lambda)\tau\lambda f_p^2}}{2(1+\tau\lambda)} \tag{3.27}$$

3.1.3.3 Despeckling experiments

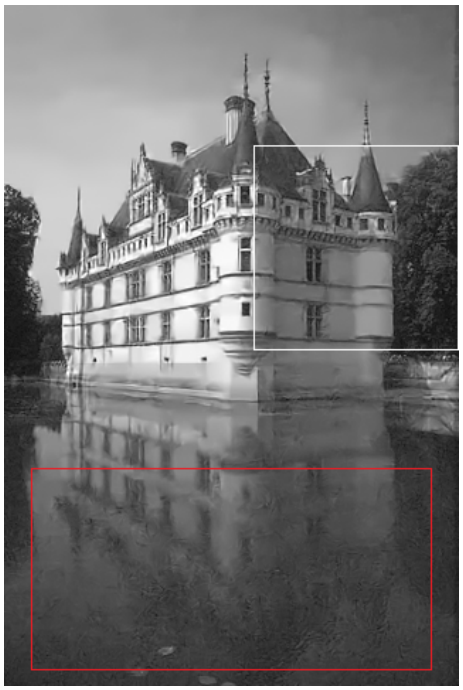
In this subsection, we evaluated the performance of our proposed variational models based on images corrupted by synthetic speckle noise and real noisy SAR images. For synthetic experiments, we calculated the common measurements PSNR and SSIM index [131], and compared the results with a state-of-the-art algorithm - SAR-BM3D [103].

The influence of data term: We started with conducting a preliminary despeckling test on a commonly used natural image contaminated by multiplicative noise with $L = 8$, using our proposed variational models (3.24) and (3.21). We carefully tuned the parameter λ to insure that the corresponding variational models achieve the best performance. The despeckling results are shown in Figure 3.17(b) and (c).

A first impression about the results is: the variational model with an accurate data fitting term, which is exactly derived from the multiplicative noise model can lead to better



(a) Noisy image (21.23/0.4267)



(b) Model (3.24) (29.30/0.8567)

(c) Model (3.21) (28.40/0.8264)

(d) Combined (3.28)(29.62/0.8794)

Figure 3.17: Despeckling results for a widely used natural image corrupted by multiplicative noise with $L = 8$, using our proposed variational models with different data terms. The recovery quality is measured by PSNR/SSIM index.

result than the model with an "inappropriate" data term. But after having a closer look

Figure 3.18: Standard test images of size 256×256 (Couple, Lenna and Peppers).

at the despeckling images, we found an interesting phenomena: these two methods possess complementary strengths and failure modes. For example, for the highly textured region (highlighted by the white rectangle), (3.24) works much better than (3.21); however, for the homogeneous region (highlighted by the red rectangle), (3.21) generates preferable result. Then an intuitive idea arises to integrate these two data terms so as to leverage their advantages.

As a result, a new variational model incorporating these two data terms comes out as follows

$$\arg\min_{w} E_{\mathrm{FoE}}(e^w) + \frac{\lambda_1}{2}(2w + f^2 e^{-2w}) + \frac{\lambda_2}{2}(e^{2w} - 2f^2 w)\,, \tag{3.28}$$

with $u = e^w$. The data fitting term is still convex, and we can utilize iPiano to solve it. The proximal map w.r.t. G is also solved using Newton's method.

We manually tuned the parameters λ_1 and λ_2, and conducted the same despeckling experiment. The final result is shown in Figure 3.17(d). One can see that the combined model indeed leads to significant improvement in terms of both PSNR and SSIM index value. From now on, we will use the combined model (3.28) for despeckling experiments.

Guideline to tune the model parameters: In our despeckling model (3.28), there are two free parameters to be tuned in practice. For the choice of the parameters λ_1 and λ_2, we used an alternate optimization strategy, namely, we set one parameter to zero and tuned the other one, such that the despeckling result is optimized (measured by PSNR and SSIM values if the ground truth is available, or by visual inspection for the case of no ground truth). Then we consider a linear combination of two data terms, i.e., we use a factor a ($0 \leq a \leq 1$) to scale the parameter λ_1 and a factor $1 - a$ to scale the parameter λ_2. Usually, the choice of $a = 0.5$ works quite well in practice. For some instances of L,

we recommend to make use of the default settings in Table 3.5.

L	λ_1	λ_2
$L = 1$	160	0.004
$L = 2$	250	0.006
$L = 3$	310	0.008
$L = 4$	390	0.01
$L = 5$	450	0.01
$L = 8$	550	0.02

Table 3.5: Default settings of the model parameters λ_1 and λ_2 for some typical L

Results on synthetic speckle noise: We first evaluated the performance of our proposed variational model (3.28) for despeckling based on synthetic noisy images. The results are compared to the best published despeckling algorithm - SAR-BM3D [103]. First of all, we considered three standard test images widely used in image processing community, see in Figure 3.18. The despeckling results at three representative numbers of look $L = 1, 3, 8$ are summarized in Table 3.6. We made the following empirical choice for the parameters λ_1 and λ_2 for different L: if $L = 8$, $\lambda_1 = 550, \lambda_2 = 0.02$; if $L = 3$, $\lambda_1 = 310, \lambda_2 = 0.008$; and if $L = 1$, $\lambda_1 = 160, \lambda_2 = 0.004$.

From Table 3.6, one can see that these two competing approaches generate very similar results, with almost identical PSNR and SSIM values. We present three despeckling examples obtained by these two algorithms in Figure 3.19.

It is clear that the despeckling performance of a particular method varies greatly for different image contents, in order to make a comprehensive comparison, we conducted additional despeckling experiments over a standard test dataset - 68 Berkeley test images identified by Roth and Black [111], which is widely used for AWGN denoising test. All the results were computed per image and then averaged over the test dataset. We present the results in Table 3.7. Again, two competing algorithms behave similarly, which implies that our proposed variational model based on the FoE prior is on par with the best published despeckling algorithm - SAR-BM3D.

In general, for the cases $L \geq 3$, our approach can surpass the performance of SAR-BM3D in most cases, in terms of PSNR and SSIM values; however, for the cases $L = 1, 2$, our method leads to somehow over-smoothing results and the block-type artifacts appear apparent in the despecked images (see Figure 3.19(c) for example), therefore inferior PSNR and SSIM values from time to time. This is reasonable, because our proposed approach is a local model, in contrast to patch-based non-local methods, such as SAR-BM3D. When the

	Lenna		Peppers		Couple	
$L = 8$	31.29	0.8948	29.69	0.8693	29.01	0.8416
	31.38	0.8968	30.64	0.8803	29.23	0.8368
$L = 3$	28.81	0.8453	27.48	0.8233	26.60	0.7559
	28.85	0.8541	28.38	0.8409	26.52	0.7510
$L = 1$	25.92	0.7432	24.95	0.7495	23.98	0.6210
	25.85	0.7771	25.34	0.7770	23.79	0.6023

Table 3.6: despeckling results of SAR-BM3D [103] and our approach. Our results are marked with blue color (with model (3.28)). The results are reported with PSNR and SSIM values.

	$L = 8$		$L = 3$		$L = 1$	
Noisy	21.61	0.5355	17.42	0.3778	12.95	0.2231
SAR-BM3D	29.35	0.8520	27.12	0.7756	24.85	0.6757
Ours	29.54	0.8481	27.07	0.7628	24.65	0.6691

Table 3.7: despeckling results on 68 Berkeley test images. The results are reported with average PSNR and SSIM values.

number of looks is quite low, e.g., $L = 1$, the input image is too noisy, which makes our local model less effective to infer the underlying structures solely from the local neighborhoods. In this case, the non-locality helps, which can integrate information from non-local patches, and the advantage of non-local models comes through.

Results on a real SAR image: In order to demonstrate the effectiveness of our proposed method on real SAR image despeckling task, we conducted a despeckling test on a real noisy SAR image, which is taken by the radar system equipped with the JSTARS aircraft [SAR] (the number of looks $L = 5$). For this experiment, we set $\lambda_1 = 50, \lambda_2 = 0.15$ for our model.

The results of different algorithms are shown in Figure 3.20. One can see that the tiny structures in the image, especially the region highlighted by the white rectangle, are well-preserved by our approach after despeckling; however, they are smoothed out to different extents by other algorithms. We provide three additional experiments for real SAR images as shown in Figure 3.21 - 3.23.

Run time: Efficiency is also an important aspect for real world SAR despeckling tasks. On the server platform of Intel X5675 3.07GHz, for images of size 481×321 exploited in our experiments, a typical run time of the SAR-BM3D algorithm is about 65.4s, which can be reduced to 5.6s, by its improved version - FANS [34], at the expense of slight performance degradation. Our method typically consumes 27s with a pure Matlab code.

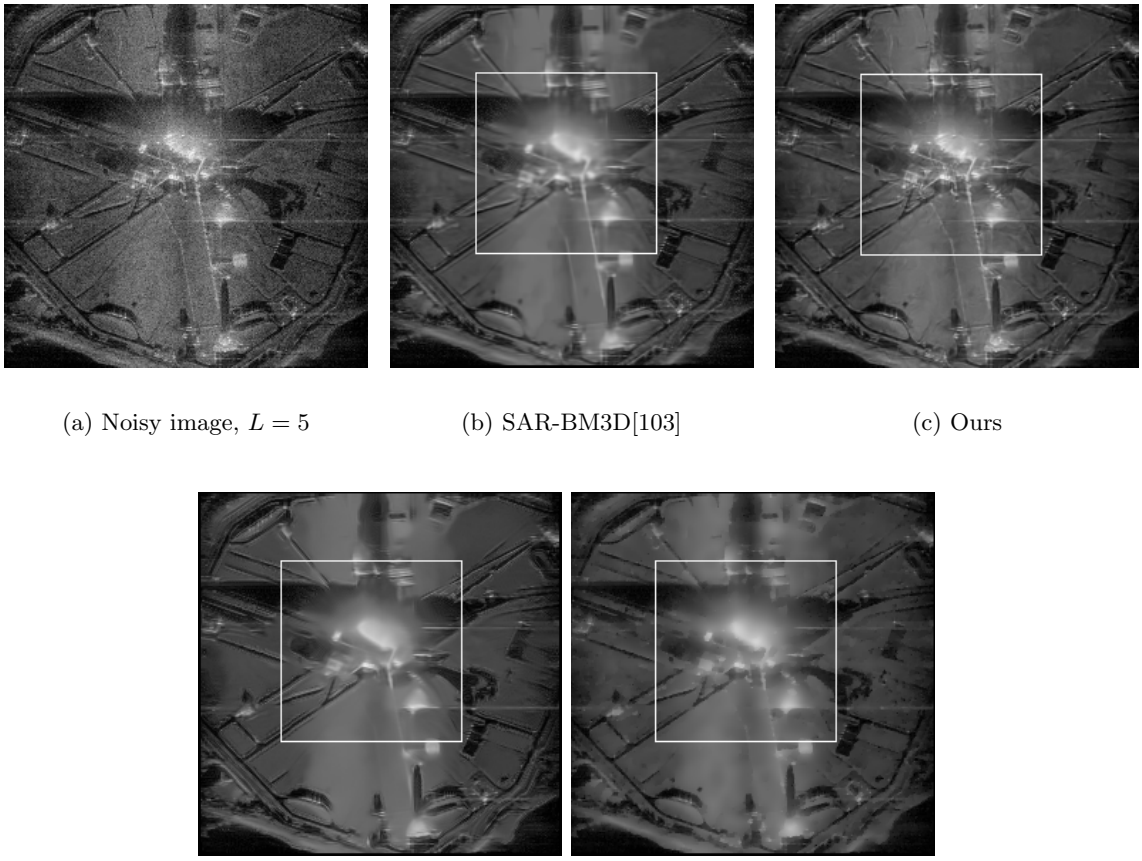
(a) $L = 1$ (12.10/0.1595) (b) SAR-BM3D (25.92/0.7432) (c) Ours (25.85/0.7771)

(d) $L = 3$ (16.48/0.2939) (e) SAR-BM3D (27.48/0.8233) (f) Ours (28.38/0.8409)

(g) $L = 8$ (23.19/0.6981) (h) SAR-BM3D (28.06/0.8663) (i) Ours (28.39/0.8647)

Figure 3.19: Performance comparison to state-of-the-art algorithm - SAR-BM3D [103] for different L. The results are reported by PSNR/SSIM index.

However, our model is a local model, which solely contains convolution of linear fil-



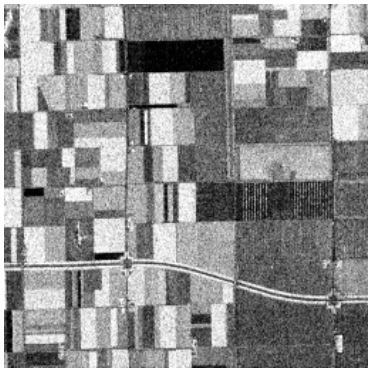
(a) Noisy image, $L = 5$ (b) SAR-BM3D[103] (c) Ours

(d) PPBit[37] (e) TGV based[46]

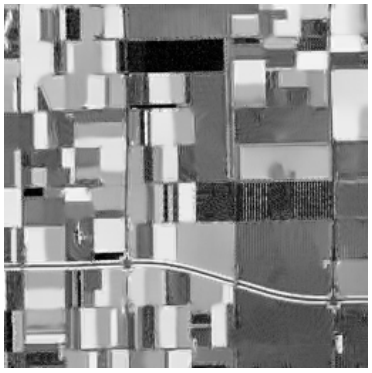
Figure 3.20: Performance comparison of different algorithms on a real SAR image.

ters with an image, in contrast to non-local models, e.g., SAR-BM3D and its variant FANS. Therefore, our model is well-suited to GPU parallel computation. Our GPU implementation based on a NVIDIA Geforce GTX 680 accelerates the inference procedure significantly; for a despeckling task with $L = 8$, it typically takes 0.6s for images of size 512×512, 0.41s for 481×321 and 0.2s for 256×256. In the case of $L = 8$, the iPiano algorithm takes about 150 iterations.

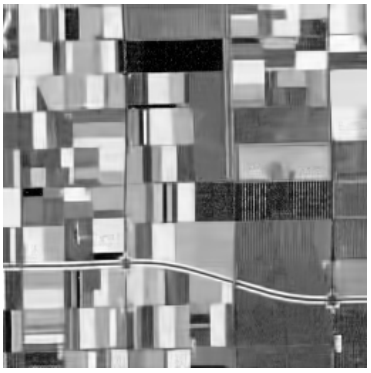
In practice, we find that for the cases of relative large L, e.g., $L \geq 3$, usually 150 iterations is enough; however, for the cases of small L, e.g., $L \leq 2$, the corresponding minimization problems become harder to solve, and therefore the iPiano algorithm takes more iterations to achieve a satisfying solution, typically 500 iterations. As a result, the run time will increase linearly.



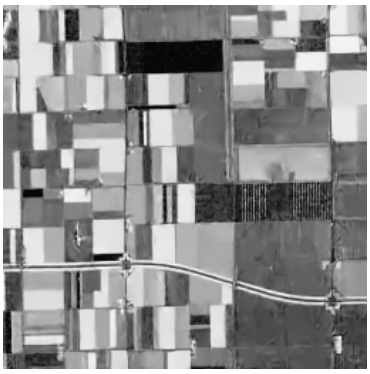
(a) Noisy image



(b) PPBit



(c) SAR-BM3D

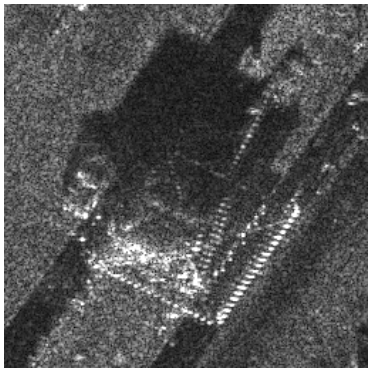


(d) Ours with $\lambda_1 = 200$, $\lambda_2 = 0.05$

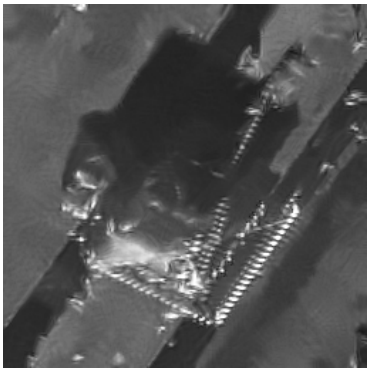
Figure 3.21: SAR image 1: sensor AirSAR, Amplitude image, number of looks $L = 6$.

3.1.3.4 Conclusion

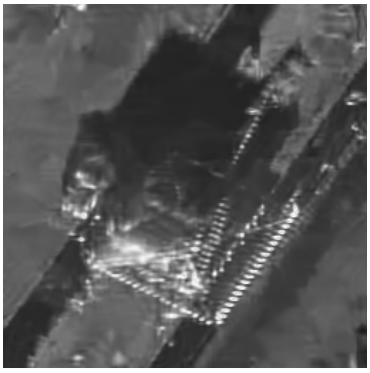
In this subsection, we have proposed a novel variational model for speckle noise reduction, based on an expressive image prior model - FoE model. Our new variational model poses a generally demanding non-convex optimization problem and we have used our proposed algorithm - iPiano to solve it efficiently. Numerical results on synthetic images corrupted by speckle noise and a real SAR image demonstrate that the performance of our method is clearly on par with the best published despeckling algorithm - SAR-BM3D. Furthermore, our model comes along with the additional advantage of simplicity and therefore well-suited to GPU programming. The GPU based implementation can conduct despeckling in less than 1s with state-of-the-art performance.



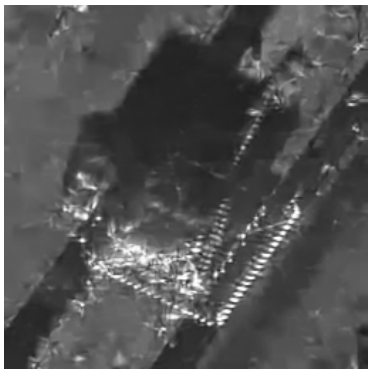
(a) Noisy image



(b) PPBit



(c) SAR-BM3D



(d) Ours with $\lambda_1 = 150$, $\lambda_2 = 0.03$

Figure 3.22: SAR image 2: sensor MiniSAR, Amplitude image, number of looks $L = 3$.

Generally speaking, training specialized FoE image prior model directly based on the despeckling task has the potential to achieve some improvements, as this is guaranteed not to decrease the training performance. This is subject to our future work.

3.2 JPEG artifacts suppression

The Block-wise Discrete Cosine Transform (B-DCT) based compression technique has been widely used in image and video coding standards. However, at high compression ratios, the coded images inevitably contain unwanted blocking artifacts, because each block is processed independently. In this section, we propose a novel variational model for

(a) Noisy image

(b) PPBit

(c) SAR-BM3D

(d) Ours with $\lambda_1 = 50$, $\lambda_2 = 0.07$

Figure 3.23: SAR image 3: sensor TerraSAR-X, Amplitude image, number of looks $L = 6.4$.

reducing blocking artifacts, which combines our learned image regularization model with the indicator function of the quantization constraint set (QCS). This new model leads to a generally hard non-convex optimization problem, and we make use of the iPiano algorithm to solve it efficiently. Numerical experiments show that in terms of PSNR and SSIM index, our deblocking approach with the learned FoE prior results in identical performance with the best published deblocking method across a range of compression levels, while our deblocking model comes along with the additional advantage of high efficiency.

3.2.1 Introduction

B-DCT has been successfully exploited in lossy image and video compression, such as JPEG, MPEG and H.263. Although it is well-known that the B-DCT coded images suffer from the so-called blocking effect, especially severe at low bit rate (i.e., high compression ratio), the classic JPEG standard is still the most popular lossy compression technique, and the dominant majority of the pictures circulated on the Internet is compressed by this standard. As a result of this fact, the development of advanced and efficient post-processing techniques to reduce the blocking artifacts is still a very active research topic. Recovery of B-DCT coded images has attracted a lot of research attention since early 1980s [85, 109], and therefore there are hundreds of publications to deal with this problem. Reviewing all the works of image deblocking is surely beyond the scope of this thesis. Instead we briefly pick out some representative works.

As a matter of fact, one can view the blocking artifacts from different aspects, which will lead to different solutions. Viewing the blocking artifacts as noise with certain structure, the deblocking problem corresponds to image denoising. There are different types of approaches proposed starting from this point of view: (1) image filtering technique [49, 85, 109]; (2) wavelet thresholding technique which originates from image denoising for Gaussian white noise [82, 136]; (3) deblocking via sparse representation using a general dictionary trained by the K-singular value decomposition (K-SVD) algorithm [69], whose original purpose is also for Gaussian noise removal; and (4) a recently introduced non-parametric image restoration model based on Regression Tree Fields (RTF), which defines a framework leveraging the advantages of existing deblocking methods by incorporating their predictions into the filed model, and therefore generates a state of the art for image deblocking [68].

On the other hand, the compression operation can be viewed as a degradation process, and many image restoration approaches are proposed to recover the original image, for example, algorithms based on projection onto convex sets (POCS) [50, 83]. In these methods, the prior information of the original image is defined as several convex sets, and an iterative projection algorithm is exploited to search the recovered image. Two commonly used convex sets are the QCS and the smoothness constraint set (SCS). The POCS-based methods are effective for reducing blocking artifacts. However, the basic difficulty is to construct appropriate convex sets to represent the image prior information of particular types.

In general, image restoration is a typical inverse problem. One of the most successful

approaches to solve inverse problems is to minimize a suitable energy functional whose minimizer provides a trade-off between a regularization term and a data term. Up to now, some researcher have investigated the regularization technique for image deblocking, see for example [15, 17, 24, 123]. These methods differ from each other in the regularization term or data term. Regarding the regularizer, the widely used Total Variation (TV), the Total Generalized Variation (TGV) of second order and a higher-order MRF image prior model based on the FoE have been investigated in [17, 24], [15] and [123] respectively. There are two methods to define the data term. One is the noise model [123], and the other is the indicator function of the QCS [15, 17, 24].

Generally speaking, the FoE image prior model is more expressive for natural image modeling than hand-crafted models, such as TV and TGV models, since it explicitly captures the statistical properties of natural images. For the data term, although the Gaussian noise model exploited in [123] can produce better recovery results than its predecessor, it is only a coarse approximate model for the quantization noise (blocking artifacts). However, the indicator function of the QCS exactly depicts the image compression process, as the QCS defines an accurate convex set of original image, and therefore, it is a more accurate data fidelity term. However, there is no work to combine the FoE-based prior term with the QCS-based data term up to now.

Our contributions. We introduce a novel variational model for image deblocking based on the FoE image prior model [111] and the QCS. This model incorporates better modes for both the regularization term and data term, and therefore defines a more accurate variational model for this task.

As mentioned before, image deblocking has a long history, and there exist numerous image deblocking methods. In order to demonstrate the effectiveness of our proposed method, we compare our approach with four representative methods: (1) recently published the best deblocking method based on RTF [68]†; (2) method based on dictionary learning and sparse representation [24]; and two similar variational approaches (3) FoE prior based method (different in data term) [123] and (4) TGV regularized model (different in regularization term) [15].

Numerical experiments show that in terms of PSNR and SSIM index, our variational model with the trained FoE image regularization term can obtain strongly competitive results to the best published image deblocking results [68], and significantly surpass the results of (1) a previous deblocking model also based on the FoE prior [123]; (2) TGV

†We consider the RTF model that relies on the output of the SADCT method [49]

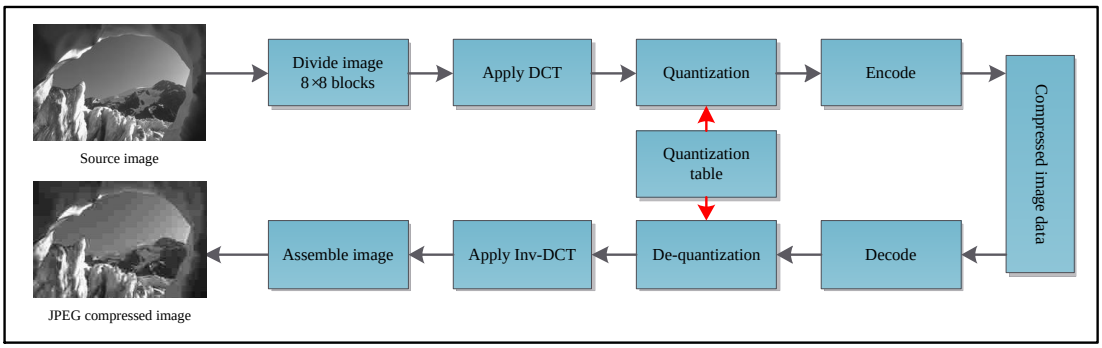


Figure 3.24: Schematic overview of the JPEG compression and decompression procedure

regularized deblocking model [15]; (3) deblocking method based on TV regularization and sparse representation [24] and (4) the image filtering based algorithm - SADCT [49].

In addition, our deblocking algorithm is relative easy to understand and implement, and well-suited to GPU parallel computation, which will make the inference procedure extremely efficient.

3.2.2 A novel variational model for image deblocking

We first give a brief overview of the JPEG compression process, and introduce the QCS, which will be employed in our model. Then we propose a variational model based on the FoE prior and the QCS, and introduce an efficient algorithm to solve the corresponding optimization problem.

3.2.2.1 JPEG compression and the QCS

In our work, we only consider gray-value images. Note that an extension of our model to color images is straightforward. Figure 3.24 illustrates all the steps of the JPEG compression and decompression procedure, see [130] for a more detailed explanation. In the step of quantization, the transformed DCT coefficients of each 8×8 block are point-wise divided by the quantization matrix, and then the values are rounded to integer, which is where the loss of data takes place, as the rounding operation is a mapping of "$\infty \to 1$". Given an integer number d, any number in the interval $[d-\frac{1}{2}, d+\frac{1}{2}]$ is a possible candidate for the original number which is rounded to d.

With the compressed image data, we only know the integer coefficient data $(d^q_{i,j})_{1\leq i,j\leq 8}$,

where q indicates a 8×8 block indexed by q, and the quantization matrix $(Q_{i,j})_{1 \leq i,j \leq 8}$. Therefore, the possible original DCT coefficients, which yield $(d^q_{i,j})$ in the quantization and rounding step are given by the interval

$$S^q_{i,j} = [Q_{i,j}(d^q_{i,j} - \frac{1}{2}), Q_{i,j}(d^q_{i,j} + \frac{1}{2})] \,.$$

This result is for the block q. For the full size image, we just need to repeat this result for each distinct block. All the intervals $S^q_{i,j}$ associated with each 8×8 block form the so-called QCS, which is simply a box constraint determining all possible source images.

In order to simplify the notation, the interval S is represented by two column vectors $a \in \mathbb{R}^N$ and $b \in \mathbb{R}^N$, which correspond to the lower and upper bounds of the intervals $S^p_{i,j}$, respectively.

3.2.2.2 Variational model for image deblocking

In our formulations, an image u of size $m \times n$ is written as a column vector $u \in \mathbb{R}^N$ with $N = m \times n$. We further define a highly sparse matrix $D \in \mathbb{R}^{N \times N}$, which makes Du equivalent to the B-DCT transform applied to the two-dimensional image u. Given the compressed image data, the QCS is given as the box constraint $S = [a, b]$, and the set of possible source image of the compression process is defined as

$$U = \{u \in \mathbb{R}^N | (Du)_p \in [a_p, b_p]\} \,.$$

Concerning the prior term, we directly make use of the trained FoE image prior model of the last Chapter. Namely, we exploited the same FoE model as shown in Figure 3.16. Recall that the FoE image prior model is formulated as

$$E_{FoE}(u) = \sum_{i=1}^{N_k} \alpha_i \phi(k_i * u) \,, \tag{3.29}$$

where $\phi(k_i * u) = \sum_{p=1}^{N} \phi((k_i * u)_p)$, N is the number of pixels in image u, N_k is the number of filters, k_i is a set of learned linear filters with the corresponding weights $\alpha_i > 0$, $k_i * u$ denotes the convolution of the filter k_i with a two-dimensional image u, and $\phi(\cdot)$ denotes the Lorentzian potential function $\phi(x) = \log(1 + x^2)$, which is derived from the student-t distribution.

Having this, now we can define our variational model, which reads as

$$\min_{u\in U} E(u) = \sum_{i=1}^{N_k} \alpha_i \phi(k_i * u)\,. \tag{3.30}$$

This is a constrained optimization problem, and it can be rewritten as

$$\min_{u} E(u) = \sum_{i=1}^{N_k} \alpha_i \phi(k_i * u) + \mathcal{I}_S(Du)\,, \tag{3.31}$$

where

$$\mathcal{I}_S(Du) = \begin{cases} 0 & \text{if } Du \in S, \\ \infty & \text{else}\,. \end{cases}$$

In this formulation, we directly exploit the convex set S instead of set U, as S is a box constraint, which is simpler than U.

3.2.2.3 Solving the variational deblocking model

An immediate question about the proposed image deblocking model is how to solve it. Due to the non-convexity of the prior term and the non-smoothness of the data term, it turns out that (3.31) is a very hard optimization problem. Fortunately, our proposed non-convex optimization algorithm - iPiano is applicable for solving this problem.

Recall that the iPiano algorithm is designed for a structured non-smooth non-convex optimization problem, which is composed of a smooth (possibly non-convex) function F and a convex (possibly non-smooth) function G:

$$\min_{u} h(u) = F(u) + G(u)\,. \tag{3.32}$$

Casting (3.31) in the form of (3.32), we see that $F(u) = \sum_{i=1}^{N_k} \alpha_i \phi(k_i * u)$ and $G(u) = \mathcal{I}_S(Du)$. It is clear that $F(u)$ is smooth and $G(u)$ is convex, and hence the iPiano algorithm can be applied. In order to use this algorithm, we need to calculate the gradient of F and the proximal map with respect to G. It is easy to check that

$$\nabla F(u) = \sum_{i=1}^{N_k} \alpha_i K_i^\top \phi'(K_i u)\,,$$

where K_i is an $N \times N$ highly sparse matrix, which is implemented as 2D convolution of the

image u with filter kernel k_i, i.e., $K_i u \Leftrightarrow k_i * u$, $\phi'(K_i u) = (\phi'((K_i u)_1), \cdots, \phi'((K_i u)_N))^\top \in \mathbb{R}^N$, with $\phi'(x) = 2x/(1+x^2)$.

The proximal map with respect to G is given as the following minimization problem

$$(I + \tau \partial G)^{-1}(\hat{u}) = \arg\min_u \frac{\|u - \hat{u}\|_2^2}{2} + \tau \mathcal{I}_S(Du)\,. \tag{3.33}$$

As DCT is a orthogonal transform, i.e., $D^\top D = DD^\top = \mathbf{I}$, then we have

$$\|Du - D\hat{u}\|_2^2 = (u-\hat{u})^\top D^\top D(u-\hat{u}) = (u-\hat{u})^\top (u-\hat{u}) = \|u-\hat{u}\|_2^2\,.$$

For problem 3.33, let

$$c = Du,\ \hat{c} = D\hat{u}\,.$$

Note that the connection between c and u (also $\hat{c}$ and $\hat{u}$) is a mapping of one-to-one.

It turns out that

$$\arg\min_u \frac{\|u-\hat{u}\|_2^2}{2} + \tau \mathcal{I}_S(Du) \Longleftrightarrow \arg\min_c \frac{\|c-\hat{c}\|_2^2}{2} + \tau \mathcal{I}_S(c)\,.$$

Obviously, the solution for the minimization problem of right side is given as the following point-wise projection onto the interval

$$\tilde{c}_p = \begin{cases} \hat{c}_p & \text{if } \hat{c}_p \in S_p = [a_p, b_p] \\ b_p & \text{if } \hat{c}_p > b_p \\ a_p & \text{if } \hat{c}_p < a_p\,. \end{cases}$$

Finally, the the solution of u is given as $\tilde{u} = D^\top \tilde{c}$.

In experiments, we found that the deblocking results generated by the variational model Equation (3.31) is somehow over-smoothing, and therefore inferior PSNR and SSIM index values relative to the RTF based model [68]. The reason lies in the data term involved in the variational model (3.31), which is the indicator function of an interval. On the one hand, the indicator function makes no difference for the points lying in the interval [‡]; and on the other hand, the FoE image regularization is essentially a smoothing term. Therefore, the variational model (3.31) prefers those smoother solutions lying in the boundary of the interval.

In order to alleviate the over-smoothing effect, a natural idea is to modify the data

[‡]Note that the input blocky image is the median point of this interval.

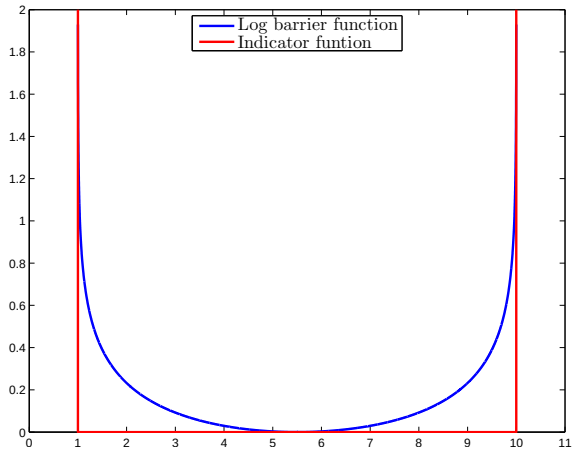


Figure 3.25: The log-barrier function and the indicator function of an interval.

term such that it can push away the minimizer from the boundary of the interval, i.e., the penalty will increase when the solution is approaching the boundary of the interval. A feasible solution is to exploit the following log-barrier function, which is typically used to approximate the indicator function $\mathcal{I}_{S_p=[a_p,b_p]}$,

$$\psi_p(x) = -\frac{1}{t}\left(\log(x - a_p) + \log(b_p - x)\right) ,$$

where t is a parameter to control the approximation, the larger, the better. An illustrative example of the log-barrier function is shown in Figure 3.25. Then the modified version of our proposed variational model is given as

$$\arg\min_u E(u) = \sum_{i=1}^{N_k} \alpha_i \phi(K_i u) + \psi(Du) , \tag{3.34}$$

where $\psi(Du) = \sum_{p=1}^{N} \psi_p((Du)_p)$. In order to make use of the iPiano algorithm to solve (3.34), we need to recalculate the proximal map with respect to the new function $G(u) = \psi(Du)$. By using the orthogonality of the DCT transform, the solution is given as $u = D^\top \tilde{c}$, with $\tilde{c}$ is the solution of the following point-wise subproblem

$$\tilde{c}_p = \arg\min_c \frac{\|c - \hat{c}_p\|_2^2}{2} + \frac{-\tau}{t}\left(\log(c - a_p) + \log(b_p - c)\right) , \tag{3.35}$$

where $\hat{c} = D\hat{u}$. In order to solve the above minimization problem, we calculate the gradient

with respect to c, then let it equal to 0. Let $k = \frac{\tau}{t}$,

$$c - \hat{c} + k(\frac{-1}{c-a} + \frac{-1}{c-b}) = 0$$

Multiplying both sides with $(c-a)(c-b)$, it is given as

$$(c - \hat{c})(c-a)(c-b) + k(b - c + a - c) = 0$$

$$c^3 - (a + b + \hat{c})c^2 + (ab + a\hat{c} + b\hat{c} - 2k)c + (a+b)k - ab\hat{c} = 0$$

Let $A = 1,\ B = -(a + b + \hat{c}),\ C = ab + a\hat{c} + b\hat{c} - 2k,\ D = (a+b)k - ab\hat{c}$,

the above cubic equation is given as

$$Ac^3 + Bc^2 + Cc + D = 0\,. \tag{3.36}$$

For the general cubic equation 3.36, the general formula for the roots, in terms of the coefficients, is as follows: [§]

$$c_i = \frac{-1}{3A}(B + s_i Q + \frac{\Delta_0}{s_i Q}), \quad i \in \{1, 2, 3\}\,,$$

where

$$s_1 = 1, \quad s_2 = \frac{-1 + i\sqrt{3}}{2}, \quad s_3 = \frac{-1 - i\sqrt{3}}{2}$$

are the three cube roots of unity, and where

$$Q = \sqrt[3]{\frac{\Delta_1 + \sqrt{\Delta_1^2 - 4\Delta_0^3}}{2}}$$

with

$$\Delta_0 = B^2 - 3AC$$

$$\Delta_1 = 2B^3 - 9ABC + 27A^2D$$

However, for the subproblem 3.35, the solution must be unique, i.e., we need to identify the correct solution from three roots c_i. For our specific problem, we succeed, and the wanted solution is given by c_3 with $s_3 = \frac{-1-i\sqrt{3}}{2}$.

For example, choose

$\triangleright a = 5,\ b = 10,\ k = 0.1,\ \hat{c} = 18 \Longrightarrow c_1 = 4.9923,\ c_2 = 18.0201,\ c_3 = 9.9876$, only

[§] `http://en.wikipedia.org/wiki/Cubic_function`

with $c_3 \in (5, 10)$.

$\triangleright a = 5,\ b = 10,\ k = 0.1,\ \hat{c} = -18 \Longrightarrow c_1 = -18.0079,\ c_2 = 10.0036,\ c_3 = 5.0043$, only with $c_3 \in (5, 10)$.

$\triangleright a = -5,\ b = 10,\ k = 1,\ \hat{c} = -18 \Longrightarrow c_1 = -18.1118,\ c_2 = 10.0358,\ c_3 = -4.9239$, only with $c_3 \in (-5, 10)$.

3.2.3 Experimental results

In order to evaluate the performance of our proposed variational image deblocking model based on a set of learned filters, we applied the proposed model to suppress the blocking artifacts in JPEG compressed images with different compression quality q. The compressed images are quantized using the quantization matrix $Q_q = \text{round}(50Q_{50}/q)$, where Q_{50} is the standard quantization matrix [130] given as

$$Q_{50} = \begin{bmatrix} -16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

As we know, the image deblocking performance of a specific method varies greatly for different image contents, in order to make a fair comparison with other competing methods, we conducted deblocking experiments over a standard test dataset - BSD500, consisting of 200 natural images, which was firstly used in [68] for deblocking performance evaluation.

We followed the test procedure in [68]. The test images were converted to gray-value, and scaled by a factor of 0.5, resulting images of size 240×160. We distorted the images by JPEG blocking artifacts. We considered three compression quality settings $q = 10, 20$ and 30 for the JPEG encoder.

For our proposed variational model, we investigate two different data terms: (1) the indicator function and (2) the log-barrier function. We compared our models with four representative methods: (1) the published best deblocking method based on RTF [68]; (2) method based on dictionary learning and sparse representation (SR) [24]; as well as

q	JPEG decoder	TGV [15] deblocking	Dictionary SR [24]	FoE [123] deblocking	RTF [68]	Ours (log-barrier)	Ours (indicator)
10	26.59/76.10	26.96/77.80	27.15/77.87	27.40/79.10	**27.68/79.47**	**27.68/79.66**	27.34/78.32
20	28.77/84.47	29.01/85.03	29.03/83.76	29.54/86.47	**29.83/86.68**	**29.86/87.02**	29.47/85.60
30	30.05/88.04	30.25/88.33	30.13/86.35	30.86/89.63	**31.14/89.85**	**31.18/90.14**	30.76/88.93

Table 3.8: JPEG deblocking results for natural images in terms of PSNR value and SSIM index ($\times 100$). We compare our method to four representative image deblocking methods, including the best published deblocking method based on RTF [68]. We highlight the state of the art results.

	TGV [15] deblocking	Dictionary SR [24]	FoE [123] deblocking	SADCT [49]	Ours (log-barrier)	Ours (indicator)
T(s)	7.42	7.23	50	3.53	5.57 (**0.062**)	5.29 (**0.058**)
PSNR	26.96	27.15	27.40	27.43	27.68	27.34
SSIM	77.80	77.87	79.10	78.64	79.66	78.32

Table 3.9: Typical run time (CPU computation) of the deblocking methods for a 240×160 image ($q = 10$) on a server (Intel X5675, 3.07GHz). The highlighted number is the run time of the GPU implementation.

two variational approaches (3) FoE prior based method [123] and (4) TGV regularized model[15]. For the RTF based model, we exploited the PsnrRTF$_{\text{SADCT}}$ system, which includes SA-DCT as a base method and is optimized for the PSNR performance measure. For the sparse representation based method, we used the dictionary model without TV regularization term. The deblocking performance is reported with the commonly used objective measurements: PSNR and SSIM index [131].

The average PSNR and SSIM results of the considered methods over the test dataset are summarized in Table 3.8. In terms of the objective measurements, one can see that (1) our proposed deblocking method with the log-barrier data term can lead to better performance compared to the model with the indicator function, (2) the log-barrier data term based model has surpassed three competing methods and has achieved identical performance with the best one [68]. We present the deblocking results of our log-barrier based model and other competing methods in Figure 3.26 for the case of compression quality $q = 10$, in Figure 3.27 for the compression quality $q = 20$, and Figure 3.28 for the compression quality $q = 30$.

Concerning the visual inspection, generally speaking, in the case of relatively high compression rate, e.g., $q = 10$, the results given by TGV based method are over-smoothed as the minimizer of the TGV regularized model is piece-wise affine. The dictionary based

	240×160	480×320	512×512	1024×1024
Ours(ms)	62.2	153	230	822
SADCT(ms)	3530	11790	17650	70978

Table 3.10: The run time of our deblocking approach (log-barrier based model) for different image size by using GPU computation (based on NVIDIA Geforce GTX 780Ti). We also present the CPU computation time for the SADCT algorithm (based on Intel X5675, 3.07GHz), which is the strongest competitor in terms of run time.

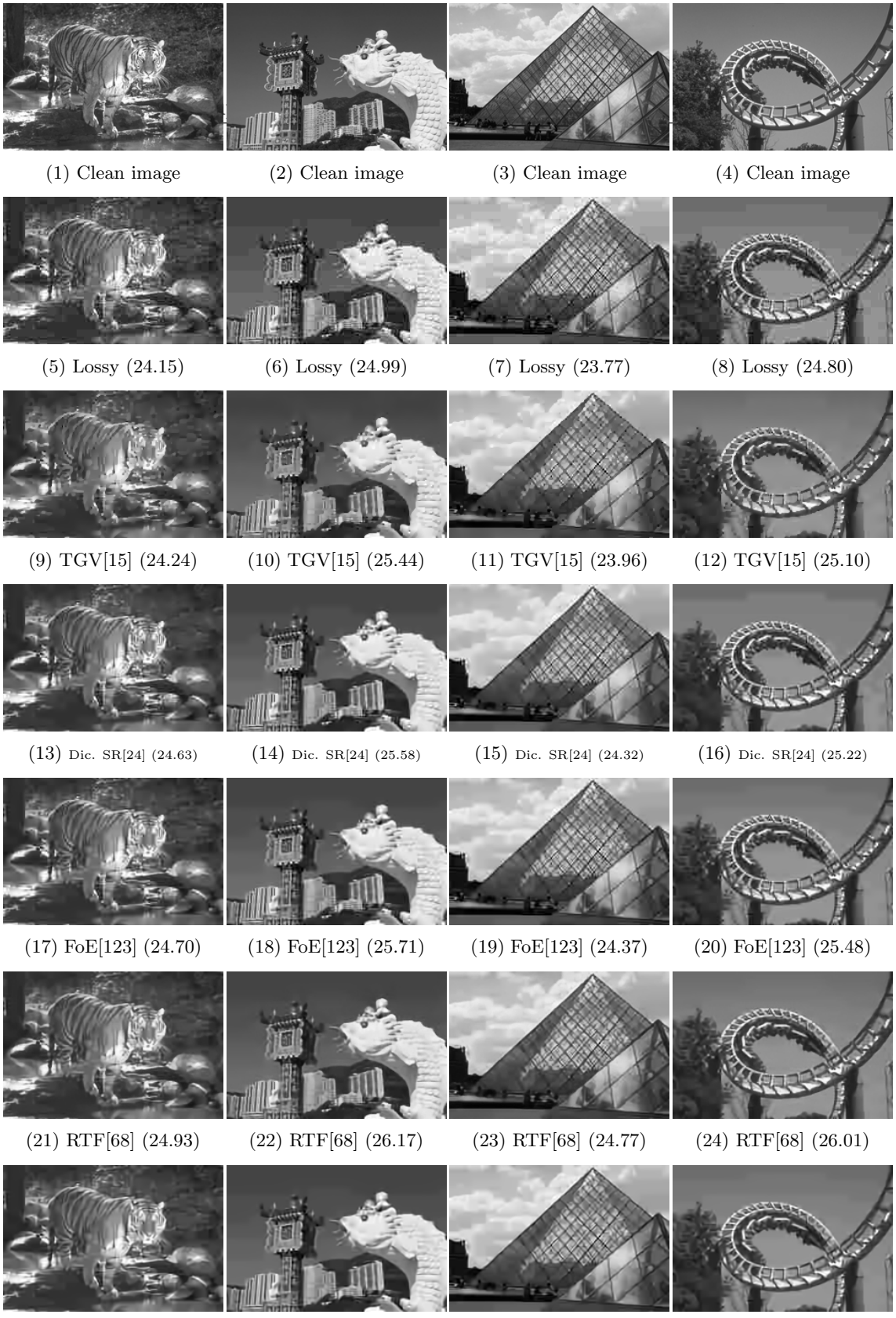
method frequently fails to remove the blocking artifacts in the homogeneous regions, such as in the sky. The RTF based method, FoE prior regularized model and our approach usually generate visually plausible deblocking results, but the FoE prior regularized model in previous work [123] is prone to produce slightly blurred edges, thus inferior PSNR results. See Figure 3.26 for example.

After having a closer look at the results obtained by our log-barrier based model, we also find that it is less effective to remove the blocking artifacts in the homogeneous regions, e.g., in the sky. However, the indicator function based method, which usually provides over-smoothing results (thus inferior PSNR and SSIM values) can address this problem better, see Figure 3.29 for an example. We see that the indicator function based method even though owns inferior PSNR and SSIM values, it can more effectively suppress the blocky artifacts in the sky. Meanwhile, we also see that the SADCT method also usually fails to remove the blocking artifacts in the sky. It seems for the task of JPEG deblocking, the PSNR and SSIM index are not the best quality measurement.

Run time: We conducted a direct run time comparison for different deblocking algorithms based on CPU implementation. In Table 4.2, we show the average run time of the considered deblocking methods on 240×160 images. We use the iPiano algorithm to solve our proposed models, and typically it takes 40 iterations to converge to a stationary point. For the investigated algorithms, we make use of the codes provided by the authors as is[¶].

Our proposed deblocking model comes along with the additional advantage of simplicity, as it solely involves convolution of filters with an image. Therefore, our model is well-suited to GPU parallel computation. Table 3.10 presents the GPU computation time of our log-barrier function based deblocking model for different image dimensions. Note that for GPU computation, it is not appropriate to solve the subproblem Equation

[¶]However, we are not able to present the runtime of RTF method [68], as its implementation is not available. But we know it relies on the output of the SADCT method, thus its computation time is the runtime of SADCT plus the execution time of RTF (i.e., RTF based method is slower than the SADCT algorithm).



(1) Clean image (2) Clean image (3) Clean image (4) Clean image

(5) Lossy (24.15) (6) Lossy (24.99) (7) Lossy (23.77) (8) Lossy (24.80)

(9) TGV[15] (24.24) (10) TGV[15] (25.44) (11) TGV[15] (23.96) (12) TGV[15] (25.10)

(13) Dic. SR[24] (24.63) (14) Dic. SR[24] (25.58) (15) Dic. SR[24] (24.32) (16) Dic. SR[24] (25.22)

(17) FoE[123] (24.70) (18) FoE[123] (25.71) (19) FoE[123] (24.37) (20) FoE[123] (25.48)

(21) RTF[68] (24.93) (22) RTF[68] (26.17) (23) RTF[68] (24.77) (24) RTF[68] (26.01)

(25) Ours (24.94) (26) Ours (26.03) (27) Ours (24.86) (28) Ours (25.95)

Figure 3.26: Image deblocking results for images compressed by JPEG encoder with the quality $q = 10$.

(1) Clean image (2) Clean image (3) Clean image (4) Clean image

(5) Lossy (32.73) (6) Lossy (32.12) (7) Lossy (26.58) (8) Lossy (27.89)

(9) TGV[15] (33.18) (10) TGV[15] (32.30) (11) TGV[15] (27.34) (12) TGV[15] (27.88)

(13) Dic. SR[24] (29.94) (14) Dic. SR[24] (31.86) (15) Dic. SR[24] (27.74) (16) Dic. SR[24] (28.00)

(17) FoE[123] (33.94) (18) FoE[123] (32.72) (19) FoE[123] (27.45) (20) FoE[123] (28.40)

(21) RTF[68] (34.12) (22) RTF[68] (32.88) (23) RTF[68] (28.77) (24) RTF[68] (28.72)

(25) Ours (34.12) (26) Ours (32.86) (27) Ours (28.36) (28) Ours (28.69)

Figure 3.27: Image deblocking results for images compressed by JPEG encoder with the quality $q = 20$.

(1) Clean image (2) Clean image (3) Clean image (4) Clean image

(5) Lossy (28.68) (6) Lossy (30.26) (7) Lossy (29.43) (8) Lossy (32.41)

(9) TGV[15] (28.92) (10) TGV[15] (30.60) (11) TGV[15] (30.26) (12) TGV[15] (33.41)

(13) Dic. SR[24] (29.21) (14) Dic. SR[24] (30.77) (15) Dic. SR[24] (29.79) (16) Dic. SR[24] (32.71)

(17) FoE[123] (29.52) (18) FoE[123] (31.27) (19) FoE[123] (30.34) (20) FoE[123] (33.63)

(21) RTF[68] (29.80) (22) RTF[68] (31.43) (23) RTF[68] (31.00) (24) RTF[68] (34.35)

(25) Ours (29.85) (26) Ours (31.57) (27) Ours (30.80) (28) Ours (34.10)

Figure 3.28: Image deblocking results for images compressed by JPEG encoder with the quality $q = 30$.

(a) Lossy JPEG $q = 10$ (27.14/90.88)

(b) SADCT (27.87/92.31, CPU: 23.5s)

(c) Indicator based model (27.43/90.69, GPU: 0.306s)

(d) Log-barrier based model (28.11/92.88, GPU: 0.324s)

Figure 3.29: Our proposed model with the log-barrier data term and the SADCT method fail to remove the bocking artifacts in the sky, while the indicator function based model succeeds, in despite of inferior PSNR and SSIM values. The corresponding run time is also reported: (1) CPU implementation for the SADCT method and GPU implementation for our proposed models; (2) image size 768×512.

(3.35) by using the direct method, because it involves the calculation of complex number. Instead, we make use of the Newton's method to solve this subproblem. In practice,we find that this scheme has a quite fast convergence (usually less than 10 iterations.)

Note that the TGV based method [15] and the previous FoE prior based method [123] are also suited for GPU parallel computation, but we didn't implement it.[‖] Also note

[‖]In the work of [15], for the TGV based deblocking algorithm (1000 iterations), the authors quoted a GPU computation time of 1.2s for the image size 512×512 based on NVIDIA Geforce GTX 580.

that the RTF based approach, e.g., the PsnrRTFsadct system relies on the output of the SADCT algorithm [49]. However, our proposed model is an independent algorithm, which dose not rely on the output of any existing methods.

3.2.4 Discussion

In this section, we have proposed a novel variational model for image deblocking based on (1) an expressive image prior model - FoE model and (2) the indicator function of the QCS, which exactly defines a convex set of possible source images given the compressed image. This new variational model poses a generally demanding non-convex minimization problem and we introduced our proposed algorithm iPiano to solve it. We directly applied our trained FoE image prior model in the case of Gaussian denoising to the deblocking model. Experimental results on a set of natural images demonstrate that our deblocking model with the learned FoE prior model leads to identical performance with the best published deblocking method. Our proposed model comes along with the additional advantage that the inference is extremely efficient. The GPU based implementation can conduct image deblocking in less than 1s for image sizes up to 1024×1024.

3.3 Other image restoration problems

In the framework of variational models, it is straightforward to exploit the trained FoE image prior model for additional image restoration problems, such as non-blind image deconvolution, single (or multi-frame) image super resolution and image inpainting. In this section, we investigate these additional image restoration problems by using our trained image prior model.

3.3.1 Non-blind image deconvolution

The FoE image prior regularized deconvolution model is defined as

$$u^* = \arg\min_u E(u) = \sum_{i=1}^{N_k} \alpha_i \phi(K_i u) + \frac{\lambda}{2} \|h * u - f\|_2^2 \,, \tag{3.37}$$

where the noisy input image f is corrupted by a linear kernel h, which is assumed to be known in our work, and λ is again a trade-off parameter to be tuned in the experiments. Regarding the FoE image prior term, we still make use of the model shown in Figure 3.16.

The corresponding minimization problem Equation (3.37) also can be solved by the iPiano algorithm. In order to make use of the iPiano algorithm, we just need to compute the proximal mapping operator with respect to $G(u) = \frac{\lambda}{2}\|h * u - f\|_2^2$, i.e., we need to solve the following subproblem

$$\begin{aligned} u &= \arg\min_u \frac{\|u - \hat{u}\|_2^2}{2\tau} + \frac{\lambda}{2}\|h * u - f\|_2^2 \\ &\iff u - \hat{u} + \tau\lambda\bar{h} * (h * u - f) = 0\,. \end{aligned} \tag{3.38}$$

Assuming the above convolution operation is performed with periodic boundary condition, according to the well-known convolution theorem, the corresponding linear system can be rewritten as following problem by using the FFT transform

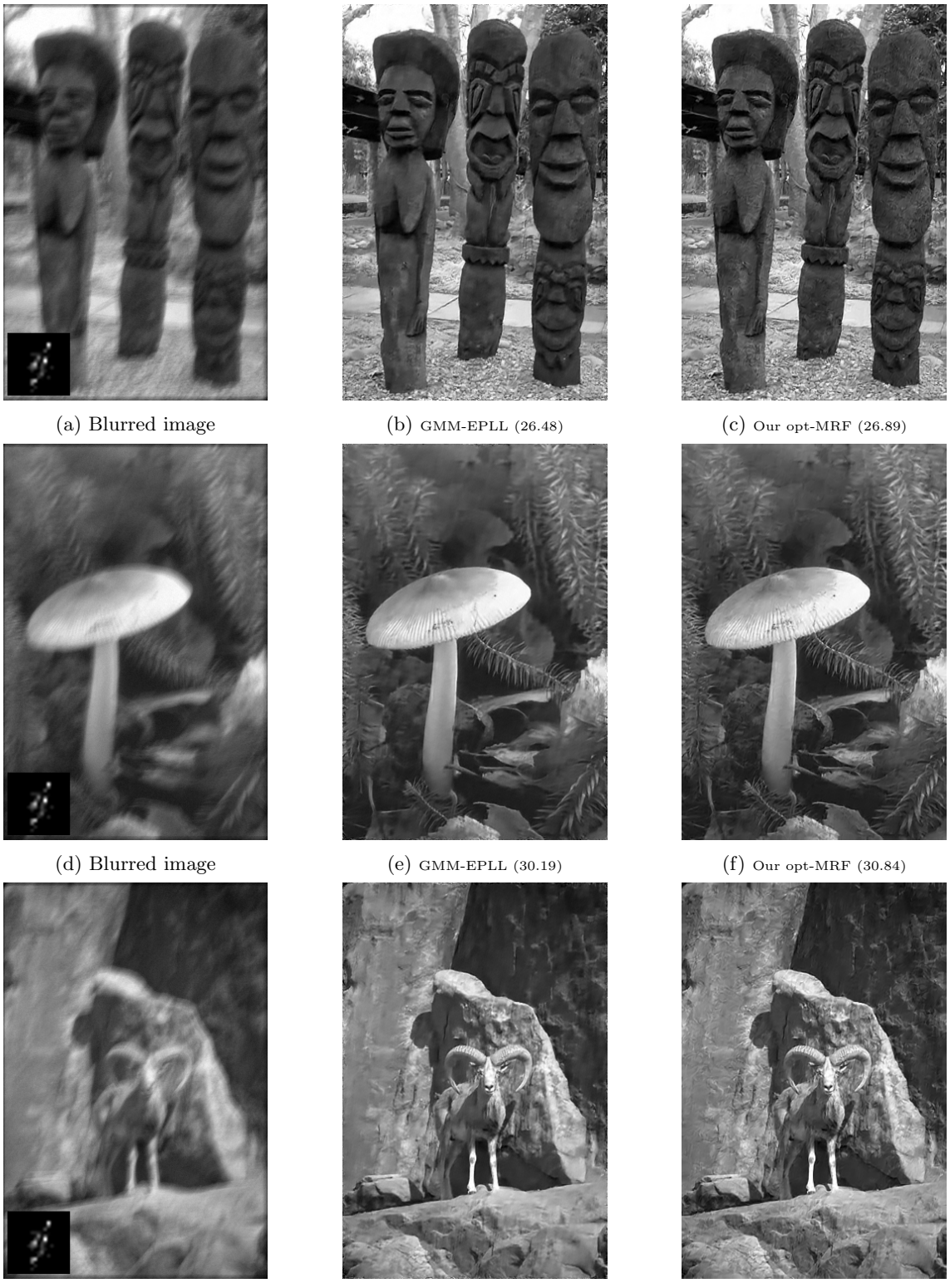
$$\begin{aligned} \mathcal{F}(u) - \mathcal{F}(\hat{u}) + \tau\lambda\mathcal{F}(h)^* \odot (\mathcal{F}(h) \odot \mathcal{F}(u) - \mathcal{F}(f)) &= 0 \\ \iff \mathcal{F}(u) &= \frac{\tau\lambda\mathcal{F}(h)^* \odot \mathcal{F}(f) + \mathcal{F}(\hat{u})}{\tau\lambda\mathcal{F}(h)^* \odot \mathcal{F}(h)} \\ \iff u &= \mathcal{F}^{-1}\left(\frac{\tau\lambda\mathcal{F}(h)^* \odot \mathcal{F}(f) + \mathcal{F}(\hat{u})}{\tau\lambda\mathcal{F}(h)^* \odot \mathcal{F}(h) + 1}\right), \end{aligned} \tag{3.39}$$

where $\mathcal{F}(\cdot)$ and $\mathcal{F}^{-1}(\cdot)$ denote the FFT and inverse FFT, respectively; $\odot$ means the point-wise operation, and the division operator is also understood in the point-wise manner; $\mathcal{F}(h)^*$ denotes the complex conjugate of $\mathcal{F}(h)$.

Now we can apply the variational model (3.37) to the image deconvolution problem. To present a comparison, we conducted the same experiment as [143]: convolved 68 image (the same as in previous sections) with the blur kernels from [71], then added 1% white Gaussian noise. We tried to restore the clean images by using GMM-EPLL framework and our opt-MRF model. Results are shown in Table 3.11 and Figure 3.30 - Figure 3.31. We can see that both methods present pretty good results. Figure 3.30 shows deblurring results for the convolution kernel 2 19×19 from [71]. Figure 3.31 shows deblurring results for the convolution kernel 1 17×17. All the results are compared with GMM-EPLL [143].

	GMM-EPLL	Opt-MRF
Kernel 1 17×17	28.99	29.69
Kernel 2 19×19	29.95	30.49

Table 3.11: Deconvolution results for 68 test images (average PSNR).

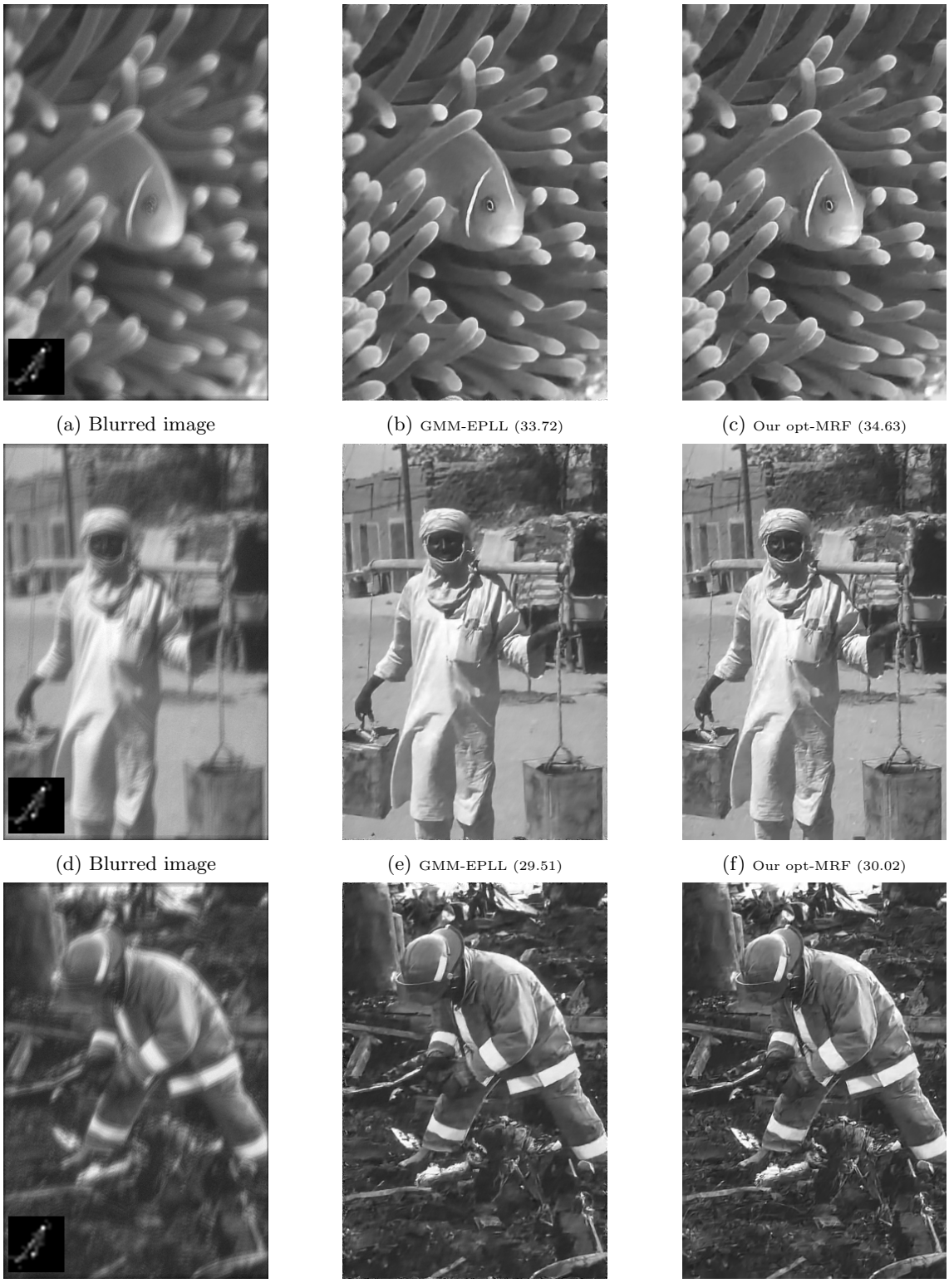


(a) Blurred image (b) GMM-EPLL (26.48) (c) Our opt-MRF (26.89)

(d) Blurred image (e) GMM-EPLL (30.19) (f) Our opt-MRF (30.84)

(g) Blurred image (h) GMM-EPLL (27.55) (i) Our opt-MRF (28.02)

Figure 3.30: Delurring results using GMM-EPLL and our opt-MRF model for Kernel 2 19×19.

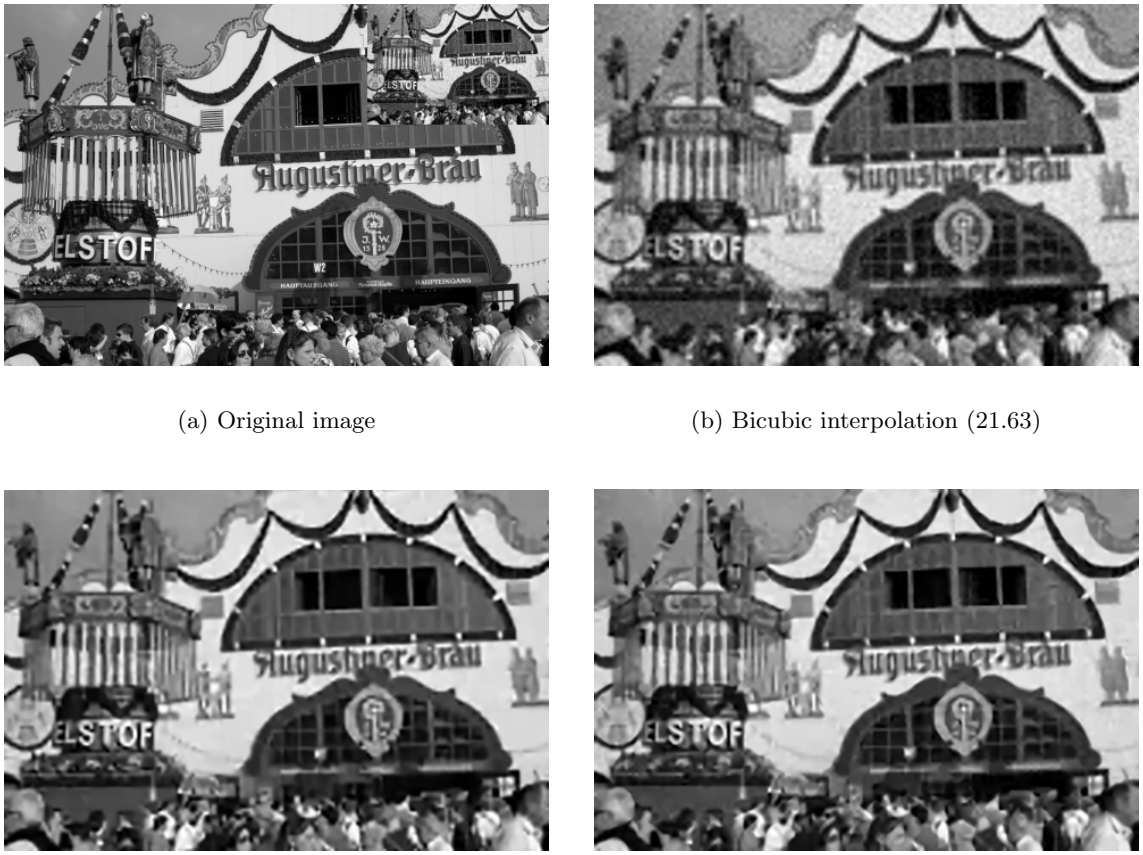


(a) Blurred image (b) GMM-EPLL (33.72) (c) Our opt-MRF (34.63)

(d) Blurred image (e) GMM-EPLL (29.51) (f) Our opt-MRF (30.02)

(g) Blurred image (h) GMM-EPLL (27.26) (i) Our opt-MRF (27.95)

Figure 3.31: Delurring results using GMM-EPLL and our opt-MRF model for Kernel 1 17×17.



(a) Original image

(b) Bicubic interpolation (21.63)

(c) GOAL [60] (22.45)

(d) Ours: $\log(1 + z^2)$ (22.57)

Figure 3.32: Single image super-resolution results for magnifying a noisy low resolution image by a factor of 3. The low resolution image was degraded by Gaussian noise with $\sigma = 8$. The numbers shown in the brackets refer to PSNR values w.r.t. the clean image.

3.3.2 Image super resolution

The learned FoE image prior is also applicable for the task of image super resolution. In our work, we consider two cases (1) single image super resolution, where we upsample a single noisy low resolution image by a factor of $s > 1$ and (2) multi-frame super resolution, where we resolve a high resolution image from a low resolution image sequence (e.g., from a video).

3.3.2.1 Single image super resolution

For the problem of single image super-resolution by using our trained image regularizers, we are to solve the following optimization problem

$$\arg\min_u E(u) = \sum_{i=1}^{N_k} \alpha_i \phi(K_i u) + \frac{\lambda}{2}\|Hu - f\|_2^2\,, \tag{3.40}$$

where the image regularization term is again given as the FoE image prior model shown in Figure 3.16, the linear operator H is constructed by a decimation operator Φ and a blurring operator B, i.e., $H = \Phi B$.

The first question about the above optimization problem (3.40) is how to solve it by using the iPiano algorithm. The first option is to set $F(u) = \sum_{i=1}^{N_k} \alpha_i \phi(K_i u)$ and $G(u) = \frac{\lambda}{2}\|Hu - f\|_2^2$. Then we need to compute the proximal mapping with respect to G, which is given as the following minimization problem.

$$\begin{aligned} u &= \arg\min_u \frac{\|u - \hat{u}\|_2^2}{2\tau} + \frac{\lambda}{2}\|Hu - f\|_2^2 \\ \Longleftrightarrow & u - \hat{u} + \tau\lambda H^\top (Hu - f) = 0 \\ \Longleftrightarrow & u = \left(\tau\lambda H^\top H + \mathbf{I}\right)^{-1} (\tau\lambda H^\top f + \hat{u})\,. \end{aligned} \tag{3.41}$$

Even though this subproblem leads to a closed-from solution, solving the above linear system is quite computationally expensive in practice. Therefore, it is not the best choice to solve the optimization problem (3.40) in this manner.

In our work, we consider another option by setting $F(u) = \sum_{i=1}^{N_k} \alpha_i \phi(K_i u) + \frac{\lambda}{2}\|Hu - f\|_2^2$ and $G(u) = 0$. Now it is straightforward to compute the gradient of F with respect to u, which is simple given as

$$\nabla_u F(u) = \sum_{i=1}^{N_k} \alpha_i K_i^\top \phi'(K_i u) + \lambda H^\top (Hu - f)\,, \tag{3.42}$$

and the proximal mapping with respect to G is given as

$$u = (\mathbf{I} + \tau\partial G)^{-1}(\hat{u}) \Longleftrightarrow u = \hat{u}\,. \tag{3.43}$$

In order to perform a better comparison with the latest analysis model GOAL [60], we conducted the same single image super-resolution experiment. We artificially created

a low resolution image by downsampling a ground-truth image by a factor of 3 using bicubic interpolation. Then the low resolution image was corrupted by Gaussian noise with $\sigma = 8$. We magnified the noisy low resolution image by the same factor using (a) bicubic interpolation, (b) the GOAL method [60], (c) our learned model, respectively. Figure 3.32 shows the results for different methods. One can see that two analysis models present similar results, which are visually and quantitatively better than the bicubic method.

3.3.2.2 Multi-frame super resolution

In this subsection, we consider the problem of multi-frame super resolution by using the variational framework proposed by Unger et al. [129]. In the original work [129], the widely used total variation (TV) regularizer was employed. While the TV regularization has the advantage that it allows sharp edges in the image, it can not reflect the complex statistics of natural images. In our work, we replace the TV regularization with our trained image regularizer.

In our work, we embed our learned FoE prior into the variational super resolution framework proposed in the previous work [129], resulting in the following variational model for multi-frame image super resolution problem.

$$\arg\min_u E(u) = \sum_{i=1}^{N_k} \alpha_i \phi(K_i u) + \lambda \sum_{i=1}^{n} \|\Phi B W_i u - f_i\|_{1,\varepsilon}, \tag{3.44}$$

where $\phi(K_i x) = \sum_{p=1}^{N} \phi((K_i x)_p)$, $\phi(z) = \log(1 + z^2)$ and $|z|_{1,\varepsilon} = \begin{cases} \frac{z^2}{2\varepsilon} & \text{if} |z| \leq \varepsilon \\ |z| - \frac{\varepsilon}{2} & \text{if} |z| > \varepsilon. \end{cases}$

In (3.44), the first term is our learned filters-based MRF prior shown in Figure 3.16. The second term consists of the input multi-frames f_i, where the linear operator Φ, B and W denote down-sampling, blurring and warping operators. The warping operator W is derived from the computed optical flow. More details refer to [129].

We still make use of the iPiano algorithm to solve the minimization problem (3.44) by setting $F(u) = E(u)$ and $G(u) = 0$. The gradient of F is defined as

$$\nabla_u F(u) = \sum_{i=1}^{N_k} \alpha_i K_i^\top \phi'(K_i u) + \lambda \sum_{i=1}^{n} (\Phi B W_i)^\top \varphi(\Phi B W_i u - f_i), \tag{3.45}$$

where $\varphi(z) = \frac{z}{\max\{\varepsilon, |z|\}}$.

We present an example to illustrate the performance of our proposed FoE image prior

(a) frame 1 (b) frame 8 (c) frame 12 (d) frame 18

(e) frame 22 (f) frame 27 (g) frame 31 (h) frame 36

Figure 3.33: Input low resolution frames

based multi-frame super resolution approach. In this experiment, we moved the "Hautlauer" AD page slowly, and then recorded this scene. We selected a sequence consisting of 36 images to conduct the super resolution task. In order to reduce the computation cost, we only chose the cropped region containing the moving AD page, which we are interested in. The input low resolution frames are shown in Figure 3.33. The super resolved result for this sequence is shown in Figure 3.34, together with closeup of the corresponding images. One can easily see the improvements brought by our proposed multi-frame super resolution approach.

3.3.3 Image inpainting

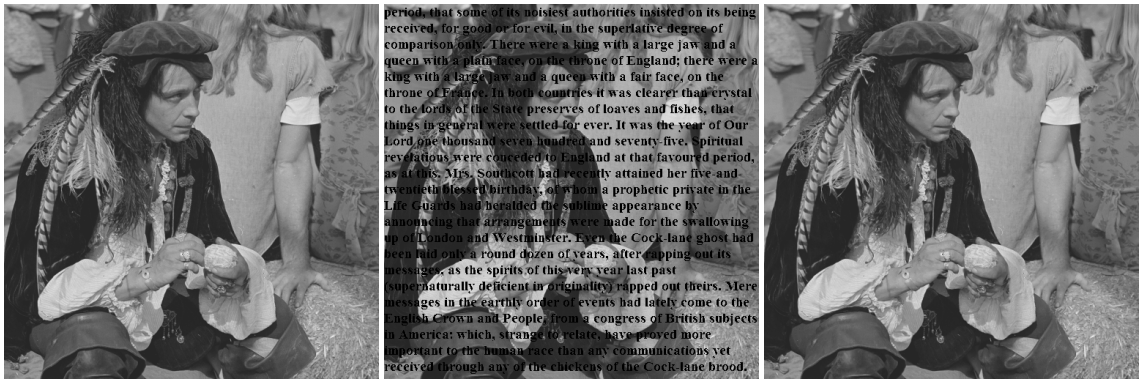
Image inpainting is the process of filling in the lost image data such that the resulting image is visually appealing. Typically, the positions of the pixels to be filled up are assumed to be known. The image inpainting model based on our trained image regularizer is defined

(a) original low resolution image (b) Bi-cubic (c) MRF prior based

Figure 3.34: Super-resolved frame for the Hautlauer AD

as

$$\arg\min_u E(u) = \sum_{i=1}^{N_k} \alpha_i \phi(K_i u) + \frac{\lambda}{2} \|Hu - f\|_2^2 \,, \tag{3.46}$$



(a) Clean image (b) Corrupted image with overlaying text (c) Inpainting result (PSNR: 37.27)

Figure 3.35: Overlaying text removal by using the variational model with our trained image regularizer.

where the image regularization term is again given as the FoE image prior model shown in Figure 3.16, and the linear operator H is simply a sampling matrix. Each row of H contains exactly one entry equal to one, and its position indicates a pixel with given value. The parameter λ corresponds to joint inpainting and denoising, and the choice $\lambda \to +\infty$ means pure inpainting. In our experiment since we assumed the test images are noise free, we empirically selected $\lambda = 10^3$ for pure inpainting.

We still employ the iPiano algorithm to solve the corresponding minimization problem (3.46) by setting $F(u) = \sum_{i=1}^{N_k} \alpha_i \phi(K_i u)$ and $G(u) = \frac{\lambda}{2}\|Hu - f\|_2^2$. In this case, the proximal mapping operator with respect to G can be easily calculated.

$$u = (I + \tau \partial g)^{-1}(\hat{u}) \Longleftrightarrow u_p = \begin{cases} \hat{u}_p & \text{if} \quad p \in I \\ \frac{\tau\lambda f_p + \hat{u}_p}{1+\tau\lambda} & \text{else} \end{cases}. \tag{3.47}$$

where I denotes the set of indices of the inpainting domain.

In this subsection, we consider two typical image inpainting tasks. The first one is to remove the overlaying text from the corrupted image. An illustrative example is shown in Figure 3.35. One can see that the overlaying text is removed successfully.

The second one is to fill up the random missing pixels in an image. We destroyed the ground-truth " Lena" image (512×512) artificially by randomly masking 60% and 90% of the entire pixels, respectively. Then we reconstructed the incomplete image using our

(a) 60% missing pixels (b) GOAL [60] (35.6) (c) Our opt-MRF (37.79)

(d) 90% missing pixels (e) GOAL [60] (28.57) (f) Our opt-MRF (29.10)

Figure 3.36: Inpainting results of our opt-MRF model and the approach in [60] for the "Lena" image 512×512 from 40% and 10% pixels, respectively.

learned analysis model - $\log(1 + z^2)$-based model. In order to present a comparison, we also give the inpainting result of the GOAL model [60]. Inpainting results are shown in Figure 3.36. From Figure 3.36, one can see that the result of our learned analysis model achieves better results relative to the GOAL model in terms of PSNR value, while both methods provide visually similar results.

3.4 Discussion

In this chapter, we have thoroughly exploited the capability of our trained image prior model for a class of image restoration problems, including image denoising for various of

noise types, JPEG deblocking, image super resolution, image deconvolution and image inpainting. Even though our image prior model is discriminatively trained based on the problem of Gaussian denoising, the resulting image prior model is not heavily tailored to the Gaussian denoising task. In contrast, extensive experiments demonstrate that the trained image prior model generalize well for many image restoration problems, and the corresponding variational models can usually generate state-of-the-art results for the investigated problems. Therefore, the trained image prior model can be indeed treated as an image regularization term for a variety of image restoration problems.

Concerning the problem of how to solve the corresponding minimization problems, in general, the resulting variational models pose generally demanding non-convex optimization problems. Fortunately, our proposed iPiano algorithm is applicable for all of these problems. With the iPiano algorithm, the corresponding minimization problems can be efficiently solved. In additional, our models with the trained FoE image prior have the advantage of simplicity, as they only consist of the operation of image convolution with a set of linear kernels. Therefore, they are well-suited for GPU parallel computation, which can typically accelerate the inference procedure with a factor of $60\times \sim 80\times$ relative to a multi-threaded CPU implementation.

Concerning potential improvements of the FoE prior based variational methods, we believe that there are at least two possible directions. One is to learn specialized filters for specific task, i.e., with specific data term. The other one is to consider more flexible penalty function. In our current models, the penalty function is fixed the same for each filter. If we free the shape of the penalty function, our model will possess more freedom for optimization, which will probably boost the performance.

Chapter 4

Learning effective reaction diffusion processes

Contents

As stated in Section 1.6.3, we are motivated to train a nonlinear diffusion process like

$$u^{t+1} = u^t - \Delta t \cdot \left.\frac{\partial E}{\partial u}\right|_{u^t},$$

where E is defined by the FoE prior regularized energy functional such as

$$E(u) = \sum_{i=1}^{N_k} \alpha_i \phi(K_i u) + \frac{\lambda}{2}\|u - f\|_2^2 .$$

It turns out that the resulting gradient descent process leads to a nonlinear reaction diffusion process with higher-order filters. We find that the proposed nonlinear diffusion model is also motivated by the investigation of the conventional nonlinear reaction diffusion model.

4.1 Introduction

For several decades, image restoration remains an active research topic in low-level computer vision and hence new approaches are constantly emerging. However, many recently proposed algorithms achieve state-of-the-art performance only at the expense of very high computation time, which clearly limits their practical relevance. In this work, we propose an effective approach with both high computational efficiency and high restoration quality. We extend conventional nonlinear reaction diffusion models by several parametrized linear filters as well as several parametrized influence functions. We propose to train the parameters of the filters and the influence functions through a loss based approach. Experiments show that our trained nonlinear reaction diffusion models largely benefit from the training of the parameters and finally lead to the best reported performance on common test datasets for image restoration. Due to their structural simplicity, our trained models are highly efficient and are also well-suited for parallel computation on GPUs.

4.1.1 Background

Image restoration is the process of estimating uncorrupted images from noisy or blurred ones. It is one of the most fundamental operation in image processing, video processing, and low-level computer vision. There exists a huge amount of literature addressing the topic of image restoration problems, see for example [93] for a survey. Broadly speaking, most state-of-the-art techniques mainly concentrate on achieving utmost image restoration quality, with little consideration on the computational efficiency [57, 91, 143]. However, there are two notable exceptions, BM3D [36] and the recently proposed Cascade of Shrinkage Fields (CSF) [120] model, which simultaneously offer high efficiency and high image restoration quality.

It is well-known that BM3D is a highly engineered method, specialized for Gaussian noise. Moreover, it involves a block matching process, which is challenging for parallel computation on GPUs, alluding to the fact that it is not straightforward to accelerate BM3D algorithm on parallel architectures. In contrast, the recently proposed CSF model offers high levels of parallelism, making it well suited for GPU implementation, thus owning high computational efficiency.

Among the approaches to tackle the problem of image restoration, nonlinear anisotropic diffusion [104] defines a class of efficient approaches, as each diffusion step merely contains the convolution operation with a few linear filters. However, up to now, the image restoration quality of diffusion based approaches is still far away from the state-of-the-art,

although with many improvements [39, 58, 59, 106].

We give a brief review of nonlinear diffusion based approaches and then introduce our proposed diffusion model. In the seminal work [104], Perona and Malik (P-M) demonstrated that nonlinear diffusion models yield very impressive results for image processing. This has given rise to many revised models with various formulations. A notable variant is the so-called biased anisotropic diffusion (also known as reaction diffusion) proposed by Nordström [95], which introduces a bias term (forcing term) to free the user from the difficulty of specifying an appropriate stopping time for the P-M diffusion process. This additional term reacts against the strict smoothing effect of the pure P-M diffusion, therefore resulting in a nontrivial steady-state.

Tsiotsios et al. [128] discussed the choice of some crucial parameters in the P-M model, such as the diffusivity function, the gradient threshold parameter and the stopping time of the iterative process. Some works consider modification to the diffusion term or the reaction term for the reaction diffusion model [2, 33, 44, 95, 106], e.g., Acton et al. [2] and Plonkna et al. [106] exploited a more complicated reaction term to enhance oriented textures; [9, 124] proposed to replace the ordinary diffusion term with a flow equation based on mean curvature.

Gilboa et al. [54] proposed a forward and backward diffusion process, which incorporates explicit inverse diffusion with negative diffusivity coefficient by carefully choosing the diffusivity function. The resultant diffusion processes can adaptively switch between forward and backward diffusion process. In a latter work [134], the theoretical foundations for discrete forward-and-backward diffusion filtering were investigated. As demonstrated in [134], in spite of its negative diffusivity, forward and backward diffusion becomes well-posed if a nonstandard space discretization is used. It guarantees a positive diffusivity in discrete extrema.

Researchers also propose to exploit higher-order nonlinear diffusion filtering, which involves larger linear filters, e.g., fourth-order diffusion models [38, 39, 58, 59]. Meanwhile, theoretical properties about the stability and local feature enhancement of higher-order nonlinear diffusion filtering are established in [38].

In this chapter we focus on nonlinear diffusion process due to its high efficiency and propose a trainable nonlinear diffusion model, which is parameterized by the linear filters and the influence functions. The trained diffusion model contains many special influence functions (see Fig. 4.6 for an illustration), which greatly differ from usual influence functions employed in conventional diffusion models. It turns out that the trained diffusion

processes can lead to effective image restoration with state-of-the-art performance, while preserve the property of high efficiency of diffusion based approaches. At present, we are not aware of any previous works that simultaneously optimize the linear filters and influence functions of a nonlinear diffusion process.

Our proposed nonlinear diffusion process has several remarkable benefits as follows:

1) It is conceptually simple as it is merely a standard nonlinear diffusion model with trained filters and influence functions;

2) It has broad applicability to a variety of image restoration problems. In principle, all the diffusion based models can be revisited with appropriate training;

3) It achieves very high levels of recovery quality surpassing very recent state-of-the-arts;

4) It is highly computationally efficient, and well suited for parallel computation on GPUs.

4.1.2 Motivations of the proposed reaction diffusion process

In this section, we start with conventional nonlinear diffusion processes, then propose a training based reaction diffusion model for image restoration. Finally we show the relations between the proposed model and existing image restoration models.

4.1.2.1 Perona and Malik diffusion model

In the whole chapter, we stick to the fully discrete setting, where images are represented as column vectors, i.e., $u \in \mathbb{R}^N$. Therefore, the discrete version of the well-known Perona-Malik type nonlinear diffusion process [104] can be formulated as the following discrete partial differential equation (PDE) with an explicit finite difference scheme

$$\frac{u_{t+1} - u_t}{\Delta t} = - \sum_{i=\{x,y\}} \nabla_i^\top \Lambda(u_t) \nabla_i u_t \doteq - \sum_{i=\{x,y\}} \nabla_i^\top \phi(\nabla_i u_t)\,, \tag{4.1}$$

where matrices ∇_x and $\nabla_y \in \mathbb{R}^{N\times N}$ are finite difference approximation of the gradient operators in x-direction and y-direction, respectively and Δt denotes the time step. $\Lambda(u_t) \in \mathbb{R}^{N\times N}$ is defined as a diagonal matrix

$$\Lambda(u_t) = \operatorname{diag}\left(g\left(\sqrt{(\nabla_x u_t)_p^2 + (\nabla_y u_t)_p^2}\right)\right)_{p=1,\cdots,N}\,,$$

where function g is known as edge-stopping function [13] or diffusivity function [132], a typical g function given as $g(z) = 1/(1+z^2)$. If ignoring the coupled relation between $\nabla_x u$ and $\nabla_y u$, the P-M model can be also written as the second formula on the right side in (4.1), where $\phi(\nabla_i u) = (\phi(\nabla_i u)_1, \cdots, \phi(\nabla_i u)_N)^\top \in \mathbb{R}^N$ with function $\phi(z) = zg(z)$, known as influence function [13] or flux function [132]. In the upcoming subsection, we will stick to this decoupled formulation.

4.1.2.2 Proposed nonlinear diffusion model

Clearly, the matrix-vector product, $\nabla_x u$ can be interpreted as a 2D convolution of u with the linear filter $k_x = [-1, 1]$ (∇_y corresponds to the linear filter $k_y = [-1, 1]^\top$). Intuitively, in order to improve the capacity of the diffusion model, we can employ more filters of larger kernel size, in contrast to previous works that typically involve few filters with relatively small kernel size. We can additionally consider different influence functions for different filters, rather than an unique one. Moreover, the parameters of each iteration can vary across iterations. Taking the reaction term into account, our proposed nonlinear reaction diffusion model is formulated as

$$\frac{u_t - u_{t-1}}{\Delta t} = \underbrace{-\sum_{i=1}^{N_k} {K_i^t}^\top \phi_i^t(K_i^t u_{t-1})}_{\text{diffusion term}} \; - \; \underbrace{\psi(u_{t-1}, u_0)}_{\text{reaction term}}, \tag{4.2}$$

where $K_i \in \mathbb{R}^{N \times N}$ is a highly sparse matrix, implemented as 2D convolution of the image u with the filter kernel k_i, i.e., $K_i u \Leftrightarrow k_i * u$, K_i is a set of linear filters and N_k is the number of filters. In practice, we set $\Delta t = 1$, as we can freely scale the formula on the right side. Note that in our proposed diffusion model, the influence functions are adjustable and can be different from each other.

The specific formulation for the reaction term $\psi(u)$ depends on applications. For classical image restoration problems, such as Gaussian denoising, image deblurring, image super resolution and image inpainting, we can set the reaction term to be the gradient of a data term, i.e. $\psi(u) = \nabla_u \mathcal{D}(u)$. For example, if $\mathcal{D}(u, u_0) = \frac{\lambda}{2}\|Au - u_0\|_2^2$, $\psi(u) = \lambda A^\top(Au - u_0)$, where u_0 is the initial input degraded image, A is the associated linear operator, and λ is related to the strength of the reaction term. In the case of Gaussian denoising, A is the identity matrix.

In our work, instead of making use of the well-chosen filters and influence functions, we train the nonlinear diffusion process for specific image restoration problem, including

both the linear filters and the influence functions. As the diffusion process is an iterative approach, typically we run it for certain iterations. In order to make our proposed diffusion process more flexible, we train the parameters of the diffusion model for each single iteration. Finally, we arrive at a diffusion process which merely involves several iterations (referred to as stages).

4.2 Related works

Previous works [95, 118] show that in the nonlinear diffusion framework, there exist natural relations between reaction diffusion and regularization based energy functional. First of all, we can interpret (4.2) as one gradient descent step at u_{t-1} of a certain energy functional given by

$$E(u, u_0) = \sum_{i=1}^{N_k} \mathcal{R}_i(u) + \mathcal{D}(u, u_0)\,, \tag{4.3}$$

where $\mathcal{R}_i(u) = \sum_{p=1}^{N} \rho_i^t((K_i^t u)_p)$ are the regularizers and the functions ρ_i^t are the so-called penalty functions. Note that $\rho'(z) = \phi(z)$. Since the parameters $\{K_i^t, \rho_i^t\}$ vary across the stages, (4.3) is a dynamic energy functional, which changes at each iteration.

In the case of fixed $\{K_i^t, \rho_i^t\}$ across the stages t, it is obvious that functional (4.3) is exactly the fields of experts image prior regularized variational model for image restoration [27, 29, 111]. In our work, we do not exactly solve this minimization problem anymore, but in contrast, we run the gradient descent step for several stages, and each gradient descent step is optimized by training.

In a very recent work [120], Schmidt et al. exploited an additive form of half-quadratic optimization to solve the same problem (4.3), which finally leads to a fast and effective image restoration model called cascade of shrinkage fields (CSF). The CSF model makes an assumption that the data term in (4.3) is quadratic and the operator A can be interpreted as a convolution operation, such that the corresponding subproblem has fast closed-form solution based on discrete Fourier transform (DFT). This restrains its applicability to many other problems such as image super resolution. However, our proposed diffusion model does not have this restriction on the data term. In principle, any smooth data term is appropriate. Moreover, as shown in the following sections, we can even handle the case of non-smooth data term.

There exist some previous works [8, 40], also trying to train an optimized gradient descent algorithm for the energy functional similar to (4.3). In their works, the Gaussian

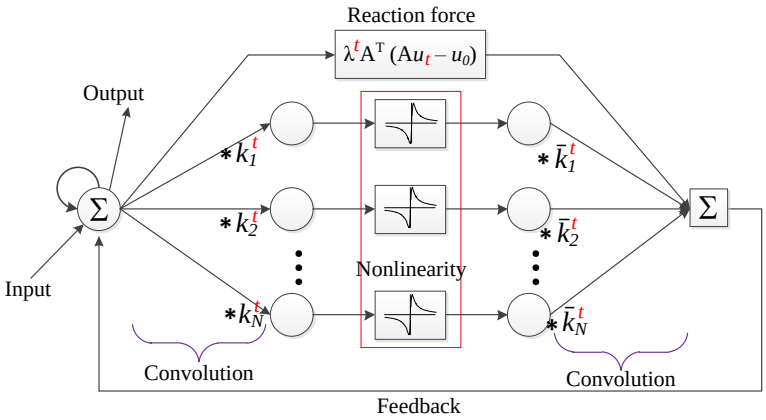


Figure 4.1: The architecture of our proposed diffusion model. Note that the additional convolution step with the rotated kernels $\bar{k}_i$ (*cf.* Equ. 4.7) does not appear in conventional feed-forward CNNs. Our model can be interpreted as a CNN with a feedback step, which makes it different from conventional feed-forward networks. Due to the feedback step, it can be categorized into recurrent neural networks [56].

denoising problem is considered, and the trained gradient descent algorithm typically involves less than 10 iterations. However, their model is much more constrained, in the sense that, they exploited the same filters for each gradient descent step. More importantly, the influence function in their model is fixed to be a unique one. This clearly restricts the model performance, as demonstrated in Sec. 3.1.

There are also few preliminary works, e.g., [87] to go beyond traditional PDEs of the form (4.1), and propose to learn optimal PDEs for image restoration via optimal control. However, the investigated PDE model in [87] is too simple to generate a promising performance, as they only optimize the linear combination coefficients of a few predefined terms, which depend on selected derivative filters.

The proposed diffusion model also bears an interesting link to the convolutional neural networks (CNNs) employed for image restoration problems [67]. One can see that each iteration (stage) of our proposed diffusion process involves the convolution operation with a set of linear filters, and thus it can be treated as a convolutional neural network. The architecture of our proposed network is shown in Figure 4.1, where one can see that it is not a pure feed-forward network any more, because it has a feedback step. Therefore, the structure of our CNN model is different from conventional feed-forward networks. Due to this feedback step, it can be categorized into recurrent neural networks [56]. Moreover, the nonlinearity (e.g., influence functions in the context of nonlinear diffusion) in our proposed network are trainable. However, conventional CNNs make use of fixed activation function,

e.g., ReLU functions or sigmoid functions.

4.3 Learning framework

In this paper, we train our models for two representative image restoration problems: (1) denoising of images corrupted by Gaussian noise and (2) JPEG blocking artifacts reduction, which is formulated as a non-smooth problem. We use a loss minimization scheme to learn the model parameters $\Theta_t = \{\lambda^t, \phi_i^t, k_i^t\}$ for each stage t of the diffusion process, given S training samples $\{u_0^{(s)}, u_{gt}^{(s)}\}_{s=1}^S$, where $u_0^{(s)}$ is a noisy input and $u_{gt}^{(s)}$ is the corresponding ground truth clean image.

We firstly consider a greedy training strategy to train the diffusion processes stage-by-stage, i.e., at stage t, we minimize the cost function

$$\mathcal{L}(\Theta_t) = \sum_{s=1}^{S} \ell(u_t^{(s)}, u_{gt}^{(s)}) \,, \tag{4.4}$$

where $u_t^{(s)}$ is the output of stage t of the diffusion process. We prefer the usual quadratic loss function to the negative PSNR used in [120], because the latter one imposes more weights on those samples with relatively smaller cost, and thus leads to slightly inferior results in practice. The loss function is given as

$$\ell(u_t^{(s)}, u_{gt}^{(s)}) = \frac{1}{2}\|u_t^{(s)} - u_{gt}^{(s)}\|_2^2 \,. \tag{4.5}$$

Parameterizing the influence functions ϕ_i^t: We parameterize the influence function via standard radial basis functions (RBFs), i.e., each function ϕ is represented as a weighted linear combination of a family of RBFs as follows

$$\phi_i^t(z) = \sum_{i=1}^{M} w_{ij}^t \varphi\left(\frac{|z - \mu_j|}{\gamma_j}\right) \,, \tag{4.6}$$

where φ represents different RBFs. In this paper, we exploit RBFs with equidistant centers μ_j and unified scaling γ_j. We investigate two typical RBFs [64]: (1) Gaussian radial basis and (2) triangular-shaped radial basis.

In general, the Gaussian RBF can provide better approximation for generally smooth function than the triangular-shaped RBF with the same number of basis functions. However, the triangular-shaped RBF based function parameterization has the advantage of

computational efficiency. More details can be found in the following section. In our work, we consider both function parameterization methods, but only present the results achieved based on the Gaussian RBF. A comprehensive comparison of both function approximation methods is subject to future work.

Training for denoising: According to the diffusion equation (4.2), for image denoising, the output of stage t is given as

$$u_t = u_{t-1} - \left(\sum_{i=1}^{N_k} \bar{k}_i^t * \phi_i^t(k_i^t * u_{t-1}) + \lambda^t (u_{t-1} - u_0) \right) , \tag{4.7}$$

where we explicitly use a convolution kernel $\bar{k}_i$ (obtained by rotating the kernel k_i 180 degrees) to replace the $K_i^\top$ for the sake of model simplicity *.

Training for deblocking: As described in the last chapter Section 3.2, we consider a new variational model for JPEG deblocking based on the FoE image prior model

$$\arg\min_u E(u) = \sum_{i=1}^{N_k} \rho_i(k_i * u) + \mathcal{I}_{QCS}(Du) , \tag{4.8}$$

where $\mathcal{I}_{QCS}$ is a indicator function over the set QCS (quantization constraint set), which is a box constraint determining all possible source image given the input JPEG compressed data. The sparse matrix $D \in \mathbb{R}^{N \times N}$ denotes the block DCT transform.

We derive the diffusion process with respect to the variational model (4.8) using the proximal gradient method [94], which reads as

$$u_t = D^\top \text{proj}_{QCS} \left(D \left(u_{t-1} - \sum\nolimits_{i=1}^{N_k} \bar{k}_i^t * \phi_i^t(k_i^t * u_{t-1}) \right) \right) , \tag{4.9}$$

where $\text{proj}_{QCS}(\cdot)$ denotes the projection operation onto QCS. More details can be found in the following section.

Gradients: We minimize (4.4) with commonly used gradient based L-BFGS algorithm [86]. The gradient of the loss function at stage t w.r.t the model parameters Θ_t is computed using standard chain rule, given as

$$\frac{\partial \ell(u_t, u_{gt})}{\partial \Theta_t} = \frac{\partial u_t}{\partial \Theta_t} \cdot \frac{\partial \ell(u_t, u_{gt})}{\partial u_t} , \tag{4.10}$$

*We use the symmetric boundary condition in our work. In this case, $K_i^\top$ can be interpreted as the convolution kernel $\bar{k}_i$ only in the central region. Therefore, we actually slightly modify the original model.

where $\frac{\partial \ell(u_t, u_{gt})}{\partial u_t} = u_t - u_{gt}$ is directly derived from (4.5), $\frac{\partial u_t}{\partial \Theta_t}$ is computed from (4.7) for the training of denoising task or (4.38) for the deblocking training, respectively. We do not present the derivatives for specific model parameters due to space limitation. All derivatives can be found in the following section.

Joint training: In (4.4), each stage is trained greedily such that the output of each stage is optimized according to the loss function, regardless of the total stages T used in the diffusion process. A better strategy would be to jointly train all the stages simultaneously. The joint training task is formulated as

$$\mathcal{L}(\Theta_{1,\cdots,T}) = \sum_{s=1}^{S} \ell(u_T^{(s)}, u_{gt}^{(s)}) \,, \tag{4.11}$$

where the loss function only depends on u_T (the output of the final stage T). The gradients of the loss function w.r.t Θ_t is given as

$$\frac{\partial \ell(u_T, u_{gt})}{\partial \Theta_t} = \frac{\partial u_t}{\partial \Theta_t} \cdot \frac{\partial u_{t+1}}{\partial u_t} \cdots \frac{\partial \ell(u_T, u_{gt})}{\partial u_T} \,,$$

which is the standard back-propagation technique widely used in the neural networks learning [75]. Compared with the greedy training, we additionally need to calculate $\frac{\partial u_{t+1}}{\partial u_t}$. See the following section for the derivations.

4.4 Computing gradients

As shown in the last section, the main issue for the training is to compute the gradient of the loss function with respect to the training parameters in each step. In this section, we provide the derivations to compute these gradients.

4.4.1 Preliminaries

When we modify the original diffusion equation

$$u_t = u_{t-1} - \left(\sum_{i=1}^{N_k} K_i^{t\top} \phi_i^t(K_i^t u_{t-1}) + \lambda^t (u_{t-1} - u_0) \right) , \tag{4.12}$$

to the following version

$$u_t = u_{t-1} - \left(\sum_{i=1}^{N_k} \bar{k}_i^t * \phi_i^t(k_i^t * u_{t-1}) + \lambda^t (u_{t-1} - u_0) \right) , \tag{4.13}$$

we find that it introduces some imperfections in the image boundary. The basic reason lies in the fact that , in the case of symmetric boundary condition used in our work, $K_i^\top$ can be interpreted as the convolution kernel $\bar{k}_i$ [†] only in the central region, while it can not in the boundary. However, in the diffusion equation (4.13), the convolution kernel $\bar{k}_i$ is applied to the whole image, thus bringing some artifacts in the boundary. The benefit to use the diffusion equation (4.13), other than (4.12) is that the revised model is more tractable in practice, especially for training, as everything can be done by the convolution operation efficiently.

In order to remove this artifacts, we pad the input image u_{t-1} of stage t, as well as the noisy image u_0, with mirror reflections of itself. This operation is formulated by the sparse "padding" matrix P. After a diffusion step, we only crop the central region of the output image u_t for usage. This operation is formulated by the sparse "cropping" matrix T. When we apply the matrix $P_T = P \times T$ to an image u, $P_T u$ corresponds to two operations: it first crops the central region of u, then pads it with mirror reflections.

After taking into account the operation of boundary handling, the exact diffusion process is illustrate in Figure 4.2. There we have $u_{tp} = P_T u_t$.

In our derivations, we use the symmetric boundary condition for the convolution operation $k * u$ (image $u \in \mathbb{R}^{m\times n}$, $k \in \mathbb{R}^{r\times r}$). As we know, it is equivalent to the matrix-vector product formulation Ku, where $K \in \mathbb{R}^{N\times N}$ is a highly sparse matrix and u is a column vector $u \in \mathbb{R}^N$ with $N = m \times n$. The result $k * u$ can also be interpreted with Uk, where matrix $U \in \mathbb{R}^{N\times R}$ is constructed from image u and k is a column vector $k \in \mathbb{R}^R$ with $R = r \times r$. This formulation is very helpful for the computation of the gradients of the loss function with respect to. the kernel k, as $U^\top v$ ($v \in \mathbb{R}^N$ is a column vector) can be explicitly interpreted as a convolution operation, which is widely used in classic convolutional neural networks [14]. In the following derivations, we will make use of this equivalence frequently, i.e.,

$$k * u \Longleftrightarrow Ku \Longleftrightarrow Uk\,.$$

4.4.2 Derivations of learning problem with respect to Gaussian denoising

Given S training samples $\{u_0^{(s)}, u_{gt}^{(s)}\}_{s=1}^S$, where $u_0^{(s)}$ is the noisy input and $u_{gt}^{(s)}$ is the corresponding ground truth clean image. Let assume the original image size is $m \times n$. We first pad the noisy image u_0 with ω pixels, then the resulting image has size $O =$

[†]Recall that kernel $\bar{k}_i$ is obtained by rotating k_i 180 degrees.

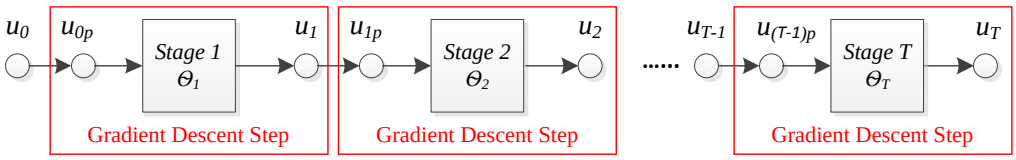


Figure 4.2: Proposed nonlinear diffusion process with careful boundary handling operation. Note that $u_{tp} = P_T u_t$.

$(m+2\omega)\times(n+2\omega)$. We have the corresponding matrix $T \in \mathbb{R}^{N\times O}$, $P \in \mathbb{R}^{O\times N}$, $P_T \in \mathbb{R}^{O\times O}$ and $u_0 \in \mathbb{R}^O, u_{gt} \in \mathbb{R}^N$.

4.4.2.1 Greedy training

In the greedy training for stage t, we are to minimize the following loss function with respect to the model parameters $\Theta_t = \{\lambda^t, \phi_i^t, k_i^t\}$ of stage t,

$$L(\Theta_t) = \sum_{s=1}^{S} \ell(u_t^{(s)}, u_{gt}^{(s)}) = \sum_{s=1}^{S} \frac{1}{2} \|T u_t^{(s)} - u_{gt}^{(s)}\|_2^2 , \tag{4.14}$$

where

$$u_t^s = u_{(t-1)p}^s - \left(\sum_{i=1}^{N_k} \bar{k}_i^t * \phi_i^t(k_i^t * u_{(t-1)p}^s) + \lambda^t (u_{(t-1)p}^s - u_{0p}^s) \right) . \tag{4.15}$$

Note that in the training for stage t, the images $u_{(t-1)p}$ are fixed, served as the input of this feed-forward step.

As the gradient of overall loss function on the whole training datasets can be decomposed to the sum over training samples, in the following derivation, we only consider the case of one training sample for the sake of brevity. The basic result of the gradient of the loss function with respect to the training parameters $\Theta_t = \{\lambda^t, \phi_i^t, k_i^t\}$ is given

$$\frac{\partial \ell(u_t, u_{gt})}{\partial \Theta_t} = \frac{\partial u_t}{\partial \Theta_t} \cdot \frac{\partial \ell(u_t, u_{gt})}{\partial u_t} , \tag{4.16}$$

where $\frac{\partial \ell(u_t, u_{gt})}{\partial u_t}$ is simply given as

$$\frac{\partial \ell(u_t, u_{gt})}{\partial u_t} = T^\top (T u_t - u_{gt}) .$$

Let us define a column vector $e \in \mathbb{R}^O$ as

$$e = T^\top (T u_t - u_{gt})\,.$$

Therefore, the main issue is to calculate $\frac{\partial u_t}{\partial \Theta_t}$ from (4.15).

Weight parameter λ^t: It is easy to see that

$$\frac{\partial u_t}{\partial \lambda^t} = \big(u_{(t-1)p} - u_{0p}\big)^\top . \tag{4.17}$$

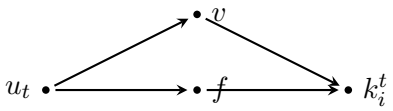
Thus $\frac{\partial \ell}{\partial \lambda^t}$ is given as

$$\boxed{\frac{\partial \ell}{\partial \lambda^t} = \big(u_{(t-1)p} - u_{0p}\big)^\top e\,.} \tag{4.18}$$

Filters k_i^t: Concerning the dependency of u_t on parameters k_i^t, it is easy to see the following relationship

$$u_t^s \to \underbrace{-\bar{k}_i^t}_{f} * \underbrace{\phi_i^t(k_i^t * u_{(t-1)p})}_{v}\,,$$

where f and v are two auxiliary variables defined as $f = -\bar{k}_i^t$ and $v = \phi_i^t(k_i^t * u_{(t-1)p})$. Therefore, we get the following dependency relationship,



According to the chain rule, we have

$$\frac{\partial u_t}{\partial k_i^t} = \frac{\partial f}{\partial k_i^t} \cdot \frac{\partial u_t}{\partial f} + \frac{\partial v}{\partial k_i^t} \cdot \frac{\partial u_t}{\partial v}\,. \tag{4.19}$$

Note that $f = -\bar{k}_i^t$, which can be formulated as $f = -P_{inv} k_i^t$ with matrix P_{inv} inverting the kernel vector k_i^t. Recall the equivalence

$$f * v \Longleftrightarrow Fv \Longleftrightarrow Vf\,.$$

Therefore, the first term of (4.19) is given as

$$\frac{\partial f}{\partial k_i^t} \cdot \frac{\partial u_t}{\partial f} = -P_{inv}^\top V^\top .$$

For the second term, we introduce an additional auxiliary variable z, defined as $z =$

$k_i^t * u_{(t-1)p}$. Then we have $v = \phi_i^t(z)$. Recall that

$$z = k_i^t * u_{(t-1)p} \Longleftrightarrow U_{(t-1)p} k_i^t \,.$$

Therefore, we obtain

$$\frac{\partial v}{\partial k_i^t} \cdot \frac{\partial u_t}{\partial v} = \frac{\partial z}{\partial k_i^t} \cdot \frac{\partial v}{\partial z} \cdot \frac{\partial u_t}{\partial v} = U_{(t-1)p}^\top \Lambda F^\top \,,$$

where Λ is a diagonal matrix $\Lambda = \mathrm{diag}(\phi_i^{t\prime}(z_1), \cdots, \phi_i^{t\prime}(z_p))$ ($\phi_i^{t\prime}$ is the first order derivative of function ϕ_i^t). Note that $F = -\bar{K}_i^t$. In summary, $\frac{\partial u_t}{\partial k_i^t}$ is given as

$$\underline{\frac{\partial u_t}{\partial k_i^t} = -\left(P_{inv}^\top V^\top + U_{(t-1)p}^\top \Lambda \bar{K}_i^{t\top}\right) \,.} \tag{4.20}$$

Finally, we arrive at the desired gradients

$$\boxed{\frac{\partial \ell}{\partial k_i^t} = -\left(P_{inv}^\top V^\top + U_{(t-1)p}^\top \Lambda \bar{K}_i^{t\top}\right) e \,.} \tag{4.21}$$

In practice, we do not need to explicitly construct the matrices $V, U, \bar{K}_i^t$. Recall that the product of matrices $V^\top, U_{(t-1)p}^\top$ and a vector can be accomplished by the convolution operator [14]. As shown in a previous work [29], $\bar{K}_i^{t\top}$ can also be accomplished by the convolution operation with the kernel k_i^t with careful boundary handling. Matrix $P_{inv}^\top$ is merely a linear operation which inverts the vectorized kernel k. In the case of a square kernel k, it is equivalent to the Matlab command

$$P_{inv}^\top k \Longleftrightarrow rot90(rot90(k)) \,.$$

If we have a closer look at the diffusion equation (4.15), we find that it has a scaling problem with respect to the filters k_i^t. First we know the function ϕ_i^t is free to tune in the training. In this case, if we scale the filter k_i^t by a factor of h to generate a new filter $\hat{k}_i^t = hk_i^t$, and the corresponding new function $\hat{\phi}_i^t$ is selected such that $\hat{\phi}_i^t(hz) = \frac{1}{h}\phi_i^t(z)$, then we will see that the term $\bar{k}_i^t * \phi_i^t(k_i^t * u_{(t-1)p})$ keep unchanged, i.e., two different set of parameters $\{k_i^t, \phi_i^t\}$ and $\{\hat{k}_i^t, \hat{\phi}_i^t\}$ own exactly the same loss function $\ell(u_t, u_{gt})$. In order to get rid of this ambiguity, it is necessary to fix the scale of the filters. In practice, we learn filters with fixed unit norm. Motivated by the statement in [29] that meaningful filters should be zero-mean, we also construct the training filter k from the DCT basis

$\mathcal{B} \in \mathbb{R}^{R\times(R-1)}$ (without the DC-component). Therefore, we define each filter $k \in \mathbb{R}^R$ with

$$k = \mathcal{B}\frac{c}{\|c\|_2}\,, \tag{4.22}$$

where $c \in \mathbb{R}^{R-1}$. Now the training parameters become c, and we need to calculate $\frac{\partial \ell}{\partial c}$. As shown in (4.21), we already have $\frac{\partial \ell}{\partial k}$, according to the chain rule, we have

$$\frac{\partial \ell}{\partial c} = \frac{\partial k}{\partial c} \cdot \frac{\partial \ell}{\partial k}\,,$$

where $\frac{\partial k}{\partial c}$ is computed from (4.22). Let us define an auxiliary variable $v = \frac{c}{\|c\|_2}$, we have

$$\begin{aligned}
\frac{\partial k}{\partial c} &= \frac{\partial v}{\partial c} \cdot \frac{\partial k}{\partial v} && \text{(4.23a)}\\
&= \frac{\partial v}{\partial c} \cdot \mathcal{B}^\top && \text{(4.23b)}\\
&= \left(\frac{\mathbf{I}}{\|c\|_2} + \frac{\partial[(c^\top c)^{-\frac{1}{2}}]}{\partial c} \cdot c^\top\right) \cdot \mathcal{B}^\top && \text{(4.23c)}\\
&= \left(\frac{\mathbf{I}}{\|c\|_2} + \frac{\partial[(c^\top c)]}{\partial c} \cdot (-\frac{1}{2}\frac{1}{\|c\|_2^3}) \cdot c^\top\right) \cdot \mathcal{B}^\top && \text{(4.23d)}\\
&= \left(\frac{\mathbf{I}}{\|c\|_2} + 2c \cdot (-\frac{1}{2}\frac{1}{\|c\|_2^3}) \cdot c^\top\right) \cdot \mathcal{B}^\top && \text{(4.23e)}\\
&= \frac{1}{\|c\|_2}\left(\mathbf{I} - \frac{c}{\|c\|_2} \cdot \frac{c^\top}{\|c\|_2}\right) \cdot \mathcal{B}^\top\,, && \text{(4.23f)}
\end{aligned}$$

where $\mathbf{I} \in \mathbb{R}^{(R-1)\times(R-1)}$ is the identity matrix. Combining the Equation (4.21) and (4.31), we can obtain the desired gradients of the loss function with respect to the training parameter c_i^t, given as

$$\boxed{\frac{\partial \ell}{\partial c_i^t} = -\frac{1}{\|c_i^t\|_2}\left(\mathbf{I} - \frac{c_i^t}{\|c_i^t\|_2} \cdot \frac{c_i^{t\top}}{\|c_i^t\|_2}\right) \cdot \mathcal{B}^\top \cdot \left(P_{inv}^\top V^\top + U_{(t-1)p}^\top \Lambda \bar{K}_i^{t\top}\right) e} \tag{4.24}$$

Influence functions ϕ: According to diffusion equation (4.15), the dependency of u_t on the influence function ϕ_i^t is given as

$$u_t \to -\bar{K}_i^t \cdot \phi_i^t(K_i^t \cdot u_{(t-1)p})\,. \tag{4.25}$$

Let us define an auxiliary variable $y \in \mathbb{R}^O$ as

$$y = K_i^t \cdot u_{(t-1)p} \,. \tag{4.26}$$

In our work, function ϕ_i^t is represented as

$$\phi_i^t(z) = \sum_{j=1}^{M} w_{ij}^t \varphi\left(\frac{|z-\mu_j|}{\gamma}\right), \tag{4.27}$$

Therefore, the column vector $\phi_i^t(y) \in \mathbb{R}^O$ can be reformulated via a matrix equation

$$\phi_i^t(y) = G(y) \cdot w_i^t \,,$$

where $w_i^t \in \mathbb{R}^M$ is the vectorized version of parameters w_{ij}^t, matrix $G(y) \in \mathbb{R}^{O \times M}$ is given as

$$\underbrace{\begin{bmatrix} \varphi(\frac{|y_1-\mu_1|}{\gamma}) & \varphi(\frac{|y_1-\mu_2|}{\gamma}) & \cdots & \varphi(\frac{|y_1-\mu_M|}{\gamma}) \\ \varphi(\frac{|y_2-\mu_1|}{\gamma}) & \varphi(\frac{|y_2-\mu_2|}{\gamma}) & \cdots & \varphi(\frac{|y_2-\mu_M|}{\gamma}) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi(\frac{|y_O-\mu_1|}{\gamma}) & \varphi(\frac{|y_O-\mu_2|}{\gamma}) & \cdots & \varphi(\frac{|y_O-\mu_M|}{\gamma}) \end{bmatrix}}_{G(y)} \underbrace{\begin{bmatrix} w_{i1} \\ w_{i2} \\ \vdots \\ w_{iM} \end{bmatrix}}_{w_i} = \underbrace{\begin{bmatrix} \phi_i(y_1) \\ \phi_i(y_2) \\ \vdots \\ \phi_i(y_Q) \end{bmatrix}}_{\phi_i(y)}$$

Figure 4.3: G matrix

Now, it is straightforward to obtain $\frac{\partial u_t}{\partial w_i^t}$, given as

$$\frac{\partial u_t}{\partial w_i^t} = -G^\top \bar{K_i^t}^\top . \tag{4.28}$$

Then we can obtain the desired gradients of the loss function with respect to the parameters of the influence function, written as

$$\boxed{\frac{\partial \ell}{\partial w_i^t} = -G^\top \bar{K_i^t}^\top e \,.} \tag{4.29}$$

In this chapter, we investigate two typical RBFs [64]: (1) Gaussian radial basis φ_g and (2) triangular-shaped radial basis φ_t, which are given as

$$\varphi_g(z) = \varphi\left(\frac{|z-\mu|}{\gamma}\right) = \exp\left(-\frac{(z-\mu)^2}{2\gamma^2}\right)$$

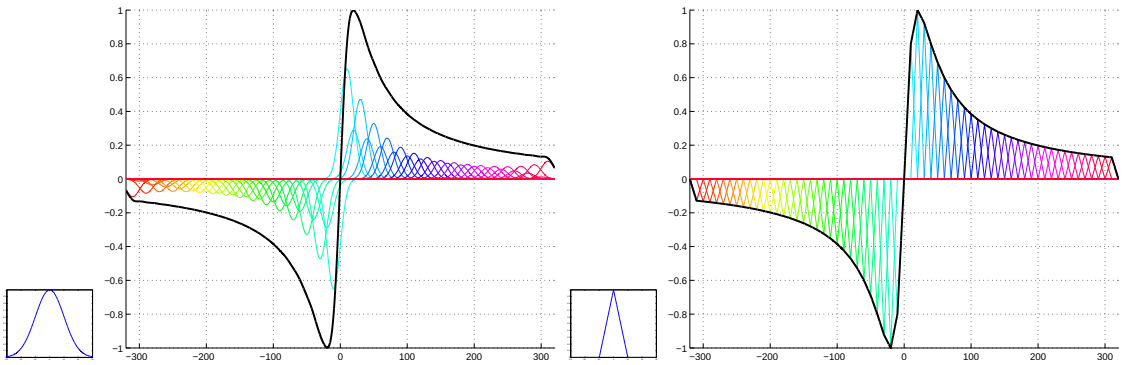


Figure 4.4: Function approximation via Gaussian $\varphi_g(z)$ or triangular-shaped $\varphi_t(z)$ radial basis function, respectively.

and

$$\varphi_t(z) = \varphi\left(\frac{|z-\mu|}{\gamma}\right) = \begin{cases} 1 - \frac{|z-\mu|}{\gamma} & |z-\mu| \leq \gamma \\ 0 & |z-\mu| > \gamma \end{cases}$$

respectively. The basis functions are shown in Figure 4.4, together with an example of the function approximation by using two different RBF methods.

In Figure 4.4, we can see that in the case of triangular-shaped RBF based function approximation any input variable z only involves two basis functions, i.e., each row of the G matrix (4.3) only has two non-zero numbers. Therefore, we can explicitly make use of this property in the implementation to speed up the computation of Equation (4.29), i.e., the triangular-shaped RBF based training process is generally faster than the Gaussian RBF based one.

In the training, the first order derivative of the influence function ϕ is also required as in Equation (4.24). In the case of Gaussian RBF, the first order derivative is given as

$$\phi_i'(z) = -\gamma \sum_{i=1}^{M} w_{ij} \exp\left(-\frac{\gamma}{2}(z-\mu_j)^2\right) \cdot (z-\mu_j)\,.$$

In the case of triangular-shaped RBF, ϕ' is defined by a piece-constant function as ϕ is a piece-wise linear function. Although the influence function ϕ and its derivative ϕ' is not smooth, the training still works quite well.

In practice, in order to speed up the computation of the function value $\phi(z)$ and its derivative $\phi'(z)$ for the case Gaussian RBF, we approximate these functions with piece-wise linear functions (the function values at discrete points are precomputed and stored in a lookup-table), then the function values at point z can be retrieved efficiently using

linear interpolation.

Concerning the function approximation accuracy, in general, the Gaussian RBF can provide a better approximation for generally smooth function than the triangular-shaped RBF with the same number of basis functions, as the latter generates a piece-wise linear function for approximation. In order to improve the approximation accuracy of the triangular-shaped RBF based method, usually we need to exploit more basis functions relative to the Gaussian RBF based method. However, using more basis functions will bring an unwanted problem of over-fitting. Therefore, certain regularization technique is required. Unfortunately, up to now we have not figured out the best choice for the regularization term.

In our work, we have investigated both function approximation methods, and we find that they generate similar results. We only present the results obtained by the Gaussian RBF due to space limitation. We do not provide a comprehensive comparison of two methods, as it is out of the scope of this chapter.

4.4.2.2 Joint training

In the joint training, the parameters of all stages T are optimized simultaneously. The joint training task is formulated as

$$L(\Theta_{1,\cdots,T}) = \sum_{s=1}^{S} \ell(u_T^{(s)}, u_{gt}^{(s)}) \,, \tag{4.30}$$

where the loss function only depends on u_T, the output of the final stage T. The gradients of the loss function with respect to Θ_t is given as

$$\frac{\partial \ell(u_T, u_{gt})}{\partial \Theta_t} = \frac{\partial u_t}{\partial \Theta_t} \cdot \frac{\partial \ell(u_T, u_{gt})}{\partial u_t} \,,$$

where $\frac{\partial u_t}{\partial \Theta_t}$ has been already done in the preceding subsection. Now the main issue is to calculate $\frac{\partial \ell(u_T, u_{gt})}{\partial u_t}$. As we only know

$$e = \frac{\partial \ell(u_T, u_{gt})}{\partial u_T} = T^\top (T u_T - u_{gt}) \,,$$

the standard back-propagation technique widely used in neural networks learning [75] can be used to calculate the desired gradients, which is written as

$$\frac{\partial \ell(u_T, u_{gt})}{\partial u_t} = \frac{\partial u_{t+1}}{\partial u_t} \cdot \frac{\ell(u_T, u_{gt})}{\partial u_{t+1}} \tag{4.31a}$$

$$= \frac{\partial u_{t+1}}{\partial u_t} \cdot \frac{\partial u_{t+2}}{\partial u_{t+1}} \cdot \frac{\ell(u_T, u_{gt})}{\partial u_{t+2}} \tag{4.31b}$$

$$= \frac{\partial u_{t+1}}{\partial u_t} \cdot \frac{\partial u_{t+2}}{\partial u_{t+1}} \cdots \frac{\partial u_T}{\partial u_{T-1}} \cdot e\,. \tag{4.31c}$$

In practice, (4.31) is accomplished with a backward manner starting from the last stage. Now the only thing we need to calculate is $\frac{\partial u_{t+1}}{\partial u_t}$. Recall the diffusion process shown in Figure 4.2, it is straightforward to see that

$$\frac{\partial u_{t+1}}{\partial u_t} = \frac{\partial u_{tp}}{\partial u_t} \cdot \frac{\partial u_{t+1}}{\partial u_{tp}}\,,$$

where $\frac{\partial u_{tp}}{\partial u_t} = P_T^\top$ according to the equation $u_{tp} = P_T u_t$, and $\frac{\partial u_{t+1}}{\partial u_{tp}}$ can be obtained from the diffusion equation (4.15).

$$\frac{\partial u_{t+1}}{\partial u_{tp}} = (1 - \lambda^{t+1})\mathbf{I} - \sum_{i=1}^{N_k} {K_i^{t+1}}^\top \cdot \Lambda_i \cdot \left(\bar{K}_i^{t+1}\right)^\top\,,$$

where Λ_i is a diagonal matrix $\Lambda_i = \operatorname{diag}(\phi_i^{t+1\prime}(z_1), \cdots, \phi_i^{t+1\prime}(z_p))$ with $z = k_i^{t+1} * u_{tp}$. Therefore, the overall $\frac{\partial u_{t+1}}{\partial u_t}$ is given as

$$\frac{\partial u_{t+1}}{\partial u_t} = P_T^\top \cdot \left((1 - \lambda^{t+1})\mathbf{I} - \sum_{i=1}^{N_k} {K_i^{t+1}}^\top \cdot \Lambda_i \cdot \left(\bar{K}_i^{t+1}\right)^\top \right).$$

Then the gradients of $\frac{\partial \ell(u_T, u_{gt})}{\partial u_t}$ can be computed using the backward recurrence described above. Once we have obtained the results of $\frac{\partial \ell(u_T, u_{gt})}{\partial u_t}$, it is straightforward to calculate $\frac{\partial \ell(u_T, u_{gt})}{\partial \Theta_t}$ using the derivations in previous subsection.

4.4.3 Training for JPEG deblocking

Recall that we consider the JPEG deblocking problem by defining a new variational model, which incorporates the FoE image prior model and the quantization constraint set (QCS). More details can be found in the last chapter Section 3.2.

4.4.3.1 Variational model for image deblocking

In our training, an image u of size $m \times n$ is padded with ω pixels (we set $\omega = 8$ for this problem). In order to simplify the notation, the interval S is represented by two column vectors $a \in \mathbb{R}^O$ and $b \in \mathbb{R}^O$ ($O = (m+2\omega) \times (n+2\omega)$), which correspond to the lower and upper bounds of the intervals $S^I_{i,j}$, respectively. We further define a highly sparse matrix $D \in \mathbb{R}^{O \times O}$, which makes Du equivalent to the block-wise DCT transform applied to the two-dimensional image u.

Given the compressed image data, the QCS is given as the box constraint $S = [a, b]$, and the set of possible source image of the compression process is defined as

$$U = \{u \in \mathbb{R}^O \mid (Du)_p \in [a_p, b_p]\}\,.$$

Then we can define our variational model based on the FoE image prior model and QCS, which reads as

$$\arg\min_{u \in U} E(u) = \sum_{i=1}^{N_k} \rho_i(k_i * u)\,. \tag{4.32}$$

This is a constrained optimization problem, and it can be rewritten as

$$\arg\min_{u} E(u) = \sum_{i=1}^{N_k} \rho_i(k_i * u) + \mathcal{I}_S(Du)\,, \tag{4.33}$$

where

$$\mathcal{I}_S(Du) = \begin{cases} 0 & \text{if } Du \in S, \\ \infty & \text{else}\,. \end{cases}$$

In this formulation, we exploit the convex set S instead of set U, as S is a box constraint, which is simpler than U.

As the minimization problem (4.33) contains the non-smooth indicator function, the standard gradient descent algorithm is not applicable. Therefore, we resort to the more general proximal gradient method [94], which is applicable to solve a class of the following minimization problems

$$\min_{u} h(u) = f(u) + g(u)\,, \tag{4.34}$$

where f is a smooth function and g is convex (possibly non-smooth) function. The proximal gradient method is defined as

$$u_t = (\mathbf{I} + \tau \partial g)^{-1}(u_{t-1} - \tau \nabla f(u_{t-1}))\,,$$

where τ is the step size parameter, $(\mathbf{I} + \tau\partial g)^{-1}$ denotes the proximal mapping operator. Casting the problem (4.33) in the form of (4.34), we have $f(u) = \sum_{i=1}^{N_k} \rho_i(k_i * u)$ and $g(u) = \mathcal{I}_S(Du)$. It is easy to check that

$$\nabla f(u) = \sum_{i=1}^{N_k} \bar{k}_i * \phi_i(k_i * u)\,,$$

with $\phi_i = \rho_i'$. We again make the modification $\bar{k}_i$ to the rigorous formulation $K_i^\top$. The proximal mapping with respect to g is given as the following minimization problem

$$(I + \tau\partial g)^{-1}(\hat{u}) = \arg\min_u \frac{\|u - \hat{u}\|_2^2}{2} + \tau\mathcal{I}_S(Du)\,. \tag{4.35}$$

As shown in the last chapter Section 3.2, the solution of this problem is given as

$$\tilde{u} = D^\top \tilde{c}\,,$$

with

$$\tilde{c}_p = \begin{cases} \hat{c}_p & \text{if } \hat{c}_p \in S_p = [a_p, b_p] \\ b_p & \text{if } \hat{c}_p > b_p \\ a_p & \text{if } \hat{c}_p < a_p\,. \end{cases} \tag{4.36}$$

Finally, the the solution of u is given as $\tilde{u} = D^\top \tilde{c}$. Therefore, the overall gradient descent step is given as

$$u_t = D^\top \mathrm{proj}_{QCS}\left(D\left(u_{t-1} - \sum_{i=1}^{N_k} \bar{k}_i * \phi_i(k_i * u_{t-1})\right)\right)\,, \tag{4.37}$$

where $\mathrm{proj}_{QCS}(\cdot)$ denotes the orthogonal projection onto QCS (4.36). In our training model, as we consider different filters and influence functions for each stage t, the accurate diffusion process is given as

$$u_t = D^\top \mathrm{proj}_{QCS}\left(D\left(u_{t-1} - \sum_{i=1}^{N_k} \bar{k}_i^t * \phi_i^t(k_i^t * u_{t-1})\right)\right)\,. \tag{4.38}$$

The projection operator can be represented by the function $\eta(z)$ show in Figure 4.5. Therefore, the corresponding gradient descent step (4.38) can be rewritten as the following

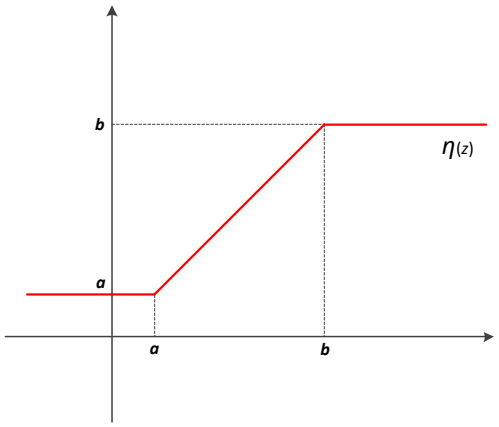


Figure 4.5: The projection function $\eta(z)$.

formulas by introducing the auxiliary variables $v, z,$ and y.

$$\begin{cases} u_t = D^\top v \\ v = \eta(z) \\ z = Dy \\ y = u_{t-1} - \sum_{i=1}^{N_k} \bar{k}_i^t * \phi_i^t(k_i^t * u_{t-1})\,. \end{cases} \tag{4.39}$$

4.4.3.2 Gradients with respect to the deblocking training

For the training of this new problem, the training parameters Θ_t is given by $\Theta_t = \{\phi_i^t, k_i^t\}$. According to the result of (4.16), for this new training problem, we can make use of the framework presented in the last section and we just need to recalculate $\frac{\partial u_t}{\partial \Theta_t}$ from (4.39), which is given as

$$\begin{aligned} \frac{\partial u_t}{\partial \Theta_t} =& \frac{\partial y}{\partial \Theta_t} \cdot \frac{\partial z}{\partial y} \cdot \frac{\partial v}{\partial z} \cdot \frac{\partial u_t}{\partial v} \\ =& \frac{\partial y}{\partial \Theta_t} \cdot D^\top \cdot \operatorname{diag}(\eta'(z)) \cdot D\,, \end{aligned} \tag{4.40}$$

where $\eta'(z) = (\eta'(z_1), \eta'(z_2), \cdots, \eta'(z_O))^\top \in \mathbb{R}^O$ with $\eta'(z_p)$ defined as

$$\eta'(z_p) = \begin{cases} 1 & \text{if} \quad z_p \in [a_p, b_p]\,, \\ 0 & \text{else} \end{cases} \tag{4.41}$$

Even though we do not consider any smoothing technique for the non-smooth function η, in practice we find that it is not a problem for the training procedure by using the above discontinuous derivative. According to the derivations (4.20) and (4.28) in the last section, it is straightforward to calculate $\frac{\partial y}{\partial \Theta_t}$, which leads to exactly the same results.

Concerning the joint training model, we can still make use of the same framework presented in the last section and we just need to additionally compute $\frac{\partial u_t}{\partial u_{t-1}}$. Taking into account the operation of boundary handling, $\frac{\partial u_t}{\partial u_{t-1}}$ is given as

$$\begin{aligned}\frac{\partial u_t}{\partial u_{t-1}} =& P_T^\top \cdot \frac{\partial y}{\partial u_{t-1}} \cdot \frac{\partial z}{\partial y} \cdot \frac{\partial v}{\partial z} \cdot \frac{\partial u_t}{\partial v} \\ =& P_T^\top \cdot \left(\mathbf{I} - \sum_{i=1}^{N_k} {K_i^t}^\top \cdot \Lambda_i \cdot \bar{K_i^t}^\top \right) \cdot D^\top \cdot \operatorname{diag}(\eta'(z)) \cdot D \end{aligned} \tag{4.42}$$

Combining the derivation results of (4.40), (4.42) and the framework presented in the last section, we can reach the formulas required for the training of the JPEG deblocking problem.

4.5 Training experiments for image denoising and deblocking

As the calculation of the gradients of the loss function in (4.10) can be accomplished with convolution technique efficiently, we can train our models on a relatively large dataset, even with a simple Matlab implementation. We used the same 400 training images as [120], and cropped a 180×180 region from each image, resulting in a total of 400 training samples of size 180×180, i.e., roughly 13 million pixels.

We trained the proposed diffusion process with at most 8 stages to observe its saturation behavior after some stages. We first greedily trained T stages of our model with specific model capacity, then conducted a joint training for the parameters of the whole T stages.

In our work, we mainly considered two trained reaction diffusion (TRD) models.

$$\begin{aligned}&\mathrm{TRD}_{5\times5}^{T}\text{,Fully trained model with 24 filters of size } 5 \times 5\,,\\ &\mathrm{TRD}_{7\times7}^{T}\text{,Fully trained model with 48 filters of size } 7 \times 7\,,\end{aligned}$$

where $\mathrm{TRD}_{m\times m}^{T}$ denotes a nonlinear diffusion process of stage T with filters of size $m \times m$.

The filters number is $m^2 - 1$, if not specified.

In training, computing the gradients $\frac{\partial \mathcal{L}}{\partial \Theta}$ with respect to the parameters of one stage for 400 images of size 180×180 takes about 35s ($\text{TRD}_{5\times5}$), 75s ($\text{TRD}_{7\times7}$) or 165s ($\text{TRD}_{9\times9}$) with Matlab implementation on a server with CPUs: Intel(R) Xeon E5-2680 @ 2.80GHz (eight parallel threads, 63 Gaussian RBF kernels for the influence function parameterization). We typically run 200 L-BFGS iterations for optimization. Therefore, the total training time, e.g., for the $\text{TRD}^5_{7\times7}$ model is about $5 \times (200 \times 75)/3600 = 20.8h$.

4.5.1 Image denoising experiments

We started with the training model of $\text{TRD}^T_{5\times5}$. We first considered the greedy scheme to train a diffusion process up to 8 stages (i.e., $T \leq 8$), in order to observe the asymptotic behavior of the diffusion process. After the greedy training was completed, we conducted joint training for a diffusion model of certain stages (e.g., $T = 5$), by simultaneously tuning the parameters in all stages.

We initialized the joint training with the parameters obtained from greedy training, as this is guaranteed not to decrease the training performance.[‡] In previous work [120], it is shown that joint training a model with filters of size 5×5 or larger hardly makes a difference relative to the result obtained by the greedy training. However, in our work we observed that joint training always improves the result of greedy training.

Note that for the models trained in the greedy manner, we can stop the inference at any stage, as its output of each stage is optimized. However, for the jointly trained models, we have to run T stages, as in this case only the output of the T^{th} stage is optimized.

We first trained our diffusion models for the Gaussian denoising problem with standard deviation $\sigma = 25$. The noisy training images were generated by adding synthetic Gaussian noise with $\sigma = 25$ to the clean images. Once we have trained a diffusion model, we evaluated its performance on a standard test dataset of 68 test images.[§]

We present the final results of the joint training in Table 4.1, together with a selection of recent state-of-the-art denoising algorithms, namely BM3D [36], LSSC [91], EPLL [143], opt-MRF [27], RTF_5 model [68] and two very recent methods: the CSF model [120] and WNNM [57]. We downloaded these algorithms from the corresponding author's homepage, and used them as is.[¶]

Concerning the performance of our $\text{TRD}^T_{5\times5}$ models, we find that joint training usually

[‡]In practice, we also find that a plain initialization can actually lead to closely similar results.
[§]The test images are strictly separate from the training datasets.
[¶]The implementation of the RTF_5 model is not available, and we quoted its result from [120].

Method	σ		St.	$\sigma = 15$	
	15	25		$\text{TRD}_{5\times5}$	$\text{TRD}_{7\times7}$
BM3D	31.08	28.56	2	31.14	31.30
LSSC	31.27	28.70	5	31.30	**31.42**
EPLL	31.19	28.68	8	31.34	**31.43**
opt-MRF	31.18	28.66		$\sigma = 25$	
RTF_5	–	28.75		$\text{TRD}_{5\times5}$	$\text{TRD}_{7\times7}$
WNNM	31.37	28.83	2	28.58	28.77
$\text{CSF}^5_{5\times5}$	31.14	28.60	5	28.78	**28.91**
$\text{CSF}^5_{7\times7}$	31.24	28.72	8	28.83	**28.95**

Table 4.1: Average PSNR (dB) on 68 images from [111] for image denoising with $\sigma = 15, 25$.

leads to an improvement of about 0.1dB in the cases of $T \geq 5$. From Table 4.1, one can see that (1) the performance of the $\text{TRD}^T_{5\times5}$ model saturates after stage 5, i.e., in practice, 5 stages are typically enough; (2) our $\text{TRD}^5_{5\times5}$ model has achieved significant improvement (28.78 *vs.*28.60), compared to a similar model $\text{CSF}^5_{5\times5}$, which has the same model capacity and (3) moreover, our $\text{TRD}^8_{5\times5}$ model is on par with so far the best-reported algorithm - WNNM.

When comparing with some closely related models such as the FoE prior based variational model [27], the FoE derived CSF model [120] and convolutional neural networks (CNNs) [67], our trained models can provide significantly superior performance. Therefore, a natural question arises: what is the critical factor in the effectiveness of the trained diffusion models? There are actually two main aspects in our training model: (1) the linear filters and (2) the influence functions. In order to have a better understanding of the trained models, we went through a series of experiments to investigate the impact of these two aspects.

4.5.1.1 Analysis of the proposed diffusion process

Concentrating on the model capacity of 24 filters of size 5×5, we considered the training of a diffusion process with 10 steps, i.e., $T = 10$ for the Gaussian denoising of noise level $\sigma = 25$. We exploited two main classes of configurations: (A) the parameters of every stage are the same and (B) every diffusion stage is different from each other. In both configurations, we consider two cases: (I) only train the linear filters with fixed influence function $\phi(z) = 2z/(1+z^2)$ and (II) simultaneously train the filters and influence functions.

Based on the same training dataset and test dataset, we obtained the following results: (A.I) every diffusion step is the same, and only the filters are optimized with fixed influence

function. This is a similar configuration to previous works [8, 40]. The trained model achieves a test performance of 28.47dB. (A.II) with additional tuning of the influence functions, the resulting performance is boosted to 28.60dB. (B.I) every diffusion step can be different, but only train the linear filters with fixed influence function. The corresponding model obtains the result of 28.56dB, which is equivalent to the variational model [27] with the same model capacity. Finally (B.II) with additional optimization of the influence functions, the trained model leads to a significant improvement with the result of 28.86dB.

The analysis experiments demonstrate that without the training of the influence functions, there is no chance to achieve significant improvements over previous works, no matter how hard we tune the linear filters. Therefore, we believe that the additional freedom to tune the influence functions is the critical factor of our proposed training model. After having a closer look at the learned influence functions of the $\text{TRD}^5_{5\times5}$ model, these functions reinforce our argument.

4.5.1.2 Learned influence functions

After we go through all 120 learned influence functions in the $\text{TRD}^5_{5\times5}$ model, we find that the form of the learned penalty functions $\rho^{\|}$ in the $\text{TRD}^5_{5\times5}$ model can be divided into four classes (see the corresponding subfigures in Figure 4.6):

(a) Truncated convex penalty functions with low values around zero to encourage smoothness.

(b) Negative Mexican hat functions, which have a local minimum at zero and two symmetric local maxima.

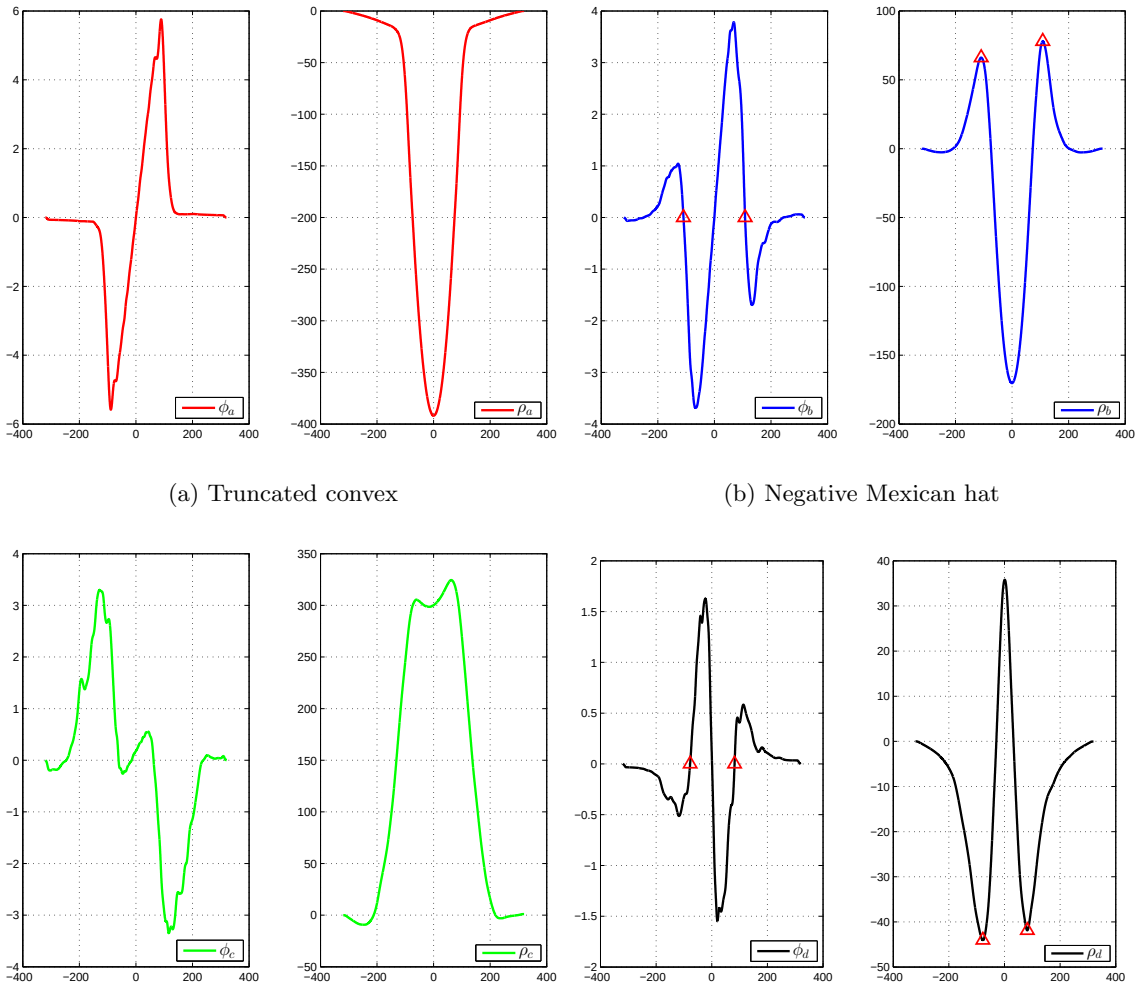
(c) Truncated concave functions with smaller values at the two tails.

(d) Double-well functions, which have a local maximum (not a minimum any more) at zero and two symmetric local minima.

At first glance, the learned penalty functions (except (a)) differ significantly from the usually adopted penalty functions used in PDE and energy minimization methods. However, it turns out that they have a clear meaning for image regularization.

Regarding the penalty function (b), there are two critical points (indicated by red triangles). When the magnitude of the filter response is relatively small (i.e., less than

$^{\|}$The penalty function $\rho(z)$ is integrated from the influence function $\phi(z)$ according to the relation $\phi(z) = \rho'(z)$



(a) Truncated convex

(b) Negative Mexican hat

(c) Truncated concave

(d) Double-well penalty

Figure 4.6: The figure shows four characteristic influence functions (left plot in each subfigure) together with their corresponding penalty functions (right plot in each subfigure), learned by our proposed method. A major finding in this paper is that our learned penalty functions significantly differ from the usual penalty functions adopted in partial differential equations and energy minimization methods. In contrast to their usual robust smoothing properties which is caused by a single minimum around zero, most of our learned functions have multiple minima different from zero and hence are able to enhance certain image structures.

the critical points), probably it is stimulated by the noise and therefore the penalty function encourages smoothing operation as it has a local minimum at zero. However, once the magnitude of the filter response is large enough (i.e., across the critical points), the corresponding local patch probably contains a real image edge or certain structure. In

this case, the penalty function encourages to increase the magnitude of the filter response, alluding to an image sharpening operation. Therefore, the diffusion process controlled by the influence function (b), can adaptively switch between image smoothing (forward diffusion) and sharpening (backward diffusion). We find that the learned influence function (b) is closely similar to an elaborately designed function in a previous work [54], which leads to an adaptive forward-and-backward diffusion process.

A similar penalty function to the learned function (c) with a concave shape is also observed in previous work on image prior learning [142]. This penalty function also encourages to sharpen the image edges. Concerning the learned penalty function (d), as it has local minima at two specific points, it prefers specific image structure, implying that it helps to form certain image structure. We also find that this penalty function is exactly the type of bimodal expert functions for texture synthesis employed in [62].

Now it is clear that the diffusion process involving the learned influence functions does not perform pure image smoothing any more for image processing. In contrast, it leads to a diffusion process for adaptive image smoothing and sharpening, distinguishing itself from previous commonly used image regularization techniques.

4.5.1.3 Training for other configurations

In order to investigate the influence of the model capacity, we increase the filter size to 7×7 and 9×9. We find that increasing the filter size from 5×5 to 7×7 brings a significant improvement of 0.14dB ($\text{TRD}^5_{7\times7}$ *vs.*$\text{TRD}^5_{5\times5}$) as show in Table 4.1. However, if we further increase the filter size to 9×9, the resulting $\text{TRD}^5_{9\times9}$ leads to a performance of 28.96dB (a slight improvement of 0.05dB relative to the $\text{TRD}^5_{7\times7}$ model). In practice, we prefer the $\text{TRD}^5_{7\times7}$ model as it provides the best trade-off between performance and computation time.

We also trained diffusion models for the noise level of $\sigma = 15$, and the test performance is shown in Table 4.1. In experiments, we observed that joint training can always gain an improvement of about 0.1dB over the greedy training for the cases of $T \geq 5$.

From Table 4.1, one can see that for both noise levels, the resulting $\text{TRD}_{7\times7}$ model achieves the highest average PSNR. The $\text{TRD}^5_{7\times7}$ model outperforms the benchmark BM3D method by 0.35dB in average. This is a notable improvement as few methods can surpass BM3D more than 0.3dB in average [79]. Moreover, the $\text{TRD}^5_{7\times7}$ model also surpasses the best-reported algorithm - WNNM method, which is very slow as shown in Table 4.2. In summary, our $\text{TRD}^5_{7\times7}$ model outperforms all the recent state-of-the-arts

Method	256^2	512^2	1024^2	2048^2	3072^2
BM3D [36]	1.1	4.0	17	76.4	176.0
$\text{CSF}^5_{7\times7}$ [120]	3.27	11.6	40.82	151.2	494.8
WNNM [57]	122.9	532.9	2094.6	–	–
$\text{TRD}^5_{5\times5}$	0.51	1.53	5.48	24.97	53.3
	0.43	0.78	2.25	8.01	21.6
	0.005	0.015	0.054	0.18	0.39
$\text{TRD}^5_{7\times7}$	1.21	3.72	14.0	62.2	135.9
	0.56	1.17	3.64	13.01	30.1
	0.01	0.032	0.116	0.40	0.87

Table 4.2: Runtime comparison for image denoising (in seconds) with different implementations. (1) The runtime results with gray background are evaluated with the single-threaded implementation on Intel(R) Xeon(R) CPU E5-2680 v2 @ 2.80GHz; (2) the blue colored runtimes are obtained from the multi-threaded implementation on a server with the above CPUs; (3) the runtime results colored in red are executed on a NVIDIA GeForce GTX 780Ti GPU.

on the exploited test dataset, meanwhile it is the fastest method even with the CPU implementation.

4.5.1.4 Run time

The algorithm structure of our TRD model is closely similar to the CSF model, which is well-suited for parallel computation on DSP or GPU. We implemented our trained models on GPU using CUDA programming to speed up the inference procedure, and finally it indeed lead to a significantly improved runtime, see Table 4.2. We see that for the images of size up to 3K $\times$ 3K, the $\text{TRD}^5_{7\times7}$ model is still able to accomplish the denoising task in less than 1s.

We make a runtime comparison to other denoising algorithms based on strictly enforced single-threaded CPU computation (e.g., start Matlab with -singleCompThread) for a fair comparison, see Table 4.2. We only present the results of some selective algorithms, which either have the best denoising performance or runtime result. We refer to [120] for a comprehensive runtime comparison of various algorithms.

We see that our trained TRD model is generally faster than the CSF model with the same model capacity. It is reasonable, because in each stage the CSF model involves additional DFT and inverse DTF operations, i.e., our model only requires a portion of the computation of the CSF model. Even though the BM3D is a non-local model, it still possesses high computation efficiency, in contrast to another non-local model WNNM, which achieves compelling denoising results at the expense of computation time. Moreover,

the WNNM algorithm is hardly applicable for high resolution images (e.g., 10 mega-pixels) due to its huge memory requirement. Note that our model can be also easily implemented with multi-threaded CPU computation, which is generally able to speed up the single-threaded implementation by 2-4 times, see Table 4.2.

4.5.1.5 Denoising examples

We present the trained filters of the model $\text{TRD}^5_{5\times5}$ in the case of Gaussian noise level $\sigma = 25$ in Figure 4.7 to illustrate the property of the learned filters. We see find the first, second derivative filters and their rotated versions, as well as higher-order filters.

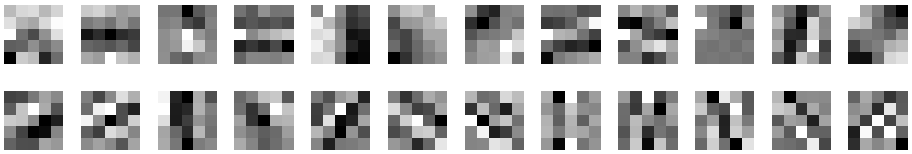
We present three Gaussian denoising examples in Figure 4.8 - Figure 4.10. We compare the results of our trained TRD models with a selection of representative state-of-the-art approaches: (1) the benchmark denoising algorithm - BM3D, (2) our learned opt-MRF model in the last chapter, one of the best-performing MRFs for image restoration, (3) WNNM algorithm, the currently best-reported denoising algorithm, (4) $\text{CSF}^5_{7\times7}$ model, having the best trade-off between performance and computation time. For the exploited image size of 481×321, the corresponding run time of the competing algorithms are shown in Figure 4.9 based on either CPU or GPU computation.

Note that in Figure 4.10, the test image contains many repeated local patterns, e.g., the T-shirt region, the nonlocal methods (BM3D and WNNM) generally should work better than the local methods (CSF model and our TRD model), because the nonlocal models explicitly exploit nonlocal self-similarity across the image. Even though the nonlocal methods can benefit from these repeated local patterns, our trained $\text{TRD}^5_{7\times7}$ still show strongly competitive performance.

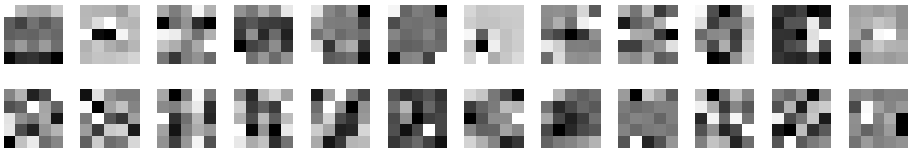
4.5.2 JPEG deblocking experiments

We also trained diffusion models for the JPEG deblocking problem. We followed the test procedure in [68] for performance evaluation. The test images were converted to gray-value, and scaled by a factor of 0.5, resulting images of size 240×160. We distorted the images by JPEG blocking artifacts. We considered three compression quality settings $q = 10, 20$ and 30 for the JPEG encoder. Results are shown in Table 4.3, compared with several representative deblocking approaches, as well as the opt-MRF model based deblocking algorithm presented in the last chapter Section 3.2.

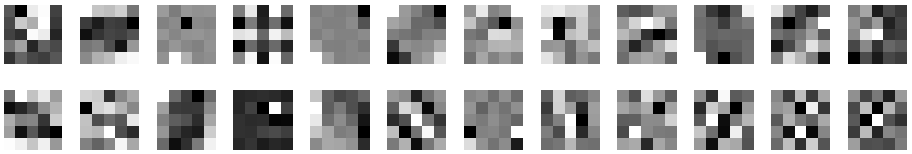
We see that our trained $\text{TRD}^4_{7\times7}$ outperforms all the competing approaches in terms of the PSNR index. Furthermore, our model is very fast on GPU, e.g., for a common



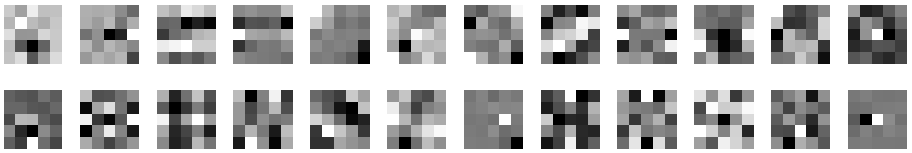
(a) Learned filters of stage 1



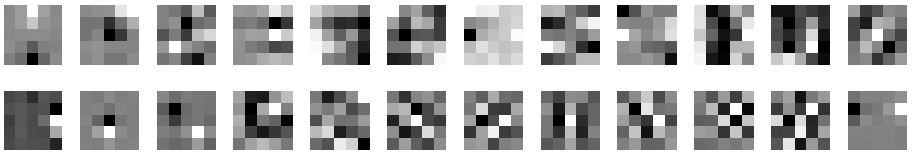
(b) Learned filters of stage 2



(c) Learned filters of stage 3



(d) Learned filters of stage 4

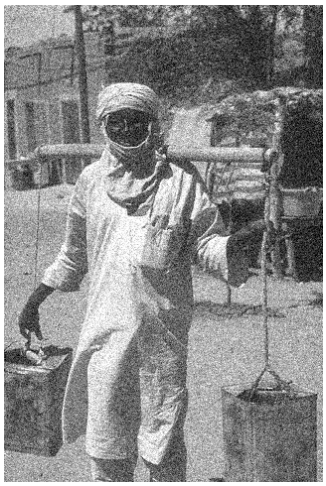


(e) Learned filters of stage 5

Figure 4.7: The trained filters of the model $\text{TRD}^5_{5\times5}$ in the case of Gaussian noise level $\sigma = 25$.

image size of 1024×1024, our model takes about 95ms, while the strongest competitor - SADCT consumes about 56.5s with CPU computation. We present four JPEG deblocking

(a) clean image



(b) $\sigma = 25$ (20.17)

(c) BM3D (28.60)

(d) $\text{CSF}^5_{7\times7}$ (28.83)

(e) WNNM (28.82)

(f) opt-MRF (28.61)

(g) $\text{TRD}^5_{5\times5}$ (28.85)

(h) $\text{TRD}^5_{7\times7}$ (28.97)

Figure 4.8: Denoising example for an image with noise $\sigma = 25$.

(a) clean image

(b) $\sigma = 25$ (20.17)

(c) BM3D (36.78)/**CPU: 2.5s**

(d) $\text{CSF}^5_{7\times7}$ (37.15)/**GPU: 0.55s**

(e) WNNM (36.95)/**CPU: 393.2s**

(f) opt-MRF (36.84)/**GPU: 0.138s**

(g) $\text{TRD}^5_{5\times5}$ (37.04)/**GPU: 9.1ms**

(h) $\text{TRD}^5_{7\times7}$ (37.64)/**GPU: 20.3ms**

Figure 4.9: Denoising example for an image with noise $\sigma = 25$ together with the corresponding computation time on GPU or CPU.

(a) clean image (b) Noisy, 20.17dB (c) BM3D, 33.24dB

(d) $\text{CSF}^5_{7\times7}$, 32.93dB (e) WNNM, 33.77dB (f) opt-MRF (32.27)

(g) $\text{TRD}^5_{5\times5}$, 32.91dB (h) $\text{TRD}^5_{7\times7}$, 33.34dB

Figure 4.10: Denoising results on a test images ($\sigma = 25$). Note the effectiveness of our trained $\text{TRD}^5_{7\times7}$ model for those regions with repeated local pattern (indicated by the red arrow).

q	JPEG decoder	TGV [16]	Dic-SR[24]	SADCT [49]	RTF[68]	$\text{TRD}^4_{7\times7}$
10	26.59	26.96	27.15	27.43	27.68	**27.85**
20	28.77	29.01	29.03	29.46	29.83	**30.06**
30	30.05	30.25	30.13	30.67	31.14	**31.41**

Table 4.3: JPEG deblocking results for natural images, reported with average PSNR values.

examples in Figure 4.11. Note that the variational model (opt-MRF) sometimes fails to remove the blocking artifacts in the sky, while the newly trained $\text{TRD}^4_{7\times7}$ succeeds.

An additional deblocking example based on common size of 512×512 is shown in Figure 4.12, together with the corresponding computation time. While the competing approaches achieve similar deblocking performance, the trained $\text{TRD}^4_{7\times7}$ provide the best run time.

4.6 Discussion

In this chapter, we have proposed a trainable reaction diffusion model for effective image restoration. Its critical point lies in the training of the influence functions. We have trained our models for the problem of Gaussian denoising and JPEG deblocking. Based on standard test datasets, the trained models result in the best-reported results. Meanwhile, the trained models are very simple and well-suited for parallel computation on GPUs. As a consequence, the resulting algorithms are significantly faster than all competing algorithms and hence are also applicable to the restoration of high resolution images.

We believe that the effectiveness of the trained diffusion models is attributed to the following desired properties of the models

- *Anisotropy.* In the trained filters, we can find rotated derivative filters in different directions, *cf.* Fig 4.7.
- *Higher order.* The learned filters contain first, second and higher-order derivative filters, *cf.* Fig 4.7.
- *Adaptive forward/backward diffusion through the learned nonlinear functions.* Nonlinear functions corresponding to explicit backward diffusion appear in the learned nonlinearity, *cf.* Fig 4.6.

(a) Clean image (b) Clean image (c) Clean image (d) Clean image

(e) Lossy (30.02) (f) Lossy (27.59) (g) Lossy (24.24) (h) Lossy (28.56)

(i) SADCT[49] (30.69) (j) SADCT[49] (29.63) (k) SADCT[49] (25.54) (l) SADCT[49] (29.53)

(m) RTF[68] (30.93) (n) RTF[68] (29.16) (o) RTF[68] (26.37) (p) RTF[68] (29.94)

(q) opt-MRF (30.93) (r) opt-MRF (29.03) (s) opt-MRF (25.96) (t) opt-MRF (30.00)

(u) $\mathrm{TRD}^4_{7\times7}$ (31.04) (v) $\mathrm{TRD}^4_{7\times7}$ (29.47) (w) $\mathrm{TRD}^4_{7\times7}$ (27.05) (x) $\mathrm{TRD}^4_{7\times7}$ (30.17)

Figure 4.11: Image deblocking for images compressed by JPEG encoder with the quality $q = 10$. **Note the difference in the sky.**

(a) Clean image

(b) Compressed with $q = 10$ (30.41)

(c) SA-DCT (31.84)/**CPU: 17.6s**

(d) opt-MRF (31.09)/**GPU: 0.23s**

(e) $\text{TRD}^4_{7\times7}$ (32.22)/**GPU: 25ms**

Figure 4.12: Debloking of the "Lena" of a ordinary size of 512×512, together with corresponding computation time.

Chapter 5

Conclusion

In this thesis, we have proposed two fast and effective image restoration models, namely:

- variational model with learned FoE image regularization, and
- non-linear diffusion model optimized with loss-minimization.

For the variational image restoration model, we concentrate on the problem of learning better image regularization term. Due to its simplicity and effectiveness, we mainly focused on the Fields of Experts (FoE) image prior model. We have reviewed existing learning approaches to train the FoE model, and we are motivated to revisit the loss-specific training strategy, because we believe that the performance of the loss-specific training approach has been underestimated in previous work. We proposed a refined training algorithm to investigate the loss-specific training model, which is defined as a bi-level optimization problem. Our refined training algorithm suggested to solve the lower-level problem in the bi-level framework with higher accuracy, i.e., using a more conservative stopping criterion for the corresponding minimization problem. It turns out that this seemingly minor modification has brought significant improvements relative to previous work. Numerical experiments have demonstrated that solving the lower-level problem with higher accuracy is indeed beneficial for the training.

We also have revealed an interesting link between the FoE image prior model and a recently proposed sparse representation based model called analysis operator learning (analysis prior model). We expressed our insights into the analysis prior model. We proposed to go beyond existing patch-based models, and to exploit the framework of FoE model to define a image prior over the entire image, rather than image patches. Then we pointed out that the image based analysis model is equivalent to the FoE model. Based

on this conclusion, we suggested to employ the bi-level training approach (i.e., loss-specific training) for analysis operator learning. By using our training framework, we have carefully investigated the effect of different aspects of the analysis prior model including the filter size, the number of filters and the penalty function. Numerical results have demonstrated that the penalty function is the most important factor for this model.

The FoE image prior regularized variational model naturally leads to a non-convex optimization problem, due to the non-convex penalty function involved in the FoE model. In general, the resulting minimization problems pose demanding non-convex optimization problems. In order to efficiently solve these non-convex optimization problems, we have proposed a fast non-convex optimization algorithm called iPiano. Our proposed iPiano algorithm is applicable for a class of non-convex problems, which are composed of a differentiable (possibly non-convex) and a convex (possibly non-differentiable) function. The algorithm combines forward-backward splitting with an inertial force called heavy ball method.

All the investigated image restoration problems can be efficiently solved by using our proposed iPiano algorithm, typically in less than 100 iterations. Due to the structure simplicity of the FoE image prior model, the resulting variational models are well-suited for GPU parallel computation. We have implemented some of our variational image restoration models on GPU with CUDA programming, which can typically accelerate the inference procedure with a factor of $40\times$ relative to a multi-threaded CPU implementation.

Concerning the image reconstruction quality, the image restoration performance of our trained variational models have been compared with corresponding state-of-the-art algorithms to demonstrate the effectiveness. Numerical results have shown that the variational models with learned image regularizer can lead to recovery quality on par with state-of-the-arts, meanwhile possess the additional advantage of high computational efficiency.

In addition, motivated by the natural link between the energy functional minimization and nonlinear diffusion process, we have proposed a trainable reaction diffusion model for effective image restoration, which are derived from the FoE image prior model regularized variational model. The training model is parameterized by the linear filters in the FoE model and the nonlinearity imposed to the filter response (also known as influence function in the context of nonlinear diffusion process). We have demonstrated that the additional flexibility to tune the nonlinearity is the critical point of this training model.

In this thesis, we have trained effective reaction diffusion processes for two classic image restoration problems: (1) the standard test bed - Gaussian denoising and (2) a non-

smooth problem - JPEG deblocking task. Based on standard test datasets, the trained nonlinear diffusion models lead to very compelling results, which exhibit the best-reported performance. Meanwhile, as the trained diffusion processes only involves $4-5$ steps, they are very fast. As we know, the FoE model is very suited for GPU structure. Therefore, the trained nonlinear diffusion processes, which are derived from the FoE image prior model, are also well-suited for GPU parallel computation. As a consequence, the resulting algorithms based on GPU are faster than related algorithms and are undoubtedly applicable to the restoration of high resolution images. With the GPU implementation, the trained diffusion model has successfully accomplished the exploited image restoration problems in less than $1s$ for the images of size up to $3K \times 3K$.

5.1 Future work

In this thesis, we have demonstrated two effective image restoration models closely related to higher-order MRFs, specifically filter-based MRF model called FoE proposed by Roth and Black [111]. We concentrated on the MAP inference other than the sampling based MMSE inference to use the higher-order MRF models, due to its efficiency. Keeping in mind the good results achieved in this thesis, we think the following issues are worthwhile to investigate in future work.

5.1.1 Further investigation of the variational model with learned image regularizer

As shown in this thesis, although the FoE image prior model is trained based on the Gaussian denoising problem, it is not heavily tied to this specific problem. In contrast, it can be applied to many other image restoration problems. Therefore, a straightforward work is to extend the learned image prior model to other possible image processing tasks to renew existing results. Another possible extension of the application is to retrain the FoE image prior model for specialized image type, e.g., medical images like MRI, CT etc., to achieve some specialized image prior model.

It is also interesting to investigate whether it is helpful to retrain the FoE model for specific image restoration problem, such as image deconvolution or image super resolution.

In the training of the nonlinear diffusion process, we have revealed some unconventional penalty functions (e.g., the negative Mexican hat function), which have significant meaning in visual computation. These penalty functions, such as the negative Mexican hat function,

goes a step further relative to the conventional non-convex penalty function, which is a increasing function of $|z|$, and always has a minimum at zero*. Clearly, the conventional non-convex penalty function based regularization is merely a smooth term. However, the unconventional penalty function based regularization is not a smooth term any more; in contrast, it encourages some image patterns. Therefore, it is very interesting to investigate the regularization technique based on unconventional penalty functions. Obviously, the corresponding minimization problem will become harder to solve. This is subject to our future work.

5.1.2 Trainable nonlinear reaction diffusion models

Given the significant meaning and the effectiveness of the TRD models, we believe that it is worthwhile to revisit nonlinear diffusion based image processing problems with appropriate training, such as image super resolution, image deconvolution, image enhancement, etc. As shown in our work, the trained nonlinear diffusion model leads to a diffusion process which can automatically switch between forward (image smoothing) and backward (image sharpening) process. In addition, this diffusion process can generate image patterns or structure from some slight evidence. This property is desired for many image processing task such as image super resolution. Therefore, it is definitely worthwhile to train more nonlinear diffusion models for specialized image processing tasks.

There are also some work we can do for the training model itself. In our current work, even though we have achieved nonlinear diffusion models which can generate very compelling results, we find that there are some problems with the trained models. For example, some trained influence functions are not smooth, and there are oscillations. We believe that the reason lies in insufficient training data. This should be the consequence of overfitting. We believe that universal influence functions should be smooth. A straightforward solution for this issue is to make use of sufficiently larger training datasets. However, with our current CPU implementation for training, it is not feasible to drastically increase the training datasets. A feasible solution is to implement the training procedure on GPU, as the we know this diffusion process is well-suited for GPU parallel computation. Our future

*We know that although the diffusion process involving the conventional non-convex penalty function owns the property of edge preserving, it can not stop the diffusion across the edge, and it can only slow down the diffusion across the edge, as the penalty function always has a punishment on those non-zero filter response. Therefore, the final state of this diffusion process is a trivial constant value. However, for the unconventional penalty function controlled diffusion process, if the filter response is large enough, the penalty function does not impose punishment on the filter response any more, and in contrast, it encourages to increase the filter response.

work is to execute the training on GPU. With highly efficiently GPU implementation, we can investigate more possible configurations for the training.

Another issue is to study and compare two different function approximation approaches, namely RBF with Gaussian radial basis and triangular-shaped radial basis.

5.1.3 Some relations to convolutional neural networks

As shown in our work, our proposed nonlinear diffusion process can be regarded as a convolutional neural network with special architecture. Our training experiences show that optimizing the nonlinear functions in a CNN model has great potential to improve the performance. This provides a good motivation to tune the nonlinearity in the standard CNN models, which typical employ the fixed activation function.

On the other hand, our proposed nonlinear diffusion process corresponds to a recurrent neural network. In our case, the CNN model is derived from an energy functional. This new architecture might inspire a new type of CNN models, which could also be applied to pattern recognition problems, as in the case of conventional CNN models.

Appendix A

Implementation of the transpose operation $K^\top$

In the trained model, we need to compute the matrix multiplication result of the transpose matrix $K^\top$ with a vector v. As in practice explicitly constructing the huge matrix K is very time consuming, a fast implementation to compute $K^\top v$ is required.

If Ku is implemented with the periodic boundary condition, the result of $K^\top v$ is simply given by a convolution operation $\bar{k} * v$, where $\bar{k}$ is obtained by mirroring the convolution kernel k around its center point, e.g.,

$$k = \begin{bmatrix} -0.3902 & -0.2338 & 0.2690 \\ 0.4338 & 0.2978 & -0.1040 \\ -0.3125 & -0.0124 & -0.2271 \end{bmatrix} \Longrightarrow \bar{k} = \begin{bmatrix} -0.2271 & -0.0124 & -0.3125 \\ -0.1040 & 0.2978 & 0.4338 \\ 0.2690 & -0.2338 & -0.3902 \end{bmatrix}$$

However, in our model, we are employing the symmetric boundary condition, where $K^\top v$ can be interpreted as the convolution with the kernel $\bar{k}$ only in the central region of image v. Therefore, we need to carefully handle the boundary regions. We implement the computation of $K^\top v$ with C programming as follows *mex_convolution_transpose.cpp*.

```
#include "mex.h"
#include <malloc.h>
#include <math.h>
//////////////////////////////////////////////////////////
// compile with: mex  mex_convolution_transpose.cpp
//////////////////////////////////////////////////////////
// entry function
void mexFunction(int nlhs, mxArray *plhs[],int nrhs, const mxArray *prhs[]){
```

```
double *input,*output,*k,*filter;
int filter_size,nRow,nCol,N;
if (nrhs !=4 || nlhs!=1)
 mexErrMsgTxt("Invalid number of input/output arguments!");
input = mxGetPr(prhs[0]);
k = mxGetPr(prhs[1]);
filter_size = mxGetM(prhs[1]);
nRow = mxGetScalar(prhs[2]);
nCol = mxGetScalar(prhs[3]);
N = (filter_size - 1)/2;
plhs[0] = mxCreateDoubleMatrix(nRow*nCol,1,mxREAL);
output = mxGetPr(plhs[0]);
if (filter_size != mxGetN(prhs[1]) || filter_size%2 != 1)
 mexErrMsgTxt("Invalid filter(only valid for square and odd size filter)!");
// reverse filter
filter = (double *)mxCalloc(filter_size*filter_size,sizeof(double));
for (int idx = 0; idx < filter_size*filter_size; idx++)
 filter[idx] = k[filter_size*filter_size - idx - 1];
// handle central region for example
int col,row;
for (col = N; col < nCol-N; col++)
{
 for (row = N; row < nRow-N; row++)
 {
  int idx = 0;
  int idx_col,idx_row;
  double response = 0;
  for (int filter_col = -N; filter_col<=N; filter_col++)
  {
   for (int filter_row = -N; filter_row<=N; filter_row++)
   {
    idx_col = col + filter_col;
    idx_row = row + filter_row;
    response = response + input[idx_row + idx_col*nRow] * filter[idx];
    idx++;
   }
  }
  output[row + col*nRow] = response;
 }
}
// handle boudaryies
// left side boundary
for (col = 0; col < N; col++)
```

```
{
 double *temp_filter = (double *)mxCalloc(filter_size*filter_size,sizeof(double));
 for (int idx_col = 0; idx_col < N-col; idx_col++)
 {
  for (int idx_row = 0; idx_row < filter_size; idx_row++)
   temp_filter[idx_row + idx_col*filter_size] = filter[idx_row + (2*N-idx_col)*filter_size];
 }
 for (int idx_col = N-col; idx_col < filter_size; idx_col++)
 {
  for (int idx_row = 0; idx_row < filter_size; idx_row++)
   temp_filter[idx_row + idx_col*filter_size] = filter[idx_row + idx_col*filter_size];
 }
 for (row = N; row < nRow-N; row++)
 {
  int idx = 0;
  int idx_col,idx_row;
  double response = 0;
  for (int filter_col = -N; filter_col<=N; filter_col++)
  {
   for (int filter_row = -N; filter_row<=N; filter_row++)
   {
    idx_col = col + filter_col;
    idx_row = row + filter_row;
    if (idx_col < 0)
     idx_col = -1 - idx_col;
    response = response + input[idx_row + idx_col*nRow] * temp_filter[idx];
    idx++;
   }
  }
  output[row + col*nRow] = response;
 }
}
// right side boundary
for (col = nCol-N; col < nCol; col++)
{
 double *temp_filter = (double *)mxCalloc(filter_size*filter_size,sizeof(double));
 for (int idx_col = 0; idx_col < N-(nCol-col)+1; idx_col++)
 {
  for (int idx_row = 0; idx_row < filter_size; idx_row++)
   temp_filter[idx_row + (2*N-idx_col)*filter_size] = filter[idx_row + idx_col*filter_size];
 }
 for (int idx_col = N-(nCol-col)+1; idx_col < filter_size; idx_col++)
 {
```

```
  for (int idx_row = 0; idx_row < filter_size; idx_row++)
   temp_filter[idx_row + (2*N-idx_col)*filter_size] = filter[idx_row + (2*N-idx_col)*filter_size];
 }
 for (row = N; row < nRow-N; row++)
 {
  int idx = 0;
  int idx_col,idx_row;
  double response = 0;
  for (int filter_col = -N; filter_col<=N; filter_col++)
  {
   for (int filter_row = -N; filter_row<=N; filter_row++)
   {
    idx_col = col + filter_col;
    idx_row = row + filter_row;
    if (idx_col >= nCol)
     idx_col = 2*nCol - (idx_col+1);
    response = response + input[idx_row + idx_col*nRow] * temp_filter[idx];
    idx++;
   }
  }
  output[row + col*nRow] = response;
 }
}
// top boundary
for (row = 0; row < N; row++)
{
 double *temp_filter = (double *)mxCalloc(filter_size*filter_size,sizeof(double));
 for (int idx_row = 0; idx_row < N-row; idx_row++)
 {
  for (int idx_col = 0; idx_col < filter_size; idx_col++)
   temp_filter[idx_row + idx_col*filter_size] = filter[2*N - idx_row + idx_col*filter_size];
 }
 for (int idx_row = N-row; idx_row < filter_size; idx_row++)
 {
  for (int idx_col = 0; idx_col < filter_size; idx_col++)
   temp_filter[idx_row + idx_col*filter_size] = filter[idx_row + idx_col*filter_size];
 }
 for (col = N; col < nCol-N; col++)
 {
  int idx = 0;
  int idx_col,idx_row;
  double response = 0;
  for (int filter_col = -N; filter_col<=N; filter_col++)
```

```
  {
   for (int filter_row = -N; filter_row<=N; filter_row++)
   {
    idx_col = col + filter_col;
    idx_row = row + filter_row;
    if (idx_row < 0)
     idx_row = -1 - idx_row;
    response = response + input[idx_row + idx_col*nRow] * temp_filter[idx];
    idx++;
   }
  }
  output[row + col*nRow] = response;
 }
}
// bottom boudanry
for (row = nRow-N; row < nRow; row++)
{
 double *temp_filter = (double *)mxCalloc(filter_size*filter_size,sizeof(double));
 for (int idx_row = 0; idx_row < N-(nRow-row)+1; idx_row++)
 {
  for (int idx_col = 0; idx_col < filter_size; idx_col++)
   temp_filter[2*N - idx_row + idx_col*filter_size] = filter[idx_row + idx_col*filter_size];
 }
 for (int idx_row = N-(nRow-row)+1; idx_row < filter_size; idx_row++)
 {
  for (int idx_col = 0; idx_col < filter_size; idx_col++)
   temp_filter[2*N - idx_row + idx_col*filter_size] = filter[2*N - idx_row + idx_col*filter_size];
 }
 for (col = N; col < nCol-N; col++)
 {
  int idx = 0;
  int idx_col,idx_row;
  double response = 0;
  for (int filter_col = -N; filter_col<=N; filter_col++)
  {
   for (int filter_row = -N; filter_row<=N; filter_row++)
   {
    idx_col = col + filter_col;
    idx_row = row + filter_row;

    if (idx_row >= nRow)
     idx_row = 2*nRow - (idx_row+1);
    response = response + input[idx_row + idx_col*nRow] * temp_filter[idx];
```

```
    idx++;
   }
  }
  output[row + col*nRow] = response;
 }
}
// handle left two blocks
for (col = 0; col < N; col++)
{
 double *temp_filter = (double *)mxCalloc(filter_size*filter_size,sizeof(double));
 for (int idx_col = 0; idx_col < N-col; idx_col++)
 {
  for (int idx_row = 0; idx_row < filter_size; idx_row++)
   temp_filter[idx_row + idx_col*filter_size] = filter[idx_row + (2*N-idx_col)*filter_size];
 }
 for (int idx_col = N-col; idx_col < filter_size; idx_col++)
 {
  for (int idx_row = 0; idx_row < filter_size; idx_row++)
   temp_filter[idx_row + idx_col*filter_size] = filter[idx_row + idx_col*filter_size];
 }
 // top block
 for (row = 0; row < N; row++)
 {
  double *temp_filter2 = (double *)mxCalloc(filter_size*filter_size,sizeof(double));
  for (int idx_row = 0; idx_row < N-row; idx_row++)
  {
   for (int idx_col = 0; idx_col < filter_size; idx_col++)
    temp_filter2[idx_row + idx_col*filter_size] = temp_filter[2*N - idx_row + idx_col*filter_size];
  }
  for (int idx_row = N-row; idx_row < filter_size; idx_row++)
  {
   for (int idx_col = 0; idx_col < filter_size; idx_col++)
    temp_filter2[idx_row + idx_col*filter_size] = temp_filter[idx_row + idx_col*filter_size];
  }
  int idx = 0;
  int idx_col,idx_row;
  double response = 0;
  for (int filter_col = -N; filter_col<=N; filter_col++)
  {
   for (int filter_row = -N; filter_row<=N; filter_row++)
   {
    idx_col = col + filter_col;
    idx_row = row + filter_row;
```

```
    if (idx_col < 0)
     idx_col = -1 - idx_col;
    if (idx_row < 0)
     idx_row = -1 - idx_row;
    response = response + input[idx_row + idx_col*nRow] * temp_filter2[idx];
    idx++;
   }
  }
  output[row + col*nRow] = response;
}
// bottom block
for (row = nRow-N; row < nRow; row++)
{
 double *temp_filter2 = (double *)mxCalloc(filter_size*filter_size,sizeof(double));
 for (int idx_row = 0; idx_row < N-(nRow-row)+1; idx_row++)
 {
  for (int idx_col = 0; idx_col < filter_size; idx_col++)
   temp_filter2[2*N - idx_row + idx_col*filter_size] = temp_filter[idx_row + idx_col*filter_size];
 }
 for (int idx_row = N-(nRow-row)+1; idx_row < filter_size; idx_row++)
 {
  for (int idx_col = 0; idx_col < filter_size; idx_col++)
   temp_filter2[2*N - idx_row + idx_col*filter_size] = temp_filter[2*N - idx_row + idx_col*filter_size];
 }
 int idx = 0;
 int idx_col,idx_row;
 double response = 0;
 for (int filter_col = -N; filter_col<=N; filter_col++)
 {
  for (int filter_row = -N; filter_row<=N; filter_row++)
  {
   idx_col = col + filter_col;
   idx_row = row + filter_row;
   if (idx_col < 0)
    idx_col = -1 - idx_col;
   if (idx_row >= nRow)
    idx_row = 2*nRow - (idx_row+1);
   response = response + input[idx_row + idx_col*nRow] * temp_filter2[idx];
   idx++;
  }
 }
 output[row + col*nRow] = response;
}
```

```
}
// handle right two blocks
for (col = nCol-N; col < nCol; col++)
{
 double *temp_filter = (double *)mxCalloc(filter_size*filter_size,sizeof(double));
 for (int idx_col = 0; idx_col < N-(nCol-col)+1; idx_col++)
 {
  for (int idx_row = 0; idx_row < filter_size; idx_row++)
   temp_filter[idx_row + (2*N-idx_col)*filter_size] = filter[idx_row + idx_col*filter_size];
 }
 for (int idx_col = N-(nCol-col)+1; idx_col < filter_size; idx_col++)
 {
  for (int idx_row = 0; idx_row < filter_size; idx_row++)
   temp_filter[idx_row + (2*N-idx_col)*filter_size] = filter[idx_row + (2*N-idx_col)*filter_size];
 }
 // top block
 for (row = 0; row < N; row++)
 {
  double *temp_filter2 = (double *)mxCalloc(filter_size*filter_size,sizeof(double));
  for (int idx_row = 0; idx_row < N-row; idx_row++)
  {
   for (int idx_col = 0; idx_col < filter_size; idx_col++)
    temp_filter2[idx_row + idx_col*filter_size] = temp_filter[2*N - idx_row + idx_col*filter_size];
  }
  for (int idx_row = N-row; idx_row < filter_size; idx_row++)
  {
   for (int idx_col = 0; idx_col < filter_size; idx_col++)
    temp_filter2[idx_row + idx_col*filter_size] = temp_filter[idx_row + idx_col*filter_size];
  }
  int idx = 0;
  int idx_col,idx_row;
  double response = 0;
  for (int filter_col = -N; filter_col<=N; filter_col++)
  {
   for (int filter_row = -N; filter_row<=N; filter_row++)
   {
    idx_col = col + filter_col;
    idx_row = row + filter_row;
    if (idx_col >= nCol)
     idx_col = 2*nCol - (idx_col+1);
    if (idx_row < 0)
     idx_row = -1 - idx_row;
    response = response + input[idx_row + idx_col*nRow] * temp_filter2[idx];
```

```
    idx++;
   }
  }
  output[row + col*nRow] = response;
 }
 // bottom block
 for (row = nRow-N; row < nRow; row++)
 {
  double *temp_filter2 = (double *)mxCalloc(filter_size*filter_size,sizeof(double));
  for (int idx_row = 0; idx_row < N-(nRow-row)+1; idx_row++)
  {
   for (int idx_col = 0; idx_col < filter_size; idx_col++)
    temp_filter2[2*N - idx_row + idx_col*filter_size] = temp_filter[idx_row + idx_col*filter_size];
  }
  for (int idx_row = N-(nRow-row)+1; idx_row < filter_size; idx_row++)
  {
   for (int idx_col = 0; idx_col < filter_size; idx_col++)
    temp_filter2[2*N - idx_row + idx_col*filter_size] = temp_filter[2*N - idx_row + idx_col*filter_size];
  }
  int idx = 0;
  int idx_col,idx_row;
  double response = 0;
  for (int filter_col = -N; filter_col<=N; filter_col++)
  {
   for (int filter_row = -N; filter_row<=N; filter_row++)
   {
    idx_col = col + filter_col;
    idx_row = row + filter_row;
    if (idx_col >= nCol)
     idx_col = 2*nCol - (idx_col+1);
    if (idx_row >= nRow)
     idx_row = 2*nRow - (idx_row+1);
    response = response + input[idx_row + idx_col*nRow] * temp_filter2[idx];
    idx++;
   }
  }
  output[row + col*nRow] = response;
 }
}
}
```

Bibliography

[SAR] `http://academic.emporia.edu/aberjame/student/graves1/project.html`.

[2] Acton, S. T., Prasad Mukherjee, D., Havlicek, J. P., and Conrad Bovik, A. (2001). Oriented texture completion by AM-FM reaction-diffusion. *IEEE TIP*, 10(6):885–896.

[3] Aharon, M., Elad, M., and Bruckstein, A. (2006). K-SVD: An algorithm for designing over-complete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322.

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

[5] Argenti, F., Bianchi, T., Lapini, A., and Alparone, L. (2012). Fast MAP despeckling based on laplacian-gaussian modeling of wavelet coefficients. *IEEE Geosci. Remote Sensing Lett.*, 9(1):13–17.

[6] Attouch, H., Goudou, X., and Redont, P. (2000). The heavy ball with friction method, i. the continuous dynamical system: global exploration of the local minima of a real-valued function by asymptotic analysis of a dissipative dynamical system. *Communications in Contemporary Mathematics*, 2(01):1–34.

[7] Aubert, G. and Aujol, J.-F. (2008). A variational approach to removing multiplicative noise. *SIAM Journal of Applied Mathematics*, 68(4):925–946.

[8] Barbu, A. (2009). Training an active random field for real-time image denoising. *IEEE Trans. on Image Proc*, 18(11):2451–2462.

[9] Barcelos, C. A. Z., Boaventura, M., and Silva Jr, E. C. (2003). A well-balanced flow equation for noise removal and edge detection. *IEEE TIP*, 12(7):751–763.

[10] Benning, M., Brune, C., Burger, M., and Müller, J. (2013). Higher-order TV methods - enhancement via bregman iteration. *Journal of Scientific Computing*, 54(2-3):269–310.

[11] Bertsekas, D. P. (1999). *Nonlinear Programming*. Athena Scientific, 2nd edition.

[12] Bianchi, T., Argenti, F., and Alparone, L. (2008). Segmentation-based MAP despeckling of SAR images in the undecimated wavelet domain. *IEEE T. Geoscience and Remote Sensing*, 46(9):2728–2742.

[13] Black, M., Sapiro, G., Marimont, D., and Heeger, D. (1997). Robust anisotropic diffusion and sharpening of scalar and vector images. In *ICIP*, pages 263–266. IEEE.

[14] Bouvrie, J. (2006). Notes on convolutional neural networks.

[15] Bredies, K. and Holler, M. (2012a). Artifact-free JPEG decompression with total generalized variation. In *VISAPP (1)*, pages 12–21.

[16] Bredies, K. and Holler, M. (2012b). Artifact-free jpeg decompression with total generalized variation. In *VISAPP (1)*, pages 12–21.

[17] Bredies, K. and Holler, M. (2012c). A total variation-based JPEG decompression model. *SIAM J. Imaging Sciences*, 5(1):366–393.

[18] Bredies, K., Kunisch, K., and Pock, T. (2010). Total generalized variation. *SIAM Journal on Imaging Sciences*, 3(3):492–526.

[19] Brox, T., Welk, M., Steidl, G., and Weickert, J. (2003). Equivalence results for tv diffusion and tv regularisation. In *Scale Space Methods in Computer Vision*, pages 86–100. Springer.

[20] Candes, E. J., Wakin, M. B., and Boyd, S. P. (2008). Enhancing sparsity by reweighted ℓ_1 minimization. *Journal of Fourier analysis and applications*, 14(5-6):877–905.

[21] Chambolle, A. and Pock, T. (2011a). A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145.

[22] Chambolle, A. and Pock, T. (2011b). A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145.

[23] Chan, T., Marquina, A., and Mulet, P. (2000). High-order total variation-based image restoration. *SIAM Journal on Scientific Computing*, 22(2):503–516.

[24] Chang, H., Ng, M. K., and Zeng, T. (2014). Reducing artifact in JPEG decompression via a learned dictionary. *IEEE Transactions on Signal Processing*, 62(3):718–728.

[25] Chen, Y., Feng, W., Ranftl, R., Qiao, H., and Pock, T. (2014a). A higher-order MRF based variational model for multiplicative noise reduction. *IEEE Signal Process. Lett.*, 21(11):1370–1374.

[26] Chen, Y., Pock, T., and Bischof, H. (2012). Learning ℓ_1-based analysis and synthesis sparsity priors using bi-level optimization. In *Workshop on Analysis Operator Learning vs. Dictionary Learning, NIPS 2012*.

[27] Chen, Y., Pock, T., Ranftl, R., and Bischof, H. (2013). Revisiting loss-specific training of filter-based mrfs for image restoration. In *Pattern Recognition - 35th German Conference, GCPR 2013, Saarbrücken, Germany, September 3-6, 2013. Proceedings*, pages 271–281.

[28] Chen, Y., Ranftl, R., and Pock, T. (2014b). A bi-level view of inpainting - based image compression. In *19th Computer Vision Winter Workshop, CVWW 2014, Křiny, Czech Republic, 3-5th February 2014*, pages 19–26.

[29] Chen, Y., Ranftl, R., and Pock, T. (2014c). Insights into analysis operator learning: From patch-based sparse models to higher order mrfs. *IEEE Transactions on Image Processing*, 23(3):1060–1072.

[30] Chen, Y., Yu, W., and Pock, T. (2014d). On learning optimized reaction diffusion processes for effective image restoration. *Preprint*.

[31] Colson, B., Marcotte, P., and Savard, G. (2007). An overview of bilevel optimization. *Annals OR*, 153(1):235–256.

[32] Corless, R. M., Gonnet, G. H., Hare, D. E. G., Jeffrey, D. J., and Knuth, D. E. (1996). On the lambert W function. *Adv. Comput. Math.*, 5(1):329–359.

[33] Cottet, G.-H. and Germain, L. (1993). Image processing through reaction combined with nonlinear diffusion. *Mathematics of Computation*, pages 659–673.

[34] Cozzolino, D., Parrilli, S., Scarpa, G., Poggi, G., and Verdoliva, L. (2014). Fast adaptive nonlocal SAR despeckling. *IEEE Geosci. Remote Sensing Lett.*, 11(2):524–528.

[35] Csiszár, I. (1991). Why least squares and maximum entropy? an axiomatic approach to inference for linear inverse problems. *The annals of statistics*, 19(4):2032–2066.

[36] Dabov, K., Foi, A., Katkovnik, V., and Egiazarian, K. (2007). Image denoising by sparse 3-d transform-domain collaborative filtering. *Image Processing, IEEE Transactions on*, 16(8):2080–2095.

[37] Deledalle, C.-A., Denis, L., and Tupin, F. (2009). Iterative weighted maximum likelihood denoising with probabilistic patch-based weights. *IEEE Transactions on Image Processing*, 18(12):2661–2672.

[38] Didas, S., Weickert, J., and Burgeth, B. (2005). Stability and local feature enhancement of higher order nonlinear diffusion filtering. In *Pattern Recognition*, pages 451–458. Springer.

[39] Didas, S., Weickert, J., and Burgeth, B. (2009). Properties of higher order nonlinear diffusion filtering. *JMIV*, 35(3):208–226.

[40] Domke, J. (2012). Generic methods for optimization-based modeling. *Journal of Machine Learning Research - Proceedings Track*, 22:318–326.

[41] Donoho, D. L., Elad, M., and Temlyakov, V. N. (2006). Stable recovery of sparse overcomplete representations in the presence of noise. *IEEE Transactions on Information Theory*, 52(1):6–18.

[42] Elad, M. and Aharon, M. (2006). Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, 15(12):3736–3745.

[43] Elad, M., Milanfar, P., and Rubinstein, R. (2007). Analysis versus synthesis in signal priors. *Inverse Problems*, 23(3):947–968.

[44] Esclarín, J. and Alvarez, L. (1997). Image quantization using reaction-diffusion equations. *SIAP*, 57(1):153–175.

[45] Evans, L. C. and Gariepy, R. F. (1991). *Measure theory and fine properties of functions*, volume 5. CRC press.

[46] Feng, W., Lei, H., and Gao, Y. (2014a). Speckle reduction via higher order total variation approach. *Image Processing, IEEE Transactions on*, 23(4):1831–1843.

[47] Feng, W., Qiao, H., and Chen, Y. (2014b). Poisson noise reduction with higher-order MRF prior. *Preprint*.

[48] Feng, X. and Prohl, A. (2003). Analysis of total variation flow and its finite element approximations. *ESAIM: Mathematical Modelling and Numerical Analysis*, 37(03):533–556.

[49] Foi, A., Katkovnik, V., and Egiazarian, K. (2007). Pointwise shape-adaptive DCT for high-quality denoising and deblocking of grayscale and color images. *Image Processing, IEEE Transactions on*, 16(5):1395–1411.

[50] Gan, X., Liew, A. W.-C., and Yan, H. (2005). A smoothness constraint set based on local statistics of BDCT coefficients for image postprocessing. *Image and Vision Computing*, 23(8):731–737.

[51] Ganan, S. and McClure, D. (1985). Bayesian image analysis: an application to single photon emission tomography. In *American Statistical Association*, pages 12–18.

[52] Gao, Q. and Roth, S. (2012). How well do filter-based MRFs model natural images? In *DAGM/OAGM Symposium*, pages 62–72.

[53] Geman, S. and Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):721–741.

[54] Gilboa, G., Sochen, N., and Zeevi, Y. Y. (2002). Forward-and-backward diffusion processes for adaptive image enhancement and denoising. *IEEE TIP*, 11(7):689–703.

[55] Goodman, J. W. (1975). Statistical properties of laser speckle patterns. In *Laser speckle and related phenomena*, pages 9–75. Springer.

[56] Graves, A. and Schmidhuber, J. (2009). Offline handwriting recognition with multi-dimensional recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 545–552.

[57] Gu, S., Zhang, L., Zuo, W., and Feng, X. (2014). Weighted nuclear norm minimization with application to image denoising. In *CVPR*.

[58] Guidotti, P. and Longo, K. (2011). Two enhanced fourth order diffusion models for image denoising. *JMIV*, 40(2):188–198.

[59] Hajiaboli, M. R. (2011). An anisotropic fourth-order diffusion filter for image noise removal. *IJCV*, 92(2):177–191.

[60] Hawe, S., Kleinsteuber, M., and Diepold, K. (2013). Analysis operator learning and its application to image reconstruction. *IEEE Transactions on Image Processing*, 22(6):2138–2150.

[61] Hebert, T. and Leahy, R. (1989). A generalized em algorithm for 3-d bayesian reconstruction from poisson data using gibbs priors. *Medical Imaging, IEEE Transactions on*, 8(2):194–202.

[62] Heess, N., Williams, C. K. I., and Hinton, G. E. (2009). Learning generative texture models with extended fields-of-experts. In *BMVC*, pages 1–11.

[63] Hinton, G. E. (1999). Products of experts. In *ICANN*, pages 1–6.

[64] Hu, Y. H. and Hwang, J.-N. (2010). *Handbook of neural network signal processing*. CRC press.

[65] Huang, J. and Mumford, D. (1999). Statistics of natural images and models. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 1. IEEE.

[66] Huber, P. J. (1981). *Robust statistics*. Wiley, New York.

[67] Jain, V. and Seung, S. (2009). Natural image denoising with convolutional networks. In *NIPS*, pages 769–776.

[68] Jancsary, J., Nowozin, S., and Rother, C. (2012). Loss-specific training of non-parametric image restoration models: A new state of the art. In *ECCV*, pages 112–125.

[69] Jung, C., Jiao, L., Qi, H., and Sun, T. (2012). Image deblocking via sparse representation. *Signal Processing: Image Communication*, 27(6):663–677.

[70] Krajsek, K. and Scharr, H. (2010). Diffusion filtering without parameter tuning: Models and inference tools. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2536–2543. IEEE.

[71] Krishnan, D. and Fergus, R. (2009). Fast image deconvolution using hyper-laplacian priors. In *Advances in Neural Information Processing Systems*, pages 1033–1041.

[72] Kuan, D. T., Sawchuk, A., Strand, T. C., and Chavel, P. (1987). Adaptive restoration of images with speckle. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 35(3):373–383.

[73] Kunisch, K. and Pock, T. (2013). A bilevel optimization approach for parameter learning in variational models. *SIAM Journal on Imaging Sciences*, 6(2):938–983.

[74] Leclerc, Y. G. (1989). Constructing simple stable descriptions for image partitioning. *International journal of computer vision*, 3(1):73–102.

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

[76] Lee, H., Battle, A., Raina, R., and Ng, A. Y. (2006). Efficient sparse coding algorithms. In *NIPS*, pages 801–808.

[77] Lee, J.-S. (1980). Digital image enhancement and noise filtering by use of local statistics. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (2):165–168.

[78] Lefkimmiatis, S., Bourquard, A., and Unser, M. (2012). Hessian-based norm regularization for image restoration with biomedical applications. *Image Processing, IEEE Transactions on*, 21(3):983–995.

[79] Levin, A. and Nadler, B. (2011). Natural image denoising: Optimality and inherent bounds. In *CVPR*, pages 2833–2840. IEEE.

[80] Li, F., Shen, C., Fan, J., and Shen, C. (2007). Image restoration combining a total variational filter and a fourth-order filter. *Journal of Visual Communication and Image Representation*, 18(4):322–330.

[81] Li, G.-T., Wang, C.-L., Huang, P.-P., and Yu, W.-D. (2013). SAR image despeckling using a space-domain filter with alterable window. *IEEE Geosci. Remote Sensing Lett.*, 10(2):263–267.

[82] Liew, A.-C. and Yan, H. (2004). Blocking artifacts suppression in block-coded images using overcomplete wavelet representation. *IEEE Trans. Circuits Syst. Video Techn.*, 14(4):450–461.

[83] Liew, A.-C., Yan, H., and Law, N.-F. (2005). POCS-based blocking artifacts suppression using a smoothness constraint set with explicit region modeling. *IEEE Trans. Circuits Syst. Video Techn.*, 15(6):795–800.

[84] Lions, P. L. and Mercier, B. (1979). Splitting algorithms for the sum of two nonlinear operators. 16(6):964–979.

[85] List, P., Joch, A., Lainema, J., Bjontegaard, G., and Karczewicz, M. (2003). Adaptive deblocking filter. *IEEE transactions on circuits and systems for video technology*, 13(7):614–619.

[86] Liu, D. C. and Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503–528.

[87] Liu, R., Lin, Z., Zhang, W., and Su, Z. (2010). Learning PDEs for image restoration via optimal control. In *ECCV*, pages 115–128.

[88] Lysaker, M., Lundervold, A., and Tai, X.-C. (2003). Noise removal using fourth-order partial differential equation with applications to medical magnetic resonance images in space and time. *Image Processing, IEEE Transactions on*, 12(12):1579–1590.

[89] Lysaker, M. and Tai, X.-C. (2006). Iterative image restoration combining total variation minimization and a second-order functional. *International Journal of Computer Vision*, 66(1):5–18.

[90] Mairal, J., Bach, F., Ponce, J., and Sapiro, G. (2009a). Online dictionary learning for sparse coding. In *ICML*, pages 689–696.

[91] Mairal, J., Bach, F., Ponce, J., Sapiro, G., and Zisserman, A. (2009b). Non-local sparse models for image restoration. In *ICCV*, pages 2272–2279.

[92] Mairal, J., Elad, M., and Sapiro, G. (2008). Sparse representation for color image restoration. *IEEE Transactions on Image Processing*, 17(1):53–69.

[93] Milanfar, P. (2013). A tour of modern image filtering: new insights and methods, both practical and theoretical. *Signal Processing Magazine, IEEE*, 30(1):106–128.

[94] Nesterov, Y. and Nesterov, I. E. (2004). *Introductory lectures on convex optimization: A basic course*, volume 87. Springer.

[95] Niklas Nordström, K. (1990). Biased anisotropic diffusion: a unified regularization and diffusion approach to edge detection. *Image and Vision Computing*, 8(4):318–327.

[96] Nikolova, M., Ng, M. K., and Tam, C.-P. (2010). Fast nonconvex nonsmooth minimization methods for image restoration and reconstruction. *Image Processing, IEEE Transactions on*, 19(12):3073–3088.

[97] Ochs, P., Chen, Y., Brox, T., and Pock, T. (2014a). iPiano: Inertial proximal algorithm for nonconvex optimization. *SIAM J. Imaging Sciences*, 7(2):1388–1419.

[98] Ochs, P., Dosovitskiy, A., Brox, T., and Pock, T. (2013). An iterated l1 algorithm for non-smooth non-convex optimization in computer vision. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1759–1766. IEEE.

[99] Ochs, P., Dosovitskiy, A., Brox, T., and Pock, T. (2014b). An iteratively reweighted algorithm for non-smooth non-convex optimization in computer vision. Technical Report.

[100] Oh, S., Woo, H., Yun, S., and Kang, M. (2013). Non-convex hybrid total variation for image denoising. *Journal of Visual Communication and Image Representation*, 24(3):332–344.

[101] Ophir, B., Elad, M., Bertin, N., and Plumbley, M. D. (2011). Sequential minimal eigenvalues – an approach to analysis dictionary learning. In *European Signal Processing Conference*, pages 1465–1469.

[102] Papafitsoros, K. and Schönlieb, C.-B. (2014). A combined first and second order variational approach for image reconstruction. *Journal of mathematical imaging and vision*, 48(2):308–338.

[103] Parrilli, S., Poderico, M., Angelino, C. V., and Verdoliva, L. (2012). A nonlocal SAR image denoising algorithm based on LLMMSE wavelet shrinkage. *IEEE T. Geoscience and Remote Sensing*, 50(2):606–616.

[104] Perona, P. and Malik, J. (1990). Scale-space and edge detection using anisotropic diffusion. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(7):629–639.

[105] Peyré, G. and Fadili, J. (2011). Learning analysis sparsity priors. In *Proc. of Sampta'11*.

[106] Plonka, G. and Ma, J. (2008). Nonlinear regularized reaction-diffusion filters for denoising of images with textures. *IEEE TIP*, 17(8):1283–1294.

[107] Pock, T. and Chambolle, A. (2011). Diagonal preconditioning for first order primal-dual algorithms. In *ICCV*.

[108] Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17.

[109] Reeve, H. C. and Lim, J. S. (1983). Reduction of blocking effect in image coding. In *ICASSP*, pages 1212–1215.

[110] Roth, S. and Black, M. J. (2005). Fields of experts: A framework for learning image priors. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 860–867. IEEE.

[111] Roth, S. and Black, M. J. (2009). Fields of experts. *International Journal of Computer Vision*, 82(2):205–229.

[112] Rubinstein, R., Faktor, T., and Elad, M. (2012). K-SVD dictionary-learning for the analysis sparse model. In *ICASSP*.

[113] Rubinstein, R., Peleg, T., and Elad, M. (2013). Analysis K-SVD: A dictionary-learning algorithm for the analysis sparse model. *IEEE Transactions on Signal Processing*, 61(3):661–677.

[114] Rudin, L. I., Osher, S., and Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259–268.

[115] Samuel, K. G. G. and Tappen, M. (2009). Learning optimized map estimates in continuously-valued MRF models. In *CVPR*.

[116] Saquib, S. S., Bouman, C. A., and Sauer, K. (1998). Ml parameter estimation for markov random fields with applications to bayesian tomography. *Image Processing, IEEE Transactions on*, 7(7):1029–1044.

[117] Scharr, H., Black, M. J., and Haussecker, H. W. (2003). Image statistics and anisotropic diffusion. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 840–847. IEEE.

[118] Scherzer, O. and Weickert, J. (2000). Relations between regularization and diffusion filtering. *Journal of Mathematical Imaging and Vision*, 12(1):43–63.

[119] Schmidt, U., Gao, Q., and Roth, S. (2010). A generative perspective on MRFs in low-level vision. In *CVPR*, pages 1751–1758.

[120] Schmidt, U. and Roth, S. (2014). Shrinkage fields for effective image restoration. In *CVPR*.

[121] Steidl, G. and Teuber, T. (2010). Removing multiplicative noise by douglas-rachford splitting methods. *Journal of Mathematical Imaging and Vision*, 36(2):168–184.

[122] Strong, D. M. and Chan, T. F. (1996). *Relation of regularization parameter and scale in total variation based image denoising*. Department of Mathematics, University of California, Los Angeles.

[123] Sun, D. and kuen Cham, W. (2007). Postprocessing of low bit-rate block DCT coded images based on a fields of experts prior. *IEEE Transactions on Image Processing*, 16(11):2743–2751.

[124] Surya Prasath, V. and Vorotnikov, D. (2014). Weighted and well-balanced anisotropic diffusion scheme for image denoising and restoration. *Nonlinear Analysis: Real World Applications*, 17:33–46.

[125] Teh, Y. W., Welling, M., Osindero, S., and Hinton, G. E. (2003). Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research*, 4:1235–1260.

[126] Tikhonov, A. N. (1943). On the stability of inverse problems. In *Dokl. Akad. Nauk SSSR*, volume 39, pages 195–198.

[127] Tikhonov, A. N. and Arsenin, V. Y. (1977). Solutions of ill-posed problems.

[128] Tsiotsios, C. and Petrou, M. (2013). On the choice of the parameters for anisotropic diffusion in image processing. *Pattern Recognition*, 46(5):1369–1381.

[129] Unger, M., Pock, T., Werlberger, M., and Bischof, H. (2010). A convex approach for variational super-resolution. In *Proceedings German Association for Pattern Recognition (DAGM)*, volume 6376 of *LNCS*, pages 313–322. Springer.

[130] Wallace, G. K. (1991). The JPEG still picture compression standard. *Communications of the ACM*, 34(4):30–44.

[131] Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612.

[132] Weickert, J. (1998). *Anisotropic diffusion in image processing*. Teubner Stuttgart.

[133] Weickert, J. (2001). Applications of nonlinear diffusion in image processing and computer vision. *Acta Math. Univ. Comenianae*, 70(1):33–50.

[134] Welk, M., Gilboa, G., and Weickert, J. (2009). Theoretical foundations for discrete forward-and-backward diffusion filtering. In *Scale Space and Variational Methods in Computer Vision*, pages 527–538. Springer.

[135] Welling, M., Hinton, G. E., and Osindero, S. (2002). Learning sparse topographic representations with products of student-t distributions. In *NIPS*, pages 1359–1366.

[136] Xiong, Z., Orchard, M. T., and Zhang, Y.-Q. (1997). A deblocking algorithm for JPEG compressed images using overcomplete wavelet representations. *IEEE Trans. Circuits Syst. Video Techn.*, 7(2):433–437.

[137] Yaghoobi, M., Nam, S., Gribonval, R., and Davies, M. E. (2011). Analysis operator learning for over-complete co-sparse representations. In *European Signal Processing Conference*.

[138] Yaghoobi, M., Nam, S., Gribonval, R., and Davies, M. E. (2012). Noise aware analysis operator learning for approximately cosparse signals. In *ICASSP*.

[139] You, Y.-L. and Kaveh, M. (2000). Fourth-order partial differential equations for noise removal. *Image Processing, IEEE Transactions on*, 9(10):1723–1730.

[140] Yun, S. and Woo, H. (2012). A new multiplicative denoising variational model based on mth root transformation. *IEEE Transactions on Image Processing*, 21(5):2523–2533.

[141] Zavriev, S. and Kostyuk, F. (1993). Heavy-ball method in nonconvex optimization problems. *Computational Mathematics and Modeling*, 4(4):336–341.

[142] Zhu, S. C. and Mumford, D. (1997). Prior learning and gibbs reaction-diffusion. *IEEE TPAMI*, 19(11):1236–1250.

[143] Zoran, D. and Weiss, Y. (2011). From learning models of natural image patches to whole image restoration. In *ICCV*, pages 479–486.