Linear Expansion of Thresholds

A Tool for Approximating Image Processing Algorithms

Thierry Blu

Department of Electronic Engineering
The Chinese University of Hong Kong

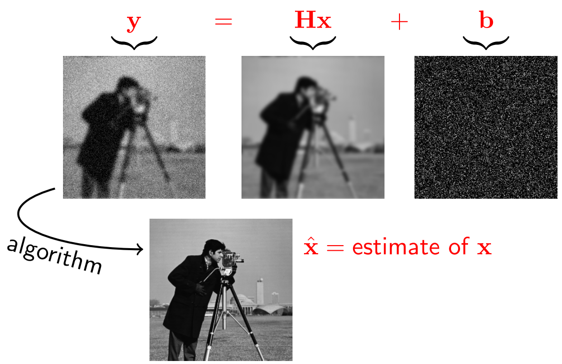
CISP-BMEI
October 15, 2016

Outline

Outline

Image restoration

Image processing algorithms frequently amount to transforming an input image into another ("better") one. Main example: Image restoration



NOTE: A more general formulation would be $\mathbf{y} \sim \mathscr{P}\{\mathbf{y}|\mathbf{Hx}\}$.

Image restoration

Standard approaches for image restoration

- Bayesian: both $\mathbf{x}$ and $\mathbf{y}$ are *random* with *known* joint probability
  - Maximum a Posteriori $\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} \mathscr{P}\{\mathbf{x}|\mathbf{y}\}$
  - Minimum MSE $\hat{\mathbf{x}} = \mathscr{E}\{\mathbf{x}|\mathbf{y}\}$
- Regularization: Wiener/Tikhonov, total-variation, ℓ^1
  $$\min_{\mathbf{x}} \|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2 + \lambda\|\mathbf{D}\mathbf{x}\|_{\ell^1}$$
- Filtering: Bilateral Filter, median
- Patch-based: Non-Local Means, BM3D
- etc.

Image restoration is usually viewed as an *approximation* problem on $\mathbf{x}$.

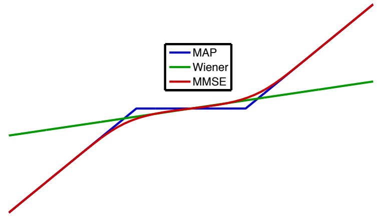
Bayesian denoising example

Assuming a *Laplace* prior $\mathscr{P}\{\mathbf{x}\} = \prod_{n=1}^{N} \frac{\lambda}{2} \mathrm{e}^{-\lambda|x_n|}$ and additive *Gaussian* white noise with variance σ^2, these statistical approaches yield pointwise thresholding involving $T = \lambda\sigma^2$:

MAP $\hat{x}_n = \mathrm{soft}_T(y_n)$

Wiener $\hat{x}_n = \dfrac{y_n}{1 + \frac{T^2}{2\sigma^2}}$

MMSE $\hat{x}_n = y_n - T \dfrac{\mathrm{e}^{-\lambda y_n} \operatorname{erfc}\left(\frac{-y_n+T}{\sigma\sqrt{2}}\right) - \mathrm{e}^{\lambda y_n} \operatorname{erfc}\left(\frac{y_n+T}{\sigma\sqrt{2}}\right)}{\mathrm{e}^{-\lambda y_n} \operatorname{erfc}\left(\frac{-y_n+T}{\sigma\sqrt{2}}\right) + \mathrm{e}^{\lambda y_n} \operatorname{erfc}\left(\frac{y_n+T}{\sigma\sqrt{2}}\right)}$



Function Approximation

Instead of considering the restoration problem as an *image approximation* problem, consider it as a *function approximation* problem:

find a "good" $\mathbf{F}(\cdot)$ such that $\hat{\mathbf{x}} = \mathbf{F}(\mathbf{y})$

Lay the emphasis on how the restoration $\hat{\mathbf{x}}$ changes when the observation $\mathbf{y}$ changes — not on the pixelwise description of $\hat{\mathbf{x}}$.

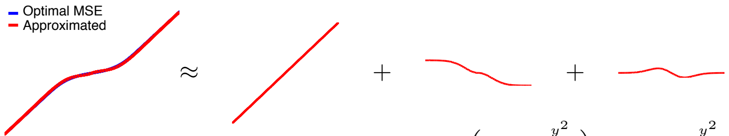
Use of standard linear approximation techniques to parametrize the processing $\mathbf{F}(\cdot) \rightsquigarrow$ find a good representation basis.

Linear approximation

Functions can often be efficiently approximated onto adapted bases.

Standard bases: wavelets ($\mathbf{L}^2$ functions), sinc kernels (bandlimited functions), radial basis functions (scattered points interpolation), etc.

Example with the MMSE Laplace prior denoising function:



$$\approx \quad a \times y \quad + \quad b \times \mathrm{sign}(y)\left(1 - \mathrm{e}^{-\frac{y^2}{2T^2}}\right) \quad + \quad c \times y\mathrm{e}^{-\frac{y^2}{2T^2}}$$

Linear Expansion of Thresholds

An approximation of the optimal denoising process as a (finite) linear combination of elementary processes

$$\mathbf{F}(\mathbf{y}) = \sum_{k=1}^{K} a_k \mathbf{F}_k(\mathbf{y})$$

In image denoising problems, $\mathbf{F}_k(\mathbf{y})$ are thresholding functions in some *sparse* transformed domain.

The linear space approximation proves particularly useful when combined with a *quadratic optimization criterion* (e.g., MSE or SURE), as the optimization boils down to solving a *linear system of equations*.

The idea of LET is that a genuine *approximation* of the optimal processing can be sufficient, while having useful *linear* properties.

LET optimization

Several examples of use of this approximation of processings

- Minimization of the MSE (or an estimate of) for
  - image denoising
  - image deconvolution
- Iterative minimization of an ℓ^1 criterion

Outline

Minimum MSE

The minimization of the MSE, $\|\mathbf{F}(\mathbf{y}) - \mathbf{x}\|^2$, for the LET coefficients a_k yields, for all $k = 1, 2, \ldots, K$

$$\sum_{l=1}^{K} \mathbf{F}_k(\mathbf{y})^{\mathrm{T}} \mathbf{F}_l(\mathbf{y})\, a_l = \mathbf{F}_k(\mathbf{y})^{\mathrm{T}} \mathbf{x}$$

This also boils down to solving a *linear system of equations*

$$\mathbf{a} = \mathbf{M}^{-1}\mathbf{c}' \quad \text{where} \quad \left| \begin{array}{l} \mathbf{M} = \left[\mathbf{F}_k(\mathbf{y})^{\mathrm{T}} \mathbf{F}_l(\mathbf{y})\right]_{1\le k,l\le K} \\ \mathbf{c}' = \left[\mathbf{F}_k(\mathbf{y})^{\mathrm{T}} \mathbf{x}\right]_{1\le k\le K} \end{array} \right.$$

But, how to evaluate the MSE, since $\mathbf{x}$ is unknown?

$\rightsquigarrow$ *Stein's Unbiased Risk Estimate*

Stein's Unbiased Risk Estimate (SURE)

MSE estimation

Consider the random variable[a]

$$\mathrm{SURE}(\mathbf{y}) = \frac{1}{N}\|\mathbf{F}(\mathbf{y}) - \mathbf{y}\|^2 + \frac{2\sigma^2}{N}\mathrm{div}\,\{\mathbf{F}(\mathbf{y})\} - \sigma^2$$

Under the *additive white Gaussian noise* hypothesis, this random variable is an *unbiased estimate of the MSE* Stein et al. 1981

$$\mathscr{E}\,\{\mathrm{SURE}(\mathbf{y})\} = \mathscr{E}\,\{\|\mathbf{F}(\mathbf{y}) - \mathbf{x}\|^2/N\}$$

The SURE is all the closer to the MSE as N is larger.

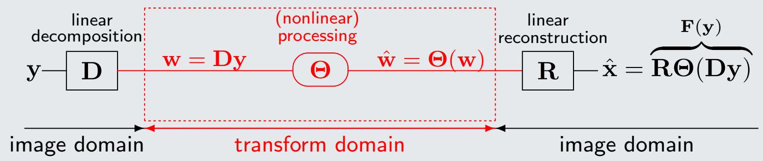
[a]Divergence operator: $\mathrm{div}\,\{\mathbf{F}(\mathbf{y})\} \overset{\mathrm{def}}{=} \sum_k \frac{\partial F_k(\mathbf{y})}{\partial y_k}$.

The original signal $\mathbf{x}$ may, or may not be random.
No assumptions on $\mathbf{x}$ are needed.

Transform-domain denoising

Consider a "sparsifying" linear transformation $\mathbf{D}$ (DCT, wavelet, etc.) and another linear transformation $\mathbf{R}$ such that $\mathbf{RD} = \mathbf{Id}$.

Transform-domain thresholding



The LET basis $\mathbf{F}_k(\mathbf{y})$ is then specified as follows

$$\boldsymbol{\Theta}(\mathbf{w}) = \sum_{k=1}^{K} a_k \boldsymbol{\Theta}_k(\mathbf{w}) \rightsquigarrow \mathbf{F}_k(\mathbf{y}) = \mathbf{R}\boldsymbol{\Theta}_k(\mathbf{D}\mathbf{y})$$

NOTE: The transformations involved may or may not be redundant.

Example: nonredundant wavelet thresholding

Choice of an orthonormal wavelet transform transform can be used (e.g., symlet 8). Then, the processing in subband j is a simple thresholding $\hat{w}_{j,n} = \theta_j(w_{j,n})$ for each of the coordinates $n = 1, 2, \ldots, N_j$ of $\mathbf{w}_j$, and

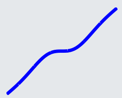
$$\mathrm{SURE}_j(\mathbf{w}_j) = \frac{1}{N_j}\left(\sum_{n=1}^{N_j} |\theta_j(w_{j,n}) - w_{j,n}|^2 + 2\sigma^2\theta_j'(w_{j,n})\right) - \sigma^2$$

SURE-LET simple threshold

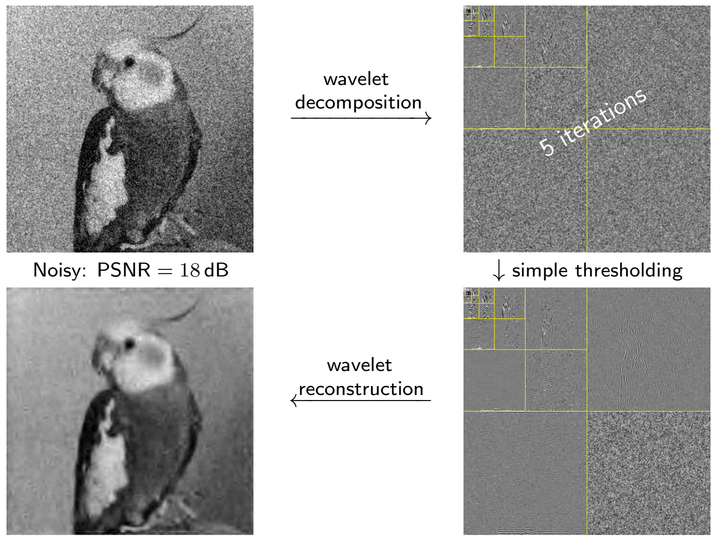
A two-parameter zone-selection function

$$\theta_j(w) = a_j w + b_j w e^{-\frac{w^2}{12\sigma^2}}$$

where a_j and b_j are obtained by minimizing $\mathrm{SURE}_j(\mathbf{w}_j)$.



NOTE: SureShrink Donoho 1995 makes the choice $\theta_j(w) = \mathrm{soft}_{T_j}(w)$ and minimizes $\mathrm{SURE}_j(\mathbf{w}_j)$ for T_j.



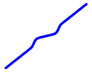
Noisy: PSNR = 18 dB

Denoised: PSNR = 29.06 dB (SureShrink: PSNR = 28.73 dB)

Example: undecimated wavelet thresholding

Hard-like[1] thresholding rule

In each wavelet subband j, the noisy coefficients are thresholded using



$$\theta_j(w) = a_j w + b_j w\left(1 - \mathrm{e}^{-\left(\frac{w}{3\sigma}\right)^8}\right)$$

where (a_j, b_j) change from subband to subband — i.e., two parameters per subband.

The optimal set of parameters $\{a_j, b_j\}$ is then found by minimizing the global image-domain SURE.

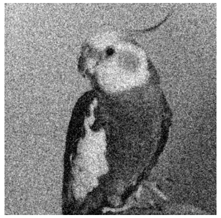
NOTE: For J iterations of the wavelet transform, $J \times 3 \times 2$ LET coefficients have to be found: 30 for 5 iterations.

[1]Hard threshold cannot be optimized using SURE (not differentiable).

Undecimated results

Undecimated discrete symlet 8 transform

Noisy



PSNR=18 dB

SureShrink

PSNR=28.73 dB

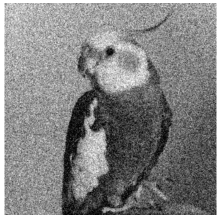
SURE-LET

PSNR=31.15 dB

Undecimated results

Undecimated discrete Haar wavelet transform

Noisy



PSNR=18 dB

SureShrink

PSNR=28.73 dB

SURE-LET

PSNR=31.91 dB

NOTE: Surprisingly, it is the simplest wavelet type (Haar) that works best. Shortest support?

Recapitulation of the SURE-LET approach

1. Instead of finding an *approximation of the signal* $\mathbf{x}$, find an *approximation of the processing* $\mathbf{F}(\mathbf{y})$ that transforms $\mathbf{y}$ into $\hat{\mathbf{x}}$;
2. Instead of minimizing the MSE between $\hat{\mathbf{x}}$ and $\mathbf{x}$, minimize an (unbiased) *estimate* of this MSE, based on $\mathbf{y}$ alone (SURE);
3. Express $\mathbf{F}(\mathbf{y})$ as a linear decomposition (LET) $\sum_k a_k \mathbf{F}_k(\mathbf{y})$ of basis processings $\mathbf{F}_k(\mathbf{y})$ $\rightsquigarrow$ linear system of equations (fast, unique).

NOTE: The number K of elementary processings is chosen very small (usually, $K < 200$), compared to the number of pixels N.
$\rightsquigarrow$ faster algorithm, and better agreement between MSE and SURE.

Extensions

- **Multivariate** wavelet thresholding: taking into account both *interscale* and *local* wavelet dependencies;
- Thresholding (possibly multivariate) in a **dictionary** of transforms.
- **Multiframe** video denoising: involving motion compensation;

Dictionary of two transforms (UWT Haar & 12×12-BDCT)

Noisy

PSNR=18 dB

SURE-LET Dictionary

PSNR=28.41 dB

SURE-LET Multivariate Dictionary

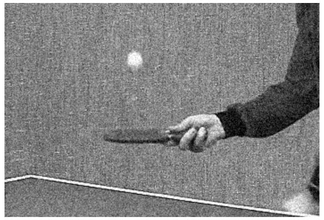
PSNR=28.80 dB

Extensions

- **Multivariate** wavelet thresholding: taking into account both *interscale* and *local* wavelet dependencies;
- Thresholding (possibly multivariate) in a **dictionary** of transforms.
- **Multiframe** video denoising: involving motion compensation;

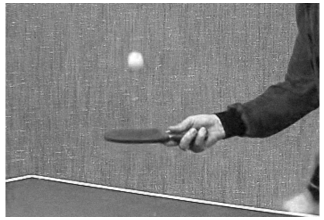
Orthonormal discrete symlet 8 transform

Noisy



PSNR=22.11 dB

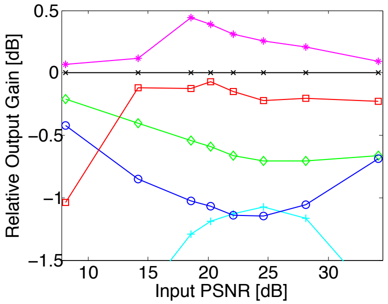
Multiframe SURE-LET



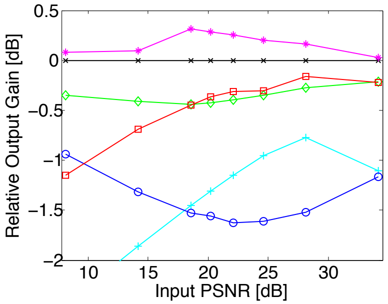
PSNR=30.85 dB

Results: PSNR Comparisons

Peppers 256×256



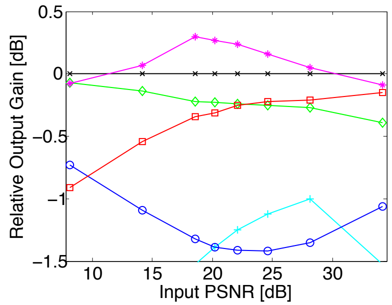
Coco 256×256



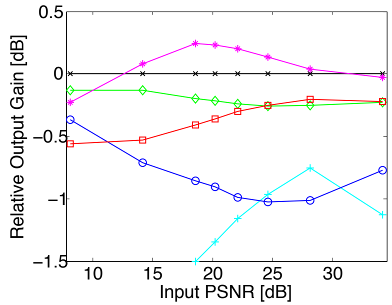
Multivariate SURE-LET (baseline)
NLmeans Buades *et al.* 2005
BLS-GSM Portilla *et al.* 2003
BM3D Dabov *et al.* 2007
Fast TV Chambolle 2004
K-SVD Elad & Aharon 2006

Results: PSNR Comparisons

Boat 512×512



Goldhill 512×512



Multivariate SURE-LET (baseline)
NLmeans Buades *et al.* 2005
BLS-GSM Portilla *et al.* 2003
BM3D Dabov *et al.* 2007
Fast TV Chambolle 2004
K-SVD Elad & Aharon 2006

Results: Visual Comparisons

Original

Average SSIM: 1.000

Noisy

Average SSIM: 0.263

Results: Visual Comparisons

Original

Average SSIM: 1.000

Multivariate SURE-LET

Average SSIM: 0.739

Results: Visual Comparisons

NLmeans

Average SSIM: 0.662

Multivariate SURE-LET

Average SSIM: 0.739

Results: Visual Comparisons

Fast TV

Average SSIM: 0.704

Multivariate SURE-LET

Average SSIM: 0.739

Results: Visual Comparisons

BLS-GSM

Average SSIM: 0.732

Multivariate SURE-LET

Average SSIM: 0.739

Results: Visual Comparisons

K-SVD

Average SSIM: 0.711

Multivariate SURE-LET

Average SSIM: 0.739

Results: Visual Comparisons

BM3D

Average SSIM: 0.754

Multivariate SURE-LET

Average SSIM: 0.739

Convolution SURE

Consider $\mathbf{H}_\beta^{-1} = (\mathbf{H}^{\mathrm{T}}\mathbf{H} + \beta\mathbf{S}^{\mathrm{T}}\mathbf{S})^{-1}\mathbf{H}^{\mathrm{T}}$ an approximate inverse of $\mathbf{H}$ where $\mathbf{S}$ is such[1] that $\|\mathbf{H}_\beta^{-1}\mathbf{H}\mathbf{x}-\mathbf{x}\| \ll \|\mathbf{x}\|$ and β is a constant that depends on the noise variance only.

MSE estimation

Under the *additive white Gaussian noise* hypothesis, the random variable

$$\mathsf{SURE}(\mathbf{y}) = \frac{1}{N}\|\mathbf{F}(\mathbf{y}) - \mathbf{H}_\beta^{-1}\mathbf{y}\|^2 + \frac{2\sigma^2}{N}\mathrm{div}\left\{\mathbf{H}_\beta^{-1}\mathbf{F}(\mathbf{y})\right\} - \sigma^2\|\mathbf{H}_\beta^{-1}\|^2_{\mathrm{Fro}}$$

is such that: $\mathscr{E}\{\mathsf{SURE}(\mathbf{y})\} \approx \mathscr{E}\left\{\|\mathbf{F}(\mathbf{y}) - \mathbf{x}\|^2/N\right\}$.

NOTE: Contrary to the denoising application, it is necessary to add a hypothesis on $\mathbf{x}$ to find a reliable estimate of the MSE.

[1] for usual images, the operator $\mathbf{S}$ is typically a high-pass filter (Laplacian).

Example: Wiener Deconvolution

The result of the the ℓ^2 regularization problem $\|\mathbf{Hx}-\mathbf{y}\|^2+\lambda\|\mathbf{Sx}\|^2$ is

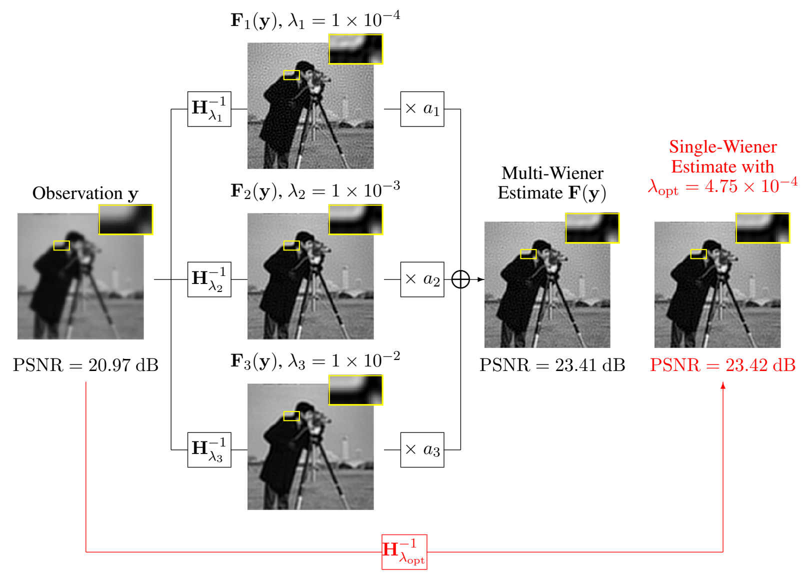
$$\hat{\mathbf{x}} = \mathbf{F}(\mathbf{y}) = \underbrace{(\mathbf{H}^{\mathrm{T}}\mathbf{H}+\lambda\mathbf{S}^{\mathrm{T}}\mathbf{S})^{-1}\mathbf{H}^{\mathrm{T}}}_{\mathbf{H}_\lambda^{-1}}\mathbf{y}$$

where the parameter λ should be optimized by minimizing the MSE $\|\hat{\mathbf{x}}-\mathbf{x}\|^2$ or by minimizing the convolution SURE — non-linear minimization.

However, it is also possible to approximate the processing $\mathbf{F}(\cdot)$ as a LET with K basis elements

$$\mathbf{F}_k(\mathbf{y}) = \mathbf{H}_{\lambda_k}^{-1}\mathbf{y}, \qquad k = 1, 2, \dots, K$$

where λ_k are fixed. Then, the SURE optimization yields a linear system of equations and the MSE result is empirically *equivalent* to that of the non-linear optimization.



SURE-LET Deconvolution

Principle

- Apply several (typ. 3) Wiener filters with different (fixed) parameters
- Perform undecimated Haar wavelet thresholding
- Optimize the convolution SURE for the LET parameters

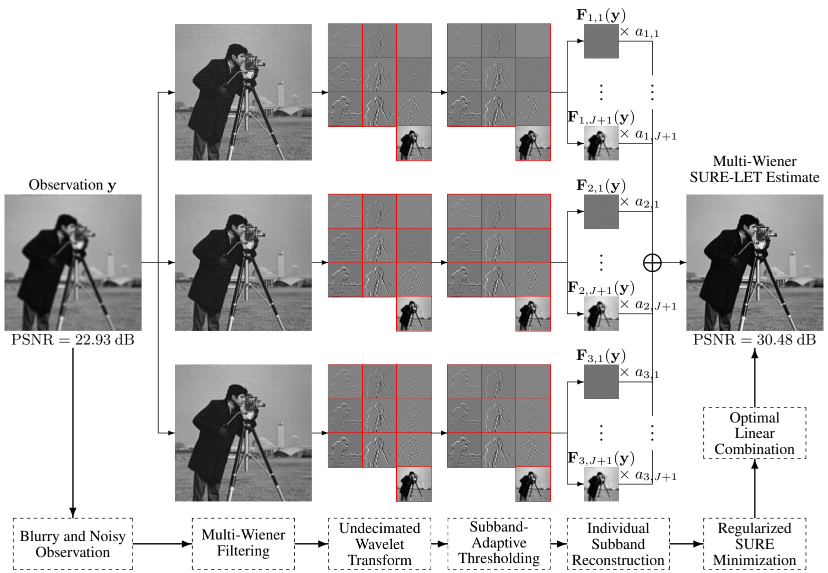
The LET basis corresponding to this algorithm are

$$\mathbf{F}_{k,l}(\mathbf{y}) = \mathbf{R}\boldsymbol{\Theta}_l(\mathbf{D}\mathbf{H}_{\lambda_k}^{-1}\mathbf{y})$$

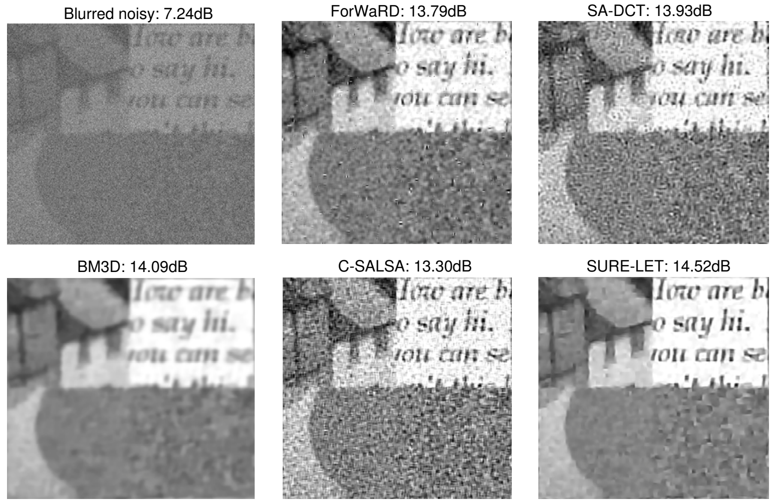
involving two hard-like thresholding rules.

On the whole, we have three times more LET coefficients than for image denoising plus three (low-pass coefficients): 93 for 5 iterations.

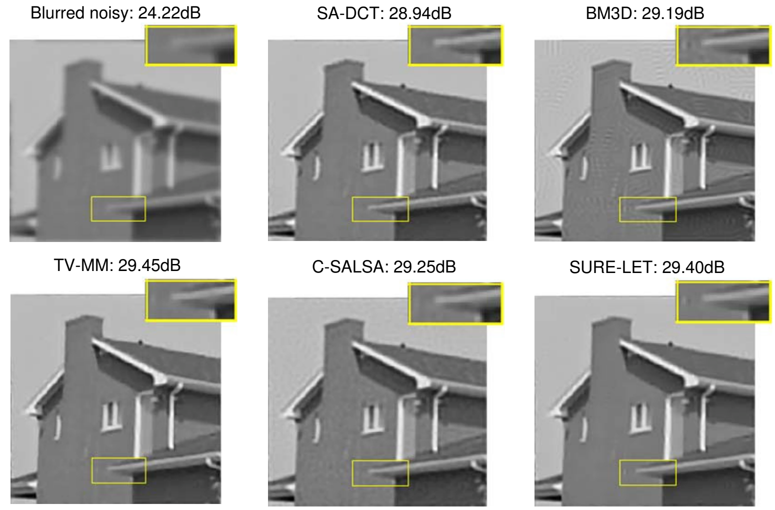
Typical results reach the state-of-the-art, while being *much faster* than high-quality algorithms: 0.7 s for a 256×256 image, 2.8 s for a 512×512 image on a standard PC.



Results: PSNR comparisons



Results: PSNR comparisons



Outline

Sparse restoration

A standard approach to restore images is to regularize the problem with a sparsifying norm

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \underbrace{\|\mathbf{y} - \mathbf{HRw}\|^2 + \lambda\|\mathbf{w}\|_{\ell^1}}_{J(\mathbf{w})}$$

and $\hat{\mathbf{x}} = \mathbf{R}\hat{\mathbf{w}}$.

Several iterative algorithms for solving this problem

- IST: Iterative Shrinkage Algorithm Daubechies et al. 2004
- FISTA Beck et al. 2009
- SALSA Afonso et al. 2010

NOTE: It will be easier to express $\hat{\mathbf{w}} = \mathbf{F}(\mathbf{y})$ in this setting.

i-LET

The idea is to express $\mathbf{F}(\mathbf{y})$ as a LET and to minimize $J(\mathbf{w})$ for the few LET coefficients. However, in order to be able to refine the solution it is necessary to make the basis change with the iteration order i

$$\mathbf{F}^{(i)}(\mathbf{y}) = \sum_{k=1}^{K} a_k \mathbf{F}_k^{(i)}(\mathbf{y})$$

The coefficients a_k can very efficiently be obtained by, e.g., an *iterated reweighted least-squares (IRLS)* algorithm.

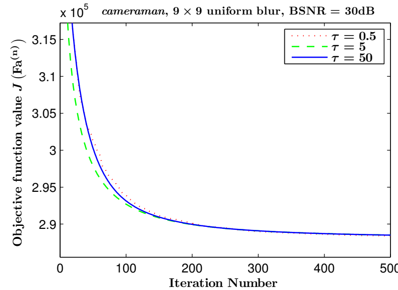
How to ensure that the sequence $\mathbf{F}^{(i)}(\mathbf{y})$ converges to the final solution of the sparse regularization problem when $i \to \infty$?

i-LET

Define $\overline{\nabla}_\tau J(\mathbf{w}) = \mathbf{w} - \mathrm{soft}_{\lambda\tau/2}\big(\mathbf{w} - \tau \mathbf{R}^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}(\mathbf{HRw} - \mathbf{y})\big)$ where $\mathrm{soft}_\lambda(\cdot)$ is the soft-threshold with parameter λ, and τ is any positive number.

Convergence result

If $\mathbf{F}_1^{(i)}(\mathbf{y}) = \hat{\mathbf{w}}^{(i-1)}$ and $\mathbf{F}_2^{(i)}(\mathbf{y}) = \overline{\nabla}_\tau J(\hat{\mathbf{w}}^{(i-1)})$ then the i-LET algorithm converges *unconditionally*.



NOTE: The IST would diverge for $\tau \geq 1$.

i-LET

In practice, in order for the i-LET iterations to converge fastly in all situations (high, or low noise level), we choose the following five LET basis elements

$$\begin{aligned}
\mathbf{F}_0^{(i)}(\mathbf{y}) &= \hat{\mathbf{w}}^{(i-2)}\\
\mathbf{F}_1^{(i)}(\mathbf{y}) &= \hat{\mathbf{w}}^{(i-1)}\\
\mathbf{F}_2^{(i)}(\mathbf{y}) &= \overline{\nabla}_\tau J(\hat{\mathbf{w}}^{(i-1)})\\
\mathbf{F}_3^{(i)}(\mathbf{y}) &= (\mathbf{R}^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}\mathbf{HR} + \tau^{-1}\mathbf{Id})^{-1}\overline{\nabla}_\tau J(\hat{\mathbf{w}}^{(i-1)})\\
\mathbf{F}_4^{(i)}(\mathbf{y}) &= (\mathbf{R}^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}\mathbf{HR} + 10\tau^{-1}\mathbf{Id})^{-1}\overline{\nabla}_\tau J(\hat{\mathbf{w}}^{(i-1)})
\end{aligned}$$

Results: nonredundant wavelets

BSNR		10	15	20	25	30	35	40	10	15	20	25	30	35	40
Method		***cameraman*** (256 × 256) type **1** blur							***bank*** (512 × 512) type **1** blur						
FISTA	iterations / computation time	18 / 0.30	33 / 0.49	64.1 / 0.92	72 / 1.02	112 / 1.58	147 / 2.08	176 / 2.45	21 / 1.67	31 / 2.37	55 / 4.16	89 / 6.68	130 / 9.72	162 / 12.55	176 / 14.23
SALSA	iterations / computation time	60 / 1.26	58 / 1.28	75 / 1.57	40 / 0.86	40 / 0.9	31 / 0.66	20 / 0.46	72 / 7.65	46 / 4.78	51 / 5.31	55 / 5.7	52 / 5.44	38 / 3.99	22 / 2.39
PCD-SESOP-7	iterations / computation time	234.1* / 10.82	151.7 / 7.19	46.1 / 2.19	34.6 / 1.64	43.4 / 2.05	56.3 / 2.64	72.4 / 3.39	741.2* / 165.92	84.8 / 18.97	44 / 9.86	39.1 / 8.75	48.6 / 10.88	61.3 / 13.66	76.5 / 17.09
i-LET	iterations / computation time	**6** / **0.28**	**7** / **0.29**	**10** / **0.40**	**8** / **0.32**	**8.9** / **0.35**	**9.2** / **0.36**	**9.9** / **0.39**	**7** / **1.42**	**6** / **1.23**	**8** / **1.57**	**10** / **1.93**	**10.4** / **2.03**	**10.1** / **1.98**	**10.5** / **2.04**
Method		***cameraman*** (256 × 256) type **2** blur							***bank*** (512 × 512) type **2** blur						
FISTA	iterations / computation time	9 / 0.16	14 / 0.22	18 / 0.27	22 / 0.33	36 / 0.53	52 / 0.74	63 / 0.90	9 / 0.74	13 / 1.05	16 / 1.26	19 / 1.49	25 / 1.93	41 / 3.37	53 / 4.32
SALSA	iterations / computation time	19 / 0.42	15 / 0.34	11 / 0.27	7 / 0.18	7 / 0.18	6 / **0.16**	4 / **0.12**	18 / 1.99	13 / 1.50	9 / 1.09	6 / 0.78	4 / **0.60**	**4** / **0.59**	**3** / **0.48**
PCD-SESOP-7	iterations / computation time	238 / 11.10	74.1 / 3.53	36.2 / 1.73	16.6 / 0.80	10 / 0.48	10.2 / 0.49	11 / 0.53	219.7 / 48.85	72.8 / 16.30	36.5 / 8.20	16.7 / 3.78	10 / 2.26	10.1 / 2.29	10.9 / 2.45
i-LET	iterations / computation time	**3** / **0.14**	**3.5** / **0.16**	**3** / **0.14**	**3** / **0.13**	**4** / **0.17**	**4** / 0.17	**3.4** / 0.15	**3** / **0.65**	**3** / **0.65**	**3** / **0.65**	**3** / **0.65**	**3** / 0.66	**4** / 0.85	**3** / 0.66
Method		***cameraman*** (256 × 256) type **3** blur							***bank*** (512 × 512) type **3** blur						
FISTA	iterations / computation time	5 / **0.09**	6 / **0.10**	7 / **0.11**	52 / 0.74	107.9 / 1.5	172.9 / 2.43	249.8 / 3.48	4 / **0.37**	5 / **0.45**	6 / 0.51	8 / 0.66	62 / 4.66	116 / 9.39	166 / 13.42
SALSA	iterations / computation time	21 / 0.48	9 / 0.23	5 / 0.14	54 / 1.13	93 / 1.98	109 / 2.32	107 / 2.30	18 / 2.02	8 / 0.98	4 / 0.58	**3** / **0.48**	34 / 3.60	52 / 5.38	52 / 5.45
PCD-SESOP-7	iterations / computation time	116.4 / 5.54	77.9 / 3.76	41.4 / 1.94	26.9 / 1.27	35.4 / 1.68	62 / 2.96	90.7 / 4.28	114 / 25.43	71.7 / 16.09	42 / 9.42	24.8 / 5.59	27.8 / 6.27	54.4 / 12.31	77.6 / 17.41
i-LET	iterations / computation time	**3.2** / 0.15	**3.1** / 0.15	**3** / 0.14	**11** / **0.44**	**16.2** / **0.61**	**18.6** / **0.71**	**19.4** / **0.74**	**3** / 0.64	**3.1** / 0.67	**2.1** / **0.49**	3.1 / 0.67	**11.2** / **2.21**	**15.3** / **2.94**	**14.9** / **2.86**

NOTE: Results averaged over 10 trials. Computation times are in seconds.

Results: undecimated wavelets

BSNR		**25**	**30**	**35**	**40**
Method		***cameraman*** type **1** blur			
FISTA	$\frac{\text{iterations}}{\text{computation time}}$	$\frac{640}{60.11}$	$\frac{733}{68.86}$	$\frac{864.2}{81.89}$	$\frac{879.8}{82.8}$
SALSA	$\frac{\text{iterations}}{\text{computation time}}$	$\frac{880}{143.04}$	$\frac{575}{93.52}$	$\frac{410}{66.98}$	$\frac{252}{40.96}$
PCD-SESOP-7	$\frac{\text{iterations}}{\text{computation time}}$	$\frac{636.5}{195.23}$	$\frac{723.6}{222.42}$	$\frac{864}{265.39}$	$\frac{891.4}{273.87}$
i-LET	$\frac{\text{iterations}}{\text{computation time}}$	$\frac{\mathbf{154}}{\mathbf{34.88}}$	$\frac{\mathbf{113.6}}{\mathbf{25.63}}$	$\frac{\mathbf{91.9}}{\mathbf{20.77}}$	$\frac{\mathbf{67.4}}{\mathbf{15.17}}$
Method		***cameraman*** type **2** blur			
FISTA	$\frac{\text{iterations}}{\text{computation time}}$	$\frac{154}{14.47}$	$\frac{111}{10.56}$	$\frac{98}{9.25}$	$\frac{88}{8.38}$
SALSA	$\frac{\text{iterations}}{\text{computation time}}$	$\frac{83}{13.62}$	$\frac{25}{4.24}$	$\frac{10}{1.80}$	$\frac{\mathbf{5}}{\mathbf{0.95}}$
PCD-SESOP-7	$\frac{\text{iterations}}{\text{computation time}}$	$\frac{134.2}{41.51}$	$\frac{98.5}{30.56}$	$\frac{103.6}{32.15}$	$\frac{119.5}{37.1}$
i-LET	$\frac{\text{iterations}}{\text{computation time}}$	$\frac{\mathbf{23.4}}{\mathbf{5.36}}$	$\frac{\mathbf{7.5}}{\mathbf{1.84}}$	$\frac{\mathbf{5.7}}{\mathbf{1.42}}$	$\frac{\mathbf{5}}{1.25}$
Method		***cameraman*** type **3** blur			
FISTA	$\frac{\text{iterations}}{\text{computation time}}$	$\frac{30.2}{2.87}$	$\frac{58}{5.53}$	$\frac{145}{13.67}$	$\frac{208.7}{19.62}$
SALSA	$\frac{\text{iterations}}{\text{computation time}}$	$\frac{11}{1.93}$	$\frac{15}{2.6}$	$\frac{42}{7}$	$\frac{45}{7.43}$
PCD-SESOP-7	$\frac{\text{iterations}}{\text{computation time}}$	$\frac{69.7}{21.63}$	$\frac{98.8}{30.63}$	$\frac{159.9}{49.56}$	$\frac{240.4}{74.45}$
i-LET	$\frac{\text{iterations}}{\text{computation time}}$	$\frac{\mathbf{7.5}}{\mathbf{1.83}}$	$\frac{\mathbf{8.4}}{\mathbf{2.05}}$	$\frac{\mathbf{14.4}}{\mathbf{3.38}}$	$\frac{\mathbf{16.4}}{\mathbf{3.84}}$

Conclusion

Thanks to

- **Florian Luisier** : SURE-LET denoising (and PURE-LET etc.)
- **Feng Xue** : SURE-LET deconvolution
- **Hanjie Pan** : *i*-LET restoration

Main papers

Blu T. and Luisier F., "The SURE-LET Approach to Image Denoising", *IEEE Transactions on Image Processing*, Vol. 16 (11), pp. 2778-2786, November 2007.

Xue F., Luisier F. and Blu T., "Multi-Wiener SURE-LET Deconvolution", *IEEE Transactions on Image Processing*, 22(5), pp. 1954–1968, May 2013.

Pan H. and Blu T., "An iterative linear expansion of thresholds for ℓ_1-based image restoration.", *IEEE Transactions on Image Processing*, 22(9), pp. 3715–3728, September 2013.

Demos & Software

http://www.ee.cuhk.edu.hk/~tblu/demos/
http://scholar.harvard.edu/fluisier/software/image-denoising