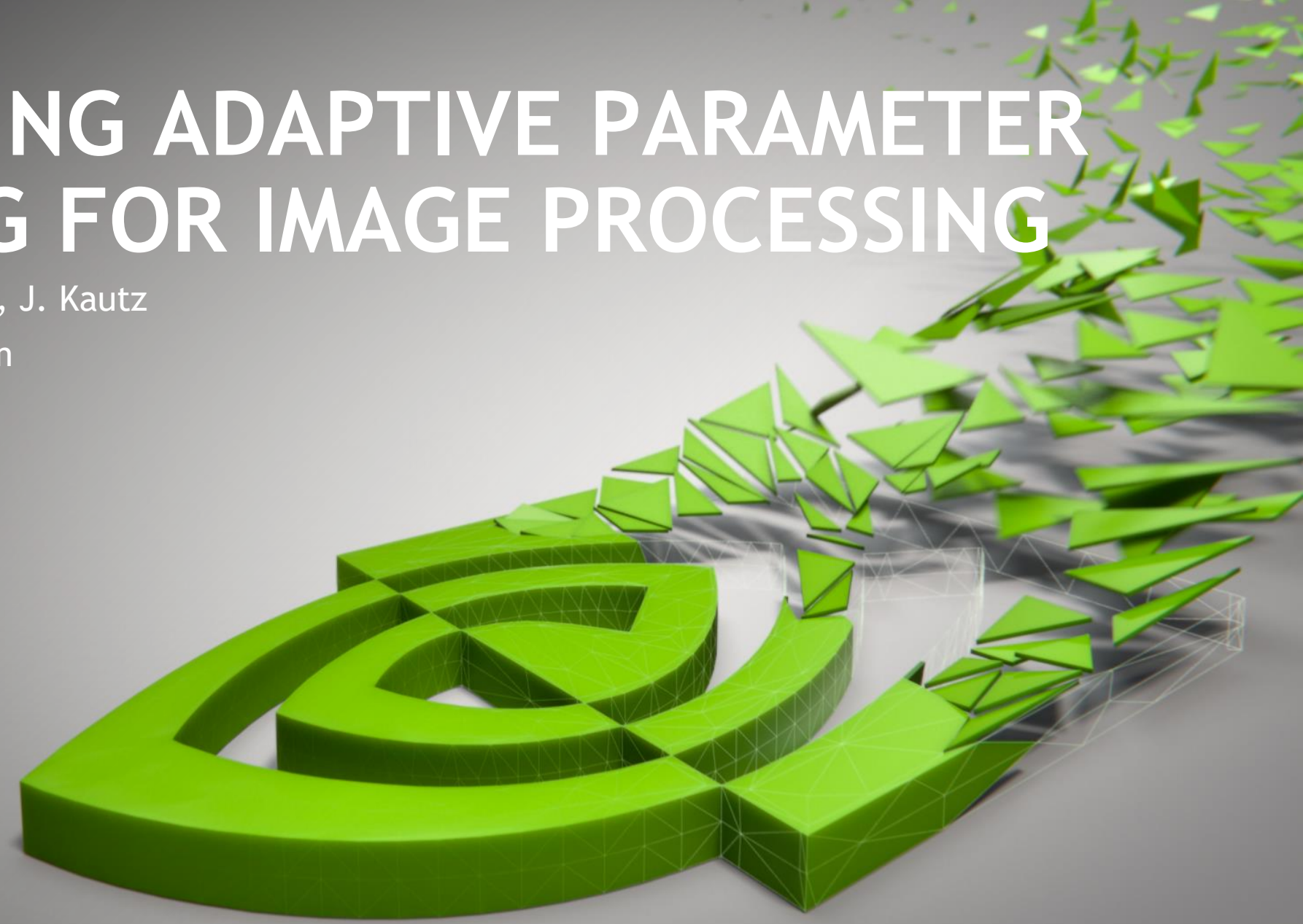
LEARNING ADAPTIVE PARAMETER TUNING FOR IMAGE PROCESSING

J. Dong, I. Frosio*, J. Kautz

ifrosio@nvidia.com



MOTIVATION

NON-ADAPTIVE VS. ADAPTIVE FILTERING

Box-filtering example

Ground truth

Noisy, PSNR = 18.61dB

NON-ADAPTIVE VS. ADAPTIVE FILTERING

Box-filtering example

Ground truth

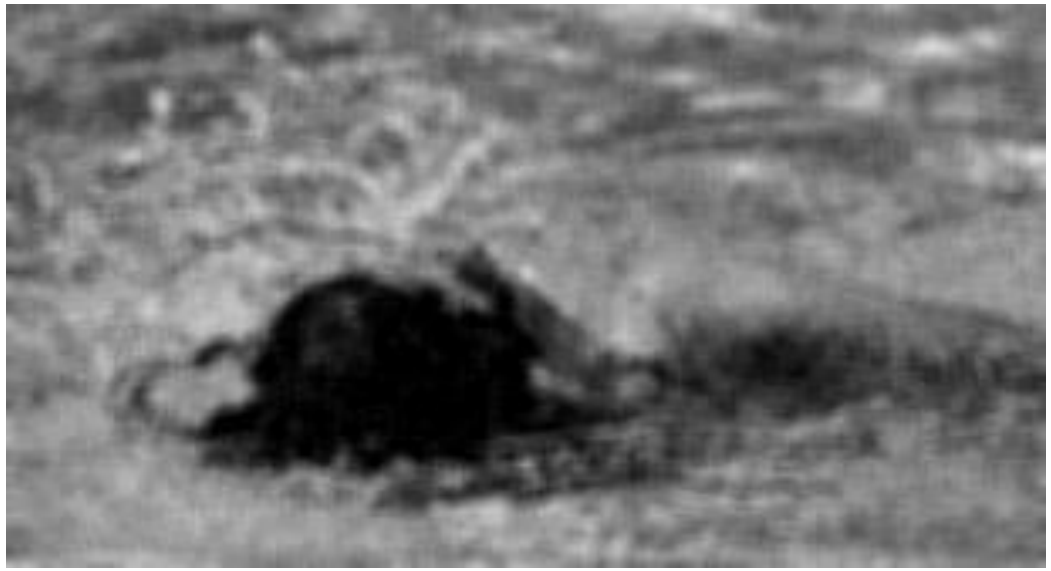
Noisy + 3x3 box filter, PSNR = 23.47dB

NON-ADAPTIVE VS. ADAPTIVE FILTERING

Box-filtering example

Ground truth

Noisy + 5x5 box filter, PSNR = 22.14dB



NON-ADAPTIVE VS. ADAPTIVE FILTERING

Box-filtering example

Ground truth

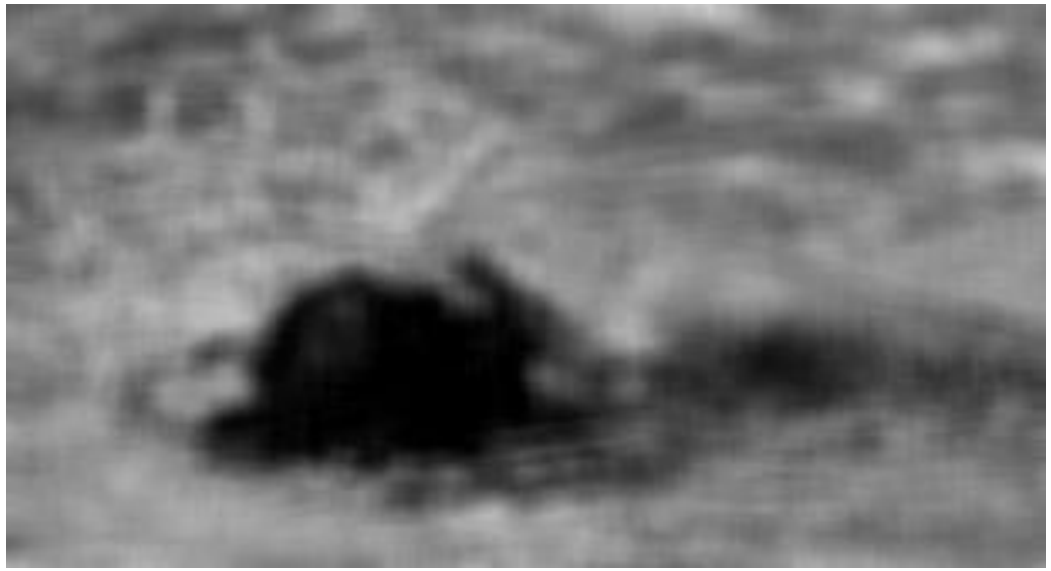
Noisy + 7x7 box filter, PSNR = 21.12dB

NON-ADAPTIVE VS. ADAPTIVE FILTERING

Box-filtering example

Ground truth

Noisy + 9x9 box filter, PSNR = 20.62dB

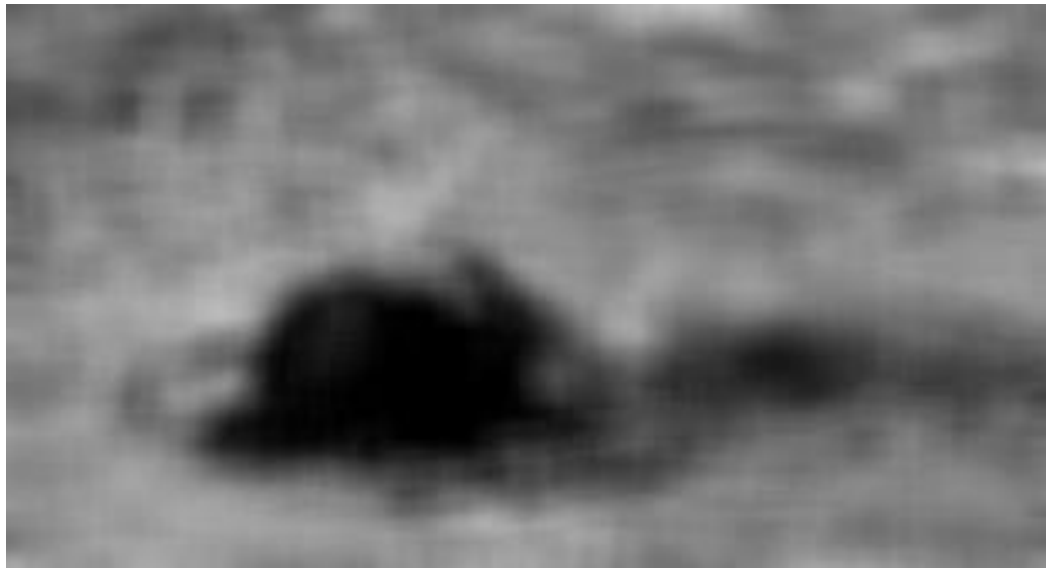


NON-ADAPTIVE VS. ADAPTIVE FILTERING

Box-filtering example

Ground truth

Noisy + 13x13 box filter, PSNR = 19.81dB

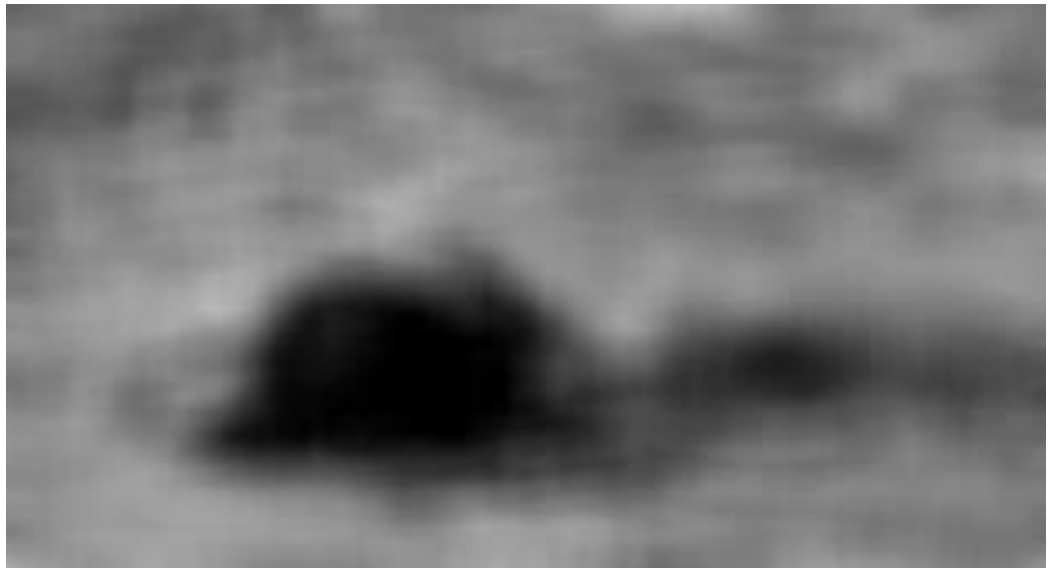


NON-ADAPTIVE VS. ADAPTIVE FILTERING

Box-filtering example

Ground truth

Noisy + 17x17 box filter, PSNR = 19.28dB

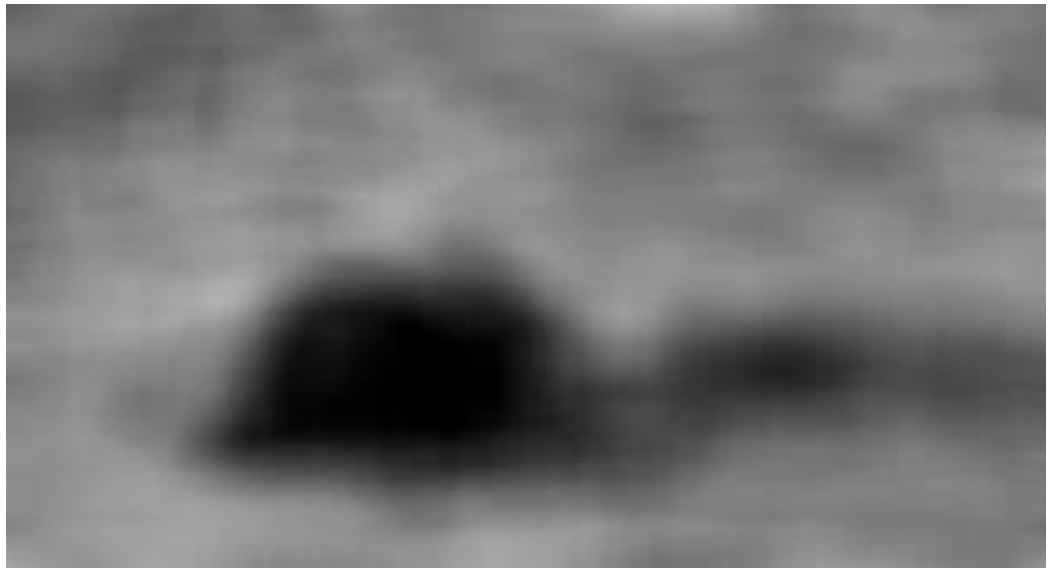


NON-ADAPTIVE VS. ADAPTIVE FILTERING

Box-filtering example

Ground truth

Noisy + 21x21 box filter, PSNR = 18.90dB

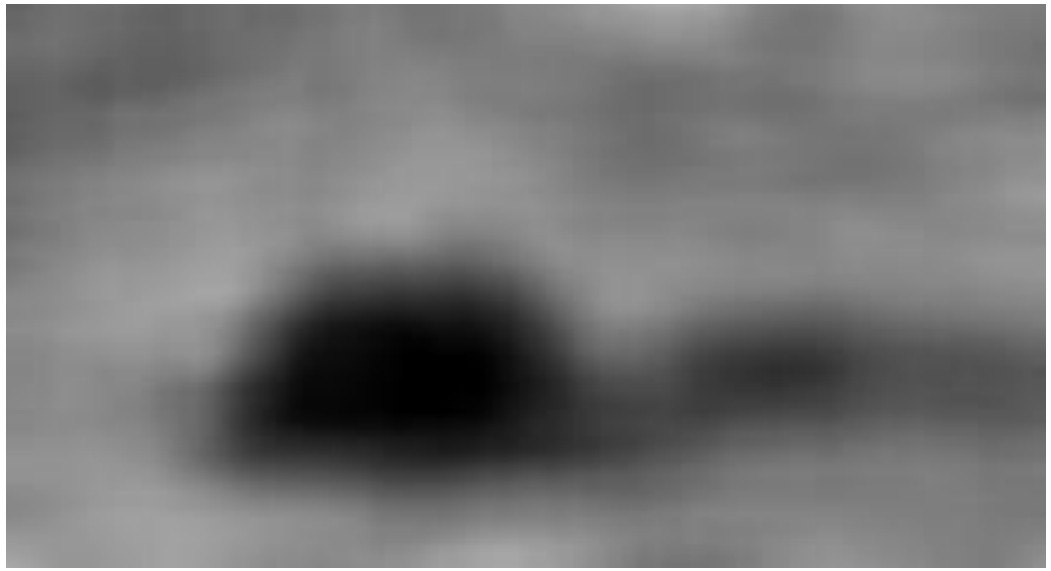


NON-ADAPTIVE VS. ADAPTIVE FILTERING

Box-filtering example

Ground truth

Noisy + 29x29 box filter, PSNR = 18.41dB

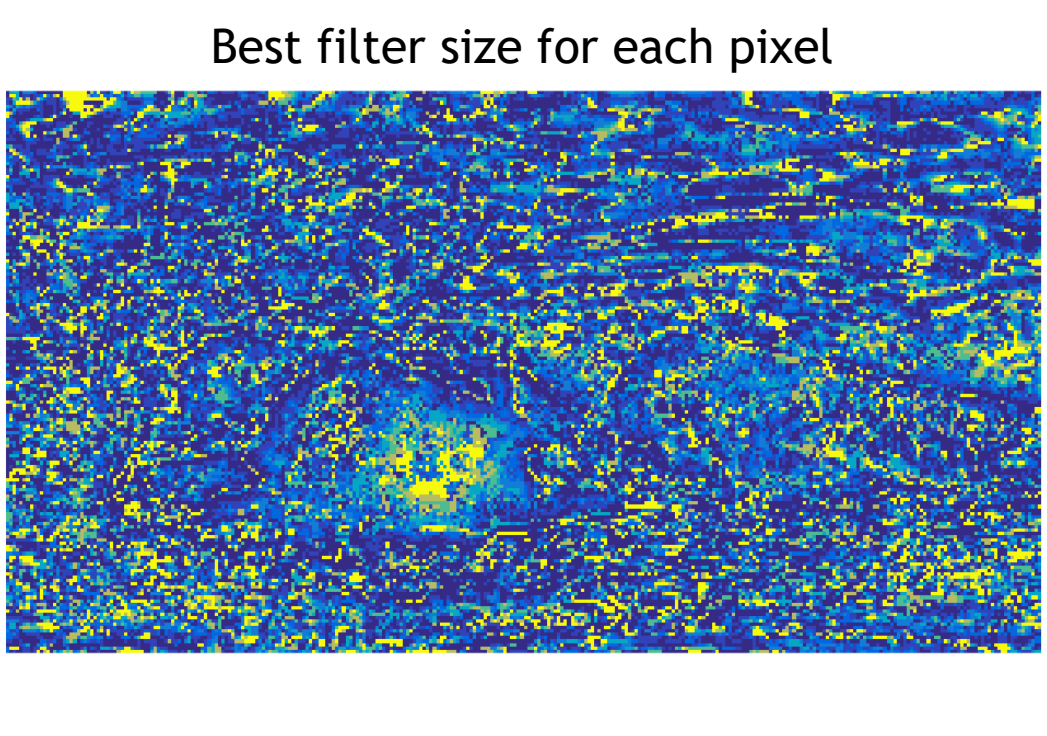


NON-ADAPTIVE VS. ADAPTIVE FILTERING

Box-filtering example

Ground truth

Best filter size for each pixel



NON-ADAPTIVE VS. ADAPTIVE FILTERING

Box-filtering example

Ground truth

Adaptive box filter, PSNR = 26.34dB

NON-ADAPTIVE VS. ADAPTIVE FILTERING

Box-filtering example

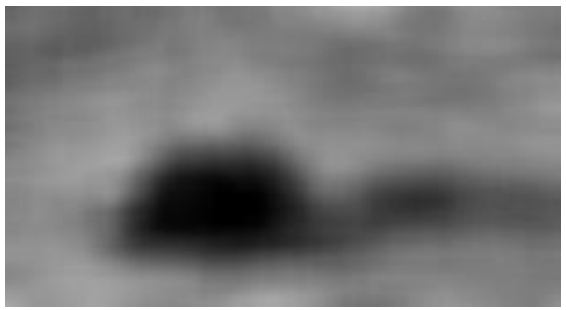
Ground truth

Noisy + 3x3 box filter, PSNR = 23.47dB

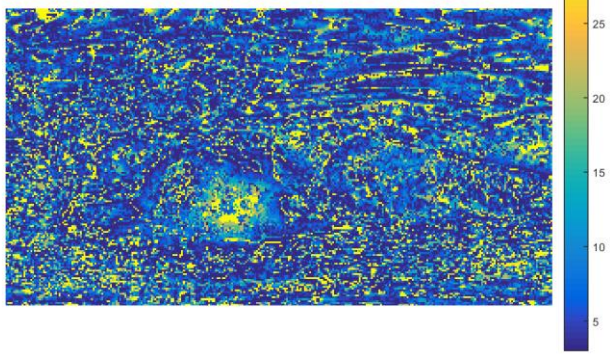
Adaptive box filter, PSNR = 26.34dB

Noisy, PSNR = 18.61dB

Noisy + 29x29 box filter, PSNR = 18.41dB



Best filter size for each pixel



A LIST OF PROS…

… For adaptive filters

A generalization of an existing filtering technique:

- **Explainability** (vs. pure ML)
- Power / computationally **efficient** (vs. pure ML)

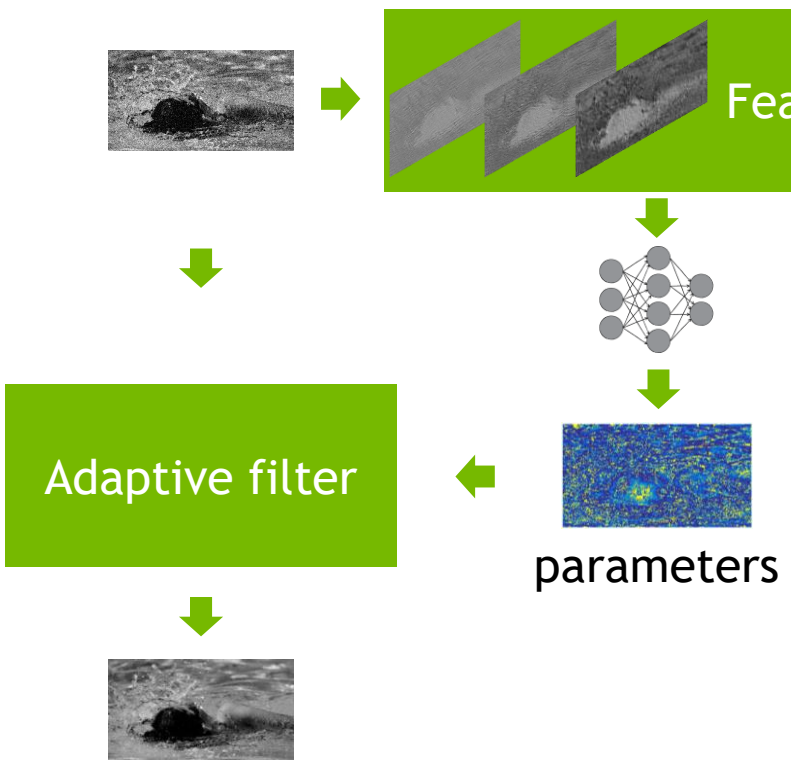
Open question:

- **Learn** parameter tuning (requires ML)

SUMMARY

SUMMARY

Learning Adaptive Parameter Tuning for Image Processing



$$\mathbf{f}_{x,y} = \left[f_{x,y}^{0} \; f_{x,y}^{1} \; \cdots \; f_{x,y}^{F-1} \right]^{\top}$$

$$h\left(\mathbf{f}_{x,y}; \boldsymbol{\theta}_k\right) = \theta_k^0 + {\boldsymbol{\theta}_k^1}^{\top} \mathbf{f}_{x,y} + \mathbf{f}_{x,y}^{\top} \boldsymbol{\Theta}_k^2 \mathbf{f}_{x,y}$$

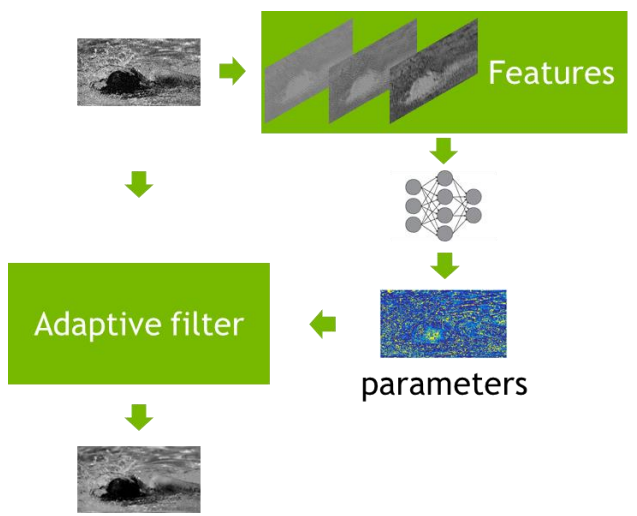
$$p_{x,y}^{k} = p_{\min}^{k} + \frac{p_{\max}^{k} - p_{\min}^{k}}{1 + e^{-h(\mathbf{f}_{x,y}; \boldsymbol{\theta}_k)}}$$

[1] I. Frosio, K. Egiazarian, K. Pulli, Machine Learning for Adaptive Bilateral Filtering, 2015.

SUMMARY

Learning Adaptive Parameter Tuning for Image Processing

1. Method:
    1. Feature design
    2. Training
2. Results:
    1. NLM denoising
    2. Demosaicing



$$\mathbf{f}_{x,y} = \left[f^0_{x,y}\ f^1_{x,y}\ \dots\ f^{F-1}_{x,y}\right]^\top$$

$$h\left(\mathbf{f}_{x,y};\boldsymbol{\theta}_k\right) = \theta^0_k + \boldsymbol{\theta}^{1\top}_k \mathbf{f}_{x,y} + \mathbf{f}^\top_{x,y} \boldsymbol{\Theta}^2_k \mathbf{f}_{x,y}$$

$$p^k_{x,y} = p^k_{\min} + \frac{p^k_{\max} - p^k_{\min}}{1 + e^{-h(\mathbf{f}_{x,y};\boldsymbol{\theta}_k)}}$$

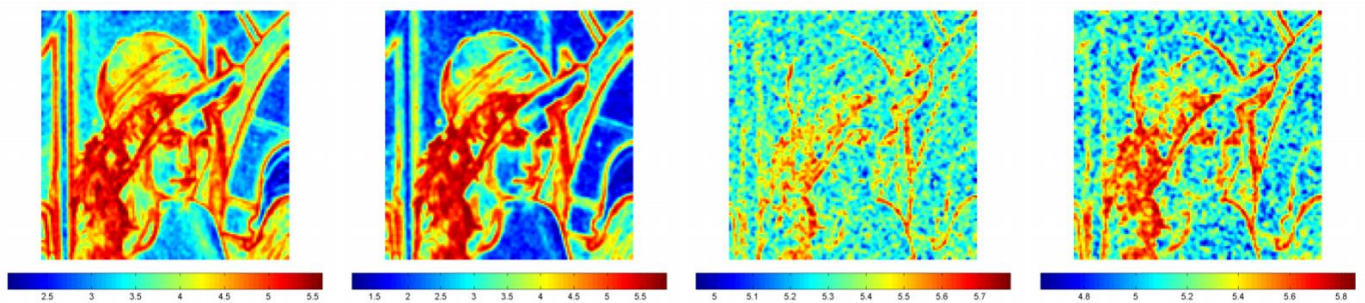
[1] I. Frosio, K. Egiazarian, K. Pulli, Machine Learning for Adaptive Bilateral Filtering, 2015.

METHOD

METHOD: HAND PICKED FEATURES

Local entropy / gradient entropy (noise free)

Local entropy / gradient entropy (noisy)



(a) $e_{\mathbf{i}}(x, y)$, $\sigma_n = 0$. (b) $e_{\mathbf{g}}(x, y)$, $\sigma_n = 0$. (c) $e_{\mathbf{i}}(x, y)$, $\sigma_n = 20$. (d) $e_{\mathbf{g}}(x, y)$, $\sigma_n = 20$.

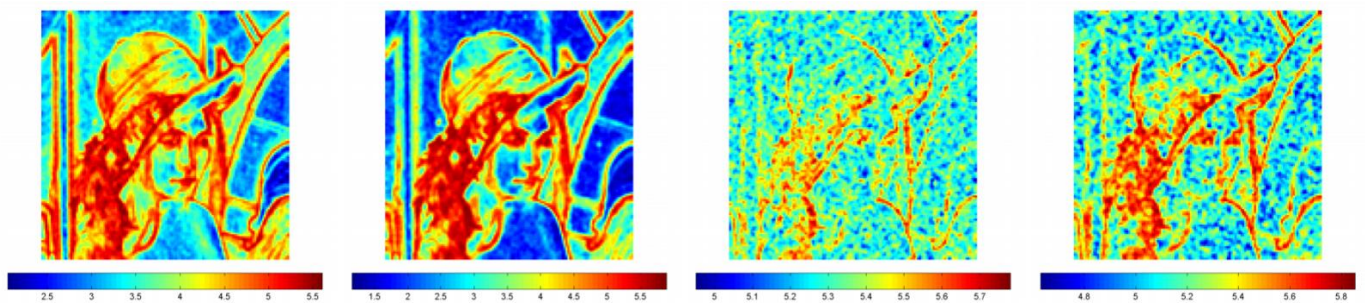
- Computationally efficient features [1]:
  - Local (3x3, 5x5) variance
  - Local (3x3, 7x7) entropy
  - Local (3x3, 7x7) gradient entropy

[1] I. Frosio, K. Egiazarian, K. Pulli, Machine Learning for Adaptive Bilateral Filtering, 2015.

METHOD: HAND PICKED FEATURES

Local entropy / gradient entropy (noise free)

Local entropy / gradient entropy (noisy)



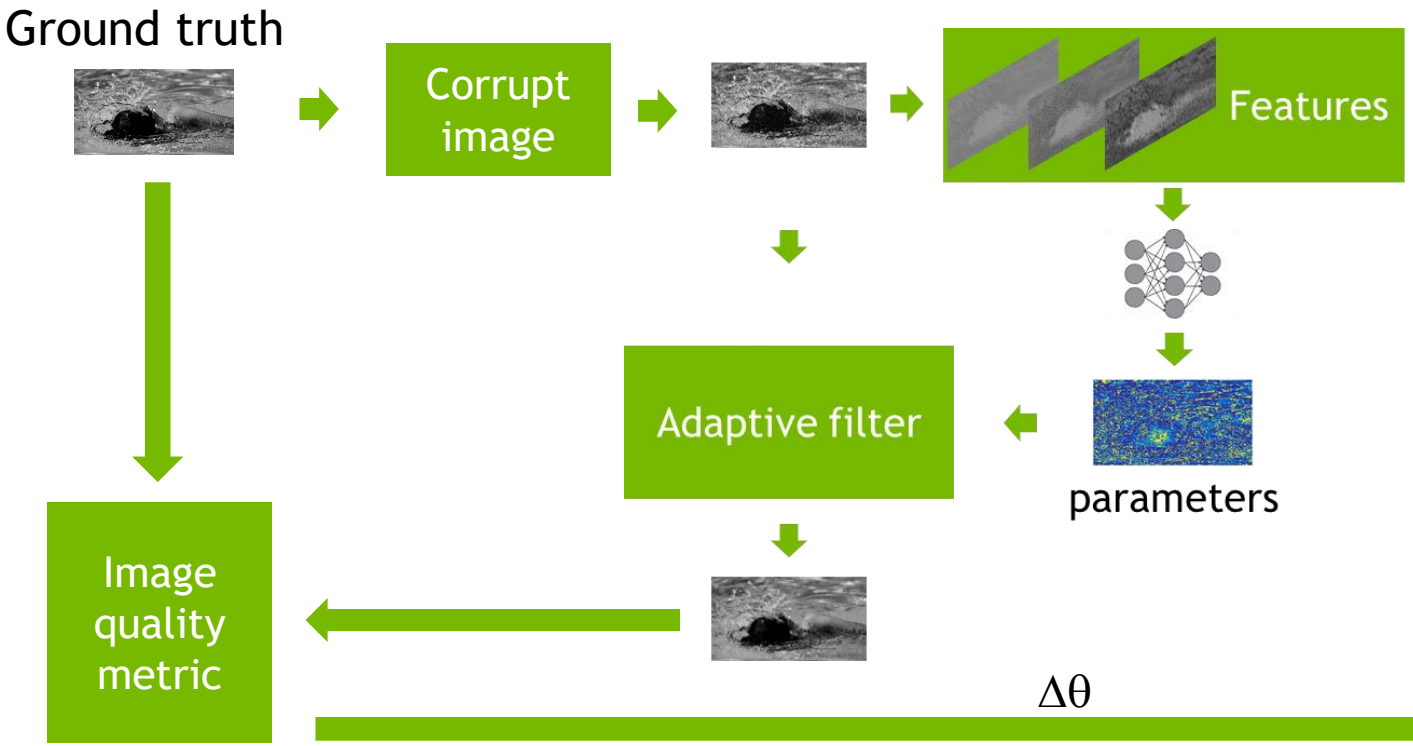
(a) $e_{\mathbf{i}}(x, y)$, $\sigma_n = 0.$ (b) $e_{\mathbf{g}}(x, y)$, $\sigma_n = 0.$ (c) $e_{\mathbf{i}}(x, y)$, $\sigma_n = 20.$ (d) $e_{\mathbf{g}}(x, y)$, $\sigma_n = 20.$

- Computationally efficient features [1]:
  - Local (3x3, 5x5) variance: **edge / texture / noise intensity**
  - Local (3x3, 7x7) entropy: **how many sets (homogeneous / textured / edge area)**
  - Local (3x3, 7x7) gradient entropy: **how many gradient sets (texture / edge area)**

[1] I. Frosio, K. Egiazarian, K. Pulli, Machine Learning for Adaptive Bilateral Filtering, 2015.

METHOD: TRAINING

Cost function



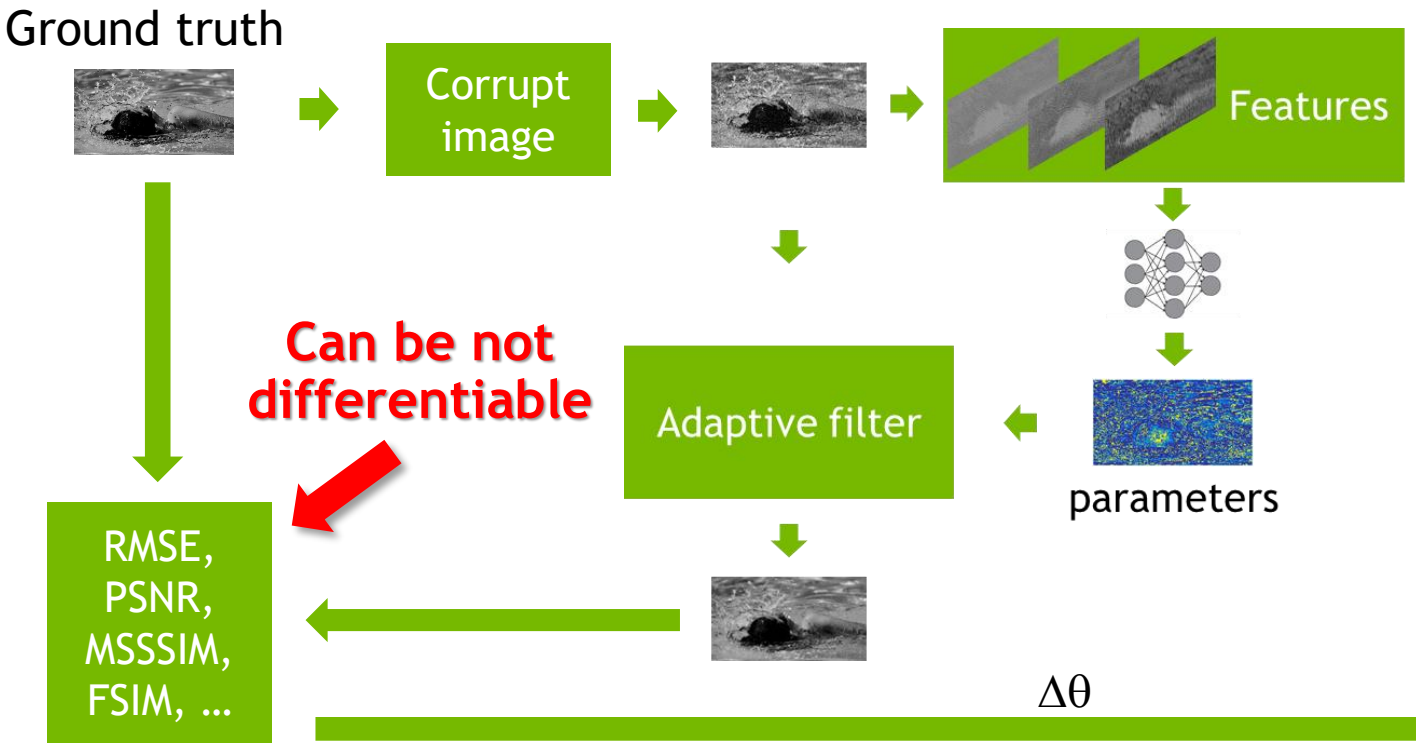
$$\mathbf{f}_{x,y} = \left[f_{x,y}^{0} \; f_{x,y}^{1} \; \cdots \; f_{x,y}^{F-1} \right]^{\top}$$

$$h\left(\mathbf{f}_{x,y}; \boldsymbol{\theta}_k\right) = \theta_k^0 + \boldsymbol{\theta}_k^{1\top} \mathbf{f}_{x,y} + \mathbf{f}_{x,y}^{\top} \boldsymbol{\Theta}_k^2 \mathbf{f}_{x,y}$$

$$p_{x,y}^k = p_{\min}^k + \frac{p_{\max}^k - p_{\min}^k}{1 + e^{-h(\mathbf{f}_{x,y}; \boldsymbol{\theta}_k)}}$$

METHOD: TRAINING

Differentiable / not differentiable cost function / filter



$$\mathbf{f}_{x,y} = \left[f_{x,y}^{0} \; f_{x,y}^{1} \; \dots \; f_{x,y}^{F-1} \right]^{\top}$$

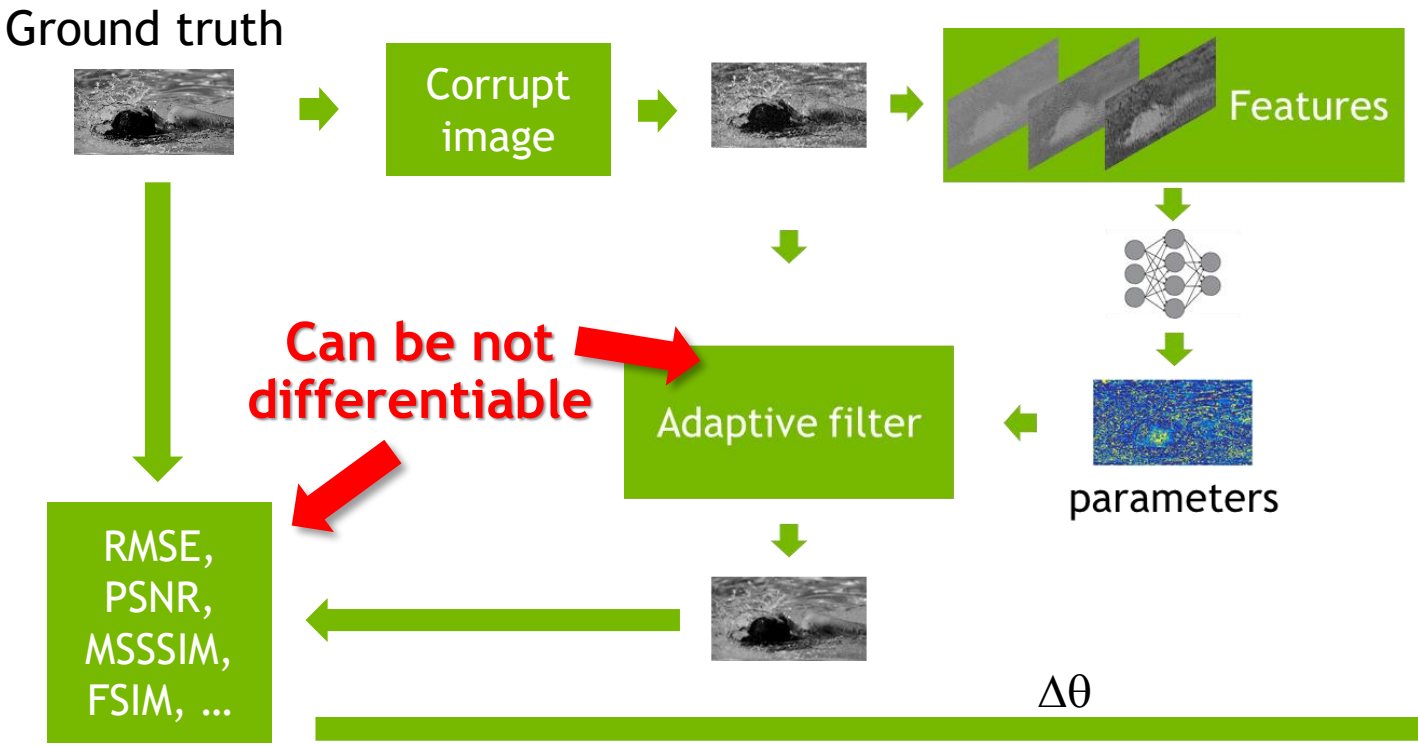
$$h\left(\mathbf{f}_{x,y}; \boldsymbol{\theta}_k\right) = \theta_k^0 + \boldsymbol{\theta}_k^{1\top} \mathbf{f}_{x,y} + \mathbf{f}_{x,y}^{\top} \boldsymbol{\Theta}_k^2 \mathbf{f}_{x,y}$$

$$p_{x,y}^k = p_{\min}^k + \frac{p_{\max}^k - p_{\min}^k}{1 + e^{-h(\mathbf{f}_{x,y}; \boldsymbol{\theta}_k)}}$$

N. Ponomarenko et. al, Color Image Database TID2013: Peculiarities and Preliminary Results, 2013.

METHOD: TRAINING

Differentiable / not differentiable cost function / filter



$$\mathbf{f}_{x,y} = \left[f_{x,y}^{0} \; f_{x,y}^{1} \; \dots \; f_{x,y}^{F-1} \right]^{\top}$$

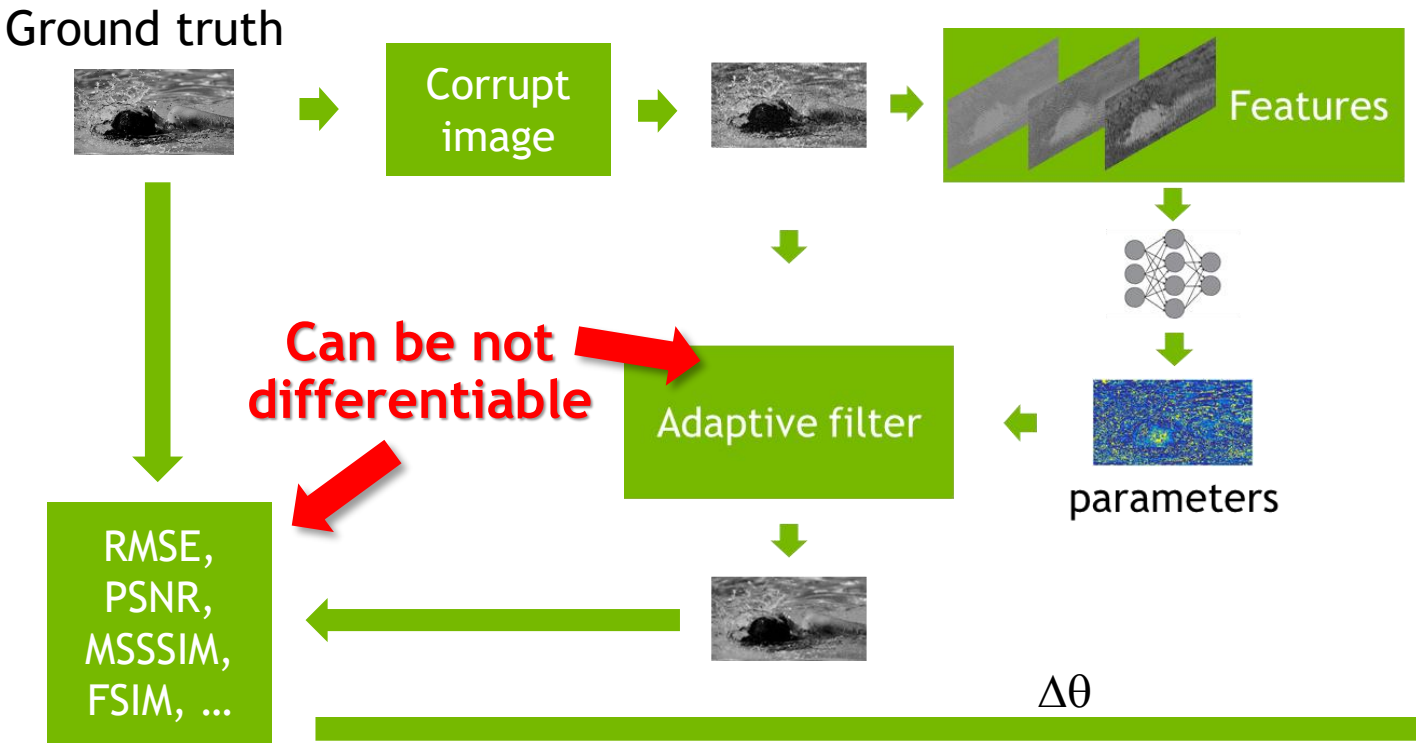
$$h\left(\mathbf{f}_{x,y}; \boldsymbol{\theta}_k\right) = \theta_k^0 + \boldsymbol{\theta}_k^{1\top} \mathbf{f}_{x,y} + \mathbf{f}_{x,y}^{\top} \boldsymbol{\Theta}_k^2 \mathbf{f}_{x,y}$$

$$p_{x,y}^k = p_{\min}^k + \frac{p_{\max}^k - p_{\min}^k}{1 + e^{-h(\mathbf{f}_{x,y};\boldsymbol{\theta}_k)}}$$

[1] A. Buades, B. Coll, and J. M. Morel, “A review of image denoising algorithms, with a new one”, 2005.
[2] I. Frosio, C. Olivieri, M. Lucchese, N. Borghese, and P. Boccacci, “Bayesian denoising in digital radiography: A comparison in the dental field”, 2013.

METHOD: TRAINING

Differentiable / not differentiable cost function / filter



$$\mathbf{f}_{x,y} = \left[f^0_{x,y} \; f^1_{x,y} \; \dots \; f^{F-1}_{x,y} \right]^\top$$

$$h\left(\mathbf{f}_{x,y}; \boldsymbol{\theta}_k\right) = \theta^0_k + \boldsymbol{\theta}^{1\top}_k \mathbf{f}_{x,y} + \mathbf{f}^\top_{x,y} \boldsymbol{\Theta}^2_k \mathbf{f}_{x,y}$$

$$p^k_{x,y} = p^k_{\min} + \frac{p^k_{\max} - p^k_{\min}}{1 + e^{-h(\mathbf{f}_{x,y}; \boldsymbol{\theta}_k)}}$$

Nelder-Mead simplex (does not require derivatives)

RESULTS

RESULTS: NLM DENOISING

Non-adaptive filter

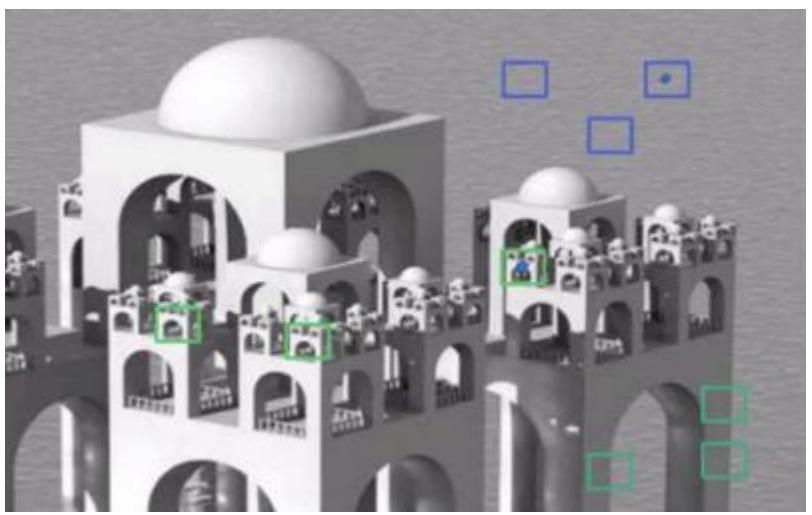


image from opencv.org

$$\mathbf{q}^d = \sum_{j=1}^{N} w_j \ \mathbf{r}_j, \tag{1}$$

$$w_j = e^{-\max\{d^2(\mathbf{q}^n,\mathbf{r}_j)/(p^0)^2-2\sigma^2,0\}/(p^1\sigma)^2}, \tag{2}$$

p^0: patch size [optimal image scale]

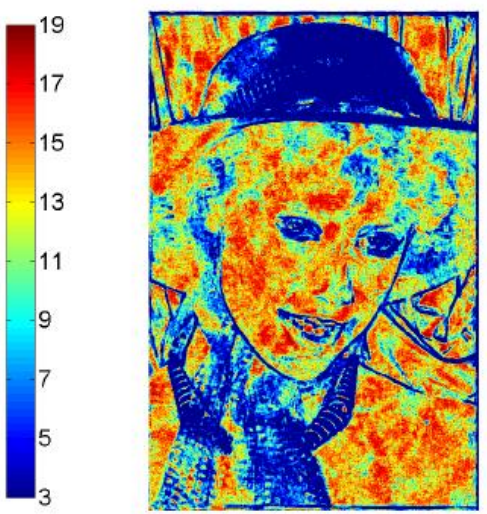
p^1: affinity measure [bias/variance tradeoff]

[1] A. Buades, B. Coll, and J. M. Morel, A review of image denoising algorithms, with a new one 2005.

RESULTS: NLM DENOISING

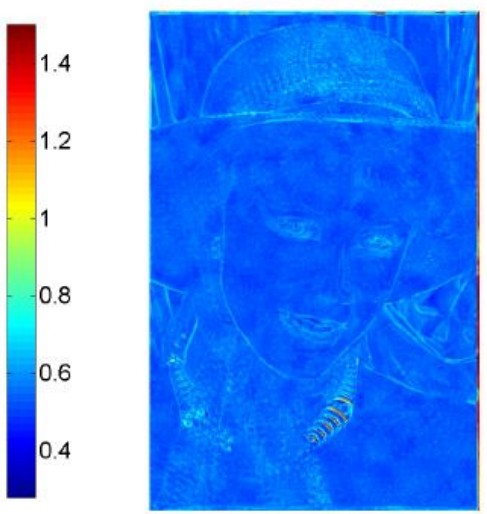
AWGN, σ=20 - learned adaptive filter, maximize PSNR

Patch size



(a) $\text{aANLM}_{\text{PSNR}}$, p^0

Filtering par



(b) $\text{aANLM}_{\text{PSNR}}$, p^1

- Large patch size in homogeneous areas
- Small patch size for high frequency details
- Favor variance reduction in homogeneous areas
- Small bias close to the edges, use similarity
- Analytical results [1] suggesting a similar strategy

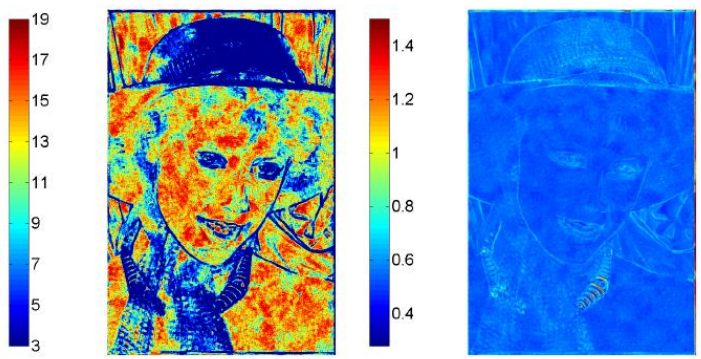
[1] V. Duval, J.-F. Aujol, and Y. Gousseau,"On the parameter choice for the non-local means, 2011

RESULTS: NLM DENOISING

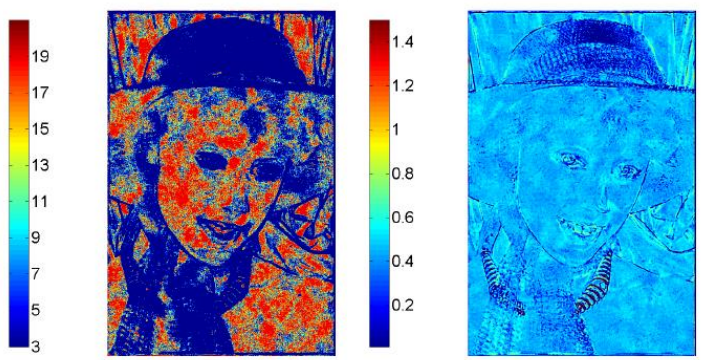
AWGN, σ=20 - learned adaptive filter, maximize MS-SSIM

Patch size

Filtering par



(a) aANLM$_{PSNR}$, p^0 (b) aANLM$_{PSNR}$, p^1

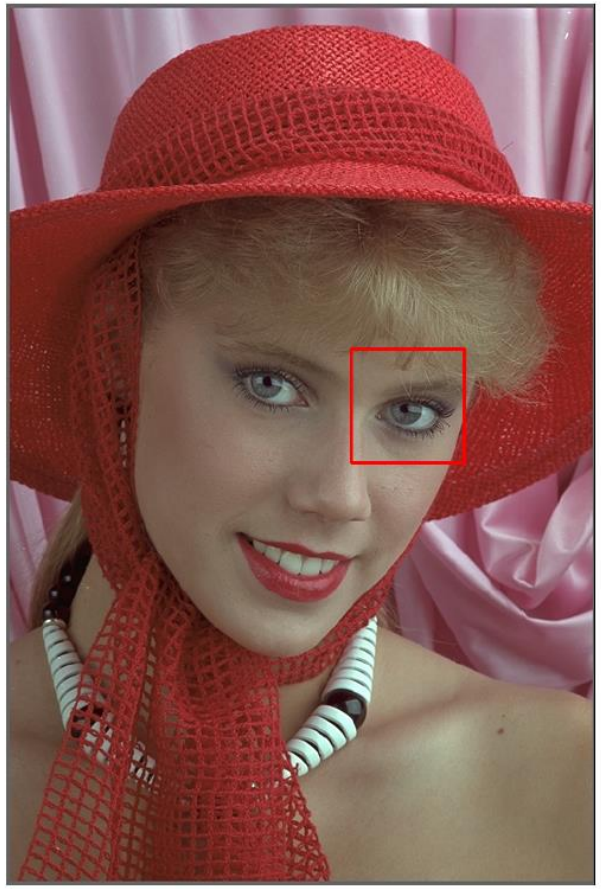


(c) aANLM$_{MS\text{-}SSIM}$, p^0 (d) aANLM$_{MS\text{-}SSIM}$, p^1

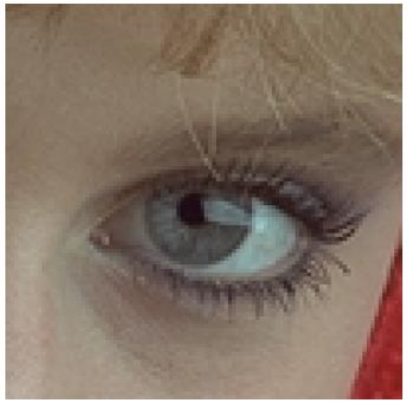
[1] Z. Zwang, E. Simoncelli, A. Bovik, Multiscale Structural Similarity for Image Quality Assessement, 2003.

RESULTS: NLM DENOISING

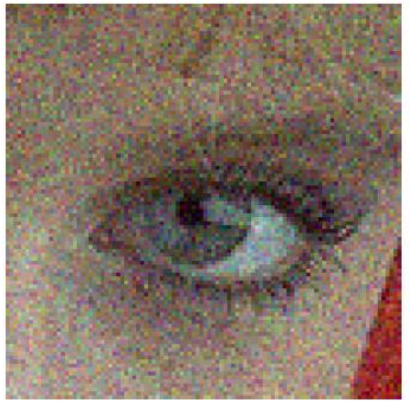
PSNR vs. MS-SSIM



(a) Original image



(b) Ground truth

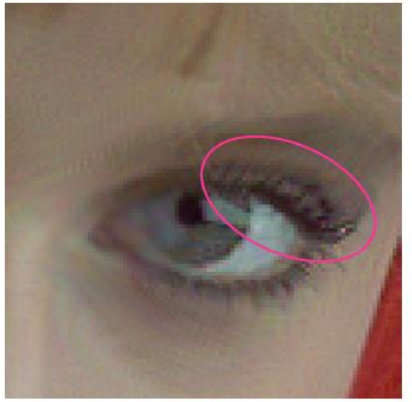


(c) Noisy

(d) ANLM

(e) $\text{ANLM}_{\text{PSNR}}$

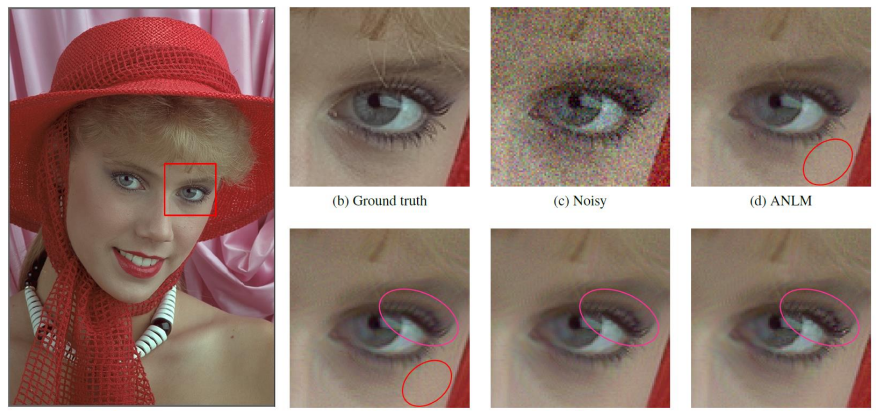
(f) $\text{aANLM}_{\text{PSNR}}$



(g) $\text{aANLM}_{\text{MS-SSIM}}$

RESULTS: NLM DENOISING

Numerical results



(a) Original image (b) Ground truth (c) Noisy (d) ANLM (e) $\text{ANLM}_{\text{PSNR}}$ (f) $\text{aANLM}_{\text{PSNR}}$ (g) $\text{aANLM}_{\text{MS-SSIM}}$

	Noisy	ANLM	$\text{ANLM}_{\text{PSNR}}$	$\text{ANLM}_{\text{MS-SSIM}}$	$\text{aANLM}_{\text{PSNR}}$	$\text{aANLM}_{\text{MS-SSIM}}$
p^0	-	5	9	19	adaptive, learned	adaptive, learned
$p^1 \cdot \sigma$	-	0.40σ	0.51σ	0.49σ	adaptive, learned	adaptive, learned
PSNR	22.11	30.06	30.27	30.00	**30.80**	30.48
SSIM	0.6750	0.8952	0.8992	0.9006	0.9056	**0.9084**
MS-SSIM	0.8192	0.9417	0.9445	0.9461	0.9480	**0.9496**

RESULTS: DEMOSAICING

Blending filters

- SOA algorithms for demosaicing:
  - ECC [1]
  - ARI [2]
  - CS [3]
- Blending outputs:
  - p^0 *ECC + p^1*ARI + p^2*CS
  - $p^0 + p^1 + p^2 = 1$

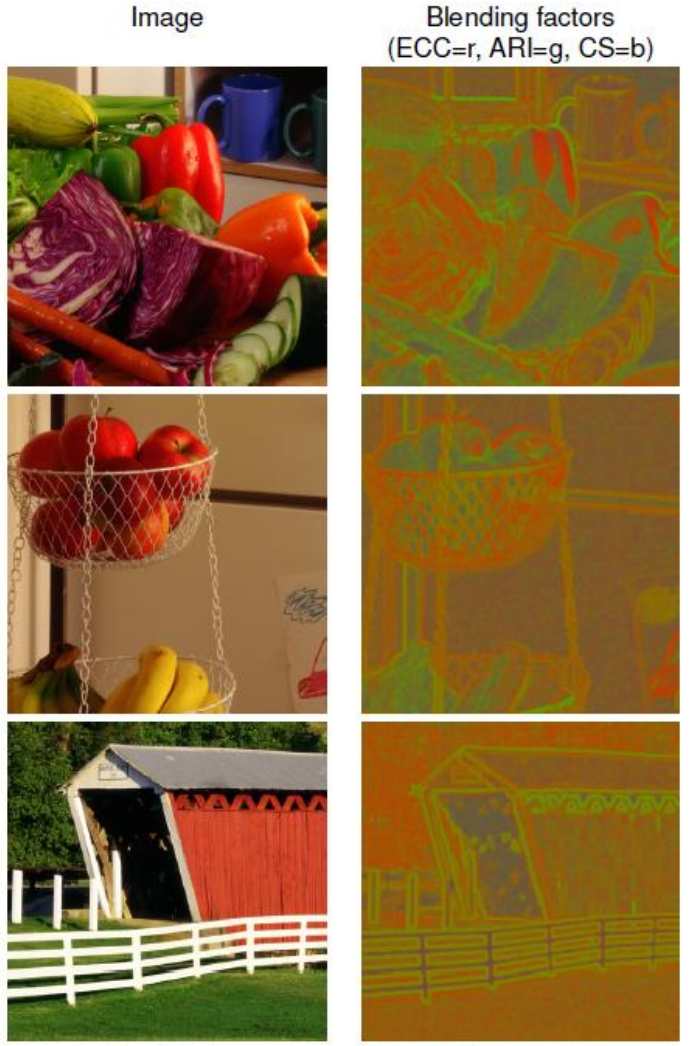


Figure 3: Blending factors learned by $aMIX_{MS\text{-}SSIM}$ (right column) for three images of the McMaster dataset [24]. The blending factor for ECC, $p^0/\sum p^k$, is associated with the red channel, whereas those of ARI ($p^1/\sum p^k$) and CS ($p^2/\sum p^k$) are associated with the green and blue channels, respectively. ECC and ARI are the preferred methods for edges and textured areas, whereas CS has more importance in very dark or bright areas of the images.

[1] S. P. Jaiswal, O. C. Au, V. Jakhetiya, Y. Yuan, and H. Yang, “Exploitation of inter-color correlation for color image demosaicking,” in ICIP, 2014.
[2] Y. Monno, D. Kiku, M. Tanaka, and M. Okutomi, “Adaptive residual interpolation for color image demosaicking,” in ICIP, 2015.
[3] P. Getreuer, “Image demosaicking with contour stencils,” IPOL, 2012.

RESULTS: DEMOSAICING

Numerical evaluation

	ECC	ARI	CS	Mix_{PSNR}	$\text{aMix}_{\text{PSNR}}$	$\text{aMix}_{\text{SSIM}}$	$\text{aMix}_{\text{MS-SSIM}}$
PSNR	38.85	38.37	36.67	39.71	**39.81**	39.80	**39.81**
SSIM	0.9666	0.9633	0.9516	0.9717	**0.9725**	**0.9725**	**0.9725**
MS-SSIM	0.9951	0.9945	0.9913	0.9963	0.9963	0.9963	**0.9964**

CONCLUSION

CONCLUSION

Learnable adaptive filters

- Explainability
- Computational / power effectiveness
- Tunability for different metrics

Possible future directions

- Feature learning
- Reinforcement Learning
- Other applications (beyond image processing)

Learning Adaptive Parameter Tuning for Image Processing

J. Dong, I. Frosio*, J. Kautz

ifrosio@nvidia.com