Supplementary Material: 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

This supplementary document contains details that were omitted from the main paper due to a lack of space. These are in particular:

1. *Further notes regarding the expressive power of the RTF model;*
2. *A procedural guide detailing how to perform inference in the RTF model;*
3. *Analytic expressions for computation of the negative log-pseudolikelihood objective function and the gradient with respect to the model parameters, as well as a proof of convexity of the former;*
4. *Further result images.*

1. Details on: Expressive Power of RTF Model (Sec. 1)

In the introduction of the main paper, we mention that in discrete labeling tasks, a high-dimensional mapping of the discrete labels into continuous space increases the expressive power of the RTF model. This effect is empirically demonstrated in Section 4; here, we provide an additional perspective on the matter.

Consider a learning task involving m discrete labels. In our model, we encode these discrete labels using m orthonormal basis vectors (*e.g.* $[1,0,0]^T, [0,1,0]^T, [0,0,1]^T$ for $m = 3$). The energy of a pairwise term assumes the form

$$E^p(\boldsymbol{y}_{ij}, \boldsymbol{x}, \mathcal{W}) = \frac{1}{2}\langle\!\langle \boldsymbol{y}_{ij}\boldsymbol{y}_{ij}^T, \boldsymbol{\Theta}_{ij}(\cdot)\rangle\!\rangle - \langle \boldsymbol{y}_{ij}, \boldsymbol{\theta}_{ij}(\cdot)\rangle,$$

where $\boldsymbol{y}_{ij} = [\boldsymbol{y}_i, \boldsymbol{y}_j]^T \in \mathbb{R}^{2m}$ is the vector of stacked pixel labels. In contrast, in a discrete model, the energy of a particular pairwise labeling $(\boldsymbol{y}_i, \boldsymbol{y}_j)$ is determined by a $m \times m$ table that assigns each label configuration a particular energy. Using the above m-dimensional orthonormal basis encoding, it is always possible to choose the coefficients $\boldsymbol{\Theta} \in \mathbb{S}^{2m}_{++}$ and $\boldsymbol{\theta} \in \mathbb{R}^{2m}$ such that each continuously encoded discrete label indeed receives precisely the energy assigned by *any* discrete $m \times m$ energy table. Using an additional constant bias term in the quadratic form, the same property can be achieved by encoding the m discrete labels via $m-1$ orthonormal basis vectors and a single $\mathbf{0}$ vector.[1] Even repulsive energy tables can be "fitted" this way. This is illustrated by Figure 1 for the special case of $m = 2$, *i.e.* binary labels. Note that such fitting of energy tables is not in general possible using lower-dimensional encodings. To verify this, we tried to fit the coefficients of a quadratic form (including a bias term) to random 3×3 energy tables (*i.e.* $m = 3$), where each entry was chosen uniformly at random from $(0, 5)$ at each run, and the coefficients of the quadratic form were chosen such as to minimize the sum of squared errors of the assigned energies (subject to the positive-definite constraint on the quadratic coefficients). The experiment was repeated 10,000 times, using a one-dimensional encoding of the three discrete labels as $\{0, \frac{1}{2}, 1\}$, and a two-dimensional encoding $\{[1,0]^T, [0,1]^T, [0,0]^T\}$. Using the latter, it was *always* possible to achieve a residual of zero, whereas using the former, the residual was *never* zero, with a mean residual of 8.88 and a variance of 27.67.

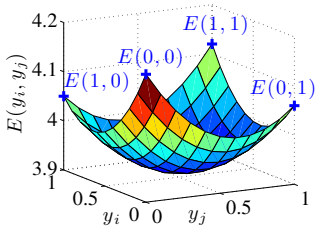


Figure 1. Quadratic fit of a repulsive pairwise discrete energy table assigning $E(0,0) = 4.16, E(0,1) = 4.06, E(1,0) = 4.12$ and $E(1,1) = 4.05$.

However, while the ability to match energy values at any given set of points is a necessary condition for accurate modeling of the underlying world distribution, this property alone is not sufficient. For instance, there may be a large probability mass far away from any discrete labeling (*c.f.* Figure 1). We leave the question of formalizing this trade-off for further study.

2. Details on: Efficient Inference (Sec. 2.4)

One of the immediate benefits of restricting the global energy $E(\boldsymbol{y}, \boldsymbol{x}, \mathcal{W})$ to be quadratic is that the labelling that minimizes this energy can be readily found in closed form, $\boldsymbol{y}^* = [\boldsymbol{\Theta}(\boldsymbol{x}, \mathcal{W})]^{-1}\boldsymbol{\theta}(\boldsymbol{x}, \mathcal{W})$. Incidentally, this is precisely the mean of the associated Gaussian density, as well as

[1]In our actual model, we always use m-dimensional basis vectors (rather than $m-1$) because the quadratic forms do not include a bias term.

the solution of the linear system $\boldsymbol{\Theta}(\boldsymbol{x}, \mathcal{W})\boldsymbol{y} = \boldsymbol{\theta}(\boldsymbol{x}, \mathcal{W})$. Direct linear algebra methods that solve this problem are readily available. However, their asymptotic complexity of $\mathcal{O}(m^3|\mathcal{V}|^3)$ is typically prohibitive.

Several efficient iterative methods are available that can exploit the sparsity of $\boldsymbol{\Theta}(\boldsymbol{x}, \mathcal{W})$, *e.g.* Gaussian belief propagation [4] and the conjugate gradient (CG) method. We find CG particularly convenient because it is guaranteed to converge and does not require line search. Procedurally, this translates into the following steps at test time:

- For each factor, find the selected leaf node of the regression tree associated with its type u or p, and precompute the linear basis functions, if any.
- Add up the contributions of the local terms (Figure 4) in order to instantiate the right-hand side $\boldsymbol{\theta}(\boldsymbol{x}, \mathcal{W})$.
- Solve the linear system using conjugate gradient, which determines our prediction $\boldsymbol{y}^*$.

The final step only requires sparse matrix-vector products of the kind $\boldsymbol{\Theta}(\boldsymbol{x}, \mathcal{W})\boldsymbol{y}$, which can be computed on-the-fly without ever actually instantiating $\boldsymbol{\Theta}(\boldsymbol{x}, \mathcal{W})$. This operation is summarized in Figure 4. An attractive alternative is to pre-compute the matrix, since for typical neighbourhood structures, $\boldsymbol{\Theta}(\boldsymbol{x}, \mathcal{W})$ exposes a banded sparsity pattern that allows for storage in DIA format, which is particularly efficient for CG implementation on a GPU [1].

3. Details on: Negative Log-Pseudolikelihood (Sec. 3.2)

We next demonstrate that the maximum pseudolikelihood estimation problem is convex, and point out how to compute the objective and the gradient with respect to the model parameters analytically.

Convexity of the estimation problem. Recall that the energy $E(\boldsymbol{y}_i, \boldsymbol{y}_{\mathcal{V}\setminus i}, \boldsymbol{x}, \mathcal{W})$ of labeling $\boldsymbol{y}_i$ of a conditioned subgraph around pixel i assumes the form

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

This function is linear in $\mathcal{W}$, and hence convex. Consider next the logarithm of the partition function normalizing

$$p(\boldsymbol{y}_i \mid \boldsymbol{y}_{\mathcal{V}\setminus i}, \boldsymbol{x}; \mathcal{W}) \propto \exp(-E(\boldsymbol{y}_i, \boldsymbol{y}_{\mathcal{V}\setminus i}, \boldsymbol{x}, \mathcal{W})),$$

defined as

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

Convexity of this function in the model parameters can most easily be seen from its variational representation [3]:

$$\begin{aligned}\sup_{(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)} \Big\{ &-\tfrac{1}{2}\langle\!\langle \boldsymbol{\mu}_i\boldsymbol{\mu}_i^T, \boldsymbol{\Theta}_i(\boldsymbol{x}, \mathcal{W})\rangle\!\rangle \\ &+ \langle \boldsymbol{\mu}_i, \boldsymbol{\theta}_i(\boldsymbol{y}_{\mathcal{V}\setminus i}, \boldsymbol{x}, \mathcal{W})\rangle + H(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\Big\}, \quad (10)\\ \text{sb.t.} \quad &\boldsymbol{\Sigma}_j - \boldsymbol{\mu}_j\boldsymbol{\mu}_j^T \succ 0.\end{aligned}$$

The objective is linear in $\mathcal{W}$, and the problem attains a unique optimum at $\boldsymbol{\mu}_i^* = [\boldsymbol{\Theta}_i(\boldsymbol{x}, \mathcal{W})]^{-1}\boldsymbol{\theta}_i(\boldsymbol{y}_{\mathcal{V}\setminus i}, \boldsymbol{x}, \mathcal{W})$ and $\boldsymbol{\Sigma}_i^* = [\boldsymbol{\Theta}_i(\boldsymbol{x}, \mathcal{W})]^{-1} + \boldsymbol{\mu}_i\boldsymbol{\mu}_i^T$, so by Danskin's theorem [2, Proposition B.25], $A(\cdot)$ is convex in $\mathcal{W}$. Together with linearity of the energy and convexity of constraint set Ω, this establishes convexity of the overall problem. □

Computation of objective and gradient. First of all, it is necessary to understand how the canonical parameters $\boldsymbol{\theta}_i(\boldsymbol{y}_{\mathcal{V}\setminus i}, \boldsymbol{x}, \mathcal{W})$ and $\boldsymbol{\Theta}_i(\boldsymbol{x}, \mathcal{W})$ of a conditioned subgraph around pixel i come into existence. In writing the energy $E(\boldsymbol{y}_i, \boldsymbol{y}_{\mathcal{V}\setminus i}, \boldsymbol{x}, \mathcal{W})$ in canonical form, we need to ensure that—modulo constants that do not involve $\boldsymbol{y}_i$—it contains the same terms that are also present in the global energy $E(\boldsymbol{y}, \boldsymbol{x}, \mathcal{W})$. Observe that this is the case for the expressions given in Figure 2. Using these definitions, one can then compute the objective and the gradient analytically, as shown in Figure 3. The gradient can most easily be derived by again writing the log partition function in its variational representation (10) and invoking Danskin's theorem.

4. Further Experimental Results (Sec. 4)

We show some additional test set predictions on the structured noise denoising task in Figure 5; for the regular pixel-independent denoising task, we show in Figure 6 some further test set results for all methods we evaluated. Further face colorization results are shown in Figure 7.

Finally, exemplary training data and test predictions of the detection and registration task are given in Figures 8–9. Note that the goal of this task is to demonstrate the wide range of applications for our model. In particular this task is a mix of continuous and discrete variables. There are many alternatives methods to address this problem, such as wide-baseline optical flow paired with binary segmentation.

References

[1] N. Bell and M. Garland. Efficient sparse matrix-vector multiplication on CUDA. NVIDIA Technical Report NVR-2008-004, NVIDIA Corporation, Dec. 2008. 2

[2] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, second edition, 1999. 2

[3] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1–2):1–305, 2008. 2

[4] Y. Weiss and W. T. Freeman. Correctness of belief propagation in gaussian graphical models of arbitrary topology. *Neural Computation*, 13(10):2173–2200, 2001. 2

Remark regarding the notation in the below figures.
We use the bracket notation $[\cdot]_j$ to denote the $m \times 1$ block of a stacked vector corresponding to pixel j, and $[\cdot]_{ij}$ to denote the $m \times m$ block of a $2m \times 2m$ matrix involving the rows of pixel i and the columns of pixel j. For instance, in $[\boldsymbol{\Theta}^p_{ij}(\boldsymbol{x}, \mathcal{W})]_{ji}$, the $m \times m$ block in the lower left corner is selected (the top rows of the matrix correspond to pixel i, while the bottom half corresponds to j; similarly, the columns on the left concern pixel i, while the right half concerns pixel j).

$$\boldsymbol{\theta}_j(\boldsymbol{y}_{\mathcal{V}\setminus j}, \boldsymbol{x}, \mathcal{W}) = \sum_u \delta_{j\in\mathcal{V}^u} \boldsymbol{\theta}^u_j(\boldsymbol{x}, \boldsymbol{w}) + \sum_{p,(i,j)\in\mathcal{E}^p} \Big\{ [\boldsymbol{\theta}^p_{ij}(\boldsymbol{x}, \mathcal{W})]_j - \tfrac{1}{2}\big([\boldsymbol{\Theta}^p_{ij}(\boldsymbol{x}, \mathcal{W})]^T_{ij} + [\boldsymbol{\Theta}^p_{ij}(\boldsymbol{x}, \mathcal{W})]_{ji}\big)\boldsymbol{y}_i \Big\} + \sum_{p,(j,k)\in\mathcal{E}^p} \Big\{ [\boldsymbol{\theta}^p_{jk}(\boldsymbol{x}, \mathcal{W})]_j - \tfrac{1}{2}\big([\boldsymbol{\Theta}^p_{jk}(\boldsymbol{x}, \mathcal{W})]^T_{kj} + [\boldsymbol{\Theta}^p_{jk}(\boldsymbol{x}, \mathcal{W})]_{jk}\big)\boldsymbol{y}_k \Big\} \tag{1}$$

$$\boldsymbol{\Theta}_j(\boldsymbol{x}, \mathcal{W}) = \sum_u \delta_{j\in\mathcal{V}^u} \boldsymbol{\Theta}^u_j(\boldsymbol{x}, \mathcal{W}) + \sum_{p,(i,j)\in\mathcal{E}^p} [\boldsymbol{\Theta}^p_{ij}(\boldsymbol{x}, \mathcal{W})]_{jj} + \sum_{p,(j,k)\in\mathcal{E}^p} [\boldsymbol{\Theta}^p_{jk}(\boldsymbol{x}, \mathcal{W})]_{jj} \tag{2}$$

Figure 2. Explicit expressions for the canonical parameters of a conditioned subgraph centered around j. Recall that $\boldsymbol{\theta}^u_j(\boldsymbol{x}, \mathcal{W})$ selects model parameter $\boldsymbol{w}^{u_{l^*}}$ via $l^* = \text{Leaf}(u, i, \boldsymbol{x})$, and the other local terms are defined likewise.

$$\ell_j(\mathcal{W}) = \langle \boldsymbol{\theta}_j(\boldsymbol{y}_{\mathcal{V}\setminus j}, \boldsymbol{x}, \mathcal{W}), \boldsymbol{\mu}_j - \boldsymbol{y}_j \rangle + \frac{1}{2}\left(\langle\!\langle \boldsymbol{\Theta}_j(\boldsymbol{x}, \mathcal{W}), \boldsymbol{y}_j\boldsymbol{y}_j^T - \boldsymbol{\Sigma}_j \rangle\!\rangle + \log\det(\boldsymbol{\Sigma}_j - \boldsymbol{\mu}_j\boldsymbol{\mu}_j^T) + m\log(2\pi e) \right) \tag{3}$$

$$\nabla_{\boldsymbol{w}^{u_l}} \ell_j(\mathcal{W}) = \delta_{j\in\mathcal{V}^{u_l}} \left[\boldsymbol{\mu}_j - \boldsymbol{y}_j\right] \tag{4}$$

$$\nabla_{\boldsymbol{w}^{p_l}} \ell_j(\mathcal{W}) = \delta_{\exists i:(i,j)\in\mathcal{E}^{p_l}} \begin{bmatrix} \mathbf{0} & \boldsymbol{\mu}_j - \boldsymbol{y}_j \end{bmatrix}^T + \delta_{\exists k:(j,k)\in\mathcal{E}^{p_l}} \begin{bmatrix} \boldsymbol{\mu}_j - \boldsymbol{y}_j & \mathbf{0} \end{bmatrix}^T \tag{5}$$

$$\nabla_{\boldsymbol{W}^{u_l}} \ell_j(\mathcal{W}) = \frac{1}{2}\delta_{j\in\mathcal{V}^{u_l}} \left[\boldsymbol{y}_j\boldsymbol{y}_j^T - \boldsymbol{\Sigma}_j\right] \tag{6}$$

$$\nabla_{\boldsymbol{W}^{p_l}} \ell_j(\mathcal{W}) = \frac{1}{2}\left(\delta_{\exists i:(i,j)\in\mathcal{E}^{p_l}} \begin{bmatrix} \mathbf{0} & \boldsymbol{y}_i\boldsymbol{y}_j^T - \boldsymbol{y}_i\boldsymbol{\mu}_j^T \\ \boldsymbol{y}_j\boldsymbol{y}_i^T - \boldsymbol{\mu}_j\boldsymbol{y}_i^T & \boldsymbol{y}_j\boldsymbol{y}_j^T - \boldsymbol{\Sigma}_j \end{bmatrix} + \delta_{\exists k:(j,k)\in\mathcal{E}^{p_l}} \begin{bmatrix} \boldsymbol{y}_j\boldsymbol{y}_j^T - \boldsymbol{\Sigma}_j & \boldsymbol{y}_j\boldsymbol{y}_k^T - \boldsymbol{\mu}_j\boldsymbol{y}_k^T \\ \boldsymbol{y}_k\boldsymbol{y}_j^T - \boldsymbol{y}_k\boldsymbol{\mu}_j^T & \mathbf{0} \end{bmatrix} \right) \tag{7}$$

Figure 3. Analytic expressions for the negative log-pseudolikelihood $\ell_j(\mathcal{W}) \overset{\text{def}}{=} -\log p(\boldsymbol{y}_j, \boldsymbol{y}_{\mathcal{V}\setminus j}\boldsymbol{x}; \mathcal{W})$ and the gradient with respect to the parameters for a single conditioned subgraph centered around j. Recall that the mean parameters are given by $\boldsymbol{\mu}_j = [\boldsymbol{\Theta}_j(\boldsymbol{x}, \mathcal{W})]^{-1}\boldsymbol{\theta}_j(\boldsymbol{y}_{\mathcal{V}\setminus j}, \boldsymbol{x}, \mathcal{W})$ and $\boldsymbol{\Sigma}_j = [\boldsymbol{\Theta}_j(\boldsymbol{x}, \mathcal{W})]^{-1} + \boldsymbol{\mu}_j\boldsymbol{\mu}_j^T$.

$$[\boldsymbol{\theta}(\boldsymbol{x}, \mathcal{W})]_j = \sum_u \delta_{j\in\mathcal{V}^u} \boldsymbol{\theta}^u_j(\boldsymbol{x}, \boldsymbol{w}) + \sum_{p,(i,j)\in\mathcal{E}^p} [\boldsymbol{\theta}^p_{ij}(\boldsymbol{x}, \mathcal{W})]_j + \sum_{p,(j,k)\in\mathcal{E}^p} [\boldsymbol{\theta}^p_{jk}(\boldsymbol{x}, \mathcal{W})]_j \tag{8}$$

$$[\boldsymbol{\Theta}(\boldsymbol{x}, \mathcal{W})\boldsymbol{y}]_j = \sum_u \delta_{j\in\mathcal{V}^u} \boldsymbol{\Theta}^u_j(\boldsymbol{x}, \mathcal{W})\boldsymbol{y}_j + \sum_{p,(i,j)\in\mathcal{E}^p} \big([\boldsymbol{\Theta}^p_{ij}(\boldsymbol{x}, \mathcal{W})]_{jj}\boldsymbol{y}_j + [\boldsymbol{\Theta}^p_{ij}(\boldsymbol{x}, \mathcal{W})]_{ji}\boldsymbol{y}_i \big) + \sum_{p,(j,k)\in\mathcal{E}^p} \Big([\boldsymbol{\Theta}^p_{jk}(\boldsymbol{x}, \mathcal{W})]_{jj}\boldsymbol{y}_j + [\boldsymbol{\Theta}^p_{jk}(\boldsymbol{x}, \mathcal{W})]_{jk}\boldsymbol{y}_k \Big) \tag{9}$$

Figure 4. Expressions required to solve the linear system $\boldsymbol{\Theta}(\boldsymbol{x}, \mathcal{W})\boldsymbol{y} = \boldsymbol{\theta}(\boldsymbol{x}, \mathcal{W})$ for $\boldsymbol{y}$ using conjugate gradient at test time. This involves computing the right-hande side, as well as the product of the system matrix with an arbitrary vector. The equations give the m-dimensional block corresponding to pixel j, both of the right-hand side and the matrix-vector product. This enables trivial parallelization over the pixels.

Figure 5. Additional representative test set results for image denoising with structured noise. First and third row: input images with simulated dust on the camera lens. Second and last row: RTF denoising results with a 3x3 model and decision trees of depth ten (PSNR 27.85).

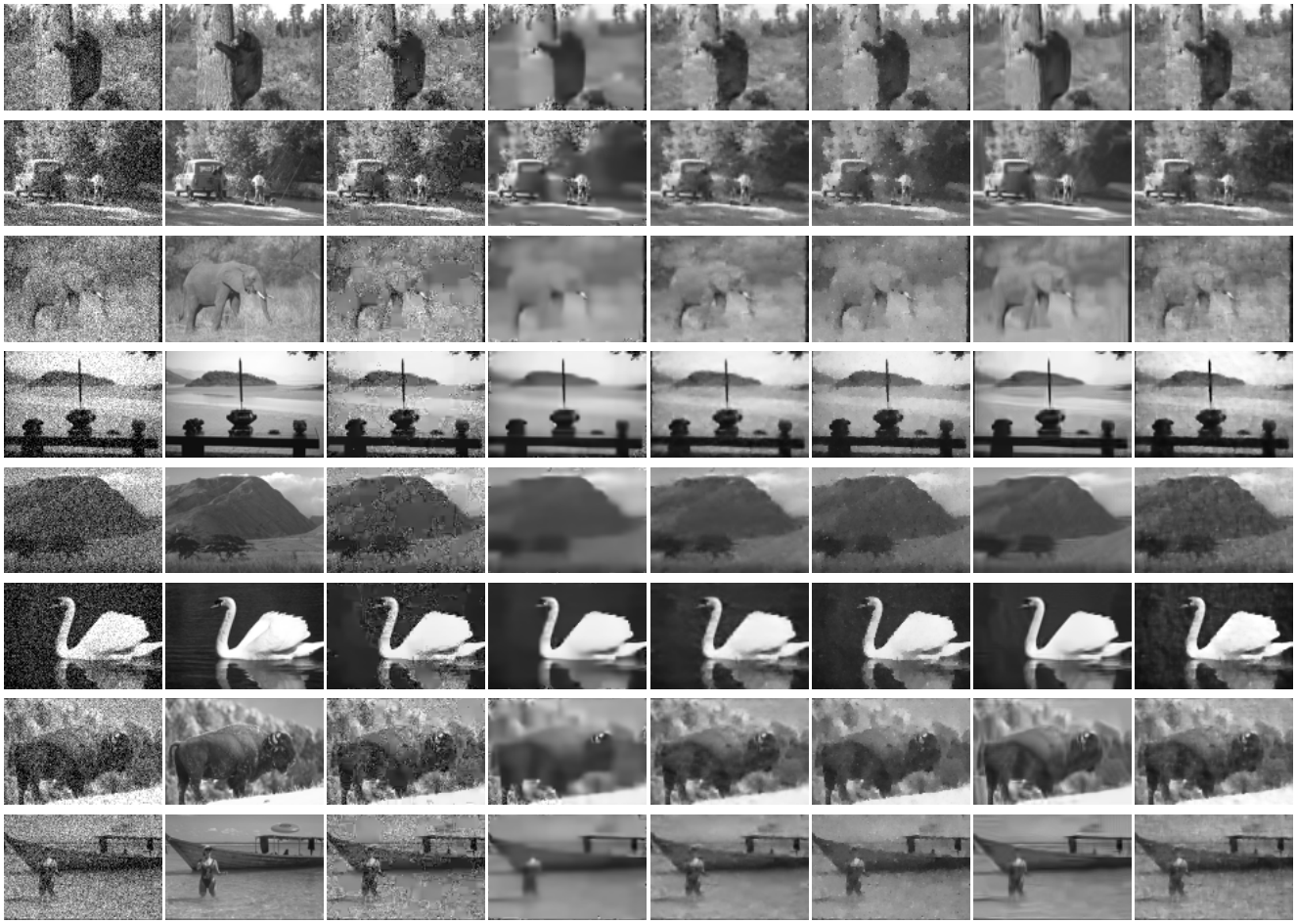
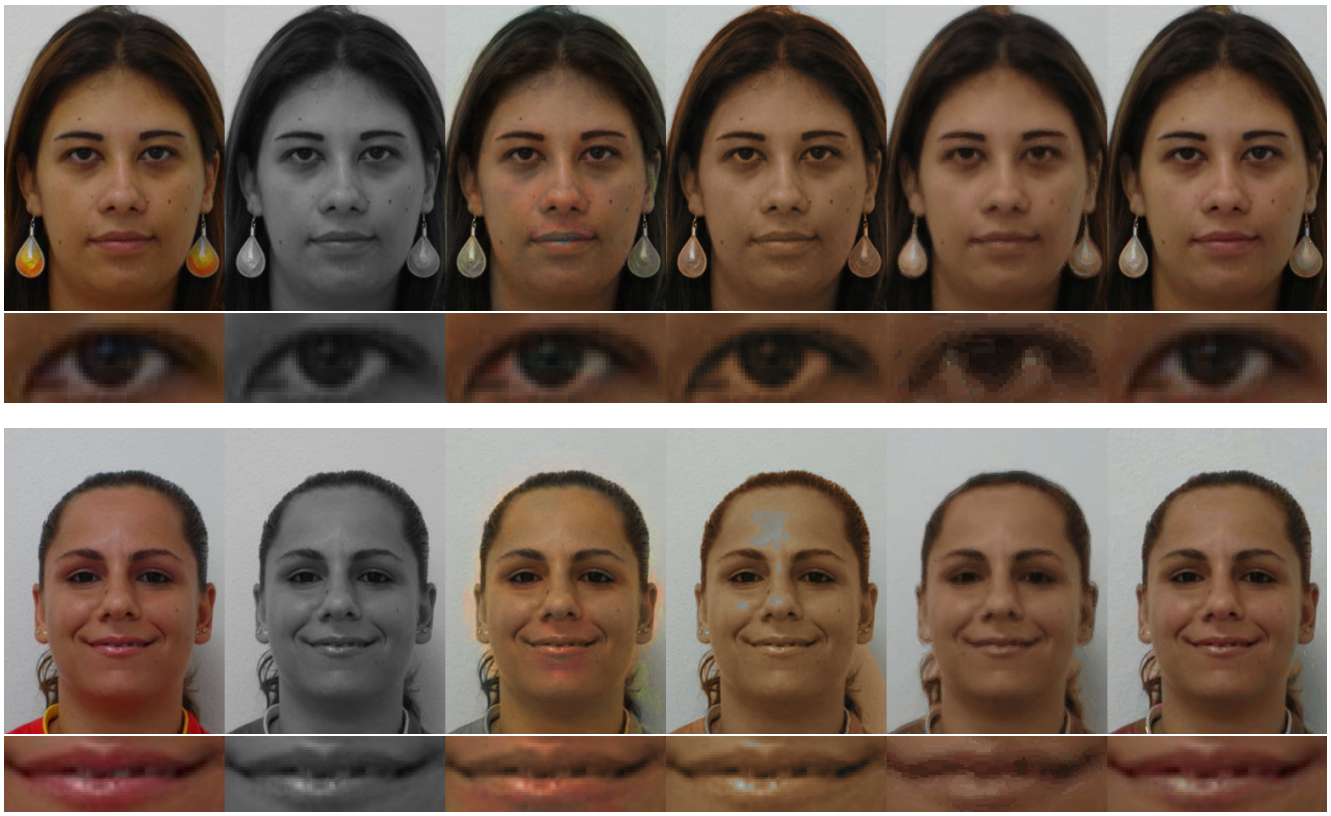
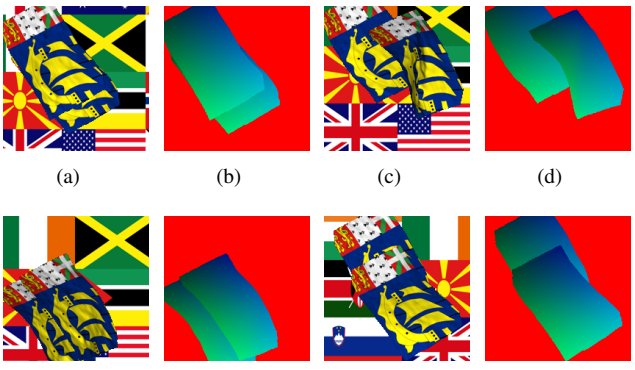


Figure 6. Additional test set denoising results, $\sigma = 30$. From left to right: Noisy input image; ground truth; FoE MAP 3x3; FoE MAP 5x5; FoE MMSE 3x3; FoE MMSE PW; BM3D; RTF 3x3.



(a) (b) (c) (d) (e) (f)

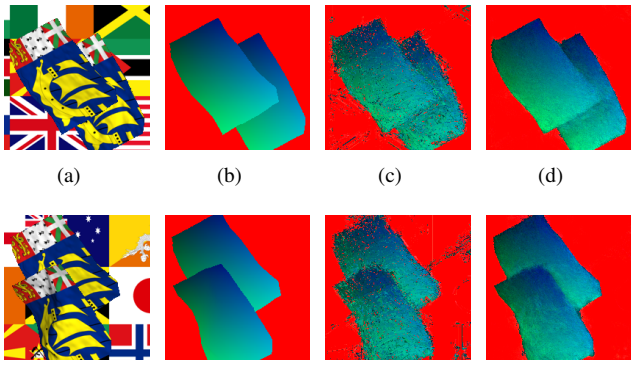
Figure 7. **Face colorization** (top rows: full images, bottom rows: zoom-in). Given a gray-scale test image **(b)**, the goal is to recover its color. **(a)** The ground truth. **(c,d)** Two competitors (see Fig. 12 in main paper for details). **(e)** Our result with unaries only (one tree, depth ten). 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.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 8. Training data for experiment: Detection and Registration (main paper: Sec. 4.4 and Figure 13). Left is input image and right is ground truth labeling.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 9. Results on test data for experiment: Detection and Registration (main paper: Sec. 4.4 and Figure 13). Each row from left to right: input image, ground truth (RGB), unary prediction, RTF 3x3 prediction.