Fast external denoising using pre-learned transformations

Shibin Parameswaran, Enming Luo, Charles-Alban Deledalle and Truong Nguyen

NTIRE 2017

(in conjunction with **CVPR 2017**)

Image denoising

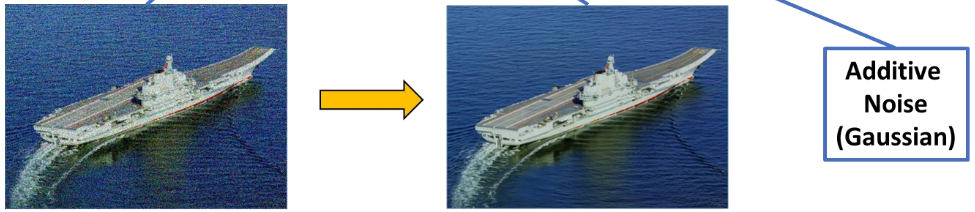
- **Goal:** Estimate the underlying clean image from the observed noisy image
- **Patch-based Image denoising:** Denoise an image patch-by-patch by leveraging information contained in similar patches
  - Patch to be denoised = Query patch
  - Similar patches = Reference patches

$$\hat{\boldsymbol{p}} = \boldsymbol{\Phi}(\boldsymbol{q}; \boldsymbol{p}_1, \boldsymbol{p}_2, \dots \boldsymbol{p}_k)$$

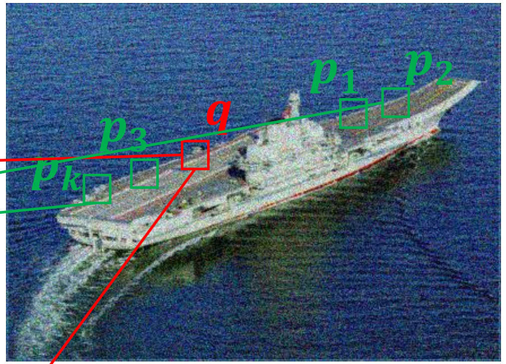
$\boldsymbol{\Phi}$: Some linear or non-linear function

$\hat{\boldsymbol{p}}$: Estimate of clean patch $\boldsymbol{p}$

$$\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{\eta}$$



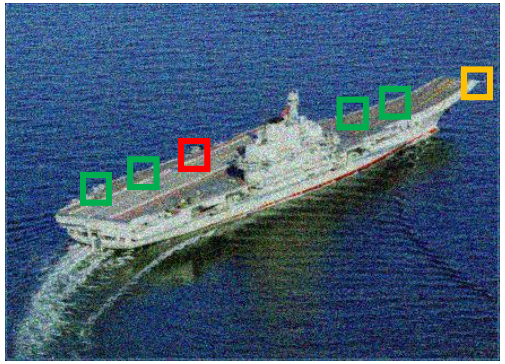
Noisy image (no blur)   Clean image



$$\boldsymbol{q} = \boldsymbol{p} + \boldsymbol{\eta} \quad \textbf{where} \quad \boldsymbol{\eta} \in N(\boldsymbol{0}, \boldsymbol{\sigma}^2 \boldsymbol{I}_n)$$

Types of patch-based denoising

Internal denoising
From the noisy image



e.g. NLM, BM3D, LPG-PCA
Drawbacks:
- Limited performance [Chatterjee et al. `09, `11]
- Rare patches

External denoising
From an external database

e.g. EPLL, eNLM, eBM3D, eLPG-PCA
Drawbacks:
- Computational complexity
- Marginal improvement

Targeted denoising
From a domain specific database

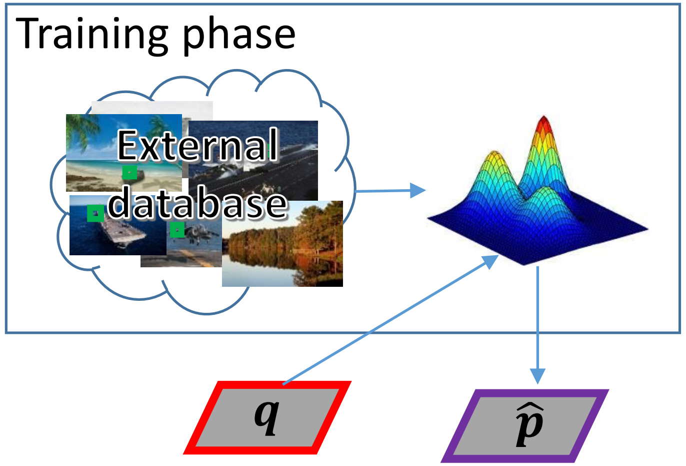
e.g. TID, tBM3D, tLPG-PCA, tEPLL
- Smaller database can be used
- Improved performance

Drawbacks:
- Slower than internal methods
- Database selection

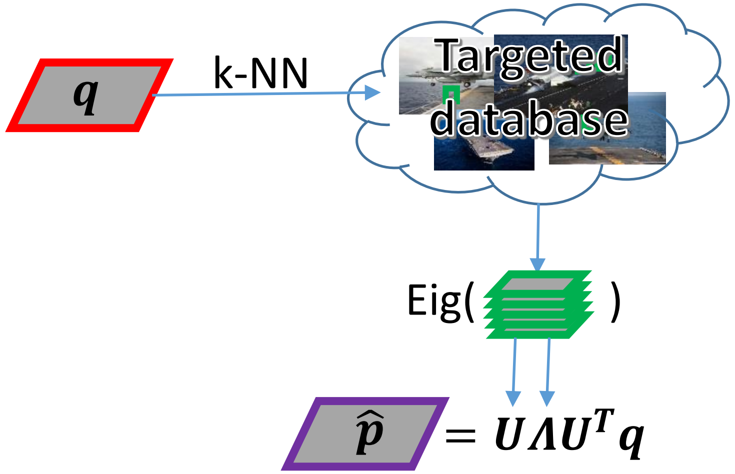
: Noisy query patch : Reference patches : Rare patch

Related methods

- Expected patch log-likelihood (EPLL) [Zoran et al. 2011]
  - Learns patch priors using Gaussian Mixture Models
  - One of the most efficient external denoising algorithms
  - Slow to converge to high quality solutions
    - Many iterations involving Mahalanobis distance calculations
    - Heavily overlapped patches



- Targeted Image Denoising (TID) [Luo et al. 2015]
  - Powerful patch-specific denoising filters
  - Converges to high quality solution in 2 iterations
  - Computationally expensive
    - Per-patch filter design



Existing external denoising methods are too slow

Need faster methods

Design of a fast denoising algorithm

- Whole image denoising formulation

$$\min_{\boldsymbol{x}, \{\boldsymbol{z}_i\}} \underbrace{\frac{1}{2\sigma^2}\|\boldsymbol{x}-\boldsymbol{y}\|_2^2}_{\text{Data fidelity term}} + \frac{\beta}{2}\sum_{i=1}^{N}\underbrace{[\|\boldsymbol{P}_i\boldsymbol{x}-\boldsymbol{A}\boldsymbol{z}_i\|_2^2+\lambda\|\boldsymbol{z}_i\|_2^2]}_{\text{Patch reconstruction term}}$$

Notations:

- $\boldsymbol{x}$: Clean image
- $\boldsymbol{y}$: Noisy image (given)
- σ^2 : Noise variance (given)
- $\boldsymbol{P}_i$: Patch extractor ($\boldsymbol{P}_i\boldsymbol{x} \in \mathbb{R}^d$)
- $\boldsymbol{A}$: Dictionary of patches
- $\{\boldsymbol{z}_i\}$: Coefficient vectors
- β, λ : Optimization parameters

- Solve by alternating between optimal $\boldsymbol{x}$ and $\{\boldsymbol{z}_i\}$
  - Fix $\{\boldsymbol{z}_i\}$:

$$\boxed{\hat{\boldsymbol{x}} = \left(\frac{1}{\sigma^2}\boldsymbol{I}_N + \beta\sum_{i=1}^{N}\boldsymbol{P}_i^T\boldsymbol{P}_i\right)^{-1}\left(\frac{1}{\sigma^2}\boldsymbol{y} + \beta\sum_{i=1}^{N}\boldsymbol{P}_i^T\boldsymbol{A}\boldsymbol{z}_i\right)}$$

  - Fix $\boldsymbol{x}$:

$$\boxed{\boldsymbol{A}\hat{\boldsymbol{z}}_i = \boldsymbol{U}\frac{\boldsymbol{S}}{\boldsymbol{S}+\lambda\boldsymbol{I}_k}\boldsymbol{U}^T\boldsymbol{P}_i\boldsymbol{x} \quad \text{where} \quad \mathrm{EIG}(\boldsymbol{A}\boldsymbol{A}^T) = [\boldsymbol{U},\boldsymbol{S}]}$$

Choosing $\boldsymbol{A}$ matrix

- The entire patch database
  - Bad choice with ℓ_2 norm

$$\|\boldsymbol{P}_i \boldsymbol{x} - \boldsymbol{A}\boldsymbol{z}_i\|_2^2 + \lambda \|\boldsymbol{z}_i\|_2^2$$

- Dictionaries tailored to each $\boldsymbol{P}_i \boldsymbol{x}$
  - Similar to TID algorithm [Luo et al. 2015]
  - Inefficient

- Identify and tailor the dictionaries to a set of anchor patches
  - Anchor patches $\{\boldsymbol{a}_1, \dots, \boldsymbol{a}_k\}$: Representatives of patch database
  - Build $\{\boldsymbol{A}_1, \dots, \boldsymbol{A}_k\}$ using m nearest neighbors of $\{\boldsymbol{a}_1, \dots, \boldsymbol{a}_k\}$ as:

$$\boldsymbol{A}_k = \boldsymbol{A}\boldsymbol{W}_k^{0.5} \quad \text{where} \quad \boldsymbol{W}_k = \frac{\boldsymbol{1}}{\boldsymbol{\alpha}} \,\mathbf{diag}\,[\boldsymbol{w}_1, \dots, \boldsymbol{w}_m]$$

$$\text{and} \quad \boldsymbol{w}_j = \mathbf{exp}\left(-\frac{\|\boldsymbol{a}_k - \boldsymbol{p}_j\|^2}{2\boldsymbol{h}^2}\right)$$

Fast external denoising (FED) algorithm

Database of Patches

m-Nearest neighbors

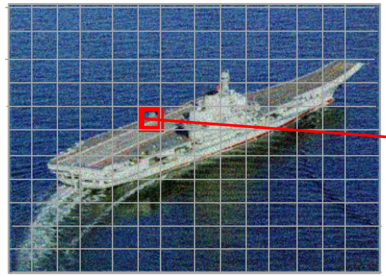
k-means clustering
Or
Dictionary learning

Anchor Dictionaries
$\{\boldsymbol{A}_1\boldsymbol{W}_1^{0.5}, \dots, \boldsymbol{A}_k\boldsymbol{W}_k^{0.5}\}$

Anchor Patches
$\{\boldsymbol{a}_1, \dots, \boldsymbol{a}_k\}$

$$[\boldsymbol{U}_1, \boldsymbol{S}_1^{0.5}] = \mathrm{svd}(\boldsymbol{A}_1\boldsymbol{W}_1^{0.5})$$
$$\dots$$
$$[\boldsymbol{U}_k, \boldsymbol{S}_k^{0.5}] = \mathrm{svd}(\boldsymbol{A}_k\boldsymbol{W}_k^{0.5})$$

Training phase (offline)



$\boldsymbol{q}$

$$\underset{t}{\mathrm{argmin}} \|\boldsymbol{q} - \boldsymbol{a}_t\|$$

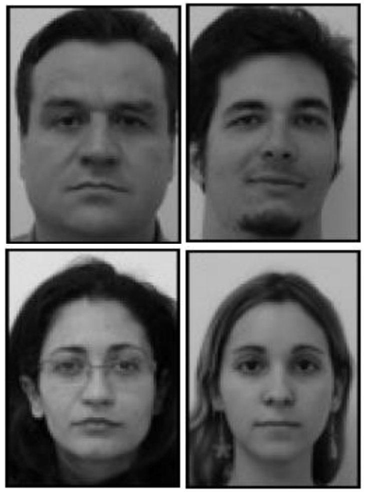
$$\boldsymbol{\Lambda} = [\boldsymbol{S}_t + \sigma^2\boldsymbol{I}]^{-1}[\boldsymbol{S}_t]$$
$$\boldsymbol{U} = \boldsymbol{U}_t$$
$$= \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T\boldsymbol{q}$$

$$\hat{\boldsymbol{x}} = \left(\frac{1}{\sigma^2}\boldsymbol{I}_N + \beta\sum_{i=1}^{N}\boldsymbol{P}_i^T\boldsymbol{P}_i\right)^{-1}\left(\frac{1}{\sigma^2}\boldsymbol{y} + \beta\sum_{i=1}^{N}\boldsymbol{P}_i^T\boldsymbol{A}\boldsymbol{z}_i\right)$$

Datasets

- Face image dataset
  - 100 images of distinct individuals
    - Test: 10 images
    - Validation: 10 images
    - Database: 80 images
    - Size: 90x65

Query image



Database

- License plate dataset
  - 110 images of license plates cropped from Caltech Cars dataset
    - Test: 10 images
    - Validation: 10 images
    - Database: 90 images
    - Avg. size: 44x92

Query image

Database

Results on face dataset

	$\sigma \times 255$	**BM3D** 2 iterations $N_s = [6, 4]$	**EPLL** 5 iterations $N_s = [1]$	**tar-EPLL** 5 iterations $N_s = [1]$	**tar-EPLL3** 3 iterations $N_s = [4, 2, 1]$	**TID** 3 iterations $N_s = [4, 2, 1]$	**FED** 3 iterations $N_s = [4, 2, 1]$
PSNR:	20	31.37	31.40	31.99	31.15	32.26	32.11
	40	27.63	27.86	28.32	27.09	28.51	28.20
	60	25.70	25.68	26.08	24.67	26.09	25.60
	80	24.37	24.29	24.56	23.00	24.27	23.73
SSIM:	20	0.9054	0.9048	0.9160	0.8860	0.9201	0.9164
	40	0.8176	0.8136	0.8283	0.7554	0.8273	0.8094
	60	0.7576	0.7477	0.7612	0.6381	0.7524	0.7193
	80	0.6973	0.6859	0.6964	0.5483	0.6741	0.6288
Time (seconds):	-	0.05	2.74	2.73	0.87	878.20	0.71
Speed up:	-	x0.07	x3.84	x3.83	x1.23	x1236.90	x1.00

Visual comparison of denoised face images

(a) Original

(b) Noisy ($\sigma = \frac{30}{255}$)

(c) BM3D (28.96, 0.8572)

(d) EPLL (28.82, 0.8518)

(e) tar-EPLL (29.18, 0.8652)

(f) tar-EPLL3 (28.42, 0.8311)

(g) TID (29.17, 0.8608)

(h) FED (28.98, 0.8533)

Visual results of license plate dataset

(a) Original

(b) Noisy ($\sigma = \frac{20}{255}$) | (c) BM3D (25.77, 0.9497) | (d) EPLL (25.51, 0.9508) | (e) tar-EPLL (26.84, 0.9624) | (f) tar-EPLL3 (26.47, 0.9586) | (g) TID (25.37, 0.9525) | (h) FED (26.47, 0.9541)

(i) Noisy ($\sigma = \frac{50}{255}$) | (j) BM3D (19.48, 0.7977) | (k) EPLL (19.64, 0.8295) | (l) tar-EPLL (21.36, 0.8801) | (m) tar-EPLL3 (20.99, 0.8734) | (n) TID (22.72, 0.9101) | (o) FED (21.99, 0.8833)

(p) Noisy ($\sigma = \frac{80}{255}$) | (q) BM3D (17.07, 0.6766) | (r) EPLL (16.97, 0.6862) | (s) tar-EPLL (18.72, 0.7885) | (t) tar-EPLL3 (18.26, 0.7826) | (u) TID (19.43, 0.7840) | (v) FED (19.81, 0.8183)

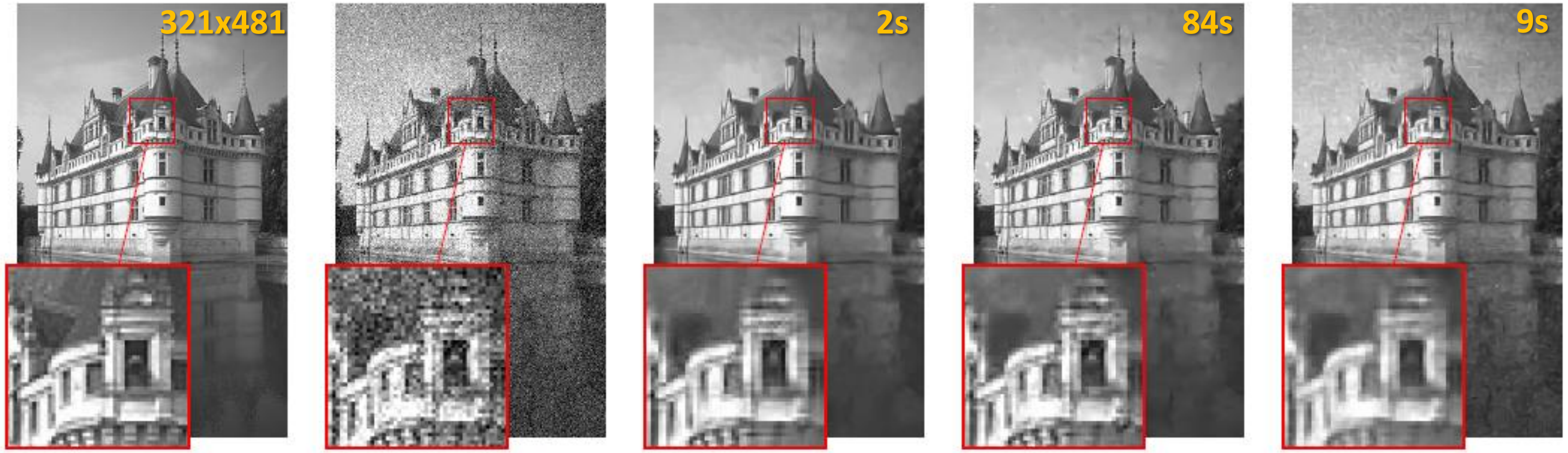
Speed up: x0.07 | x3.90 | x3.91 | x1.02 | x693 | x1.00

Conclusions

- Introduced a fast external denoising algorithm
  - Orders of magnitude faster than TID
  - Faster and better than EPLL with targeted database

- Speed can be further improved using approximate nearest neighbors

- Limitations
  - Database mismatch



(a) Original (b) Noisy ($\sigma = \frac{30}{255}$) (c) BM3D (28.59, 0.8300) (d) EPLL (28.54, 0.8262) (e) Generic FED (27.25, 0.7323)

Thank you