Globalized BM3D using Fast Eigenvalue Filtering

Tokyo University of Agriculture and Technology

Koki Suwabe, Masaki Onuki, Yuki Iizuka and Yuchi Tanaka
Outline

- Image denoising
- Previous method
  - Improving method by eigenvalue filtering for denoising
  - Eigenvalue filtering using Chebyshev polynomial approximation
  - BM3D
- Proposed method
- Evaluation
- Conclusion
Image Denoising

Image denoising: estimating the true image from the observed image

Observation model

\[ z = y + n \]

- \( z \in \mathbb{R}^N \): Observed image
- \( y \in \mathbb{R}^N \): True image
- \( n \in \mathbb{R}^N \): Noise signal
- \( N \): The number of pixels

\[ n \sim \mathcal{N}(0, \sigma^2) \]
Filter Matrix and Its Decomposition

• **Denoising methods can be expressed as** $W \in \mathbb{R}^{N \times N}$
  
  Ex.) Gaussian Filter, Bilateral Filter, Non-local means

  **Restored image**
  
  $\hat{y} = Wz$

• **The filter matrix is decomposed as**
  
  $W = VSV^{-1}$

  *Eigenvalue matrix*
  
  $S = \text{diag} [\lambda_1 \cdots \lambda_N]$

  *Eigenvector matrix*
  
  $V = [v_1 \cdots v_N]$

**Eigenvalue**

- Large: $\lambda_2, \lambda_3, \lambda_4 \cdots \lambda_i \cdots \lambda_N$
- Small

**Elements of eigenvector**

- Oscillated slowly: $v_2, v_3, v_4 \cdots v_i \cdots v_N$
- Oscillated rapidly
Eigenvalue Filtering for the Filter Matrix

Input image

Eigenvalues

\[ \lambda_2, \lambda_3, \lambda_4, \lambda_5, \ldots, \lambda_N \]

Eigenvectors

\[ v_2, v_3, v_4, v_5, \ldots, v_N \]

Output image

Small eigenvalues are truncated

Eigenvalue filtering

The restored image becomes smoother
Eigenvalue Filtering for the Filter Matrix

Eigenvalue filtering

\[ \mathcal{H}(W) = V \text{diag}(h(\lambda_1), \cdots, h(\lambda_i), \cdots, h(\lambda_N)) V^{-1} \]

\( h(\cdot) \): Arbitrary filter kernel

- The smoothing strength is controlled according to the filter kernel
**Parameter Selection of Eigenvalue Filtering**

I. Perform eigenvalue filtering using various filter kernels controlled by the parameter $\mathcal{H}_1(W)$, $\mathcal{H}_2(W)$, ..., $\mathcal{H}_P(W)$.

II. Obtain restored images using each eigenvalue-filtered matrices $\hat{y}_1$, $\hat{y}_2$, ..., $\hat{y}_P$.

III. Estimate MSEs of each restored image.

IV. Select an optimal output (an image having minimum MSE).
Improving Method by Eigenvalue Filtering

Filter kernel

$$h_p(\lambda) = \text{sign}(\lambda)|\lambda|^p$$

MSE transition according to the parameter $p$

Optimal smoothing strength

$p = 0.8$
Approximation of Filter Kernels by CPA

- Eigendecomposition takes much computational cost

Eigenvale filtering by **Chebyshev polynomial approximation** (CPA) [1]

**CPA for scalar function**

\[
h(y) = \frac{1}{2}c_0 + \sum_{k=1}^{\infty} c_k T_k(y)
\]

Chebyshev polynomial

\[T_k(y) = \cos(k \arccos(y))\]

Chebyshev coefficient

\[
c_k = \frac{2}{\pi} \int_{-1}^{1} \frac{T_k(y)h(y)}{\sqrt{1-y^2}} dy = \frac{2}{\pi} \int_{0}^{\pi} \cos(k\theta)h(\cos \theta) d\theta
\]

Chebyshev polynomials are obtained by recurrence relation

**Recurrence relation**

\[T_k(y) = 2yT_{k-1}(y) - T_{k-2}(y)\]

**Initial conditions**

\[T_0(y) = 1, \quad T_1(y) = y\]

Eigenvalue Filtering by CPA

- Eigenvalue filtering can be realized without eigendecomposition

\[
\hat{y} = \mathcal{H}(W)z = \left( \frac{1}{2} c_0 I + \sum_{k=1}^{d} c_k T_k(W) \right) z
\]

CPA for a filter matrix

\[
\mathcal{H}(W) = \frac{1}{2} c_0 I + \sum_{k=1}^{\infty} c_k T_k(W)
\]

Chebyshev polynomial

\[
T_k(W) = V \text{diag}(\cos k\theta_1, \ldots, \cos k\theta_i, \ldots, \cos k\theta_N) V^{-1}
\]

Chebyshev coefficient

\[
c_k = \frac{2}{\pi} \int_{0}^{\pi} \cos (k\theta) h(\cos \theta) \, d\theta
\]

\(h(\cdot)\) : Arbitrary function

Recurrence relation

\[
T_k(W) = 2W T_{k-1}(W) - T_{k-2}(W)
\]

Initial conditions

\[
T_0(W) = I, \quad T_1(W) = W
\]
Purpose of Proposed Method

**Purpose** Applying eigenvalue filtering to state-of-the-art methods (i.e.) BM3D [2]


BM3D matrix

Next topic

- BM3D algorithm and its matrix representation
- Problem of matrix construction
- Solution (Proposed method)
BM3D Algorithm

Block Matching and 3D Filtering (BM3D):

- Redundant filtering using similarity among blocks

- Grouping
- 3D transform
- Spatial domain
- Frequency domain

- Aggregation

- High denoising performance
- Fast processing

Inverse 3D transform
BM3D is expressed as a filter matrix

\[ \hat{y} = \mathcal{F}_{\text{BM3D}}(z) = \Psi \Gamma \Phi z = Az \]

Construction of \( \Phi \) and \( \Psi \) needs much computational cost

Ex.) Image with 1024×1024 pixels

BM3D matrix

\[ A = \begin{bmatrix} \Psi & \Gamma \Phi \end{bmatrix} \]

Synthesis

Shrinkage

Analysis

Inverse 3D transform

Grouping

3D transform

Aggregation

Filtering

It is hard to construct the BM3D matrix
Proposed Method

Restored image using eigenvalue filtering by CPA

\[ \hat{y}_p = \mathcal{H}_p(A)z = \left( \frac{1}{2}c_0 I + \sum_{k=1}^{d} c_k \mathcal{T}_k(A) \right) z = \frac{1}{2} c_0 z + \sum_{k=1}^{d} c_k \mathcal{T}_k(A)z \]

Previous method

\[ \mathcal{T}_k(A)z = 2A\mathcal{T}_{k-1}(A)z - \mathcal{T}_{k-2}(A)z \]
\[ \mathcal{T}_0(A)z = z , \quad \mathcal{T}_1(A)z = Az \]

Replacing

\[ \mathcal{B}_k(z) = \mathcal{T}_k(A)z \quad \mathcal{F}_{BM3D}(z) = Az \]

Proposed method

\[ \mathcal{T}_k(A)z \simeq \mathcal{B}_k(z) = 2\mathcal{F}_{BM3D}(\mathcal{B}_{k-1}(z)) - \mathcal{B}_{k-2}(z) \]
\[ \mathcal{T}_0(A)z = \mathcal{B}_0(z) = z , \quad \mathcal{T}_1(A)z = \mathcal{B}_1(z) = \mathcal{F}_{BM3D}(z) \]

\[ \rightarrow \text{Matrix construction is not required} \]
Fast Eigenvalue Filtering

Restored image using CPA

\[
\hat{y} = \mathcal{H}(W)z = \frac{1}{2}c_0 z + \sum_{k=1}^{d} c_k T_k(W)z
\]

Eigenvalue filtering is realized only by using BM3D operators and Chebyshev coefficients
Problem: Input-dependency of the BM3D

**CPA:** \( \text{A must be fixed regardless of the degree of polynomials} \)

\[
\mathcal{T}_k(A)z = 2A\mathcal{T}_{k-1}(A)z - \mathcal{T}_{k-2}(A)z
\]

**Needs verification**

**BM3D:** \( \mathcal{F}_{BM3D} \text{ is adaptive to the input image} \)

\[
\mathcal{T}_k(A)z = 2\mathcal{F}_{BM3D}(\mathcal{T}_{k-1}(A)z) - \mathcal{T}_{k-2}(A)z
\]

Due to Block matching and filter coefficients
Verification Experiment

- Verify eigenvalue distributions according to iteration numbers

Test sub-image $Z$

Mandrill ($32 \times 32$)

Eigenvalue distributions could be assumed to be consistent regardless of the iteration number.

$A_n$: $n$th matrix of BM3D given by $\mathcal{T}_{n-1}(A)z$
Summary of Proposed Method

Eigenvalue filtering by CPA

$$\mathcal{H}(A) = \frac{1}{2} c_0 I + \sum_{k=1}^{d} c_k \mathcal{T}_k(A)$$

Previous method

$$\mathcal{T}_k(A) = 2A \mathcal{T}_{k-1}(A) - \mathcal{T}_{k-2}(A)$$

$$\mathcal{T}_0(A) = I$$

$$\mathcal{T}_1(A) = A$$

Proposed method

$$\mathcal{B}_k(z) = 2\mathcal{F}_{BM3D}(\mathcal{B}_{k-1}(z)) - \mathcal{B}_{k-2}(z)$$

$$\mathcal{B}_0(z) = z$$

$$\mathcal{B}_1(z) = \mathcal{F}_{BM3D}(z)$$
## Experiment

### Denoising performance assessment

**Comparison**  
BM3D, Global Image Denoising (GLIDE) \[3\]  
GLIDE: Improving method by eigenvalue filtering

**Test images**  
*Bridge*, *Mandrill*, *Goldhill*, *Building*

**Noise strength**  
\( \sigma \in \{10, 20, 30, 40, 50\} \)

**Measure**  
PSNR, SSIM

**Conditions**  
Intel Xeon E5-2690 2.9GHz CPU  
62.9 GB RAM  
12 core parallel computing

---

**Global Image Denoising (GLIDE)**

estimate eigenvalue/eigenvector from a portion of a pre-filtered image

**Pre-filtering**

**Sampling**

<table>
<thead>
<tr>
<th>Advantage</th>
<th>Fast processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disadvantage</td>
<td>Eigenvalue filtering can not be performed exactly</td>
</tr>
</tbody>
</table>

Experiment

Denoising performance assessment

Comparison  BM3D, Global Image Denoising (GLIDE) \(^3\)
GLIDE: Improving method by eigenvalue filtering

Test images  Bridge, Mandrill, Goldhill, Building

Noise strength  \(\sigma \in \{10, 20, 30, 40, 50\}\)

Measure  PSNR, SSIM

Conditions
Intel Xeon E5-2690 2.9GHz CPU
62.9 GB RAM
12 core parallel computing

# Performance Comparison

<table>
<thead>
<tr>
<th>$\sigma$</th>
<th>Method</th>
<th><strong>Bridge</strong></th>
<th><strong>Mandrill</strong></th>
<th><strong>Goldhill</strong></th>
<th><strong>Building</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>BM3D</td>
<td>29.84 / 0.911</td>
<td>30.56 / 0.905</td>
<td>31.80 / 0.880</td>
<td>33.16 / 0.939</td>
</tr>
<tr>
<td></td>
<td>GLIDE</td>
<td>29.81 / 0.913</td>
<td>30.54 / 0.904</td>
<td>31.72 / 0.881</td>
<td>32.91 / 0.938</td>
</tr>
<tr>
<td></td>
<td>Proposed</td>
<td>29.86 / 0.913</td>
<td>30.57 / 0.906</td>
<td>31.86 / 0.884</td>
<td>33.16 / 0.939</td>
</tr>
<tr>
<td>20</td>
<td>BM3D</td>
<td>25.46 / 0.765</td>
<td>26.39 / 0.773</td>
<td>28.50 / 0.775</td>
<td>29.35 / 0.862</td>
</tr>
<tr>
<td></td>
<td>GLIDE</td>
<td>25.62 / 0.784</td>
<td>26.55 / 0.788</td>
<td>28.57 / 0.785</td>
<td>29.30 / 0.865</td>
</tr>
<tr>
<td></td>
<td>Proposed</td>
<td>24.66 / 0.789</td>
<td>26.56 / 0.791</td>
<td>28.59 / 0.784</td>
<td>29.40 / 0.866</td>
</tr>
<tr>
<td>30</td>
<td>BM3D</td>
<td>23.55 / 0.647</td>
<td>24.33 / 0.651</td>
<td>26.91 / 0.706</td>
<td>27.32 / 0.790</td>
</tr>
<tr>
<td></td>
<td>GLIDE</td>
<td>23.68 / 0.678</td>
<td>24.57 / 0.686</td>
<td>26.71 / 0.711</td>
<td>27.26 / 0.792</td>
</tr>
<tr>
<td></td>
<td>Proposed</td>
<td>23.73 / 0.679</td>
<td>24.58 / 0.689</td>
<td>26.96 / 0.714</td>
<td>27.37 / 0.794</td>
</tr>
<tr>
<td>40</td>
<td>BM3D</td>
<td>22.51 / 0.572</td>
<td>23.10 / 0.558</td>
<td>25.84 / 0.654</td>
<td>25.89 / 0.722</td>
</tr>
<tr>
<td></td>
<td>GLIDE</td>
<td>22.43 / 0.584</td>
<td><strong>23.23</strong> / 0.573</td>
<td>25.70 / 0.640</td>
<td>25.87 / 0.729</td>
</tr>
<tr>
<td></td>
<td>Proposed</td>
<td><strong>22.55</strong> / 0.586</td>
<td>23.19 / 0.582</td>
<td>25.83 / 0.655</td>
<td>25.90 / 0.724</td>
</tr>
<tr>
<td>50</td>
<td>BM3D</td>
<td>21.81 / 0.509</td>
<td>22.43 / 0.489</td>
<td>25.04 / 0.610</td>
<td>24.93 / 0.663</td>
</tr>
<tr>
<td></td>
<td>GLIDE</td>
<td>21.81 / 0.547</td>
<td>22.60 / 0.518</td>
<td>25.01 / 0.616</td>
<td>24.85 / 0.680</td>
</tr>
<tr>
<td></td>
<td>Proposed</td>
<td><strong>21.93</strong> / 0.540</td>
<td>22.59 / 0.525</td>
<td>25.04 / 0.615</td>
<td><strong>24.95</strong> / 0.673</td>
</tr>
</tbody>
</table>

PSNR[dB] / SSIM
Original image

GLIDE
PSNR 22.43[dB]
SSIM 0.584

BM3D
PSNR 22.53[dB]
SSIM 0.571

Proposed
PSNR 22.71[dB]
SSIM 0.604

Bridge \( \sigma = 40 \)
Visual Assessment

Original image

BM3D
22.53[dB] / 0.571

GLIDE
22.43[dB] / 0.584

BM3D
22.71[dB] / 0.604

Proposed
Visual Assessment

Original image
BM3D
22.53[dB] / 0.571

GLIDE
22.43[dB] / 0.584

GLIDE
Proposed
22.71[dB] / 0.604
Visual Assessment

Original image
BM3D 22.53[dB] / 0.571
GLIDE 22.43[dB] / 0.584

Proposed
22.71[dB] / 0.604
### Execution Time

<table>
<thead>
<tr>
<th>Image size</th>
<th>BM3D</th>
<th>GLIDE</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>256x256</td>
<td>0.8</td>
<td>115.4</td>
<td>51.8</td>
</tr>
<tr>
<td>512x512</td>
<td>3.1</td>
<td>Out of Memory</td>
<td>225.1</td>
</tr>
<tr>
<td>1024x1024</td>
<td>18.1</td>
<td>Out of Memory</td>
<td>946.4</td>
</tr>
</tbody>
</table>

- **Faster** than GLIDE
- **Can be executed in commodity computers**

**Conditions**
- Intel Xeon E5-2690 2.9GHz CPU
- 62.9 GB RAM
- 12 core parallel computing
Conclusion

- **Purpose**
  Improvement of denoising performance for BM3D

- **Method**
  Eigenvalue filtering by CPA **without matrix construction**

- **Result**
  Better denoising performance **visually and numerically**
  Faster execution than GLIDE

- **Future work**
  Improvement of MSE estimation
Reference List

- **Eigenvalue filtering using CPA**

- **BM3D**

- **Global image denoising**