Loss-Specific Training of Non-Parametric Image Restoration Models: A New State of the Art

Jeremy Jancsary[1], Sebastian Nowozin[2], and Carsten Rother[2]

[1] Vienna University of Technology, Austria
[2] Microsoft Research Cambridge, United Kingdom

Abstract. After a decade of rapid progress in image denoising, recent methods seem to have reached a performance limit. Nonetheless, we find that state-of-the-art denoising methods are visually clearly distinguishable and possess complementary strengths and failure modes. Motivated by this observation, we introduce a powerful non-parametric image restoration framework based on Regression Tree Fields (RTF). Our restoration model is a densely-connected tractable conditional random field that leverages existing methods to produce an image-dependent, globally consistent prediction. We estimate the conditional structure and parameters of our model from training data so as to directly optimize for popular performance measures. In terms of peak signal-to-noise-ratio (PSNR), our model improves on the best published denoising method by at least 0.26dB across a range of noise levels. Our most practical variant still yields statistically significant improvements, yet is over 20× faster than the strongest competitor. Our approach is well-suited for many more image restoration and low-level vision problems, as evidenced by substantial gains in tasks such as removal of JPEG blocking artefacts.

1 Introduction

Image restoration has a rich history in image processing, with special cases such as image denoising having received significant attention over the years. In general terms, the problem can be defined as follows: a natural image $\mathbf{y}$ is corrupted by a distortion process $\mathbf{x} = \mathbf{f}(\mathbf{y})$. We are only given the corrupted image $\mathbf{x}$ and our goal is to recover the original image through an estimate $\hat{\mathbf{y}}$. Ideally, the estimate could be obtained through the inverse process $\mathbf{f}^{-1}$. However, in practice $\mathbf{f}$ is either stochastic in nature, or deterministic but non-invertible. As a consequence, perfect reconstruction of $\mathbf{y}$ is impossible most of the time.

While one can still aim at finding a reconstruction $\hat{\mathbf{y}}$ that is close to the original image, this immediately raises the question how the quality of such a reconstruction should be measured. In the past, the squared error $\|\mathbf{y} - \hat{\mathbf{y}}\|_2^2$ has often been chosen because it is convenient computationally. More recently, measures of *perceived quality*, such as the structural similarity index [1], have become popular. When designing an image restoration method, one should be aware of the implications of optimizing the algorithm for one measure over the other. As Fig. 1 shows, this choice can considerably impact the reconstructions. Yet, for many approaches, it is not clear which performance measure they optimize.

Ground truth	**Noisy input**	**Output** IssimRTF	**Output** MseRTF
MSE 0, ISSIM 1	MSE 3230, ISSIM 0.785	MSE 313, ISSIM 0.932	MSE 273, ISSIM 0.893

Fig. 1: Bias imposed by the loss function: We show reconstructions of an image corrupted by structured noise (cf. Section 5.3), by two models with identical specifications, except for one being optimized for information content-weighted structural similarity (ISSIM) [2], and the other for mean squared error (MSE).

Our contributions. We introduce a novel image restoration framework based on the recently introduced, non-parametric Regression Tree Fields (RTF) [3], and extend it such that the method can be optimized for any differentiable loss function. This enables us to directly optimize all aspects of our model for involved measures of perceived quality, such as structural similarity.

Our model is a highly-connected conditional random field that produces globally consistent image reconstructions tailored to specific loss functions. Both image features and reconstructions made by existing restoration methods are seamlessly integrated into the random field, and their dependency on the local image context is represented non-parametrically.

In terms of structural similarity (SSIM), peak signal-to-noise-ratio (PSNR), and mean absolute error (MAE), we obtain the best published image denoising results by a statistically significant margin. Our method is visibly better than the best published methods [4, 5]. We further present results in removal of JPEG blocking artefacts that surpass the state-of-the-art SA-DCT method [6].

Related work. Our *learning* approach is closely related to Gaussian conditional random fields [7], Regression Tree Fields [3], and their discrete counterpart [8].

Image denoising has a rich history in image processing and a wide variety of image denoising methods exist. Patch-averaging methods such as BM3D [9] build weighted averages of noisy image patches and combine these into a single prediction. Sparse coding as in the LSSC method [5] optimize a dictionary of image patches within each image. Fields-of-Experts (FoE) [10, 11] use a higher-order Markov random field as generative probabilistic image model and combine it with an analytic noise model to obtain a posterior distribution over noise-free images. The recent *expected patch log likelihood* (EPLL) method [4] uses an image patch model but combines all individual patch predictions to jointly maximize the expected patch likelihood of the predicted image.

The importance of optimizing the right *loss function* has been pointed out by [12, 13], among others. In [14], a denoising method was explicity optimized for SSIM, but only using a few manually tuned hyper parameters. Complimentarity of multiple methods was previously noticed for optical flow estimation [15].

2 Background

We first give a brief introduction to Gaussian conditional random fields [7] and point out their relation to regression tree fields [3]. Our main technical contributions will be introduced in Sec. 3.

Notation. Bold uppercase letters denote matrices and functions returning matrices, while vectors and functions that map to vectors are denoted by bold lowercase letters. Scalar entities are set in regular typeface.

2.1 Gaussian Conditional Random Fields

The *conditional* version of Gaussian random fields was first introduced in [7]. Consider an observed input image $\mathbf{x}$, and a corresponding labeling, the output image $\mathbf{y}$. Gaussian CRFs model the probability of each output given an input image, $p(\mathbf{y} \mid \mathbf{x}; \mathbf{w}) \propto \exp[-E(\mathbf{y} \mid \mathbf{x}; \mathbf{w})]$, via a quadratic energy

$$E(\mathbf{y} \mid \mathbf{x}; \mathbf{w}) \stackrel{\text{def}}{=} \tfrac{1}{2}\mathbf{y}^\mathsf{T}\mathbf{Q}(\mathbf{x}, \mathbf{w})\mathbf{y} - \mathbf{y}^\mathsf{T}\mathbf{l}(\mathbf{x}, \mathbf{w}). \tag{1}$$

Together with the input $\mathbf{x}$, the model parameters $\mathbf{w}$ determine the coefficients $\mathbf{Q}(\mathbf{x}, \mathbf{w}) \succ 0$ and $\mathbf{l}(\mathbf{x}, \mathbf{w})$ of the energy. For a given input $\mathbf{x}$, the prediction $\hat{\mathbf{y}}(\mathbf{x}, \mathbf{w})$ under this model is given by

$$\hat{\mathbf{y}}(\mathbf{x}, \mathbf{w}) \stackrel{\text{def}}{=} \operatorname*{argmax}_{\mathbf{y}} \; p(\mathbf{y} \mid \mathbf{x}; \mathbf{w}) = [\mathbf{Q}(\mathbf{x}, \mathbf{w})]^{-1}\mathbf{l}(\mathbf{x}, \mathbf{w}), \tag{2}$$

which is typically found by solving the sparse linear system $\mathbf{Q}(\mathbf{x}, \mathbf{w})\mathbf{y} = \mathbf{l}(\mathbf{x}, \mathbf{w})$. The solution is both the mean and the mode of the Gaussian density $p(\mathbf{y} \mid \mathbf{x}; \mathbf{w})$.

Parameter Estimation. Exact maximum likelihood estimation is deemed computationally infeasible in [7], so the authors estimate the parameters $\mathbf{w}$ from training data $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{N}$ by minimizing the empirical risk

$$R_\ell(\mathcal{D}, \mathbf{w}) \stackrel{\text{def}}{=} \frac{1}{N}\sum_{i}^{N} \ell\left(\hat{\mathbf{y}}(\mathbf{x}^{(i)}, \mathbf{w}), \mathbf{y}^{(i)}\right) \approx \mathbb{E}_{p(\mathbf{y},\mathbf{x})}\left[\ell\left(\hat{\mathbf{y}}(\mathbf{x}, \mathbf{w}), \mathbf{y}\right)\right]. \tag{3}$$

The loss function $\ell(\hat{\mathbf{y}}, \mathbf{y})\colon \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$ measures the error present in the prediction $\hat{\mathbf{y}}$ relative to the ground truth $\mathbf{y}$. By choosing $\hat{\mathbf{w}} = \operatorname{argmin}_{\mathbf{w}} R(\mathcal{D}, \mathbf{w})$, the model is determined such that its predictions incur the least possible loss on the training data. As we will demonstrate empirically throughout this paper, this approach has benefits beyond the computational perspective.

2.2 Regression Tree Fields

Regression tree fields [3] extend the original Gaussian CRF model in two ways. First, the energy of a labeling is specified in terms of local models over subsets of pixels. The parameters of these local models include a linear term and an inverse covariance matrix, both estimated from data. Second, the local model that is in effect at a given position of the image is determined via a regression tree (cf. Fig. 2), rendering the approach non-parametric.

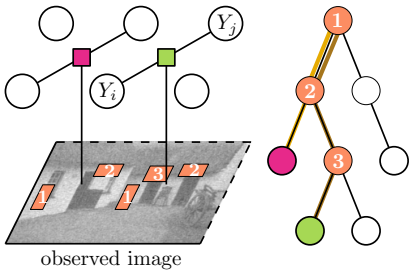


Fig. 2: Illustration of how regression trees and random fields are combined in the regression tree field: a pairwise factor type is instantiated on a grid of random variables. At each instantiation a regression tree is evaluated on the surrounding image content, performing a sequence of tests (1, 2 and 3) until a leaf node is reached. Because the image content differs around each factor instantiation, the leaf node reached can also be different. The selected leaf node determines which effective interaction is used for the factor. The conditional model now becomes a Gaussian random field, enabling efficient inference as a solution to a linear system of equations.

Parameterization. Each local energy term working on a subset of pixels is called a *factor* and denoted by F. The components of $\mathbf{y}$ corresponding to the pixels covered by factor F will be denoted by column vector $\mathbf{y}_F$. Factors sharing the same parameters are grouped into *types*. The set of all factors F of type t is denoted by $\mathcal{F}_t$. Each factor type t defines a regression tree that stores at its leaves $l \in \mathcal{L}_t$ a set of parameters $\mathbf{w}_t = \{\mathbf{L}_t^{(l)}, \mathbf{Q}_t^{(l)}\}_{l \in \mathcal{L}_t}$. We define $\mathbf{L}_t(\mathbf{x}_F)$ and $\mathbf{Q}_t(\mathbf{x}_F)$ as maps to the parameters $\mathbf{Q}_t^{(l_*)}$ and $\mathbf{L}_t^{(l_*)}$ of the particular leaf l_* that was selected for factor F of type t given the observed input image $\mathbf{x}$.

The energy of a particular factor F of type t then assumes the form

$$E_t(\mathbf{y}_F \mid \mathbf{x}_F; \mathbf{w}_t) \stackrel{\text{def}}{=} \tfrac{1}{2}\mathbf{y}_F^{\mathsf{T}}\mathbf{Q}_t(\mathbf{x}_F)\mathbf{y}_F - \mathbf{y}_F^{\mathsf{T}}\mathbf{L}_t(\mathbf{x}_F)\mathbf{b}_t(\mathbf{x}_F), \tag{4}$$

where $\mathbf{b}_t(\mathbf{x}_F) \in \mathbb{R}^{B_t}$ is a linear basis vector whose dimensionality depends on the factor type. In the simplest case, this term is constant, $\mathbf{b}_t(\cdot) = 1 \in \mathbb{R}$, but we will in addition use more general image features in our experiments. For the parameters, if $\mathbf{y}_F \in \mathbb{R}^{D_t}$, we have $\mathbf{L}_t(\mathbf{x}_F) \in \mathbb{R}^{D_t \times B_t}$ and $\mathbf{Q}_t(\mathbf{x}_F) \in \mathbb{R}^{D_t \times D_t} \succ 0$. The positive-definiteness constraint on the latter implies that all $\mathbf{Q}_t^{(l)}$ parameters must also be positive-definite and ensures that

$$E(\mathbf{y} \mid \mathbf{x}; \mathbf{w}) \stackrel{\text{def}}{=} \sum_t \sum_{F \in \mathcal{F}_t} E_t(\mathbf{y}_F \mid \mathbf{x}_F; \mathbf{w}_t) \tag{5}$$

leads to a valid Gaussian density $p(\mathbf{y} \mid \mathbf{x}; \mathbf{w}) \propto \exp[-E(\mathbf{y} \mid \mathbf{x}; \mathbf{w})]$.

For factors of any size, (5) can again be written compactly as in (1), so higher-order factors do not increase expressiveness of the model. The entries of $\mathbf{Q}(\mathbf{x}, \mathbf{w})$ and $\mathbf{l}(\mathbf{x}, \mathbf{w})$ arise as sums of per-factor contributions $\mathbf{Q}_t(\mathbf{x}_F)$ and $\mathbf{l}_t(\mathbf{x}_F) \stackrel{\text{def}}{=} \mathbf{L}_t(\mathbf{x}_F)\mathbf{b}_t(\mathbf{x}_F)$, respectively. Predictions are again obtained as in (2).

$$
\begin{aligned}
\frac{\partial \ell(\hat{\mathbf{y}}(\mathbf{x},\mathbf{w}),\mathbf{y})}{\partial w_i} &= \frac{\partial \ell(\hat{\mathbf{y}},\mathbf{y})}{\partial \hat{\mathbf{y}}} \frac{\partial([\mathbf{Q}(\mathbf{x},\mathbf{w})]^{-1}\mathbf{l}(\mathbf{x},\mathbf{w}))}{\partial w_i} && \text{(by chain rule)} \\
&= \tfrac{\partial \ell(\hat{\mathbf{y}},\mathbf{y})}{\partial \hat{\mathbf{y}}}[\mathbf{Q}(\mathbf{x},\mathbf{w})]^{-1} \times \\
&\quad \left[\tfrac{\partial \mathbf{Q}(\mathbf{x},\mathbf{w})}{\partial w_i}[\mathbf{Q}(\mathbf{x},\mathbf{w})]^{-1}\mathbf{l}(\mathbf{x},\mathbf{w}) + \tfrac{\partial \mathbf{l}(\mathbf{x},\mathbf{w})}{\partial w_i}\right] && \text{(by product rule)} \\
&= \hat{\mathbf{c}}^{\mathsf{T}} \tfrac{\partial \mathbf{Q}(\mathbf{x},\mathbf{w})}{\partial w_i}\hat{\mathbf{y}} + \hat{\mathbf{c}}^{\mathsf{T}} \tfrac{\partial \mathbf{l}(\mathbf{x},\mathbf{w})}{\partial w_i}. && (\hat{\mathbf{c}} \stackrel{\text{def}}{=} [\mathbf{Q}(\mathbf{x},\mathbf{w})]^{-1}[\tfrac{\partial \ell(\hat{\mathbf{y}},\mathbf{y})}{\partial \hat{\mathbf{y}}}]^{\mathsf{T}})
\end{aligned}
$$

Fig. 3: Derivative of the loss function with respect to a single model parameter. It is straightforward to further develop $\frac{\partial \mathbf{Q}(\mathbf{x},\mathbf{w})}{\partial w_i}$ and $\frac{\partial \mathbf{l}(\mathbf{x},\mathbf{w})}{\partial w_i}$ since the entries of $\mathbf{Q}(\mathbf{x},\mathbf{w})$ and $\mathbf{l}(\mathbf{x},\mathbf{w})$ are simply affine functions of $\mathbf{w}$, as per factor energy (4).

3 Loss-Specific Training of Regression Tree Fields

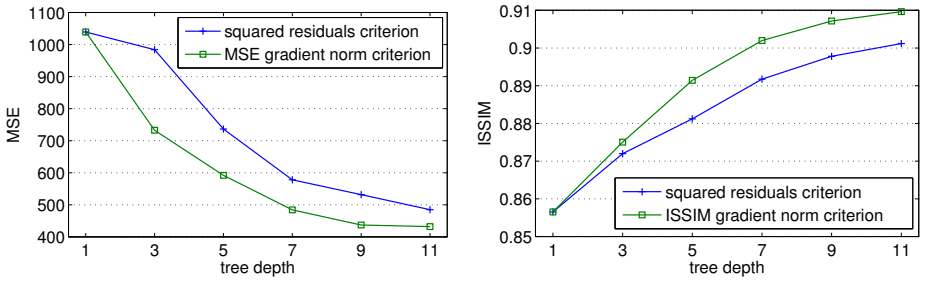
The RTF parameterization is powerful, but how can we find trees and leaf parameters to minimize the empirical risk? Ideally both the structure of the regression trees as well as their parameters are jointly chosen to minimize this objective. But because the RTF model is a random field, all parts of the model interact with each other and this makes joint minimization challenging. We now show that joint loss-based training is indeed possible, and can be achieved at computational cost similar to the original pseudo-likelihood approach of [3]. This is our main technical contribution.

3.1 Parameter Optimization

Consider first how the model parameters $\mathbf{w}$ can be optimized for a given set of regression trees associated with the factor types of our model. As first noted by [7], the prediction of a Gaussian CRF under the current model, $\hat{\mathbf{y}}(\mathbf{x},\mathbf{w})$, can be differentiated with respect to the model parameters $\mathbf{w}$. Since the RTF parameterization admits the canonical form of a Gaussian CRF, this approach is applicable in our case.

In Fig. 3, we develop the derivative of the loss function with respect to a single model parameter w_i. To evaluate the loss, the prediction $\hat{\mathbf{y}}$ on the training image is required. The derivative furthermore requires the solution $\hat{\mathbf{c}}$ to a second sparse linear system of equal dimensionality. To evaluate the full gradient of the empirical risk (3), these two solutions need to be obtained once per training image. A minor technical complication is that the $\mathbf{Q}_t^{(l)}$ parameters must remain positive-definite, which can be handled efficiently by projecting onto the convex cone of positive-definite matrices [3]. The parameters can then be optimized using any projected gradient method [16, 17].

Note that this procedure measures the quality of the actual predictions of our model on the training data and adjusts the model parameters so as to optimize these predictions in the specific sense of loss function ℓ. This can be thought of as a "self correcting" mechanism. The practical benefits of this approach over pseudo-likelihood estimation, as used in [3], are explored in detail in Sec. 5.



(a) **MSE:** lower is better (b) **ISSIM:** higher is better

Fig. 4: Benefits of splitting tree nodes according to the largest increase in gradient norm. We plot the restoration performance of an RTF that is trained for the appropriate loss as a function of the depth of the regression trees. The performance is evaluated on test data of the structured noise dataset (cf. Sec. 5). Nodes are split either by maximizing the norm of the gradient with respect to the model parameters (gradient norm criterion), or using the classic squared residuals criterion. (a) Comparison in terms of MSE: the squared residuals criterion aims at the appropriate loss, but cannot take into account that the trees are combined in a random field. (b) Comparison in terms of ISSIM: again, the squared residuals criterion disregards dependencies among trees, and additionally optimizes the wrong loss, so the gradient norm criterion is even more important.

3.2 Tree Induction

Conversely, assume that the model parameters have been optimized for the current tree structure. To allow for further descent in the objective, it is desirable to further grow the trees, effectively introducing new model parameters at the newly added leaf nodes. A common approach in growing stand-alone regression trees is to select splits that minimize the sum of *squared residuals*, i.e. the sum of squared distances of individual data points from their mean [18]. This approach is not well-motivated when learning an RTF. First, it is often desirable to use a loss function other than squared error, and second, the regression trees of the factor types interact with each other in the random field, so it is misguided to grow each tree as if their predictions were made separately.

In [3], the authors show that for the pseudo-likelihood objective, it is possible to efficiently split tree nodes based on the largest increase in gradient norm. An increase in the gradient norm indicates that further decrease in the objective function is possible. This approach avoids the mistake of learning trees separately, but still does not account for the loss function at test time.

We now demonstrate that a similar approach is possible using *any* differentiable loss function ℓ, and indeed it results in considerable gains in practice (cf. Fig. 4). The idea is to consider the gradient contributions by individual factors as separate data points, in terms of which the split criterion can be evaluated efficiently. Let us make this more precise.

Lemma 1 *For any differentiable loss function $\ell(\cdot,\cdot)$, the derivative of $R_\ell(\mathcal{D},\mathbf{w})$ with respect to the parameters of leaf l of factor type t, $\frac{\partial R_\ell(\mathcal{D},\mathbf{w})}{\partial \mathbf{w}_t^{(l)}}$, decomposes into contributions of the factors $F \in \mathcal{F}_t^{(l)}$ for which leaf l is active.*

Proof. Consider the derivative of the loss function with respect to a single model parameter w_i, given in Fig. 3. By further noting that the entries of $\mathbf{Q}(\mathbf{x},\mathbf{w})$ and $\mathbf{l}(\mathbf{x},\mathbf{w})$ are affine functions of the $\mathbf{w}_t^{(l)} = \{\mathbf{Q}_t^{(l)}, \mathbf{L}_t^{(l)}\}$ parameters, arising from (5), we develop the derivatives with respect to these parameters as

$$\frac{\partial \ell(\hat{\mathbf{y}}(\mathbf{x},\mathbf{w}),\mathbf{y})}{\partial \mathbf{Q}_t^{(l)}} = \sum_{F \in \mathcal{F}_t^{(l)}} \hat{\mathbf{c}}_F \hat{\mathbf{y}}_F^\mathsf{T} \quad \text{and} \quad \frac{\partial \ell(\hat{\mathbf{y}}(\mathbf{x},\mathbf{w}),\mathbf{y})}{\partial \mathbf{L}_t^{(l)}} = \sum_{F \in \mathcal{F}_t^{(l)}} \hat{\mathbf{c}}_F [\mathbf{b}_t(\mathbf{x}_F)]^\mathsf{T}, \quad (6)$$

where we again use $\hat{\mathbf{y}}$ and $\hat{\mathbf{c}}$ to denote the solutions to the sparse linear systems that must be solved (cf. Fig. 3), and $\hat{\mathbf{y}}_F$ and $\hat{\mathbf{c}}_F$ denote column vectors containing only the components corresponding to the pixels covered by F. Notably, $\hat{\mathbf{c}}$ is the only term in (6) that depends on the loss function, so the decomposition over factor contributions holds irrespective of the definition of $\ell(\cdot,\cdot)$, as long as the function is differentiable and $\hat{\mathbf{c}}$ is thus well-defined. □

To state our main result, let $\mathbf{w}$ denote the model parameters before a leaf l is split into two new leaves l_{left} and l_{right}. We denote by $\mathbf{w}'$ the parameters after a particular split. Since l is no longer a leaf in the new tree, and two new leaves are added, we have $\mathbf{w}' = \{\mathbf{w} \setminus \mathbf{w}_t^{(l)}\} \cup \{\mathbf{w}_t^{(l_{\text{left}})}, \mathbf{w}_t^{(l_{\text{right}})}\}$.

Proposition 1 *The increase in gradient norm, $\Delta = \|\frac{\partial R_\ell(\mathcal{D},\mathbf{w}')}{\partial \mathbf{w}'}\| - \|\frac{\partial R_\ell(\mathcal{D},\mathbf{w})}{\partial \mathbf{w}}\|$, achieved by a split of leaf l of factor type t can be computed purely locally in terms of the contributions by the factors $F \in \mathcal{F}_t^{(l)}$ for which leaf l is active.*

Proof. Consider the gradient norm before a split, $C \stackrel{\text{def}}{=} \|\frac{\partial R_\ell(\mathcal{D},\mathbf{w})}{\partial \mathbf{w}}\|$. The squared norm decomposes over the components of the individual leaves, so we obtain

$$\Delta = \sqrt{C^2 - \left\|\tfrac{\partial R_\ell(\mathcal{D},\mathbf{w})}{\partial \mathbf{w}_t^{(l)}}\right\|_2^2 + \left\|\tfrac{\partial R_\ell(\mathcal{D},\mathbf{w}')}{\partial \mathbf{w}_t^{(l_{\text{left}})}}\right\|_2^2 + \left\|\tfrac{\partial R_\ell(\mathcal{D},\mathbf{w}')}{\partial \mathbf{w}_t^{(l_{\text{right}})}}\right\|_2^2} - C. \quad (7)$$

Note that C remains constant among splits and can be pre-computed. By our result of Lemma 1, the other terms depend only on the individual contributions of factors $F \in \mathcal{F}_t^{(l)} = \mathcal{F}_t^{(l_{\text{left}})} \cup \mathcal{F}_t^{(l_{\text{right}})}$ and can thus be computed efficiently. □

In practice, when evaluating split candidates, we initialize the parameters of the candidate leaves to $\mathbf{w}_t^{(l_{\text{left}})} = \mathbf{w}_t^{(l_{\text{right}})} = \mathbf{w}_t^{(l)}$. This way, the increase in gradient norm achieved by a split can be interpreted as a measure of how much gain is possible over the current parameter setting. Moreover, this approach ensures monotonic decrease in the objective function, since immediately after a split, the same local factor models are in effect as before. However, the degrees of freedom have increased, so further progress in the objective may be possible.

Start with trees consisting solely of root nodes;
repeat
  (Re-)optimize all parameters $\mathbf{w}$ at the current leaf nodes ;
  foreach *training image* i **do**
    Solve two sparse linear systems to obtain $\hat{\mathbf{y}}^{(i)}$ and $\hat{\mathbf{c}}^{(i)}$, as in Fig. 3;
  foreach *factor type* t *and its regression tree* **do**
    foreach *training image* i **do**
      foreach *factor* $F \in \mathcal{F}_t$ *of matching type* **do**
        Compute the gradient contribution via $\hat{\mathbf{y}}_F^{(i)}$ and $\hat{\mathbf{c}}_F^{(i)}$, as in (6) ;
        Sort F and its contribution into the target leaf ;
    foreach *leaf* l **do**
      From the contributions, find the split that maximizes $\|\frac{\partial R_\ell(\mathcal{D},\mathbf{w}')}{\partial \mathbf{w}'}\|$;
      Split node l into new child leaves $(l_{\text{left}}, l_{\text{right}})$;
      Set $\mathbf{w}_t^{(l_{\text{left}})} \leftarrow \mathbf{w}_t^{(l)}$ and $\mathbf{w}_t^{(l_{\text{right}})} \leftarrow \mathbf{w}_t^{(l)}$;
until *maximum depth reached*;
Optimize all parameters $\mathbf{w}$ to final accuracy ;

Alg. 1: The parameters and the tree structure of an RTF can be optimized jointly for loss function ℓ in a greedy iterative algorithm. The lines that changed compared to the original joint pseudo-likelihood training algorithm in [3] are highlighted in blue.

3.3 Summary of Loss-Specific Training

In this section, we developed procedures for optimizing the model parameters given a fixed set of regression trees, and for splitting the trees given the model parameters that are optimal for the current tree structure. Using these building blocks, one can start from regression tree stumps consisting solely of root nodes and optimize over the model parameters and the tree structure in a greedy manner. At each iteration, the model parameters are first optimized, and the leaves of the trees are then split according to the largest increase in gradient norm to enable further progress in the objective. This iterative scheme is outlined in Alg. 1. The main hyper parameter is the maximum depth of regression trees, which we suggest should be determined from validation data.

4 Application to Image Denoising and Deblocking

The image restoration problem maps into our framework as follows. The observed input image $\mathbf{x}$ denotes the corrupted image, which is generated from ground truth $\mathbf{y}$ via some perturbation process. In the classical image denoising setting, an additive white Gaussian noise assumption is made, that is, $\mathbf{x} = \mathbf{y} + \mathbf{z}$ for $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$. However, we will also consider images corrupted by JPEG blocking artefacts and a structured noise model (cf. Fig. 1) in our experiments. In fact, the ability to handle arbitrary noise models is a major strength of our approach.

The restored image $\hat{\mathbf{y}}$ is then obtained as the prediction of our model given the corrupted input, i.e. $\hat{\mathbf{y}} \stackrel{\text{def}}{=} \hat{\mathbf{y}}(\mathbf{x}, \mathbf{w}) = [\mathbf{Q}(\mathbf{x}, \mathbf{w})]^{-1}\mathbf{l}(\mathbf{x}, \mathbf{w})$.

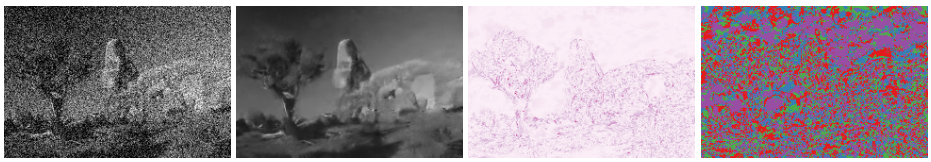


Fig. 5: Existing denoising methods contain complementary information: we process a noisy image (left, $\sigma = 40$) using four denoising methods (BM3D, EPLL, FoE, LSSC). For each pixel and using the ground truth we select the best possible prediction among the methods (2nd column). Compared to the ground truth this prediction has some remaining error (3rd column). In different parts of the image, different methods are selected over larger regions (4th column), indicating that the methods have varying strengths that depend on the image content.

Feature engineering. Remember that the entries of $\mathbf{Q}(\mathbf{x}, \mathbf{w})$ and $\mathbf{l}(\mathbf{x}, \mathbf{w})$ are sums of per-factor contributions $\mathbf{Q}_t(\mathbf{x}_F)$ and $\mathbf{l}_t(\mathbf{x}_F) \stackrel{\text{def}}{=} \mathbf{L}_t(\mathbf{x}_F)\mathbf{b}_t(\mathbf{x}_F)$, which depend on the evaluation of a regression tree.

Depending on our system configuration, the basis vector $\mathbf{b}_t(\mathbf{x}_F)$ in the leaf model of a unary or pairwise factor is initialized from one or more of the following sources: a) the corrupted image itself, b) responses of a fixed filterbank, and c) predictions by base methods; at the position of the pixels covered by the factor.

In the regression trees, we use feature tests inspecting the input image, the filter responses and the output of base methods at offsets relative to the position of a factor. For JPEG deblocking, we use two more feature tests indicating whether the position of the factor lies at the boundary of a 4×4 or 8×8 block.

For the filter responses, we use the RFS filterbank[3] to derive 38 responses per pixel of the input image. The use of base methods varies depending on the restoration task and will be described per experiment. Our motivation is as follows: for many established image restoration tasks, there exist highly engineered task-specific methods. These competing approaches often contain complementary information, as illustrated for denoising in Fig. 5. In our non-parametric field model, the relative contribution of the base methods can be learned per image context, such that their complementary strenghts can be exploited.

Model selection and training. We use a similar RTF specification as in [3], considering random fields models with dense pairwise connectivity in either a 3×3 or a 5×5 window centered around the current pixel, and tree depths of 1, 3, 5, 7, 8 or 9. The best settings are chosen based on validation data (in most cases, a 5×5 field at depth 8 or 9 was selected).

We follow the regularization procedure of [3], i.e., we restrict each $\mathbf{Q}_t^{(l)}$ to be positive-definite and furthermore bound its eigenvalues by $(10^{-2}, 10^2)$. We found loss-specific training to be rather insensitive to the choice of these hyper parameters, so we did not have to choose them based on model selection.

[3] `http://www.robots.ox.ac.uk/~vgg/research/texclass/filters.html`

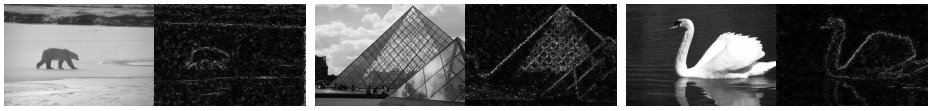


Fig. 6: What has our model learned about images? On the test set we visualize the original image and the difference image between our best method, PsnrRTF$_{\text{All}}$, and the uniform average of our competitors' predictions (UniformAvg). One can clearly see structure in the difference: our model has learned to refine smooth areas (left), texture patterns (middle), and edges (right).

Ground truth **Input** ($\sigma = 40$) **Out.** PsnrRTF$_{\text{All}}$ **Output** EPLL

Fig. 7: Visual improvement in denoising quality: Our PsnrRTF$_{\text{All}}$-system clearly produces more natural restorations than EPLL [4], see the supplement.

We train and evaluate our systems using peak-signal-to-noise ratio (PSNR); mean absolute error (MAE); and unweighted structural similarity (SSIM), defined over fixed 8×8 windows as in [2]. To *train* for MAE, we use the smoothed, differentiable version in [7], but evaluate in terms of the original definition. All measures are computed per image and then averaged over the number of images.

5 Experiments

We adhere to a strict experimental protocol, using the disjoint training, validation and test splits from the BSDS500 database [19] (images scaled by a factor of 0.5). In particular, we pay attention to clearly separate the model selection from the final performance evaluation. We perform model selection using the validation set only and evaluate the performance on the test set only once. Given the final results on the test set, we perform a Wilcoxon signed-ranks test [20] testing for the null-hypothesis of equal performance between competing methods.

We consider twelve configurations of our method, based on the combinations of loss functions we optimize (PsnrRTF, MaeRTF, SsimRTF, NlplRTF) and three different feature sets: using only the filterbank (RTF$_{\text{Plain}}$), the filterbank and the output of BM3D (RTF$_{\text{Bm3d}}$), as well as the filterbank, FoE, BM3D, LSSC and EPLL (RTF$_{\text{All}}$). Note that the NlplRTF-systems are trained to minimize the negative log-pseudolikelihood, so NlplRTF$_{\text{Plain}}$ corresponds to the system configuration in [3]. As for the loss-specific systems, we use joint training of trees and parameters for the NlplRTF-systems, since [3] have shown this approach works at least as well as separate training of regression trees.

Table 1: Denoising test set results for natural images. We compare state-of-the-art competitors to configurations of our method (RTF). For each measure, the result of the strongest competitor is printed in blue, and the best RTF result is printed in green. The gain of our method is statistically significant as per Wilcoxon signed-ranks test ($p < 10^{-5}$ for each blue-green pair in each column).

Method	**PSNR** (↑ better)				**MAE** (↓ better)				**SSIM** (↑ better)			
σ	20	30	40	50	20	30	40	50	20	30	40	50
Input	22.11	18.59	16.09	14.15	15.96	23.93	31.91	39.89	0.541	0.401	0.307	0.242
FoE [11]	28.87	26.81	25.45	24.47	6.79	8.56	10.03	11.24	0.848	0.776	0.712	0.660
BM3D [9]	29.25	27.32	25.98	25.09	6.40	7.95	9.25	10.22	0.855	0.793	0.741	0.699
LSSC [5]	29.40	27.39	26.08	25.09	6.39	7.96	9.23	10.33	0.861	0.799	0.745	0.700
EPLL [4]	29.38	27.44	26.17	25.22	6.37	7.90	9.12	10.17	0.864	0.800	0.747	0.703
UniformAvg	29.47	27.50	26.21	25.25	6.30	7.84	9.08	10.12	0.863	0.802	0.749	0.705
PsnrRTF$_{\text{Plain}}$	28.95	26.97	25.71	24.76	6.78	8.44	9.72	10.85	0.840	0.771	0.716	0.666
PsnrRTF$_{\text{Bm3d}}$	29.52	27.58	26.24	25.38	6.23	7.73	8.99	9.92	0.863	0.803	0.750	0.711
PsnrRTF$_{\text{All}}$	29.67	27.72	26.43	25.51	6.14	7.62	8.80	9.78	0.868	0.809	0.758	0.717
MaeRTF$_{\text{Plain}}$	28.92	26.94	25.69	24.75	6.78	8.43	9.71	10.81	0.840	0.771	0.715	0.669
MaeRTF$_{\text{Bm3d}}$	29.53	27.58	26.22	25.36	6.21	7.71	8.96	9.88	0.863	0.803	0.750	0.711
MaeRTF$_{\text{All}}$	29.67	27.72	26.43	25.50	6.12	7.59	8.77	9.74	0.867	0.808	0.758	0.717
SsimRTF$_{\text{Plain}}$	28.49	26.55	25.31	24.41	7.17	8.92	10.23	11.34	0.844	0.778	0.721	0.676
SsimRTF$_{\text{Bm3d}}$	29.17	27.13	25.69	24.85	6.60	8.31	9.80	10.79	0.868	0.809	0.757	0.719
SsimRTF$_{\text{All}}$	29.23	27.14	25.67	24.75	6.60	8.39	9.96	11.06	0.872	0.815	0.766	0.726
NlplRTF$_{\text{Plain}}$	28.61	26.66	25.32	24.42	7.09	8.80	10.28	11.37	0.828	0.758	0.694	0.653
NlplRTF$_{\text{Bm3d}}$	29.43	27.44	26.10	25.21	6.32	7.88	9.16	10.13	0.861	0.799	0.747	0.708
NlplRTF$_{\text{All}}$	29.60	27.64	26.34	25.40	6.20	7.71	8.92	9.93	0.866	0.806	0.755	0.714

5.1 Denoising

We perturb the images of the BSDS500 database with additive white Gaussian noise (AWGN), for noise levels $\sigma \in \{20, 30, 40, 50\}$. The results achieved by our system configurations, as well as the strongest competitors, are shown in Table 1.

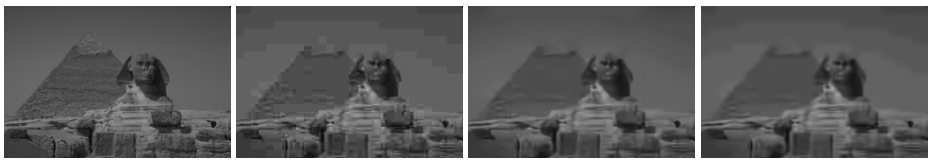
In all cases, an RTF$_{\text{All}}$-system trained for the specific loss achieves the best result. In terms of PSNR, the gains over the best published method range from 0.26dB to 0.29dB across the different noise levels. This is a substantial improvement and is clearly visible, as shown in Fig. 7. The gains are even more pronounced in terms of MAE and SSIM. Note that it is not at all apparent how the other systems could be made to take into account these measures.

Many applications require the right trade-off between speed and quality. In Table 2, we show the average running time of the considered denoising methods on 241×161 pixel images. The improvement of our RTF$_{\text{Bm3d}}$-systems over the best published methods (LSSC and EPLL) ranges from 0.07dB to 0.16dB and is statistically significant, yet the method runs over twenty times faster.

Observe that RTFs trained for a specific loss perform much better than the NlplRTF-systems of [3]. The impressive difference between PsnrRTF$_{\text{Plain}}$ and NlplRTF$_{\text{Plain}}$ ranges from 0.31db to 0.39db. This gap narrows as more

Table 2: Typical running time of denoising methods, for a single natural image (241×161 pixels) on an 8-core Intel Xeon machine (2.4GHz). RTF using BM3D as a feature (RTF_{BM3D}) offers the best computational trade-off, as it is better than the strongest competitor (EPLL), yet about twenty times faster.

	FoE	**BM3D**	**LSSC**	**EPLL**	$\textbf{RTF}_{\text{PLAIN}}$	$\textbf{RTF}_{\text{BM3D}}$	$\textbf{RTF}_{\text{ALL}}$
Running time (s)	1,063	0.9	172	38	0.7	1.6	1,275
PSNR ($\sigma = 30$)	26.81	27.32	27.39	27.44	26.97	27.58	27.72



Ground truth **Lossy JPEG** **Out.** $\text{MAERTF}_{\text{SADCT}}$ **Output** SA-DCT

Fig. 8: Improvement in JPEG deblocking (quality 10): SA-DCT fails to remove the blocking artefacts in the sky while our $\text{MAERTF}_{\text{SADCT}}$-system succeeds.

powerful features are added to the models, but remains statistically significant. However, training of NLPLRTF-systems is typically faster (for example 22h for $\text{NLPLRTF}_{\text{BM3D}}$ versus 35h for $\text{PSNRRTF}_{\text{BM3D}}$), and it supports subsampling of pixels both for parameter estimation and node splitting, while subsampling is only possible for the latter in our approach.

A natural question is whether the gains of our approach simply stem from averaging of strong base methods. This is not the case – in Table 1, we show the performance achieved by averaging the predictions of our competitors uniformly (UNIFORMAVG). Our RTF_{All}-systems outperform this naïve strategy by a wide margin. The difference is statistically significant and clearly visible, cf. Fig. 6.

5.2 Deblocking

To demonstrate once more that our approach is very flexible and can be applied to numerous low-level vision and imaging problems, we distort the images of the BSDS500 database by JPEG blocking artefacts. We use the JPEG quality settings 10, 20, 30 and 40 in MATLAB JPEG encoder. Again, we compare our loss-specific system configurations to the original RTF approach based on pseudo-likelihood (NLPLRTF) [3], as well as the state-of-the-art deblocking method SA-DCT [6]. We consider configurations of our system that use only the filterbank ($\text{RTF}_{\text{PLAIN}}$), as well as those that include SA-DCT as a base method ($\text{RTF}_{\text{SADCT}}$).

Again, loss-specific training of RTFs achieves the best results (cf. Table 3). The gains over SA-DCT are statistically significant and clearly visible, as demonstrated in Fig. 8. The PSNR and MAE measures are strongly correlated in this task, so there is little difference between PSNRRTF and MAERTF, but SSIMRTF achieves better results in terms of the loss it optimizes.

Table 3: JPEG deblocking results for natural images. We compare SA-DCT, a state-of-the-art deblocking method, to configurations of our method (RTF). The best RTF result is printed in green. Statistically significant gains are underlined.

Method	**PSNR** (↑ better))				**MAE** (↓ better)				**SSIM** (↑ better)			
quality	10	20	30	40	10	20	30	40	10	20	30	40
Input	26.62	28.80	30.08	31.01	8.64	6.64	5.70	5.11	0.790	0.868	0.900	0.918
SA-DCT [6]	27.44	29.48	30.70	31.58	7.67	6.00	5.20	4.69	0.810	0.880	0.909	0.926
PsnrRTF$_{\text{Plain}}$	27.66	29.84	31.15	32.10	7.49	5.78	4.95	4.44	0.817	0.886	0.914	0.930
PsnrRTF$_{\text{Sadct}}$	27.70	29.86	31.17	32.12	7.43	5.75	4.94	4.42	0.819	0.887	0.915	0.931
MaeRTF$_{\text{Plain}}$	27.66	29.83	31.16	32.10	7.46	5.77	4.94	4.43	0.817	0.886	0.914	0.930
MaeRTF$_{\text{Sadct}}$	27.71	29.87	31.17	32.13	7.40	5.73	4.93	4.41	0.818	0.887	0.915	0.930
SsimRTF$_{\text{Plain}}$	27.18	29.47	30.81	31.80	8.07	6.12	5.23	4.66	0.823	0.889	0.916	0.932
SsimRTF$_{\text{Sadct}}$	27.25	29.49	30.82	31.82	7.97	6.10	5.22	4.64	0.824	0.890	0.917	0.932
NlplRTF$_{\text{Plain}}$	27.50	29.69	31.01	31.96	7.64	5.90	5.05	4.53	0.813	0.883	0.913	0.928
NlplRTF$_{\text{Sadct}}$	27.61	29.76	31.06	32.00	7.52	5.84	5.01	4.49	0.816	0.885	0.913	0.929

5.3 Structured Noise

We simulate synthetic dust artifacts as follows. For each image we sample a random number of dust particles (Poisson-distributed with $\lambda = 20$), and then for each particle we sample a position uniformly at random on the image plane. Each dust particle decreases the image intensity according to a fixed 2D Gaussian-shaped function with scaling of $s = 5$ (small dust) or $s = 20$ (large dust) pixels. Our restoration framework is highly capable of recovering the images, even for the large-dust case (cf. Fig. 1), while all other denoising methods we discussed are unsuitable. For lack of a competitor, further results are shown in the supplement.

6 Conclusion

We proposed a novel framework for image restoration, based on three ideas. First, non-parametric regression tree fields as a flexible representation. Second, loss-specific training, selecting all aspects of the model so as to optimize a task-specific loss such as SSIM. Third, making efficient use of existing restoration methods, combining and improving their predictions. All three ideas *together* produce a new state-of-the-art in image denoising and JPEG deblocking.

Importantly, we leveraged the work that has been invested into specialized methods for these tasks by incorporating their predictions into our field model. This makes our model future-proof and applicable to a wide variety of tasks.

Is image denoising solved? We believe it is not, since common performance measures are just a proxy for the quality as perceived by a human. With our model we can efficiently optimize for a given measure, and by analyzing the loss-specific predictions, we hope that in the future this will provide insight into the shortcomings that are possibly remaining in measures such as SSIM.

Acknowledgments. The authors would like to thank Jamie Shotton and Toby Sharp for helpful discussions and Peter Gehler for help with the experiments.

References

1. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. **13**(4) (2004) 600–612
2. Wang, Z., Simoncelli, E.P.: Maximum differentiation (MAD) competition: A methodology for comparing computational models of perceptual discriminability. Journal of Vision **8**(12) (2008) 1–13
3. Jancsary, J., Nowozin, S., Sharp, T., Rother, C.: Regression tree fields – An efficient, non-parametric approach to image labeling problems. In: CVPR. (2012)
4. Zoran, D., Weiss, Y.: From learning models of natural image patches to whole image restoration. In: ICCV. (2011)
5. Mairal, J., Bach, F., Ponce, J., Shapiro, G., Zisserman, A.: Non-local sparse models for image restoration. In: ICCV. (2009)
6. Foi, A., Katkovnik, V., Egiazarian, K.O.: Pointwise shape-adaptive DCT for high-quality denoising and deblocking of grayscale and color images. IEEE Trans. Image Process. **16**(5) (2007) 1395–1411
7. Tappen, M.F., Liu, C., Adelson, E.H., Freeman, W.T.: Learning Gaussian conditional random fields for low-level vision. In: CVPR. (2007)
8. Nowozin, S., Rother, C., Bagon, S., Sharp, T., Yao, B., Kohli, P.: Decision tree fields. In: ICCV. (2011)
9. Dabov, K., Foi, A., Katkovnik, V., Egiazarian, K.O.: Image denoising by sparse 3-D transform-domain collaborative filtering. IEEE Trans. Image Process. **16**(8) (2007) 2080–2095
10. Roth, S., Black, M.: Fields of experts. International Journal of Computer Vision **82** (2009) 205–229
11. Schmidt, U., Gao, Q., Roth, S.: A generative perspective on MRFs in low-level vision. In: CVPR. (2010)
12. Samuel, K.G.G., Tappen, M.F.: Learning optimized MAP estimates in continuously-valued MRF models. In: CVPR. (2009)
13. Pletscher, P., Nowozin, S., Kohli, P., Rother, C.: Putting MAP back on the map. In Mester, R., Felsberg, M., eds.: Pattern Recognition. Volume 6835 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2011) 111–121
14. Estrada, F., Fleet, D., Jepson, A.: Stochastic image denoising. In: BMVC. (2009)
15. Aodha, O.M., Brostow, G.J., Pollefeys, M.: Segmenting video into classes of algorithm-suitability. In: CVPR. (2010)
16. Birgin, E.G., Martinez, J.M., Raydan, M.: Nonmonotone spectral projected gradient methods on convex sets. SIAM J. Optim. **10**(4) (2000) 1196–1211
17. Schmidt, M., van den Berg, E., Friedlander, M., Murphy, K.: Optimizing costly functions with simple constraints: A limited-memory projected quasi-Newton algorithm. In: AISTATS. (2009)
18. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Wadsworth Publishing Company, Belmont, California, U.S.A. (1984)
19. Arbelaez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **33**(5) (2011) 898–916
20. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research **7** (January 2006) 1–30