Shrinkage Fields for Effective Image Restoration
– Supplemental Material –

Uwe Schmidt Stefan Roth
Department of Computer Science, TU Darmstadt

1. Derivations

While the following derivations may look complex on a first glance, they require only standard (multivariate) calculus. We use the *numerator layout notation*[1] for all derivations.

Please also note that all derived formulae can be computed efficiently and implemented compactly (Matlab code for learning and inference is available on the authors' webpages).

We derive the iterative update equation for the image variables $\mathbf{x}$ in Eq. (6) of the main paper as:

$$
\begin{aligned}
g_\beta(\mathbf{z}) &= \arg\min_{\mathbf{x}} E_\beta(\mathbf{x}|\mathbf{z},\mathbf{y}) = \arg\max_{\mathbf{x}} \exp\big(-E_\beta(\mathbf{x}|\mathbf{z},\mathbf{y})\big) && (1)\\
&= \arg\max_{\mathbf{x}} \exp\left(-\frac{\lambda}{2}\|\mathbf{y}-\mathbf{K}\mathbf{x}\|^2 - \sum_{i=1}^{N}\sum_{c\in\mathcal{C}}\left(\frac{\beta}{2}\left(\mathbf{f}_i^\mathsf{T}\mathbf{x}_{(c)} - z_{ic}\right)^2 + \rho_i(z_{ic})\right)\right) && (2)\\
&= \arg\max_{\mathbf{x}} \exp\left(-\frac{\lambda}{2}(\mathbf{y}-\mathbf{K}\mathbf{x})^\mathsf{T}(\mathbf{y}-\mathbf{K}\mathbf{x}) - \frac{\beta}{2}\sum_{i=1}^{N}(\mathbf{F}_i\mathbf{x}-\mathbf{z}_i)^\mathsf{T}(\mathbf{F}_i\mathbf{x}-\mathbf{z}_i)\right) && (3)\\
&= \arg\max_{\mathbf{x}} \exp\left(-\frac{1}{2}\mathbf{x}^\mathsf{T}\left(\lambda\mathbf{K}^\mathsf{T}\mathbf{K} + \beta\sum_{i=1}^{N}\mathbf{F}_i^\mathsf{T}\mathbf{F}_i\right)\mathbf{x} + \mathbf{x}^\mathsf{T}\left(\lambda\mathbf{K}^\mathsf{T}\mathbf{y} + \beta\sum_{i=1}^{N}\mathbf{F}_i^\mathsf{T}\mathbf{z}_i\right)\right) && (4)\\
&= \arg\max_{\mathbf{x}} \mathcal{N}\left(\left[\lambda\mathbf{K}^\mathsf{T}\mathbf{K} + \beta\sum_{i=1}^{N}\mathbf{F}_i^\mathsf{T}\mathbf{F}_i\right]^{-1}\left[\lambda\mathbf{K}^\mathsf{T}\mathbf{y} + \beta\sum_{i=1}^{N}\mathbf{F}_i^\mathsf{T}\mathbf{z}_i\right], \left[\lambda\mathbf{K}^\mathsf{T}\mathbf{K} + \beta\sum_{i=1}^{N}\mathbf{F}_i^\mathsf{T}\mathbf{F}_i\right]^{-1}\right) && (5)\\
&= \left[\lambda\mathbf{K}^\mathsf{T}\mathbf{K} + \beta\sum_{i=1}^{N}\mathbf{F}_i^\mathsf{T}\mathbf{F}_i\right]^{-1}\left[\lambda\mathbf{K}^\mathsf{T}\mathbf{y} + \beta\sum_{i=1}^{N}\mathbf{F}_i^\mathsf{T}\mathbf{z}_i\right] && (6)\\
&= \left[\frac{\lambda}{\beta}\mathbf{K}^\mathsf{T}\mathbf{K} + \sum_{i=1}^{N}\mathbf{F}_i^\mathsf{T}\mathbf{F}_i\right]^{-1}\left[\frac{\lambda}{\beta}\mathbf{K}^\mathsf{T}\mathbf{y} + \sum_{i=1}^{N}\mathbf{F}_i^\mathsf{T}\mathbf{z}_i\right] && (7)\\
&= \mathcal{F}^{-1}\left[\frac{\mathcal{F}\left(\frac{\lambda}{\beta}\mathbf{K}^\mathsf{T}\mathbf{y} + \sum_{i=1}^{N}\mathbf{F}_i^\mathsf{T}\mathbf{z}_i\right)}{\frac{\lambda}{\beta}\check{\mathbf{K}}^* \circ \check{\mathbf{K}} + \sum_{i=1}^{N}\check{\mathbf{F}}_i^* \circ \check{\mathbf{F}}_i}\right]. && (8)
\end{aligned}
$$

As in the main paper, convolution is denoted by $\mathbf{F}\mathbf{x} = [\mathbf{f}^\mathsf{T}\mathbf{x}_{(\mathcal{C}_1)}, \ldots, \mathbf{f}^\mathsf{T}\mathbf{x}_{(\mathcal{C}_{|\mathcal{C}|})}]^\mathsf{T} \equiv \mathbf{f}\otimes\mathbf{x} \equiv \mathcal{F}^{-1}(\check{\mathbf{F}}\circ\mathcal{F}(\mathbf{x}))$. The optical transfer function $\check{\mathbf{F}} \equiv \mathcal{F}(\mathbf{f})$ is derived from filter (point spread function) $\mathbf{f}$, where $\mathcal{F}$ denotes the discrete Fourier transform (DFT). Note that division and multiplication ($\circ$) are applied element-wise in Eq. (8); $\check{\mathbf{F}}^*$ denotes the complex conjugate of $\check{\mathbf{F}}$.

The last step is possible since $\mathbf{K}$ and the $\mathbf{F}_i$ are block-circulant matrices with circulant blocks as they correspond to convolutions with circular boundary conditions. Consequently, they are diagonalized by DFTs, allowing element-wise operations.

[1] *cf*. http://en.wikipedia.org/w/index.php?title=Matrix_calculus&oldid=576691987#Numerator-layout_notation.

1.1. Boundary handling

To reduce boundary artifacts, which may arise due to using convolutions with circular boundary conditions, we use padding of the input image. Specifically, we first take the input image $\mathbf{x} \in \mathbb{R}^D$ (of width w and height h with $D = h \cdot w$) and replicate b pixels of its boundary on all sides by multiplication with the sparse "padding" matrix $\mathbf{P}_b$[2]. In image denoising, b can be freely chosen (we use $b = 10$); for deconvolution $b = (r-1)/2$ with square blur kernel $\mathbf{k} \in \mathbb{R}^{r^2}$ of size $r \times r$ pixels[3].

For deconvolution we additionally apply "edge-tapering" to the padded input image $\mathbf{P}_b\mathbf{x}$ so that it better matches the assumptions of applying convolution with circular boundary handling. In particular, we applied u iterations of Matlab's `edgetaper` function (where $\boldsymbol{\alpha}_\mathbf{k}$ is a weighting vector based on $\mathbf{k}$), which can be formalized as

$$\begin{aligned}
\texttt{edgetaper}(\mathbf{x}, \mathbf{k}) &= \boldsymbol{\alpha}_\mathbf{k} \circ \mathbf{x} + (\mathbf{1} - \boldsymbol{\alpha}_\mathbf{k}) \circ (\mathbf{k} \otimes \mathbf{x}) && (9)\\
&= \operatorname{diag}\{\boldsymbol{\alpha}_\mathbf{k}\}\mathbf{x} + \operatorname{diag}\{\mathbf{1} - \boldsymbol{\alpha}_\mathbf{k}\}(\mathbf{K}\mathbf{x}) && (10)\\
&= \big(\operatorname{diag}\{\boldsymbol{\alpha}_\mathbf{k}\} + \operatorname{diag}\{\mathbf{1} - \boldsymbol{\alpha}_\mathbf{k}\}\mathbf{K}\big)\mathbf{x} && (11)\\
&= \mathbf{E}_\mathbf{k}\mathbf{x}. && (12)
\end{aligned}$$

Performing u iterations of edge-tapering can thus be expressed by multiplication with the matrix $\mathbf{E}_\mathbf{k} = \operatorname{diag}\{\boldsymbol{\alpha}_\mathbf{k}\} + \operatorname{diag}\{\mathbf{1} - \boldsymbol{\alpha}_\mathbf{k}\}\mathbf{K}$ raised to the u^{th} power. (For denoising, $\mathbf{E}_\mathbf{k}^u = \mathbf{I}$ is set to the identity matrix.)

We denote the boundary-padded and possibly edge-tapered input image as $\mathbf{x}^\mathsf{P} = \mathbf{E}_\mathbf{k}^u \mathbf{P}_b \mathbf{x}$, and as $\hat{\mathbf{x}}^\mathsf{B}$ the output image where the padded boundary region has not been removed yet. Consequently, we use the following minor variation of $g_{\boldsymbol{\Theta}}(\mathbf{x})$ (Eqs. (10) and (11) of the main paper) in practice:

$$\begin{aligned}
g_{\boldsymbol{\Theta}}(\mathbf{x}) &= \mathbf{T}_b\, \mathcal{F}^{-1}\left[\frac{\mathcal{F}\left(\lambda \mathbf{K}^\mathsf{T}\mathbf{y} + \sum_{i=1}^N \mathbf{F}_i^\mathsf{T} f_{\boldsymbol{\pi}_i}(\mathbf{F}_i \mathbf{x}^\mathsf{P})\right)}{\lambda \check{\mathbf{K}}^* \circ \check{\mathbf{K}} + \sum_{i=1}^N \check{\mathbf{F}}_i^* \circ \check{\mathbf{F}}_i}\right] && (13)\\
&= \mathbf{T}_b \underbrace{\left[\lambda \mathbf{K}^\mathsf{T}\mathbf{K} + \sum_{i=1}^N \mathbf{F}_i^\mathsf{T}\mathbf{F}_i\right]^{-1}}_{\boldsymbol{\Omega}^{-1}} \underbrace{\left[\lambda \mathbf{K}^\mathsf{T}\mathbf{y} + \sum_{i=1}^N \mathbf{F}_i^\mathsf{T} f_{\boldsymbol{\pi}_i}(\mathbf{F}_i \mathbf{x}^\mathsf{P})\right]}_{\boldsymbol{\eta}} && (14)\\
&= \mathbf{T}_b \hat{\mathbf{x}}^\mathsf{B}. && (15)
\end{aligned}$$

The uncropped output image is given as $\hat{\mathbf{x}}^\mathsf{B} = \boldsymbol{\Omega}^{-1}\boldsymbol{\eta}$. To obtain an output image $\hat{\mathbf{x}} = g_{\boldsymbol{\Theta}}(\mathbf{x})$ with the same size as the input image $\mathbf{x}$, we remove the padded boundary region by multiplication with the sparse ("cropping") matrix $\mathbf{T}_b$.

It is important to note that padding, edge-tapering, and cropping can all be expressed as linear transformations, which allows us to easily compute gradients of the transformed images. Furthermore, please note that for inference and learning (Sec. 1.2), the convolution matrices $\mathbf{K}$ and $\mathbf{F}_i$ are never explicitly constructed (also applies to $\mathbf{K}^\mathsf{T}$ and $\mathbf{F}_i^\mathsf{T}$), since all matrix-vector products can be efficiently computed through convolutions.

1.2. Learning

In the following we use $g_{\boldsymbol{\Theta}}(\mathbf{x})$ with boundary handling as defined in Eq. (15).

Recall from the main paper that given training data $\{\mathbf{x}_{\text{gt}}^{(s)}, \mathbf{y}^{(s)}, \mathbf{k}^{(s)}\}_{s=1}^S$, we learn all model parameters $\boldsymbol{\Theta}_t = \{\lambda_t, \boldsymbol{\pi}_{ti}, \mathbf{f}_{ti}\}_{i=1}^N$ greedily stage-by-stage for $t = 1, \ldots, T$ via

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

or jointly for all T stages via

$$J(\boldsymbol{\Theta}_{1,\ldots,T}) = \sum_{s=1}^S \ell(\hat{\mathbf{x}}_T^{(s)}; \mathbf{x}_{\text{gt}}^{(s)}). \tag{17}$$

[2]In Matlab, $\mathbf{P}_b\mathbf{x} \equiv$ `reshape(padarray(reshape(x,h,w),[b,b],'replicate','both'),D,1)`.

[3]Square kernel only assumed for simplicity here.

To that end, at each stage of greedy learning $J(\mathbf{\Theta}_t)$ we need to compute Eq. (16) and its gradient $\frac{\partial J(\mathbf{\Theta}_t)}{\partial \mathbf{\Theta}_t}$; for joint training $J(\mathbf{\Theta}_{1,\ldots,T})$ we need to compute Eq. (17) and its gradient $\left[\frac{\partial J(\mathbf{\Theta}_{1,\ldots,T})}{\partial \mathbf{\Theta}_1}, \ldots, \frac{\partial J(\mathbf{\Theta}_{1,\ldots,T})}{\partial \mathbf{\Theta}_T}\right]$.

1.2.1 Gradient of loss function

We can address one training example at a time since the gradients of Eqs. (16) and (17) decompose into sums over training examples. The gradient w.r.t. the model parameters of the last/current stage can be computed as follows:

$$\frac{\partial \ell(\hat{\mathbf{x}}_t; \mathbf{x}_{\text{gt}})}{\partial \mathbf{\Theta}_t} = \frac{\partial \ell(\hat{\mathbf{x}}_t; \mathbf{x}_{\text{gt}})}{\partial \hat{\mathbf{x}}_t} \cdot \frac{\partial \mathbf{T}_b \mathbf{\Omega}_t^{-1} \boldsymbol{\eta}_t}{\partial \mathbf{\Theta}_t} \tag{18}$$

$$= \frac{\partial \ell(\hat{\mathbf{x}}_t; \mathbf{x}_{\text{gt}})}{\partial \hat{\mathbf{x}}_t} \mathbf{T}_b \mathbf{\Omega}_t^{-1} \left[-\frac{\partial \mathbf{\Omega}_t}{\partial \mathbf{\Theta}_t} \mathbf{\Omega}_t^{-1} \boldsymbol{\eta}_t + \frac{\partial \boldsymbol{\eta}_t}{\partial \mathbf{\Theta}_t} \right] \tag{19}$$

$$= \hat{\mathbf{c}}_t^{\mathsf{T}} \left[\frac{\partial \boldsymbol{\eta}_t}{\partial \mathbf{\Theta}_t} - \frac{\partial \mathbf{\Omega}_t}{\partial \mathbf{\Theta}_t} \hat{\mathbf{x}}_t^{\mathsf{B}} \right] \quad \text{with} \quad \hat{\mathbf{c}}_t = \mathbf{\Omega}_t^{-1} \mathbf{T}_b^{\mathsf{T}} \left[\frac{\partial \ell(\hat{\mathbf{x}}_t; \mathbf{x}_{\text{gt}})}{\partial \hat{\mathbf{x}}_t} \right]^{\mathsf{T}}. \tag{20}$$

The gradient of the chosen loss function (negative PSNR)

$$\ell(\hat{\mathbf{x}}; \mathbf{x}_{\text{gt}}) = -20 \log_{10} \left(\frac{R\sqrt{D}}{\|\hat{\mathbf{x}} - \mathbf{x}_{\text{gt}}\|} \right), \tag{21}$$

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

$$\frac{\partial \ell(\hat{\mathbf{x}}; \mathbf{x}_{\text{gt}})}{\partial \hat{\mathbf{x}}} = -\frac{20}{\log 10} \frac{\partial}{\partial \hat{\mathbf{x}}} \log \left(\frac{R\sqrt{D}}{\|\hat{\mathbf{x}} - \mathbf{x}_{\text{gt}}\|} \right) \tag{22}$$

$$= \frac{20}{\log 10} \frac{\partial \log \|\hat{\mathbf{x}} - \mathbf{x}_{\text{gt}}\|}{\partial \hat{\mathbf{x}}} \tag{23}$$

$$= \frac{20}{\log 10} \frac{1}{\|\hat{\mathbf{x}} - \mathbf{x}_{\text{gt}}\|} \frac{\partial \|\hat{\mathbf{x}} - \mathbf{x}_{\text{gt}}\|}{\partial \hat{\mathbf{x}}} \tag{24}$$

$$= \frac{20}{\log 10} \frac{1}{\|\hat{\mathbf{x}} - \mathbf{x}_{\text{gt}}\|} \frac{(\hat{\mathbf{x}} - \mathbf{x}_{\text{gt}})^{\mathsf{T}}}{\|\hat{\mathbf{x}} - \mathbf{x}_{\text{gt}}\|} \tag{25}$$

$$= \frac{20}{\log 10} \frac{(\hat{\mathbf{x}} - \mathbf{x}_{\text{gt}})^{\mathsf{T}}}{\|\hat{\mathbf{x}} - \mathbf{x}_{\text{gt}}\|^2}. \tag{26}$$

For joint training we additionally require gradients w.r.t. the model parameters of previous stages. The gradient w.r.t. the second-to-last stage is obtained as ($t \geq 2$)

$$\frac{\partial \ell(\hat{\mathbf{x}}_t; \mathbf{x}_{\text{gt}})}{\partial \mathbf{\Theta}_{t-1}} = \hat{\mathbf{c}}_t^{\mathsf{T}} \left[\frac{\partial \boldsymbol{\eta}_t}{\partial \mathbf{\Theta}_{t-1}} \right] \tag{27}$$

$$= \hat{\mathbf{c}}_t^{\mathsf{T}} \frac{\partial}{\partial \mathbf{\Theta}_{t-1}} \left(\sum_{i=1}^{N} \mathbf{F}_{ti}^{\mathsf{T}} f_{\boldsymbol{\pi}_{ti}}(\mathbf{F}_{ti} \hat{\mathbf{x}}_{t-1}^{\mathsf{P}}) \right) \tag{28}$$

$$= \hat{\mathbf{c}}_t^{\mathsf{T}} \sum_{i=1}^{N} \mathbf{F}_{ti}^{\mathsf{T}} f'_{\boldsymbol{\pi}_{ti}}(\mathbf{F}_{ti} \hat{\mathbf{x}}_{t-1}^{\mathsf{P}}) \mathbf{F}_{ti} \mathbf{E}_{\mathbf{k}}^{u} \mathbf{P}_b \frac{\partial \hat{\mathbf{x}}_{t-1}}{\partial \mathbf{\Theta}_{t-1}} \tag{29}$$

$$= \left[\sum_{i=1}^{N} (\mathbf{F}_{ti} \hat{\mathbf{c}}_t)^{\mathsf{T}} f'_{\boldsymbol{\pi}_{ti}}(\mathbf{F}_{ti} \hat{\mathbf{x}}_{t-1}^{\mathsf{P}}) \mathbf{F}_{ti} \right] \mathbf{E}_{\mathbf{k}}^{u} \mathbf{P}_b \frac{\partial \mathbf{T}_b \mathbf{\Omega}_{t-1}^{-1} \boldsymbol{\eta}_{t-1}}{\partial \mathbf{\Theta}_{t-1}} \tag{30}$$

$$= \left[\mathbf{T}_b^{\mathsf{T}} \mathbf{P}_b^{\mathsf{T}} \mathbf{E}_{\mathbf{k}}^{u\,\mathsf{T}} \sum_{i=1}^{N} \mathbf{F}_{ti}^{\mathsf{T}} f'_{\boldsymbol{\pi}_{ti}}(\mathbf{F}_{ti} \hat{\mathbf{x}}_{t-1}^{\mathsf{P}}) \mathbf{F}_{ti} \hat{\mathbf{c}}_t \right]^{\mathsf{T}} \mathbf{\Omega}_{t-1}^{-1} \left[\frac{\partial \boldsymbol{\eta}_{t-1}}{\partial \mathbf{\Theta}_{t-1}} - \frac{\partial \mathbf{\Omega}_{t-1}}{\partial \mathbf{\Theta}_{t-1}} \hat{\mathbf{x}}_{t-1}^{\mathsf{B}} \right] \tag{31}$$

$$= \hat{\mathbf{c}}_{t-1}^{\mathsf{T}} \left[\frac{\partial \boldsymbol{\eta}_{t-1}}{\partial \mathbf{\Theta}_{t-1}} - \frac{\partial \mathbf{\Omega}_{t-1}}{\partial \mathbf{\Theta}_{t-1}} \hat{\mathbf{x}}_{t-1}^{\mathsf{B}} \right] \text{ with } \hat{\mathbf{c}}_{t-1} = \mathbf{\Omega}_{t-1}^{-1} \mathbf{T}_b^{\mathsf{T}} \mathbf{P}_b^{\mathsf{T}} \mathbf{E}_{\mathbf{k}}^{u\,\mathsf{T}} \sum_{i=1}^{N} \mathbf{F}_{ti}^{\mathsf{T}} f'_{\boldsymbol{\pi}_{ti}}(\mathbf{F}_{ti} \hat{\mathbf{x}}_{t-1}^{\mathsf{P}}) \mathbf{F}_{ti} \hat{\mathbf{c}}_t, \tag{32}$$

where $f'_{\boldsymbol{\pi}_{ti}}(\mathbf{F}_{ti}\hat{\mathbf{x}}^{\mathsf{P}}_{t-1}) = \frac{\partial f_{\boldsymbol{\pi}_{ti}}(\mathbf{F}_{ti}\hat{\mathbf{x}}^{\mathsf{P}}_{t-1})}{\partial \mathbf{F}_{ti}\hat{\mathbf{x}}^{\mathsf{P}}_{t-1}}$ (*cf*. Sec. 1.2.2). Note that the form is the same as of Eq. (20), which means we can apply this recursively to compute gradients for earlier stages ($t \geq 2, s = 1, \ldots, t-1$):

$$\frac{\partial \ell(\hat{\mathbf{x}}_t; \mathbf{x}_{\text{gt}})}{\partial \boldsymbol{\Theta}_{t-s}} = \hat{\mathbf{c}}^{\mathsf{T}}_{t-s} \left[\frac{\partial \boldsymbol{\eta}_{t-s}}{\partial \boldsymbol{\Theta}_{t-s}} - \frac{\partial \boldsymbol{\Omega}_{t-s}}{\partial \boldsymbol{\Theta}_{t-s}} \hat{\mathbf{x}}^{\mathsf{B}}_{t-s} \right] \text{ with} \tag{33}$$

$$\hat{\mathbf{c}}_{t-s} = \boldsymbol{\Omega}^{-1}_{t-s} \mathbf{T}^{\mathsf{T}}_b \mathbf{P}^{\mathsf{T}}_b \mathbf{E}^{u\,\mathsf{T}}_{\mathbf{k}} \sum_{i=1}^{N} \mathbf{F}^{\mathsf{T}}_{t-s+1,i} f'_{\boldsymbol{\pi}_{t-s+1,i}}(\mathbf{F}_{t-s+1,i}\hat{\mathbf{x}}^{\mathsf{P}}_{t-s}) \mathbf{F}_{t-s+1,i} \hat{\mathbf{c}}_{t-s+1}. \tag{34}$$

Note that jointly training T stages only takes approximately T times as long as training a single stage.

1.2.2 Gradients for specific model parameters

Now that we have derived the generic gradients for the given loss function, we need to derive the specific gradients w.r.t. the actual model parameters $\boldsymbol{\Theta}_t = \{\lambda_t, \boldsymbol{\pi}_{ti}, \mathbf{f}_{ti}\}_{i=1}^{N}$ at all stages t.

Regularization weight λ. We define $\lambda_t = \exp(\tilde{\lambda}_t)$ to ensure positive values of λ_t and learn $\tilde{\lambda}_t$ via its partial derivative

$$\frac{\partial \ell(\hat{\mathbf{x}}_t; \mathbf{x}_{\text{gt}})}{\partial \tilde{\lambda}_t} = \frac{\partial \ell(\hat{\mathbf{x}}_t; \mathbf{x}_{\text{gt}})}{\partial \lambda_t} \cdot \frac{\partial \exp(\tilde{\lambda}_t)}{\partial \tilde{\lambda}_t} \tag{35}$$

$$= \hat{\mathbf{c}}^{\mathsf{T}}_t \left(\mathbf{K}^{\mathsf{T}}\mathbf{y} - \mathbf{K}^{\mathsf{T}}\mathbf{K}\hat{\mathbf{x}}^{\mathsf{B}}_t \right) \cdot \lambda_t \tag{36}$$

$$= \lambda_t (\mathbf{K}\hat{\mathbf{c}}_t)^{\mathsf{T}} (\mathbf{y} - \mathbf{K}\hat{\mathbf{x}}^{\mathsf{B}}_t). \tag{37}$$

Shrinkage function. Recall that the shrinkage function is modeled as a linear combination of Gaussian RBF kernels:

$$f_{\boldsymbol{\pi}}(v) = \sum_{j=1}^{M} \pi_j \exp\left(-\frac{\gamma}{2}(v - \mu_j)^2 \right). \tag{38}$$

Its derivatives w.r.t. the input and weights can be computed as

$$\frac{\partial f_{\boldsymbol{\pi}}(v)}{\partial \pi_j} = \exp\left(-\frac{\gamma}{2}(v - \mu_j)^2 \right) \tag{39}$$

$$\frac{\partial f_{\boldsymbol{\pi}}(v)}{\partial v} = -\sum_{j=1}^{M} \pi_j \exp\left(-\frac{\gamma}{2}(v - \mu_j)^2 \right) \cdot \gamma(v - \mu_j). \tag{40}$$

Concerning vector-valued inputs $\mathbf{v} = [v_1, \ldots, v_L]^{\mathsf{T}}$ to the univariate shrinkage function, please note that

$$f_{\boldsymbol{\pi}}(\mathbf{v}) = [f_{\boldsymbol{\pi}}(v_1), \ldots, f_{\boldsymbol{\pi}}(v_L)]^{\mathsf{T}} \tag{41}$$

$$\frac{\partial f_{\boldsymbol{\pi}}(\mathbf{v})}{\partial \mathbf{v}} = \operatorname{diag}\left\{ \frac{\partial f_{\boldsymbol{\pi}}(v_1)}{\partial v_1}, \ldots, \frac{\partial f_{\boldsymbol{\pi}}(v_L)}{\partial v_L} \right\} =: f'_{\boldsymbol{\pi}}(\mathbf{v}) \in \mathbb{R}^{L \times L}. \tag{42}$$

For practical purposes, we not only precompute and store in a lookup-table (LUT) the shrinkage function $f_{\boldsymbol{\pi}}(v)$ for all (sensible) v, but all its derivatives that are required for learning the model, which are quickly retrieved from the LUT via linear interpolation.

Concretely, the relevant gradient of the shrinkage function weights is computed as

$$\frac{\partial \ell(\hat{\mathbf{x}}_t; \mathbf{x}_{\text{gt}})}{\partial \boldsymbol{\pi}_{ti}} = \hat{\mathbf{c}}^{\mathsf{T}}_t \left(\mathbf{F}^{\mathsf{T}}_{ti} \frac{\partial f_{\boldsymbol{\pi}_{ti}}(\mathbf{F}_{ti}\hat{\mathbf{x}}^{\mathsf{P}}_{t-1})}{\partial \boldsymbol{\pi}_{ti}} \right) \tag{43}$$

$$= (\mathbf{F}_{ti}\hat{\mathbf{c}}_t)^{\mathsf{T}} \frac{\partial f_{\boldsymbol{\pi}_{ti}}(\mathbf{F}_{ti}\hat{\mathbf{x}}^{\mathsf{P}}_{t-1})}{\partial \boldsymbol{\pi}_{ti}}, \tag{44}$$

where $\frac{\partial f_{\boldsymbol{\pi}_{ti}}(\mathbf{F}_{ti}\hat{\mathbf{x}}^{\mathrm{P}}_{t-1})}{\partial \boldsymbol{\pi}_{ti}} \in \mathbb{R}^{D\times M}$ is a matrix, which contains in each row the derivatives w.r.t. all M entries of vector $\boldsymbol{\pi}_{ti}$ for all D filter responses $\mathbf{F}_{ti}\hat{\mathbf{x}}^{\mathrm{P}}_{t-1}$.

Filter. We define each filter of size $m\times n$ w.r.t. a basis $\mathbf{B} \in \mathbb{R}^{mn\times V}$ as $\mathbf{f} = \mathbf{B}\tilde{\mathbf{f}}$ and learn the entries of $\tilde{\mathbf{f}} \in \mathbb{R}^V$. We follow Chen *et al.* [1] and choose DCT filters for $\mathbf{B}$, omitting the DC-component to guarantee zero-mean filters.

First, it is useful to denote $\mathbf{f} \otimes \mathbf{x} \equiv \mathbf{F}\mathbf{x} = (\mathbf{f}^\mathsf{T}[\mathbf{x}]_\mathcal{C})^\mathsf{T} = [\mathbf{x}]_\mathcal{C}^\mathsf{T}\mathbf{f}$, where $[\mathbf{x}]_\mathcal{C} \in \mathbb{R}^{mn\times D}$ is a matrix of all cliques $\mathcal{C}$ of $\mathbf{x} \in \mathbb{R}^D$ that filter $\mathbf{f}$ is applied to. Then, we can differentiate the convolved image w.r.t. the filter coefficients $\tilde{\mathbf{f}}$ as follows

$$\frac{\partial \mathbf{F}\mathbf{x}}{\partial \tilde{\mathbf{f}}} = \frac{\partial [\mathbf{x}]_\mathcal{C}^\mathsf{T}\mathbf{B}\tilde{\mathbf{f}}}{\partial \tilde{\mathbf{f}}} = [\mathbf{x}]_\mathcal{C}^\mathsf{T}\mathbf{B} \in \mathbb{R}^{D\times V} \tag{45}$$

The gradient of the loss w.r.t. the filter coefficients is derived as

$$\begin{aligned}
\frac{\partial \ell(\hat{\mathbf{x}}_t;\mathbf{x}_{\mathrm{gt}})}{\partial \tilde{\mathbf{f}}_{ti}} &= \hat{\mathbf{c}}_t^\mathsf{T}\left[\frac{\partial \mathbf{F}_{ti}^\mathsf{T} f_{\boldsymbol{\pi}_{ti}}(\mathbf{F}_{ti}\hat{\mathbf{x}}^{\mathrm{P}}_{t-1})}{\partial \tilde{\mathbf{f}}_{ti}} - \frac{\partial \mathbf{F}_{ti}^\mathsf{T}\mathbf{F}_{ti}}{\partial \tilde{\mathbf{f}}_{ti}}\hat{\mathbf{x}}^{\mathrm{B}}_t\right] && (46)\\
&= f_{\boldsymbol{\pi}_{ti}}(\mathbf{F}_{ti}\hat{\mathbf{x}}^{\mathrm{P}}_{t-1})^\mathsf{T}\frac{\partial \mathbf{F}_{ti}\hat{\mathbf{c}}_t}{\partial \tilde{\mathbf{f}}_{ti}} + (\mathbf{F}_{ti}\hat{\mathbf{c}}_t)^\mathsf{T}\left(\frac{\partial f_{\boldsymbol{\pi}_{ti}}(\mathbf{F}_{ti}\hat{\mathbf{x}}^{\mathrm{P}}_{t-1})}{\partial \mathbf{F}_{ti}\hat{\mathbf{x}}^{\mathrm{P}}_{t-1}}\cdot\frac{\partial \mathbf{F}_{ti}\hat{\mathbf{x}}^{\mathrm{P}}_{t-1}}{\partial \tilde{\mathbf{f}}_{ti}}\right) - \frac{\partial (\mathbf{F}_{ti}\hat{\mathbf{c}}_t)^\mathsf{T}\mathbf{F}_{ti}\hat{\mathbf{x}}^{\mathrm{B}}_t}{\partial \tilde{\mathbf{f}}_{ti}} && (47)\\
&= f_{\boldsymbol{\pi}_{ti}}(\mathbf{F}_{ti}\hat{\mathbf{x}}^{\mathrm{P}}_{t-1})^\mathsf{T}[\hat{\mathbf{c}}_t]_\mathcal{C}^\mathsf{T}\mathbf{B} + (\mathbf{F}_{ti}\hat{\mathbf{c}}_t)^\mathsf{T} f'_{\boldsymbol{\pi}_{ti}}(\mathbf{F}_{ti}\hat{\mathbf{x}}^{\mathrm{P}}_{t-1})[\hat{\mathbf{x}}^{\mathrm{P}}_{t-1}]_\mathcal{C}^\mathsf{T}\mathbf{B} - (\mathbf{F}_{ti}\hat{\mathbf{c}}_t)^\mathsf{T}[\hat{\mathbf{x}}^{\mathrm{B}}_t]_\mathcal{C}^\mathsf{T}\mathbf{B} - (\mathbf{F}_{ti}\hat{\mathbf{x}}^{\mathrm{B}}_t)^\mathsf{T}[\hat{\mathbf{c}}_t]_\mathcal{C}^\mathsf{T}\mathbf{B} && (48)\\
&= \left[[\hat{\mathbf{c}}_t]_\mathcal{C}\left(f_{\boldsymbol{\pi}_{ti}}(\mathbf{F}_{ti}\hat{\mathbf{x}}^{\mathrm{P}}_{t-1}) - \mathbf{F}_{ti}\hat{\mathbf{x}}^{\mathrm{B}}_t\right) + [\hat{\mathbf{x}}^{\mathrm{P}}_{t-1}]_\mathcal{C} f'_{\boldsymbol{\pi}_{ti}}(\mathbf{F}_{ti}\hat{\mathbf{x}}^{\mathrm{P}}_{t-1})\mathbf{F}_{ti}\hat{\mathbf{c}}_t - [\hat{\mathbf{x}}^{\mathrm{B}}_t]_\mathcal{C}\mathbf{F}_{ti}\hat{\mathbf{c}}_t\right]^\mathsf{T}\mathbf{B}. && (49)
\end{aligned}$$

To avoid duplicating degrees of freedom, in practice we learn filters with fixed unit norm $\mathbf{f} = \frac{\mathbf{B}\tilde{\mathbf{f}}}{\|\mathbf{B}\tilde{\mathbf{f}}\|} = \mathbf{B}\tilde{\mathbf{f}}\cdot\left(\tilde{\mathbf{f}}^\mathsf{T}(\mathbf{B}^\mathsf{T}\mathbf{B})\tilde{\mathbf{f}}\right)^{-1/2}$. To perform learning for these normalized filters, it is again useful to first derive

$$\begin{aligned}
\frac{\partial \mathbf{F}\mathbf{x}}{\partial \tilde{\mathbf{f}}} &= \frac{\partial}{\partial \tilde{\mathbf{f}}}[\mathbf{x}]_\mathcal{C}^\mathsf{T}\mathbf{B}\tilde{\mathbf{f}}\cdot\left(\tilde{\mathbf{f}}^\mathsf{T}(\mathbf{B}^\mathsf{T}\mathbf{B})\tilde{\mathbf{f}}\right)^{-1/2} && (50)\\
&= [\mathbf{x}]_\mathcal{C}^\mathsf{T}\mathbf{B}\cdot\left(\left(\tilde{\mathbf{f}}^\mathsf{T}(\mathbf{B}^\mathsf{T}\mathbf{B})\tilde{\mathbf{f}}\right)^{-1/2}\cdot\mathbf{I} - \tilde{\mathbf{f}}\cdot\frac{1}{2}\left(\tilde{\mathbf{f}}^\mathsf{T}(\mathbf{B}^\mathsf{T}\mathbf{B})\tilde{\mathbf{f}}\right)^{-3/2}\cdot 2\tilde{\mathbf{f}}^\mathsf{T}\mathbf{B}^\mathsf{T}\mathbf{B}\right) && (51)\\
&= \|\mathbf{B}\tilde{\mathbf{f}}\|^{-1}\cdot\left([\mathbf{x}]_\mathcal{C}^\mathsf{T}\mathbf{B} - [\mathbf{x}]_\mathcal{C}^\mathsf{T}\mathbf{B}\tilde{\mathbf{f}}\cdot\|\mathbf{B}\tilde{\mathbf{f}}\|^{-1}\|\mathbf{B}\tilde{\mathbf{f}}\|^{-1}\cdot\tilde{\mathbf{f}}^\mathsf{T}\mathbf{B}^\mathsf{T}\mathbf{B}\right) && (52)\\
&= \|\mathbf{B}\tilde{\mathbf{f}}\|^{-1}\cdot\left([\mathbf{x}]_\mathcal{C}^\mathsf{T}\mathbf{B} - [\mathbf{x}]_\mathcal{C}^\mathsf{T}\mathbf{f}\cdot\mathbf{f}^\mathsf{T}\mathbf{B}\right) && (53)\\
&= \left([\mathbf{x}]_\mathcal{C}^\mathsf{T} - \mathbf{F}\mathbf{x}\cdot\mathbf{f}^\mathsf{T}\right)\frac{\mathbf{B}}{\|\mathbf{B}\tilde{\mathbf{f}}\|} && (54)
\end{aligned}$$

to then derive the gradient of the loss w.r.t. the filter coefficients

$$\begin{aligned}
\frac{\partial \ell(\hat{\mathbf{x}}_t;\mathbf{x}_{\mathrm{gt}})}{\partial \tilde{\mathbf{f}}_{ti}} &= \left[[\hat{\mathbf{c}}_t]_\mathcal{C}\left(f_{\boldsymbol{\pi}_{ti}}(\mathbf{F}_{ti}\hat{\mathbf{x}}^{\mathrm{P}}_{t-1}) - \mathbf{F}_{ti}\hat{\mathbf{x}}^{\mathrm{B}}_t\right) + [\hat{\mathbf{x}}^{\mathrm{P}}_{t-1}]_\mathcal{C} f'_{\boldsymbol{\pi}_{ti}}(\mathbf{F}_{ti}\hat{\mathbf{x}}^{\mathrm{P}}_{t-1})\mathbf{F}_{ti}\hat{\mathbf{c}}_t - [\hat{\mathbf{x}}^{\mathrm{B}}_t]_\mathcal{C}\mathbf{F}_{ti}\hat{\mathbf{c}}_t\right]^\mathsf{T}\frac{\mathbf{B}}{\|\mathbf{B}\tilde{\mathbf{f}}_{ti}\|} && (55)\\
&\quad - \left[(\mathbf{F}_{ti}\hat{\mathbf{c}}_t)^\mathsf{T}\left(f_{\boldsymbol{\pi}_{ti}}(\mathbf{F}_{ti}\hat{\mathbf{x}}^{\mathrm{P}}_{t-1}) - \mathbf{F}_{ti}\hat{\mathbf{x}}^{\mathrm{B}}_t\right) + (\mathbf{F}_{ti}\hat{\mathbf{x}}^{\mathrm{P}}_{t-1})^\mathsf{T} f'_{\boldsymbol{\pi}_{ti}}(\mathbf{F}_{ti}\hat{\mathbf{x}}^{\mathrm{P}}_{t-1})\mathbf{F}_{ti}\hat{\mathbf{c}}_t - (\mathbf{F}_{ti}\hat{\mathbf{x}}^{\mathrm{B}}_t)^\mathsf{T}\mathbf{F}_{ti}\hat{\mathbf{c}}_t\right]\frac{\mathbf{f}^\mathsf{T}\mathbf{B}}{\|\mathbf{B}\tilde{\mathbf{f}}_{ti}\|} && (56)\\
&= \left[[\hat{\mathbf{c}}_t]_\mathcal{C}\left(f_{\boldsymbol{\pi}_{ti}}(\mathbf{F}_{ti}\hat{\mathbf{x}}^{\mathrm{P}}_{t-1}) - \mathbf{F}_{ti}\hat{\mathbf{x}}^{\mathrm{B}}_t\right) + [\hat{\mathbf{x}}^{\mathrm{P}}_{t-1}]_\mathcal{C} f'_{\boldsymbol{\pi}_{ti}}(\mathbf{F}_{ti}\hat{\mathbf{x}}^{\mathrm{P}}_{t-1})\mathbf{F}_{ti}\hat{\mathbf{c}}_t - [\hat{\mathbf{x}}^{\mathrm{B}}_t]_\mathcal{C}\mathbf{F}_{ti}\hat{\mathbf{c}}_t\right]^\mathsf{T}\frac{\mathbf{B}}{\|\mathbf{B}\tilde{\mathbf{f}}_{ti}\|} && (57)\\
&\quad - (\mathbf{F}_{ti}\hat{\mathbf{c}}_t)^\mathsf{T}\left[f_{\boldsymbol{\pi}_{ti}}(\mathbf{F}_{ti}\hat{\mathbf{x}}^{\mathrm{P}}_{t-1}) - 2\mathbf{F}_{ti}\hat{\mathbf{x}}^{\mathrm{B}}_t + f'_{\boldsymbol{\pi}_{ti}}(\mathbf{F}_{ti}\hat{\mathbf{x}}^{\mathrm{P}}_{t-1})\mathbf{F}_{ti}\hat{\mathbf{x}}^{\mathrm{P}}_{t-1}\right]\frac{\mathbf{f}^\mathsf{T}\mathbf{B}}{\|\mathbf{B}\tilde{\mathbf{f}}_{ti}\|}, && (58)
\end{aligned}$$

where we can re-use most of our previous derivation (Eq. 49).

2. Proof of Lemma 1

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

Proof. If $\epsilon = 0$, Lemma 1 is trivially true; hence assume $\epsilon > 0$ from now on. Let

$$g(z) = f(z) - \epsilon z, \tag{59}$$

$$\hat{z}_f = \arg\min_z f(z), \tag{60}$$

$$\hat{z}_g = \arg\min_z g(z), \tag{61}$$

then the inequalities

$$f(\hat{z}_g) \geq f(\hat{z}_f) = \min_z f(z) \tag{62}$$

$$g(\hat{z}_f) \geq g(\hat{z}_g) = \min_z g(z) \tag{63}$$

hold. Combining above two inequalities proves the Lemma:

$$g(\hat{z}_f) \geq g(\hat{z}_g) \tag{64}$$

$$\Rightarrow \quad f(\hat{z}_f) - \epsilon\hat{z}_f \geq f(\hat{z}_g) - \epsilon\hat{z}_g \tag{65}$$

$$\Rightarrow \quad f(\hat{z}_f) - \epsilon\hat{z}_f \geq f(\hat{z}_f) - \epsilon\hat{z}_g \tag{66}$$

$$\Rightarrow \quad -\epsilon\hat{z}_f \geq -\epsilon\hat{z}_g \tag{67}$$

$$\Rightarrow \quad \hat{z}_f \leq \hat{z}_g \tag{68}$$

$$\Rightarrow \quad \arg\min_z f(z) \leq \arg\min_z (f(z) - \epsilon z) \tag{69}$$

□

Acknowledgements. We thank *StackExchange (Mathematics)* user *JiK* for their advice.

References

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