Fast external denoising using pre-learned transformations

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

NTIRE 2017
(in conjunction with CVPR 2017)
**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

\[
\hat{p} = \Phi(q; p_1, p_2, \ldots, p_k)
\]

\(\Phi\): Some linear or non-linear function

\(\hat{p}\): Estimate of clean patch \(p\)

\[q = p + \eta\quad \text{where} \quad \eta \in N(0, \sigma^2 I_n)\]
Types of patch-based denoising

Internal denoising
From the noisy image
- e.g. NLM, BM3D, LPG-PCA
- Drawbacks:
  - Limited performance
  - 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

\[
\text{Training phase}
\]

\[
\text{Eig}(\quad ) = U\Lambda U^T q
\]
Existing external denoising methods are too slow

*Need faster methods*
Design of a fast denoising algorithm

- Whole image denoising formulation

\[
\min_{x,\{z_i\}} \frac{1}{2\sigma^2} \|x - y\|_2^2 + \frac{\beta}{2} \sum_{i=1}^{N} [\|P_ix - Az_i\|_2^2 + \lambda \|z_i\|_2^2]
\]

**Data fidelity term**

**Patch reconstruction term**

- Solve by alternating between optimal \(x\) and \(\{z_i\}\)
  - Fix \(\{z_i\}\):
    
    \[
    \hat{x} = \left( \frac{1}{\sigma^2} I_N + \beta \sum_{i=1}^{N} P_i^T P_i \right)^{-1} \left( \frac{1}{\sigma^2} y + \beta \sum_{i=1}^{N} P_i^T A z_i \right)
    \]
  - Fix \(x\):

\[
A\hat{z}_i = US_{S+\lambda I_k}U^TP_i x \quad \text{where} \quad \text{EIG}(AA^T) = [U, S]
\]

**Notations:**
- \(x\): Clean image
- \(y\): Noisy image (given)
- \(\sigma^2\): Noise variance (given)
- \(P_i\): Patch extractor \((P_i x \in \mathbb{R}^d)\)
- \(A\): Dictionary of patches
- \(\{z_i\}\): Coefficient vectors
- \(\beta, \lambda\): Optimization parameters
Choosing $A$ matrix

- The entire patch database
  - Bad choice with $\ell_2$ norm

- Dictionaries tailored to each $P_i x$
  - Similar to TID algorithm [Luo et al. 2015]
  - Inefficient

Identify and tailor the dictionaries to a set of anchor patches

- Anchor patches $\{a_1, ..., a_k\}$: Representatives of patch database
- Build $\{A_1, ..., A_k\}$ using $m$ nearest neighbors of $\{a_1, ..., a_k\}$ as:

$$A_k = AW_k^{0.5} \quad \text{where} \quad W_k = \frac{1}{\alpha} \text{diag}[w_1, ..., w_m]$$

and

$$w_j = \exp \left( -\frac{\|a_k-p_j\|^2}{2h^2} \right)$$

$$\|P_i x - Az_i\|_2^2 + \lambda \|z_i\|_2^2$$
**Fast external denoising (FED) algorithm**

**Training phase (offline)**

- **Database of Patches**
- **m-Nearest neighbors**
- **k-means clustering** or **Dictionary learning**
- **Anchor Patches** \( \{a_1, \ldots, a_k\} \)
- **Anchor Dictionaries** \( \{A_1 W_1^{0.5}, \ldots, A_k W_k^{0.5}\} \)

\[
[U_1, S_1^{0.5}] = \text{svd}(A_1 W_1^{0.5}) \\
\ldots \\
[U_k, S_k^{0.5}] = \text{svd}(A_k W_k^{0.5})
\]

**argmin**\(_t\)||q - a_t||

\[
\Lambda = [S_t + \sigma^2 I]^{-1} [S_t] \\
U = U_t \\
= U\Lambda U^T q
\]

\[
\hat{x} = \left( \frac{1}{\sigma^2} I_N + \beta \sum_{i=1}^{N} P_i^T P_i \right)^{-1} \left( \frac{1}{\sigma^2} y + \beta \sum_{i=1}^{N} P_i^T A z_i \right)
\]
Datasets

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

• 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
## Results on face dataset

<table>
<thead>
<tr>
<th>$\sigma \times 255$</th>
<th>BM3D 2 iterations, $N_s = [6, 4]$</th>
<th>EPLL 5 iterations, $N_s = [1]$</th>
<th>tar-EPLL 5 iterations, $N_s = [1]$</th>
<th>tar-EPLL3 3 iterations, $N_s = [4, 2, 1]$</th>
<th>TID 3 iterations, $N_s = [4, 2, 1]$</th>
<th>FED 3 iterations, $N_s = [4, 2, 1]$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PSNR:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>31.37</td>
<td>31.40</td>
<td>31.99</td>
<td>31.15</td>
<td>32.26</td>
<td>32.11</td>
</tr>
<tr>
<td>40</td>
<td>27.63</td>
<td>27.86</td>
<td>28.32</td>
<td>27.09</td>
<td>28.51</td>
<td>28.20</td>
</tr>
<tr>
<td>60</td>
<td>25.70</td>
<td>25.68</td>
<td>26.08</td>
<td>24.67</td>
<td>26.09</td>
<td>25.60</td>
</tr>
<tr>
<td>80</td>
<td>24.37</td>
<td>24.29</td>
<td>24.56</td>
<td>23.00</td>
<td>24.27</td>
<td>23.73</td>
</tr>
<tr>
<td><strong>SSIM:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>0.9054</td>
<td>0.9048</td>
<td>0.9160</td>
<td>0.8860</td>
<td>0.9201</td>
<td>0.9164</td>
</tr>
<tr>
<td>40</td>
<td>0.8176</td>
<td>0.8136</td>
<td>0.8283</td>
<td>0.7554</td>
<td>0.8273</td>
<td>0.8094</td>
</tr>
<tr>
<td>60</td>
<td>0.7576</td>
<td>0.7477</td>
<td>0.7612</td>
<td>0.6381</td>
<td>0.7524</td>
<td>0.7193</td>
</tr>
<tr>
<td>80</td>
<td>0.6973</td>
<td>0.6859</td>
<td>0.6964</td>
<td>0.5483</td>
<td>0.6741</td>
<td>0.6288</td>
</tr>
<tr>
<td><strong>Time (seconds):</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>0.05</td>
<td>2.74</td>
<td>2.73</td>
<td>0.87</td>
<td>878.20</td>
<td>0.71</td>
</tr>
<tr>
<td><strong>Speed up:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>x0.07</td>
<td>x3.84</td>
<td>x3.83</td>
<td>x1.23</td>
<td>x1236.90</td>
<td>x1.00</td>
</tr>
</tbody>
</table>
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:  
\[ \times 0.07 \] \quad \times 3.90 \quad \times 3.91 \quad \times 1.02 \quad \times 693 \quad \times 1.00 \]
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
Thank you