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.
Model

\[ y = Ax + w \]

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

Deconvolution subject to noise

\[ y = \text{Blur } A \ast x + w \]

Goal: Retrieve the sharp and clean image \( x \) from \( y \)
### Linear least square estimator

\[
\hat{x} \in \arg\min_x \frac{1}{2\sigma^2} \|Ax - y\|_2^2
\]

One solution is the Moore-Penrose pseudo inverse:

\[
\hat{x} = A^+ y = \lim_{\varepsilon \to 0} (A^t A + \varepsilon \text{Id}_N)^{-1} A^t y
\]

### Example (Deconvolution)

\[
A = \mathcal{F}^{-1} \Phi \mathcal{F} : \text{circulant matrix}
\]

\[
\mathcal{F} : \text{Fourier transform}
\]

\[
\Phi = \text{diag}(\phi_1, \ldots, \phi_N) : \text{blur Fourier coefficients}
\]

Linear least square solution

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

\[
\text{and } \quad c = \mathcal{F} y
\]
Linear least square estimator

\[ \hat{x} \in \arg \min_x \frac{1}{2\sigma^2} \|Ax - y\|_2^2 \]

One solution is the Moore-Penrose pseudo inverse:

\[ \hat{x} = A^+y = \lim_{\varepsilon \to 0} (A^tA + \varepsilon \text{Id}_N)^{-1} A^ty \]

Example (Deconvolution)

(a) Observation \( y \)  
(b) \( c = \mathcal{F}y \)  
(c) \( \hat{c} \)  
(d) \( \hat{x} = \mathcal{F}^{-1}\hat{c} \)
### Motivations – Variational models

**Variational model: Regularized linear least-square**

\[
\hat{x} \in \arg\min_x \frac{1}{2\sigma^2} \|Ax - y\|_2^2 + R(x)
\]

#### Example (Maximum A Posteriori (MAP))

\[
\frac{1}{2\sigma^2} \|Ax - y\|_2^2 = -\log p(y|x) \quad \text{(likelihood for Gaussian noises)}
\]

\[
R(x) = -\log p(x) \quad \text{(a priori)}
\]
Motivations – Variational models

Variational model: Regularized linear least-square

\[ \hat{x} \in \arg\min_x \frac{1}{2\sigma^2} \|Ax - y\|^2_2 + R(x) \]

Example (Maximum A Posteriori (MAP))

\[ \frac{1}{2\sigma^2} \|Ax - y\|^2_2 = -\log p(y|x) \quad \text{(likelihood for Gaussian noises)} \]

\[ R(x) = -\log p(x) \quad \text{(a priori)} \]

What about a Gaussian prior?

Gaussian prior \( x \sim \mathcal{N}(\mu, \Sigma) \)
Motivations – Variational models

### Variational model: Regularized linear least-square

\[
\hat{x} \in \arg\min_x \frac{1}{2\sigma^2} \|Ax - y\|_2^2 + R(x)
\]

### Example (Wiener deconvolution / Tikhonov regularization)

\[
R(x) = \|\Lambda^{-1/2}F \hat{x}\|_2^2 = \sum_i \left(\frac{c_i}{\lambda_i}\right)^2 \quad \text{with} \quad c = Fx
\]

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

Solution is linear: \[
\hat{x} = (A^tA + \sigma^2\Gamma^t\Gamma)^{-1}A^ty
\]
**Variational model: Regularized linear least-square**

\[
\hat{x} \in \arg\min_x \frac{1}{2\sigma^2} \|Ax - y\|_2^2 + R(x)
\]

**Example (Wavelet shrinkage/thresholding)**

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

\[\mathcal{W}: \text{Wavelet transform or Frame } (\mathcal{W}^+\mathcal{W} = \text{Id}_N)\]

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

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

(a) \( y \)  
(b) \( \hat{x} \)
### Variational model: Regularized linear least-square

\[ \hat{x} \in \arg\min_x \frac{1}{2\sigma^2} \|Ax - y\|_2^2 + R(x) \]

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

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

\( \nabla \) : 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 \( \lambda \)  (c) Small \( \lambda \)  (d) Medium \( \lambda \)  (e) Huge \( \lambda \)
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.

\[ \hat{x} \in \arg\min_x \frac{P}{2\sigma^2} \|Ax - y\|_2^2 + \sum_{i=1}^{N} R(P_i x) \]

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

\[ P_i : \mathbb{R}^N \rightarrow \mathbb{R}^P \] extracts a patch with \( P \) pixels centered at location \( i \).

Linear operator. Typically, \( P = 8 \times 8 \).
**Motivations – Patch priors**

**Regularized linear least-square with patch priors**

\[ \hat{x} \in \arg\min_{x} \frac{P}{2\sigma^2} \|Ax - y\|_2^2 + \sum_{i=1}^{N} R(P_i x) \]

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

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

Since 1699, when French explorers landed at the great bend of the Mississippi River and celebrated the first Mardi Gras in North America, New Orleans has brewed a fascinating melange of cultures. It was French, then Spanish, then French again, then sold to the United States. Through all these years, and even into the 1900s, others arrived from everywhere: Acadians (Cajuns), Africans, indigenes...
Regularized linear least-square with patch priors

\[ \hat{x} \in \arg\min_x \frac{P}{2\sigma^2} \|Ax - y\|_2^2 + \sum_{i=1}^{N} R(P_i x) \]

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

- \( R(z) = \frac{1}{\lambda} \|\Gamma z\|_0 = \#\{c_i \neq 0\} \) with \( c = \Gamma z \)
- \( \|\cdot\|_0: \ell_0 \) 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).
Motivations – Patch priors

Regularized linear least-square with patch priors

\[ \hat{x} \in \arg\min_{x} \frac{P}{2\sigma^2} \|Ax - y\|_2^2 + \sum_{i=1}^{N} R(P_i x) \]

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

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

and \[ p(z) = \sum_{k=1}^{K} w_k \frac{1}{(2\pi)^{P/2} |\Sigma_k|^{1/2}} \exp \left( -\frac{1}{2} z^t \Sigma_k^{-1} z \right) , \]

- \( K \): number of Gaussians (clusters)
- \( w_k \): weights \( \sum_k w_k = 1 \) (frequency of each clusters)
- \( \Sigma_k \): \( P \times P \) 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**

\[
\hat{x} \in \arg\min_x \frac{P}{2\sigma^2} \|Ax - y\|^2_2 + \sum_{i=1}^N R(P_i x)
\]

**Example (EPLL, Zoran & Weiss, 2011)**

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

and

\[
p(z) = \sum_{k=1}^K w_k \frac{1}{(2\pi)^{P/2} \left| \Sigma_k \right|^{1/2}} \exp \left( -\frac{1}{2} z^t \Sigma_k^{-1} z \right),
\]

\((w_k, \Sigma_k)\) learned by EM on 2 million patches.

Patch size: \(P = 8 \times 8\)

#Gaussians: \(K = 200\)

100 randomly generated patches from the learned model
Regularized linear least-square with patch priors

\[
\hat{x} \in \arg\min_{x} \frac{P}{2\sigma^2} \|Ax - y\|_2^2 + \sum_{i=1}^{N} R(P_i x)
\]

Example (EPLL, Zoran & Weiss, 2011)

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

(a) Reference \(x\)  
(b) Noisy image \(y\)  
(c) EPLL result \(\hat{x}\)
Regularized linear least-square with patch priors

\[ \hat{x} \in \arg\min_x \frac{P}{2\sigma^2} \|Ax - y\|_2^2 + \sum_{i=1}^{N} R(\mathcal{P}_i x) \]

Example (EPLL, Zoran & Weiss, 2011)

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

(a) Reference \( x \) / Blur kernel
(b) Blurry image \( y \)
(c) EPLL result \( \hat{x} \)
Regularized linear least-square with patch priors

\[ \hat{x} \in \arg \min_x \frac{P}{2\sigma^2} \|Ax - y\|_2^2 + \sum_{i=1}^{N} R(P_i \hat{x}) \]

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
Regularized linear least-square with patch priors

\[ \hat{x} \in \arg\min_x \frac{P}{2\sigma^2} \|Ax - y\|_2^2 + \sum_{i=1}^{N} R(P_i x) \]

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

\[ \hat{x} \in \arg\min_x \frac{P}{2\sigma^2} \|Ax - y\|_2^2 - \sum_{i=1}^{N} \log p(P_i x - \overline{P_i x}) \]

Half-quadratic splitting

- Introduce \( N \) auxiliary vectors \( z_i \in \mathbb{R}^P \) and solve instead:

\[ \lim_{\beta \to \infty} \arg\min_{x, z_1, \ldots, z_N} \frac{P}{2\sigma^2} \|Ax - y\|_2^2 + \frac{\beta}{2} \sum_{i=1}^{N} \|P_i x - z_i\|_2^2 - \sum_{i=1}^{N} \log p(z_i - \overline{z}_i). \]

- Use an alternating optimization scheme on \( z_i \) and \( x \). Repeat:

\[ z_i \leftarrow \arg\min_{z_i} \frac{\beta}{2} \|P_i \hat{x} - z_i\|_2^2 - \log p(z_i - \overline{z}_i), \text{ for all } 1 \leq i \leq N \]

\[ \hat{x} \leftarrow \arg\min_x \frac{P}{2\sigma^2} \|Ax - y\|_2^2 + \frac{\beta}{2} \sum_{i=1}^{N} \|P_i x - \hat{z}_i\|_2^2 \]

\[ \beta \leftarrow \text{increase}(\beta) \]
Optimization on $\hat{x}$:

$$\hat{x} \in \arg\min_x \frac{P}{2\sigma^2} \|Ax - y\|_2^2 + \frac{\beta}{2} \sum_{i=1}^N \|P_i x - \hat{z}_i\|_2^2$$

$$= \left( A^t A + \frac{\beta \sigma^2}{P} \sum_{i=1}^N P_i^t P_i \right)^{-1} \left( A^t y + \frac{\beta \sigma^2}{P} \sum_{i=1}^N P_i^t \hat{z}_i \right)$$

$$= \left( A^t A + \beta \sigma^2 \text{Id}_N \right)^{-1} \left( A^t y + \beta \sigma^2 \tilde{x} \right)$$

with

$$\tilde{x} = \frac{1}{P} \sum_{i=1}^N P_i^t \hat{z}_i$$

In general, $O(N)$ or $O(N \log N)$

Otherwise, conjugate gradient

Patch reprojection
Optimization on $x$:

$$
\hat{x} \in \arg\min_x \frac{P}{2\sigma^2} \|Ax - y\|^2_2 + \frac{\beta}{2} \sum_{i=1}^{N} \|P_i x - \hat{z}_i\|^2_2
$$

$$
= \left( A^t A + \frac{\beta \sigma^2}{P} \sum_{i=1}^{N} P_i^t P_i \right)^{-1} \left( A^t y + \frac{\beta \sigma^2}{P} \sum_{i=1}^{N} P_i^t \hat{z}_i \right)
$$

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

In general, $O(N)$ or $O(N \log N)$

Otherwise, conjugate gradient

**Example (Deconvolution)**

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

$$
\hat{x} = \mathcal{F}^{-1} \hat{c} \quad \text{where} \quad \hat{c}_i = \frac{\phi_i^* c_i + \beta \sigma^2 \tilde{c}_i}{|\phi_i|^2 + \beta \sigma^2} \quad \text{with} \quad \left\{ \begin{array}{l}
c = \mathcal{F} y \\
\tilde{c} = \mathcal{F} \tilde{x}
\end{array} \right.
$$
Part 1/3: EPLL Algorithm (Zoran & Weiss, 2011)

Optimization on $z$:

$$\hat{z} \in \arg\min_{z} \frac{\beta}{2} \|\tilde{z} - z\|_2^2 - \log p(z - \tilde{z})$$

$$= \bar{z} + \arg\min_{z} \frac{\beta}{2} \|\tilde{z} - \bar{z} - z\|_2^2 - \log p(z)$$
Optimization on $z$:

$$\hat{z} \in \arg\min_z \frac{\beta}{2} \|\tilde{z} - z\|_2^2 - \log p(z - \tilde{z})$$

$$= \bar{z} + \arg\min_z \frac{\beta}{2} \|\tilde{z} - \bar{z} - z\|_2^2 - \log p(z)$$

For the sake of simplicity consider

$$\hat{z} \in \arg\min_z \frac{\beta}{2} \|\tilde{z} - z\|_2^2 - \log p(z)$$

$$= \arg\min_z \frac{\beta}{2} \|\tilde{z} - z\|_2^2 - \log \sum_{k=1}^{K} w_k \exp \left(-\frac{1}{2} z^t \Sigma_k^{-1} z\right) \quad \text{(Non convex)}$$

$$\approx \arg\min_z \frac{\beta}{2} \|\tilde{z} - z\|_2^2 + \frac{1}{2} z^t \Sigma_{k^*}^{-1} z \quad \text{(Keep only 1 ⇒ Convex)}$$

$$= \left(\Sigma_{k^*} + \frac{1}{\beta} \text{Id}_P\right)^{-1} \Sigma_{k^*} \tilde{z} \quad \text{(Explicit solution)}$$

How to choose the optimal $k^*$?
Zoran & Weiss (2011) interpret
\[
\arg\min_z \frac{\beta}{2} \|\tilde{z} - z\|_2^2 + \frac{1}{2} z^t \Sigma_k^{-1} z
\]
as a MAP denoising problem where
\[
\begin{align*}
\tilde{z} \mid z & \sim \mathcal{N}(z, \frac{1}{\beta} \text{Id}_P) \\
z \mid k & \sim \mathcal{N}(0_P, \Sigma_k)
\end{align*}
\]
\[
= \mathcal{N}(0_P, \Sigma_k) * \mathcal{N}(0_P, \frac{1}{\beta} \text{Id}_P)
\]
(\text{convolution})
Zoran & Weiss (2011) interpret

$$\arg\min_z \frac{\beta}{2} \|\tilde{z} - z\|_2^2 + \frac{1}{2} z^t \Sigma^{-1}_k z$$

as a MAP denoising problem where

$$\tilde{z} \mid z \sim \mathcal{N}(z, \frac{1}{\beta} \text{Id}_P)$$

$$z \mid k \sim \mathcal{N}(0_P, \Sigma_k)$$

\[\begin{align*}
\text{Marginalization} & \quad \Rightarrow \quad \tilde{z} \mid k \sim \mathcal{N}(0_P, \Sigma_k + \frac{1}{\beta} \text{Id}_P) \\
& = \mathcal{N}(0_P, \Sigma_k) \ast \mathcal{N}(0_P, \frac{1}{\beta} \text{Id}_P) \quad \text{(convolution)}
\end{align*}\]

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

$$k^* \in \arg\max_{1 \leq k \leq K} p(k \mid \tilde{z}) = \arg\max_{1 \leq k \leq K} \mathbb{P}(k)p(\tilde{z} \mid k) = \arg\max_{1 \leq k \leq K} w_k p(\tilde{z} \mid k)$$

$$= \arg\min_{1 \leq k \leq K} -2 \log w_k + \log |\Sigma_k + \frac{1}{\beta} \text{Id}_P| + \tilde{z}^t (\Sigma_k + \frac{1}{\beta} \text{Id}_P)^{-1} \tilde{z}$$

Discrepancy of patch $\tilde{z}$ against component $k$
EPLL Algorithm

In practice:

- 5 iterations are used
- \( \beta = \frac{1}{\sigma^2} \{1, 4, 8, 16, 32\} \)
Algorithm: The five steps of an EPLL iteration

for all $1 \leq i \leq N$

1. $\tilde{z}_i \leftarrow \mathcal{P}_i \hat{x}$ (Patch extraction)

2. $k_i^* \leftarrow \arg\min_{1 \leq k_i \leq K} -2 \log w_{k_i} + \log \left| \Sigma_{k_i} + \frac{1}{\beta} \text{Id}_P \right| + \tilde{z}_i^t \left( \Sigma_{k_i} + \frac{1}{\beta} \text{Id}_P \right)^{-1} \tilde{z}_i$ (Gaussian selection)

3. $\hat{z}_i \leftarrow \left( \Sigma_{k_i^*} + \frac{1}{\beta} \text{Id}_P \right)^{-1} \Sigma_{k_i^*} \tilde{z}_i$ (Patch estimation)

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

5. $\hat{x} \leftarrow (A^t A + \beta \sigma^2 \text{Id}_N)^{-1} (A^t y + \beta \sigma^2 \tilde{x})$ (Image estimation)
**Algorithm:** The five steps of an EPLL iteration

for all $1 \leq i \leq N$

- $O(NP)$: $\tilde{z}_i \leftarrow P_i \hat{x}$  
  (Patch extraction)

- $O(NKP^2)$: $k_i^* \leftarrow \arg\min_{1 \leq k_i \leq K} -2 \log w_{k_i} + \log \left| \Sigma_{k_i} + \frac{1}{\beta} \text{Id}_P \right| + \tilde{z}_i^t \left( \Sigma_{k_i} + \frac{1}{\beta} \text{Id}_P \right)^{-1} \tilde{z}_i$  
  (Gaussian selection)

- $O(NP^2)$: $\hat{z}_i \leftarrow \left( \Sigma_{k_i^*} + \frac{1}{\beta} \text{Id}_P \right)^{-1} \Sigma_{k_i^*} \tilde{z}_i$  
  (Patch estimation)

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

- $O(N \log N)$: $\hat{x} \leftarrow (A^tA + \beta \sigma^2 \text{Id}_N)^{-1} (A^t y + \beta \sigma^2 \tilde{x})$  
  (Image estimation)

**Global complexity:** $O(NKP^2)$
**Algorithm 1** The five steps of an EPLL iteration

<table>
<thead>
<tr>
<th>Step</th>
<th>Time</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. $\tilde{z}_i \leftarrow \mathcal{P}_i x$ (Patch extraction)</td>
<td>0.46s</td>
<td>1 %</td>
</tr>
<tr>
<td>2. $k_i^* \leftarrow \arg\min_{1 \leq k_i \leq K} \log w_{k_i}^{-2} + \log \left</td>
<td>\Sigma_{k_i} + \frac{1}{\beta} \text{Id}_P \right</td>
<td>$ (Gaussian selection)</td>
</tr>
<tr>
<td>3. $\tilde{z}<em>i \leftarrow \left( \Sigma</em>{k_i^<em>} + \frac{1}{\beta} \text{Id}<em>P \right)^{-1} \Sigma</em>{k_i^</em>} \tilde{z}_i$ (Patch estimation)</td>
<td>0.95s</td>
<td>2 %</td>
</tr>
<tr>
<td>4. $\tilde{x} \leftarrow \left( \sum_{i \in I} \mathcal{P}_i^t \mathcal{P}<em>i \right)^{-1} \sum</em>{i \in I} \mathcal{P}_i^t \tilde{z}_i$ (Patch reproject)</td>
<td>0.23s</td>
<td>1 %</td>
</tr>
<tr>
<td>5. $\hat{x} \leftarrow \left( A^t A + \beta \sigma^2 \text{Id}_N \right)^{-1} (A^t y + \beta \sigma^2 \tilde{x})$</td>
<td>Others</td>
<td>0.52s</td>
</tr>
<tr>
<td>Total</td>
<td>45.69s</td>
<td></td>
</tr>
</tbody>
</table>

Gaussian selection represents 95% of computation time!
### Gaussian selection represents 95% of computation time!

<table>
<thead>
<tr>
<th>Algorithm 1 The five steps of an EPLL iteration</th>
<th>Time</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tilde{z}_i \leftarrow P_i x$</td>
<td>0.46s</td>
<td>1 %</td>
</tr>
<tr>
<td>$k^*<em>i \leftarrow \arg\min</em>{1 \leq k_i \leq K} \log w_{k_i}^{-2} + \log \left</td>
<td>\Sigma_{k_i} + \frac{1}{\beta} Id_P \right</td>
<td>+ \tilde{z}<em>i (\Sigma</em>{k_i} + \frac{1}{\beta} Id_P)^{-1} \tilde{z}_i$</td>
</tr>
<tr>
<td>$\tilde{z}<em>i \leftarrow (\Sigma</em>{k^<em><em>i} + \frac{1}{\beta} Id_P)^{-1} \Sigma</em>{k^</em>_i} \tilde{z}_i$</td>
<td>0.95s</td>
<td>2 %</td>
</tr>
<tr>
<td>$\tilde{x} \leftarrow (\sum_{i \in I} P_i^t P_i)^{-1} \sum_{i \in I} P_i^t \tilde{z}_i$</td>
<td>0.23s</td>
<td>1 %</td>
</tr>
<tr>
<td>$\tilde{x} \leftarrow (A^t A + \beta \sigma^2 Id_N)^{-1} (A^t y + \beta \sigma^2 \tilde{x})$</td>
<td>0.52s</td>
<td>1 %</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>45.69s</td>
<td></td>
</tr>
</tbody>
</table>

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

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

Simple idea to accelerate the optimization on $z_i$:

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

$O(|\mathcal{I}|P)$ \hspace{1cm} $\tilde{z}_i \leftarrow \mathcal{P}_i \hat{x}$ \hspace{1cm} (Patch extraction)

$O(|\mathcal{I}|KP^2)$ \hspace{1cm} $k_i^* \leftarrow \operatorname{argmin}_{1 \leq k_i \leq K} -2 \log w_{k_i} + \log |\Sigma_{k_i} + \frac{1}{\beta} \mathbf{Id}_P| + \tilde{z}_i^t \left( \Sigma_{k_i} + \frac{1}{\beta} \mathbf{Id}_P \right)^{-1} \tilde{z}_i$ \hspace{1cm} (Gaussian selection)

$O(|\mathcal{I}|P^2)$ \hspace{1cm} $\hat{z}_i \leftarrow \left( \Sigma_{k_i^*} + \frac{1}{\beta} \mathbf{Id}_P \right)^{-1} \Sigma_{k_i^*} \tilde{z}_i$ \hspace{1cm} (Patch estimation)
Contribution 1: consider only a subset $\mathcal{I} \subseteq [1, \ldots N]$ of patch indices!

But, it slows down the optimization on $x$:

$$\hat{x} \in \arg\min_x \frac{P}{2\sigma^2} \|Ax - y\|_2^2 + \frac{\beta}{2} \sum_{i=1}^{N} \|P_i x - \hat{z}_i\|_2^2$$

$$\approx \arg\min_x \frac{P}{2\sigma^2} \|Ax - y\|_2^2 + \frac{\beta}{2} \sum_{i \in \mathcal{I}} \|P_i x - \hat{z}_i\|_2^2$$

$$= \left( A^t A + \frac{\beta \sigma^2}{P} \sum_{i \in \mathcal{I}} P_i^t P_i \right)^{-1} \left( A^t y + \frac{\beta \sigma^2}{P} \sum_{i \in \mathcal{I}} P_i^t \hat{z}_i \right)$$

- $(\sum_{i \in \mathcal{I}} P_i^t P_i)_{jj} = \# \text{patches covering pixel with index } j$
- The matrices $A^t A$ and $\sum_{i \in \mathcal{I}} P_i^t P_i$ 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

$$
\hat{x} \in \arg\min_x \frac{P}{2\sigma^2} \|Ax - y\|^2_2 + \frac{\beta}{2} \sum_{i=1}^{N} \|P_i x - \hat{z}_i\|^2_2
$$

$$
= \left( A^t A + \frac{\beta \sigma^2}{P} \sum_{i=1}^N P_i^t P_i \right)^{-1} \left( A^t y + \frac{\beta \sigma^2}{P} \sum_{i=1}^N P_i^t \hat{z}_i \right)
$$

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

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

$\Rightarrow$ Every other steps will be accelerated, and this step will be unchanged.
Part 2/3: Fast EPLL – Stochastic patch sub-sampling

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: $\mathcal{O}(NP^2K) \rightarrow \mathcal{O}(NP^2K/s^2)$
Part 2/3: Fast EPLL – Stochastic patch sub-sampling

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

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

Can we reduce the term in $P^2$?
Part 2/3: Fast EPLL – Flat tail approximation

Contribution 2: approximate the spectrum of covariance matrices

- Keep only $1 \leq r_k \leq P$ 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?
Contributions

- Do not set them to zero (low-rank approximation).
- Replace least eigenvalues by their average.
Part 2/3: Fast EPLL – Flat tail approximation

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?
Recall we have to compute

\[ k^* \leftarrow \arg\min_{1 \leq k \leq K} -2 \log w_k + \log \left| \Sigma_k + \frac{1}{\beta} \text{Id}_P \right| + \hat{z}^t \left( \Sigma_k + \frac{1}{\beta} \text{Id}_P \right)^{-1} \hat{z} \]

\[ \hat{z} \leftarrow \left( \Sigma_{k^*} + \frac{1}{\beta} \text{Id}_P \right)^{-1} \Sigma_{k^*} \hat{z} \]

Decompose \( \Sigma_k = U_k \Lambda_k U_k^t \) with \( \Lambda_k = \text{diag}(\lambda_{k,1}^2, \ldots, \lambda_{k,P}^2) \), and \( U_k \) unitary.

\[ \tilde{c}_k \leftarrow U_k^t \hat{z}, \quad \text{for all } 1 \leq k \leq K \quad O(P^2 K) \]

\[ k^* \leftarrow \arg\min_{1 \leq k \leq K} -2 \log w_k + \sum_{j=1}^{P} \left( \log(\lambda_{k,j}^2 + \frac{1}{\beta}) + \frac{\tilde{c}_{k,j}^2}{\lambda_{k,j}^2 + \frac{1}{\beta}} \right) \quad O(PK) \]

\[ \hat{c}_j \leftarrow \frac{\lambda_{k^*,j}^2}{\lambda_{k^*,j}^2 + \frac{1}{\beta}} \tilde{c}_{k^*,j}, \quad \text{for all } 1 \leq j \leq P \quad O(P) \]

\[ \hat{z} \leftarrow U_{k^*} \hat{c} \quad O(P^2) \]
Part 2/3: Fast EPLL – Flat tail approximation

Contribution 2: approximate the spectrum of covariance matrices

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

\[
\tilde{c}^k \leftarrow \bar{U}_k^t \tilde{z}, \quad \text{for all } 1 \leq k \leq K \quad \mathcal{O}(P\bar{r}K)
\]

\[
k^* \leftarrow \arg\min_{1 \leq k \leq K} -2 \log w_k + (P - r) \log(\alpha_k^2 + \frac{1}{\beta}) + \frac{\|\tilde{z}\|_2^2}{\alpha_k^2 + \frac{1}{\beta}} + \sum_{j=1}^{r_k} \left( \log(\lambda_{k,j}^2 + \frac{1}{\beta}) + \frac{\tilde{c}_{k,j}^2}{\lambda_{k,j}^2 + \frac{1}{\beta}} - \frac{\tilde{c}_{k,j}^2}{\alpha_k^2 + \frac{1}{\beta}} \right) \quad \mathcal{O}(\bar{r}K)
\]

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

\[
\hat{z} \leftarrow \bar{U}_{k^*} \hat{c} + \frac{\alpha_{k^*}^2}{\alpha_{k^*}^2 + \frac{1}{\beta}} \tilde{z} \quad \mathcal{O}(P\bar{r}_k)
\]

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

38
Contribution 2: approximate the spectrum of covariance matrices

(a) Noisy

(b) $\rho = 0.5$

(c) $\rho = 0.8$

(d) $\rho = 0.95$

(e) $\rho = 1$

Results for varying $\rho$

Limit of -0.2dB
Part 2/3: Fast EPLL – Binary search tree

Contribution 3: binary balanced search tree

- Avoid comparing each patch $z_i$ against each of the $K$ components

$$k^*_i \leftarrow \arg\min_{1 \leq k \leq K} - 2 \log w_k + \log \left| \Sigma_k + \frac{1}{\beta} \mathbf{I}_K \right| + \tilde{z}_i^t \left( \Sigma_k + \frac{1}{\beta} \mathbf{I}_K \right)^{-1} \tilde{z}_i$$

- Use a balanced (almost) binary search tree

\[ O(K) \]

\[ O(\log_2 K) \]

- 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.
Part 2/3: Fast EPLL

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

<table>
<thead>
<tr>
<th>Algorithm 1</th>
<th>The five steps of an EPLL iteration</th>
<th>Without accelerations</th>
<th>With the proposed accelerations</th>
</tr>
</thead>
<tbody>
<tr>
<td>for all $i \in \mathcal{I}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\hat{z}_i \leftarrow \mathcal{P}_i \hat{x}$</td>
<td>(Patch extraction)</td>
<td>0.46s</td>
<td>1%</td>
</tr>
<tr>
<td>$k_i^* \leftarrow \arg\min_{1 \leq k_i \leq K} \log w_{k_i}^{-2} + \log \left</td>
<td>\Sigma_{k_i} + \frac{1}{2} \mathbf{I} P \right</td>
<td>$</td>
<td>(Gaussian selection)</td>
</tr>
<tr>
<td>$\tilde{z}<em>i \leftarrow \left( \Sigma</em>{k_i} + \frac{1}{2} \mathbf{I} P \right)^{-1} \Sigma_{k_i} \hat{z}_i$</td>
<td>(Patch estimation)</td>
<td>0.95s</td>
<td>2%</td>
</tr>
<tr>
<td>$\hat{x} \leftarrow \left( \sum_{i \in \mathcal{I}} P_i^t P_i \right)^{-1} \sum_{i \in \mathcal{I}} P_i^t \tilde{z}_i$</td>
<td>(Patch reprojection)</td>
<td>0.23s</td>
<td>1%</td>
</tr>
<tr>
<td>$\hat{x} \leftarrow (A^t A + \beta \sigma^2 \mathbf{I} N)^{-1} (A^t y + \beta \sigma^2 \hat{x})$</td>
<td>Others</td>
<td>0.52s</td>
<td>1%</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td>45.69s</td>
<td></td>
</tr>
</tbody>
</table>

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

- $N$ image size
- $P = 8 \times 8$
- $K = 200$
- $\lceil \log_2 K \rceil = 7$
- $s^2 = 36$
- $\bar{r} = 19.6$ ($\rho = .95$)
Part 2/3: Fast EPLL

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

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

Noise standard deviation $\sigma = 20$. 

![Graph showing PSNR vs. time for different methods]
Part 2/3: Fast EPLL

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

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

Noise standard deviation $\sigma = 20$. 
Part 2/3: Fast EPLL

(a) Ref $x$ / kernel $\frac{24.9}{.624}$
(b) Blurry image $y$ $\frac{31.4}{.891} (0.17^* s)$
(c) CSF result $\hat{x}$ $\frac{32.2}{.910} (1.17 s)$
(d) RoG result $\hat{x}$ $\frac{32.7}{.924} (0.46 s)$
(e) FEPLL result $\hat{x}$

<table>
<thead>
<tr>
<th>Algo.</th>
<th>Berkeley</th>
<th>Classic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSNR/SSIM</td>
<td>Time (s)</td>
</tr>
<tr>
<td>iPiano</td>
<td>29.5 / .824</td>
<td>29.53</td>
</tr>
<tr>
<td>CSF$_{pw}$</td>
<td>30.2 / .875</td>
<td>0.50 (0.14*)</td>
</tr>
<tr>
<td>RoG</td>
<td>31.3 / .897</td>
<td>1.19</td>
</tr>
<tr>
<td>FEPLL</td>
<td>33.1 / .928</td>
<td>0.40</td>
</tr>
<tr>
<td>FEPLL’</td>
<td>33.2 / .930</td>
<td>1.01</td>
</tr>
</tbody>
</table>

Using the blur kernel of iPiano and noise standard deviation $\sigma = 0.5$. 
Part 2/3: Fast EPLL

- Works likewise for several inverse problems.
- Less than 0.4s in all cases (for images of size $481 \times 321$).
- 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?

- 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.

\[ j = 1, \quad \lambda = 0.15 \]

\[ j = 2, \quad \lambda = 0.13 \]

\[ j = 3, \quad \lambda = 0.066 \]

\[ j = 4, \quad \lambda = 0.056 \]

\[ j = 62, \quad \lambda = 0.003 \]

\[ j = 63, \quad \lambda = 0.0026 \]
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

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

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

- \( \lambda \): scale parameter (standard deviation),
- \( \nu \): shape parameter (\( \nu = 2 \): Gaussian, \( \nu = 1 \): Laplacian).
What if we look for $\nu$ that best fits?

$j = 1, \quad \lambda = 0.15, \quad \nu = 0.91$

$j = 2, \quad \lambda = 0.13, \quad \nu = 0.69$

$j = 3, \quad \lambda = 0.066, \quad \nu = 1.17$

$j = 4, \quad \lambda = 0.056, \quad \nu = 1.15$

$j = 62, \quad \lambda = 0.003, \quad \nu = 1.62$

$j = 63, \quad \lambda = 0.0026, \quad \nu = 1.50$
Part 3/3: GGMM-EPLL

What about multi-variate GGD?

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

where \( K_\nu = \prod_{j=1}^{P} \frac{\nu_j}{\Gamma(1/\nu_j)} \) and \( \Sigma_\nu^{1/2} = U \Lambda^{1/2} \left( \begin{array}{c} \sqrt{\frac{\Gamma(1/\nu_1)}{\Gamma(3/\nu_1)}} \\ \vdots \\ \sqrt{\frac{\Gamma(1/\nu_P)}{\Gamma(3/\nu_P)}} \end{array} \right). \)

- \( \ell_\rho \) prior: \( \|x\|_\rho^\rho = \sum_k |x_k|^\rho \)
- convexity: \( \rho \geq 1 \)
- sparsity: \( \rho \leq 1 \)

2-dim vector:

- \( \|x\|_0 = 0 \)
- \( \|x\|_0 = 1 \)
- \( \|x\|_0 = 2 \)

null image
sparse image
dense image
(Source: G. Peyré)
Part 3/3: GGMM-EPLL

**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 $U_k$.
- Coefficients for all directions $j$ are likely in the range $[-2\lambda_{k,j}, 2\lambda_{k,j}]$.

**GGMM**

$$p(z) = \sum_{k=1}^{K} w_k G(z; 0_P, \Sigma_k, \nu_k)$$

- Clusters have ellipsoidal ($\nu_{k,j} > 1$) or star shaped ($\nu_{k,j} \leq 1$) directions.
- Dense ($\nu_{k,j} > 1$) or sparse ($\nu_{k,j} \leq 1$) combinations of the columns of $U_k$.
- 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
Parameters \((\Sigma_k, \nu_k)\) estimated by Expectation-Maximization on a training set of 2 million clean \(8 \times 8\) patches.

Set of 100 generated random patches for each model.
Parameters \((\Sigma_k, \nu_k)\) estimated by Expectation-Maximization on a training set of 2 million clean \(8 \times 8\) patches.

- GGMM consistently fits best patches of each images of the testing set.
- Adding an extra degree of freedom (shape \(\nu\)) did not lead to overfitting.
Part 3/3: GGMM-EPLL

How to extend EPLL to GGMM patch priors?

- EPLL uses the Gaussian clusters through two equations:

\[
k^* \leftarrow \arg\min_{1 \leq k \leq K} -2 \log w_k + 2 \sum_{j=1}^{P} \left( \frac{1}{2} \log(\lambda_{k,j}^2 + \frac{1}{\beta}) + \frac{1}{2} \frac{\tilde{c}_{k,j}^2}{\lambda_{k,j}^2 + \frac{1}{\beta}} \right) = f(\tilde{c}_{k,j}; 1/\beta, \lambda_{k,j})
\]

\[
\hat{c}_j \leftarrow \frac{\lambda_{k^*,j}^2}{\lambda_{k^*,j}^2 + \frac{1}{\beta}} \tilde{c}_{k^*,j}, \quad s(\tilde{c}_{k^*,j}; 1/\beta, \lambda_{k^*,j})
\]

for all \(1 \leq j \leq P\)
Part 3/3: GGMM-EPLL

How to extend EPLL to GGMM patch priors?

• EPLL uses the Gaussian clusters through two equations:

\[
k^* \leftarrow \arg\min_{1 \leq k \leq K} -2 \log w_k + 2 \sum_{j=1}^{P} \left( \frac{1}{2} \log(\lambda_{k,j}^2 + \frac{1}{\beta}) + \frac{1}{2} \frac{\tilde{c}_{k,j}^2}{\lambda_{k,j}^2 + \frac{1}{\beta}} \right) = f(\tilde{c}_{k,j}; \frac{1}{\beta}, \lambda_{k,j})
\]

\[
\hat{c}_j \leftarrow \frac{\lambda_{k^*,j}^2}{\lambda_{k^*,j}^2 + \frac{1}{\beta}} \tilde{c}_{k^*,j}, \quad s(\tilde{c}_{k^*,j}; \frac{1}{\beta}, \lambda_{k^*,j})
\]

for all \(1 \leq j \leq P\)

• Where \(f\) and \(s\) were arising from:

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

\[
s(x; \sigma, \lambda) \in \arg\min_{t \in \mathbb{R}} \frac{(t-x)^2}{2\sigma^2} + \frac{t^2}{2\lambda^2}
\]
How to extend EPLL to GGMM patch priors?

- EPLL uses the Gaussian clusters through two equations:

\[ k^* \leftarrow \arg\min_{1 \leq k \leq K} -2 \log w_k + 2 \sum_{j=1}^{P} f(\tilde{c}_{k,j}; 1/\beta, \lambda_{k,j}) \]

\[ \hat{c}_j \leftarrow s(\tilde{c}_{k^*,j}; 1/\beta, \lambda_{k^*,j}), \quad \text{for all } 1 \leq j \leq P \]

- Where \( f \) and \( s \) were arising from:

\[
\begin{align*}
    f(x; \sigma, \lambda) &= \log \int_{\mathbb{R}} \frac{1}{2\pi\sigma\lambda} \exp \left( -\frac{(t-x)^2}{2\sigma^2} - \frac{t^2}{2\lambda^2} \right) \, dt \\
    s(x; \sigma, \lambda) &\in \arg\min_{t \in \mathbb{R}} \frac{(t-x)^2}{2\sigma^2} + \frac{t^2}{2\lambda^2}
\end{align*}
\]
How to extend EPLL to GGMM patch priors?

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

\[
\begin{align*}
    k^* & \leftarrow \text{argmin}_{1 \leq k \leq K} -2 \log w_k + 2 \sum_{j=1}^{P} f(\tilde{c}_{k,j}; 1/\beta, \lambda_{k,j}, \nu_{k,j}) \\
    \hat{c}_j & \leftarrow s(\tilde{c}_{k^*,j}; 1/\beta, \lambda_{k^*,j}, \nu_{k^*,j}), \quad \text{for all } 1 \leq j \leq P
\end{align*}
\]

- Where \( f \) and \( s \) can be updated as:

\[
\begin{align*}
    f(x; \sigma, \lambda, \nu) &= \log \int_{\mathbb{R}} \frac{1}{\sqrt{2\pi} \sigma} \frac{\kappa_{\nu}}{2 \lambda_{\nu}} \exp\left(-\frac{(t - x)^2}{2\sigma^2} - \frac{|t|^\nu}{\lambda_{\nu}^\nu}\right) dt \\
    s(x; \sigma, \lambda, \nu) &\in \text{argmin}_{t \in \mathbb{R}} \frac{(t - x)^2}{2\sigma^2} + \frac{|t|^\nu}{\lambda_{\nu}^\nu}
\end{align*}
\]
Part 3/3: GGMM-EPLL

GGMM-EPLL Algorithm

(i) For each $\hat{z}$, evaluate discrepancy against $\Sigma$, $\nu$ and $\beta$.
- Extract eigendecomposition $\Sigma = U\Lambda U^T$.
- Consider noise level being $\sigma = \frac{1}{\sqrt{\beta}}$.
- Perform whitening $\hat{\hat{z}} = U^T \hat{z}$.
- For each dimension $j$:
  - Get $\lambda^2 = \Lambda_{jj}$ and $\nu = \nu_j$.
  - Evaluate $f_{\sigma,\lambda}^\nu(\hat{\hat{x}}_j)$.
- Sum discrepancies for all dimensions $j$.

(ii) Pick the optimal $k^*$.

Patch denoising step with GGMM prior
- For all $k$
- Add $-\log w_k$
- $\hat{\hat{z}}$
- $\hat{\hat{\hat{z}}}$
- $\hat{\hat{\hat{\hat{\hat{z}}}}}$

(iii) Whitening and (iv) non-linear shrinkage for $\Sigma$, $\nu$ and $\beta$.
- For each dimension $j$:
  - Get $\lambda^2 = \Lambda_{jj}$ and $\nu = \nu_j$.
  - Shrink $\hat{x}_j = \hat{s}_{\nu,\lambda}^\nu(\hat{\hat{x}}_j)$.
- Reconstruct $\hat{z} = U \hat{x}$.

Discrepancy function:

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

Shrinkage function:

$$s_{\sigma,\lambda}^\nu(x) \in \arg\min_{t \in \mathbb{R}} \frac{(t - x)^2}{2\sigma^2} + \frac{|t|^{\nu}}{\lambda_\nu^{\nu}}$$
GGMM-EPLL Algorithm

(i) For each \( \tilde{z} \), evaluate discrepancy against \( \Sigma, \nu \) and \( \beta \).

- Extract eigendecomposition \( \Sigma = U \Lambda U^t \).
- Consider noise level being \( \sigma = \frac{1}{\sqrt{\beta}} \).
- Perform whitening \( \tilde{x} = U^t \tilde{z} \).
- For each dimension \( j \):
  - Get \( \lambda^2 = \Lambda_{jj} \) and \( \nu = \nu_j \).
  - Evaluate \( \hat{f}_{\nu, \lambda}^{\nu} (\tilde{x}_j) \).
- Sum discrepancies for all dimensions \( j \).

(ii) Pick the optimal \( k^* \).

Patch denoising step with GGMM prior

- For all \( k \)

(iii) Whitening and (iv) non-linear shrinkage for \( \Sigma, \nu \) and \( \beta \).

- Extract eigendecomposition \( \Sigma = U \Lambda U^t \).
- Consider noise level being \( \sigma = \frac{1}{\sqrt{\beta}} \).
- Perform whitening with \( \tilde{x} = U^t \tilde{z} \).
- For each dimension \( j \):
  - Get \( \lambda^2 = \Lambda_{jj} \) and \( \nu = \nu_j \).
  - Shrink \( \hat{x}_j = \hat{s}_{\nu, \lambda}^{\nu} (\tilde{x}_j) \).
- Reconstruct \( \hat{z} = U \hat{x} \).

Discrepancy function:

\[
f_{\nu, \lambda}^{\nu}(x) = \log \int_{\mathbb{R}} \frac{1}{\sqrt{2\pi\sigma}} \frac{\kappa_{\nu}}{2\lambda_{\nu}} \exp \left( -\frac{(t - x)^2}{2\sigma^2} - \frac{|t|^\nu}{\lambda_{\nu}^\nu} \right) \, dt
\]

Shrinkage function:

\[
s_{\nu, \lambda}^{\nu}(x) \in \arg\min_{t \in \mathbb{R}} \frac{(t - x)^2}{2\sigma^2} + \frac{|t|^\nu}{\lambda_{\nu}^\nu}
\]

Closed-form?
Part 3/3: GGMM-EPLL

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 $128 \times 128$ image!

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

<table>
<thead>
<tr>
<th>$\nu$</th>
<th>Shrinkage $s_{\sigma, \lambda}^{\nu}(x)$</th>
<th>Remark</th>
</tr>
</thead>
</table>
| $< 1$ | $\begin{cases} 
    x - \gamma x^{\nu-1} + O(x^{2(\nu-1)}) & \text{if } |x| \geq \tau_{\lambda}^{\nu} \\
    0 & \text{otherwise}
  \end{cases}$ | $\approx$ Hard-thresholding [Moulin, 1999] |
| $1$ | $\text{sign}(x) \max \left( |x| - \frac{\sqrt{2}\sigma^{2}}{\lambda} , 0 \right)$ | 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] |
| $3/2$ | $\text{sign}(x) \frac{(\sqrt{\gamma^2 + 4 |x|} - \gamma)^2}{4}$ | [Chaux et al., 2007] |
| $2$ | $\frac{\lambda^2}{\lambda^2 + \sigma^2} \cdot x$ | Wiener (LMMSE) |

Otherwise: No closed-forms

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

\[
\tau_{\lambda}^{\nu} = (2 - \nu)(2 - 2\nu)^{-\frac{1-\nu}{2-\nu}} (\sigma^2 \lambda_{\nu}^{-\nu} \lambda_{\nu}^{-\nu})^{\frac{1}{2-\nu}}
\]
Part 3/3: GGMM-EPLL

Shrinkage functions

**Properties:**

\[
s^\nu_{\sigma, \lambda}(x) = \sigma s^\nu_{1, \frac{\lambda}{\sigma}} \left( \frac{x}{\sigma} \right)
\]

(reduction)

\[
s^\nu_{\sigma, \lambda}(x) = -s^\nu_{\sigma, \lambda}(-x)
\]

(odd)

\[
s^\nu_{\sigma, \lambda}(x) \in \begin{cases} [0, x] & \text{if } x \geq 0 \\ [x, 0] & \text{otherwise} \end{cases}
\]

(shrinkage)

\[ x \mapsto s^\nu_{\sigma, \lambda}(x) \text{ increasing (increasing with } x) \]

\[ \lambda \mapsto s^\nu_{\sigma, \lambda}(x) \text{ increasing (increasing with } \lambda) \]

\[ \lim_{\frac{\lambda}{\sigma} \to 0} s^\nu_{1, \frac{\lambda}{\sigma}}(x) = 0 \quad \text{(kill low SNR)} \]

\[ \lim_{\frac{\lambda}{\sigma} \to +\infty} s^\nu_{1, \frac{\lambda}{\sigma}}(x) = x \quad \text{(keep high SNR)} \]
Shrinkage functions

\[ s_{\sigma, \lambda}^\nu(x) \in \arg\min_{t \in \mathbb{R}} \left( \frac{(t - x)^2}{2\sigma^2} + \frac{|t|^\nu}{\lambda^\nu} \right) \]

Choose one of the closed-form expressions by nearest neighbor on \( \nu \).
Discrepancy functions

\[ f_{\sigma,\lambda}^{\nu}(x) = \log \int_{\mathbb{R}} \frac{1}{\sqrt{2\pi\sigma}} \frac{\kappa_{\nu}}{2\lambda_{\nu}} \exp \left( -\frac{(t-x)^2}{2\sigma^2} - \frac{|t|^\nu}{\lambda_{\nu}^\nu} \right) dt \]

Properties

\[
\begin{align*}
    f_{\sigma,\lambda}^{\nu}(x) &= \log \sigma + f_{1,\lambda/\sigma}^{\nu}(x/\sigma) , & \text{(reduction)} \\
    f_{\sigma,\lambda}^{\nu}(x) &= f_{\sigma,\lambda}^{\nu}(-x) , & \text{(even)} \\
    |x| \geq |y| &\iff f_{\sigma,\lambda}^{\nu}(|x|) \geq f_{\sigma,\lambda}^{\nu}(|y|) , & \text{(unimodality)} \\
    \min_{x \in \mathbb{R}} f_{\sigma,\lambda}^{\nu}(x) &= f_{\sigma,\lambda}^{\nu}(0) > -\infty . & \text{(lower bound at 0)}
\end{align*}
\]
Discrepancy functions

\[
f_{\sigma,\lambda}^{\nu}(x) = \log \int_{\mathbb{R}} \frac{1}{\sqrt{2\pi\sigma}} \frac{\kappa_\nu}{2\lambda_\nu} \exp \left( -\frac{(t-x)^2}{2\sigma^2} - \frac{|t|^\nu}{\lambda_\nu^\nu} \right) \, dt
\]

Properties

\[
\begin{align*}
f_{\sigma,\lambda}^{\nu}(x) &= \log \sigma + f_{1,\lambda/\sigma}^{\nu}(x/\sigma) \quad \text{(reduction)} \\
f_{\sigma,\lambda}^{\nu}(x) &= f_{\sigma,\lambda}^{\nu}(-x) \quad \text{(even)} \\
|x| \geq |y| &\iff f_{\sigma,\lambda}^{\nu}(|x|) \geq f_{\sigma,\lambda}^{\nu}(|y|) \quad \text{(unimodality)} \\
\min_{x \in \mathbb{R}} f_{\sigma,\lambda}^{\nu}(x) &= f_{\sigma,\lambda}^{\nu}(0) > -\infty \quad \text{(lower bound at 0)}
\end{align*}
\]

⇒ Consider instead the log-discrepancy function \( \varphi_{\lambda}^{\nu} \):

\[
\varphi_{\lambda}^{\nu}(|x|) = \log \left[ f_{1,\lambda}^{\nu}(x) - \gamma_{\lambda}^{\nu} \right] \quad \text{and} \quad \gamma_{\lambda}^{\nu} = f_{1,\lambda}^{\nu}(0)
\]
Part 3/3: GGMM-EPLL

Discrepancy functions

\[ f_{\sigma, \lambda}^\nu(x) = \log \int_{\mathbb{R}} \frac{1}{\sqrt{2\pi\sigma}} \frac{\kappa_\nu}{2\lambda_\nu} \exp \left( -\frac{(t-x)^2}{2\sigma^2} - \frac{|t|^{\nu}}{\lambda_\nu^{\nu}} \right) dt \]

Case \( \nu = 2 \)

\[ \varphi_\lambda^2(x) = \alpha \log x + \beta , \]

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

\[ f_{\sigma, \lambda}^{\nu}(x) = \log \int_{\mathbb{R}} \frac{1}{\sqrt{2\pi}\sigma} \frac{\kappa_{\nu}}{2\lambda_{\nu}} \exp \left( -\frac{(t-x)^2}{2\sigma^2} - \frac{|t|^\nu}{\lambda_{\nu}^\nu} \right) \, dt \]

Case \( \nu = 1 \)

\[ \varphi_{\lambda}^{1}(x) \sim 0 \alpha_{1} \log x + \beta_{1}, \]

where \( \alpha_{1} = 2 \) and \( \beta_{1} = -\log \lambda + \log \left[ \frac{1}{\sqrt{\pi}} \frac{\exp \left(-\frac{1}{\lambda^2}\right)}{\text{erfc} \left( \frac{1}{\lambda} \right)} - \frac{1}{\lambda} \right] \).
Discrepancy functions

\[ f_{\sigma,\lambda}^\nu(x) = \log \int_{\mathbb{R}} \frac{1}{\sqrt{2\pi}\sigma} \frac{\kappa^\nu}{2\lambda^\nu} \exp\left(-\frac{(t-x)^2}{2\sigma^2} - \frac{|t|^\nu}{\lambda^\nu}\right) \, dt \]

**Case \( \nu = 1 \)**

\[ g(0, \nu = 1, \lambda) \ast \mathcal{N}(0, 1) \]

\[ f_{1,\lambda}^1(x) \]

\[ \varphi_{\lambda}^1(x) \]

\[ \varphi_{\lambda}^1(x) \sim \alpha_2 \log x + \beta_2 , \]

where \( \alpha_2 = 1 \) and \( \beta_2 = \frac{1}{2} \log 2 - \log \lambda \).
Discrepancy functions

\[ f_{\sigma, \lambda}(x) = \log \int_{\mathbb{R}} \frac{1}{\sqrt{2\pi \sigma}} \kappa_{\nu} \exp \left( -\frac{(t-x)^2}{2\sigma^2} - \frac{|t|^\nu}{\lambda_{\nu}} \right) dt \]

**Case** $\frac{2}{3} \leq \nu < 2$

\[ \varphi_{\lambda}^{\nu}(x) \sim 0 \log x + \beta_{1}, \]

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] dt}{\int_{-\infty}^{\infty} e^{-\frac{t^2}{2}} \exp \left[ -\left( \frac{|t|}{\lambda_{\nu}} \right)^\nu \right] dt} \right). \]
Discrepancy functions

\[ f_{\sigma, \lambda}^{\nu}(x) = \log \int_{\mathbb{R}} \frac{1}{\sqrt{2\pi\sigma}} \frac{\kappa_{\nu}}{2\lambda_{\nu}} \exp\left( -\frac{(t - x)^2}{2\sigma^2} - \frac{|t|^{\nu}}{\lambda_{\nu}^{\nu}} \right) dt \]

**Case** \( \frac{2}{3} \leq \nu < 2 \)

\[ \varphi_{\lambda}^{\nu}(x) \sim \alpha_2 \log x + \beta_2, \]

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

\[ f_{\sigma,\lambda}^{\nu}(x) = \log \int_{\mathbb{R}} \frac{1}{\sqrt{2\pi}\sigma} \frac{\kappa_{\nu}}{2\lambda_{\nu}} \exp\left( -\frac{(t-x)^2}{2\sigma^2} - \frac{|t|^{\nu}}{\lambda_{\nu}^{\nu}} \right) dt \]

**Case** \( \frac{2}{3} \leq \nu < 2 \)

\[ \varphi_{\lambda}^{\nu}(x) \sim \alpha_2 \log x + \beta_2 , \]

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

\[ f_{\sigma, \lambda}^{\nu}(x) = \log \int_{\mathbb{R}} \frac{1}{\sqrt{2\pi\sigma}} \frac{\kappa_{\nu}}{2\lambda_{\nu}} \exp \left( -\frac{(t - x)^2}{2\sigma^2} - \frac{|t|^\nu}{\lambda_{\nu}^\nu} \right) \, dt \]

Approximation

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

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

\[ f_{\nu, \lambda}(x) = \log \int_{\mathbb{R}} \frac{1}{\sqrt{2\pi} \sigma} \frac{\kappa_{\nu}}{2\lambda_{\nu}} \exp \left( -\frac{(t - x)^2}{2\sigma^2} - \frac{|t|^{\nu}}{\lambda_{\nu}^{\nu}} \right) dt \]

- Given \((\lambda/\sigma, \nu)\), get \((\gamma_{\lambda/\sigma}, \beta_1, \beta_2, h)\) from lookup tables (LUTs).
- Compute the log-discrepancy based on asymptoptics and softplus.
  - Deduce the discrepancy.
### Performance in denoising

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

GGMM offers best performance in average compared to GMM/LMM/HLMM.
Part 3/3: GGMM-EPLL

Performance in denoising

<table>
<thead>
<tr>
<th>$\sigma$</th>
<th>Algo.</th>
<th>BSDS</th>
<th>barbara</th>
<th>camera</th>
<th>man</th>
<th>hill</th>
<th>house</th>
<th>lena</th>
<th>mandrill</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>BM3D</td>
<td>37.33</td>
<td>38.30</td>
<td>38.28</td>
<td>36.04</td>
<td>39.82</td>
<td>38.70</td>
<td>35.26</td>
<td>37.36</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GGMM</td>
<td>37.33</td>
<td>37.73</td>
<td>38.12</td>
<td>35.95</td>
<td>38.94</td>
<td>38.52</td>
<td>35.23</td>
<td>37.33</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>BM3D</td>
<td>33.06</td>
<td>34.95</td>
<td>34.10</td>
<td>31.88</td>
<td>36.69</td>
<td>35.90</td>
<td>30.58</td>
<td>33.15</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GGMM</td>
<td>33.10</td>
<td>33.87</td>
<td>34.01</td>
<td>31.81</td>
<td>35.72</td>
<td>35.59</td>
<td>30.58</td>
<td>33.15</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>BM3D</td>
<td>29.38</td>
<td>31.73</td>
<td>30.42</td>
<td>28.56</td>
<td>33.81</td>
<td>33.02</td>
<td>26.60</td>
<td>29.50</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GGMM</td>
<td>29.43</td>
<td>30.02</td>
<td>30.24</td>
<td>28.48</td>
<td>33.03</td>
<td>32.59</td>
<td>26.64</td>
<td>29.50</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>BM3D</td>
<td>26.28</td>
<td>27.97</td>
<td>27.16</td>
<td>25.89</td>
<td>30.69</td>
<td>29.81</td>
<td>23.07</td>
<td>26.38</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GGMM</td>
<td>26.26</td>
<td>26.17</td>
<td>27.03</td>
<td>25.70</td>
<td>29.89</td>
<td>29.42</td>
<td>23.21</td>
<td>26.32</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GGMM</td>
<td>24.64</td>
<td>24.03</td>
<td>25.17</td>
<td>24.25</td>
<td>27.80</td>
<td>27.52</td>
<td>21.50</td>
<td>24.67</td>
<td></td>
</tr>
</tbody>
</table>

GGMM-EPLL offers slightly worse performance than BM3D.
Part 3/3: GGMM-EPLL

Performance in denoising

GMM

30.43 / 0.8678

GMM

30.17 / 0.8690
Part 3/3: GGMM-EPLL

Performance in denoising

GGMM

GGMM

30.47 / 0.8686

30.28 / 0.8681
Performance in denoising

30.39 / 0.8666  LMM

30.04 / 0.8578  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


cdeledal@math.u-bordeaux.fr

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

Presentation produced using MooseTEX
http://www.math.u-bordeaux.fr/~cdeledal/moosetex
Appendix – EM for GGMMs

• **Expectation step (E-Step)**
  - For all $k = 1, \ldots, K$ and samples $i = 1, \ldots, n$, compute:
    \[
    \xi_{k,i} \leftarrow \frac{w_k G(z_i; 0_P, \Sigma_k, \nu_k)}{\sum_{l=1}^{K} w_l G(z_i; 0_P, \Sigma_l, \nu_l)}.
    \]

• **Moment step (M-Step)**
  - For all components $k = 1, \ldots, K$, update:
    \[
    w_k \leftarrow \frac{\sum_{i=1}^{n} \xi_{k,i}}{\sum_{l=1}^{K} \sum_{i=1}^{n} \xi_{l,i}} \quad \text{and} \quad \Sigma_k \leftarrow \frac{\sum_{i=1}^{n} \xi_{k,i} z_i z_i^t}{\sum_{i=1}^{n} \xi_{k,i}}.
    \]
  - Perform eigen decomposition of $\Sigma_k$:
    \[
    \Sigma_k = U_k \Lambda_k U_k^t \quad \text{where} \quad \Lambda_k = \text{diag}(\lambda_{k,1}, \lambda_{k,2}, \ldots, \lambda_{k,P})^2.
    \]
  - For all $k = 1, \ldots, K$ and dimensions $j = 1, \ldots, P$, compute:
    \[
    \chi_{k,j} \leftarrow \frac{\sum_{i=1}^{n} \xi_{k,i} |(U_k^t z_i)_j|}{\sum_{i=1}^{n} \xi_{k,i}} \quad \text{and} \quad (\nu_k)_j \leftarrow \Pi_{[.3,2]} \left[ F^{-1} \left( \frac{\chi_{k,j}^2}{\lambda_{k,j}^2} \right) \right].
    \]

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 \(0.3 \leq \nu \leq 2\) and SNR \(\lambda/\sigma\).
Figure 2 – Lookup tables used to store the values of the parameters $\gamma, \nu, \lambda, \beta_1, \beta_2$ and $h$ for various $0.3 \leq \nu \leq 2$ and $10^{-3} \leq \lambda \leq 10^3$. A regular grid of 100 values has been used for $\nu$ and a logarithmic grid of 100 values has been used for $\lambda$. This leads to a total of 10,000 combinations for each of the four lookup tables.