Accelerating GMM-based patch priors for image restoration: Three ingredients for a 100× speed-up

Shibin Parameswaran[1], Charles-Alban Deledalle[1,2], Loïc Denis[3] and Truong Q. Nguyen[1]

[1]UC San Diego, CA, USA, [2]CNRS, Univ. Bordeaux, France, [3]Univ Lyon, UJM-Saint-Etienne, France

Objectives

Our goal is to develop a fast and efficient image restoration algorithm utilizing Gaussian Mixture Model (GMM) patch priors. To this end, we perform:

1. detailed complexity analysis of Expected Patch Log-Likelihood (EPLL) algorithm
2. introduce innovative approximations to accelerate EPLL by a factor of 100

Introduction

We consider the problem of estimating an image $\boldsymbol{x} \in \mathbb{R}^N$ (N is the number of pixels) from noisy linear observations:

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

where:

$\mathcal{A} : \mathbb{R}^N \to \mathbb{R}^M$ is a linear operator

$\boldsymbol{w} \in \mathbb{R}^M$ is i.i.d noise component from $\mathcal{N}(0, \sigma^2)$

The EPLL algorithm is a image restoration method that uses a Gaussian mixture model (GMM) prior on natural image patches.

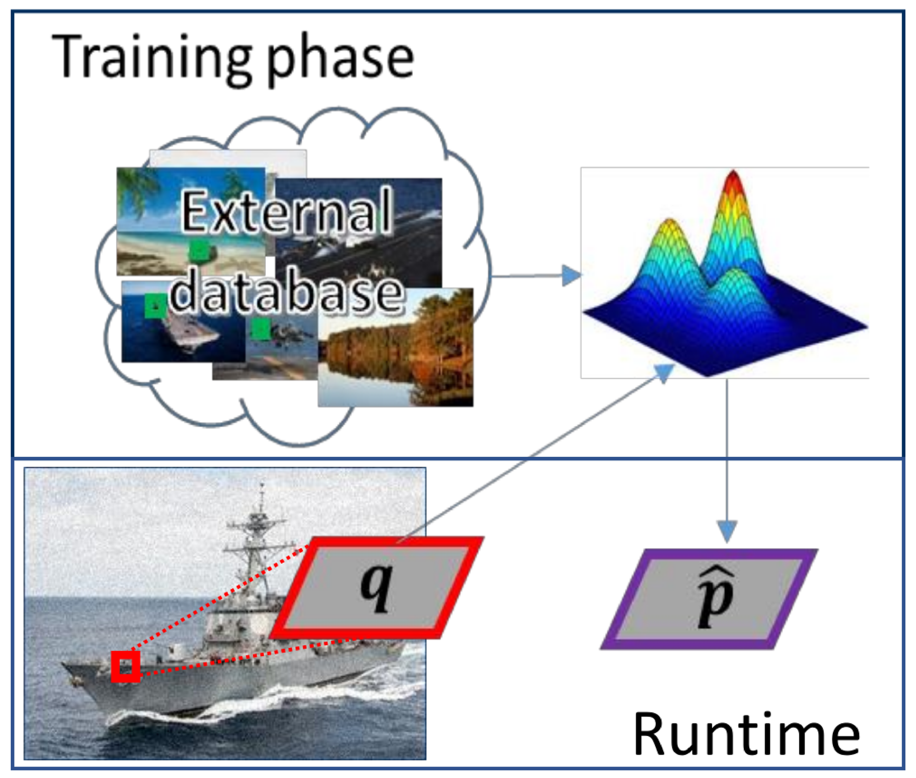
$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \ \frac{P}{2\sigma^2}\|\mathcal{A}\boldsymbol{x} - \boldsymbol{y}\|^2 - \sum_{i \in \mathcal{I}} \log p\left(\mathcal{P}_i \boldsymbol{x}\right)$$

where

$\mathcal{I} = \{1, \ldots, N\}$ is the set of pixel indices

$\mathcal{P}_i : \mathbb{R}^N \to \mathbb{R}^P$ is the patch extractor at pixel i.

$p(\cdot) = \sum_{k=1}^{K} w_k \mathcal{N}(0, \Sigma_k)$ is zero-mean GMM prior



EPLL Solution

Use Half Quadratic Splitting strategy

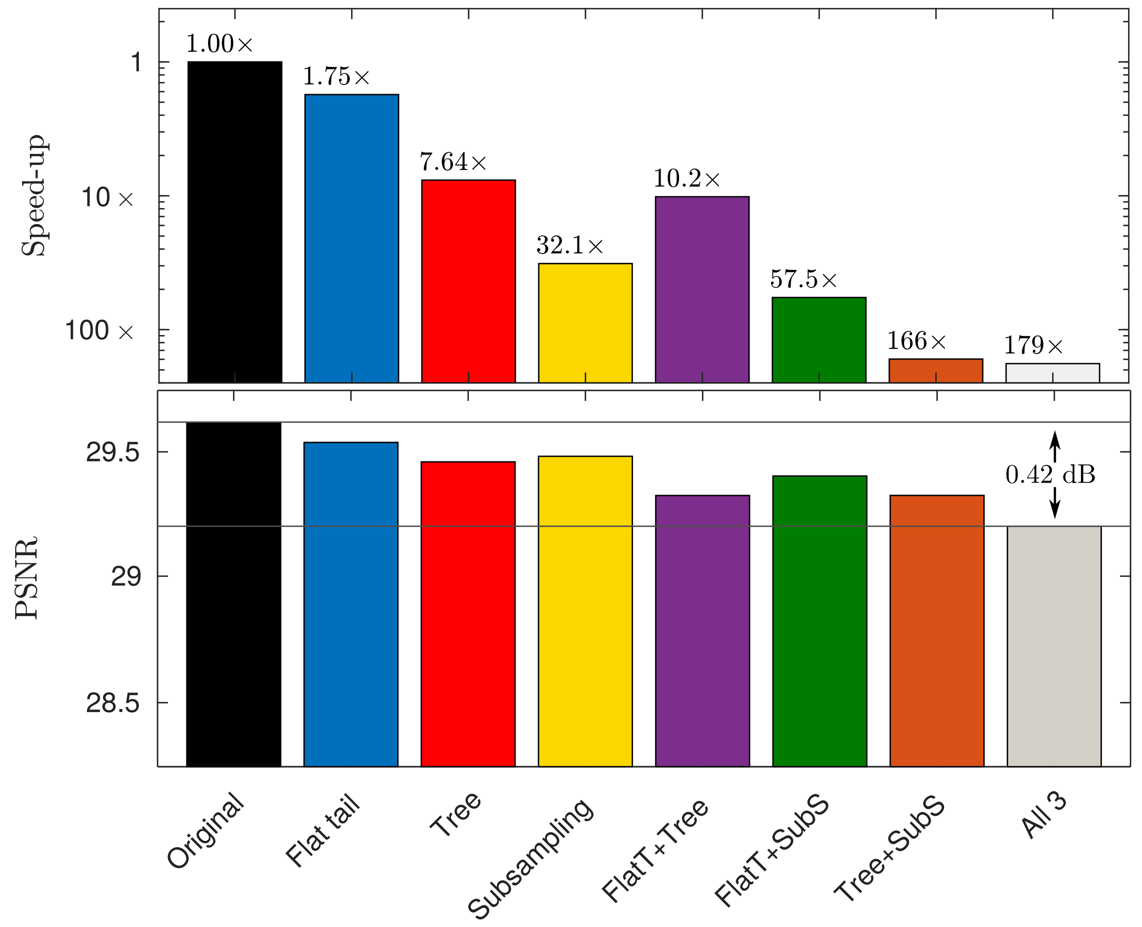
$$\underset{\boldsymbol{x}, \boldsymbol{z}_1, \ldots, \boldsymbol{z}_N}{\operatorname{argmin}} \ \frac{P}{2\sigma^2}\|\mathcal{A}\boldsymbol{x} - \boldsymbol{y}\|^2 + \frac{\beta}{2}\sum_{i \in \mathcal{I}} \|\mathcal{P}_i \boldsymbol{x} - \boldsymbol{z}_i\|^2 - \sum_{i \in \mathcal{I}} \log p\left(\boldsymbol{z}_i\right) \quad (1)$$

Solve (1) by an alternating optimization scheme:

$$\left\{ \hat{\boldsymbol{z}}_i \leftarrow \underset{\boldsymbol{z}_i}{\operatorname{argmin}} \ \frac{\beta}{2}\|\mathcal{P}_i \hat{\boldsymbol{x}} - \boldsymbol{z}_i\|^2 - \log p\left(\boldsymbol{z}_i\right) \right\}_{i=1..N}$$

$$\hat{\boldsymbol{x}} \leftarrow \underset{\boldsymbol{x}}{\operatorname{argmin}} \ \frac{P}{2\sigma^2}\|\mathcal{A}\boldsymbol{x} - \boldsymbol{y}\|^2 + \frac{\beta}{2}\sum_{i \in \mathcal{I}} \|\mathcal{P}_i \boldsymbol{x} - \hat{\boldsymbol{z}}_i\|^2.$$

Main steps of EPLL shown in Algorithm 1.



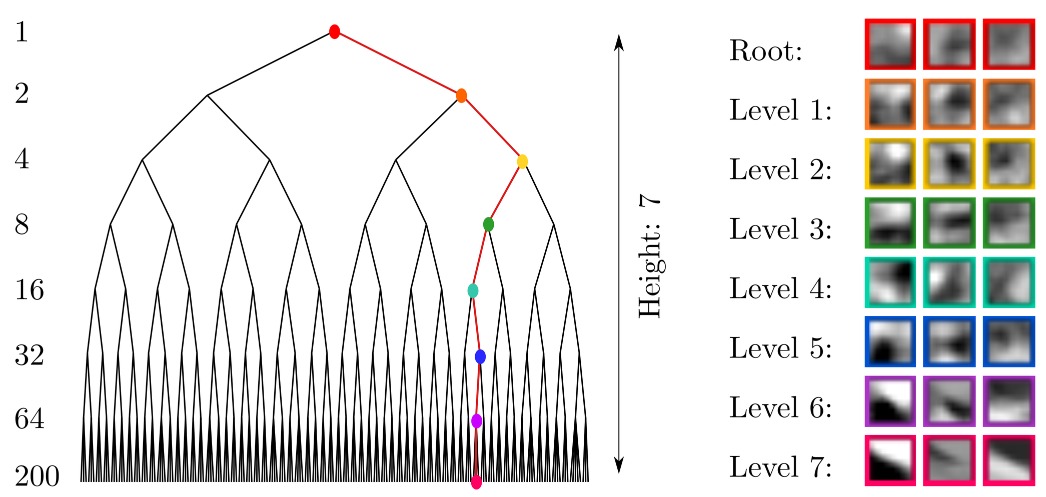
More than 100× speed-up obtained due to the proposed accelerations

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

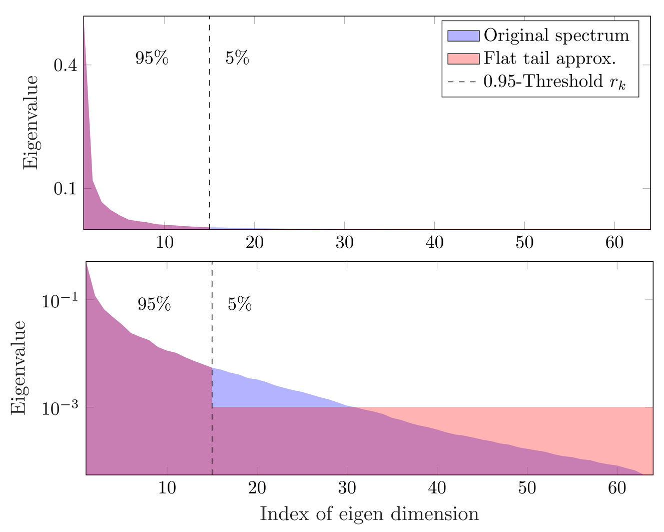
$$\mathcal{O}(NP^2K) \to \mathcal{O}(NP\bar{r}\log K/s^2)$$

Acceleration strategies



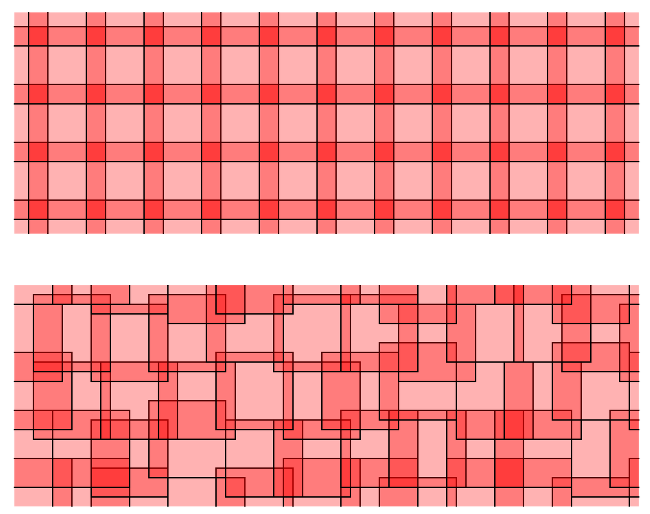
Binary search tree

Gaussian selection: $\mathcal{O}(NKP^2)$
Patch estimation: $\mathcal{O}(NP^2)$
Patch extraction: $\mathcal{O}(NP)$
Patch reprojection: $\mathcal{O}(NP)$



Flat trail approximation

Gaussian selection: $\mathcal{O}(NKP^2)$
Patch estimation: $\mathcal{O}(NP^2)$
Patch extraction: $\mathcal{O}(NP)$
Patch reprojection: $\mathcal{O}(NP)$



Random subsampling

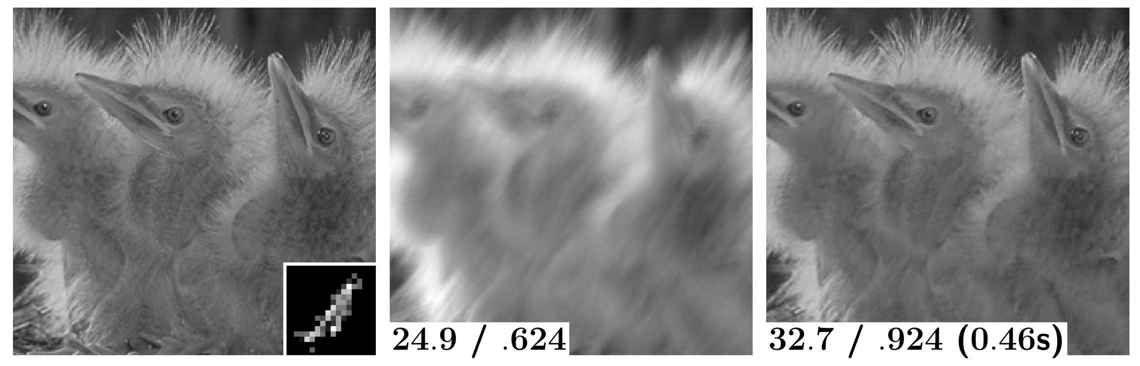
Gaussian selection: $\mathcal{O}(NKP^2)$
Patch estimation: $\mathcal{O}(NP^2)$
Patch extraction: $\mathcal{O}(NP)$
Patch reprojection: $\mathcal{O}(NP)$

Results

Denoising:

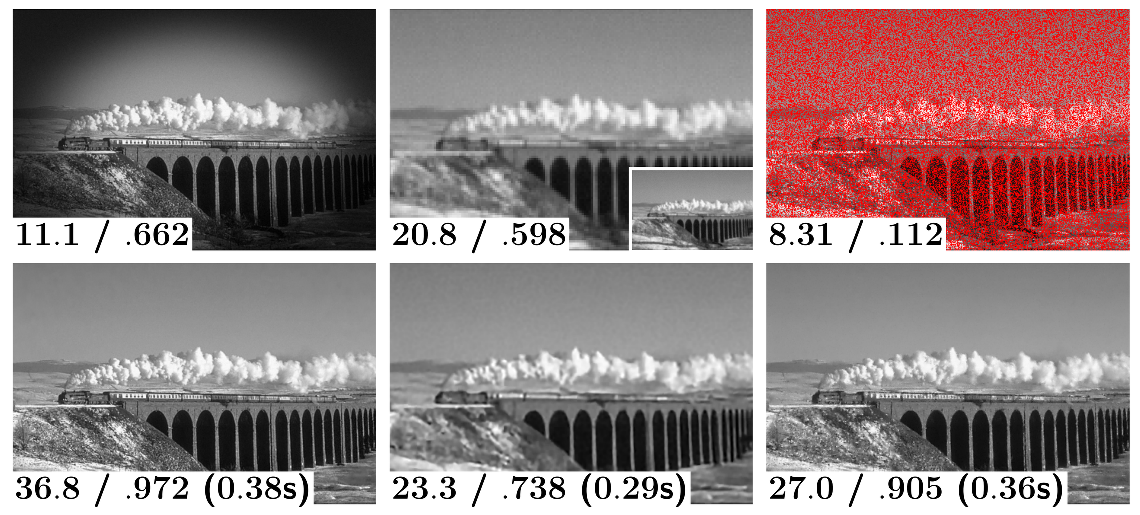
(a) Noisy image (b) EPLLc result (c) FEPLL result

Deblurring:



(d) Reference (e) Blurry image (f) FEPLL result

Other inverse problems:



(g) devignetting (h) ×3 super-resolution (i) 50% inpainting

Conclusion

We accelerate EPLL by a factor greater than 100 with negligible loss of image quality (<0.5dB). The speed-up is achieved solely by reducing the algorithmic complexity of EPLL. The genericity of our acceleration strategies makes the algorithm applicable to different inverse problems without re-training.

References

[1] Zoran, Daniel and Weiss, Yair.
From learning models of natural image patches to whole image restoration.
In *International Conference on Computer Vision*, pages 479–486. IEEE, November 2011.