LIRMM, Université de Montpellier

Séminaire du 18 janvier 2018

On modeling image patch distribution for image restoration

Charles Deledalle

Joint work with:
Shibin Parameswaran (UCSD/SPAWAR)
Loïc Denis (Télécom Saint Etienne)
Truong Nguyen (UCSD).

Institut de Mathématiques de Bordeaux, CNRS-Université Bordeaux, France
Department of Electrical and Computer Engineering, University of California, San Diego (UCSD), USA

Introduction

In many scenarios, one cannot get a perfect clean pictures of a scene:

- Camera shake • Motion • Objects out-of-focus • Low-light conditions.


In many applications, images are noisy, blurry, sub-sampled, compressed, etc:

- Microscopy • Astronomy • Remote sensing • Medical • Sonar.

Automatic image restoration algorithms are needed.
Fast computation is required to process large image data-sets.

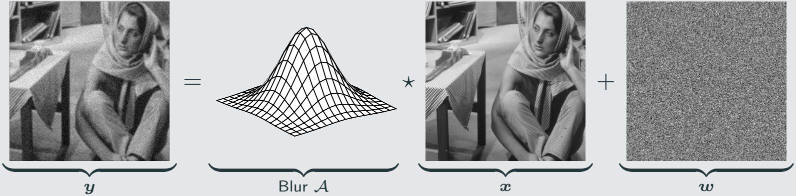
Introduction – Inverse problems

Model

$$\boldsymbol{y} = \mathcal{A}\boldsymbol{x} + \boldsymbol{w}$$

- $\boldsymbol{y} \in \mathbb{R}^M$ observed degraded image (with M pixels)
- $\boldsymbol{x} \in \mathbb{R}^N$ unknown underlying "clean" image (with N pixels)
- $\boldsymbol{w} \sim \mathcal{N}(0, \sigma^2 \mathbf{Id}_M)$ noise component (standard deviation σ)
- $\mathcal{A} : \mathbb{R}^N \to \mathbb{R}^M$: linear operator (blur, missing pixels, random projections)

Deconvolution subject to noise



Goal: Retrieve the sharp and clean image x from y

Linear least square estimator

$$\hat{\boldsymbol{x}} \in \operatorname*{argmin}_{\boldsymbol{x}} \frac{1}{2\sigma^2}\|\mathcal{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2$$

One solution is the Moore-Penrose pseudo inverse:

$$\hat{\boldsymbol{x}} = \mathcal{A}^+\boldsymbol{y} = \lim_{\varepsilon \to 0}(\mathcal{A}^t\mathcal{A} + \varepsilon\mathbf{Id}_N)^{-1}\mathcal{A}^t\boldsymbol{y}$$

Example (Deconvolution)

$\mathcal{A} = \mathcal{F}^{-1}\Phi\mathcal{F}$: circulant matrix

$\mathcal{F}$: Fourier transform

$\Phi = \mathrm{diag}(\phi_1, \ldots, \phi_N)$: blur Fourier coefficients

Linear least square solution

$$\hat{\boldsymbol{x}} = \mathcal{F}^{-1}\hat{\boldsymbol{c}} \quad \text{with} \quad \hat{c}_i = \begin{cases} \frac{\phi_i^* c_i}{|\phi_i|^2} & \text{if} \quad |\phi_i| > 0 \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad \boldsymbol{c} = \mathcal{F}\boldsymbol{y}$$

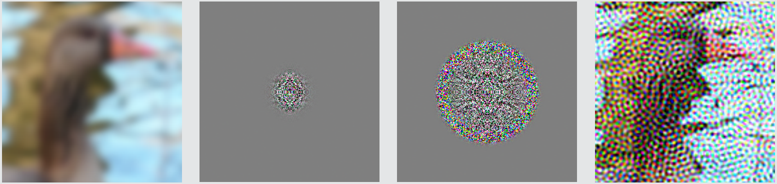
Linear least square estimator

$$\hat{\boldsymbol{x}} \in \underset{\boldsymbol{x}}{\operatorname{argmin}} \ \frac{1}{2\sigma^2} \|\mathcal{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2$$

One solution is the Moore-Penrose pseudo inverse:

$$\hat{\boldsymbol{x}} = \mathcal{A}^+ \boldsymbol{y} = \lim_{\varepsilon \to 0} (\mathcal{A}^t \mathcal{A} + \varepsilon \mathbf{Id}_N)^{-1} \mathcal{A}^t \boldsymbol{y}$$

Example (Deconvolution)



(a) Observation $\boldsymbol{y}$ (b) $\boldsymbol{c} = \mathcal{F}\boldsymbol{y}$ (c) $\hat{\boldsymbol{c}}$ (d) $\hat{\boldsymbol{x}} = \mathcal{F}^{-1}\hat{\boldsymbol{c}}$

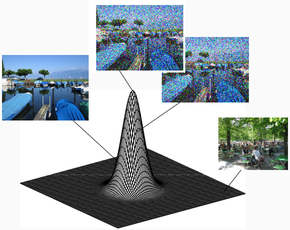
Variational model: Regularized linear least-square

$$\hat{\boldsymbol{x}} \in \operatorname*{argmin}_{\boldsymbol{x}} \frac{1}{2\sigma^2}\|\mathcal{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + R(\boldsymbol{x})$$

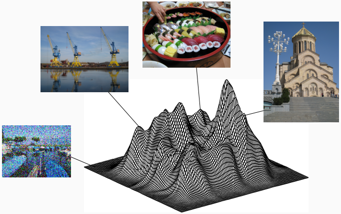
Example (Maximum A Posteriori (MAP))

$$\frac{1}{2\sigma^2}\|\mathcal{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 = -\log p(\boldsymbol{y}|\boldsymbol{x}) \quad \text{(likelihood for Gaussian noises)}$$

$$R(\boldsymbol{x}) = -\log p(\boldsymbol{x}) \quad \text{(a priori)}$$



Likelihood $x \mapsto p(y\,|\,x)$



Prior $x \mapsto p(x)$

What prior?

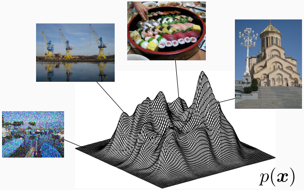
Variational model: Regularized linear least-square

$$\hat{\boldsymbol{x}} \in \operatorname*{argmin}_{\boldsymbol{x}} \frac{1}{2\sigma^2}\|\mathcal{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + R(\boldsymbol{x})$$

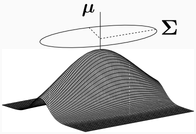
Example (Maximum A Posteriori (MAP))

$$\frac{1}{2\sigma^2}\|\mathcal{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 = -\log p(\boldsymbol{y}|\boldsymbol{x}) \quad \text{(likelihood for Gaussian noises)}$$

$$R(\boldsymbol{x}) = -\log p(\boldsymbol{x}) \quad \text{(a priori)}$$



$\approx$



Gaussian prior $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

What about a Gaussian prior?

Motivations – Variational models

Variational model: Regularized linear least-square

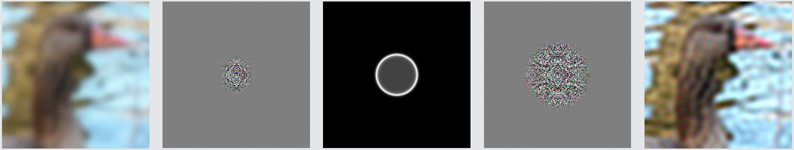
$$\hat{\boldsymbol{x}} \in \underset{\boldsymbol{x}}{\operatorname{argmin}} \ \frac{1}{2\sigma^2}\|\mathcal{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + R(\boldsymbol{x})$$

Example (Wiener deconvolution / Tikhonov regularization)

$$R(\boldsymbol{x}) = \|\underbrace{\Lambda^{-1/2}\mathcal{F}}_{\Gamma}\boldsymbol{x}\|_2^2 = \sum_i \left(\frac{c_i}{\lambda_i}\right)^2 \quad \text{with} \quad \boldsymbol{c} = \mathcal{F}\boldsymbol{x}$$

$\Lambda = \operatorname{diag}(\lambda_1^2, \ldots, \lambda_N^2)$: mean power spectral density ($\lambda_i \approx \beta|\omega_{i,j}|^{-\alpha}$)

Solution is linear: $\hat{\boldsymbol{x}} = (\mathcal{A}^t\mathcal{A} + \sigma^2\Gamma^t\Gamma)^{-1}\mathcal{A}^t\boldsymbol{y}$



(a) $\boldsymbol{y}$ (b) $\boldsymbol{c} = \mathcal{F}\boldsymbol{y}$ (c) $\frac{\phi_i^*}{|\phi_i|^2 + \sigma^2/\lambda_i^2}$ (d) $\hat{\boldsymbol{c}}$ (e) $\hat{\boldsymbol{x}} = \mathcal{F}^{-1}\hat{\boldsymbol{c}}$

Motivations – Variational models

Variational model: Regularized linear least-square

$$\hat{\boldsymbol{x}} \in \underset{\boldsymbol{x}}{\operatorname{argmin}} \ \frac{1}{2\sigma^2}\|\mathcal{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + R(\boldsymbol{x})$$

Example (Wavelet shrinkage/thresholding)

$$R(\boldsymbol{x}) = \|\Lambda^{-1/2}\mathcal{W}\boldsymbol{x}\|_1 = \sum_i \frac{|c_i|}{\lambda_i} \quad \text{with} \quad \boldsymbol{c} = \mathcal{W}\boldsymbol{x}$$

$\mathcal{W}$: Wavelet transform or Frame ($\mathcal{W}^+\mathcal{W} = \mathbf{Id}_N$)

$\Lambda = \operatorname{diag}(\lambda_1^2, \ldots, \lambda_N^2)$: energy for each sub-band ($\lambda_i \approx C2^{j_i}$)

Solution is non-linear, sparse and non-explicit (requires an iterative solver):

(a) $\boldsymbol{y}$

(b) $\hat{\boldsymbol{x}}$

Motivations – Variational models

Variational model: Regularized linear least-square

$$\hat{\boldsymbol{x}} \in \operatorname*{argmin}_{\boldsymbol{x}} \frac{1}{2\sigma^2}\|\mathcal{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + R(\boldsymbol{x})$$

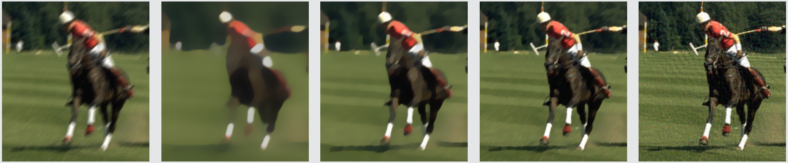
Example (Total-Variation (Rudin *et al.*, 1992))

$$R(\boldsymbol{x}) = \frac{1}{\lambda}\|\nabla \boldsymbol{x}\|_{12} = \frac{1}{\lambda}\sum_{i,j}\sqrt{|x_{i+1,j} - x_{ij}|^2 + |x_{i,j+1} - x_{ij}|^2},$$

∇ : gradient – horizontal and vertical forward finite difference

$\lambda > 0$: regularization parameter (difficult to tune)

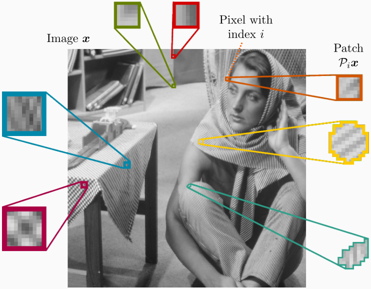
Solution is again non-linear and non-explicit (requires an iterative solver):



(a) Blurry (b) Tiny λ (c) Small λ (d) Medium λ (e) Huge λ

Motivations – Patch priors

- Modeling the distribution of images is difficult.
- Learning this distribution as well (curse of dimensionality).
- Images lie on a complex and large dimensional manifold.
- Their distribution may be spread out on different clusters.



Divide and conquer approach:

Break down images into small patches and model their distribution.

$$\hat{\boldsymbol{x}} \in \operatorname*{argmin}_{\boldsymbol{x}} \frac{P}{2\sigma^2}\|\mathcal{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \sum_{i=1}^{N} R(\mathcal{P}_i\boldsymbol{x})$$

All reconstructed overlapping patches must be well explained by the prior.

$\mathcal{P}_i : \mathbb{R}^N \to \mathbb{R}^P$ extracts a patch with P pixels centered at location i.

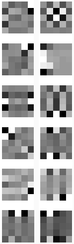
Linear operator. Typically, $P = 8\times 8$.

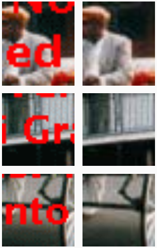
Regularized linear least-square with patch priors

$$\hat{\boldsymbol{x}} \in \underset{\boldsymbol{x}}{\operatorname{argmin}} \; \frac{P}{2\sigma^2}\|\mathcal{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \sum_{i=1}^{N} R(\mathcal{P}_i \boldsymbol{x})$$

Example (Fields of Experts, Roth *et al.*, 2005)

- $R(\boldsymbol{z}) = \sum_{k=1}^{K} \alpha_k \log\left(1 + \frac{1}{2}\langle \phi_k, \boldsymbol{z}\rangle^2\right)$, $\alpha_k > 0$, $\phi_k \in \mathbb{R}^P$ a high-pass filter.
- K Student-t experts parametrized by α_k and ϕ_k.
- Learned by maximum likelihood with MCMC.





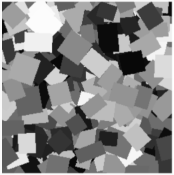
Motivations – Patch priors

Regularized linear least-square with patch priors

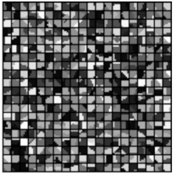
$$\hat{\boldsymbol{x}} \in \underset{\boldsymbol{x}}{\operatorname{argmin}} \; \frac{P}{2\sigma^2}\|\mathcal{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \sum_{i=1}^{N} R(\mathcal{P}_i \boldsymbol{x})$$

Example (Analysis k-SVD, Rubinstein *et al.*, 2013)

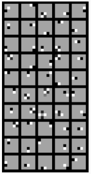
- $R(\boldsymbol{z}) = \frac{1}{\lambda}\|\Gamma \boldsymbol{z}\|_0 = \#\{c_i \neq 0\}$ with $\boldsymbol{c} = \Gamma \boldsymbol{z}$
- $\|\cdot\|_0$: ℓ_0 pseudo-norm promoting sparsity.
- $\Gamma \in \mathbb{R}^{Q \times P}$ learned from a large collection of clean patches.
- Patches distributed on an union of sub-spaces (clusters).



Training image



Training set of patches



Learned atoms

Motivations – Patch priors

Regularized linear least-square with patch priors

$$\hat{\boldsymbol{x}} \in \underset{\boldsymbol{x}}{\operatorname{argmin}} \; \frac{P}{2\sigma^2}\|\mathcal{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \sum_{i=1}^{N} R(\mathcal{P}_i \boldsymbol{x})$$

Example (Gaussian Mixture Model priors, Yu *et al.*, 2010)

$$R(\boldsymbol{z}) = -\log p(\boldsymbol{z} - \bar{\boldsymbol{z}}) \quad \text{with} \quad \bar{\boldsymbol{z}} = \frac{1}{P}\mathbf{1}_P \mathbf{1}_P^t \boldsymbol{z}$$

$$\text{and} \quad p(\boldsymbol{z}) = \sum_{k=1}^{K} w_k \frac{1}{(2\pi)^{P/2}|\boldsymbol{\Sigma}_k|^{1/2}} \exp\left(-\frac{1}{2}\boldsymbol{z}^t \boldsymbol{\Sigma}_k^{-1} \boldsymbol{z}\right),$$

- K: number of Gaussians (clusters)
- w_k: weights $\sum_k w_k = 1$ (frequency of each clusters)
- $\boldsymbol{\Sigma}_k$: $P \times P$ covariance matrix (shape of cluster)
- Zero mean assumption (contrast invariance)

Least square + GMM Patch Prior = Expected Patch Log Likelihood (EPLL)

Motivations – Patch priors

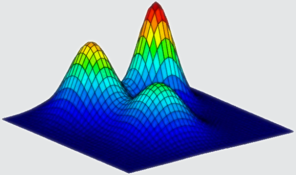
Regularized linear least-square with patch priors

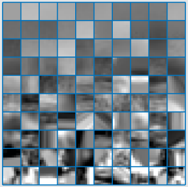
$$\hat{\boldsymbol{x}} \in \underset{\boldsymbol{x}}{\operatorname{argmin}} \; \frac{P}{2\sigma^2}\|\mathcal{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \sum_{i=1}^{N} R(\mathcal{P}_i \boldsymbol{x})$$

Example (EPLL, Zoran & Weiss, 2011)

$$R(\boldsymbol{z}) = -\log p(\boldsymbol{z} - \bar{\boldsymbol{z}}) \quad \text{with} \quad \bar{\boldsymbol{z}} = \frac{1}{P}\mathbf{1}_P \mathbf{1}_P^t \boldsymbol{z}$$

and $$p(\boldsymbol{z}) = \sum_{k=1}^{K} w_k \frac{1}{(2\pi)^{P/2}|\boldsymbol{\Sigma}_k|^{1/2}} \exp\left(-\frac{1}{2}\boldsymbol{z}^t \boldsymbol{\Sigma}_k^{-1} \boldsymbol{z}\right),$$





$(w_k, \boldsymbol{\Sigma}_k)$ learned by EM on 2 million patches.

Patch size: $P = 8 \times 8$

#Gaussians: $K = 200$

100 randomly generated patches from the learned model

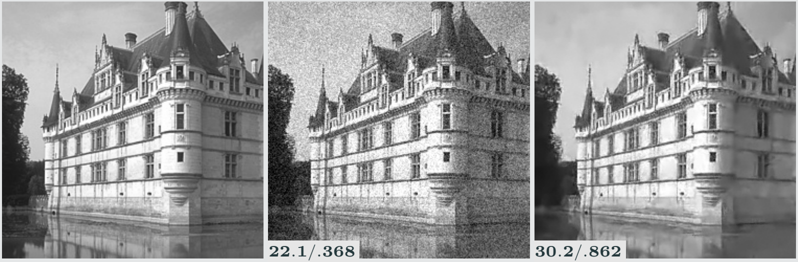
Motivations – Patch priors

Regularized linear least-square with patch priors

$$\hat{\boldsymbol{x}} \in \underset{\boldsymbol{x}}{\operatorname{argmin}} \ \frac{P}{2\sigma^2}\|\mathcal{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \sum_{i=1}^{N} R(\mathcal{P}_i \boldsymbol{x})$$

Example (EPLL, Zoran & Weiss, 2011)

Noise with standard-deviation $\sigma = 20$ (images in range $[0, 255]$)



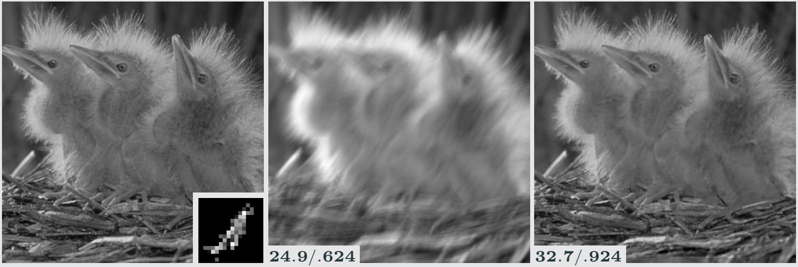
(a) Reference $\boldsymbol{x}$ (b) Noisy image $\boldsymbol{y}$ (c) EPLL result $\hat{\boldsymbol{x}}$

Regularized linear least-square with patch priors

$$\hat{\boldsymbol{x}} \in \underset{\boldsymbol{x}}{\operatorname{argmin}} \ \frac{P}{2\sigma^2} \|\mathcal{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \sum_{i=1}^{N} R(\mathcal{P}_i \boldsymbol{x})$$

Example (EPLL, Zoran & Weiss, 2011)

Motion blur subject to noise with standard-deviation $\sigma = .5$



(a) Reference $\boldsymbol{x}$ / Blur kernel (b) Blurry image $\boldsymbol{y}$ (c) EPLL result $\hat{\boldsymbol{x}}$

Motivations – Patch priors

Regularized linear least-square with patch priors

$$\hat{\boldsymbol{x}} \in \underset{\boldsymbol{x}}{\operatorname{argmin}} \ \frac{P}{2\sigma^2} \|\mathcal{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \sum_{i=1}^{N} R(\mathcal{P}_i \boldsymbol{x})$$

Example (EPLL, Zoran & Weiss, 2011)

Pros:

- Near state-of-the-art results in denoising, super-resolution, in-painting...
- No regularization parameter to tune per image-degradation pair.
- Only parameters: the patch size P and the number of components K.
- Multi-scale adaptation is straightforward (Papyan & Elad, 2016).

Cons:

- Non-convex optimization problem
- Original solver is very slow
- Some Gibbs artifacts/oscillations can be observed

Motivations – Patch priors

Regularized linear least-square with patch priors

$$\hat{\boldsymbol{x}} \in \underset{\boldsymbol{x}}{\operatorname{argmin}} \ \frac{P}{2\sigma^2}\|\mathcal{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \sum_{i=1}^{N} R(\mathcal{P}_i \boldsymbol{x})$$

Example (EPLL, Zoran & Weiss, 2011)

Pros:

- Near state-of-the-art results in denoising, super-resolution, in-painting...
- No regularization parameter to tune per image-degradation pair.
- Only parameters: the patch size P and the number of components K.
- Multi-scale adaptation is straightforward (Papyan & Elad, 2016).

Cons:

- Non-convex optimization problem .... EPLL Algorithm (Part 1)
- Original solver is very slow .... Fast EPLL (Part 2)
- Some Gibbs artifacts/oscillations can be observed ...... GGMMs (Part 3)

Part 1/3: EPLL Algorithm (Zoran & Weiss, 2011)

Least square + GMM Patch Prior

$$\hat{\boldsymbol{x}} \in \underset{\boldsymbol{x}}{\operatorname{argmin}} \ \frac{P}{2\sigma^2}\|\mathcal{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 - \sum_{i=1}^{N} \log p(\mathcal{P}_i \boldsymbol{x} - \overline{\mathcal{P}_i \boldsymbol{x}})$$

Half-quadratic splitting

- Introduce N auxiliary vectors $\boldsymbol{z}_i \in \mathbb{R}^P$ and solve instead:

$$\lim_{\beta \to \infty} \underset{\boldsymbol{x}, \boldsymbol{z}_1, \ldots, \boldsymbol{z}_N}{\operatorname{argmin}} \ \frac{P}{2\sigma^2}\|\mathcal{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \frac{\beta}{2}\sum_{i=1}^{N} \|\mathcal{P}_i \boldsymbol{x} - \boldsymbol{z}_i\|_2^2 - \sum_{i=1}^{N} \log p\left(\boldsymbol{z}_i - \bar{\boldsymbol{z}}_i\right).$$

- Use an alternating optimization scheme on $\boldsymbol{z}_i$ and $\boldsymbol{x}$. Repeat:

$$\boldsymbol{z}_i \leftarrow \underset{\boldsymbol{z}_i}{\operatorname{argmin}} \ \frac{\beta}{2}\|\mathcal{P}_i \hat{\boldsymbol{x}} - \boldsymbol{z}_i\|_2^2 - \log p\left(\boldsymbol{z}_i - \bar{\boldsymbol{z}}_i\right), \quad \text{for all } 1 \leqslant i \leqslant N$$

$$\hat{\boldsymbol{x}} \leftarrow \underset{\boldsymbol{x}}{\operatorname{argmin}} \ \frac{P}{2\sigma^2}\|\mathcal{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \frac{\beta}{2}\sum_{i=1}^{N} \|\mathcal{P}_i \boldsymbol{x} - \hat{\boldsymbol{z}}_i\|_2^2$$

$$\beta \leftarrow \text{increase}(\beta)$$

Part 1/3: EPLL Algorithm (Zoran & Weiss, 2011)

Optimization on $\boldsymbol{x}$:

$$\hat{\boldsymbol{x}} \in \underset{\boldsymbol{x}}{\operatorname{argmin}} \frac{P}{2\sigma^2}\|\mathcal{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \frac{\beta}{2}\sum_{i=1}^{N}\|\mathcal{P}_i\boldsymbol{x} - \hat{\boldsymbol{z}}_i\|_2^2$$

$$= \left(\mathcal{A}^t\mathcal{A} + \frac{\beta\sigma^2}{P}\underbrace{\sum_{i=1}^{N}\mathcal{P}_i^t\mathcal{P}_i}_{P\mathbf{Id}_N}\right)^{-1}\left(\mathcal{A}^t\boldsymbol{y} + \frac{\beta\sigma^2}{P}\sum_{i=1}^{N}\mathcal{P}_i^t\hat{\boldsymbol{z}}_i\right)$$

$$= \underbrace{\left(\mathcal{A}^t\mathcal{A} + \beta\sigma^2\mathbf{Id}_N\right)^{-1}\left(\mathcal{A}^t\boldsymbol{y} + \beta\sigma^2\tilde{\boldsymbol{x}}\right)}_{\substack{\text{In general, } O(N) \text{ or } O(N\log N) \\ \text{Otherwise, conjugate gradient}}} \quad \text{with} \quad \underbrace{\tilde{\boldsymbol{x}} = \frac{1}{P}\sum_{i=1}^{N}\mathcal{P}_i^t\hat{\boldsymbol{z}}_i}_{\text{Patch reprojection}}$$

Part 1/3: EPLL Algorithm (Zoran & Weiss, 2011)

Optimization on $\boldsymbol{x}$:

$$\hat{\boldsymbol{x}} \in \underset{\boldsymbol{x}}{\operatorname{argmin}} \frac{P}{2\sigma^2}\|\mathcal{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \frac{\beta}{2}\sum_{i=1}^{N}\|\mathcal{P}_i\boldsymbol{x} - \hat{\boldsymbol{z}}_i\|_2^2$$

$$= \Bigg(\mathcal{A}^t\mathcal{A} + \frac{\beta\sigma^2}{P}\underbrace{\sum_{i=1}^{N}\mathcal{P}_i^t\mathcal{P}_i}_{P\mathbf{Id}_N}\Bigg)^{-1}\Bigg(\mathcal{A}^t\boldsymbol{y} + \frac{\beta\sigma^2}{P}\sum_{i=1}^{N}\mathcal{P}_i^t\hat{\boldsymbol{z}}_i\Bigg)$$

$$= \underbrace{\left(\mathcal{A}^t\mathcal{A} + \beta\sigma^2\mathbf{Id}_N\right)^{-1}\left(\mathcal{A}^t\boldsymbol{y} + \beta\sigma^2\tilde{\boldsymbol{x}}\right)}_{\substack{\text{In general, } O(N) \text{ or } O(N\log N)\\ \text{Otherwise, conjugate gradient}}} \quad \text{with} \quad \underbrace{\tilde{\boldsymbol{x}} = \frac{1}{P}\sum_{i=1}^{N}\mathcal{P}_i^t\hat{\boldsymbol{z}}_i}_{\text{Patch reprojection}}$$

Example (Deconvolution)

For $\mathcal{A} = \mathcal{F}^{-1}\Phi\mathcal{F}$, $\Phi = \operatorname{diag}(\phi_1, \ldots, \phi_N)$, we get

$$\hat{\boldsymbol{x}} = \mathcal{F}^{-1}\hat{\boldsymbol{c}} \quad \text{where} \quad \hat{c}_i = \frac{\phi_i^* c_i + \beta\sigma^2\tilde{c}_i}{|\phi_i|^2 + \beta\sigma^2} \quad \text{with} \quad \begin{cases} \boldsymbol{c} = \mathcal{F}\boldsymbol{y} \\ \tilde{\boldsymbol{c}} = \mathcal{F}\tilde{\boldsymbol{x}} \end{cases}$$

Part 1/3: EPLL Algorithm (Zoran & Weiss, 2011)

Optimization on $\boldsymbol{z}$:

$$
\begin{aligned}
\hat{\boldsymbol{z}} &\in \underset{\boldsymbol{z}}{\operatorname{argmin}} \ \frac{\beta}{2}\|\tilde{\boldsymbol{z}} - \boldsymbol{z}\|_2^2 - \log p\left(\boldsymbol{z} - \bar{\boldsymbol{z}}\right) \\
&= \overline{\tilde{\boldsymbol{z}}} + \underset{\boldsymbol{z}}{\operatorname{argmin}} \ \frac{\beta}{2}\|\tilde{\boldsymbol{z}} - \overline{\tilde{\boldsymbol{z}}} - \boldsymbol{z}\|_2^2 - \log p\left(\boldsymbol{z}\right)
\end{aligned}
$$

Part 1/3: EPLL Algorithm (Zoran & Weiss, 2011)

Optimization on $\boldsymbol{z}$:

$$\hat{\boldsymbol{z}} \in \underset{\boldsymbol{z}}{\operatorname{argmin}} \frac{\beta}{2}\|\tilde{\boldsymbol{z}} - \boldsymbol{z}\|_2^2 - \log p\left(\boldsymbol{z} - \bar{\boldsymbol{z}}\right)$$

$$= \bar{\tilde{\boldsymbol{z}}} + \underset{\boldsymbol{z}}{\operatorname{argmin}} \frac{\beta}{2}\|\tilde{\boldsymbol{z}} - \bar{\tilde{\boldsymbol{z}}} - \boldsymbol{z}\|_2^2 - \log p\left(\boldsymbol{z}\right)$$

For the sake of simplicity consider

$$\hat{\boldsymbol{z}} \in \underset{\boldsymbol{z}}{\operatorname{argmin}} \frac{\beta}{2}\|\tilde{\boldsymbol{z}} - \boldsymbol{z}\|_2^2 - \log p\left(\boldsymbol{z}\right)$$

$$= \underset{\boldsymbol{z}}{\operatorname{argmin}} \frac{\beta}{2}\|\tilde{\boldsymbol{z}} - \boldsymbol{z}\|_2^2 - \log \sum_{k=1}^{K} w_k \exp\left(-\frac{1}{2}\boldsymbol{z}^t \boldsymbol{\Sigma}_k^{-1} \boldsymbol{z}\right) \qquad \text{(Non convex)}$$

$$\approx \underset{\boldsymbol{z}}{\operatorname{argmin}} \frac{\beta}{2}\|\tilde{\boldsymbol{z}} - \boldsymbol{z}\|_2^2 + \frac{1}{2}\boldsymbol{z}^t \boldsymbol{\Sigma}_{k^\star}^{-1} \boldsymbol{z} \qquad \text{(Keep only 1} \Rightarrow \text{Convex)}$$

$$= \left(\boldsymbol{\Sigma}_{k^\star} + \tfrac{1}{\beta}\mathbf{Id}_P\right)^{-1} \boldsymbol{\Sigma}_{k^\star} \tilde{\boldsymbol{z}} \qquad \text{(Explicit solution)}$$

How to choose the optimal $k^\star$?

Part 1/3: EPLL Algorithm (Zoran & Weiss, 2011)

Zoran & Weiss (2011) interpret

$$\underset{\boldsymbol{z}}{\operatorname{argmin}} \frac{\beta}{2}\|\tilde{\boldsymbol{z}} - \boldsymbol{z}\|_2^2 + \frac{1}{2}\boldsymbol{z}^t\Sigma_k^{-1}\boldsymbol{z}$$

as a MAP denoising problem where

$$\left.\begin{aligned} \tilde{\boldsymbol{z}} \mid \boldsymbol{z} &\sim \mathcal{N}(\boldsymbol{z}, \tfrac{1}{\beta}\mathbf{Id}_P) \\ \boldsymbol{z} \mid k &\sim \mathcal{N}(0_P, \boldsymbol{\Sigma}_k) \end{aligned}\right\} \xrightarrow{\text{Marginalization}} \tilde{\boldsymbol{z}} \mid k \sim \underbrace{\mathcal{N}(0_P, \boldsymbol{\Sigma}_k + \frac{1}{\beta}\mathbf{Id}_P)}_{= \mathcal{N}(0_P, \boldsymbol{\Sigma}_k) * \mathcal{N}(0_P, \frac{1}{\beta}\mathbf{Id}_P)}$$

(**convolution**)

Part 1/3: EPLL Algorithm (Zoran & Weiss, 2011)

Zoran & Weiss (2011) interpret

$$\underset{\boldsymbol{z}}{\text{argmin}} \ \frac{\beta}{2}\|\tilde{\boldsymbol{z}} - \boldsymbol{z}\|_2^2 + \frac{1}{2}\boldsymbol{z}^t \Sigma_k^{-1} \boldsymbol{z}$$

as a MAP denoising problem where

$$\left.\begin{aligned} \tilde{\boldsymbol{z}} \mid \boldsymbol{z} &\sim \mathcal{N}(\boldsymbol{z}, \tfrac{1}{\beta}\mathbf{Id}_P) \\ \boldsymbol{z} \mid k &\sim \mathcal{N}(0_P, \boldsymbol{\Sigma}_k) \end{aligned}\right\} \xrightarrow{\text{Marginalization}} \tilde{\boldsymbol{z}} \mid k \sim \underbrace{\mathcal{N}(0_P, \boldsymbol{\Sigma}_k + \frac{1}{\beta}\mathbf{Id}_P)}_{= \mathcal{N}(0_P, \boldsymbol{\Sigma}_k) * \mathcal{N}(0_P, \frac{1}{\beta}\mathbf{Id}_P)}$$

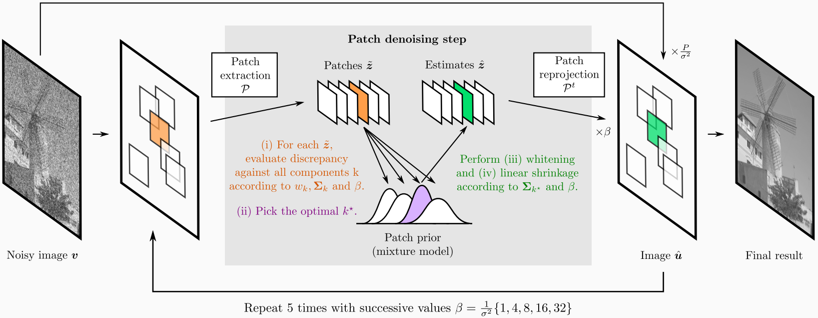
(**convolution**)

Choice of $k^\star$ by **maximum a posteriori**:

$$k^\star \in \underset{1 \leqslant k \leqslant K}{\text{argmax}} \ p(k \mid \tilde{\boldsymbol{z}}) = \underset{1 \leqslant k \leqslant K}{\text{argmax}} \ \mathbb{P}(k) p(\tilde{\boldsymbol{z}} \mid k) = \underset{1 \leqslant k \leqslant K}{\text{argmax}} \ w_k p(\tilde{\boldsymbol{z}} \mid k)$$

$$= \underset{1 \leqslant k \leqslant K}{\text{argmin}} \ \underbrace{-2 \log w_k + \log |\boldsymbol{\Sigma}_k + \frac{1}{\beta}\mathbf{Id}_P| + \tilde{\boldsymbol{z}}^t (\boldsymbol{\Sigma}_k + \frac{1}{\beta}\mathbf{Id}_P)^{-1} \tilde{\boldsymbol{z}}}_{\text{Discrepancy of patch } \tilde{\boldsymbol{z}} \text{ against component } k}$$

EPLL Algorithm



In practice:

- 5 iterations are used
- $\beta = \frac{1}{\sigma^2}\{1, 4, 8, 16, 32\}$

Part 1/3: EPLL Algorithm (Zoran & Weiss, 2011)

Algorithm: The five steps of an EPLL iteration

for all $1 \leqslant i \leqslant N$

$$\tilde{\boldsymbol{z}}_i \leftarrow \mathcal{P}_i \hat{\boldsymbol{x}} \qquad \text{(Patch extraction)}$$

$$k_i^\star \leftarrow \underset{1 \leqslant k_i \leqslant K}{\operatorname{argmin}} \; -2 \log w_{k_i} + \log \left| \boldsymbol{\Sigma}_{k_i} + \tfrac{1}{\beta} \mathbf{Id}_P \right| + \tilde{\boldsymbol{z}}_i^t \left(\boldsymbol{\Sigma}_{k_i} + \tfrac{1}{\beta} \mathbf{Id}_P \right)^{-1} \tilde{\boldsymbol{z}}_i \qquad \text{(Gaussian selection)}$$

$$\hat{\boldsymbol{z}}_i \leftarrow \left(\boldsymbol{\Sigma}_{k_i^\star} + \tfrac{1}{\beta} \mathbf{Id}_P \right)^{-1} \boldsymbol{\Sigma}_{k_i^\star} \tilde{\boldsymbol{z}}_i \qquad \text{(Patch estimation)}$$

$$\tilde{\boldsymbol{x}} \leftarrow \frac{1}{P} \sum_{i=1}^{N} \mathcal{P}_i^t \hat{\boldsymbol{z}}_i \qquad \text{(Patch reprojection)}$$

$$\hat{\boldsymbol{x}} \leftarrow \left(\mathcal{A}^t \mathcal{A} + \beta \sigma^2 \mathbf{Id}_N \right)^{-1} \left(\mathcal{A}^t \boldsymbol{y} + \beta \sigma^2 \tilde{\boldsymbol{x}} \right) \qquad \text{(Image estimation)}$$

Part 1/3: EPLL Algorithm (Zoran & Weiss, 2011)

Algorithm: The five steps of an EPLL iteration

for all $1 \leqslant i \leqslant N$

$\mathcal{O}(NP)$ $\quad \tilde{\boldsymbol{z}}_i \leftarrow \mathcal{P}_i \hat{\boldsymbol{x}}$ (Patch extraction)

$\mathcal{O}(NKP^2)$ $\quad k_i^\star \leftarrow \underset{1 \leqslant k_i \leqslant K}{\operatorname{argmin}} \; -2 \log w_{k_i} + \log \left| \boldsymbol{\Sigma}_{k_i} + \frac{1}{\beta} \mathbf{Id}_P \right| + \tilde{\boldsymbol{z}}_i^t \left(\boldsymbol{\Sigma}_{k_i} + \frac{1}{\beta} \mathbf{Id}_P \right)^{-1} \tilde{\boldsymbol{z}}_i$ (Gaussian selection)

$\mathcal{O}(NP^2)$ $\quad \hat{\boldsymbol{z}}_i \leftarrow \left(\boldsymbol{\Sigma}_{k_i^\star} + \frac{1}{\beta} \mathbf{Id}_P \right)^{-1} \boldsymbol{\Sigma}_{k_i^\star} \tilde{\boldsymbol{z}}_i$ (Patch estimation)

$\mathcal{O}(NP)$ $\quad \tilde{\boldsymbol{x}} \leftarrow \frac{1}{P} \sum_{i=1}^{N} \mathcal{P}_i^t \hat{\boldsymbol{z}}_i$ (Patch reprojection)

$\mathcal{O}(N \log N)$ $\quad \hat{\boldsymbol{x}} \leftarrow \left(\mathcal{A}^t \mathcal{A} + \beta \sigma^2 \mathbf{Id}_N \right)^{-1} \left(\mathcal{A}^t \boldsymbol{y} + \beta \sigma^2 \tilde{\boldsymbol{x}} \right)$ (Image estimation)

Global complexity: $\mathcal{O}(NKP^2)$

Part 1/3: EPLL Algorithm (Zoran & Weiss, 2011)

Gaussian selection represents 95% of computation time!

Algorithm 1 The five steps of an EPLL iteration		Time	Percentage
for all $i \in \mathcal{I}$			
$\tilde{\boldsymbol{z}}_i \leftarrow \mathcal{P}_i \boldsymbol{x}$	(Patch extraction)	0.46s	1 %
$k_i^\star \leftarrow \underset{1 \leqslant k_i \leqslant K}{\operatorname{argmin}} \log w_{k_i}^{-2} + \log \left\| \boldsymbol{\Sigma}_{k_i} + \frac{1}{\beta} \mathrm{Id}_P \right\| + \tilde{\boldsymbol{z}}_i^t \left(\boldsymbol{\Sigma}_{k_i} + \frac{1}{\beta} \mathrm{Id}_P \right)^{-1} \tilde{\boldsymbol{z}}_i$	(Gaussian selection)	43.53s	95 %
$\hat{\boldsymbol{z}}_i \leftarrow \left(\boldsymbol{\Sigma}_{k_i^\star} + \frac{1}{\beta} \mathrm{Id}_P \right)^{-1} \boldsymbol{\Sigma}_{k_i^\star} \tilde{\boldsymbol{z}}_i$	(Patch estimation)	0.95s	2 %
$\tilde{\boldsymbol{x}} \leftarrow \left(\sum_{i \in \mathcal{I}} \mathcal{P}_i^t \mathcal{P}_i \right)^{-1} \sum_{i \in \mathcal{I}} \mathcal{P}_i^t \hat{\boldsymbol{z}}_i$	(Patch reprojection)	0.23s	1 %
$\hat{\boldsymbol{x}} \leftarrow \left(\mathcal{A}^t \mathcal{A} + \beta \sigma^2 \mathrm{Id}_N \right)^{-1} \left(\mathcal{A}^t \boldsymbol{y} + \beta \sigma^2 \tilde{\boldsymbol{x}} \right)$	Others	0.52s	1 %
	Total	45.69s	

Part 1/3: EPLL Algorithm (Zoran & Weiss, 2011)

Gaussian selection represents 95% of computation time!

Algorithm 1 The five steps of an EPLL iteration		Time	Percentage
for all $i \in \mathcal{I}$			
$\tilde{\boldsymbol{z}}_i \leftarrow \mathcal{P}_i \boldsymbol{x}$	(Patch extraction)	0.46s	1 %
$k_i^\star \leftarrow \underset{1 \leqslant k_i \leqslant K}{\operatorname{argmin}} \; \log w_{k_i}^{-2} + \log \left\| \boldsymbol{\Sigma}_{k_i} + \frac{1}{\beta} \mathrm{Id}_P \right\| + \tilde{\boldsymbol{z}}_i^t \left(\boldsymbol{\Sigma}_{k_i} + \frac{1}{\beta} \mathrm{Id}_P \right)^{-1} \tilde{\boldsymbol{z}}_i$	(Gaussian selection)	43.53s	95 %
$\hat{\boldsymbol{z}}_i \leftarrow \left(\boldsymbol{\Sigma}_{k_i^\star} + \frac{1}{\beta} \mathrm{Id}_P \right)^{-1} \boldsymbol{\Sigma}_{k_i^\star} \tilde{\boldsymbol{z}}_i$	(Patch estimation)	0.95s	2 %
$\tilde{\boldsymbol{x}} \leftarrow \left(\sum_{i \in \mathcal{I}} \mathcal{P}_i^t \mathcal{P}_i \right)^{-1} \sum_{i \in \mathcal{I}} \mathcal{P}_i^t \hat{\boldsymbol{z}}_i$	(Patch reprojection)	0.23s	1 %
$\hat{\boldsymbol{x}} \leftarrow \left(\mathcal{A}^t \mathcal{A} + \beta \sigma^2 \mathrm{Id}_N \right)^{-1} \left(\mathcal{A}^t \boldsymbol{y} + \beta \sigma^2 \tilde{\boldsymbol{x}} \right)$	Others	0.52s	1 %
	Total	45.69s	

Fast EPLL (FEPLL):

- More than 100 times speedup.
- Contribution 1: stochastic patch sub-sampling.
- Contribution 2: flat tail approximation.
- Contribution 3: binary balanced search tree.

Contribution 1: consider only a subset $\mathcal{I} \subseteq [1, \ldots N]$ of patch indices!

Simple idea to accelerate the optimization on $\boldsymbol{z}_i$:

for all $i \in \mathcal{I}$

$\mathcal{O}(|\mathcal{I}|P)$ $\quad \tilde{\boldsymbol{z}}_i \leftarrow \mathcal{P}_i \hat{\boldsymbol{x}}$ (Patch extraction)

$\mathcal{O}(|\mathcal{I}|KP^2)$ $\quad k_i^\star \leftarrow \underset{1 \leqslant k_i \leqslant K}{\operatorname{argmin}} \; -2 \log w_{k_i} + \log \left| \boldsymbol{\Sigma}_{k_i} + \tfrac{1}{\beta} \mathbf{Id}_P \right| + \tilde{\boldsymbol{z}}_i^t \left(\boldsymbol{\Sigma}_{k_i} + \tfrac{1}{\beta} \mathbf{Id}_P \right)^{-1} \tilde{\boldsymbol{z}}_i$ (Gaussian selection)

$\mathcal{O}(|\mathcal{I}|P^2)$ $\quad \hat{\boldsymbol{z}}_i \leftarrow \left(\boldsymbol{\Sigma}_{k_i^\star} + \tfrac{1}{\beta} \mathbf{Id}_P \right)^{-1} \boldsymbol{\Sigma}_{k_i^\star} \tilde{\boldsymbol{z}}_i$ (Patch estimation)

Contribution 1: consider only a subset $\mathcal{I} \subseteq [1, \ldots N]$ of patch indices!

But, it slows down the optimization on $\boldsymbol{x}$:

$$\begin{aligned}
\hat{\boldsymbol{x}} &\in \underset{\boldsymbol{x}}{\operatorname{argmin}} \; \frac{P}{2\sigma^2}\|\mathcal{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \frac{\beta}{2}\sum_{i=1}^{N}\|\mathcal{P}_i\boldsymbol{x} - \hat{\boldsymbol{z}}_i\|_2^2 \\
&\approx \underset{\boldsymbol{x}}{\operatorname{argmin}} \; \frac{P}{2\sigma^2}\|\mathcal{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \frac{\beta}{2}\sum_{i\in\mathcal{I}}\|\mathcal{P}_i\boldsymbol{x} - \hat{\boldsymbol{z}}_i\|_2^2 \\
&= \Bigg(\mathcal{A}^t\mathcal{A} + \frac{\beta\sigma^2}{P}\underbrace{\sum_{i\in\mathcal{I}}\mathcal{P}_i^t\mathcal{P}_i}_{\text{diagonal but} \neq P\mathbf{Id}_N}\Bigg)^{-1}\Bigg(\mathcal{A}^t\boldsymbol{y} + \frac{\beta\sigma^2}{P}\sum_{i\in\mathcal{I}}\mathcal{P}_i^t\hat{\boldsymbol{z}}_i\Bigg)
\end{aligned}$$

- $(\sum_{i\in\mathcal{I}}\mathcal{P}_i^t\mathcal{P}_i)_{jj} = $ #patches covering pixel with index j
- The matrices $\mathcal{A}^t\mathcal{A}$ and $\sum_{i\in\mathcal{I}}\mathcal{P}_i^t\mathcal{P}_i$ do not share the same eigenspace,
- Inversion cannot be performed explicitly thanks to a fast transform,
- Use conjugate gradient $\Rightarrow$ slower than before.

Contribution 1: consider only a subset $\mathcal{I} \subseteq [1, \ldots N]$ of patch indices!

Alternative: **approximate the solution instead of the original problem**

$$\hat{\boldsymbol{x}} \in \underset{\boldsymbol{x}}{\operatorname{argmin}} \; \frac{P}{2\sigma^2}\|\mathcal{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \frac{\beta}{2}\sum_{i=1}^{N}\|\mathcal{P}_i\boldsymbol{x} - \hat{\boldsymbol{z}}_i\|_2^2$$

$$= \Bigg(\mathcal{A}^t\mathcal{A} + \frac{\beta\sigma^2}{P}\underbrace{\sum_{i=1}^{N}\mathcal{P}_i^t\mathcal{P}_i}_{P\mathbf{Id}_N}\Bigg)^{-1}\Bigg(\mathcal{A}^t\boldsymbol{y} + \frac{\beta\sigma^2}{P}\sum_{i=1}^{N}\mathcal{P}_i^t\hat{\boldsymbol{z}}_i\Bigg)$$

$$= \left(\mathcal{A}^t\mathcal{A} + \beta\sigma^2\mathbf{Id}_N\right)^{-1}\left(\mathcal{A}^t\boldsymbol{y} + \beta\sigma^2\tilde{\boldsymbol{x}}\right) \quad \text{with} \quad \tilde{\boldsymbol{x}} = \frac{1}{P}\sum_{i=1}^{N}\mathcal{P}_i^t\hat{\boldsymbol{z}}_i$$

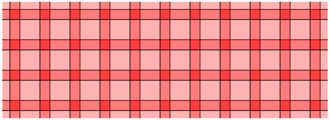
$$\approx \left(\mathcal{A}^t\mathcal{A} + \beta\sigma^2\mathbf{Id}_N\right)^{-1}\left(\mathcal{A}^t\boldsymbol{y} + \beta\sigma^2\tilde{\boldsymbol{x}}\right) \quad \text{with} \quad \tilde{\boldsymbol{x}} = \left(\sum_{i\in\mathcal{I}}\mathcal{P}_i^t\mathcal{P}_i\right)^{-1}\sum_{i\in\mathcal{I}}^{N}\mathcal{P}_i^t\hat{\boldsymbol{z}}_i$$

$\Rightarrow$ **Every other steps will be accelerated, and this step will be unchanged.**

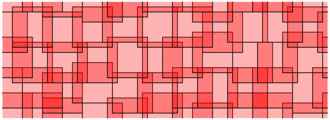
Contribution 1: consider only a subset $\mathcal{I} \subseteq [1, \ldots N]$ of patch indices!

How to sub-sample patches?

- Take every s pixels (acceleration s^2).
- Randomize the choice of the patches.
- All pixels must be covered at least once.
  $\Rightarrow$ max sub-sampling $s = P = 8$ (partition)
- All pixels must be covered by as many patches in average.
- Re-sample at each iteration.



(a) Regular patch sub-sampling



(b) Stochastic patch sub-sampling

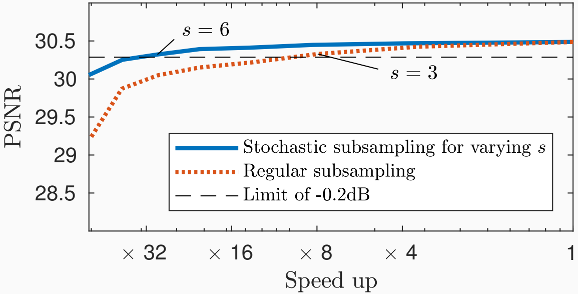
Part 2/3: Fast EPLL – Stochastic patch sub-sampling

Contribution 1: consider only a subset $\mathcal{I} \subseteq [1, \ldots N]$ of patch indices!

(a) Reference (b) Stoch. $s = 2$ (c) Stoch. $s = 4$ (d) Stoch. $s = 6$ (e) Stoch. $s = 8$

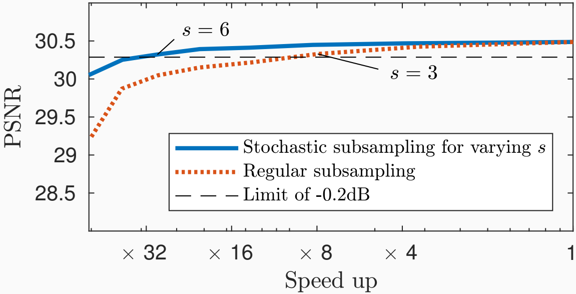
(f) Noisy (g) Regular $s = 2$ (h) Regular $s = 4$ (i) Regular $s = 6$ (j) Regular $s = 8$

Contribution 1: consider only a subset $\mathcal{I} \subseteq [1, \ldots N]$ of patch indices!



Complexity reduction: $\mathcal{O}(NP^2K) \rightarrow \mathcal{O}(NP^2K/s^2)$

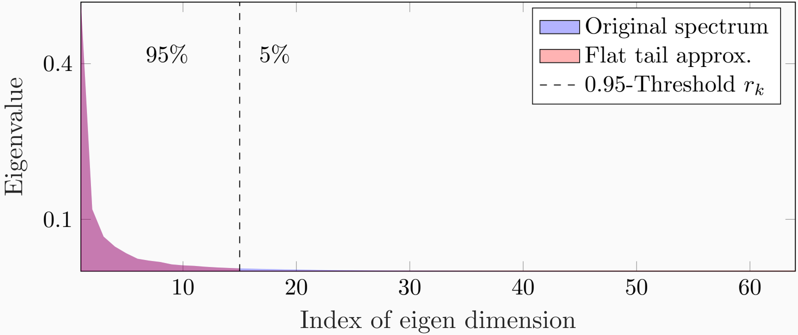
Contribution 1: consider only a subset $\mathcal{I} \subseteq [1, \dots N]$ of patch indices!



Complexity reduction: $\mathcal{O}(NP^2K) \rightarrow \mathcal{O}(NP^2K/s^2)$

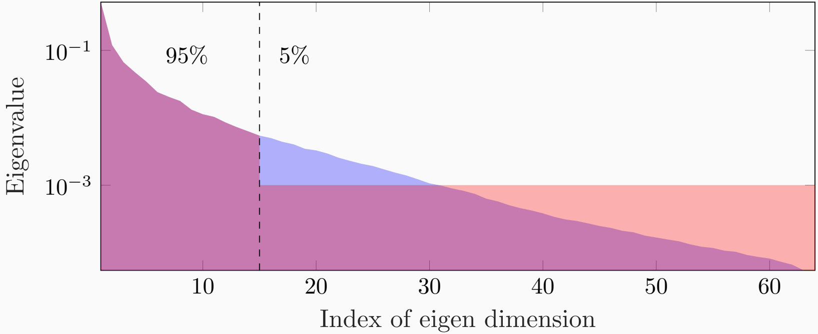
Can we reduce the term in P^2?

Contribution 2: approximate the spectrum of covariance matrices



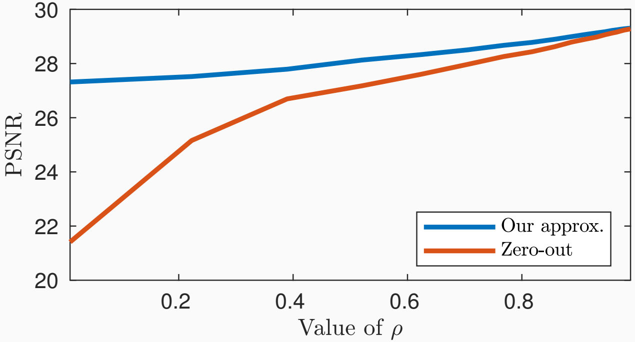
- Keep only $1 \leqslant r_k \leqslant P$ first eigen dimensions.
- Choose r_k to account for a proportion $\rho \in (0, 1]$ of the total variability.
- What to do with the other dimensions?

Contribution 2: approximate the spectrum of covariance matrices



- Do not set them to zero (low-rank approximation).
- Replace least eigenvalues by their average.

Contribution 2: approximate the spectrum of covariance matrices



- Do not set them to zero (low-rank approximation).
- Replace least eigenvalues by their average.
- Why does it help being faster?

Contribution 2: approximate the spectrum of covariance matrices

Recall we have to compute

$$k^\star \leftarrow \underset{1 \leqslant k \leqslant K}{\operatorname{argmin}} \; -2\log w_k + \log\left|\boldsymbol{\Sigma}_k + \tfrac{1}{\beta}\mathbf{Id}_P\right| + \tilde{\boldsymbol{z}}^t\left(\boldsymbol{\Sigma}_k + \tfrac{1}{\beta}\mathbf{Id}_P\right)^{-1}\tilde{\boldsymbol{z}}$$

$$\hat{\boldsymbol{z}} \leftarrow \left(\boldsymbol{\Sigma}_{k^\star} + \tfrac{1}{\beta}\mathbf{Id}_P\right)^{-1}\boldsymbol{\Sigma}_{k^\star}\tilde{\boldsymbol{z}}$$

Decompose $\boldsymbol{\Sigma}_k = \boldsymbol{U}_k \boldsymbol{\Lambda}_k \boldsymbol{U}_k^t$ with $\boldsymbol{\Lambda}_k = \operatorname{diag}(\lambda_{k,1}^2, \ldots \lambda_{k,P}^2)$, and $\boldsymbol{U}_k$ unitary.

$$\tilde{\boldsymbol{c}}_k \leftarrow \boldsymbol{U}_k^t \tilde{\boldsymbol{z}}, \quad \text{for all } 1 \leqslant k \leqslant K \qquad \mathcal{O}(P^2 K)$$

$$k^\star \leftarrow \underset{1 \leqslant k \leqslant K}{\operatorname{argmin}} \; -2\log w_k + \sum_{j=1}^{P}\left(\log(\lambda_{k,j}^2 + \tfrac{1}{\beta}) + \frac{\tilde{c}_{k,j}^2}{\lambda_{k,j}^2 + \frac{1}{\beta}}\right) \qquad \mathcal{O}(PK)$$

$$\hat{c}_j \leftarrow \frac{\lambda_{k^\star,j}^2}{\lambda_{k^\star,j}^2 + \frac{1}{\beta}}\tilde{c}_{k^\star,j}, \quad \text{for all } 1 \leqslant j \leqslant P \qquad \mathcal{O}(P)$$

$$\hat{\boldsymbol{z}} \leftarrow \boldsymbol{U}_{k^\star}\hat{\boldsymbol{c}} \qquad \mathcal{O}(P^2)$$

Contribution 2: approximate the spectrum of covariance matrices

Consider $\bar{\boldsymbol{U}} = \boldsymbol{U}_{:,1:r_k}$ with $r_k \leqslant P$ and $\lambda_{k,j} = \alpha_k$ for $r_k + 1 \leqslant j \leqslant P$

$$\tilde{\boldsymbol{c}}^k \leftarrow \bar{\boldsymbol{U}}_k^t \tilde{\boldsymbol{z}}, \quad \text{for all } 1 \leqslant k \leqslant K \qquad \mathcal{O}(P\bar{r}K)$$

$$k^\star \leftarrow \underset{1\leqslant k\leqslant K}{\operatorname{argmin}} \; -2\log w_k + (P-r)\log(\alpha_k^2 + \tfrac{1}{\beta}) + \frac{\|\tilde{\boldsymbol{z}}\|_2^2}{\alpha_k^2 + \frac{1}{\beta}}$$

$$+ \sum_{j=1}^{r_k} \left(\log(\lambda_{k,j}^2 + \tfrac{1}{\beta}) + \frac{\tilde{c}_{k,j}^2}{\lambda_{k,j}^2 + \frac{1}{\beta}} - \frac{\tilde{c}_{k,j}^2}{\alpha_k^2 + \frac{1}{\beta}} \right) \qquad \mathcal{O}(\bar{r}K)$$

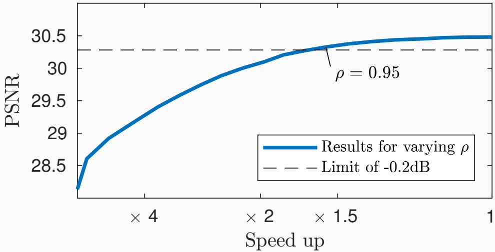
$$\hat{c}_j \leftarrow \left(\frac{\lambda_{k^\star,j}^2}{\lambda_{k^\star,j}^2 + \frac{1}{\beta}} - \frac{\alpha_{k^\star}^2}{\alpha_{k^\star}^2 + \frac{1}{\beta}} \right) \tilde{c}_{k^\star,j}, \quad \text{for all } 1 \leqslant j \leqslant r_{k^\star} \qquad \mathcal{O}(r_k)$$

$$\hat{\boldsymbol{z}} \leftarrow \bar{\boldsymbol{U}}_{k^\star} \hat{\boldsymbol{c}} + \frac{\alpha_{k^\star}^2}{\alpha_{k^\star}^2 + \frac{1}{\beta}} \tilde{\boldsymbol{z}} \qquad \mathcal{O}(Pr_k)$$

Complexity reduction: $\mathcal{O}(P^2K) \rightarrow \mathcal{O}(P\bar{r}K)$**, where** $\bar{r} = \frac{1}{K}\sum_{k=1}^K r_k$**.**

Part 2/3: Fast EPLL – Flat tail approximation

Contribution 2: approximate the spectrum of covariance matrices



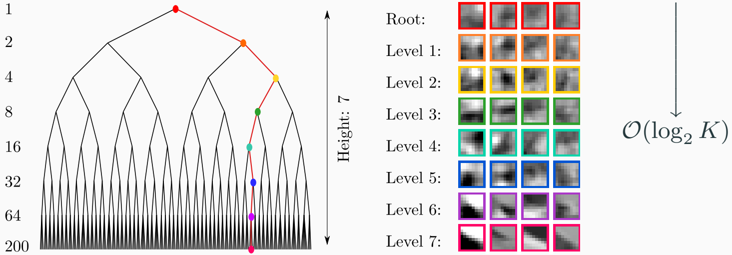
(a) Noisy (b) $\rho = 0.5$ (c) $\rho = 0.8$ (d) $\rho = 0.95$ (e) $\rho = 1$

Contribution 3: binary balanced search tree

- Avoid comparing each patch $\boldsymbol{z}_i$ against each of the K components

$$k_i^\star \leftarrow \underset{1 \leqslant k \leqslant K}{\text{argmin}} \; -2 \log w_k + \log \left| \boldsymbol{\Sigma}_k + \tfrac{1}{\beta} \mathbf{Id}_P \right| + \tilde{\boldsymbol{z}}_i^t \left(\boldsymbol{\Sigma}_k + \tfrac{1}{\beta} \mathbf{Id}_P \right)^{-1} \tilde{\boldsymbol{z}}_i$$

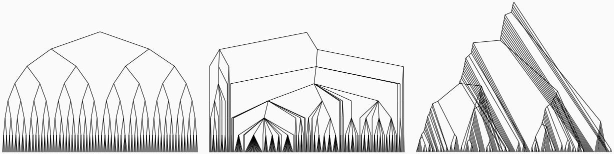
- Use a balanced (almost) binary search tree



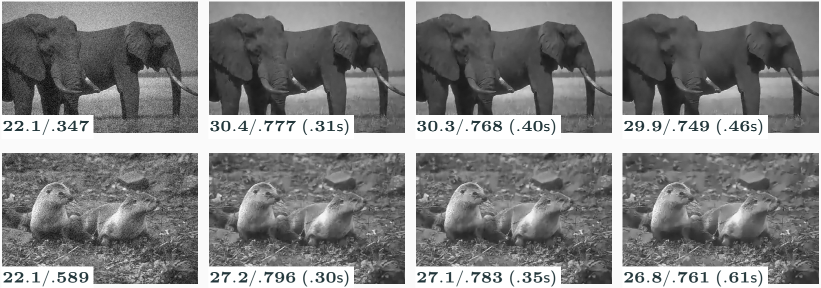
- Built by a bottom-up clustering strategy based on the Multiple Traveling Salesmen Problem (MTSP) solver proposed by (Kirk, 2014).

Part 2/3: Fast EPLL – Binary search tree

Contribution 3: binary balanced search tree



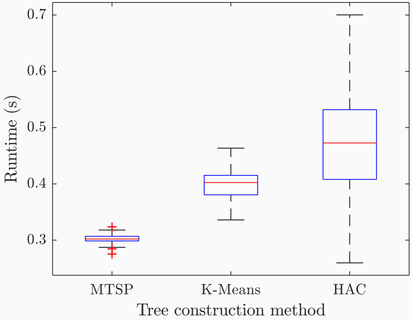
(a) height: 7 (b) height: 7 (c) height: 59



(d) Noisy (e) MTSP (f) K-Means (g) HAC

Part 2/3: Fast EPLL – Binary search tree

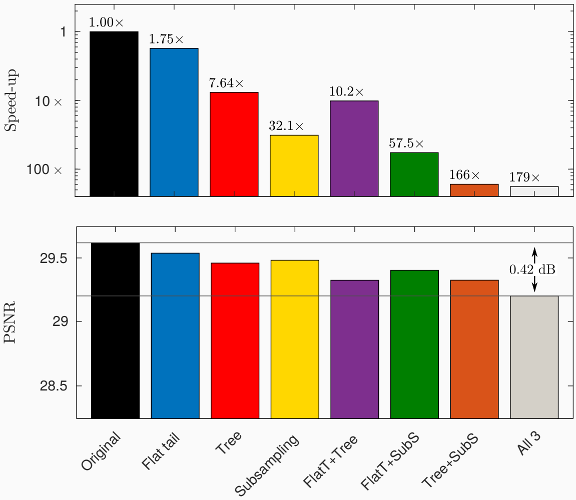
Contribution 3: binary balanced search tree



**Balanced is faster, and
computation time does not depend on the image content.**

It also provides better results!

More than $100\times$ speed-up obtained due to the 3 proposed accelerations



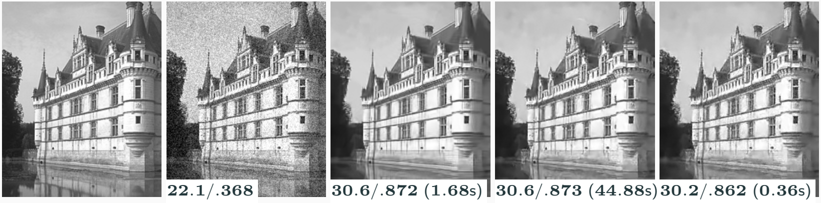
More than $100\times$ speed-up obtained due to the 3 proposed accelerations

Algorithm 1 The five steps of an EPLL iteration		Without accelerations		With the proposed accelerations	
for all $i \in \mathcal{I}$					
$\tilde{\boldsymbol{z}}_i \leftarrow \mathcal{P}_i \boldsymbol{x}$	(Patch extraction)	0.46s	1 %	0.03s	7 %
$k_i^\star \leftarrow \underset{1 \leqslant k_i \leqslant K}{\operatorname{argmin}} \log w_{k_i}^{-2} + \log \left\| \boldsymbol{\Sigma}_{k_i} + \frac{1}{\beta} \mathrm{Id}_P \right\| + \tilde{\boldsymbol{z}}_i^t \left(\boldsymbol{\Sigma}_{k_i} + \frac{1}{\beta} \mathrm{Id}_P \right)^{-1} \tilde{\boldsymbol{z}}_i$	(Gaussian selection)	43.53s	95 %	0.23s	66 %
$\hat{\boldsymbol{z}}_i \leftarrow \left(\boldsymbol{\Sigma}_{k_i^\star} + \frac{1}{\beta} \mathrm{Id}_P \right)^{-1} \boldsymbol{\Sigma}_{k_i^\star} \tilde{\boldsymbol{z}}_i$	(Patch estimation)	0.95s	2 %	0.05s	13 %
$\tilde{\boldsymbol{x}} \leftarrow \left(\sum_{i \in \mathcal{I}} \mathcal{P}_i^t \mathcal{P}_i \right)^{-1} \sum_{i \in \mathcal{I}} \mathcal{P}_i^t \hat{\boldsymbol{z}}_i$	(Patch reprojection)	0.23s	1 %	0.01s	4 %
$\hat{\boldsymbol{x}} \leftarrow \left(\mathcal{A}^t \mathcal{A} + \beta \sigma^2 \mathrm{Id}_N \right)^{-1} \left(\mathcal{A}^t \boldsymbol{y} + \beta \sigma^2 \tilde{\boldsymbol{x}} \right)$	Others	0.52s	1 %	0.03s	10 %
	Total	45.69s		0.35s	

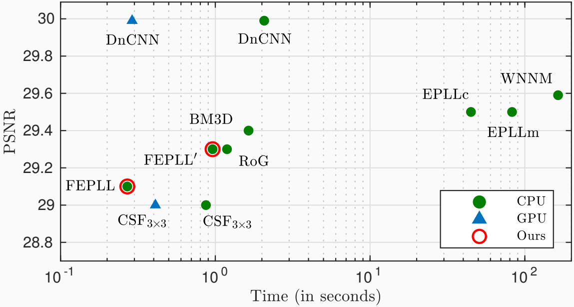
Complexity reduction: $\mathcal{O}(NP^2K) \rightarrow \mathcal{O}(NP\bar{r} \log_2 K / s^2)$

- N image size
- $P = 8 \times 8$
- $K = 200$
- $\lfloor \log_2 K \rfloor = 7$
- $s^2 = 36$
- $\bar{r} = 19.6$ $(\rho = .95)$

Part 2/3: Fast EPLL



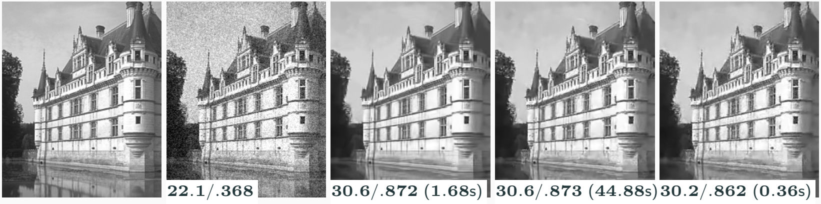
(a) Reference $\boldsymbol{x}$ (b) Noisy image $\boldsymbol{y}$ (c) BM3D $\hat{\boldsymbol{x}}$ (d) EPLLc $\hat{\boldsymbol{x}}$ (e) FEPLL $\hat{\boldsymbol{x}}$



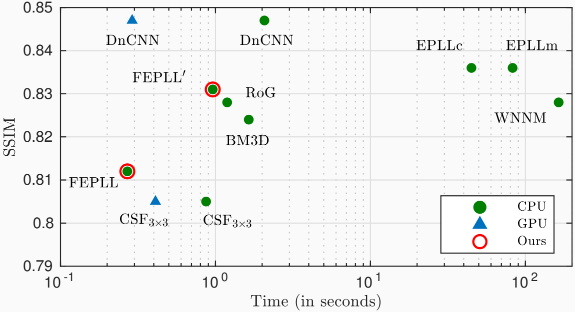
Averaged on 60 images of the BSDS test data-set.

Noise standard deviation $\sigma = 20$.

Part 2/3: Fast EPLL

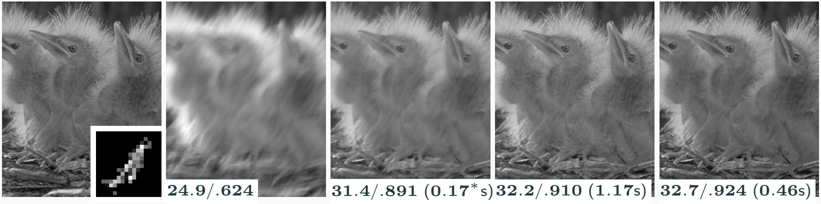


(a) Reference $\boldsymbol{x}$ (b) Noisy image $\boldsymbol{y}$ (c) BM3D $\hat{\boldsymbol{x}}$ (d) EPLLc $\hat{\boldsymbol{x}}$ (e) FEPLL $\hat{\boldsymbol{x}}$



Averaged on 60 images of the BSDS test data-set.

Noise standard deviation $\sigma = 20$.

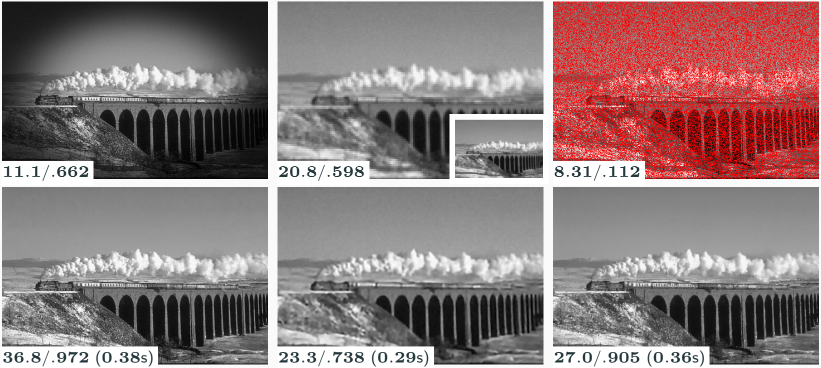


(a) Ref $\boldsymbol{x}$ / kernel (b) Blurry image $\boldsymbol{y}$ (c) CSF result $\hat{\boldsymbol{x}}$ (d) RoG result $\hat{\boldsymbol{x}}$ (e) FEPLL result $\hat{\boldsymbol{x}}$

Algo.	Berkeley		Classic	
	PSNR/SSIM	Time (s)	PSNR/SSIM	Time (s)
iPiano	29.5 / .824	29.53	29.9 / .848	59.10
CSF_{pw}	30.2 / .875	0.50 (0.14*)	30.5 / 0.870	0.47 (0.14*)
RoG	31.3 / .897	1.19	31.8 / .915	2.07
FEPLL	33.1 / .928	0.40	32.8 / .931	0.46
FEPLL'	33.2 / .930	1.01	33.0 / .933	1.82

Using the blur kernel of iPiano and noise standard deviation $\sigma = 0.5$.

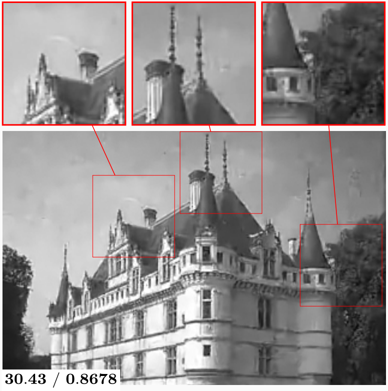
Part 2/3: Fast EPLL



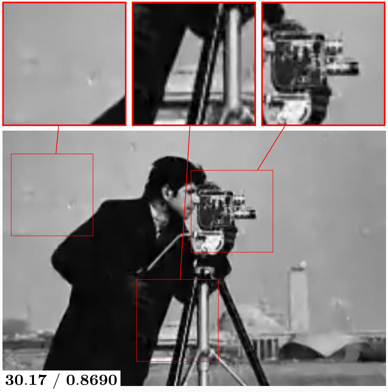
(a) devignetting (b) ×3 super-resolution (c) 50% inpainting

- Works likewise for several inverse problems.
- Less than 0.4s in all cases (for images of size 481×321).
- Out-of-the-box: no need to adjust/tune hyperparameters.
- NB: Only for 8-bits pictures (need to learn a new model otherwise).

Is the patch distribution well modeled by a GMM distribution?



GMM

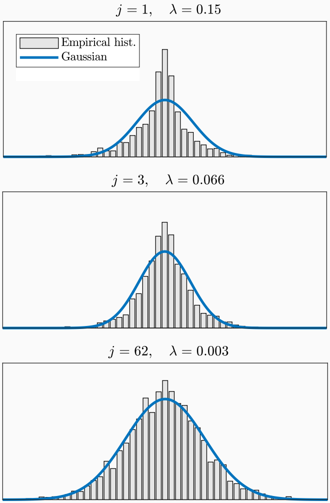


GMM

- EPLL (and FEPLL) presents many artifacts similar to Gibbs artifacts.
- Not really robust to outliers.
- Could it be due to the assumption that patches are GMM distributed?

Part 3/3: GGMM-EPLL

Let us have a look at the empirical distribution of a cluster of clean patches along some axis of its corresponding covariance matrix.



What alternative to the Gaussian distribution?

(Zero-mean) Generalized Gaussian Distribution (GGD)

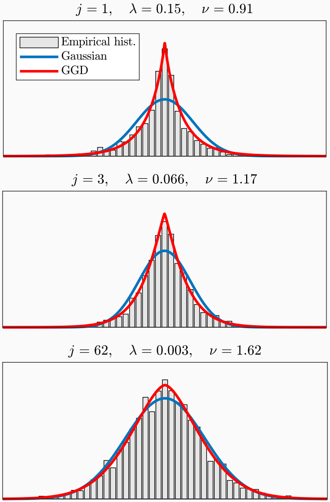
- Coefficients are zero mean.
- Some coefficients have a bell shaped distribution.
- Some others have a peaky distribution with large tails.
- A Generalized Gaussian Distribution (GGD) captures all of these

$$\mathcal{G}(z; 0, \lambda, \nu) = \frac{\kappa_\nu}{2\lambda_\nu} \exp\left[-\left(\frac{|z|}{\lambda_\nu} \right)^\nu \right]$$

where $\kappa_\nu = \dfrac{\nu}{\Gamma(1/\nu)}$ and $\lambda_\nu = \lambda \sqrt{\dfrac{\Gamma(1/\nu)}{\Gamma(3/\nu)}}$,

- λ: scale parameter (standard deviation),
- ν: shape parameter ($\nu = 2$: Gaussian, $\nu = 1$: Laplacian).

What if we look for ν that best fits?



Part 3/3: GGMM-EPLL

What about multi-variate GGD?

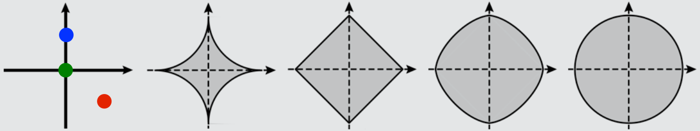
$$\mathcal{G}(\boldsymbol{z}; 0_P, \boldsymbol{\Sigma}, \boldsymbol{\nu}) = \frac{\mathcal{K}_{\boldsymbol{\nu}}}{2|\boldsymbol{\Sigma}_{\boldsymbol{\nu}}|^{1/2}} \exp\left[-\|\boldsymbol{\Sigma}_{\boldsymbol{\nu}}^{-1/2}\boldsymbol{z}\|_{\boldsymbol{\nu}}^{\boldsymbol{\nu}}\right] \quad \text{with} \quad \|\boldsymbol{x}\|_{\boldsymbol{\nu}}^{\boldsymbol{\nu}} = \sum_{j=1}^{P} |x_j|^{\nu_j},$$

$$\text{where} \quad \mathcal{K}_{\boldsymbol{\nu}} = \prod_{j=1}^{P} \frac{\nu_j}{\Gamma(1/\nu_j)} \quad \text{and} \quad \boldsymbol{\Sigma}_{\boldsymbol{\nu}}^{1/2} = \boldsymbol{U}\boldsymbol{\Lambda}^{1/2} \begin{pmatrix} \sqrt{\frac{\Gamma(1/\nu_1)}{\Gamma(3/\nu_1)}} & & \\ & \ddots & \\ & & \sqrt{\frac{\Gamma(1/\nu_P)}{\Gamma(3/\nu_P)}} \end{pmatrix}.$$

- ℓ_ρ prior: $\|\boldsymbol{x}\|_\rho^\rho = \sum_k |x_k|^\rho$
- convexity: $\rho \geqslant 1$
- sparsity: $\rho \leqslant 1$

2-dim vector:

$\|\boldsymbol{x}\|_0 = 0$	null image ●
$\|\boldsymbol{x}\|_0 = 1$	sparse image ●
$\|\boldsymbol{x}\|_0 = 2$	dense image ●



(Source: G. Peyré)

GMM

Assumption about a clean image patch:

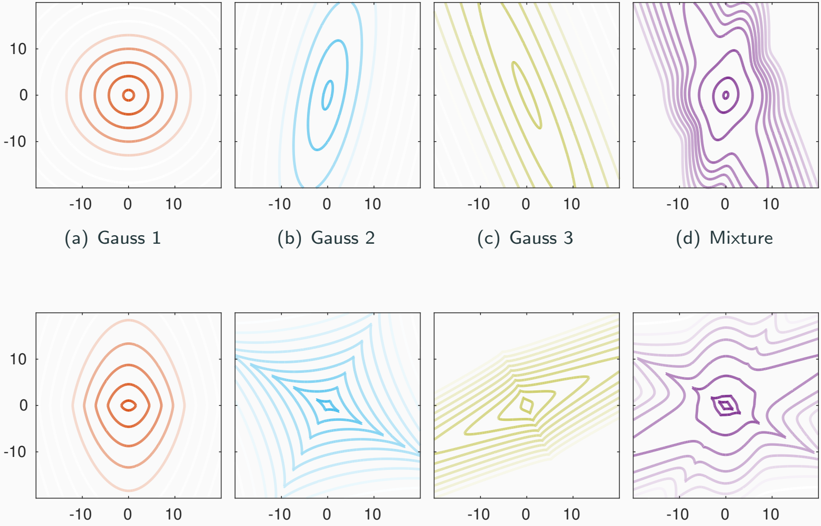
- Lies in one of the K ellipsoidal clusters (let us say the k-th).
- Dense linear combinations of the columns of $\boldsymbol{U}_k$.
- Coefficients for all directions j are likely in the range $[-2\lambda_{k,j}, 2\lambda_{k,j}]$.

GGMM

$$p(\boldsymbol{z}) = \sum_{k=1}^{K} w_k \mathcal{G}(\boldsymbol{z}; 0_P, \boldsymbol{\Sigma}_k, \boldsymbol{\nu}_k)$$

- Clusters have ellipsoidal ($\nu_{k,j} > 1$) or star shaped ($\nu_{k,j} \leqslant 1$) directions.
- Dense ($\nu_{k,j} > 1$) or sparse ($\nu_{k,j} \leqslant 1$) combinations of the columns of $\boldsymbol{U}_k$.
- Few coefficients for a given direction j can be outliers ($\nu_{k,j} < 1$).
- Behavior can be different for different directions within a same cluster.

Part 3/3: GGMM-EPLL

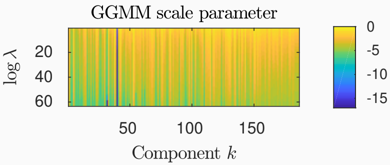


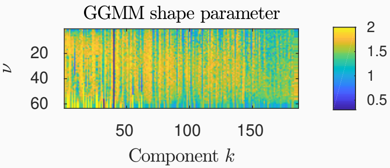
(a) Gauss 1 (b) Gauss 2 (c) Gauss 3 (d) Mixture

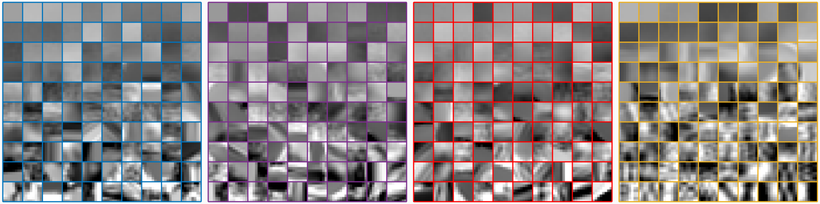
(e) GGD 1 (f) GGD 2 (g) GGD 3 (h) Mixture

Part 3/3: GGMM-EPLL

Parameters (Σ_k, ν_k) estimated by Expectation-Maximization on a training set of 2 million clean 8×8 patches.



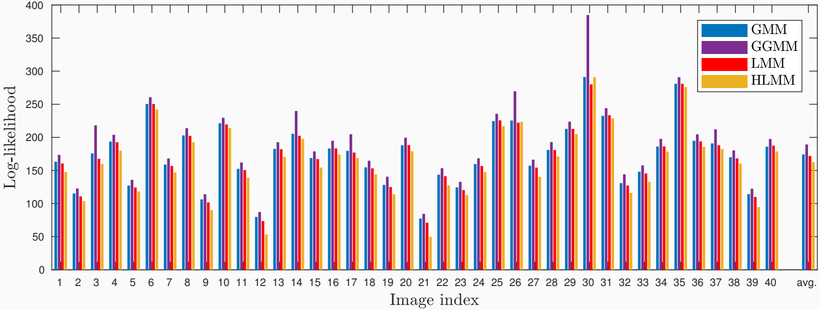




(a) GMM ($\nu = 2$) (b) GGMM ($.3 \leqslant \nu \leqslant 2$) (c) LMM ($\nu = 1$) (d) HLMM ($\nu = .5$)

Set of 100 generated random patches for each model.

Parameters $(\boldsymbol{\Sigma}_k, \boldsymbol{\nu}_k)$ estimated by Expectation-Maximization on a training set of 2 million clean 8×8 patches.



- GGMM consistently fits best patches of each images of the testing set.
- Adding an extra degree of freedom (shape ν) did not lead to overfitting.

How to extend EPLL to GGMM patch priors?

- EPLL uses the Gaussian clusters through two equations:

$$k^\star \leftarrow \underset{1 \leqslant k \leqslant K}{\operatorname{argmin}} \; -2 \log w_k + 2 \sum_{j=1}^{P} \underbrace{\left(\frac{1}{2} \log(\lambda_{k,j}^2 + \tfrac{1}{\beta}) + \frac{1}{2} \frac{\tilde{c}_{k,j}^2}{\lambda_{k,j}^2 + \frac{1}{\beta}} \right)}_{= f(\tilde{c}_{k,j}; 1/\beta, \lambda_{k,j})}$$

$$\hat{c}_j \leftarrow \underbrace{\frac{\lambda_{k^\star,j}^2}{\lambda_{k^\star,j}^2 + \frac{1}{\beta}} \tilde{c}_{k^\star,j}}_{s(\tilde{c}_{k^\star,j}; 1/\beta, \lambda_{k^\star,j})}, \qquad \text{for all } 1 \leqslant j \leqslant P$$

How to extend EPLL to GGMM patch priors?

- EPLL uses the Gaussian clusters through two equations:

$$k^\star \leftarrow \underset{1 \leqslant k \leqslant K}{\operatorname{argmin}} \; - 2 \log w_k + 2 \sum_{j=1}^{P} \underbrace{\left(\frac{1}{2} \log(\lambda_{k,j}^2 + \tfrac{1}{\beta}) + \frac{1}{2} \frac{\tilde{c}_{k,j}^2}{\lambda_{k,j}^2 + \frac{1}{\beta}} \right)}_{= f(\tilde{c}_{k,j}; 1/\beta, \lambda_{k,j})}$$

$$\hat{c}_j \leftarrow \underbrace{\frac{\lambda_{k^\star,j}^2}{\lambda_{k^\star,j}^2 + \frac{1}{\beta}} \tilde{c}_{k^\star,j}}_{s(\tilde{c}_{k^\star,j}; 1/\beta, \lambda_{k^\star,j})}, \qquad \text{for all } 1 \leqslant j \leqslant P$$

- Where f and s were arising from:

$$f(x; \sigma, \lambda) = \log \int_{\mathbb{R}} \frac{1}{2\pi\sigma\lambda} \exp\left(-\frac{(t-x)^2}{2\sigma^2} - \frac{t^2}{2\lambda^2} \right) \mathrm{d}t$$

$$s(x; \sigma, \lambda) \in \underset{t \in \mathbb{R}}{\operatorname{argmin}} \frac{(t-x)^2}{2\sigma^2} + \frac{t^2}{2\lambda^2}$$

How to extend EPLL to GGMM patch priors?

- EPLL uses the Gaussian clusters through two equations:

$$k^\star \leftarrow \underset{1 \leqslant k \leqslant K}{\operatorname{argmin}} \; -2 \log w_k + 2 \sum_{j=1}^{P} f(\tilde{c}_{k,j}; 1/\beta, \lambda_{k,j})$$

$$\hat{c}_j \leftarrow s(\tilde{c}_{k^\star,j}; 1/\beta, \lambda_{k^\star,j}), \qquad \text{for all } 1 \leqslant j \leqslant P$$

- Where f and s were arising from:

$$f(x; \sigma, \lambda) = \log \int_{\mathbb{R}} \frac{1}{2\pi\sigma\lambda} \exp\left(-\frac{(t-x)^2}{2\sigma^2} - \frac{t^2}{2\lambda^2}\right) \mathrm{d}t$$

$$s(x; \sigma, \lambda) \in \underset{t \in \mathbb{R}}{\operatorname{argmin}} \frac{(t-x)^2}{2\sigma^2} + \frac{t^2}{2\lambda^2}$$

How to extend EPLL to GGMM patch priors?

- EPLL can be extended to GGD by updating the two equations as:

$$k^\star \leftarrow \underset{1 \leqslant k \leqslant K}{\operatorname{argmin}} \; - 2 \log w_k + 2 \sum_{j=1}^{P} f(\tilde{c}_{k,j}; 1/\beta, \lambda_{k,j}, \nu_{k,j})$$

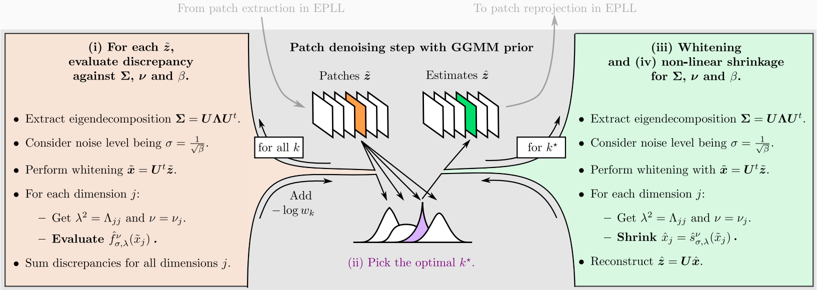
$$\hat{c}_j \leftarrow s(\tilde{c}_{k^\star,j}; 1/\beta, \lambda_{k^\star,j}, \nu_{k^\star,j}), \qquad \text{for all } 1 \leqslant j \leqslant P$$

- Where f and s can be updated as:

$$f(x; \sigma, \lambda, \nu) = \log \int_{\mathbb{R}} \frac{1}{\sqrt{2\pi}\sigma} \frac{\kappa_\nu}{2\lambda_\nu} \exp\left(- \frac{(t-x)^2}{2\sigma^2} - \frac{|t|^\nu}{\lambda_\nu^\nu} \right) \mathrm{d}t$$

$$s(x; \sigma, \lambda, \nu) \in \underset{t \in \mathbb{R}}{\operatorname{argmin}} \; \frac{(t-x)^2}{2\sigma^2} + \frac{|t|^\nu}{\lambda_\nu^\nu}$$

GGMM-EPLL Algorithm



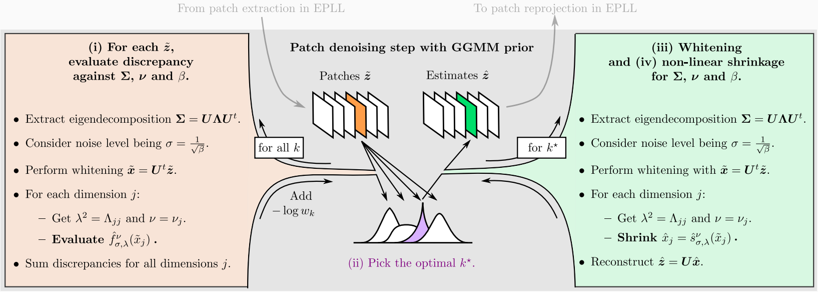
- Discrepancy function:

$$f^{\nu}_{\sigma,\lambda}(x) = \log \int_{\mathbb{R}} \frac{1}{\sqrt{2\pi}\sigma} \frac{\kappa_\nu}{2\lambda_\nu} \exp\left(-\frac{(t-x)^2}{2\sigma^2} - \frac{|t|^\nu}{\lambda^\nu_\nu} \right) \mathrm{d}t$$

- Shrinkage function:

$$s^{\nu}_{\sigma,\lambda}(x) \in \operatorname*{argmin}_{t \in \mathbb{R}} \frac{(t-x)^2}{2\sigma^2} + \frac{|t|^\nu}{\lambda^\nu_\nu}$$

GGMM-EPLL Algorithm



- Discrepancy function:

$$f^{\nu}_{\sigma,\lambda}(x) = \log \int_{\mathbb{R}} \frac{1}{\sqrt{2\pi}\sigma} \frac{\kappa_\nu}{2\lambda_\nu} \exp\left(-\frac{(t-x)^2}{2\sigma^2} - \frac{|t|^\nu}{\lambda_\nu^\nu} \right) \mathrm{d}t$$

- Shrinkage function: **Closed-form?**

$$s^{\nu}_{\sigma,\lambda}(x) \in \operatorname*{argmin}_{t \in \mathbb{R}} \frac{(t-x)^2}{2\sigma^2} + \frac{|t|^\nu}{\lambda_\nu^\nu}$$

No closed-forms but we can evaluate the integral and solve the optimization with numerical techniques.

(a) No approx. (10h 29m)

(b) Approximations (1s63)

Really slow, even for a 128×128 image!

Proposed approximations will lead to a speed-up of $\times 15,000$.

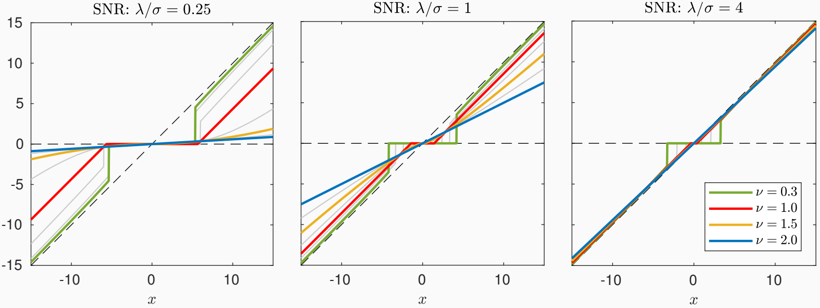
Shrinkage functions

ν	Shrinkage $s^{\nu}_{\sigma,\lambda}(x)$	Remark
< 1	$\begin{cases} x - \gamma x^{\nu-1} + O(x^{2(\nu-1)}) & \text{if } \lvert x\rvert \geqslant \tau^{\nu}_{\lambda} \\ 0 & \text{otherwise} \end{cases}$	$\approx$ Hard-thresholding [Moulin, 1999]
1	$\operatorname{sign}(x) \max\left(\lvert x\rvert - \frac{\sqrt{2}\sigma^2}{\lambda}, 0\right)$	Soft-thresholding [Donoho, 1994]
$4/3$	$x + \gamma\left(\sqrt[3]{\frac{\zeta - x}{2}} - \sqrt[3]{\frac{\zeta + x}{2}}\right)$	[Chaux et al., 2007]
$3/2$	$\operatorname{sign}(x) \frac{\left(\sqrt{\gamma^2 + 4\lvert x\rvert} - \gamma\right)^2}{4}$	[Chaux et al., 2007]
2	$\frac{\lambda^2}{\lambda^2 + \sigma^2} \cdot x$	Wiener (LMMSE)
Otherwise	No closed-forms	

with $\gamma = \nu \sigma^2 \lambda_\nu^{-\nu}$ and $\zeta = \sqrt{x^2 + 4\left(\frac{\gamma}{3}\right)^3}$.

and $\tau^{\nu}_{\lambda} = (2-\nu)(2-2\nu)^{-\frac{1-\nu}{2-\nu}} (\sigma^2 \lambda_\nu^{-\nu})^{\frac{1}{2-\nu}}$

Shrinkage functions



Properties:

$$s^{\nu}_{\sigma,\lambda}(x) = \sigma s^{\nu}_{1,\frac{\lambda}{\sigma}}\left(\frac{x}{\sigma}\right) \quad \text{(reduction)}$$

$$s^{\nu}_{\sigma,\lambda}(x) = -s^{\nu}_{\sigma,\lambda}(-x) \quad \text{(odd)}$$

$$s^{\nu}_{\sigma,\lambda}(x) \in \begin{cases} [0, x] & \text{if} \quad x \geqslant 0 \\ [x, 0] & \text{otherwise} \end{cases} \quad \text{(shrinkage)}$$

$$x \mapsto s^{\nu}_{\sigma,\lambda}(x) \text{ increasing} \quad \text{(increasing with } x\text{)}$$

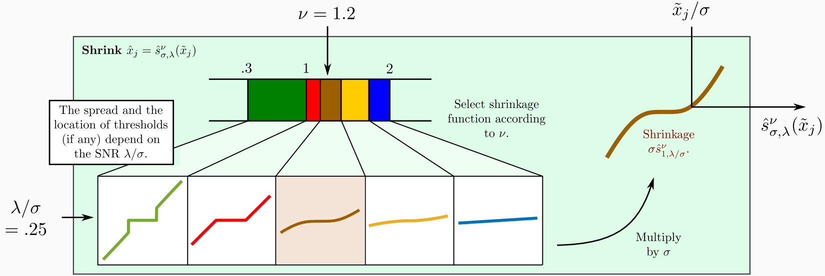
$$\lambda \mapsto s^{\nu}_{\sigma,\lambda}(x) \text{ increasing} \quad \text{(increasing with } \lambda\text{)}$$

$$\lim_{\frac{\lambda}{\sigma} \to 0} s^{\nu}_{1,\frac{\lambda}{\sigma}}(x) = 0 \quad \text{(kill low SNR)}$$

$$\lim_{\frac{\lambda}{\sigma} \to +\infty} s^{\nu}_{1,\frac{\lambda}{\sigma}}(x) = x \quad \text{(keep high SNR)}$$

Shrinkage functions

$$s^{\nu}_{\sigma,\lambda}(x) \in \underset{t \in \mathbb{R}}{\operatorname{argmin}} \frac{(t-x)^2}{2\sigma^2} + \frac{|t|^{\nu}}{\lambda^{\nu}_{\nu}}$$



Choose one of the closed-form expressions by nearest neighbor on ν.

Discrepancy functions

$$f^{\nu}_{\sigma,\lambda}(x) = \log \int_{\mathbb{R}} \frac{1}{\sqrt{2\pi}\sigma} \frac{\kappa_\nu}{2\lambda_\nu} \exp\left(-\frac{(t-x)^2}{2\sigma^2} - \frac{|t|^\nu}{\lambda_\nu^\nu} \right) \mathrm{d}t$$

Properties

$$f^{\nu}_{\sigma,\lambda}(x) = \log \sigma + f^{\nu}_{1,\lambda/\sigma}(x/\sigma) \ , \qquad \text{(reduction)}$$

$$f^{\nu}_{\sigma,\lambda}(x) = f^{\nu}_{\sigma,\lambda}(-x) \ , \qquad \text{(even)}$$

$$|x| \geqslant |y| \Leftrightarrow f^{\nu}_{\sigma,\lambda}(|x|) \geqslant f^{\nu}_{\sigma,\lambda}(|y|) \ , \qquad \text{(unimodality)}$$

$$\min_{x \in \mathbb{R}} f^{\nu}_{\sigma,\lambda}(x) = f^{\nu}_{\sigma,\lambda}(0) > -\infty \ . \qquad \text{(lower bound at 0)}$$

Discrepancy functions

$$f^\nu_{\sigma,\lambda}(x) = \log \int_{\mathbb{R}} \frac{1}{\sqrt{2\pi}\sigma} \frac{\kappa_\nu}{2\lambda_\nu} \exp\left(-\frac{(t-x)^2}{2\sigma^2} - \frac{|t|^\nu}{\lambda_\nu^\nu} \right) \mathrm{d}t$$

Properties

$$f^\nu_{\sigma,\lambda}(x) = \log \sigma + f^\nu_{1,\lambda/\sigma}(x/\sigma) \ , \qquad \text{(reduction)}$$

$$f^\nu_{\sigma,\lambda}(x) = f^\nu_{\sigma,\lambda}(-x) \ , \qquad \text{(even)}$$

$$|x| \geqslant |y| \Leftrightarrow f^\nu_{\sigma,\lambda}(|x|) \geqslant f^\nu_{\sigma,\lambda}(|y|) \ , \qquad \text{(unimodality)}$$

$$\min_{x \in \mathbb{R}} f^\nu_{\sigma,\lambda}(x) = f^\nu_{\sigma,\lambda}(0) > -\infty \ . \qquad \text{(lower bound at 0)}$$

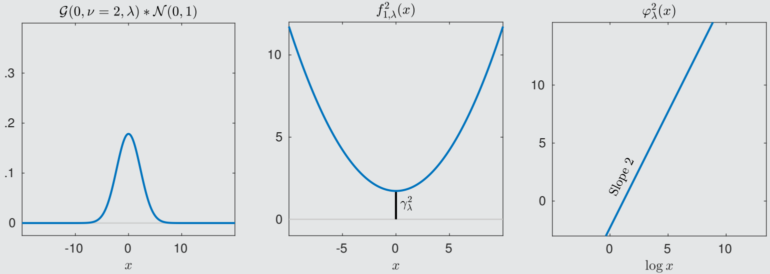
$\Rightarrow$ **Consider instead the log-discrepancy function φ^ν_λ:**

$$\varphi^\nu_\lambda(|x|) = \log \left[f^\nu_{1,\lambda}(x) - \gamma^\nu_\lambda \right] \quad \text{and} \quad \gamma^\nu_\lambda = f^\nu_{1,\lambda}(0) \ .$$

Discrepancy functions

$$f^{\nu}_{\sigma,\lambda}(x) = \log \int_{\mathbb{R}} \frac{1}{\sqrt{2\pi}\sigma} \frac{\kappa_\nu}{2\lambda_\nu} \exp\left(-\frac{(t-x)^2}{2\sigma^2} - \frac{|t|^\nu}{\lambda_\nu^\nu} \right) \mathrm{d}t$$

Case $\nu = 2$



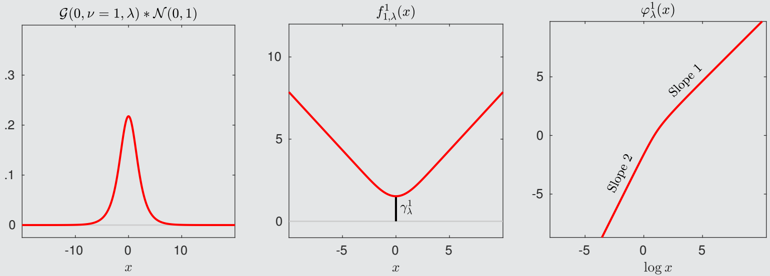
$$\varphi^2_\lambda(x) = \alpha \log x + \beta \ ,$$

where $\quad \alpha = 2 \quad$ and $\quad \beta = -\log 2 - \log(1 + \lambda^2)$.

Discrepancy functions

$$f^{\nu}_{\sigma,\lambda}(x) = \log \int_{\mathbb{R}} \frac{1}{\sqrt{2\pi}\sigma} \frac{\kappa_\nu}{2\lambda_\nu} \exp\left(-\frac{(t-x)^2}{2\sigma^2} - \frac{|t|^\nu}{\lambda_\nu^\nu}\right) \mathrm{d}t$$

Case $\nu = 1$



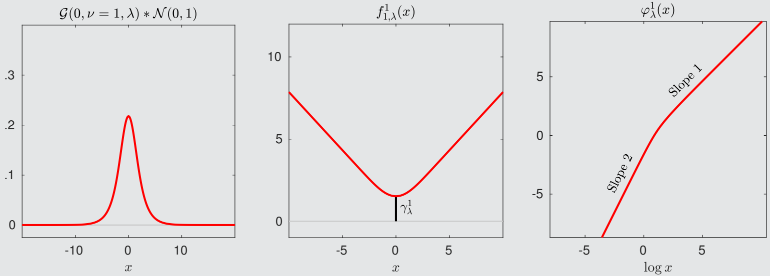
$$\varphi^1_\lambda(x) \underset{0}{\sim} \alpha_1 \log x + \beta_1 \ ,$$

where $\alpha_1 = 2$ and $\beta_1 = -\log\lambda + \log\left[\frac{1}{\sqrt{\pi}} \frac{\exp\left(-\frac{1}{\lambda^2}\right)}{\operatorname{erfc}\left(\frac{1}{\lambda}\right)} - \frac{1}{\lambda}\right]$.

Discrepancy functions

$$f^{\nu}_{\sigma,\lambda}(x) = \log \int_{\mathbb{R}} \frac{1}{\sqrt{2\pi}\sigma} \frac{\kappa_\nu}{2\lambda_\nu} \exp\left(-\frac{(t-x)^2}{2\sigma^2} - \frac{|t|^\nu}{\lambda_\nu^\nu} \right) \mathrm{d}t$$

Case $\nu = 1$



$$\varphi^1_\lambda(x) \underset{\infty}{\sim} \alpha_2 \log x + \beta_2 \ ,$$

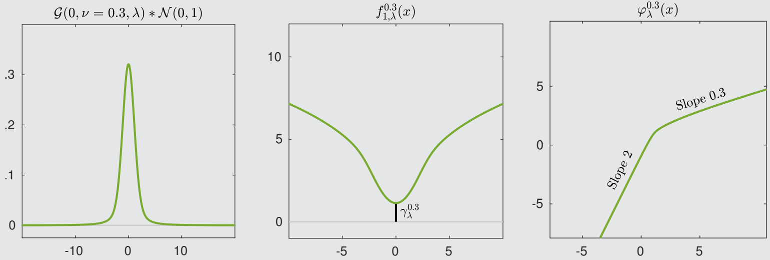
$$\text{where} \quad \alpha_2 = 1 \quad \text{and} \quad \beta_2 = \frac{1}{2} \log 2 - \log \lambda \ .$$

Part 3/3: GGMM-EPLL

Discrepancy functions

$$f^{\nu}_{\sigma,\lambda}(x) = \log \int_{\mathbb{R}} \frac{1}{\sqrt{2\pi}\sigma} \frac{\kappa_\nu}{2\lambda_\nu} \exp\left(-\frac{(t-x)^2}{2\sigma^2} - \frac{|t|^\nu}{\lambda_\nu^\nu} \right) \mathrm{d}t$$

Case $\frac{2}{3} \leqslant \nu < 2$



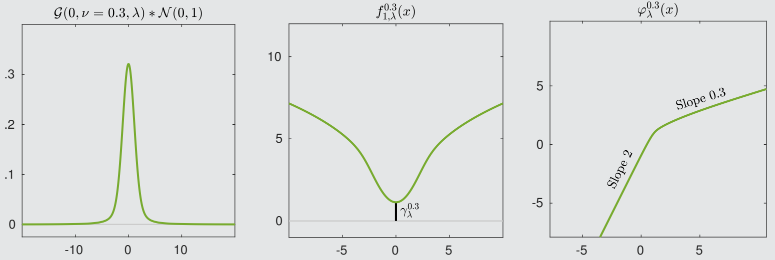
$$\varphi^{\nu}_{\lambda}(x) \underset{0}{\sim} \alpha_1 \log x + \beta_1 \ ,$$

where $\alpha_1 = 2$ and $\beta_1 = -\log 2 + \log \left(1 - \frac{\int_{-\infty}^{\infty} t^2 e^{-\frac{t^2}{2}} \exp\left[-\left(\frac{|t|}{\lambda_\nu} \right)^\nu \right] \mathrm{d}t}{\int_{-\infty}^{\infty} e^{-\frac{t^2}{2}} \exp\left[-\left(\frac{|t|}{\lambda_\nu} \right)^\nu \right] \mathrm{d}t} \right)$.

Discrepancy functions

$$f^{\nu}_{\sigma,\lambda}(x) = \log \int_{\mathbb{R}} \frac{1}{\sqrt{2\pi}\sigma} \frac{\kappa_\nu}{2\lambda_\nu} \exp\left(-\frac{(t-x)^2}{2\sigma^2} - \frac{|t|^\nu}{\lambda_\nu^\nu} \right) \mathrm{d}t$$

Case $\frac{2}{3} \leqslant \nu < 2$



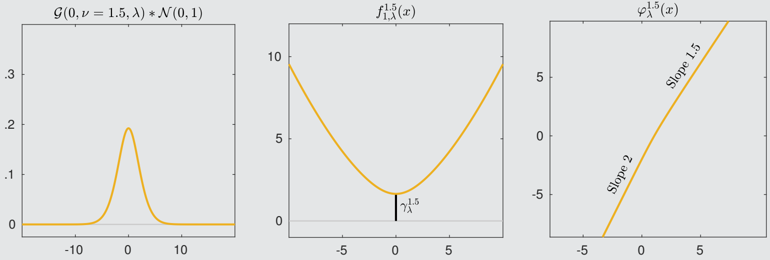
$$\varphi^{\nu}_{\lambda}(x) \underset{\infty}{\sim} \alpha_2 \log x + \beta_2 \ ,$$

where $\alpha_2 = \nu$ and $\beta_2 = -\nu \log \lambda - \frac{\nu}{2} \log \frac{\Gamma(1/\nu)}{\Gamma(3/\nu)}$.

Discrepancy functions

$$f^{\nu}_{\sigma,\lambda}(x) = \log \int_{\mathbb{R}} \frac{1}{\sqrt{2\pi}\sigma} \frac{\kappa_\nu}{2\lambda_\nu} \exp\left(-\frac{(t-x)^2}{2\sigma^2} - \frac{|t|^\nu}{\lambda_\nu^\nu} \right) \mathrm{d}t$$

Case $\frac{2}{3} \leqslant \nu < 2$



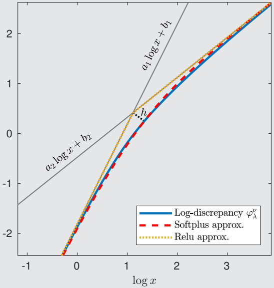
$$\varphi^{\nu}_{\lambda}(x) \underset{\infty}{\sim} \alpha_2 \log x + \beta_2 \ ,$$

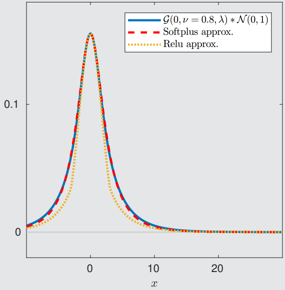
where $\alpha_2 = \nu$ and $\beta_2 = -\nu \log \lambda - \frac{\nu}{2} \log \frac{\Gamma(1/\nu)}{\Gamma(3/\nu)}$.

Discrepancy functions

$$f^{\nu}_{\sigma,\lambda}(x) = \log \int_{\mathbb{R}} \frac{1}{\sqrt{2\pi}\sigma} \frac{\kappa_\nu}{2\lambda_\nu} \exp\left(-\frac{(t-x)^2}{2\sigma^2} - \frac{|t|^\nu}{\lambda_\nu^\nu} \right) \mathrm{d}t$$

Approximation



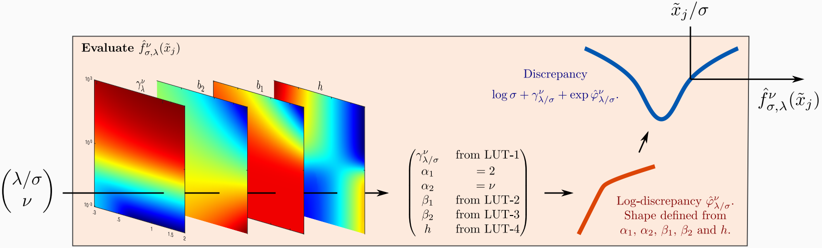


$$\hat{\varphi}^{\nu}_{\lambda}(x) = \alpha_1 \log|x| + \beta_1 - \mathtt{rec}(\alpha_1 \log|x| + \beta_1 - \alpha_2 \log|x| - \beta_2)$$

$$\mathtt{relu}(x) = \max(0, x) \quad \text{and} \quad \mathtt{softplus}(x) = h \log\left[1 + \exp\left(\frac{x}{h}\right)\right], \quad h > 0.$$

Discrepancy functions

$$f^{\nu}_{\sigma,\lambda}(x) = \log \int_{\mathbb{R}} \frac{1}{\sqrt{2\pi}\sigma} \frac{\kappa_\nu}{2\lambda_\nu} \exp\left(- \frac{(t-x)^2}{2\sigma^2} - \frac{|t|^\nu}{\lambda_\nu^\nu} \right) \mathrm{d}t$$



- Given $(\lambda/\sigma, \nu)$, get $(\gamma^{\nu}_{\lambda}, \beta_1, \beta_2, h)$ from lookup tables (LUTs).
- Compute the log-discrepancy based on asymptopts and softplus.
- Deduce the discrepancy.

Performance in denoising

σ	Algo.	BSDS	barbara	camera man	hill	house	lena	mandrill	Avg.
		PSNR							
5	GMM	37.25	37.60	38.07	35.93	38.81	38.49	35.22	37.26
	LMM	37.31	**37.83**	38.11	35.89	38.93	38.49	35.18	37.32
	HLMM	36.85	37.42	37.66	35.39	38.37	38.08	34.77	36.86
	GGMM	**37.33**	37.73	**38.12**	**35.95**	**38.94**	**38.52**	**35.23**	**37.33**
20	GMM	29.36	29.76	30.16	28.46	32.77	32.40	26.60	29.42
	LMM	29.30	**30.18**	30.04	28.36	**33.22**	**32.72**	26.43	29.37
	HLMM	28.48	29.28	29.04	27.72	32.50	32.10	25.44	28.56
	GGMM	**29.43**	30.02	**30.24**	**28.48**	33.03	32.59	**26.64**	**29.50**
60	GMM	24.57	23.95	25.10	24.21	27.53	27.28	**21.57**	24.61
	LMM	24.55	23.94	24.96	24.23	**27.91**	**27.58**	21.35	24.59
	HLMM	23.95	23.16	23.72	23.84	27.10	26.94	20.67	23.97
	GGMM	**24.64**	**24.03**	**25.17**	**24.25**	27.80	27.52	21.50	**24.67**

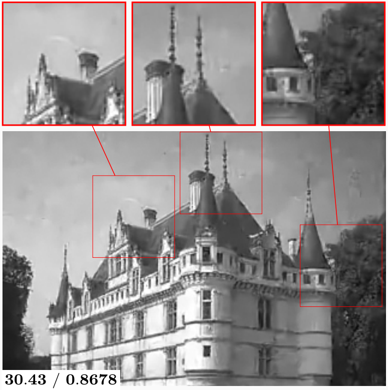
GGMM offers best performance in average compared to GMM/LMM/HLMM.

Performance in denoising

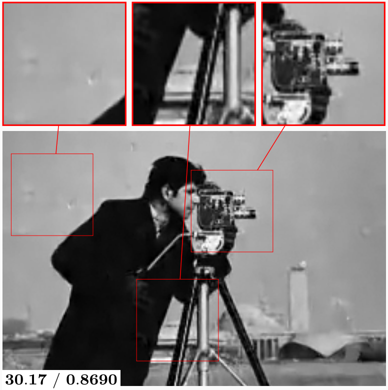
σ	Algo.	BSDS	barbara	camera man	hill	house	lena	mandrill	Avg.
					PSNR				
	BM3D	37.33	38.30	38.28	36.04	39.82	38.70	35.26	**37.36**
5	GGMM	37.33	37.73	38.12	35.95	38.94	38.52	35.23	37.33
	BM3D	33.06	34.95	34.10	31.88	36.69	35.90	30.58	**33.15**
10	GGMM	33.10	33.87	34.01	31.81	35.72	35.59	30.58	**33.15**
	BM3D	29.38	31.73	30.42	28.56	33.81	33.02	26.60	**29.50**
20	GGMM	29.43	30.02	30.24	28.48	33.03	32.59	26.64	**29.50**
	BM3D	26.28	27.97	27.16	25.89	30.69	29.81	23.07	**26.38**
40	GGMM	26.26	26.17	27.03	25.70	29.89	29.42	23.21	26.32
	BM3D	24.81	26.31	25.24	24.52	28.74	28.19	21.71	**24.90**
60	GGMM	24.64	24.03	25.17	24.25	27.80	27.52	21.50	24.67

GGMM-EPLL offers slightly worse performance than BM3D.

Performance in denoising

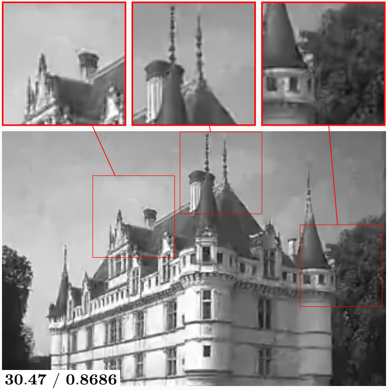


GMM

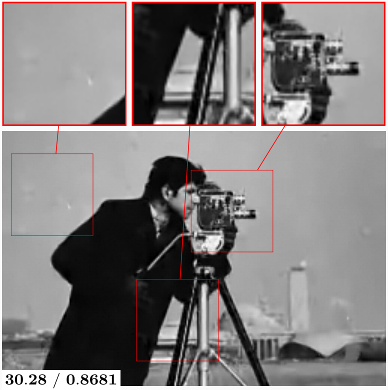


GMM

Performance in denoising

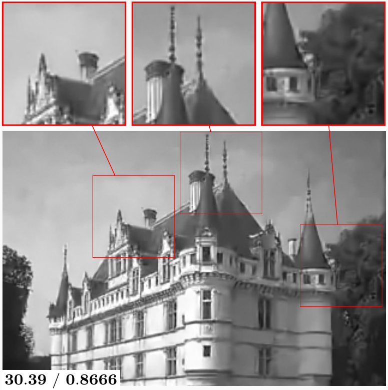


GGMM

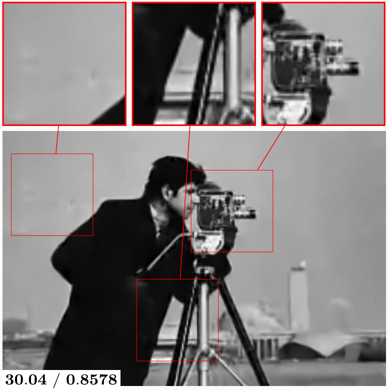


GGMM

Performance in denoising



LMM



LMM

Conclusion

Take home messages

- Image restoration with mixture model patch priors can be fast:

  faster than BM3D and faster than modern CNN approaches (on CPU).

- GGMM priors provide small improvements on GMM priors.

Difficulties

- Non-convex optimizations.

  How good are local minimizers? are comparisons GMMs/GGMMs fair?

- Is Half-Quadratic-Splitting the right solver?

  ADMM? Proximal algorithms?

- 5 iterations (early stopping) performs better than iterating more.

  Not clear what's going on. . .

What about Deep CNNs?

Advantages of patch priors based restoration compared to deep CNNs

- Patch priors are learned only once on clean data.
- Can be applied likewise for any types of degradations.
- Allows us injecting explicit knowledge on degradation models.

Patch priors + Deep CNNs

- CNNs are patch based approaches (patch=receptive fields),
- Plug-and-play ADMM with CNN denoiser (Chan, 2018),
- Deep image priors (Ulyanov, 2018),
- My own work in progress...

Thanks for your attention

References

• Parameswaran, S., Deledalle, C. A., Denis, L., & Nguyen, T. Q. (2019). Accelerating GMM-Based Patch Priors for Image Restoration: Three Ingredients for a $100\times$ Speed-Up. IEEE Transactions on Image Processing, 28(2), 687-698.

• Deledalle, C. A., Parameswaran, S., & Nguyen, T. Q. (2018). Image denoising with generalized Gaussian mixture model patch priors. SIAM Journal on Imaging Sciences, 11(4), 2568-2609.

---

cdeledal@math.u-bordeaux.fr

http://www.math.u-bordeaux.fr/~cdeledal/

- **Expectation step (E-Step)**
  - For all $k = 1, \dots, K$ and samples $i = 1, \dots, n$, compute:

$$\xi_{k,i} \leftarrow \frac{w_k \mathcal{G}(\boldsymbol{z}_i; 0_P, \boldsymbol{\Sigma}_k, \boldsymbol{\nu}_k)}{\sum_{l=1}^{K} w_l \mathcal{G}(\boldsymbol{z}_i; 0_P, \boldsymbol{\Sigma}_l, \boldsymbol{\nu}_l)} \ .$$

- **Moment step (M-Step)**
  - For all components $k = 1, \dots, K$, update:

$$w_k \leftarrow \frac{\sum_{i=1}^{n} \xi_{k,i}}{\sum_{l=1}^{K} \sum_{i=1}^{n} \xi_{l,i}} \quad \text{and} \quad \boldsymbol{\Sigma}_k \leftarrow \frac{\sum_{i=1}^{n} \xi_{k,i} \boldsymbol{z}_i \boldsymbol{z}_i^t}{\sum_{i=1}^{n} \xi_{k,i}} \ .$$

  - Perform eigen decomposition of $\boldsymbol{\Sigma}_k$:

$$\boldsymbol{\Sigma}_k = \boldsymbol{U}_k \boldsymbol{\Lambda}_k \boldsymbol{U}_k^t \quad \text{where} \quad \boldsymbol{\Lambda}_k = \operatorname{diag}(\lambda_{k,1}, \lambda_{k,2}, \dots, \lambda_{k,P})^2 \ .$$

  - For all $k = 1, \dots, K$ and dimensions $j = 1, \dots, P$, compute:

$$\chi_{k,j} \leftarrow \frac{\sum_{i=1}^{n} \xi_{k,i} |(\boldsymbol{U}_k^t \boldsymbol{z}_i)_j|}{\sum_{i=1}^{n} \xi_{k,i}} \quad \text{and} \quad (\boldsymbol{\nu}_k)_j \leftarrow \Pi_{[.3,2]} \left[F^{-1} \left(\frac{\chi_{k,j}^2}{\lambda_{k,j}^2} \right) \right] \ .$$

where $\Pi_{[a,b]}[x] = \min(\max(x, a), b)$ and $F(x) = \frac{\Gamma(2/x)^2}{\Gamma(3/x)\Gamma(1/x)}$

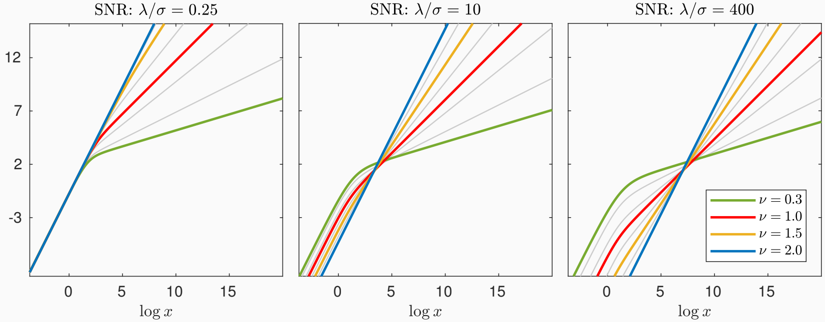


Figure 1 – Illustrations of the log-discrepancy function for various $0.3 \leqslant \nu \leqslant 2$ and SNR λ/σ.

Appendix

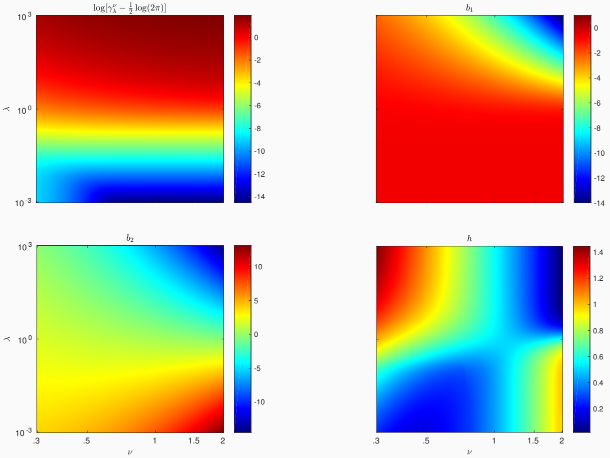


Figure 2 – Lookup tables used to store the values of the parameters γ_λ^ν, β_1, β_2 and h for various $.3 \leqslant \nu \leqslant 2$ and $10^{-3} \leqslant \lambda \leqslant 10^3$. A regular grid of 100 values has been used for ν and a logarithmic grid of 100 values has been used for λ. This leads to a total of $10,000$ combinations for each of the four lookup tables.