Edge-Optimized À-Trous Wavelets for Local Contrast Enhancement with Robust Denoising

Johannes Hanika, Holger Dammertz, Hendrik Lensch |
September 17, 2011

Motivation: Edge-Aware Image Processing

- ongoing research:
- look transfer via bilateral filtering (Dae, Paris and Durand 2006)

Motivation: Edge-Aware Image Processing

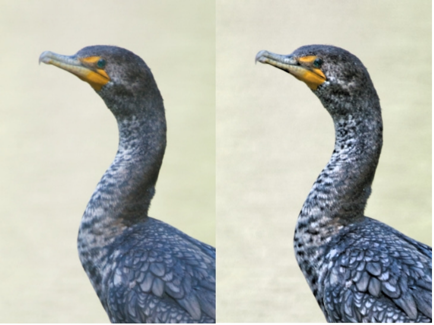
- ongoing research:
- multi scale decomposition by solving a linear system (Farbman et al. 2008)

Motivation: Edge-Aware Image Processing

- ongoing research:
- colorization via edge-avoiding wavelets (Fattal 2009)

Motivation: Edge-Aware Image Processing

- ongoing research:
- local contrast via local histograms (Kass 2010)



Motivation: Edge-Aware Image Processing

- ongoing research:
- via domain transform (Gastal and Oliveira 2011)

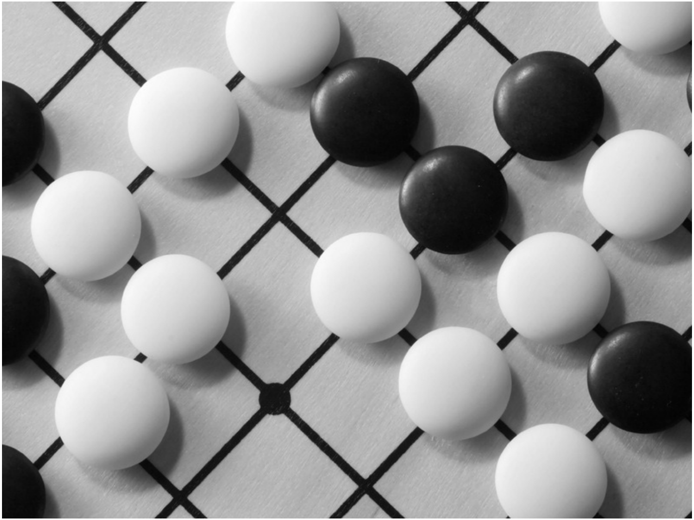
Input Photograph | Edge-aware smoothing | Detail enhancement | Stylization

Previous Work

- all based on multiscale decompositions:
- iteratively applying a bilateral filter
  - lots of techniques to speed it up
  - still high memory footprint and/or low performance
- high quality by solving a linear system
  - not meant to be high performance
- fastest methods based on decimated wavelets (Fattal 2009)

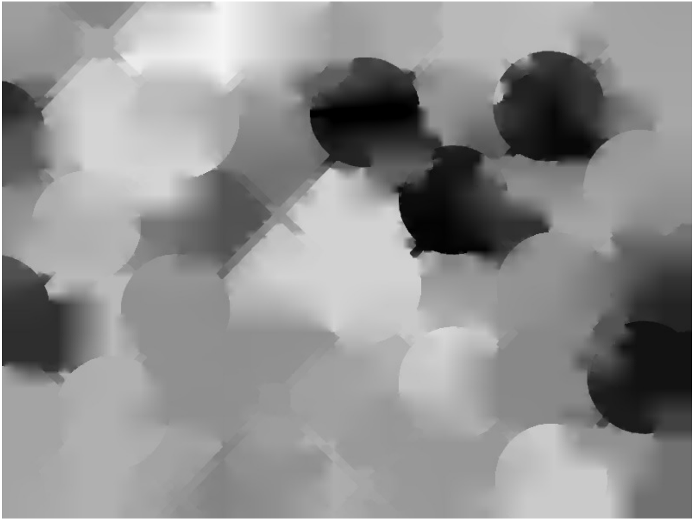
Previous Work

- decimated wavelets fail to capture edges at all scales:



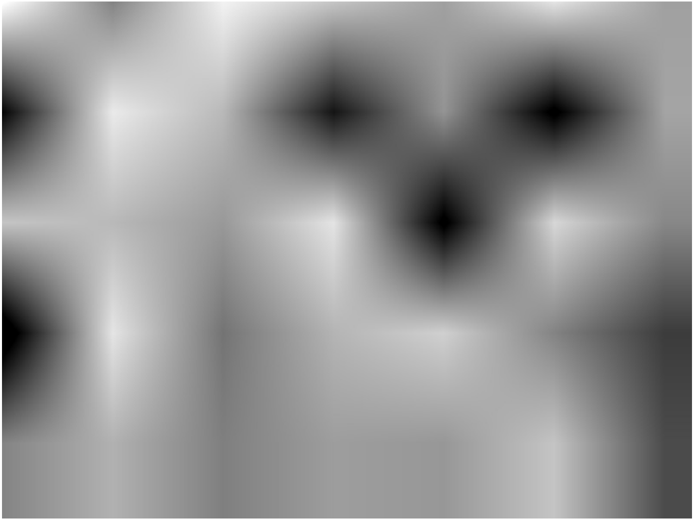
Previous Work

- decimated wavelets fail to capture edges at all scales:



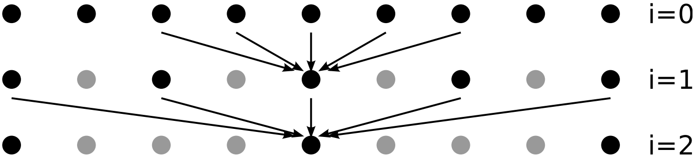
Previous Work

- because coarse coefficients are sparse



Previous Work

- use à-trous wavelet



- results in a full image (not decimated) per step
- $\Rightarrow$ the transformation is *shift invariant*

Previous Work

- à-trous wavelet decomposition
  1. level $i = 0$ starts with the input signal $c_0(p)$

Previous Work

- à-trous wavelet decomposition
  1. level $i = 0$ starts with the input signal $c_0(p)$
  2. compute next base layer (convolution with holes)

$$c_{i+1}(p) = \frac{1}{k} \sum_{q \in \Omega} h_i(q) \cdot c_i(p)$$

Previous Work

- à-trous wavelet decomposition

  1. level $i = 0$ starts with the input signal $c_0(p)$
  2. compute next base layer (convolution with holes)

$$c_{i+1}(p) = \frac{1}{k} \sum_{q \in \Omega} h_i(q) \cdot c_i(p)$$

  3. compute next detail layer (difference)

$$d_i(p) = c_i(p) - c_{i+1}(p)$$

Previous Work

- à-trous wavelet decomposition
  1. level $i = 0$ starts with the input signal $c_0(p)$
  2. compute next base layer (convolution with holes)

$$c_{i+1}(p) = \frac{1}{k} \sum_{q \in \Omega} h_i(q) \cdot c_i(p)$$

  3. compute next detail layer (difference)

$$d_i(p) = c_i(p) - c_{i+1}(p)$$

  4. if $i < N : i := i + 1$; goto 2

Previous Work

- à-trous wavelet decomposition

1. level $i = 0$ starts with the input signal $c_0(p)$
2. compute next base layer (convolution with holes)

$$c_{i+1}(p) = \frac{1}{k} \sum_{q \in \Omega} h_i(q) \cdot c_i(p)$$

3. compute next detail layer (difference)

$$d_i(p) = c_i(p) - c_{i+1}(p)$$

4. if $i < N : i := i + 1$; goto 2
5. $\{d_0, d_1, ..., d_{N-1}, c_N\}$ is the wavelet transform of c.

Previous Work

- à-trous wavelet decomposition edge-aware version

1. level $i = 0$ starts with the input signal $c_0(p)$
2. compute next base layer (convolution with holes)

$$c_{i+1}(p) = \frac{1}{k} \sum_{q \in \Omega} h_i(q) \cdot c_i(p) \cdot w_{\sigma_r}(p,q)$$

3. compute next detail layer (difference)

$$d_i(p) = c_i(p) - c_{i+1}(p)$$

4. if $i < N : i := i + 1$; goto 2
5. $\{d_0, d_1, \ldots, d_{N-1}, c_N\}$ is the wavelet transform of c.

Previous Work

- à-trous wavelet decomposition edge-aware version
  1. level $i = 0$ starts with the input signal $c_0(p)$
  2. compute next base layer (convolution with holes)

$$c_{i+1}(p) = \frac{1}{k} \sum_{q \in \Omega} h_i(q) \cdot c_i(p) \cdot w_{\sigma_r}(p, q)$$

  3. compute next detail layer (difference)

$$d_i(p) = c_i(p) - c_{i+1}(p)$$

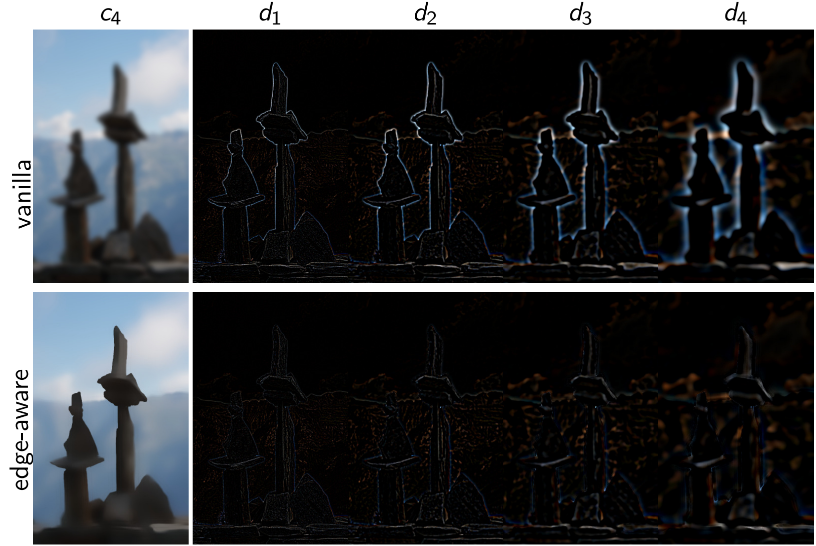
  4. if $i < N$: $i := i + 1$; goto 2
  5. $\{d_0, d_1, ..., d_{N-1}, c_N\}$ is the wavelet transform of c.

- synthesis: simply add up base and detail layers

$$c = c_N + \sum_{i=N-1}^{0} d_i.$$

Decomposition

- example coarse and detail layers



Synthesis for Local Contrast

- add up boosted detail layers

$$c = c_N + \sum_{i=N-1}^{0} \beta_i \cdot d_i.$$

Observation

- how to choose good edge weights $w_{\sigma_r}(p, q)$?

Observation

- how to choose good edge weights $w_{\sigma_r}(p,q)$?

- too strong edge weights: gradient reversals

Observation

- how to choose good edge weights $w_{\sigma_r}(p,q)$?

- too strong edge weights: gradient reversals
- too soft edge weights: halos

Observation

- how to choose good edge weights $w_{\sigma_r}(p, q)$?

- too strong edge weights: gradient reversals
- too soft edge weights: halos
- Kass and Solomon (2010) do explicit diffusion on coarse buffer as post

Decomposition is fast!

- $\Rightarrow$ optimization by synthesis to acquire σ_r per pixel!
- stay in wavelet framework

Edge-Optimized Decomposition

- at each scale, do several decompositions using $\sigma_r^j, \quad j = 0, 1, \ldots$

Edge-Optimized Decomposition

- at each scale, do several decompositions using $\sigma_r^j, \quad j = 0, 1, \ldots$
- compute error measure e_j

$$e_j = d_{i,j}^2 + \lambda \cdot \|\nabla c_{i,j}\|$$

- prefer low energy in details d and smooth base layer c

Edge-Optimized Decomposition

- at each scale, do several decompositions using $\sigma_r^j, \quad j = 0, 1, \ldots$
- compute error measure e_j

$$e_j = d_{i,j}^2 + \lambda \cdot \|\nabla c_{i,j}\|$$

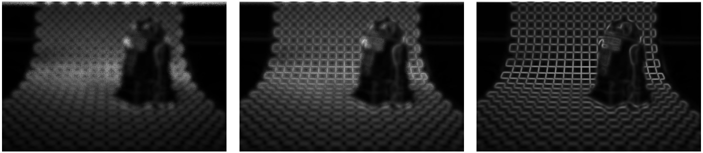
- prefer low energy in details d and smooth base layer c
- choose per-pixel edge weight

$$\sigma_r^k(p) : k = argmin_j \{e_j\}$$

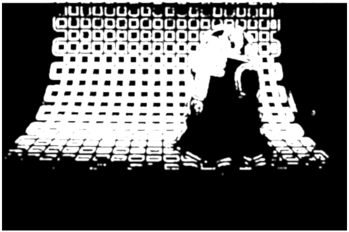
- details how to make noisy estimates of ∇c stable in the paper

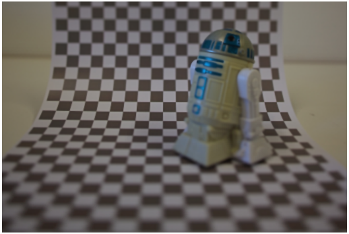
Edge-Optimized Decomposition

- error images e_j



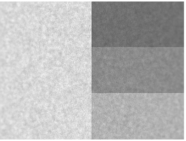
- choice of σ_r and input image





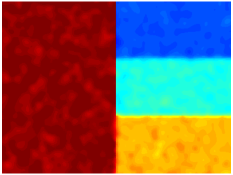
Decomposition Quality

input

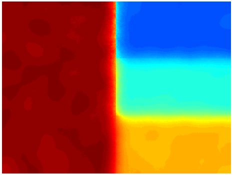


colored output for visualization as (Farbman 08)

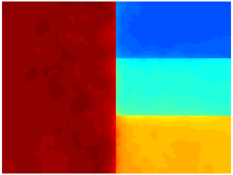
edge-avoiding



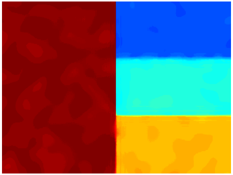
bilateral



(Farbman 08) WLS

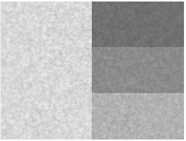


edge-optimized



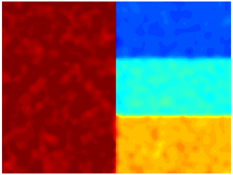
Decomposition Quality

input

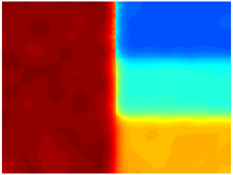


colored output for visualization as (Farbman 08)

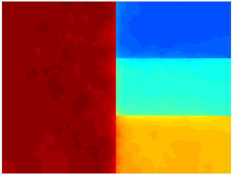
edge-avoiding



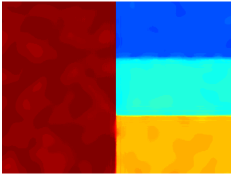
bilateral



(Farbman 08) WLS



edge-optimized



comparable quality, orders of magnitude faster

Synthesis with Denoising

- synthesis after local contrast boost also boosts noise!

Synthesis with Denoising

- synthesis after local contrast boost also boosts noise!
- wavelet framework ⇒ can use robust noise variance estimate and BayesShrink threshold

$$\begin{aligned} d_i' &= \max\{0, |d_i| - T\} \cdot sign(d_i) \\ \text{and } c_{i-1} &= c_i + \beta \cdot d_i' \end{aligned}$$

- details in the paper

Denoising Quality

input 5% noise

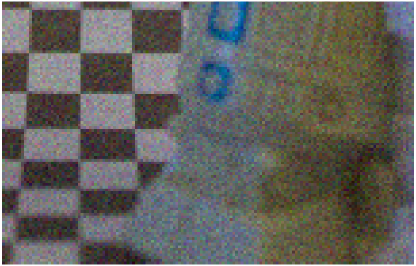
à-trous PSNR 32.5

EAW PSNR 39.1

EOW PSNR 39.8

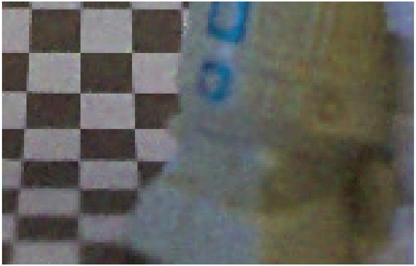
Denoising Quality

input 10% noise



à-trous PSNR 26.3

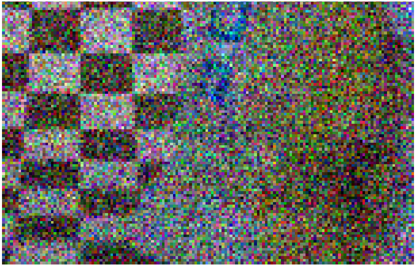
EAW PSNR 34.6



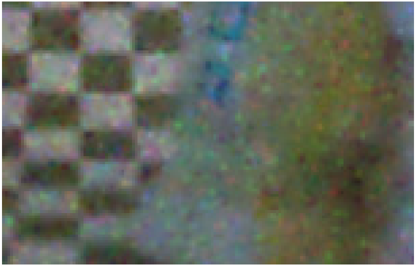
EOW PSNR 35.9

Denoising Quality

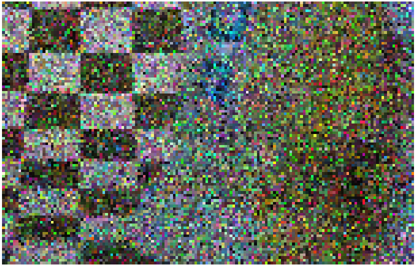
input 40% noise



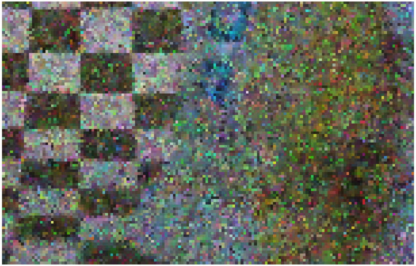
à-trous PSNR 26.5



EAW PSNR 15.0



EOW PSNR 19.6



Performance (CPU)

algorithm		wallclock
EAW (Fattal 09)	(core i7, $\alpha = 1$)	0.088s
EAW (Fattal 09)	(core i7, $\alpha = 0.8$)	0.296s
this paper	(core i7)	0.197s

- 1 megapixel, 3 scales, 4 channels per pixel Lab data
- core i7 CPU : 8 threads on 4 cores
- (Fattal 09) with $\alpha = 1$ removes expensive exponentiation

Performance (GPU)

ms	number of σ_r tested				
	1	2	3	4	5
1 scale	19	23	26	32	39
2 scales	27	35	43	51	63
3 scales	35	48	61	75	87
4 scales	42	61	81	102	120
5 scales	55	80	109	134	163

- edge-optimized wavelet transform on a GTX480 for a one megapixel image
- numbers are in milliseconds

Results (Local Contrast)

- (video)

Limitations

- high contrast, axis aligned changes (in hdr images) can lead to aliasing:


- transparently reduced by our optimization (both via d^2 and smoothness term)
- technique to further ameliorate that in the paper
- not the world's best denoising technique, but helps suppress noise enhancement during local contrast step

Summary

- edge avoiding à-trous wavelets are useful!
- they can be fast (suitable for video processing)
- and achieve high-quality coarse/detail decompositions
  - avoid gradient reversals
  - avoid halos
  - better match the assumptions of BayesShrink denoising
- parameter free, if you want it
- super simple to implement

Thank you for listening!

- some of the code is available at `http://darktable.sf.net` (hardcore SSE optimized + OpenCL)
- thanks to `Edouard Gomez` and `Rostyslav Pidgornyi` for the fast SSE version!