Digital Image

(B4M33DZO, Summer 2024)

Lecture 5:

Non-Linear Filtering

https://cw.fel.cvut.cz/wiki/courses/b4m33dzo/start

Daniel Sýkora & Ondřej Drbohlav

Department of Cybernetics

Faculty of Electrical Engineering

Czech Technical University in Prague

Motivation

noisy original

F

Motivation

noisy original

F

linear smoothing

$F * G$

Motivation

noisy original

F

linear smoothing

$F * G$

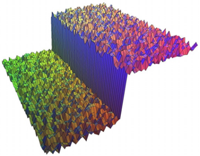
edge preserving

?

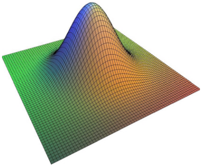
Bilateral Filter

Edge-preserving extension of convolution:

$$(F * G)[s,t] = \sum_{(x,y)\in N(s,t)} F[x,y] \cdot G[s-x, t-y]$$



F

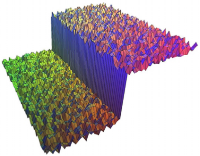


G

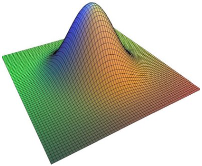
Bilateral Filter

Edge-preserving extension of convolution:

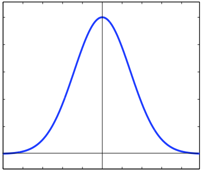
$$(F \star G_b)[s,t] \propto \sum_{(x,y)\in N(s,t)} F[x,y] \cdot G[s-x,t-y] \cdot b(F[x,y]-F[s,t])$$



F



G

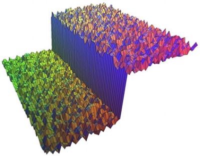


b

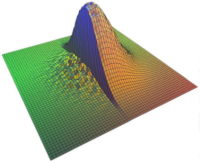
Bilateral Filter

Edge-preserving extension of convolution:

$$(F \star G_b)[s,t] \propto \sum_{(x,y)\in N(s,t)} F[x,y] \cdot \underbrace{G[s-x,t-y] \cdot b(F[x,y]-F[s,t])}$$



F

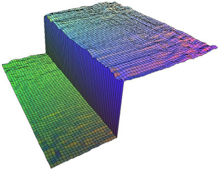


G_b

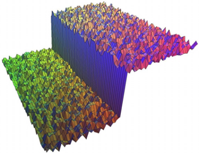
Bilateral Filter

Edge-preserving extension of convolution:

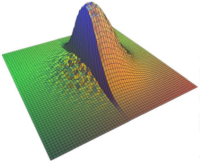
$$(F \star G_b)[s,t] \propto \sum_{(x,y)\in N(s,t)} F[x,y] \cdot \underbrace{G[s-x,t-y] \cdot b(F[x,y]-F[s,t])}$$



$F \star G_b$



F

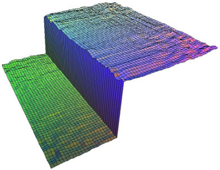


G_b

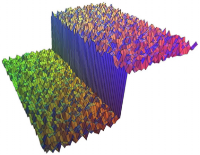
Bilateral Filter

Per-pixel normalization required:

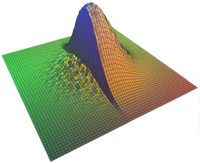
$$(F \star G_b)[s,t] = \frac{1}{W} \sum_{(x,y)\in N(s,t)} F[x,y] \cdot \underbrace{G[s-x,t-y] \cdot b(F[x,y] - F[s,t])}_{W_{(s,t)}[x,y]}$$



$F \star G_b$



F



G_b

$$W = \sum_{(x,y)\in N(s,t)} W_{(s,t)}[x,y] = \sum_{(x,y)\in N(s,t)} G[s-x,t-y] \cdot b(F[x,y] - F[s,t])$$

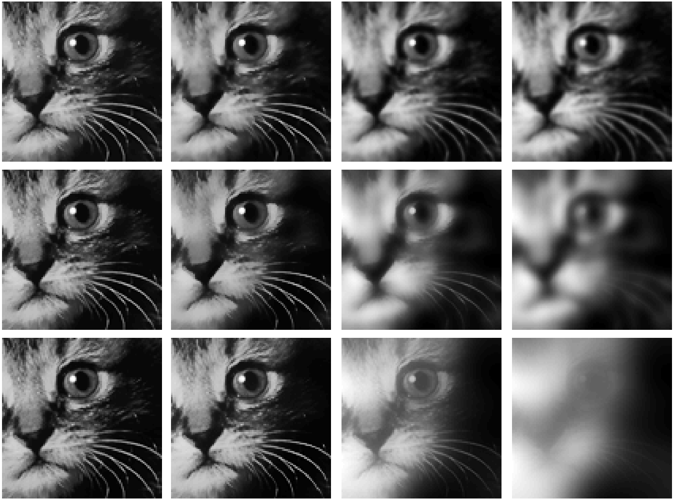
Parameters

$\sigma_b = 10$ $\sigma_b = 30$ $\sigma_b = 100$ $\sigma_b = 300$

$\sigma_G = 1$

$\sigma_G = 3$

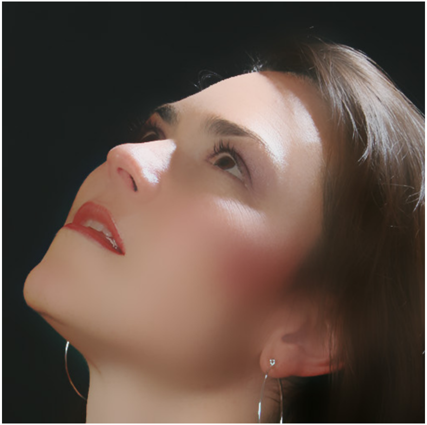
$\sigma_G = 10$



Linear Scale

F

$F \star G_b$



$b(F[x,y] - F[s,t])$

Logarithmic Scale

F

$F \star G_b^{\log}$

$$b(\log F[x,y] - \log F[s,t])$$

Computational Complexity

Bilateral filter:

$$(F \star G_b)[s,t] \propto \sum_{(x,y)\in N(s,t)} F[x,y] \cdot G[s-x,t-y] \cdot b(F[x,y] - F[s,t])$$

Computational Complexity

Bilateral filter:

$$(F \star G_b)[s,t] \propto \sum_{(x,y)\in N(s,t)} F[x,y] \cdot G[s-x,t-y] \cdot b(F[x,y]-F[s,t])$$

Brute force: $\mathcal{O}(|F| \cdot |G|)$

Computational Complexity

Bilateral filter:

$$(F \star G_b)[s,t] \propto \sum_{(x,y)\in N(s,t)} F[x,y] \cdot G[s-x,t-y] \cdot b(F[x,y] - F[s,t])$$

Brute force: $\mathcal{O}(|F| \cdot |G|)$

Problem: expensive for larger $|G|$

How to compute it faster?

Computational Complexity

Bilateral filter:

$$(F \star G_b)[s,t] \propto \sum_{(x,y)\in N(s,t)} F[x,y]\cdot G[s-x,t-y]\cdot b(F[x,y]-F[s,t])$$

Brute force: $\mathcal{O}(|F|\cdot|G|)$

Problem: expensive for larger $|G|$

How to compute it faster?

method	complexity
Piecewise-linear	$\mathcal{O}(\|F\|)$
Bilateral grid	$\mathcal{O}(\|F\|\|b\|)$
Box kernel	$\mathcal{O}(\|F\|\cdot\log\|G\|)$

Fast Approximations

Bilateral filter with box kernel:

$$(F \star G_b)[s,t] \propto \sum_{(x,y)\in N(s,t)} F[x,y] \cdot G[s-x,t-y] \cdot b(F[x,y] - F[s,t])$$

Fast Approximations

Bilateral filter with box kernel:

$$(F \star G_b)[s,t] \propto \sum_{(x,y)\in N(s,t)} F[x,y] \cdot \qquad b(F[x,y] - F[s,t])$$

Fast Approximations

Bilateral filter with box kernel:

$$(F \star G_b)[s,t] \propto \sum_{(x,y)\in N(s,t)} F[x,y] \cdot b(F[x,y] - F[s,t])$$

Fast Approximations

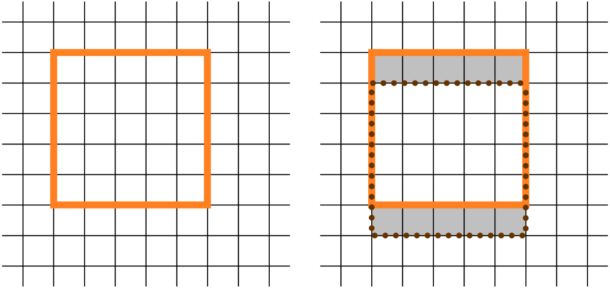
Bilateral filter with box kernel:

$$(F \star G_b)[s,t] \propto \sum_i i H_{s,t}[i] \cdot b(i - F[s,t])$$

Fast Approximations

Bilateral filter with box kernel:

$$(F \star G_b)[s,t] = \sum_i i H_{s,t}[i] \cdot b(i - F[s,t]) / W, \;\; W = \sum_i H_{s,t}[i] \cdot b(i - F[s,t])$$



Fast Approximations

Piecewise-linear bilateral filter:

$$(F \star G_b)[s,t] \propto \sum_{(x,y)\in N(s,t)} F[x,y] \cdot G[s-x,t-y] \cdot b(F[x,y] - F[s,t])$$

Fast Approximations

Piecewise-linear bilateral filter:

$$(F \star G_b)[s,t] \propto \sum_{(x,y)\in N(s,t)} F[x,y]\cdot G[s-x,t-y]\cdot b(F[x,y]-i)$$

Fast Approximations

Piecewise-linear bilateral filter:

$$(F \star G_b)[s,t] \propto \sum_{(x,y)\in N(s,t)} F[x,y] \cdot b(F[x,y]-i) \cdot G[s-x,t-y]$$

Fast Approximations

Piecewise-linear bilateral filter:

$$(B_i * G)[s,t] \propto \sum_{(x,y)\in N(s,t)} B_i[x,y] \cdot G[s-x,t-y]$$

Fast Approximations

Piecewise-linear bilateral filter:

$$\underbrace{(B_i * G)[s,t]}_{L_i[s,t]} \propto \sum_{(x,y)\in N(s,t)} B_i[x,y] \cdot G[s-x,t-y]$$

$$(F \star G_b)[s,t] \approx (i_H - F[s,t]) \cdot L_{i_L}[s,t] + (F[s,t] - i_L) \cdot L_{i_H}[s,t]$$

Fast Approximations

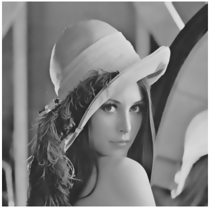
Piecewise-linear bilateral filter:

$b(F[x,y]-50)$

$b(F[x,y]-100)$

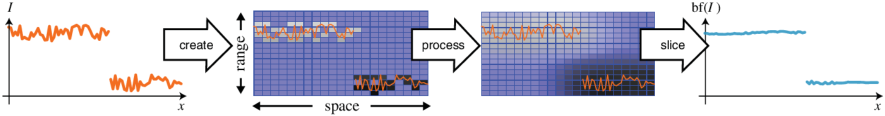
$b(F[x,y]-150)$

$b(F[x,y]-200)$



Fast Approximations

Bilateral grid:



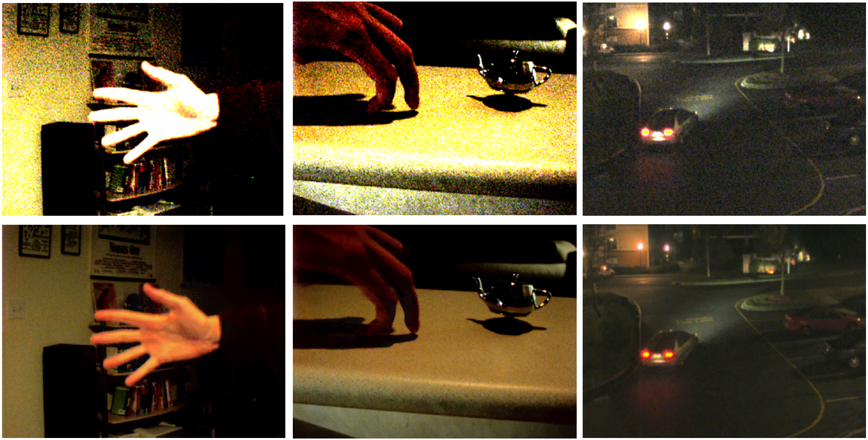
(a) input 1D image (b) grid created from 1D image (c) filtered grid and slicing image (d) filtered 1D image

Noise Suppression

original

bilateral filter

Video: bilateral filtering in time (virtual exposures)



Tone Mapping

How to display HDR image on a device having limited range?

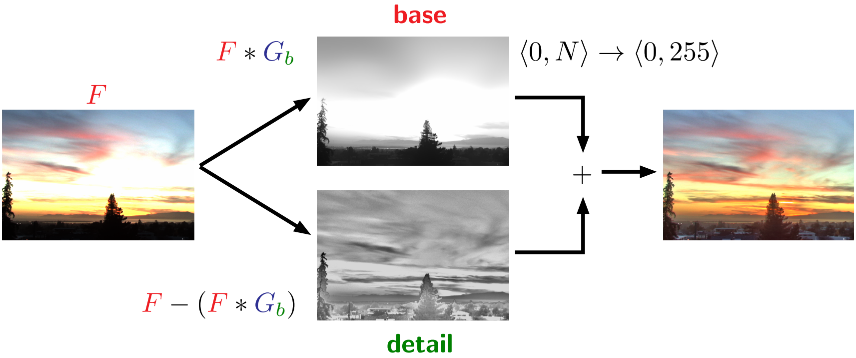
scaling

tone mapping

e.g.: $\langle 0, 65535 \rangle$ **to** $\langle 0, 255 \rangle$

Tone Mapping

Decompose HDR image to base and detail components:



Apply scaling on base image and add details.

Tone Mapping

linear filter

bilateral filter

Edge-preserving bilateral filter avoids unnatural halo effect.

Impulse noise

Median filter