Regression Tree Fields — An Efficient, Non-parametric Approach to Image Labeling Problems

Jeremy Jancsary[1], Sebastian Nowozin[2], Toby Sharp[2], and Carsten Rother[2]
[1]Vienna University of Technology [2]Microsoft Research Cambridge

Abstract

We introduce Regression Tree Fields (RTFs), a fully conditional random field model for image labeling problems. RTFs gain their expressive power from the use of non-parametric regression trees that specify a tractable Gaussian random field, thereby ensuring globally consistent predictions. Our approach improves on the recently introduced decision tree field (DTF) model [14] in three key ways: (i) RTFs have tractable test-time inference, making efficient optimal predictions feasible and orders of magnitude faster than for DTFs, (ii) RTFs can be applied to both discrete and continuous vector-valued labeling tasks, and (iii) the entire model, including the structure of the regression trees and energy function parameters, can be efficiently and jointly learned from training data. We demonstrate the expressive power and flexibility of the RTF model on a wide variety of tasks, including inpainting, colorization, denoising, and joint detection and registration. We achieve excellent predictive performance which is on par with, or even surpassing, DTFs on all tasks where a comparison is possible.

1. Introduction

Probabilistic graphical models have emerged as a standard tool for building computer vision models [4, 12]. They allow us to make predictions given noisy image observations by relating the observed image to the variables of interest in a coherent way. In many applications—*e.g.* stereo reconstruction, denoising, and registration—the variables we would like to infer are vector-valued, one for each pixel.

There are three *key challenges* that need to be overcome in order to use graphical models to solve computer vision tasks: parameterization, inference, and learning. *Parameterization* is the specification of the model structure and its parameters that need to be estimated from training data. *Inference* refers to the test-time task of reasoning about the state of the variables that interest us, given the observation. *Learning* means to estimate model parameters from training data so as to make good predictions at test-time. All these tasks are related to each other and even for simple models turn out to be intractable, necessitating approximations in model specification, inference, and estimation [12].

Our work addresses all three challenges. We propose Regression Tree Fields (RTF), which are non-parametric Gaussian conditional random fields. RTFs are *parameterized* by non-parametric regression trees, allowing universal specification of interactions between image observations and variables. Being a Gaussian model, RTFs allow *exact and efficient inference* at test-time. The structure and parameters of the RTF model can be efficiently *learned* from training data; learning is scalable and fully parallelizable.

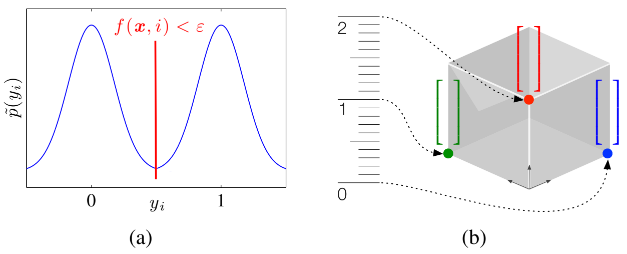


Figure 1. Expressiveness of the RTF model. (a) Via conditioning, multi-modal empirical distributions can be split into distributions that are closer to being Gaussian. (b) High-dimensional encoding of discrete labels allows for richer interaction terms.

Is a Gaussian model expressive enough? A Gaussian model is tractable but restrictive, *e.g.* it is always uni-modal and symmetric. The RTF model gains its expressive power in two ways (see Figure 1): First, by conditioning via non-parametric regression trees, it can draw on different Gaussian models in different image-dependent contexts. This mitigates the uni-modality restriction and extends earlier parametric Gaussian CRF models [21]. Second, in discrete tasks, the ability to learn all coefficients of the underlying quadratic energies—together with high-dimensional encoding of the labels—lifts the common restriction to associative interactions [22, 23]. We will demonstrate empirically that the expressive power of the interaction terms in our model is comparable to discrete random fields.

1.1. Related Work

In a sequence of papers, [21, 22, 16] Tappen and colleagues proposed a CRF model for continuous labels that is closest to our work. In [21] a Gaussian CRF is used where the energy function is defined by means of squared difference between observed image and filter responses of the labeling. The model is trained discriminatively because likelihood maximization is deemed infeasible, yielding a non-convex optimization problem. In an extension [22] the model is made applicable to discrete labeling tasks using a logistic function to map continuous outputs to per-class probabilities. Finally, in [16] non-quadratic energies are considered and a non-convex learning problem is solved to optimize the empirical risk. Our proposed RTF model differs in three ways from this line of work; (i) our conditional interactions are non-parametric and therefore less restrictive, (ii) we do not use a restricted quadratic form but allow arbitrary positive-definite precision matrices to be learned, and (iii) we use a *convex* likelihood-based learning objective such that the resulting model represents probabilities.

The Fields-of-Experts model (FoE) [15, 18] is a successful natural image prior based on a non-convex energy function incorporating responses of filters evaluated on a large number of random variables. Compared to FoE, we are limited to pairwise interactions, allowing us to predict very fast at test-time. By evaluating filters on the *input image*, as in [21], we achieve expressivity similar to the FoE model.

Variational minimization and convex relaxation approaches are also related to our work in that they solve similar computer vision tasks and are very efficient at test-time. For a recent overview, see [6]. Our work is restricted to quadratic energies and addresses the parametrization and learning problems, but it may be possible to extend our approach to more general convex energies such as [19].

2. Model

We use $\boldsymbol{x} \in \mathcal{X}$ to refer to an observed image. Our goal is to infer a joint continuous labeling $\boldsymbol{y} \in \mathcal{Y}$, one for each pixel, $\boldsymbol{y}_i \in \mathbb{R}^m$, with $\boldsymbol{y} = \{\boldsymbol{y}_i\}_{i \in \mathcal{V}}$, where $\mathcal{V}$ denotes the set of all pixels. We define the relationship between $\boldsymbol{x}$ and $\boldsymbol{y}$ by a quadratic energy function E,[1]

$$E(\boldsymbol{y}, \boldsymbol{x}, \mathcal{W}) = \tfrac{1}{2}\langle\!\langle \boldsymbol{y}\boldsymbol{y}^T, \boldsymbol{\Theta}(\boldsymbol{x}, \mathcal{W}) \rangle\!\rangle - \langle \boldsymbol{y}, \boldsymbol{\theta}(\boldsymbol{x}, \mathcal{W}) \rangle. \quad (1)$$

We use $\mathcal{W}$ to denote our model parameters, which determine the vector $\boldsymbol{\theta}(\boldsymbol{x}, \mathcal{W}) \in \mathbb{R}^{m|\mathcal{V}|}$ and the positive-definite matrix $\boldsymbol{\Theta}(\boldsymbol{x}, \mathcal{W}) \in \mathbb{S}_{++}^{m|\mathcal{V}|}$ in a manner that will be made precise shortly. These are the canonical parameters of the corresponding $m|\mathcal{V}|$-dimensional Gaussian density

$$p(\boldsymbol{y} \mid \boldsymbol{x}; \mathcal{W}) \propto \exp\{-E(\boldsymbol{y}, \boldsymbol{x}, \mathcal{W})\}, \quad (2)$$

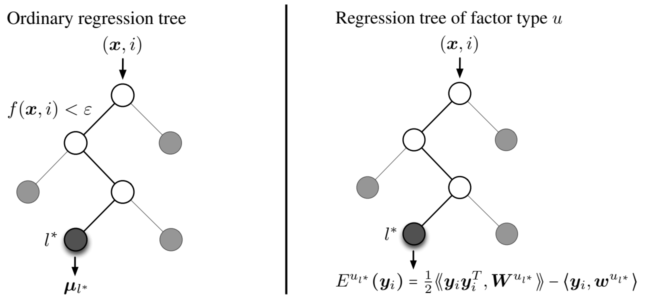
in which $\boldsymbol{\Theta}(\boldsymbol{x}, \mathcal{W})$ plays the role of the inverse covariance or precision matrix and is typically sparse. The energy (1) can be decomposed into a sum over local energy terms, or *factors*, over single pixels i and pairs of pixels (i, j).

2.1. Parameter Sharing

As proposed in [14], we also group the energy terms into factors of common unary type $u \in \mathcal{U}$ or pairwise type $p \in \mathcal{P}$ that share the same energy function E^u or E^p, but act on different variables and image contents. Thus (1) becomes

$$E(\boldsymbol{y}, \boldsymbol{x}, \mathcal{W}) = \sum_{u, i \in \mathcal{V}^u} E^u(\boldsymbol{y}_i, \boldsymbol{x}, \mathcal{W}) + \sum_{p, (i,j) \in \mathcal{E}^p} E^p(\boldsymbol{y}_{ij}, \boldsymbol{x}, \mathcal{W}),$$

[1]Here, $\langle\!\langle \cdot \rangle\!\rangle$ denotes Frobenius inner product, *i.e.* $\langle\!\langle \boldsymbol{y}\boldsymbol{y}^T, \boldsymbol{\Theta} \rangle\!\rangle = \boldsymbol{y}^T\boldsymbol{\Theta}\boldsymbol{y}$.



Ordinary regression tree
$(\boldsymbol{x}, i)$
$f(\boldsymbol{x}, i) < \varepsilon$
l^*
$\boldsymbol{\mu}_{l^*}$
Regression tree of factor type u
$(\boldsymbol{x}, i)$
l^*
$E^{u_{l^*}}(\boldsymbol{y}_i) = \frac{1}{2}\langle\!\langle \boldsymbol{y}_i\boldsymbol{y}_i^T, \boldsymbol{W}^{u_{l^*}} \rangle\!\rangle - \langle \boldsymbol{y}_i, \boldsymbol{w}^{u_{l^*}} \rangle$

Figure 2. Regression trees: (left) the prediction is determined by the path to leaf l^* storing sample mean $\boldsymbol{\mu}_{l^*}$; (right) instead of a mean, a quadratic energy is stored, determining a local model.

where $\mathcal{V}^u$ and $\mathcal{E}^p$ denote the sets of pixels i and pairs of pixels (i, j) covered by a unary factor of type u or a pairwise factor of type p, respectively. We instantiate the factors of a type in a repetitive manner relative to each pixel, specified in terms of offsets of the factor variables.

The local energy function E^u associated with a unary factor type is of the form

$$E^u(\boldsymbol{y}_i, \boldsymbol{x}, \mathcal{W}) = \tfrac{1}{2}\langle\!\langle \boldsymbol{y}_i\boldsymbol{y}_i^T, \boldsymbol{\Theta}_i^u(\boldsymbol{x}, \mathcal{W}) \rangle\!\rangle - \langle \boldsymbol{y}_i, \boldsymbol{\theta}_i^u(\boldsymbol{x}, \mathcal{W}) \rangle,$$

while a pairwise factor type p assigns $\boldsymbol{y}_{ij} \in \mathbb{R}^{2m}$ a similar energy E^p defined in terms of $\boldsymbol{\theta}_{ij}^p(\boldsymbol{x}, \mathcal{W})$ and $\boldsymbol{\Theta}_{ij}^p(\boldsymbol{x}, \mathcal{W})$.

The local coefficients $\{\boldsymbol{\theta}_i^u, \boldsymbol{\Theta}_i^u\}$ and $\{\boldsymbol{\theta}_{ij}^p, \boldsymbol{\Theta}_{ij}^p\}$ can depend on $\boldsymbol{x}$ in an almost arbitrary manner: the observed data determines the local Gaussian model that is in effect. The only requirement is that the global matrix $\boldsymbol{\Theta}(\boldsymbol{x}, \mathcal{W})$ stays positive-definite so (2) remains a valid distribution. This is trivially achieved by setting $\boldsymbol{\theta}_i^u(\boldsymbol{x}, \mathcal{W}) = \boldsymbol{w}^u \in \mathbb{R}^m$ and $\boldsymbol{\Theta}_i^u(\boldsymbol{x}, \mathcal{W}) = \boldsymbol{W}^u \in \mathbb{S}_{++}^m$ (and likewise for the pairwise terms), resulting in a set of model parameters $\mathcal{W} = \{\boldsymbol{w}^u, \boldsymbol{W}^u, \boldsymbol{w}^p, \boldsymbol{W}^p\}_{u \in \mathcal{U}, p \in \mathcal{P}}$. But such simple parametrization fails to exploit the full power of a conditional model.

2.2. Parametrization via Regression Trees

We now discuss how a valid *non-parametric map* from $\boldsymbol{x}$ to valid local models can be realized using regression trees. Regression trees are commonly employed as follows (see Figure 2): when inferring a prediction about label $y_i \in \mathbb{R}^m$ of pixel i from observations $\boldsymbol{x}$, one follows a path from the root of the tree to a leaf l^*. This path is determined by the branching decisions made at each node, typically by computing a feature score from the input image relative to the position of i and comparing it to a threshold. The label y_i is then chosen as the mean vector $\boldsymbol{\mu}_{l^*} \in \mathbb{R}^d$ of those training examples that previously ended up at the selected leaf l^*.

In our model, we use a similar approach to determine the parameterization of the unary local energy terms in a context-dependent manner. Instead of mean vectors, we associate with each leaf l the parameters of a quadratic energy $E^{u_l}(\boldsymbol{y}_i) = \frac{1}{2}\langle\!\langle \boldsymbol{y}_i\boldsymbol{y}_i^T, \boldsymbol{W}^{u_l} \rangle\!\rangle - \langle \boldsymbol{y}_i, \boldsymbol{w}^{u_l} \rangle$, with $\boldsymbol{W}^{u_l} \in \mathbb{S}_{++}^m$. In a standalone regression tree approach, the label could then

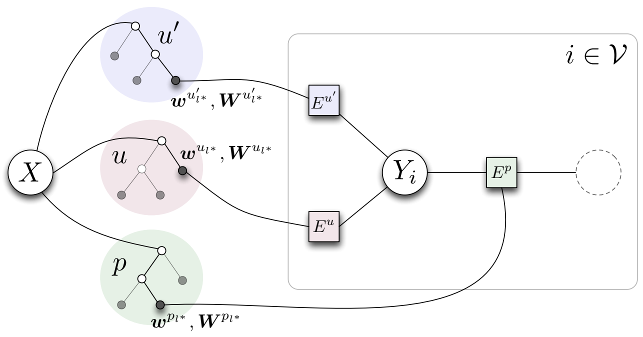

u'
$\boldsymbol{w}^{u'_{l^*}}, \boldsymbol{W}^{u'_{l^*}}$
$i \in \mathcal{V}$
$E^{u'}$
X
u
$\boldsymbol{w}^{u_{l^*}}, \boldsymbol{W}^{u_{l^*}}$
Y_i
E^p
E^u
p
$\boldsymbol{w}^{p_{l^*}}, \boldsymbol{W}^{p_{l^*}}$

Figure 3. Example of a regression tree field: regression trees (left) determine effective interactions u, u', and p, based on the image X, by selecting learned weights stored at their leaf nodes. The model structure on the right is replicated once for each pixel $i \in \mathcal{V}$.

be predicted as the minimizer of this local energy, which can be found in closed-form and is just the mean of the corresponding Gaussian. Instead, our goal here is for the selected leaf node to determine the parameterization of the local energy terms of our conditional random field, viz.:

$$\boldsymbol{\theta}_i^u(\cdot) = \boldsymbol{w}^{u_{l^*}} \text{ and } \boldsymbol{\Theta}_i^u(\cdot) = \boldsymbol{W}^{u_{l^*}}, \quad l^* = \text{Leaf}(u, i, \boldsymbol{x}).$$

Consequently, we associate with each unary factor type u a regression tree that determines the parameterization of the unary energy terms in the way outlined above.

The parameterization of the pairwise energy terms is determined in the same manner, *i.e.* we associate with each pairwise factor type p a regression tree defined over *pairs* of pixels (i, j) whose leaves store $2m$-dimensional models $E^{p_l}(\boldsymbol{y}_{ij}) = \frac{1}{2}\langle\!\langle \boldsymbol{y}_{ij}\boldsymbol{y}_{ij}^T, \boldsymbol{W}^{p_l}\rangle\!\rangle - \langle \boldsymbol{y}_{ij}, \boldsymbol{w}^{p_l}\rangle$ and proceed by defining $\boldsymbol{\theta}_{ij}^p(\boldsymbol{x}, \mathcal{W})$ and $\boldsymbol{\Theta}_{ij}^p(\boldsymbol{x}, \mathcal{W})$ to return the parameters of the selected leaf l^*. The full collection of model parameters is thus given by all parameters residing at the leaves of the regression trees, $\mathcal{W} = \{\boldsymbol{w}^{u_l}, \boldsymbol{W}^{u_l}, \boldsymbol{w}^{p_l}, \boldsymbol{W}^{p_l}\}$.

Summary. As illustrated in Figure 3, our model consists of several factor types, each of which is associated with a regression tree that stores at its leaves the parameters of a local quadratic energy. A factor type specifies how factors are instantiated relative to a given pixel. Factors of a common type share a local energy function that is parametrized via the quadratic models at the leaves of the associated tree. The image contents relative to the position of a factor determines the path from the root of the tree to the selected leaf, and hence selects the local Gaussian model that is in effect. The sum of local energy functions over the entire image determines the overall energy function.

2.3. Incorporating Linear Basis Functions

In principle, the non-parametric nature of regression trees allows us to learn arbitrary maps from input images $\boldsymbol{x} \in \mathcal{X}$ to labelings $\boldsymbol{y} \in \mathcal{Y}$. However, in many cases the mapping to the output is locally well approximated as a linear function of some derived image features. In this case using regression trees to represent this linear mapping is inefficient and requires a large number of nodes. Instead, we propose to use an arbitrary *linear model* in each leaf node using a set of application-dependent *basis* functions.

Such basis functions $\{\phi^b\}_{b\in\mathcal{B}}$ can be readily employed in our model, and can depend on $\boldsymbol{x}$ and the position within the image in an arbitrary manner. For instance, in the energy term E^u of a unary factor at pixel i, the linear term $\langle \boldsymbol{y}_i, \boldsymbol{w}^{u_{l^*}}\rangle$ can be replaced by $\sum_b \phi^b(i, \boldsymbol{x})\langle \boldsymbol{y}_i, \boldsymbol{w}^{u_{l^*};b}\rangle$. As a consequence, each leaf l of the regression tree stores not only a single parameter vector $\boldsymbol{w}^{u_l} \in \mathbb{R}^m$, but a collection of vectors $\{\boldsymbol{w}^{u_l;b}\}_{b\in\mathcal{B}}$, where again $\boldsymbol{w}^{u_l;b} \in \mathbb{R}^m$.

2.4. Efficient Inference

Because our global energy function is a quadratic form, the minimizing labeling can be found in closed form, *i.e.* $\boldsymbol{y}^* = [\boldsymbol{\Theta}(\boldsymbol{x}, \mathcal{W})]^{-1}\boldsymbol{\theta}(\boldsymbol{x}, \mathcal{W})$. This is also the mean of the associated Gaussian density and solves the linear system $\boldsymbol{\Theta}(\boldsymbol{x}, \mathcal{W})\boldsymbol{y} = \boldsymbol{\theta}(\boldsymbol{x}, \mathcal{W})$. The use of direct methods is prohibitive due to the large number of $m|\mathcal{V}|$ variables, hence we resort to iterative methods. We use the conjugate gradient (CG) method to obtain a solution to a residual norm of 10^{-4}. As we will discuss shortly, we can directly control the convergence behavior of CG by bounding the eigenvalues of the learned inverse covariance matrices.

3. Learning of Regression Tree Fields

We now discuss how to learn a model from given i.i.d. training data $\mathcal{D} = \{(\boldsymbol{x}^{(p)}, \boldsymbol{y}^{(p)})\}_{p=1}^P$. For simplicity, we treat the set of training examples as a single collection of labelled pixels $(\boldsymbol{x}, \boldsymbol{y})$, as in [20].

3.1. Estimating the Parameters

Assume for now that the structure of the regression trees and hence the collection of parameters $\mathcal{W}$ is fixed. We then wish to estimate these parameters from our training data $\mathcal{D}$. Ideally, we would be able to use the maximum likelihood estimate (MLE) of the parameters, because it is asymptotically consistent and has low asymptotic variance [10]:

$$\hat{\mathcal{W}}_{\text{MLE}} = \text{argmin}_{\mathcal{W}\in\Omega} - \log p(\boldsymbol{y} \mid \boldsymbol{x}; \mathcal{W}), \tag{5}$$

where constraint set Ω enforces positive-definiteness of the parameters $\{\boldsymbol{W}^{u_l}, \boldsymbol{W}^{p_l}\}$, the inverse covariance matrices of the local models. Unfortunately, to optimize (5), one requires the so-called *mean parameters*,

$$\boldsymbol{\mu} \overset{\text{def}}{=} \mathbb{E}_{\boldsymbol{y}\sim p(\boldsymbol{y}|\boldsymbol{x};\mathcal{W})}[\boldsymbol{y}] \text{ and } \boldsymbol{\Sigma} \overset{\text{def}}{=} \mathbb{E}_{\boldsymbol{y}\sim p(\boldsymbol{y}|\boldsymbol{x};\mathcal{W})}[\boldsymbol{y}\boldsymbol{y}^{\boldsymbol{T}}], \tag{6}$$

which are given in closed form as $\boldsymbol{\mu} = [\boldsymbol{\Theta}(\cdot)]^{-1}\boldsymbol{\theta}(\cdot)$ and $\boldsymbol{\Sigma} = [\boldsymbol{\Theta}(\cdot)]^{-1} + \boldsymbol{\mu}\boldsymbol{\mu}^T$. While polynomial-time, the complexity of this computation is $\mathcal{O}(m^3|\mathcal{V}|^3)$ and hence prohibitive even for instances of modest size.

$$-\log p(\boldsymbol{y}_i \mid \boldsymbol{y}_{\mathcal{V}\setminus i}, \boldsymbol{x}; \mathcal{W}) = E(\boldsymbol{y}_i, \boldsymbol{y}_{\mathcal{V}\setminus i}, \boldsymbol{x}, \mathcal{W}) + \log \int_{\mathbb{R}^m} \exp(-E(\hat{\boldsymbol{y}}_i, \boldsymbol{y}_{\mathcal{V}\setminus i}, \boldsymbol{x}, \mathcal{W}))\mathrm{d}\hat{\boldsymbol{y}}_i, \tag{3}$$

$$\nabla_{\mathcal{W}}[-\log p(\boldsymbol{y}_i \mid \boldsymbol{y}_{\mathcal{V}\setminus i}, \boldsymbol{x}; \mathcal{W})] = \nabla_{\mathcal{W}} E(\boldsymbol{y}_i, \boldsymbol{y}_{\mathcal{V}\setminus i}, \boldsymbol{x}, \mathcal{W}) - \mathbb{E}_{\hat{\boldsymbol{y}}_i \sim p(\hat{\boldsymbol{y}}_i \mid \boldsymbol{y}_{\mathcal{V}\setminus i}, \boldsymbol{x}, \mathcal{W})}\left[\nabla_{\mathcal{W}} E(\hat{\boldsymbol{y}}_i, \boldsymbol{y}_{\mathcal{V}\setminus i}, \boldsymbol{x}, \mathcal{W})\right]. \tag{4}$$

Figure 4. General form of the negative log-pseudolikelihood and the gradient with respect to $\mathcal{W}$ around a single conditioned subgraph.

3.2. Maximum Pseudolikelihood Estimation

We now show how the computational limitations of MLE can be overcome by maximizing the *pseudolikelihood* (MPLE) [2], defined as

$$\hat{\mathcal{W}}_{\text{MPLE}} = \operatorname{argmin}_{\mathcal{W}\in\Omega} - \textstyle\sum_{i\in\mathcal{V}} \log p(\boldsymbol{y}_i \mid \boldsymbol{y}_{\mathcal{V}\setminus i}, \boldsymbol{x}; \mathcal{W}). \tag{7}$$

Notably, the objective decomposes into likelihoods of single pixels conditioned on the observed labels of the other pixels,

$$p(\boldsymbol{y}_i \mid \boldsymbol{y}_{\mathcal{V}\setminus i}, \boldsymbol{x}, \mathcal{W}) = \frac{\exp(-E(\boldsymbol{y}_i, \boldsymbol{y}_{\mathcal{V}\setminus i}, \boldsymbol{x}, \mathcal{W}))}{\int_{\mathbb{R}^m} \exp(-E(\hat{\boldsymbol{y}}_i, \boldsymbol{y}_{\mathcal{V}\setminus i}, \boldsymbol{x}, \mathcal{W}))\mathrm{d}\hat{\boldsymbol{y}}_i}.$$

In our model, these *conditioned subgraphs* are just m-dimensional Gaussians, so we can again write the energy of a label $\boldsymbol{y}_i$ of a conditioned subgraph in canonical form,

$$E(\cdot) = \tfrac{1}{2}\langle\!\langle \boldsymbol{y}_i\boldsymbol{y}_i^T, \boldsymbol{\Theta}_i(\boldsymbol{x}, \mathcal{W})\rangle\!\rangle - \langle \boldsymbol{y}_i, \boldsymbol{\theta}_i(\boldsymbol{y}_{\mathcal{V}\setminus i}, \boldsymbol{x}, \mathcal{W})\rangle.$$

The canonical parameters $\boldsymbol{\theta}_i(\cdot) \in \mathbb{R}^m$ now depend on the labels $\boldsymbol{y}_{\mathcal{V}\setminus i}$ of the pixels on which the subgraph conditions. Unlike MLE, the inverse covariance matrices $\boldsymbol{\Theta}_i(\cdot) \in \mathbb{S}^m_{++}$ are low-dimensional, which renders the approach efficient. The corresponding mean parameters $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ are computed analogously to (6). Akin to MLE, they are needed to obtain the gradient with respect to $\boldsymbol{\theta}_i(\cdot)$ and $\boldsymbol{\Theta}_i(\cdot)$, from which we can derive the gradient with respect to the actual model parameters via the chain rule. We outline the general form of this gradient in Figure 4 and refer to the supplementary material for detailed analytically computable expressions. Note that (7) is *convex* and can be solved efficiently using projected gradient methods [3, 17], at a complexity of $\mathcal{O}(m^3|\mathcal{V}|)$ per computation of the gradient, which can be trivially parallelized over the set of pixels $\mathcal{V}$.

Furthermore, as first proposed in [14], pseudolikelihood estimation allows us to use a subsample of the training set. We resample a fraction of all the pixels in the training set, uniformly with replacement, to obtain an unbiased estimate of the pseudolikelihood objective (7).

3.3. Efficient Regularization

The RTF model is expressive but can overfit easily when deep regression trees are used. To counter this, we use a novel form of *regularization* for the matrix parameters to prevent overconfident predictions. We achieve this by lower- and upper-bounding all eigenvalues of $\{\boldsymbol{W}^{u_l}, \boldsymbol{W}^{p_l}\}$ to be no smaller than a tiny positive number $\underline{\varepsilon}$, and no larger than a large positive number $\overline{\varepsilon}$. The set of matrices that fulfil these constraints is again convex (see Figure 5).

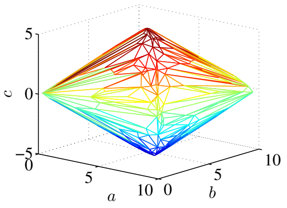


Figure 5. Convex set of 2×2 matrices $[a \quad c; c \quad b]$ whose eigenvalues are restricted to lie within $(\underline{\varepsilon} = 0.1, \overline{\varepsilon} = 10)$.

Through this restriction, we can enforce a favourable *condition number* of $\boldsymbol{\Theta}(\boldsymbol{x}, \mathcal{W})$, leading to fast convergence of the conjugate gradient method at test-time. Moreover, by adjusting $\overline{\varepsilon}$, we can push local models to be less certain of their mean, effectively regularizing the model. This can be understood as a flat prior over bounded-eigenvalue matrices, and because this set is bounded the prior is *proper* [10].

To ensure the matrices remain in this constrained set, we use a projection operator that builds on earlier results by Higham [9] and finds for any given matrix the closest matrix in Frobenius sense that satisfies our eigenvalue constraints. This is computationally efficient and requires one eigenvalue decomposition per matrix $\boldsymbol{W}^{u_l}$ or $\boldsymbol{W}^{p_l}$.

3.4. Growing Regression Trees

An important aspect that has been disregarded so far is how the structure of the regression trees can be learned. We propose two methods; the first approach trains regression trees separately, akin to [14], while the second approach learns model structure and parameters *jointly*.

Separate training is straightforward: For each factor type, we train a regression tree using the classic *reduction of variance* criterion [5]. Next, we associate a quadratic model with each leaf (*c.f*. Figure 2) and learn the parameters by means of the pseudolikelihood objective. This works for both unary and pairwise factors by regressing for the concatenated vectors in the pairwise case. While generally effective at learning the desired tree structure, one shortcoming is the *disconnect* between learning of the tree structure and subsequent estimation of the model parameters.

3.5. Learning Parameters and Trees Jointly

We next discuss how to choose the trees such as to optimize our learning objective. The idea is to choose splits that lead to the largest increase in the projected gradient norm $\|\mathcal{P}_\Omega(\nabla\mathcal{W}')\|$, where $\mathcal{W}' = (\mathcal{W} \setminus \mathcal{W}^{t_p}) \cup (\mathcal{W}^{t_l} \cup \mathcal{W}^{t_r})$ denotes the model parameters after the split, with the parameters $\mathcal{W}^{t_l}$ and $\mathcal{W}^{t_r}$ of the newly introduced children l and r initialized to the previous parameters $\mathcal{W}^{t_p}$ of the leaf p that was split. Here, t refers to either a unary or a pairwise type.

The gradient norm with respect to model parameters $\mathcal{W}^{t_l} = \{\boldsymbol{w}^{t_l}, \boldsymbol{W}^{t_l}\}$ of a given leaf l can be thought of

```
Start with trees consisting solely of root nodes;
repeat
    (Re-)optimize parameters of current leaf nodes ;
    foreach conditioned subgraph $i$ do
        Pre-compute mean parameters $\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i$ ;
    foreach factor type $t$ and its tree do
        foreach conditioned subgraph $i$ do
            foreach factor of matching type do
                Compute gradient contribution via $\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i$ ;
                Sort contribution into target leaf ;
        foreach leaf $p$ do
            Find split $(f, \varepsilon)$ maximizing $\|\mathcal{P}_\Omega(\nabla \mathcal{W}')\|$ ;
            Split node $p$ into new child leaves $(l, r)$ ;
            Set $\mathcal{W}^{t_l} \leftarrow \mathcal{W}^{t_p}$ and $\mathcal{W}^{t_r} \leftarrow \mathcal{W}^{t_p}$ ;
until maximum depth reached;
Optimize parameters of leaf nodes to final accuracy ;
```

Algorithm 1. Joint training of trees and parameters: See text.

as a measure of disagreement between the mean parameters $\{\boldsymbol{\mu}_i(\boldsymbol{x}, \mathcal{W}), \boldsymbol{\Sigma}_i(\boldsymbol{x}, \mathcal{W})\}$ and the empirical distribution of $\{\boldsymbol{y}_i, \boldsymbol{y}_i\boldsymbol{y}_i^T\}$ in the conditioned subgraphs affected by the leaf. Consequently, our criterion prefers splits introducing new parameters relevant to those subgraphs where the disagreement is largest, as these are most likely to achieve significant gains in terms of the pseudolikelihood.

Algorithm 1 gives an outline of how this works. The key to tractability is that the increase in gradient norm is computed for the parameters of the candidate child nodes set to those of their parent node. This way, the increase in overall gradient norm can be computed efficiently and purely locally in terms of the norms $\|\mathcal{P}_\Omega(\nabla \mathcal{W}^{t_l})\|$ and $\|\mathcal{P}_\Omega(\nabla \mathcal{W}^{t_r})\|$ resulting from the gradient contributions of the factors that are relevant to the respective candidate child.

By initializing the parameters of the new leaf nodes to those of their parent, the algorithm achieves monotonic descent in the negative log-pseudolikelihood. This holds even if re-optimization of the parameters at each round is approximate, which is often preferable from an efficiency perspective. While Algorithm 1 outlines joint training for the pseudolikelihood objective, we emphasize that this idea is in principle applicable to most tractable objective functions.

Preliminary Assessment. We next provide preliminary confirmation of the usefulness of joint training. Figure 6 shows the performance of the same RTF denoising model with $\sigma = 10$ (Section 4.2) for separate and joint training. Joint training optimizes the learning objective more effectively as a function of the tree depth, producing in this case more accurate predictions in terms of the error measure (PSNR).[2] Since joint training is slower by a factor of 3–5, we use separate training for the remaining experiments.

[2]For other noise levels, joint training always optimized the learning objective better, which, however, did not always improve PSNR.

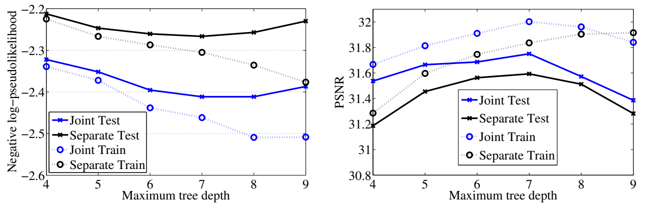


Figure 6. Joint training reduces the objective faster than separate training (left), which translates into improved PSNR (right). We report training set and test set performances for both approaches.

4. Experiments

4.1. Discrete Learning Tasks

In order to demonstrate that our model is conveniently applicable to discrete labeling tasks, we first consider two tasks that were previously tackled using DTFs [14].

Chinese Characters. The goal is to in-paint the occluded parts of handwritten Chinese characters from the KAIST Hanja2 database (Figure 7). Each character is occluded by a centred grey box of varying size. Following [14], we measure prediction accuracy on a dataset with small occlusions, and visualize the predictions on images with larger occlusions. We replicate the DTF model as closely as possible (same features and neighborhood). For RTF training, we consider 2D orthonormal basis encoding $\{[1\ 0]^T, [0\ 1]^T\}$, as well as plain 1D encoding of the binary labels. We consider a Gaussian MRF where the pairwise trees are restricted to a single leaf (GMRF) and systems with deep pairwise trees (RTF), analogous to MRF and DTF in [14].

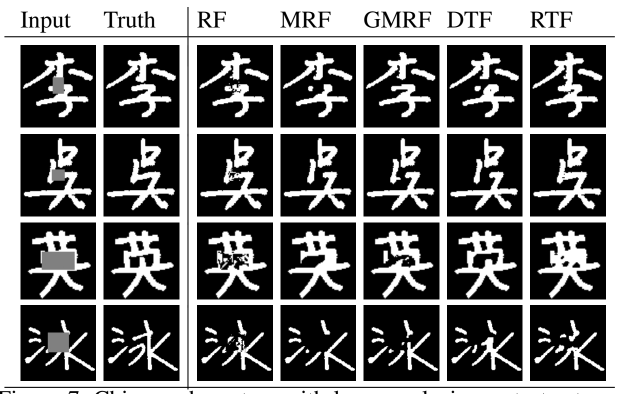


Figure 7. Chinese characters with large occlusions—test set predictions. Characters of the last 2 lines also shown in [14, Fig. 7].

The results are shown in Table 1. We include the Random Forest (RF) result of [14] as a baseline. Our 2D-encoded systems are very competitive, with a particular RTF system achieving the best result on this task so far. Moreover, the best RTF system requires typically 0.2s per prediction, which is two orders of magnitude faster than the current DTF implementation [14, private communication with authors]. The DTF predictions were obtained using simulated annealing and therefore may not be optimal,

	Depth$_u$	Depth$_p$	Test	Train
RF [14]	15	~	67.74%	~
MRF [14]	15	1	75.18%/≈20s	~
GMRF 1D	15	1	70.14%/0.19s	73.11%
GMRF 2D	15	1	74.19%/0.32s	80.97%
DTF [14]	15	6	76.01%/≈20s	~
RTF 1D	15	6	75.37%/0.27s	79.38%
RTF 2D	15	6	75.02%/0.49s	81.73%
RTF 1D*	20	20	76.39%/0.23s	94.56%
RTF 2D*	20	20	**77.55%**/0.24s	**94.91%**

Table 1. Chinese characters—accuracy on small occlusions.

	Depth$_u$	Depth$_p$	Accuracy	RMSE
RF [14]	25	~	90.30%	~
MRF [14]	25	1	91.90%	~
GMRF 1D	36	1	82.52%	0.0999
GMRF 11D	36	1	84.22%	0.1352
DTF [14]	25	15	**99.40%**	~
RTF 1D	0	10	91.14%	0.0512
RTF 11D	0	7	98.77%	**0.0268**

Table 2. Results on the "Snakes" test data, 4-connected model.

whereas inference in the RTF model is always exact. Note that 2D encoding is particularly important for GMRF, where the restricted pairwise terms in 1D encoding cannot be compensated for by conditioning. If deeper pairwise trees are allowed, as in RTF, this difference mostly vanishes. For further details and results, we refer to the sup. material.

Snakes. This is a multi-label discrete learning task with weak local evidence for any particular label; the ability of the pairwise terms to capture the relevant interactions is crucial. Each "snake" (Figure 8) consists of a sequence of adjacent pixels whose color in the input encodes the direction of the next pixel: *go north* (red), *go south* (green), as well as *go east* (yellow) and *go west* (blue). Each snake is 10 pixels long, and in output space exposes a grey-scale gradient that starts at its head in black and ends at its head in white.

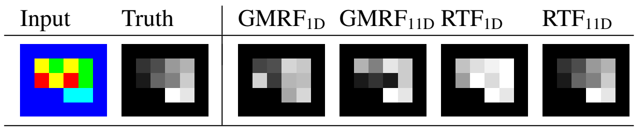


Figure 8. "Snakes" task—1D encoding seeks to minimize RMSE; 11D encoding injects a loss that is closer to multi-label error.

Again, we use the systems from the DTF paper [14] as our baseline. For RTF-training, we compare 1D encoding, which directly models the grey-scale pixel intensity, to 11D encoding that assigns an orthonormal basis label to each of the 11 different grey-scale values. The latter "injects" a particular loss function during training: Since all labels are equally close in Euclidean space, we attain invariance with respect to label permutations, and MPLE minimizes a quadratic approximation of the discrete multi-label error. In contrast, in 1D grey-scale encoding, MPLE minimizes a quadratic loss that is closely correlated with RMSE. The

Method	σ = 20	σ = 30	σ = 40
FoE MAP 3x3 [15]	25.83/41s	22.66/42s	20.10/42s
FoE MAP 5x5 [15]	27.59/170s	25.13/150s	23.59/149s
FoE MMSE 3x3 [18]	28.20/239s	26.01/230s	24.48/321s
FoE MMSE PW [18]	27.69/49s	25.71/67s	24.29/74s
BM3D [7]	**28.37**/0.07s	**26.31**/0.07s	**24.90**/0.07s
RTF 3x3	27.73/0.3s	25.67/0.3s	24.30/0.3s

Table 3. Natural image denoising results in PSNR (peak signal-to-noise ratio) with test-time running time per image. The FoE results were obtained using the models of [15, 18].

regressed label of a pixel is decoded as follows: For 1D encoding, RMSE can be computed directly from the prediction, while multi-label error is computed by rounding to the nearest discrete label. With 11D, we find the basis vector closest to the prediction and use the corresponding grey-scale value (RMSE) or discrete label (multi-label error).

The numeric results are given in Table 2, and example predictions are shown in Figure 8. Tree depths were optimized for each system. RTF using 11D encoding and DTF essentially solve the task, while all other systems fail. Consider the error rates achieved by GMRF: 11D encoding leads to smaller multi-label error, while 1D encoding favours RMSE. On the other hand, using the fully conditional pairwise terms of the RTF, 11D encoding yields better results in terms of both error metrics. This result suggests that high-dimensional encodings yield additional benefits even beyond the above loss function perspective.

4.2. Natural Image Denoising

Natural image denoising is a classic benchmark for continuous image labeling. Our model is not specifically constructed to be good at denoising, but we show that it indeed performs well against reasonable baseline approaches. We use the BSDS 500 dataset [1] and use 200 training and 200 test grayscale images scaled by a factor of 0.25. We use additive, iid Gaussian noise with known standard deviation σ. For our model we use the RFS filterbank[3] to derive 38 filter responses for each pixel in the input image which are used for the regression tree splits as well as the basis functions of the linear leaf models. We use a subsampling factor of 0.5.

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

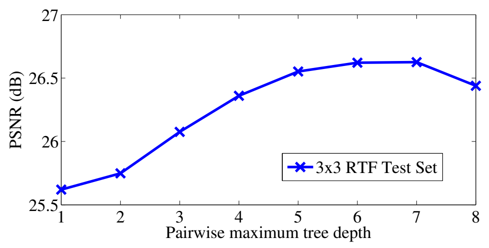


Figure 9. Conditional pairwise interactions increase denoising PSNR (σ = 25): we vary the maximum depth of pairwise regression trees from one (MRF prior) to eight. This increases PSNR on the test set from 25.62dB (depth one) to 26.63dB (depth seven).

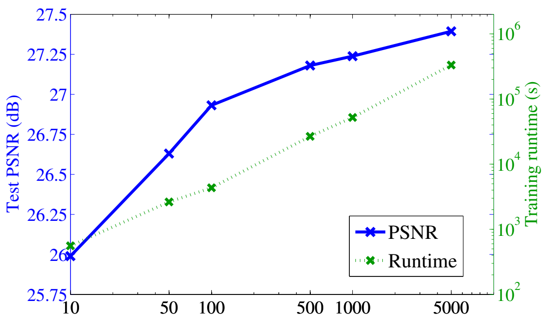


Figure 10. Training efficiency on a single computer (8 cores). We learn a 3x3 denoising RTF model for a noise level of $\sigma = 25$ from the MIRFLICKR dataset. We test on 5000 holdout images. Training time is linear in the number of images (note logarithmic axes). Test PSNR continues to increase as more training data is used.

Quantitative results for three different noise levels[4] are shown in Table 3. We outperform the FoE MAP approaches, but FoE MMSE 3x3 and BM3D perform better than our model. This may be due to the simple features which we have used. The RTF model is efficient at test-time, and achieves its predictive performance due to the pairwise conditional interactions, as shown in Figure 9. The model of pairwise tree depth one corresponds to a simple parametric Gaussian CRF [21].

Large-scale Training. To evaluate the scalability of our training procedure, we repeat the denoising experiment on the MIRFLICKR-25000 dataset [11], consisting of 25,000 natural images. We use subsets of up to 5,000 images for training. The results are shown in Figure 10. They demonstrate that our approach scales to a large number of images.

Structured Noise Model. One advantage of using our conditional model is that we can incorporate higher-order interactions such as image filters between the observations and dependent variables *without incurring additional computation costs during learning or inference*. This allows us to *learn the noise model*, instead of assuming pixelwise independent noise [15, 18]. To demonstrate that our model can learn the noise model, we simulate images with dust on the lens, as shown in Figure 11. The RTF denoising model is as before and we learn to remove these artifacts.

4.3. Face Colorization

Colorization is the task of adding color to a gray-scale image, *e.g.* an old photograph. In most works, *e.g.* [13], this under-constrained task is solved with some user guidance. Here we demonstrate a fully automatic system, which exploits domain knowledge. Given a training set of 200 frontal faces and a test set of 200 different people[5], where the face images are roughly registered (see sup. material). Given the gray-scale input, the goal is to predict the 3D (RGB) output. As features we use Haar-wavelets of size 1 to 32 pixels and various relative offsets (Gaussian-distributed with standard deviation of 10 pixels). Figure 12 shows that our result (f) is visually superior to various competitors (c-e). The improvement is also observed in the overall mean squared error (multiplied by 10^3), where (f) achieves 0.47, and the other methods obtain: (c) 0.73; (d) 0.78; (e) 0.82.

[4]Note that for each image, the Gaussian noise was first added, and the result was then saved as a PNG file again, possibly resulting in truncation.

[5]Available from http://fei.edu.br/˜cet/facedatabase.html

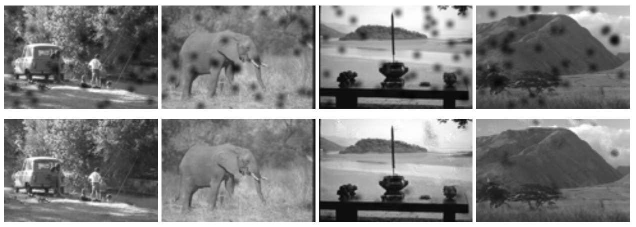


Figure 11. Representative test set results for image denoising with structured noise. Top row: input images with simulated dust on the camera lens. Bottom row: RTF denoising results with a 3x3 model and decision trees of depth ten (PSNR 27.85).

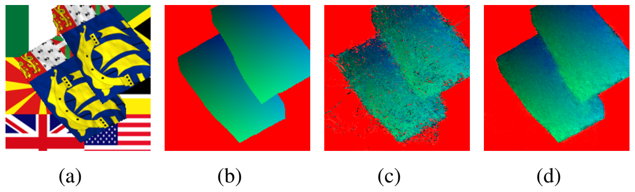


Figure 13. Detection and registration—from left to right: input image, ground truth (RGB), unary prediction, RTF 3x3 prediction.

4.4. Detection and Registration

In this task we jointly detect and register deformable objects within an image. The input, Fig. 13(a), are two flags with variable position and deformation.[6] The background is an arbitrary crop from a large mosaic of flags. The output labeling, see Fig. 13(b) is a 3D (RGB) labeling where the first channel defines fore- and background and the last two represent the mapping of each pixel to a reference frame of the flat flag. We use 400 generated training images and 100 test images, and a 3x3 RTF model with maximum tree depth 50 for all trees. Figure 13(c,d) shows that the field performs much better than unaries. This reflects in the mean squared error; $6.1 \cdot 10^{-2}$ for unary, $1.0 \cdot 10^{-2}$ for the RTF.

5. Conclusion

We presented regression tree fields, an efficient non-parametric random field model. We have demonstrated the flexibility of our model by applying it to a number of discrete and continuous image labeling tasks, obtaining high-quality results. This flexibility, together with our efficient learning and inference procedures, make RTFs attractive for a broad number of computer vision applications.

[6]We use the 60 deformations provided by [8]

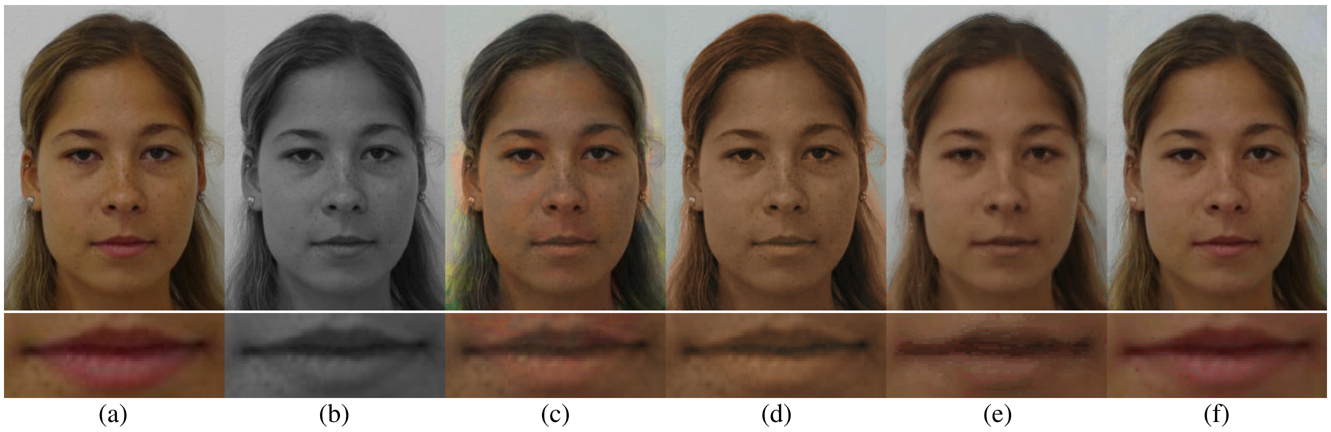


Figure 12. **Face colorization** (top row: full images, bottom row: zoom-in). Given a gray-scale test image **(b)**, the goal is to recover its color. **(a)** The ground truth. **(c)** A simple, "global-average" competitor. First, 10 nearest-neighbor images are retrieved from the training set, in terms of pixel-wise gray-scale difference. Then these images are superimposed and the median color (hue, saturation) is computed at every pixel location. Since the nearest-neighbor faces are not perfectly registered, color bleeding (e.g. around left ear) can be observed. **(d)** A second competitor, which uses the same 10 nearest-neighbors as (b). For each luminance value, the median color (hue, saturation) is derived. The result does not show color-bleeding, but suffers from the fact that the whole face and hair has virtually the same color. **(e)** Our result with unaries only (one tree, depth ten). While the overall result is encouraging the details are unfortunately blurry (see zoom-in). This is likely caused by the fact that neighboring pixels make independent decisions. **(f)** Our result with field (4-connectivity, one unary tree, two pairwise trees, all depth 10, separately trained). The overall result, as well as the zoom-in, looks very convincing.

References

[1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell*, 33(5), 2011. 6

[2] J. Besag. Efficiency of pseudolikelihood estimation for simple Gaussian fields. *Biometrica*, (64):616–618, 1977. 4

[3] E. G. Birgin, J. M. Martinez, and M. Raydan. Nonmonotone spectral projected gradient methods on convex sets. *SIAM Journal of Optimization*, 10(4):1196–1211, 2000. 4

[4] A. Blake, P. Kohli, and C. Rother. Markov random fields for vision and image processing. MIT Press, 2011. 1

[5] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth Publishing Company, Belmont, California, U.S.A., 1984. 4

[6] D. Cremers, T. Pock, K. Kolev, and A. Chambolle. Convex relaxation techniques for segmentation, stereo and multiview reconstruction. In *Advances in Markov Random Fields for Vision and Image Processing*. MIT Press, 2011. 2

[7] K. Dabov, A. Foi, V. Katkovnik, and K. O. Egiazarian. Image denoising by sparse 3-D transform-domain collaborative filtering. *IEEE Transactions on Image Processing*, 16(8):2080–2095, 2007. 6

[8] R. Garg, A. Roussos, and L. Agapito. Robust trajectory-space TV-L1 optical flow for non-rigid sequences. In *EMMCVPR*, 2011. 7

[9] N. J. Higham. Computing a nearest symmetric positive semidefinite matrix. *Linear algebra and its applications*, 103, 1988. 4

[10] R. V. Hogg, J. W. McKean, and A. T. Craig. *Introduction to Mathematical Statistics*. Pearson Education, 2005. 3, 4

[11] M. J. Huiskes and M. S. Lew. The MIR Flickr retrieval evaluation. In *MIR 2008*. ACM, 2008. 7

[12] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009. 1

[13] A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. *ACM Trans. Graph*, 23(3):689–694, 2004. 7

[14] S. Nowozin, C. Rother, S. Bagon, T. Sharp, B. Yao, and P. Kohli. Decision tree fields. In *ICCV*, 2011. 1, 2, 4, 5, 6

[15] S. Roth and M. J. Black. Fields of experts. *International Journal of Computer Vision*, 82(2):205–229, 2009. 2, 6, 7

[16] K. G. G. Samuel and M. F. Tappen. Learning optimized MAP estimates in continuously-valued MRF models. In *CVPR*, 2009. 1, 2

[17] M. Schmidt, E. van den Berg, M. Friedlander, and K. Murphy. Optimizing costly functions with simple constraints: A limited-memory projected quasi-Newton algorithm. In *AISTATS*, 2009. 4

[18] U. Schmidt, Q. Gao, and S. Roth. A generative perspective on MRFs in low-level vision. In *CVPR*, 2010. 2, 6, 7

[19] D. Singaraju, L. Grady, A. K. Sinop, and R. Vidal. Continuous valued MRFs for image segmentation. In A. Blake, P. Kohli, and C. Rother, editors, *Markov Random Fields for Vision and Image Processing*. MIT Press, 2011. 2

[20] C. Sutton and A. McCallum. An introduction to conditional random fields for relational learning. In *Introduction to Statistical Relational Learning*, chapter 4. 2007. 3

[21] M. Tappen, C. Liu, E. H. Adelson, and W. T. Freeman. Learning Gaussian conditional random fields for low-level vision. In *CVPR*, 2007. 1, 2, 7

[22] M. F. Tappen, K. G. G. Samuel, C. V. Dean, and D. Lyle. The logistic random field – a convenient graphical model for learning parameters for MRF-based labeling. In *CVPR*, 2008. 1

[23] B. Taskar, V. Chatalbashev, and D. Koller. Learning associative markov networks. In *ICML*, 2004. 1