Shrinkage Fields for Effective Image Restoration

Uwe Schmidt Stefan Roth
Department of Computer Science, TU Darmstadt

Abstract

Many state-of-the-art image restoration approaches do not scale well to larger images, such as megapixel images common in the consumer segment. Computationally expensive optimization is often the culprit. While efficient alternatives exist, they have not reached the same level of image quality. The goal of this paper is to develop an effective approach to image restoration that offers both computational efficiency and high restoration quality. To that end we propose shrinkage fields*, a random field-based architecture that combines the image model and the optimization algorithm in a single unit. The underlying shrinkage operation bears connections to wavelet approaches, but is used here in a random field context. Computational efficiency is achieved by construction through the use of convolution and DFT as the core components; high restoration quality is attained through loss-based training of all model parameters and the use of a cascade architecture. Unlike heavily engineered solutions, our learning approach can be adapted easily to different trade-offs between efficiency and image quality. We demonstrate state-of-the-art restoration results with high levels of computational efficiency, and significant speedup potential through inherent parallelism.*

1. Introduction

Image restoration methods for removing imaging artifacts, such as noise, blur, moiré *etc*. have received significant attention in both academic research, as well as in practical applications of digital imaging [*e.g*., 6]. In academic research, the focus has been predominantly on achieving utmost image quality, largely disregarding the computational effort of the restoration process [18, 21, 31]. In practical digital imaging, the computational resources are often severely constrained, however, since the processing capacity of on-camera hardware is many times lower than that of a conventional desktop PC. But even on a desktop PC state-of-the-art techniques often take minutes to denoise a small VGA-sized image (equivalent to 0.3 megapixels). Modern digital cameras take images of 16 and more megapixels, on the other hand, to which existing techniques by and large do not scale. The main notable exception is BM3D [5], which offers high efficiency and image quality, but is a heavily engineered method with years of refinement. Moreover, its use of block matching as the key computational component makes an implementation on parallel architectures, such as GPUs and DSPs, challenging. One may hope that advances in embedded hardware will make the direct on-camera usage of existing advanced restoration techniques possible in the future, but it is not unlikely that the image resolution will increase as well. Consequently, to bridge the existing gap in computational efficiency of image restoration techniques and at the same time achieve high image quality, a different image restoration approach is needed.

In this paper we introduce *shrinkage fields*, a principled image restoration architecture that is derived from existing optimization algorithms for common random field models. In particular, shrinkage fields owe their computational efficiency to a specific kind of quadratic relaxation technique that is derived from the so-called additive form of half-quadratic optimization approaches [11] – the only operations not applied at a per-pixel level are convolutions and discrete Fourier transforms (DFTs). But unlike existing additive half-quadratic approaches [15, 28], we make full use of learning through loss-based training with application-specific loss functions [*cf*. 13], which allows us to achieve higher levels of restoration quality. Moreover and in contrast to standard random fields, which are specified through potential functions, shrinkage fields model the "shrinkage functions" associated with the potential directly. This increases the flexibility over half-quadratic approaches of the additive form, since we can show that potential functions always lead to monotonic shrinkage functions. In contrast, we can – and do – learn non-monotonic shrinkage functions, similar to those that have been discriminatively learned in the context of wavelet image denoising [12]. More importantly, using shrinkage functions directly admits efficient learning, because the model prediction and its gradient w.r.t. the model parameters can be computed in closed form. Finally, our approach employs a prediction cascade [25], using multiple model stages for iterative refinement. Loosely speaking, we learn the random field model and the iterative optimization algorithm at the same time [*cf*. 2].

To appear in the Proc. of the *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Columbus, OH, June 2014.

The proposed approach has several key benefits: *(1)* It is conceptually simple and derived from standard inference procedures for random field models; *(2)* it achieves very high levels of image quality on par with, or surpassing, the current state of the art; *(3)* it is computationally very efficient with a complexity of $\mathcal{O}(D \log D)$ (where D is the number of pixels); *(4)* it offers high levels of parallelism making it well suited for GPU or DSP implementations; *(5)* unlike heavily engineered techniques, such as BM3D, all parameters can be directly learned from example data using simple gradient-based optimization, making it easy to apply and adapt to new settings, such as different trade-offs between efficiency and restoration quality.

1.1. Other related work

The connection between regularization or priors and shrinkage functions has been widely studied in wavelet image restoration [*e.g.*, 1, 26]. A connection between the additive form of half-quadratic optimization and shrinkage functions has been noted by Wang *et al*. [28]. Based on the approach of [28], Krishnan and Fergus [15] popularized additive half-quadratic optimization for the task of non-blind deconvolution [*e.g.*, 30]. We start from this connection here, but in contrast do not use fixed potential functions, but employ more general, learned shrinkage functions.

Discriminative training of continuous conditional random fields (CRFs) for image restoration has been proposed by Samuel and Tappen [22], which has recently been revisited by Chen *et al*. [3]. Gaussian CRFs and their associated loss-based training have first been introduced by Tappen *et al*. [27]. Recently, Jancsary *et al*. [13] improved upon this by introducing regression tree fields (RTFs), a more flexible Gaussian CRF that is also trained by loss minimization. In contrast to these previous approaches, the proposed shrinkage fields admit more efficient inference and can be trained very easily by means of standard gradient-based optimization. Discriminatively learning a random field model and its associated optimization algorithm has been proposed by Barbu [2]. In the same vein, Schmidt *et al*. [25] recently trained a cascade of RTFs. While [2] is very efficient, it yields lower image quality given the same model complexity, and relies on a complicated and time-consuming learning procedure. Our work is conceptually most similar to [25], which is also generally motivated by half-quadratic inference. However, here we additionally derive the model parameterization (shrinkage functions for filter responses) specifically from the additive half-quadratic form. By doing so, we trade-off modeling flexibility (compared to [25]) against far more efficient inference and ease of training.

2. Background: Half-Quadratic Optimization

As a starting point we consider restoring an image $\mathbf{x}$ from its corrupted observation $\mathbf{y}$ by combining an observation likelihood and an image prior invoking Bayes' rule:

$$p(\mathbf{x}|\mathbf{y}) \propto p(\mathbf{y}|\mathbf{x}) \cdot p(\mathbf{x}) \tag{1}$$

$$\propto \mathcal{N}(\mathbf{y}; \mathbf{K}\mathbf{x}, \mathbf{I}/\lambda) \cdot \prod_{i=1}^{N} \prod_{c \in \mathcal{C}} \exp\left(-\rho_i(\mathbf{f}_i^\mathsf{T} \mathbf{x}_{(c)})\right). \tag{2}$$

Here the corruption process is modeled with a Gaussian likelihood (or data term), where $\mathbf{K}\mathbf{x} \equiv \mathbf{k} \otimes \mathbf{x}$ denotes convolution of $\mathbf{x}$ with a kernel (point spread function) $\mathbf{k}$, and λ is related to the strength of the assumed additive Gaussian noise. Regularization is provided through a Markov random field model (fields of experts [21]) with robust potential functions ρ_i that model the responses $\mathbf{f}_i^\mathsf{T}\mathbf{x}_{(c)}$ of filters $\mathbf{f}_i$ over all cliques $c \in \mathcal{C}$ of the image $\mathbf{x}$.

The posterior distribution $p(\mathbf{x}|\mathbf{y}) \propto \exp\left(-E(\mathbf{x}|\mathbf{y})\right)$ can be expressed by its associated Gibbs energy

$$E(\mathbf{x}|\mathbf{y}) = \frac{\lambda}{2}\|\mathbf{y} - \mathbf{K}\mathbf{x}\|^2 + \sum_{i=1}^{N} \sum_{c \in \mathcal{C}} \rho_i(\mathbf{f}_i^\mathsf{T} \mathbf{x}_{(c)}), \tag{3}$$

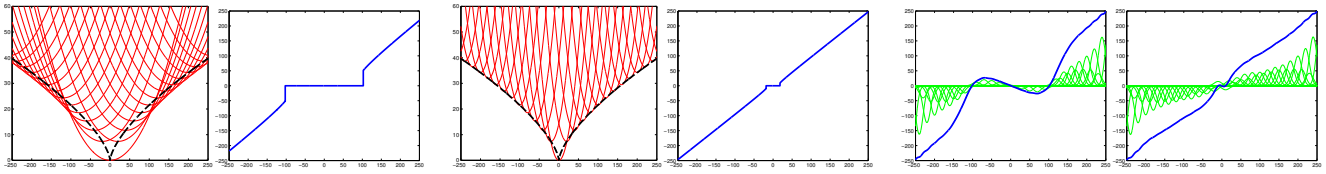
which allows to predict the restored image by finding the energy minimum $\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} E(\mathbf{x}|\mathbf{y}) = \arg\max_{\mathbf{x}} p(\mathbf{x}|\mathbf{y})$.

One way to minimize Eq. (3) is to directly employ gradient-descent algorithms. Another popular approach [10, 11, 28], one that we analyze and extend in this paper, is to first introduce independent auxiliary variables z_{ic} for all filter responses $\mathbf{f}_i^\mathsf{T}\mathbf{x}_{(c)}$ to obtain an augmented energy function $E(\mathbf{x}, \mathbf{z}|\mathbf{y})$ in such a way that $\arg\min_{\mathbf{x}} E(\mathbf{x}|\mathbf{y}) = \arg\min_{\mathbf{x},\mathbf{z}} E(\mathbf{x}, \mathbf{z}|\mathbf{y})$. A block coordinate descent strategy, which alternates between minimizing w.r.t. $\mathbf{x}$ and $\mathbf{z}$, is then used to minimize $E(\mathbf{x}, \mathbf{z}|\mathbf{y})$. This approach typically has faster convergence than minimizing $E(\mathbf{x}|\mathbf{y})$ directly, and each descent step is often relatively simple to carry out.

Specifically, auxiliary variables are introduced in such a way that $E(\mathbf{x}|\mathbf{z}, \mathbf{y})$[1] becomes a quadratic function; minimizing $E(\mathbf{z}|\mathbf{x}, \mathbf{y})$ simply amounts to solving many independent univariate optimization problems. In computer vision, this approach was first proposed by Geman and colleagues [10, 11] under the name "half-quadratic" regularization. In other words, each iteration of the algorithm uses a different quadratic relaxation $E(\mathbf{x}|\mathbf{z}, \mathbf{y})$ of the original objective function $E(\mathbf{x}|\mathbf{y})$, determined by auxiliary variables $\mathbf{z}$.

Half-quadratic approaches can be further categorized into *additive* [11] and *multiplicative* [10] forms. Without going into details, a main computational difference in practice is that $\arg\min_{\mathbf{x}} E(\mathbf{x}|\mathbf{z}, \mathbf{y}) = \boldsymbol{\Omega}(\mathbf{z}, \mathbf{y})^{-1}\boldsymbol{\eta}(\mathbf{y})$ in the multiplicative form, and $\arg\min_{\mathbf{x}} E(\mathbf{x}|\mathbf{z}, \mathbf{y}) = \boldsymbol{\Omega}(\mathbf{y})^{-1}\boldsymbol{\eta}(\mathbf{z}, \mathbf{y})$ in the additive form. Here, $\boldsymbol{\Omega} \in \mathbb{R}^{D \times D}$ is a sparse matrix with D being the number of pixels, and $\boldsymbol{\eta} \in \mathbb{R}^D$ is a vector. That implies that the quadratic function can be minimized by solving a linear equation system, where in the multiplicative form, $\mathbf{z}$ only influences the equation system matrix $\boldsymbol{\Omega}$, and in the additive form only the

[1] $p(\mathbf{x}|\mathbf{z}, \mathbf{y}) \propto \exp\left(-E(\mathbf{x}|\mathbf{z}, \mathbf{y})\right)$, other energies defined accordingly.



(a) $\rho(v), \beta = 0.0035$ (b) $f_\beta(v), \beta = 0.0035$ (c) $\rho(v), \beta = 0.035$ (d) $f_\beta(v), \beta = 0.035$ (e) Stage 1 of $\text{CSF}_{\text{pw.}}$ (f) Stage 2 of $\text{CSF}_{\text{pw.}}$

Figure 1. *(a,c)* Potential $\rho(v) = |v|^{2/3}$ (dashed, black) and its quadratic relaxation $\rho(z) + \frac{\beta}{2}(v-z)^2$ for some values of z (solid, red). *(b,d)* Associated shrinkage function $f_\beta(v) = \arg\min_z \left(\rho(z) + \frac{\beta}{2}(v-z)^2\right)$ for $\rho(z) = |z|^{2/3}$ and given β. *(e,f)* Learned shrinkage functions $f_{\boldsymbol{\pi}}(v) = \sum_{j=1}^{M} \pi_j \exp\left(-\frac{\gamma}{2}(v-\mu_j)^2\right)$ (solid, blue) of $\text{CSF}_{\text{pw.}}$ (*cf*. Sec. 4) as linear combination of Gaussian RBF kernels (solid, green).

right-hand side $\boldsymbol{\eta}$ of the equation system. Hence, the additive form is in general computationally more attractive since the equation system matrix stays constant during iterative optimization (*e.g.*, a factorization of $\boldsymbol{\Omega}$ could be re-used, or $\boldsymbol{\Omega}$ might be diagonalized with a change of basis).

However, a challenge is that the additive form is not directly applicable to many heavy-tailed potential functions ρ of practical relevance. To remedy this and to speed up convergence, Wang *et al*. [28] proposed a continuation scheme, where a parameter β is increased during the half-quadratic optimization (*cf*. Alg. 1). Concretely, the problem is cast as $\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} E(\mathbf{x}|\mathbf{y}) = \arg\min_{\substack{\mathbf{x},\mathbf{z} \\ \beta\to\infty}} E_\beta(\mathbf{x},\mathbf{z}|\mathbf{y})$ with

$$E_\beta(\mathbf{x},\mathbf{z}|\mathbf{y}) = \frac{\lambda}{2}\|\mathbf{y}-\mathbf{K}\mathbf{x}\|^2 + \sum_{i=1}^{N}\sum_{c\in\mathcal{C}}\left(\frac{\beta}{2}\left(\mathbf{f}_i^\mathsf{T}\mathbf{x}_{(c)} - z_{ic}\right)^2 + \rho_i(z_{ic})\right). \quad (4)$$

Intuitively, when $\beta \to \infty$, the auxiliary variables $z_{ic} \to \mathbf{f}_i^\mathsf{T}\mathbf{x}_{(c)}$ approach their corresponding filter responses, and Eq. (4) converges to the original Eq. (3). This approach has been popularized for non-blind image deconvolution by Krishnan and Fergus [15] in recent years.

To see why this approach is so appealing, it is instructive to take a closer look at the alternating optimization procedure, which is summarized in Alg. 1. Specifically, the two update steps are computationally very inexpensive when – what we assume from now on – 2D convolution is carried out with circular (periodic) boundary conditions. Then we can write the two algorithmic steps as:

$$f_{i,\beta}(v) = \arg\min_z \left(\rho_i(z) + \frac{\beta}{2}(v-z)^2\right) \quad (5)$$

$$g_\beta(\mathbf{z}) = \left[\frac{\lambda}{\beta}\mathbf{K}^\mathsf{T}\mathbf{K} + \sum_{i=1}^{N}\mathbf{F}_i^\mathsf{T}\mathbf{F}_i\right]^{-1}\left[\frac{\lambda}{\beta}\mathbf{K}^\mathsf{T}\mathbf{y} + \sum_{i=1}^{N}\mathbf{F}_i^\mathsf{T}\mathbf{z}_i\right] = \mathcal{F}^{-1}\left[\frac{\mathcal{F}\left(\frac{\lambda}{\beta}\mathbf{K}^\mathsf{T}\mathbf{y} + \sum_{i=1}^{N}\mathbf{F}_i^\mathsf{T}\mathbf{z}_i\right)}{\frac{\lambda}{\beta}\check{\mathbf{K}}^* \circ \check{\mathbf{K}} + \sum_{i=1}^{N}\check{\mathbf{F}}_i^* \circ \check{\mathbf{F}}_i}\right]. \quad (6)$$

$\mathbf{F}\mathbf{x} = [\mathbf{f}^\mathsf{T}\mathbf{x}_{(\mathcal{C}_1)}, \ldots, \mathbf{f}^\mathsf{T}\mathbf{x}_{(\mathcal{C}_{|\mathcal{C}|})}]^\mathsf{T} \equiv \mathbf{f} \otimes \mathbf{x}$ denotes 2D convolution with filter $\mathbf{f}$. The optical transfer function $\check{\mathbf{F}} \equiv \mathcal{F}(\mathbf{f})$ is derived from filter (point spread function) $\mathbf{f}$, where $\mathcal{F}$ denotes the discrete Fourier transform (DFT). Note that division and multiplication ($\circ$) are applied element-wise in Eq. (6); $\check{\mathbf{F}}^*$ denotes the complex conjugate of $\check{\mathbf{F}}$.

Algorithm 1 Half-quadr. minimization with continuation

Require: β-schedule $\beta_1, \ldots, \beta_T$ with $\beta_{t+1} > \beta_t$
  $\hat{\mathbf{x}}_0 \leftarrow \mathbf{y}$
  for $t \leftarrow 1$ **to** T **do**
    $\hat{z}_{ic} \leftarrow \arg\min_{z_{ic}} E_{\beta_t}(\mathbf{z}|\hat{\mathbf{x}}_{t-1}, \mathbf{y}) = f_{i,\beta_t}(\mathbf{f}_i^\mathsf{T}\hat{\mathbf{x}}_{t-1(c)})$
    $\hat{\mathbf{x}}_t \leftarrow \arg\min_{\mathbf{x}} E_{\beta_t}(\mathbf{x}|\hat{\mathbf{z}}, \mathbf{y}) = g_{\beta_t}(\hat{\mathbf{z}})$

Eq. (5) is very cheap to compute because $f_{i,\beta}(v)$ is a univariate function that can be precomputed for all possible values of v and then stored in a lookup-table for fast retrieval [15]. Crucially, only the additive half-quadratic form allows updating the image $\mathbf{x}$ via Eq. (6) very quickly in closed form, because all convolution matrices (and thus the whole equation system matrix) can be diagonalized by DFTs, which means that solving the linear equation system amounts to element-wise division in the transformed domain followed by an inverse DFT to retain the solution in the spatial domain [*e.g.*, 15, 28]. Note that this only takes $N+1$ convolutions[2] and $N+3$ DFTs with an overall complexity of $\mathcal{O}(D \log D)$, where D is the number of pixels.

2.1. Shrinkage function

The role of $f_{i,\beta}$ (Eq. 5) is known as a *shrinkage* (or *mapping*) function in the wavelet image restoration literature [*cf*. 12]. Intuitively, its purpose is to shrink small filter/wavelet coefficients, *i.e*. pull them towards zero, because they are assumed to be caused by noise instead of signal.

For now, the shape of the shrinkage function is determined solely by β and its associated potential function ρ_i (Eq. (5), see Fig. 1(a–d) for an illustration). However, we make the observation that all $f_{i,\beta}$ according to Eq. (5) are monotonically increasing functions, regardless of the potential ρ_i. In order to prove this Proposition 1, it is useful to have the following Lemma (proved in the *suppl. material*):

[2] Each convolution can be expressed through DFTs, but typically is computationally more expensive for the small filters $\mathbf{f}_i$ used in practice.

Lemma 1. *For any function* $f : \mathbb{R} \to \mathbb{R}$ *and all* $\epsilon \geq 0$, $\arg\min_z f(z) \leq \arg\min_z \left(f(z) - \epsilon z\right)$.

Proposition 1. *For all* $\epsilon, \beta \geq 0, v \in \mathbb{R}$ *and any* $\rho(z)$*, the shrinkage function* $f_\beta(v) = \arg\min_z \left(\rho(z) + \frac{\beta}{2}(v-z)^2\right)$ *is monotonically increasing,* i.e. $f_\beta(v) \leq f_\beta(v+\epsilon)$.

Proof.

$$f_\beta(v+\epsilon) = \arg\min_z \left(\rho(z) + \frac{\beta}{2}\left(v+\epsilon-z\right)^2\right) \tag{7}$$

$$= \arg\min_z \left(\rho(z) + \frac{\beta}{2}(v-z)^2 - \epsilon\beta z\right) \tag{8}$$

It follows from Lemma 1 that $f_\beta(v) \leq f_\beta(v+\epsilon)$. □

To the best of our knowledge this has not been observed before. More importantly, it implies that one can gain additional flexibility in additive half-quadratic optimization by directly modeling the shrinkage function instead of the potential function.

3. Cascade of Shrinkage Fields

As we just motivated and will further justify below, directly modeling the shrinkage function is appealing. To that end, we remove the potential function and the associated optimization problem in Eq. (5) altogether, and replace $f_{i,\beta}$ with a flexible shrinkage function modeled as a linear combination of Gaussian RBF kernels:

$$f_{\boldsymbol{\pi}_i}(v) = \sum_{j=1}^{M} \pi_{ij} \exp\left(-\frac{\gamma}{2}(v-\mu_j)^2\right). \tag{9}$$

We assume shared precision γ and place the kernels at fixed, equidistant positions μ_j. We use up to $M = 53$ Gaussian kernels and make no further assumptions about the shape of the function (two examples are shown in Fig. 1(e–f)).

Shrinkage functions are widely studied in the wavelet restoration literature. However, instead of manually choosing shrinkage functions, we learn them from data[3] through setting the weights π_{ij} of the parametric form of Eq. (9). This is in clear contrast to previous work. Attempts at discriminatively learning shrinkage functions for wavelet restoration exist [*e.g.*, 12], but are not common. Furthermore, wavelet image restoration is quite different because the pixels of the restored image are not connected via a random field, as here.

We are not aware of any previous work that has used learning in the context of this particular form of half-quadratic optimization. Consequently, the full potential of this fast optimization approach has not been unlocked, because model parameters have always been chosen by hand.

[3] A possibly more suitable name would be *mapping* instead of *shrinkage* function, since our learned functions do not necessarily shrink the associated filter responses. We keep the widely known name despite this.

Algorithm 2 Inference with a cascade of shrinkage fields

$\hat{\mathbf{x}}_0 \leftarrow \mathbf{y}$
for $t \leftarrow 1$ **to** T **do**
  $\hat{\mathbf{x}}_t \leftarrow g_{\boldsymbol{\Theta}_t}(\hat{\mathbf{x}}_{t-1})$

Furthermore, the β-continuation schedule for the number of iterations of Alg. 1 is typically manually chosen.

In the following, we show how to overcome all of these limitations while retaining the computational benefits of this approach. To that end, we learn all model parameters (other than the size and number of filters, and the number of optimization iterations) from training data.

The most important benefit of directly modeling the shrinkage functions is that it allows us to reduce the optimization procedure to a single quadratic minimization in each iteration, which we denote as the prediction of a *shrinkage field* (SF):

$$g_{\boldsymbol{\Theta}}(\mathbf{x}) = \mathcal{F}^{-1}\left[\frac{\mathcal{F}\left(\lambda\mathbf{K}^\mathsf{T}\mathbf{y} + \sum_{i=1}^{N}\mathbf{F}_i^\mathsf{T} f_{\boldsymbol{\pi}_i}(\mathbf{F}_i\mathbf{x})\right)}{\lambda\check{\mathbf{K}}^* \circ \check{\mathbf{K}} + \sum_{i=1}^{N}\check{\mathbf{F}}_i^* \circ \check{\mathbf{F}}_i}\right] \tag{10}$$

$$= \boldsymbol{\Omega}^{-1}\boldsymbol{\eta}. \tag{11}$$

A shrinkage field $\mathcal{N}\left(\boldsymbol{\Omega}^{-1}\boldsymbol{\eta}, \boldsymbol{\Omega}^{-1}\right)$ is thus a particular Gaussian conditional random field, whose moments $\boldsymbol{\eta}$ and $\boldsymbol{\Omega}$ are determined through learned model parameters $\boldsymbol{\Theta}$, the observed image $\mathbf{y}$, and the point spread function $\mathbf{k}$. A key benefit is that the shrinkage field prediction $g_{\boldsymbol{\Theta}}(\mathbf{x})$ and its gradient $\frac{\partial g_{\boldsymbol{\Theta}}(\mathbf{x})}{\partial \boldsymbol{\Theta}}$ w.r.t. the model parameters $\boldsymbol{\Theta}$ can be computed in closed form, which allows for efficient parameter learning (Sec. 3.1). This is in contrast to more complicated learning procedures in other formulations, which need to solve nested minimization problems using bi-level optimization [3, 22]. Note that we completely eliminate the continuation parameter β, which is absorbed into the weights $\boldsymbol{\pi}_i$ of Eq. (9) and fused with λ (which will be learned) in Eq. (10).

Since half-quadratic optimization typically involves several (many) iterations of Eqs. (5) and (6), we can similarly chain multiple predictions into a *cascade of shrinkage fields* (CSF), as summarized in Alg. 2. A CSF is thus a cascade of Gaussian CRFs [*cf*. 25]. Note that the concept of a shrinkage function does not exist in previous CRF cascades. RTF cascades [25], for example, use regression trees to specify unary and pairwise factors; since the resulting equation system matrix cannot be diagonalized by DFTs, they *do not* admit fast closed-form inference as in Eq. (10).

3.1. Learning

We use loss-minimization to learn the model parameters $\boldsymbol{\Theta}_t = \{\lambda_t, \boldsymbol{\pi}_{ti}, \mathbf{f}_{ti}\}_{i=1}^N$ for every stage (iteration) t of Alg. 2. By learning different model parameters for every stage of our cascade, we essentially learn tailored random

field models for each iteration of the associated optimization algorithm[4]. For non-blind deconvolution, we follow [25] and parameterize the prediction with the blur kernel, such that the instance-specific blur ($\mathbf{K}$ in Eq. 10) is provided at test time; models are *not* trained for specific blurs.

To greedily learn the model stage-by-stage from $t = 1, \dots, T$, at stage t we minimize the cost function

$$J(\boldsymbol{\Theta}_t) = \sum_{s=1}^{S} \ell(\hat{\mathbf{x}}_t^{(s)}; \mathbf{x}_{\text{gt}}^{(s)}) \tag{12}$$

with training data $\{\mathbf{x}_{\text{gt}}^{(s)}, \mathbf{y}^{(s)}, \mathbf{k}^{(s)}\}_{s=1}^{S}$, where $\hat{\mathbf{x}}_t^{(s)}$ is obtained via Alg. 2. We can, in principle, employ any continuously differentiable loss function, and concretely choose the (negative) *peak signal-to-noise ratio* (PSNR)

$$\ell(\hat{\mathbf{x}}; \mathbf{x}_{\text{gt}}) = -20 \log_{10} \left(\frac{R\sqrt{D}}{\|\hat{\mathbf{x}} - \mathbf{x}_{\text{gt}}\|} \right), \tag{13}$$

where D denotes the number of pixels of $\hat{\mathbf{x}}$ and R the maximum intensity level of a pixel (*i.e.*, $R = 255$).

We minimize Eq. (12) with the gradient-based L-BFGS method (using an implementation by [23]). To that end, we, akin to [13], differentiate the loss of the predicted restored image $\hat{\mathbf{x}}_t$ (at stage t) w.r.t. model parameters $\boldsymbol{\Theta}_t$ as

$$\frac{\partial \ell(\hat{\mathbf{x}}_t; \mathbf{x}_{\text{gt}})}{\partial \boldsymbol{\Theta}_t} = \frac{\partial \ell(\hat{\mathbf{x}}_t; \mathbf{x}_{\text{gt}})}{\partial \hat{\mathbf{x}}_t} \cdot \frac{\partial \boldsymbol{\Omega}_t^{-1} \boldsymbol{\eta}_t}{\partial \boldsymbol{\Theta}_t} \tag{14}$$

$$= \hat{\mathbf{c}}_t^{\mathsf{T}} \left[\frac{\partial \boldsymbol{\eta}_t}{\partial \boldsymbol{\Theta}_t} - \frac{\partial \boldsymbol{\Omega}_t}{\partial \boldsymbol{\Theta}_t} \hat{\mathbf{x}}_t \right] \tag{15}$$

$$\text{with } \hat{\mathbf{c}}_t = \boldsymbol{\Omega}_t^{-1} \left[\frac{\partial \ell(\hat{\mathbf{x}}_t; \mathbf{x}_{\text{gt}})}{\partial \hat{\mathbf{x}}_t} \right]^{\mathsf{T}}.$$

Similar to $\hat{\mathbf{x}}_t$, we can efficiently compute $\hat{\mathbf{c}}_t$ by solving a linear equation system via element-wise division in the transformed domain. The derivatives for specific model parameters as well as further details, such as boundary handling due to periodic convolutions and parameter constraints, are omitted here for brevity and to make the equations more readable; however, all details can be found in the *supplemental material*.

In Eq. (12), each stage is trained greedily such that the loss is as small as possible after each stage, regardless of how many stages T are actually intended to be used in the cascade; this also applies to the cascade model of [25]. However, in contrast to the cascade of [25], which uses non-differentiable regression trees to determine the parameters of a Gaussian CRF and requires custom training, our shrinkage functions are smooth and differentiable. Hence, we do not need to alternate between gradient-based and combinatorial optimization (growing regression trees). Moreover,

[4]However, if we used the same filters at each model stage, we could re-use all optical transfer functions and save a lot of runtime after stage 1.

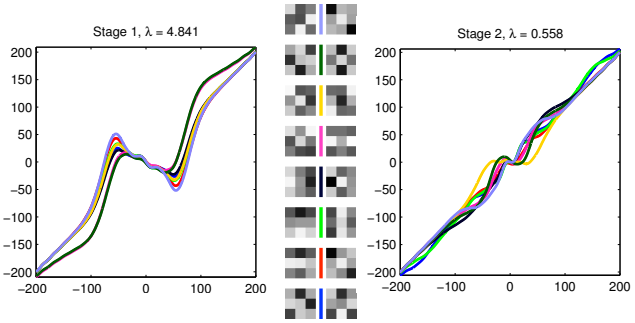


Figure 2. **First two stages of learned CSF$_{3\times3}$ model.** The shrinkage functions are color-matched with their corresponding filters.

we can use standard gradient-based methods to jointly train all T stages of the model by minimizing

$$J(\boldsymbol{\Theta}_{1,\dots,T}) = \sum_{s=1}^{S} \ell(\hat{\mathbf{x}}_T^{(s)}; \mathbf{x}_{\text{gt}}^{(s)}), \tag{16}$$

where only the loss of the final prediction $\hat{\mathbf{x}}_T$ is relevant. The derivatives w.r.t. model parameters of all stages can be computed efficiently and take the same basic form as Eq. (15), which allows for an easy implementation. Note that all stages can be learned jointly even while applying boundary operations, such as padding and truncation. All details and derivations are in the *supplemental material*.

4. Experiments

Training. Although the class of Gaussian CRFs that can be learned at one stage of our approach is restricted (compared to [13]), this limitation comes at the substantial benefit of fast prediction and learning. That means we can train our model on relatively large datasets – even with a simple Matlab implementation[5]. To generate the training data for our denoising experiments, we cropped a 256×256 pixel region from each of 400 images of the Berkeley segmentation dataset [19][6], *i.e.* our training set thus roughly contains 25 million pixels.

We have greedily trained 5 stages of four different configurations of our model with increasing capacity:

CSF$^5_{\text{pw.}}$ Pairwise model with fixed $\mathbf{f} = \{[1, -1], [1, -1]^{\mathsf{T}}\}$.
CSF$^5_{3\times3}$ Fully trained model with 8 filters of size 3×3.
CSF$^5_{5\times5}$ Fully trained model with 24 filters of size 5×5.
CSF$^5_{7\times7}$ Fully trained model with 48 filters of size 7×7.

Hence, CSF$^T_{m\times m}$ denotes a cascade of T stages with $m^2 - 1$ filters of size $m\times m$ (if $T < 5$, only T stages have been evaluated at test time; prediction can be stopped at any stage). Note that many more configurations are possible and will

[5]Code for learning and inference is available on the authors' webpages.
[6]These are strictly separate from all test images in Tabs. 1 and 2.

σ	KSVD [7]	FoE [9]	BM3D [5]	LSSC [18]	EPLL [31]	opt-MRF [3]	$\text{CSF}^5_{\text{pw.}}$	$\text{CSF}^4_{3\times3}$	$\text{CSF}^4_{5\times5}$	$\text{CSF}^5_{7\times7}$	ARF-4 [2]	RTF_5 [24]
15	30.87	30.99	31.08	**31.27**	31.19	31.18	29.99	30.78	31.12	**31.24**	30.70	—
25	28.28	28.40	28.56	**28.70**	28.68	28.66	27.47	28.29	28.58	**28.72**	28.20	**28.75**

Table 1. Average PSNR (dB) on 68 images from [21] for image denoising with $\sigma = 15, 25$; left part quoted from Chen *et al*. [3]. Training of our CSF models and denoising carried out *without* 8-bit quantization of noisy images to allow comparison with [2, 3].

lead to different performance vs. speed tradeoffs, which can be chosen to suit the particular application. Figs. 2 and 1(e–f) show the first two stages of the learned $\text{CSF}_{3\times3}$ and $\text{CSF}_{\text{pw.}}$ models, respectively, which are good examples of our observation that almost all learned shrinkage functions are *not* monotonically increasing, which means they could not have been obtained by learning a potential function (*cf*. Sec. 2).

Denoising. We first evaluated the task of image denoising (*i.e*., $\mathbf{k} = 1$), for which we trained our models to remove Gaussian noise with standard deviation $\sigma = 25$. The noisy training images were obtained by adding simulated Gaussian noise to the clean images. We subsequently quantized the intensity values of the noisy images to 8-bit to make the training data more realistic. In practice, noisy images are always integer-valued and range-limited, such as intensity values being in $\{0, \dots, 255\}$.

After training the models, we evaluate them on 68 (8-bit quantized noisy) test images originally introduced by [21], which have since become a reference set for image denoising; Fig. 4 shows a denoising example. We compare against a varied selection of recent state-of-the-art techniques. The results in Tab. 2 show that a (5-stage) cascade of regression tree fields (RTFs) [24] achieves the best performance (trained with the same data as our models). This is not surprising, since the more flexible RTFs do not make any noise assumption (in contrast to all other approaches in Tab. 2) and can thus effectively handle the additional quantization noise. Concerning the other methods, we outperform the strongest competitor BM3D by 0.22dB with our most powerful $\text{CSF}^5_{7\times7}$ model. Furthermore, our $\text{CSF}^4_{5\times5}$ model slightly outperforms BM3D and also has a faster runtime (*cf*. Fig. 3), even when only the CPU is used. Additionally, our model's inference procedure (convolutions and DFTs being the most expensive operations) is presumably much more amenable to GPU or DSP parallelization than the block-matching procedure of BM3D. It can also be observed that results of our models saturate after only 3–4 stages, hence "converge" very quickly.

We also compare against the recently introduced *opt-MRF* by Chen *et al*. [3] for two reasons: First, it currently is one of the best-performing CRFs for image restoration, achieved by using better optimization techniques with a model architecture originally proposed by [22]. Secondly, it uses a model configuration very similar to ours, that is 48 filters of size 7×7, which are fully learned from data (including associated potential functions). Moreover, we compare against the fast *active random field* (ARF) model of Barbu [2], which uses 24 filters of size 5×5. Since both of them were neither trained nor evaluated with 8-bit quantized noisy images, we use their setting to not give our model an unfair advantage. Hence, we additionally trained and evaluated our models without quantization. The results in Tab. 1 show[7] that we outperform [2, 3], and can also compete with the RTF-based cascade model [24] (trained with non-quantized images), whose additional flexibility does not seem pay off here since the image noise is truly Gaussian. The results further show that we can also compete for noise level $\sigma = 15$, for which we trained additional models.

Method	PSNR
BLS-GSM [20]	27.98
5×5 FoE [9]	28.22
LSSC [18]	28.23
BM3D [5]	28.31
RTF_5 [24]	**28.74**

St.	$\text{CSF}_{\text{pw.}}$	$\text{CSF}_{3\times3}$	$\text{CSF}_{5\times5}$	$\text{CSF}_{7\times7}$
1	26.60	27.54	27.46	27.70
2	27.26	27.93	28.26	28.38
3	27.31	28.02	28.34	28.45
4	27.36	28.05	28.37	28.52
5	27.36	28.08	28.39	**28.53**

Table 2. Average PSNR (dB) on 68 images from [21] for image denoising with $\sigma = 25$. On the right, each row shows the results from the respective stage of our models.

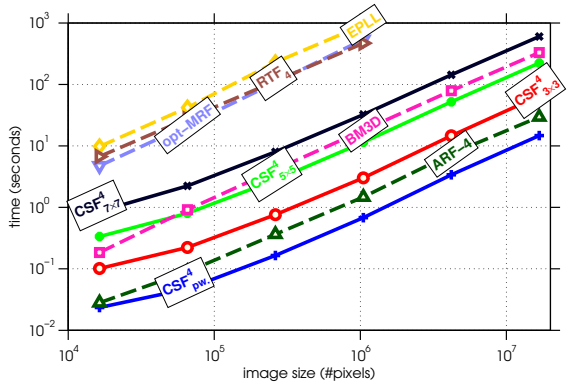
Runtime. The runtime comparison[8] for image denoising in Fig. 3 shows that our model scales to image sizes of more than 16 megapixels at reasonable runtimes (at most 10 minutes for our best model with a simple single-threaded Matlab implementation, and only 23 seconds on a GPU).

While a cascade of RTFs [24] is very flexible and yields state-of-the-art restoration results, its relatively complex and highly optimized C++ implementation hinges on multi-threading to boost runtime performance. Comparing single-threaded performance (Fig. 3), it is about an order of magnitude slower compared to our $\text{CSF}_{7\times7}$ (which exhibits competitive performance, *cf*. Tab. 1). We outperform BM3D at a faster runtime with our $\text{CSF}^4_{5\times5}$ model (*cf*. Tab. 2).

Additionally, our model's inference procedure is well suited for GPU or DSP parallelization. In order to gauge the potential speedup, we used the same code with Matlab's built-in GPU capabilities and were able to obtain significantly improved runtimes (Tab. 3). However, we should

[7] Comparing Tabs. 1 and 2 also shows how much results improve when they are obtained in a more artificial (non-quantized) setting.

[8] Matlab/C++ implementations from the respective authors, single-threading strictly enforced (incl. `-singleCompThread` for Matlab).



Method	128^2	256^2	512^2	1024^2	2048^2	4096^2
$\text{CSF}^4_{\text{pw.}}$	0.02	0.05	0.17 (0.03)	0.7 (0.05)	3.4 (0.18)	15 (0.8)
$\text{CSF}^4_{3\times3}$	0.10	0.22	0.76 (0.15)	3.0 (0.27)	14.6 (0.78)	65 (3.7)
$\text{CSF}^4_{5\times5}$	0.34	0.80	2.78 (0.44)	11.5 (0.80)	52.0 (2.42)	223 (10.8)
$\text{CSF}^4_{7\times7}$	0.86	2.23	8.00 (0.92)	32.3 (1.72)	143 (5.27)	603 (23.2)
ARF-4 [2]	0.03	0.09	0.37	1.5	7.5	29
BM3D [5]	0.18	0.92	4.09	18.0	78.9	330
opt-MRF [3]	4.73	21.7	108	538	–	–
RTF_4 [24]	6.71	27.7	113	469	–	–
EPLL [31]	9.76	41.9	229	930	–	–

Figure 3. **Runtime comparison for image denoising.** Single-threaded runtimes (in seconds) with an Intel Core i7-3930K CPU at 3.20GHz; small numbers in parentheses from simple Matlab-based GPU execution on a NVIDIA GeForce GTX 480. Runtimes of our models shown after 4 stages where performance saturates; using fewer stages takes proportionally less time, *e.g.* 2 stages take half the time. Note the logarithmic scales on both axes *(top)*. The table columns show runtimes for image sizes up to 4096×4096 pixels (about 16.78 megapixels).

expect additional speedups by using a more powerful recent GPU with an optimized implementation using CUDA or OpenCL ([3] quote a $40\times$ GPU speedup over presumably multi-threaded CPU code).

While the ARF model [2] (designed for real-time denoising) is more efficient (CPU only) than our CSF with the same number of stages, filters, and filter size, it exhibits inferior results: It performs 0.38dB worse than $\text{CSF}^4_{5\times5}$ (Tab. 1), and even our $\text{CSF}^4_{3\times3}$ model with only 8 3×3 filters surpasses the ARF in terms of restoration quality. While the ARF is twice as fast as $\text{CSF}^4_{3\times3}$, we can speed CSF up by re-using filters (*cf*. Sec. 3.1). Furthermore, our standard gradient-based learning procedure is much easier and faster, and enables learning more powerful models such as $\text{CSF}^5_{7\times7}$.

Computing the learning objective function $J(\boldsymbol{\Theta})$ (Eq. 12) and its gradient $\partial J(\boldsymbol{\Theta})/\partial\boldsymbol{\Theta}$ for $S = 400$ images of 256×256 pixels takes in total only around 7s ($\text{CSF}_{\text{pw.}}$), 24s ($\text{CSF}_{3\times3}$), 73s ($\text{CSF}_{5\times5}$), or 161s ($\text{CSF}_{7\times7}$) with our simple Matlab implementation (Intel Core i7-3930K hexa-core at 3.20GHz, six parallel threads). This allows us to thoroughly train our models by using 200 L-BFGS iterations. Another important property of our method is its predictable runtime, which is in contrast to methods (such as opt-MRF and RTF) that require iterative inference whose convergence depends on the input data. In our experience, runtime varies even more for deconvolution, mostly due to the blur kernel.

Blur kernel	[16]	[25]	$\text{CSF}^1_{\text{pw.}}$	$\text{CSF}^2_{\text{pw.}}$	$\text{CSF}^3_{\text{pw.}}$
Ground truth	32.73	**33.97**	32.48	33.50	33.48
Levin *et al.* [17]	30.05	**30.40**	29.63	30.34	**30.42**
Cho and Lee [4]	29.71	29.73	29.10	29.86	**29.99**
Fergus *et al.* [8]	28.38	**29.10**	28.36	29.02	29.01

Table 3. Average PSNR (dB) on 32 images from [17] for image deconvolution. Rows correspond to different blur kernel (estimates) provided by [17], while columns correspond to non-blind deconvolution methods. Left part of table quoted from [25], showing results from Levin *et al.* [16] and Schmidt *et al.* [25].

Joint training. While jointly training all stages of the model has the potential to yield superior results, we only partly confirm this in our denoising experiments. Since our learning objective function is not convex, the optimization often gets stuck in worse local optima than when using greedy training. Hence we tried first training each stage greedily (pre-training), and then "tuned" the model by starting joint training with the parameters obtained from pre-training. While this is guaranteed to not decrease (training set) performance, it does not always improve results considerably, especially with increasing model capacity. Jointly tuning all 5 stages of $\text{CSF}^5_{\text{pw.}}$ does pay off, by increasing PSNR performance about 0.31dB from 27.36dB to 27.67dB (*cf*. Tab. 2). However, tuning all 5 stages of our other models hardly makes a difference. Even for 3-stage tuning we observe only minor improvements, *e.g.* from 28.02dB to 28.09dB for $\text{CSF}^3_{3\times3}$, and from 28.34dB to 28.36dB for $\text{CSF}^3_{5\times5}$.

Non-blind deconvolution. As the results in Tab. 3 show, our approach can also successfully be applied to image deconvolution in the context of blind deblurring, where kernel estimates are used to deblur the image. For the task of deconvolution, we trained a $\text{CSF}_{\text{pw.}}$ model with 288 synthetically blurred images of size 320×320 pixels. For half of the blurred training images, we used an estimate instead of the correct blur kernel $\mathbf{k}$ to cope with using erroneous kernel estimates at test time (as suggested by [25]). Our $\text{CSF}^3_{\text{pw.}}$ model outperforms the non-blind deconvolution approach by Levin *et al.* [16] and can compete with the results from Schmidt *et al.* [25] for all estimated kernels (Tab. 3). We additionally applied the same learned $\text{CSF}^3_{\text{pw.}}$ model to the recent benchmark for camera shake of Köhler *et al.* [14], where we are able to improve upon the results of the best performing method by Xu and Jia [30] about 0.56dB on average, being also 0.15dB better than the best result of [25]. Restoring each of the 800×800-sized color images of the benchmark only takes around a second with our model.

(a) Noisy, 20.30dB (b) $\text{CSF}^5_{\text{pw.}}$, 28.81dB (c) $\text{CSF}^5_{3\times3}$, 29.89dB (d) $\text{CSF}^5_{5\times5}$, 30.27dB (e) $\text{CSF}^5_{7\times7}$, 30.38dB (f) BM3D, 30.05dB

Figure 4. **Denoising example** ($\sigma = 25$, cropped): Comparison of our trained models and BM3D [5]. *Best viewed magnified on screen.*

5. Conclusions and Future Work

We presented shrinkage fields, a novel random field model applicable to the restoration of high-resolution images, which is based on an extension of the additive form of half-quadratic optimization. By replacing potentials with shrinkage functions, we increased model flexibility and enabled efficient learning of all model parameters. Experiments on image denoising and deconvolution with cascaded shrinkage fields demonstrated that fast runtime and high restoration quality can go hand-in-hand.

Future work. A next step is a more efficient GPU implementation to further improve runtime for large image sizes. Another direction is to train our model for other image quality metrics [*e.g.*, 29]. Finally, one may further investigate the non-monotonic shrinkage functions or explore more powerful multivariate shrinkage operators.

Acknowledgements. We thank Qi Gao for making results available to us. Funding is partly provided by the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 307942.

References

[1] A. Antoniadis and J. Fan. Regularization of wavelet approximations. *Journal of the American Statistical Assoc.*, 96(455):939–967, 2001.

[2] A. Barbu. Training an active random field for real-time image denoising. *IEEE TIP*, 18(11):2451–2462, Nov. 2009.

[3] Y. Chen, T. Pock, R. Ranftl, and H. Bischof. Revisiting loss-specific training of filter-based MRFs for image restoration. In *GCPR 2013*.

[4] S. Cho and S. Lee. Fast motion deblurring. *ACM T. Graphics*, 28(5), 2009.

[5] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-D transform-domain collaborative filtering. *IEEE TIP*, 16(8):2080–2095, Aug. 2007.

[6] DxO Image Science. Optics Pro 9: PRIME. http://download-center.dxo.com/Press/opticspro/v9/pr/PR-DxO-Optics-Pro-9-EN.pdf, Oct. 2013.

[7] M. Elad, B. Matalon, and M. Zibulevsky. Image denoising with shrinkage and redundant representations. In *CVPR 2006*.

[8] R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, and W. T. Freeman. Removing camera shake from a single photograph. *ACM T. Graphics*, 3(25), July 2006.

[9] Q. Gao and S. Roth. How well do filter-based MRFs model natural images? In *Pattern Recognition (DAGM) 2012*.

[10] D. Geman and G. Reynolds. Constrained restoration and the recovery of discontinuities. *TPAMI*, 14(3):367–383, 1992.

[11] D. Geman and C. Yang. Nonlinear image recovery with half-quadratic regularization. *IEEE TIP*, 4(7):932–946, 1995.

[12] Y. Hel-Or and D. Shaked. A discriminative approach for wavelet denoising. *IEEE TIP*, 17(4):443–457, 2008.

[13] J. Jancsary, S. Nowozin, and C. Rother. Loss-specific training of non-parametric image restoration models: A new state of the art. In *ECCV 2012*.

[14] R. Köhler, M. Hirsch, B. Mohler, B. Schölkopf, and S. Harmeling. Recording and playback of camera shake: Benchmarking blind deconvolution with a real-world database. In *ECCV 2012*.

[15] D. Krishnan and R. Fergus. Fast image deconvolution using hyper-Laplacian priors. In *NIPS*2009*.

[16] A. Levin, R. Fergus, F. Durand, and W. T. Freeman. Image and depth from a conventional camera with a coded aperture. *ACM T. Graphics*, 26(3):70:1–70:9, July 2007.

[17] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman. Efficient marginal likelihood optimization in blind deconvolution. In *CVPR 2011*.

[18] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Non-local sparse models for image restoration. In *ICCV 2009*.

[19] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV 2001*.

[20] J. Portilla, V. Strela, M. J. Wainwright, and E. P. Simoncelli. Image denoising using scale mixtures of Gaussians in the wavelet domain. *IEEE TIP*, 12(11):1338–1351, Nov. 2003.

[21] S. Roth and M. J. Black. Fields of experts. *IJCV*, 82(2):205–229, Apr. 2009.

[22] K. G. G. Samuel and M. F. Tappen. Learning optimized MAP estimates in continuously-valued MRF models. In *CVPR 2009*.

[23] M. Schmidt. minFunc. http://www.di.ens.fr/~mschmidt/Software/minFunc.html, 2012.

[24] U. Schmidt, J. Jancsary, S. Nowozin, S. Roth, and C. Rother. Cascades of regression tree fields for image restoration. *arXiv:1404.2086*, 2014.

[25] U. Schmidt, C. Rother, S. Nowozin, J. Jancsary, and S. Roth. Discriminative non-blind deblurring. In *CVPR 2013*.

[26] E. P. Simoncelli. Bayesian denoising of visual images in the wavelet domain. In *Bayesian Inference in Wavelet-Based Models*, volume 141 of *Lecture Notes in Statistics*, pages 291–308. Springer, 1999.

[27] M. Tappen, C. Liu, E. H. Adelson, and W. T. Freeman. Learning Gaussian conditional random fields for low-level vision. In *CVPR 2007*.

[28] Y. Wang, J. Yang, W. Yin, and Y. Zhang. A new alternating minimization algorithm for Total Variation image reconstruction. *SIAM J. Imaging Sciences*, 1(3):248–272, 2008.

[29] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE TIP*, 13(4):600–612, Apr. 2004.

[30] L. Xu and J. Jia. Two-phase kernel estimation for robust motion deblurring. In *ECCV 2010*.

[31] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *ICCV 2011*.