Compressive Adaptive Bilateral Filtering

Pravin Nair, Ruturaj G. Gavaskar and Kunal N. Chaudhury

IISc

Department of Electrical Engineering,
Indian Institute of Science

pravinn@iisc.ac.in

IEEE International Conference on Acoustics, Speech and Signal Processing
(2020)

Classical bilateral filter

Nonlinear edge-preserving smoothing[1]:

$$\boldsymbol{g}(\boldsymbol{x}) = \eta(\boldsymbol{x})^{-1} \sum_{\boldsymbol{y} \in W_{\boldsymbol{x}}} \omega(\boldsymbol{x} - \boldsymbol{y}) \kappa_\theta\big(\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{f}(\boldsymbol{y})\big) \boldsymbol{f}(\boldsymbol{y}),$$

$$\eta(\boldsymbol{x}) = \sum_{\boldsymbol{y} \in W_{\boldsymbol{x}}} \omega(\boldsymbol{x} - \boldsymbol{y}) \kappa_\theta\big(\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{f}(\boldsymbol{y})\big),$$

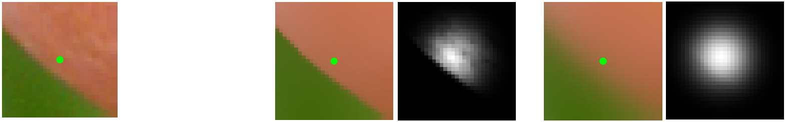
where

- $\boldsymbol{f}$ and $\boldsymbol{g}$ are the input and output RGB images.
- $\boldsymbol{f}(\boldsymbol{x})$ and $\boldsymbol{g}(\boldsymbol{x})$ are vectors.
- ω and $\kappa_\theta =$ Gaussian kernels with variance ρ^2 and θ^2.
- $W_{\boldsymbol{x}} =$ Neighbourhood around pixel $\boldsymbol{x}$ for averaging.

[1] Tomasi and Manduchi, 1998.

Role of θ

Input. Output, $\theta = 30$. Output, $\theta = 200$.



Weights Weights

Adaptation of θ

- θ (width of range kernel) controls the extent of blurring.
- A fixed θ might over- or under-smooth different regions.
- Hence, we allow θ to change at each pixel (a rule is required).
- Useful for controlling the blur in different regions, e.g., more blur to remove coarse textures in images.
- Proposed earlier for a couple of applications:
  - Image sharpening[2].
  - JPEG deblocking[3].

---

[2] Zhang and Allebach, 2008.
[3] Zhang and Gunturk, 2009.

Adaptive bilateral filter (ABF)

- Make the width of the range kernel a function of $\boldsymbol{x}$.

$$
\begin{aligned}
\boldsymbol{g}(\boldsymbol{x}) &= \eta(\boldsymbol{x})^{-1} \sum_{\boldsymbol{y} \in W_{\boldsymbol{x}}} \omega(\boldsymbol{x}-\boldsymbol{y})\,\kappa_{\theta(\boldsymbol{x})}\big(\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{f}(\boldsymbol{y})\big)\,\boldsymbol{f}(\boldsymbol{y}), \\
\eta(\boldsymbol{x}) &= \sum_{\boldsymbol{y} \in W_{\boldsymbol{x}}} \omega(\boldsymbol{x}-\boldsymbol{y})\,\kappa_{\theta(\boldsymbol{x})}\big(\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{f}(\boldsymbol{y})\big).
\end{aligned}
$$

- However, a fixed spatial kernel is used.
- Generalization of the classical bilateral filter.

Computation cost

- $O(\rho^2)$ computations per pixel.
- Higher ρ (window size) is used for higher-resolution images.
- e.g. 60 seconds for a 2 megapixel image on a CPU.
- Real-time implementation is challenging.
- Fast approximation: Approximate the original formula and hope to speed it up, without appreciable loss of visual information.

Fast algorithms for bilateral filtering

- Several fast algorithms for classical bilateral filtering (gray/color).
- Complexity does not scale with filter width ($\mathcal{O}(1)$ implementation).
- Almost all fundamentally require range kernel to be fixed.
- Only one existing fast algorithm (gray and color each) for ABF.
- Accuracy of fast color ABF algorithm[4] is not very high.

[4] Gavaskar and Chaudhury, ICIP 2019.

Our contribution

- Novel fast $\mathcal{O}(1)$ algorithm for ABF of gray and color images.
- Builds on a recently proposed fast algorithm for classical bilateral filtering[5].
- Competitive with existing fast ABF algorithms.
- Main idea:
  - Express ABF in terms of SVD of a kernel matrix K.
  - Replace K by its low-rank approximation using Nyström method.
  - Express the resulting ABF approximation in terms of a few $\mathcal{O}(1)$ convolutions.

[5] Nair and Chaudhury, 2019.

- Let
  - m = No. of pixels.
  - $(\boldsymbol{p}_1, \boldsymbol{p}_2, ..., \boldsymbol{p}_m)$ = Pixelwise RGB/intensity values in $\boldsymbol{f}$.
  - $\{\theta_1, \theta_2, ..., \theta_L\}$ = Set of unique values taken by θ ($L \leq m$).
- For $\ell = 1, \ldots, L$, define $\mathsf{K}_\ell \in \mathbb{R}^{m \times m}$ by

$$\mathsf{K}_\ell(i,j) = \kappa_{\theta_\ell}(\boldsymbol{p}_i, \boldsymbol{p}_j), \qquad i,j = 1, \ldots, m.$$

- Define the kernel matrix K by stacking $\mathsf{K}_1, \ldots, \mathsf{K}_L$:

$$\mathsf{K} = \begin{bmatrix} [\mathsf{K}_1]_{m \times m} \\ [\mathsf{K}_2]_{m \times m} \\ \vdots \\ [\mathsf{K}_L]_{m \times m} \end{bmatrix} \in \mathbb{R}^{mL \times m}.$$

- To compute $\boldsymbol{g}(\boldsymbol{x})$, we need $\kappa_{\theta(\boldsymbol{x})}\big(\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{f}(\boldsymbol{y})\big)$ for $\boldsymbol{y} \in W_{\boldsymbol{x}}$.
- By construction, for $\ell = 1, \ldots, L$ and $i, j = 1, \ldots, m$,

$$\mathsf{K}\big(m(\ell-1)+i, j\big) = \kappa_{\theta_\ell}\big(\boldsymbol{p}_i, \boldsymbol{p}_j\big).$$

- Hence, if $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{p}_i$ and $\theta(\boldsymbol{x}) = \theta_\ell$, define

$$\mathrm{r}(\boldsymbol{x}) = m(\ell-1)+i \quad \text{and} \quad \mathrm{c}(\boldsymbol{x}) = i.$$

- Then,

$$\boldsymbol{g}(\boldsymbol{x}) = \frac{\sum_{\boldsymbol{y}\in W_{\boldsymbol{x}}} \omega(\boldsymbol{x}-\boldsymbol{y})\mathsf{K}\big(\mathrm{r}(\boldsymbol{x}), \mathrm{c}(\boldsymbol{y})\big)\boldsymbol{f}(\boldsymbol{y})}{\sum_{\boldsymbol{y}\in W_{\boldsymbol{x}}} \omega(\boldsymbol{x}-\boldsymbol{y})\mathsf{K}\big(\mathrm{r}(\boldsymbol{x}), \mathrm{c}(\boldsymbol{y})\big)}.$$

- Let $\mathrm{K} = \sum_{k=1}^{m} \sigma_k \boldsymbol{u}_k \boldsymbol{v}_k^\top$ (SVD). Then,

$$\boldsymbol{g}(\boldsymbol{x}) = \frac{\sum_{\boldsymbol{y} \in W_{\boldsymbol{x}}} \omega(\boldsymbol{x} - \boldsymbol{y}) \left\{ \sum_{k=1}^{m} \sigma_k \boldsymbol{u}_k \big(\mathrm{r}(\boldsymbol{x})\big) \boldsymbol{v}_k \big(\mathrm{c}(\boldsymbol{y})\big) \right\} \boldsymbol{f}(\boldsymbol{y})}{\sum_{\boldsymbol{y} \in W_{\boldsymbol{x}}} \omega(\boldsymbol{x} - \boldsymbol{y}) \left\{ \sum_{k=1}^{m} \sigma_k \boldsymbol{u}_k \big(\mathrm{r}(\boldsymbol{x})\big) \boldsymbol{v}_k \big(\mathrm{c}(\boldsymbol{y})\big) \right\}}.$$

- Switching the summations,

$$\boldsymbol{g}(\boldsymbol{x}) = \frac{\sum_{k=1}^{m} \sigma_k \boldsymbol{u}_k \big(\mathrm{r}(\boldsymbol{x})\big) (\omega * \boldsymbol{h}_k)(\boldsymbol{x})}{\sum_{k=1}^{m} \sigma_k \boldsymbol{u}_k \big(\mathrm{r}(\boldsymbol{x})\big) (\omega * d_k)(\boldsymbol{x})},$$

  where

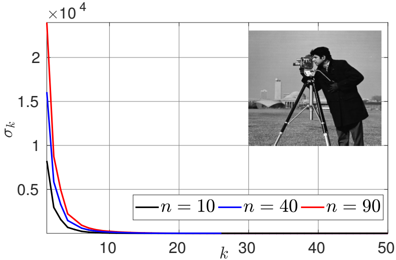
  - $d_k(\boldsymbol{x}) = \boldsymbol{v}_k \big(\mathrm{c}(\boldsymbol{x})\big)$, $\boldsymbol{h}_k(\boldsymbol{x}) = d_k(\boldsymbol{x}) \boldsymbol{f}(\boldsymbol{x})$.
  - $\omega * \boldsymbol{f}$ = Convolution of ω with an image $\boldsymbol{f}$.

- Gaussian convolutions can be implemented with $\mathcal{O}(1)$ cost[6].

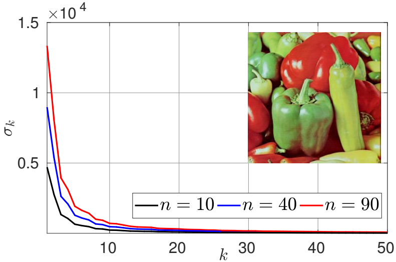
[6] Deriche, 1993.

Why SVD?

- Empirically, the singular values of K decay rapidly.
- Hence, we may keep just the top few components, say $m_0 \ll m$.
- Will reduce the no. of convolutions from m to m_0.
- Desirable because Gaussian convolutions dominate computational cost.



Cameraman (gray).



Peppers (color).

- Populating K and finding SVD is costly ($\mathcal{O}(m^3L)$ complexity).
- Hence, we use another approximation for the SVD itself.
- Nyström approximation[7]:
  - Choose m_o 'representative' points from $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_m$.
  - Choose $L_0 \ll L$ 'representative' values from $\theta_1, \ldots, \theta_L$.
  - Populate a $m_0 L_0 \times m_0$ matrix A similar to K.
  - Compute SVD of A ($\mathcal{O}(m_0^3 L_0)$ complexity).
  - Take singular values of A as the top m_0 singular values of K.
  - Estimate singular vectors of K by 'extrapolating' those of A.

[7] Nemtsov et al., 2016.

Proposed ABF approximation

- Instead of SVD of K, use its Nyström approximation,

$$\widehat{\mathrm{K}} = \sum_{k=1}^{m_0} \alpha_k \hat{\boldsymbol{u}}_k \hat{\boldsymbol{v}}_k^\top .$$

- This gives the proposed ABF approximation

$$\hat{\boldsymbol{g}}(\boldsymbol{x}) = \frac{\sum_{k=1}^{m_0} \alpha_k \hat{\boldsymbol{u}}_k\big(\mathrm{r}(\boldsymbol{x})\big)(\omega * \hat{\boldsymbol{h}}_k)(\boldsymbol{x})}{\sum_{k=1}^{m_0} \alpha_k \hat{\boldsymbol{u}}_k\big(\mathrm{r}(\boldsymbol{x})\big)(\omega * \hat{d}_k)(\boldsymbol{x})},$$

where $\hat{d}_k(\boldsymbol{x}) = \hat{\boldsymbol{v}}_k\big(\mathrm{c}(\boldsymbol{x})\big)$, $\hat{\boldsymbol{h}}_k(\boldsymbol{x}) = \hat{d}_k(\boldsymbol{x})\boldsymbol{f}(\boldsymbol{x})$.

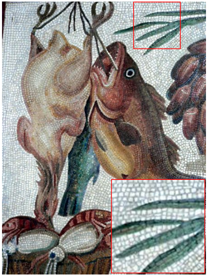
- Dominant computation: m_0 $\mathcal{O}(1)$ Gaussian convolutions.
- Accuracy of approximation increases with m_0.

Code: `https://github.com/pravin1390/CABF`

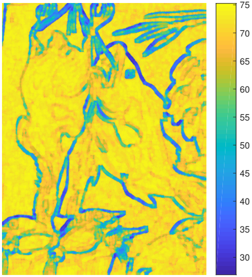
Brief overview:

- Objective: Remove coarse textures in an image.
- Can be accomplished using the ABF[8].
- $\theta(\boldsymbol{x})$ is decided using a metric that distinguishes texture from edges.
- We use our proposed algorithm for color filtering.

[8] Gavaskar and Chaudhury, TIP 2019.



Input.



θ map.

Brute-force (30s).

Proposed (40.3 dB, 1.6s).

Method 1 (30.4 dB, 0.9s).

Method 1: Fast ABF (Gavaskar and Chaudhury, ICIP 2019).

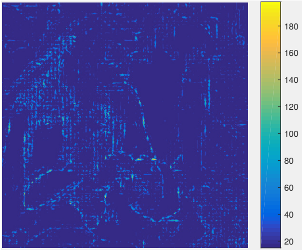
Brief overview:

- Objective: Smooth out blocking artifacts in JPEG-compressed images.
- For grayscale images, can be accomplished using the ABF[9].
- $\theta(\boldsymbol{x})$ is decided using a technique proposed previously[9].
- We use our proposed algorithm for grayscale filtering.

[9] Zhang and Gunturk, 2009.

Ground-truth.

Input.



θ map.

Brute-force
(28.6 dB, 3.3s).

Proposed
(28.5 dB, 0.15s).

Method 2
(28.5 dB, 0.16s).

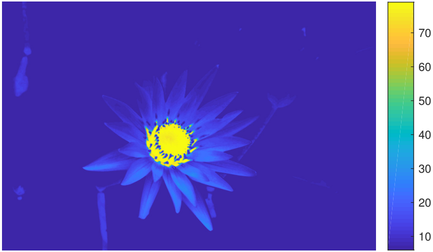
Method 2: Fast ABF (Gavaskar and Chaudhury, TIP 2019).

Brief overview:

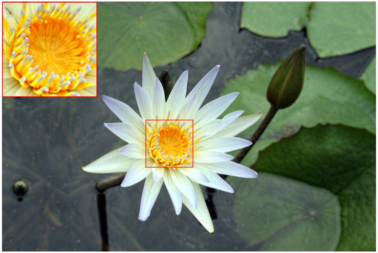
- Objective: Enhance details, but not to the same extent everywhere.
- More enhancement in regions which are more visually salient.
- Can be accomplished using the ABF[10].
- $\theta(\boldsymbol{x})$ is decided using a saliency map.
- We use our proposed algorithm for color filtering.

[10]Ghosh et al., 2019.

Input.



θ map.



Brute-force
(24s).

Proposed
(38.6 dB, 5s).

Conclusion

- Proposed $\mathcal{O}(1)$ algorithm for adaptive bilateral filtering.
- Better accuracy than the only existing $\mathcal{O}(1)$ algorithm for color ABF.
- Core idea: Low-rank Nyström approximation of a kernel matrix.
- Achieves up to $20\times$ speedup with high accuracy.
- Applicable to texture filtering, detail enhancement, and deblocking.
- Can be extended in principle to multi-channel images, and NLM filtering.

Research supported by CRG Grant SB/S3/EECE/281/2016 from DST-SERB, Government of India.

References I

- C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," Proc. IEEE International Conference on Computer Vision, pp. 839–846, 1998.

- B. Zhang and J. P. Allebach, "Adaptive bilateral filter for sharpness enhancement and noise removal," IEEE Transactions on Image Processing, vol. 17, no. 5, pp. 664–678, 2008.

- M. Zhang and B. K. Gunturk, "Compression artifact reduction with adaptive bilateral filtering," Proc. SPIE Visual Communications and Image Processing, vol. 7257, 2009.

- P. Nair and K. N. Chaudhury, "Fast high-dimensional kernel filtering," IEEE Signal Processing Letters, vol. 26, no. 2, pp. 377–381, 2019.

- R. Deriche, "Recursively implementing the Gaussian and its derivatives," Research Report RR-1893, INRIA, 1993.

References II

- R. G. Gavaskar and K. N. Chaudhury, "Fast adaptive bilateral filtering," IEEE Transactions on Image Processing, vol. 28, no. 2, pp. 779–790, 2019.
- A. Nemtsov, A. Averbuch, and A. Schclar, "Matrix compression using the Nyström method," Intelligent Data Analysis, vol. 20, no. 5, pp. 997–1019, 2016.
- S. Ghosh, R. G. Gavaskar, and K. N. Chaudhury, "Saliency guided image detail enhancement," Proc. National Conference on Communications, pp. 1–6, 2019.
- R. G. Gavaskar and K. N. Chaudhury, "Fast adaptive bilateral filtering of color images," Proc. IEEE International Conference on Image Processing, pp. 180–184, 2019.

Thanks for listening!